The Book of Postfix

State-of-the-Art Message Transport

Ralf Hildebrandt, Patrick Koetter

NO STARCH
PRESS

HIGH TECH

Postfix

Подробное руководство

Современный транспорт
для сообщений

Ральф Гильдебрандт,
Патрик Кеттер

Санкт-Петербург — Москва
2008

Серия «High tech»

Ральф Гильдебрандт, Патрик Кеттер

Postfix. Подробное руководство

Перевод П. Шера

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*Ф. Торчинский*
Редактор	*Е. Бочкарева*
Художник	*В. Гренда*
Корректор	*С. Минин*
Верстка	*Д. Орлова*

Гильдебрандт Р., Кеттер П.

Postfix. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2008. – 512 с., ил.

ISBN-10: 5-93286-109-6

ISBN-13: 978-5-93286-109-7

Практический подход книги «Postfix. Подробное руководство» будет полезен как специалистам, так и новичкам, предоставив им возможность управлять этим современным открытым почтовым сервером. Независимо от того, используете ли вы Postfix для потребностей маленькой компании, в качестве сервера-ретранслятора или же как корпоративный почтовый сервер, вы научитесь максимально полно использовать возможности этого мощного средства передачи корреспонденции и его ценные средства защиты.

Авторы, весьма уважаемые специалисты по Postfix, собрали в одной книге множество технической информации, документации и ответов на часто задаваемые вопросы. Рассказывается о наиболее распространенных способах применения Postfix и о редко используемых функциях, приводится масса практических примеров, показывающих пути решения таких повседневных задач, как защита пользователей от спама и вирусов, управление несколькими доменами и обеспечение роумингового доступа. Вы узнаете, как осуществлять интеграцию с OpenLDAP, MySQL или PostgreSQL, как ограничивать перемещение корреспонденции на основе черных списков, аутентифицировать пользователей, применять шифрование TLS и автоматизировать каждодневные операции. Руководство необходимо всем, кто заинтересован в использовании и понимании Postfix, начиная от домашнего пользователя и заканчивая администратором крупных почтовых систем.

ISBN–10: 5–93286–109–6

ISBN–13: 978–5–93286–109–7

ISBN 1–59327–001–1 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Подписано в печать 28.03.2008. Формат 70x100 1/16 . Печать офсетная.
Объем 32 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Тем, кто любит
хорошее программное обеспечение

Оглавление

Отзывы на книгу «The Book of Postfix»

Многие технические книги мало чем отличаются от пересказа документации по продукту, в то время как Кеттер и Гильдебрандт проникают в самые сокровенные глубины Postfix. Дав читателям понимание основ, они принимаются за более сложные возможности Postfix. Прочитав книгу, я подумал, что если бы подобные руководства были написаны и для других почтовых программ, технология стала бы более понятной.

– Том Томас (Tom Thomas), автор книги «Network Security First-Step» (CISCO PRESS)

Postfix получает все большее распространение, в нем возникают новые дополнительные возможности, а вместе с этим растет потребность в исчерпывающем руководстве, к которому администраторы могли бы обращаться при развертывании и сопровождении своих Postfix-систем. Патрик Кеттер и Ральф Гильдебрандт – специалисты, посвятившие себя Postfix с самых первых его дней, и их книга как раз отвечает сложившейся потребности.

– Лутц Джанике (Lutz Janicke), создатель патча TLS для Postfix

Лично меня книга Ральфа и Патрика больше всего поразила тем, как они смогли сделать сложные понятия простыми для понимания. Очевидно, что авторы знают свой предмет вдоль и поперек и предлагают его в удобной для восприятия форме. Они ничего не упустили.

– Тобиас Отикер (Tobias Oetiker), автор программ Round Robin Database Tool (RRDTOOL) и Multi Router Traffic Grapher (MRTG)

В этой книге множество практических примеров и понятных объяснений – кажется, будто рядом с вами сидит специалист по Postfix.

– Дэвид Швайкерт (David Schweikert),
автор POSTGREY (Postfix greylisting policy server)

Рекомендую эту книгу всем пользователям Postfix, а особенно тем, кто планирует использовать его для проверки на вирусы AMaViS.

– Рейнер Линк (Rainer Link),
создатель OPENANTIVIRUS.ORG

Книга абсолютно необходима любому, кто заинтересован в использовании и понимании Postfix, начиная от домашнего пользователя и заканчивая администратором самых крупных почтовых систем.

– Доктор Ливиу Дайа (Liviu Daia),
старший исследователь Института
математики Румынской академии

Об авторах

Ральф Гильдебрандт и Патрик Кеттер – активные и хорошо известные в сообществе Postfix фигуры. Гильдебрандт является исполнительным директором немецкой компании T-Systems, поставщика решений в области информационных технологий и связи (ICT). Кеттер занимается информационной архитектурой в собственной компании, которая предоставляет услуги консалтинга и проектирования корпоративных телекоммуникационных систем для клиентов из Европы и Африки. Оба делают доклады о Postfix на бизнес-конференциях и встречах специалистов, а также регулярно участвуют в работе нескольких общедоступных почтовых рассылок.

Благодарности

Нам необходимо поблагодарить за эту книгу огромное количество людей, и далее каждый из нас представит свой список.

Ральф Гильдебрандт

Когда я писал эту книгу, я заметил, что очень мало знаю о том, что находится у Postfix «под капотом». Я знал, как он ведет себя, но не знал точно почему. (По крайней мере, не в каждом отдельном компоненте и не в экзотических случаях.) В некоторых случаях я чего-то не знал, в каких-то областях мои знания (или их отсутствие) оказывались ошибочными. Для получения подробной информации мне пришлось прочесть эту чертову инструкцию и задать множество вопросов в полезных рассылках postfix-users. Эта книга не сможет заменить более чем пятилетний опыт работы с Postfix, но поможет узнать его лучше.

Известно, что в 1994 году, когда я начинал заниматься UNIX, Интернет был гораздо более безопасным местом, чем сейчас. Не было никакого спама! Я познакомился с Postfix лишь потому, что у меня сломался Sendmail. Недолго попользовавшись qmail, я открыл для себя Postfix и остался верен ему. Я никогда не оглядывался назад.

Когда Билл обратился ко мне с предложением написать книгу о Postfix, я сначала сомневался. Мне был необходим соавтор, так как объем предполагаемой работы был слишком велик для одного человека. В то время Патрик занимался тем, что проклинал SASL в рассылке. Он поклялся, что если ему удастся увидеть SASL работающим, он напишет о том, как этого добиться. У Патрика получилось, и он написал руководство. Я прочитал его, оно мне понравилось, и я пригласил Патрика в соавторы.

Как оказалось, объем работы был слишком велик и для двоих, так что к нам в качестве технического редактора присоединился Брайан Уорд (Brian Ward), чей опыт оказался очень ценным в недостаточно изведанных нами областях.

Если бы не помощь Витсе Венема (Wietse Venema), Виктора Духовны (Vi(c|k)tor Duchovni), Лутца Джанике (Lutz Janicke), Андреаса Винкельмана (Andreas Winkelmann) и Питера Берингера (Peter Bieringer), эта книга никогда не стала бы такой, какая она есть, поэтому они по-

лучат по экземпляру в подарок. Не то чтобы книга была им необходима, но она, несомненно, станет замечательным подарком. Огромная благодарность и вся моя любовь моей жене Констанце, которая выдерживала мои постоянные отговорки «Но мне еще нужно написать главу!», благодаря чему я смог завершить книгу и не дал ей превратиться в химеру. Да, когда будете читать замечания Патрика, пожалуйста, имейте в виду, что я лишь слегка ненормален.

Патрик Кеттер

Пройдут годы, прежде чем Интернет предоставит нам все необходимые услуги. Как и в случае с любой другой новой средой, первым побуждением поставщиков услуг является стимуляция роста, особенно в части увеличения количества контента и услуг. Качество обслуживания и его функциональные возможности обычно остаются на втором плане, по крайней мере, до тех пор, пока обслуживание не начнет окупаться. Пока же он беззащитен перед теми, кто предпочитает злоупотребление и разрушение движению и развитию.

Так обстояли дела с электронной почтой, когда появился Postfix, предлагающий новый уровень качества обслуживания.

Когда мне потребовался собственный SMTP-сервер, я был возмущен тем, что работа с Sendmail, похоже, требует наличия какого-нибудь диплома, особенно если надо разобраться с макросами. Тогда я решил поискать другое программное обеспечение. Поиск не был долгим – я влюбился в Postfix.

Postfix показал мне, что сложное программное обеспечение может быть настроено с помощью простого и понятного структурированного синтаксиса. Если вы знакомы с SMTP, то вы уже знаете большую часть важных элементов настройки Postfix. Когда Ральф предложил мне писать книгу вместе с ним, я по-настоящему не знал SMTP. Работа над книгой заставила меня изучить намного больше, чем я предполагал, и помогла опровергнуть ряд заблуждений.

Я очень горжусь тем, что эта книга предоставила мне возможность поделиться своими знаниями о сегодняшних компьютерах и электронной почте. Надеюсь, она укажет вам путь к творческому использованию Postfix. А лучшей основой для творчества являются знания.

Эта книга не появилась бы на свет, если бы не знания, любознательность и поддержка Витсе Венема (Wietse Venema), Виктора Духовны (Vi(c|k)tor Duchovni), Ливиу Дайа (Liviu Daia), Лутца Джанике (Lutz Janicke), Флориана Кирштейна (Florian Kirstein), Уолтера Стейнсдорфера (Walter Steinsdorfer), Роланда Роллингера (Roland Rollinger), Тома Томаса (Tom Thomas), Алексея Мельникова (Alexey Melnikov), Андреаса Винкельманна (Andreas Winkelmann), Эрика «cybertime hostmaster», а также подписчиков рассылки Postfix, чьи вопросы и проблемы показали нам, чего не хватает, когда уже казалось, что все сказано.

Что самое важное, я должен поблагодарить Ральфа, чьи знания о Postfix может превзойти лишь его же мастерство использования компьютеров. Здесь он чувствует себя как рыба в воде. Это Ральф выбрал меня своим спутником в приключении под названием «Руководство по Postfix», и я очень признателен этому ненормальному парню, который стал моим близким другом, пока мы писали книгу.

Эта книга стала большим испытанием не только для меня, но и для моей жены Биргит; ее вера в меня служила мне поддержкой на протяжении этих бесконечных строк. Когда вас приглашают сделать что-то, к чему у вас лежит душа, – это большая честь. А если, когда вы наконец сделаете это, рядом с вами есть такой человек, как Биргит, – это просто дар богов.

Предисловие

Использовать слова для описания волшебства –
это все равно, что пытаться разрезать ростбиф отверткой.

– Том Роббинс

Эта книга представляет собой пошаговое руководство по Postfix. Вы начинаете читать ее, будучи новичком, а перевернув последнюю страницу, становитесь (надеемся) специалистом. Каждая глава относится к одному из трех видов: учебное пособие, теория и практика. Учебное пособие – это «букварь», который поможет вам понять суть проблемы, прежде чем пытаться реализовать ее решение в Postfix. Теоретические главы расскажут о том, как Postfix поступает в такой ситуации. Практические главы покажут, как перейти от теории к работающей системе.

Книга состоит из четырех частей, которые делят обучение работе с Postfix на следующие этапы:

Основы

Часть I представляет основы Postfix. Вы научитесь конфигурировать Postfix для сервера с коммутируемым соединением для одного домена. Кроме того, вы кратко ознакомитесь с анатомией Postfix и узнаете, какие инструменты он предлагает.

Контроль содержимого

Postfix обеспечивает широкие возможности управления потоком сообщений в вашей системе. Часть II начинается с рассказа о том, как работает протокол SMTP и каков формат сообщений электронной почты. Далее вы узнаете, как Postfix управляет различными аспектами обработки сообщений.

Сложные конфигурации

Postfix часто взаимодействует с другими приложениями сторонних компаний, такими как SQL-серверы, Cyrus SASL, OpenSSL OpenLDAP. В части III будет описано, как это делается.

Настройка Postfix

Конфигурируемое программное обеспечение всегда оставляет возможность для настройки. Часть IV поможет найти узкие места ва-

шей почтовой системы и даст советы, как повысить ее производительность.

Дополнительные ресурсы

В дополнение к данной книге и документации, поставляемой вместе с Postfix, существуют еще два ресурса, к которым можно в случае необходимости обращаться за информацией или помощью.

Документация по Postfix, практические советы и часто задаваемые вопросы

На сайте Postfix есть страница (*http://www.postfix.org/docs.html*), содержащая документацию по Postfix, практические советы (how-to) и ответы на часто задаваемые вопросы (FAQ), предложенные Postfix-сообществом.

Списки рассылки

Витсе Венема (Wietse Venema) ведет несколько списков рассылки, которые обслуживают Postfix-сообщество. На странице сайта Postfix *http://www.postfix.org/lists.html* вы найдете информацию о подписке на следующие рассылки:

`postfix-announce@postfix.org`

Список рассылки для объявлений о выходе новых редакций и версий Postfix.

`postfix-users@postfix.org`

Обмен мнениями по поводу работы с почтовой системой Postfix. Список не модерируется и требует регистрации.

`postfix-users-digest@postfix.org`

Ежедневная рассылка статей, опубликованных в списке рассылки *postfix-users*.

`postfix-devel@postfix.org`

Малопосещаемый список для людей, заинтересованных в разработке для Postfix.

Сообщество Postfix обсуждает идеи, проблемы, ошибки, патчи и многие другие вопросы в списке рассылки *postfix-users@postfix.org*. Если у вас возникнет какая-то проблема или вам захочется получить информацию о чем-то, связанном с Postfix, велики шансы, что вы найдете то, что ищете, в архивах списка рассылки. Несколько людей и компаний поддерживают архивы *postfix-users@postfix.org*, к которым можно получить доступ через броузер. Подробный перечень архивов приведен на странице списков рассылки сайта Postfix.

Обозначения в книге

Курсив

Используется для выделения названий списков рассылки, URL и адресов электронной почты.

`Моноширинный шрифт`

Используется для выделения имен доменов, демонов, файлов, каталогов, команд, имен параметров и переменных, переменных окружения, параметров командной строки.

`Моноширинный курсив`

Используется для выделения параметров и заполнителей, которые должны быть заменены соответствующими значениями в вашей системе, а также в комментариях примеров команд и кода.

`Моноширинный полужирный`

Используется для выделения команд и параметров, вводимых в окне оболочки.

`Моноширинный полужирный курсив`

Используется для выделения отдельных строк, упоминающихся в обсуждении.

Примечание

Символ `$` обозначает обычное приглашение на ввод в командной строке, символ `#` – это приглашение на ввод в оболочке для привилегированного пользователя.

Домены и имена, используемые в книге

Книга рассказывает о почтовых услугах, поэтому мы будем очень много говорить о доставке и передаче сообщений, и для примеров нам понадобятся имена доменов, отправителей и получателей. Обычно мы будем использовать следующие имена.

Локальный домен

На протяжении всей книги мы будем считать нашим собственным домен `example.com`. Почтовый сервер предположительно будет принимать (или, по крайней мере, рассматривать) сообщения для локальных пользователей `anyuser@example.com` и `anyuser@mail.example.com`. Если вы будете использовать примеры для создания собственного сервера Postfix, необходимо будет заменить `example.com` именем вашего домена.

Примечание

Конечно же, на самом деле `example.com`, `example.org` и `example.net` нам не принадлежат. IANA (Internet Assigned Numbers Authority – уполномоченная организация по распределению нумерации в сети Интернет) зарезервировала их для использования в документации.

Наш провайдер

В качестве имени домена нашего интернет-провайдера будет фигурировать `example-isp.com`.

Сценарии

Вспомогательные сценарии и другую полезную информацию, например список опечаток, вы можете найти по адресу *http://www.postfix-book.com*.

Комментарии

Если вы обнаружите в книге ошибку или захотите отправить какой-то комментарий, используйте адрес *comments@postfix-book.com*.

1

Введение в Postfix

Postfix – это агент передачи сообщений (MTA, message transport agent), который занимается пересылкой по протоколу SMTP сообщений от пользовательского почтового агента (MUA, mail user agent), называемого также почтовым клиентом, к удаленному почтовому серверу. MTA также принимает сообщения от удаленных почтовых серверов и пересылает их другим MTA или доставляет в локальные почтовые ящики. Переслав или доставив сообщение, Postfix заканчивает свою работу. За доставку сообщения конечному пользователю отвечают другие серверы. Например, такие MTA, как серверы POP3 или IMAP[1], передают сообщения почтовым клиентам – Mutt, Outlook или Apple Mail, с помощью которых пользователь может прочитать их.

На первый взгляд работа MTA кажется совсем простой, но это не так. Особенность агентов передачи сообщений заключается в том, что им приходится пересылать информацию через границы сетей – они передают данные в другие сети и принимают данные в своей сети. Здравый смысл подсказывает, что каждый, кто использует сеть, должен принимать необходимые меры предосторожности и защищать свои серверы и данные от возможных атак. Широко распространенное мнение о том, что для этого достаточно установить межсетевой экран, контролирующий соединения между локальной и удаленными сетями в обоих направлениях, не более чем миф: межсетевой экран – это не программа, а концепция.

Самая известная часть типичной реализации межсетевого экрана – это приложение, которое контролирует и ограничивает соединения. К сожалению, межсетевые экраны, как правило, не способны анализировать содержимое сообщений, которыми обмениваются хосты; они мо-

[1] Серверы POP3 и IMAP принято относить к так называемым MAA (Mail Access Agents). – *Примеч. науч. ред.*

гут контролировать сами хосты, порты и протоколы транспортного уровня, используемые при передаче данных, но не могут ограничивать соединения, основываясь на их содержимом. Анализ передаваемых данных – намного более сложная задача, требующая применения специализированных программ, способных определить, является ли содержимое вредоносным, и решить, что с ним делать. Для электронной почты эту задачу решают MTA. Кроме того, современные MTA должны быть быстрыми, надежными и безопасными, т. к. они отвечают за передачу электронной почты – наиболее распространенных в Интернете данных.

Выбор MTA весьма широк, но большинство из них имеет те или иные ограничения. Например, в одном из них реализована превосходная модель безопасности, но команда основных разработчиков более не поддерживает проект, а это чревато потенциальными сбоями в работе и проблемами с безопасностью. Другой MTA получил повсеместное распространение вследствие того, что входит в состав популярного пакета ПО для рабочих групп, но, как выяснилось, его разработчики уделили слишком много внимания возможностям групповой работы в ущерб поддержке сетевых стандартов и противодействию спамерам и злонамеренным взломщикам. Наконец, есть MTA, прекрасно поддерживающий стандарты и легко справляющийся с одновременным обслуживанием множества пользователей, но настолько небезопасный, что для его эксплуатации вам потребуется эксперт, способный противодействовать взлому в периоды между обновлениями ПО.

Для использования Postfix вам не надо быть экспертом; установленный из дистрибутива без дополнительных настроек («из коробки»), он обеспечивает максимально возможный уровень безопасности. Безопасность в Postfix основана на его настройках по умолчанию. Если приложение в базовой конфигурации безопасно и достаточно функционально, чтобы не требовать дополнительного вмешательства, то получить на его основе безопасный MTA не составит труда. Что еще лучше, при необходимости что-либо изменить в базовой конфигурации простой структурированный синтаксис описания параметров Postfix позволяет с легкостью модифицировать поведение, установленное по умолчанию. Кроме того, Postfix имеет модульную структуру, в которой каждый отдельный модуль выполняется с наименьшим уровнем привилегий, позволяющим ему работать. Postfix проектировался с таким расчетом, чтобы начальный уровень безопасности MTA был выше, чем у собственно кода.

Превосходная работа Postfix достигается благодаря тому, что он сосредоточивается на основной задаче – передаче почты, а не изобретает велосипед, повторяя функции, уже реализованные в системе другими программами. Postfix позволяет подключать внешние программы в тех случаях, когда требуемые действия выходят за пределы области передачи сообщений. К тому же Postfix в своей работе полностью задейст-

вует мощь UNIX. Такая тесная интеграция с операционной системой не только облегчает взаимодействие с внешними программами, но и повышает производительность.

С точки зрения современного взгляда на задачу передачи и обработки сообщений Postfix представляет собой самую сердцевину комплекса почтового ПО. На рис. 1.1 Postfix находится в окружении специализированных приложений и инструментов, помогающих управлять данными, соединениями и процессом передачи.

Из этой книги вы узнаете, как настроить Postfix для работы в небольшой сети в качестве сервера-ретранслятора, антивирусного фильтра, а также в качестве корпоративного почтового сервера, интегрированного в современную IT-архитектуру. Продвигаясь от главы к главе, вы познакомитесь с теорией и методиками, выходящими далеко за пределы онлайновых руководств, которые помогут вам наиболее полно использовать возможности этого замечательного продукта.

Рис. 1.1. *Postfix в самом сердце комплекса почтовых программ*

I

Основы

Первая часть этой книги посвящена основам Postfix, и начнем наше знакомство с продуктом с требований к операционной системе и настройки почтового сервера для одного домена. В этих главах мы рассмотрим синтаксис файла конфигурации Postfix и некоторые его программные компоненты и утилиты.

Предлагаем вам краткий обзор четырех глав этой части книги:

Подготовка хоста и окружения

Перед установкой Postfix необходимо проверить, может ли SMTP-сервер работать в системе. В главе 2 рассказывается о том, как сконфигурировать операционную систему, чтобы извлечь максимум из возможностей Postfix.

Почтовый сервер для одного домена

Первым этапом при любой новой установке Postfix является создание конфигурации, в которой возможно получение почты для одного домена. В главе 3 вы узнаете, как проверить работоспособность системы и как создать основу для более сложных конфигураций.

Почтовый сервер с коммутируемым соединением для одного домена

Получить работающую конфигурацию для одного домена можно путем небольших, но важных изменений, которые приводятся в главе 4.

Анатомия Postfix

Витсе Венема (Wietse Venema) говорит, что «Postfix – это фактически маршрутизатор», который выбирает пути для сообщений, а не для IP-пакетов. В главе 5 работа Postfix подробно рассматривается «изнутри».

2

Подготовка хоста и окружения

Вначале не было ничего. Бог сказал: «Да будет свет!».
После этого тоже ничего не было, но это стало видно.

– Игнасио Шварц (Ignacio Schwartz)

Вы, наверное, сгораете от нетерпения начать работу с Postfix. Но прежде нужно кое-что узнать.

Витсе Венема, разработчик Postfix, прекрасно знает UNIX, поэтому он не стал дублировать в Postfix функциональность, которая имеется в UNIX по умолчанию. В силу этого Postfix рассчитывает, что ваша система правильно настроена – его работоспособность напрямую определяется работоспособностью операционной системы.

Не пропускайте эту главу только из-за того, что она кажется вам «детской». Найдите время и убедитесь в том, что ваша система соответствует перечисленным ниже требованиям. Ваши усилия не пропадут даром: Postfix отблагодарит вас быстрой, надежной и безопасной службой.

Вот контрольный список для Postfix:

- Правильно укажите имя хоста
- Проверьте его способность устанавливать соединения
- Поддерживайте правильное системное время
- Убедитесь, что диагностические сообщения Postfix записываются в системный журнал
- Настройте разрешение имен[1] для клиента
- Создайте в DNS записи для почтового сервера

[1] На практике это означает: настройте */etc/resolv.conf* так, чтобы ваш компьютер обращался к исправно работающему серверу имен.– *Примеч. науч. ред.*

Имя хоста

Для надежного взаимодействия с другими системами почтовый сервер должен иметь полностью определенное доменное имя (FQDN – fully qualified domain name; см. RFC 821, *ftp://ftp.rfc-editor.org/in-notes/rfc821.txt*), например *mail.example.com*. В своем приветствии удаленным почтовым клиентам и серверам Postfix автоматически подставляет это данное вами имя, если только вы вручную не задали ему другое.

Полностью определенное доменное имя важно еще и потому, что функции Postfix не ограничиваются приемом почты от клиентов – в клиентском режиме Postfix также передает сообщения другим почтовым серверам. Многие почтовые серверы проверяют заявленное клиентом имя и не принимают сообщения, если имя не является полностью определенным, а некоторые еще и проверяют, разрешается ли указанное имя в DNS.

Операционная система устанавливает имя хоста во время загрузки. Чтобы проверить, есть ли уже у вашей системы полностью определенное имя, зарегистрируйтесь и введите команду `hostname`:

```
$ hostname -f
mail.example.com
```

Если эта команда не вернула вам полностью определенное доменное имя, найдите, где оно устанавливается в вашей системе, и исправьте его. Если же такое имя у вашей системы есть, но вы хотели бы использовать в Postfix другое, оставьте настройку системы как есть. Значение по умолчанию вы замените при помощи параметра `myhostname`.

Примечание

Параметр `-f` команды `hostname` не работает в Solaris, реализации GNU и некоторых других окружениях. Если ваша команда `hostname` не работает, как здесь описано, попробуйте опустить параметр `-f`. Если и эта попытка окажется безуспешной, обратитесь к документации.

Возможность соединения

Убедитесь, что ваша машина может выйти в сеть и что другие хосты этой сети могут общаться с ней. Первая часть не вызывает затруднений: достаточно открыть онлайновую веб-страницу, чтобы убедиться, что ваш компьютер в сети. Входящие соединения проверить немного сложнее. Для этого вам понадобится другой хост в сети, с которого клиенты могли бы устанавливать соединение. Если Postfix предлагает услуги всему Интернету, вам следует проверить возможность соединения с хоста, полностью независимого от вашего сервера.

Порт 25 TCP

Убедитесь, что порт 25 TCP на вашем сервере ничем не заблокирован. Если у вас установлен межсетевой экран, убедитесь, что его политика разрешает входящие и исходящие соединения с портом 25. Имейте в виду, что некоторые интернет-провайдеры блокируют на своих маршрутизаторах исходящие соединения с портом 25, если вы явно не попросите снять это ограничение. Провайдер может отказаться отменить данный запрет и предложит вам свои почтовые серверы в качестве ретрансляторов, например с использованием SMTP-аутентификации, описанной в главе 16.

Причина, по которой порт 25 TCP должен быть открыт, заключается в том, что Postfix и другие почтовые серверы прослушивают его в ожидании соединений. Этот порт официально назначен для SMTP агентством IANA (полный список доступен по адресу *http://www.iana.org/assignments/port-numbers*). Организация IANA является главным регистратором нумерации в интернет-протоколах, распределяющим номера портов, протоколов, компаний, параметров, кодов и типов.

Системное время и временные метки

Правильная установка системного времени особенно важна, когда вы занимаетесь тонкой настройкой или устраняете проблемы. Если возникает необходимость выйти за пределы своей системы и решать возникшие проблемы совместно с администраторами других узлов, точные временные метки могут стать связующим звеном между событиями на вашем почтовом сервере и на тех серверах, которые вы не контролируете.

Postfix тщательно фиксирует свои действия в заголовках сообщений. Взгляните, к примеру, на такой заголовок:

```
Received: from mail.example.net (mail.example.net [192.0.34.166])
        by mail.example.com (Postfix) with ESMTP id 6ED90E1C65
        for <recipient@example.com>; Sat, 7 Feb 2004 10:40:55 +0100 (CET)
Reply-To: sender@example.net
From: Sender <sender@example.net>
To: Recipient <recipient@example.com>
Subject: Keep correct system time
Date: Sat, 7 Feb 2004 10:42:01 +0100
```

Кроме того, Postfix указывает дату в записях почтового журнала. Вот пример записей в журнале:

```
Feb 7 2004 10:40:55 mail postfix/pickup[32610]: 6ED90E1C65: uid=501
                                                  from=<sender>
Feb 7 2004 10:40:55 mail postfix/cleanup[398]: 6ED90E1C65:
    message-id=<20040416020209.7D62343F30@mail.example.com>
```

Поэтому вы должны убедиться, что время у вас установлено с максимальной точностью. Не доверяйте встроенному системному таймеру; дело не только в том, что «плывет» хранимое в ядре UNIX время, но и в том, что производители системных плат используют в энергонезависимом таймере дешевые микросхемы, также подверженные дрейфу с течением времени. Не надо надеяться, что полученное из локального источника время будет синхронно со временем на других серверах.

Есть два способа получения точного времени. Можно использовать протокол NTP (Network Time Protocol – протокол сетевого времени) для получения времени из сети, а можно воспользоваться приемником сигнала GPS (повсеместно) или DCF-77 (на большей части Европы). Если же эти возможности вам недоступны[1], в качестве последнего средства можно использовать программу `clockspeed` (*http://cr.yp.to/clockspeed.html*), позволяющую ввести коррекцию систематической погрешности (как положительной, так и отрицательной) системного таймера. По данным нескольких измерений времени с использованием надежного источника программа вычисляет и компенсирует отклонение таймера.

Примечание

Для обращения к серверу NTP в вашей системе должен быть установлен клиент NTP (такой клиент входит в состав практически всех операционных систем). Для работы по протоколу NTP на межсетевом экране должен быть открыт порт 123 для входящих и исходящих пакетов UDP (User Datagram Protocol – протокол дейтаграмм пользователя). Если вы не знаете, как настроить свой клиент NTP, обратитесь к сайту (*http://www.ntp.org*).

Системный журнал

Одно из основных мест, где следует искать диагностические сообщения, – это почтовый журнал. В Postfix используется стандартная для UNIX утилита регистрации `syslogd`. Для ее настройки, как правило, используется файл `/etc/syslog.conf`. Вот пример конфигурации:

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none -/var/log/messages
# The authpriv file has restricted access.
authpriv.* -/var/log/secure
# Log all the mail messages in one place.
mail.*      -/var/log/maillog
```

[1] Если внешний сервер NTP недоступен, лучше всего установить и настроить его в своей сети; современная сеть без точного времени, получаемого с сервера NTP, столь же немыслима, как паровоз без тендера или квартира без канализации. – *Примеч. науч. ред.*

```
# Log cron stuff
cron.*     -/var/log/cron
# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg    *
# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit    -/var/log/spooler
# Save boot messages also to boot.log
local7.*          /var/log/boot.log
```

Прежде всего обратите внимание на первую запись, содержащую элемент `mail.none`, предотвращающий попадание сообщений почтовой системы в `/var/log/messages`. Это важно, если вы не хотите, чтобы журнал системных сообщений засорялся записями о работе почты. Как видите, для почтового журнала имеется собственная запись с указанием на файл `/var/log/maillog`. Дефис перед именем файла сообщает, что утилита `syslogd` должна записывать сообщения асинхронно, а не обращаться к диску всякий раз, когда надо внести в журнал новую запись.

К сожалению, есть несколько типичных проблем, которые могут возникнуть при работе с `syslogd`. Если вы не видите в журнале ни одной записи, прежде всего надо проверить, действительно ли запущен демон `syslogd`. В следующем примере показано, как проверить это с помощью команды `ps`.

```
# ps auxwww | grep syslog
root    15540  0.0  0.0  1444  524 ?        S    May21  18:20 syslogd -m 0 ❶
root    22616  0.0  0.0  1444  452 pts/0    R    18:09   0:00 grep syslog
```

❶ В первой строке сообщается, что `syslogd` работает с 21 мая.

Кроме того, прежде чем давать указание `syslogd` использовать журнальные файлы, убедитесь, что они существуют и доступны для записи. В некоторых реализациях `syslogd` не предусмотрено автоматическое создание журнальных файлов и сообщений об ошибках при наличии проблем с ними. Этим славится `syslogd` в Solaris.

Очень распространенная ошибка – использование пробелов вместо символов табуляции для отделения типа журнала от имени файла в файле конфигурации `/etc/syslog.conf`. Запись в `syslog.conf` должна выглядеть так:

```
mail.*<TAB>-/var/log/maillog
```

Еще одна проблема с файлом `syslogd.conf` связана с расположением журнала на другом хосте. Будьте осторожны с записями, подобными этой:

```
mail.* @loghost
```

В данном случае `syslogd` посылает все сведения на хост `loghost`, журнал которого вам и следует изучать вместо журнала почтового сервера. Убедитесь в том, что такой хост действительно существует. Это весьма

распространенная ошибка, когда из-за опечатки в `syslogd.conf` все журнальные записи отправляются не на ту машину (или вообще в никуда).

Разрешение имен (DNS)

Прежде чем Postfix, как и любой почтовый сервер, сможет передать сообщение удаленному адресату, он должен определить его местоположение. В Интернете поиск удаленных ресурсов осуществляется с помощью службы доменных имен (DNS). Сервер имен возвращает IP-адрес, соответствующий доменному имени, и наоборот, доменное имя, соответствующее указанному IP-адресу.

Хорошо работающая служба DNS критически важна для производительности MTA. Чем скорее Postfix сможет получить искомый IP-адрес, тем раньше он свяжется с удаленным почтовым сервером и начнет передачу сообщения.

Примечание

Низкая производительность службы имен может стать главным тормозом в больших почтовых системах. В случае перебоев в работе сервера DNS может помочь кэширующий сервер. Устанавливайте кэширующий сервер имен в больших почтовых системах. Имейте в виду, что применение мер против спама может потребовать увеличения количества запросов почтового сервера к DNS в несколько раз.

Прежде чем пытаться увеличить производительность разрешения имен в своей системе, убедитесь, что оно корректно выполняется в вашей операционной системе, запросив у сервера имен запись MX (см. раздел «MX-записи» ниже в этой главе) для `postfix-book.com`. Попробуйте выполнить такую команду:

```
$ dig postfix-book.com MX
```

Результат должен выглядеть так:

```
; <<>> DiG 9.2.2-P3 <<>> postfix-book.com MX
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23929
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2
;; QUESTION SECTION:
;postfix-book.com.              IN      MX
;; ANSWER SECTION:
postfix-book.com.       86400   IN      MX      10 mail.postfix-book.com. ❶
;; AUTHORITY SECTION:
postfix-book.com.       86400   IN      NS      ns3.ray.net. ❷
postfix-book.com.       86400   IN      NS      ns.state-of-mind.de.
;; ADDITIONAL SECTION:
mail.postfix-book.com.  86400   IN      A       212.14.92.89
ns.state-of-mind.de.    81566   IN      A       212.14.92.88
;; Query time: 58 msec
```

```
;; SERVER: 212.18.0.5#53(212.18.0.5)
;; WHEN: Sat Apr 17 03:56:47 2004
;; MSG SIZE  rcvd: 145
```

❶ Эта строка показывает, что `mail.postfix-book.com` является почтовым сервером, принимающим почту для адресатов, находящихся в домене `postfix-book.com`.

❷ Эти две строки показывают, что `ns3.ray.net` и `ns.state-of-mind.de` являются официальными серверами зоны `postfix-book.com`.

Примечание

На некоторых устаревших платформах утилита `dig` не входит в стандартную поставку. Вы можете найти ее в составе дистрибутива BIND на сайте консорциума ISC (http://www.isc.org). Если вы не можете установить `dig`, то, возможно, вам удастся выполнить данный запрос с помощью команд `host` или `nslookup`; хотя последнюю использовать уже не рекомендуется.

Если поиск выполнен успешно, Postfix (теоретически) может корректно выполнять разрешение имен. Если же запрос не выполнен и имена хостов не определяются, вам необходимо срочно разобраться с проблемами DNS.

Одна из распространенных проблем с разрешением имен вызвана попытками обращения к недоступному серверу имен. Проверьте свой файл `/etc/resolv.conf`. Предположим, в нем есть записи такого вида, согласно которым ваш хост обращается к серверу имен на `localhost` (127.0.0.1), а затем, не найдя его там, обращается к хосту 134.169.9.107:

```
nameserver 127.0.0.1
nameserver 134.169.9.107
```

Все хорошо, если на локальной машине запущен кэширующий сервер имен. Но если его там нет, запрос будет ждать окончания тайм-аута.

Если впоследствии выяснится, что разрешение имен в команде `dig` работает, а Postfix тем не менее не может найти хост (например, в журнале присутствует запись «no route to host»[1]), то скорее всего Postfix запущен с помощью `chroot` и использует другой файл конфигурации при разрешении имен. Например, если вы указали в команде `chroot` путь `/var/spool/postfix`, то Postfix будет обращаться к файлу `/var/spool/postfix/etc/resolv.conf`. С помощью команды `cp -p /etc/resolv.conf /var/spool/postfix/etc/resolv.conf` убедитесь, что файлы соответствуют друг другу, затем остановите и снова запустите Postfix.

[1] Эта запись означает, что адрес хоста успешно найден по его доменному имени, но пакеты по этому адресу отправить нельзя, т. к. маршрут к нему неизвестен. Если Postfix не может найти хост, сообщение будет иным: «unknown host». – *Примеч. науч. ред.*

DNS для почтовых серверов

Вы должны настроить свой сервер имен таким образом, чтобы он мог сообщить остальному миру, что доставкой почты в данном домене занимается именно ваш почтовый сервер. Попросите системного администратора (того, кто отвечает за работу сервера имен в вашем домене) добавить следующие записи:

A-запись

Ваш почтовый сервер должен иметь полное доменное имя, чтобы клиенты могли его найти. Запись типа A устанавливает соответствие между доменным именем и IP-адресом.

PTR-запись

Имя вашего хоста должно поддаваться обратному разрешению. Почтовые серверы, получившие это имя в сеансе SMTP, должны иметь возможность убедиться, что они действительно работают с вашим сервером.

MX-запись

Записи типа MX сообщают клиентам, что ваш сервер отвечает за доставку почты для домена или определенного хоста.

A-записи

В системе доменных имен есть разные типы записей, сообщающих хостам о ресурсах Сети. Одной из наиболее важных является A-запись, сопоставляющая имена хостов их адресам. Клиент, отправляющий серверу имен имя хоста, должен получить ответ, содержащий его IP-адрес. Вот пример сеанса, показывающий, что имени `www.example.com` соответствует адрес `192.0.34.166`.

```
$ dig www.example.com A
; <<>> DiG 9.2.1 <<>> www.example.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30122
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0
;; QUESTION SECTION:
;www.example.com.               IN      A
;; ANSWER SECTION:
www.example.com.        172627  IN      A       192.0.34.166
;; AUTHORITY SECTION:
example.com.            21427   IN      NS      b.iana-servers.net.
example.com.            21427   IN      NS      a.iana-servers.net.
;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Apr 17 16:43:40 2004
;; MSG SIZE  rcvd: 97
```

PTR-записи

Аналогично A-записям записи типа PTR устанавливают соответствие адресов именам. Когда клиент посылает серверу имен IP-адрес, в ответ он должен получить соответствующее имя хоста, например:

```
$ dig -x 192.0.34.166
; <<>> DiG 9.2.1 <<>> -x 192.0.34.166
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37949
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 0
;; QUESTION SECTION:
;166.34.0.192.in-addr.arpa.     IN      PTR
;; ANSWER SECTION:
166.34.0.192.in-addr.arpa. 21374 IN     PTR     www.example.com.
;; AUTHORITY SECTION:
34.0.192.in-addr.arpa.  21374   IN      NS      ns.icann.org.
34.0.192.in-addr.arpa.  21374   IN      NS      svc00.apnic.net.
34.0.192.in-addr.arpa.  21374   IN      NS      a.iana-servers.net.
34.0.192.in-addr.arpa.  21374   IN      NS      b.iana-servers.org.
34.0.192.in-addr.arpa.  21374   IN      NS      c.iana-servers.net.
;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Apr 17 16:44:39 2004
;; MSG SIZE  rcvd: 201
```

Предупреждение

Теперь, когда спамеры заполонили Интернет, обратное разрешение A-записей с помощью PTR-записей приобрело особую важность. Многие администраторы почтовых серверов настраивают их так, чтобы они принимали почту только при условии удачного обратного разрешения имени клиента.

Однако если другие почтовые серверы отказываются принимать почту в отсутствие обратного разрешения, это не значит, что и вам следует поступать так же. Часто проблемы вызваны тем, что провайдеры не делегируют возможность обратного разрешения пользовательским серверам имен, а сами не поддерживают актуальную информацию на своих серверах.

MX-записи

Возможности сервера имен не ограничиваются разрешением имен; клиенты могут узнать от него об имеющихся в домене службах. Одна из таких служб – почтовый сервер домена. В записи MX вы можете сослаться на A-запись вашего почтового сервера.

Предупреждение

В DNS имеются также записи CNAME, определяющие синонимы для A-записей. Например, вы можете создать запись CNAME, содержащую `www.example.com` со ссылкой на `srv01.example.com`. Клиенты, запрашивающие `www.example.com`, будут получать ответ с данными о `srv01.example.com`.

Не используйте в записях MX ссылки на эти синонимы. Получивший наибольшее распространение почтовый протокол SMTP требует, чтобы доменное имя в адресе соответствовало записи типа A или MX. В нашем примере нельзя указать в MX-записи имя `www.example.com`, но можно указать `srv01.example.com`, т. к. для него существует запись типа A.

Вы можете создать несколько записей MX и назначить им приоритеты, чтобы клиенты обращались к ним в определенном порядке. Например, так:

```
$ dig m-net.de MX
; <<>> DiG 9.2.1 <<>> m-net.de MX
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3133
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 2, ADDITIONAL: 0
;; QUESTION SECTION:
;m-net.de.                      IN      MX
;; ANSWER SECTION:
m-net.de.               7200    IN      MX      50 mail-in.m-online.net. ❶
m-net.de.               7200    IN      MX      100 mx01.m-online.net. ❷
m-net.de.               7200    IN      MX      100 mx02.m-online.net.
;; AUTHORITY SECTION:
m-net.de.               7200    IN      NS      ns2.m-online.net.
m-net.de.               7200    IN      NS      ns1.m-online.net.
;; Query time: 27 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Apr 17 17:07:05 2004
;; MSG SIZE  rcvd: 140
```

❶ Сервер `mail-in.m-online.net` имеет наивысший приоритет, т. к у него наименьший номер (50). Клиенты сначала будут пытаться отправить почту через него.

❷ Серверам `mx01.m-online.net` и `mx02.m-online.net` назначен приоритет следующего уровня (100). Клиенты будут пытаться использовать один из них, если сервер с высшим приоритетом недоступен.

3

Почтовый сервер одного домена

Настройка Postfix для обслуживания одного домена – дело нескольких минут. Независимо от того, какую конфигурацию вы планируете в дальнейшем, на первом этапе всегда начинайте с конфигурации для единственного домена: так вы сможете убедиться, что Postfix работает в наиболее простом варианте.

В этой главе приведен минимальный набор параметров, необходимых Postfix для начала работы, и показано, как сопоставить длинным адресам электронной почты короткие имена пользователей в конфигурации для одного домена.

Минимальная конфигурация

Мы настроим Postfix на прием почты для одного домена, чтобы он доставлял сообщения с разными адресами в пределах этого домена в разные почтовые ящики.

В задачи Postfix будет входить обработка почты только для этого домена. Мы приведем минимальный набор изменений, которые необходимо выполнить в конфигурации после типовой установки. Типичная архитектура сети для минимальной конфигурации показана на рис. 3.1.

Почтовый сервер имеет постоянное соединение с Интернетом и статический IP-адрес. Прямые (типа A) и обратные записи DNS для почтового сервера созданы.

Для базовой конфигурации существуют начальные требования. Убедитесь, что вы правильно настроили свой хост, как это описано в главе 2.

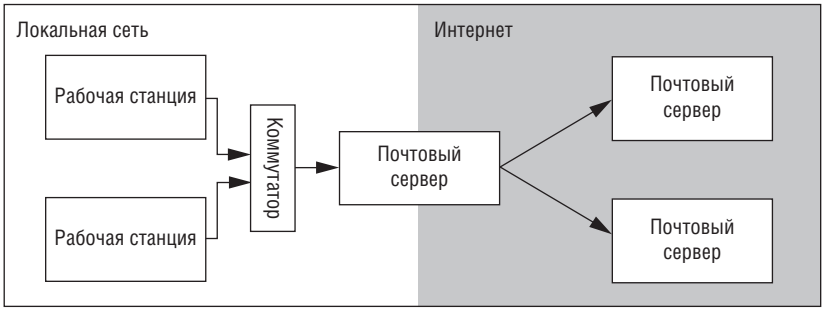

***Рис. 3.1.** Сеть с одним доменом, обслуживаемая Postfix*

Настройка Postfix

В этой главе мы будем настраивать Postfix для приема почты единственного домена. Наш компьютер будет называться `mail.example.com`, а домен – `example.com`. Выполним следующие шаги:

1. Настроим в Postfix корректное имя хоста в приветствии почтовых клиентов.
2. Настроим Postfix на прием почты для домена `example.com`.
3. Настроим Postfix на добавление `example.com` к сообщениям, отправленным только с именем пользователя.
4. Настроим Postfix на доставку почты, адресованной пользователю `root`, в другой почтовый ящик.
5. Настроим Postfix на доставку почты различным пользователям в соответствии с указанным адресом.
6. Установим для Postfix разрешение на пересылку исходящей почты вашей сети.

Настройка имени хоста в заголовке smtpd

Когда почтовый клиент и сервер «встречаются», они приветствуют друг друга, называя свои DNS-имена. Первое, что мы сделаем, – это определим имя, с которым Postfix будет представляться почтовым клиентам. Если имя вашего хоста совпадает с тем, которое вы хотите использовать в приветствии, вам повезло: ничего менять не надо.

Если же в вашей системе установлено имя хоста `www.example.com`, а вы хотите, чтобы Postfix, работающий на этой же машине, использовал в своем приветствии имя `mail.example.com`, то в этом нет ничего сложного.

Предупреждение

Когда Postfix передает сообщения другим почтовым серверам, он выступает в качестве почтового клиента. Представляясь почтовому серверу, он по умол-

чанию использует параметр myhostname в приветствии HELO. Некоторые почтовые серверы настроены так, что отвергают сообщение, если имя в HELO не соответствует полностью определенному доменному имени, полученному обратным разрешением. Либо убедитесь, что имя хоста, заданное для Postfix, соответствует имени, получаемому по IP-адресу вашего сервера, либо установите значение параметра smtp_helo_name в соответствии с официальным именем из пространства имен DNS.

Есть два способа назначить другое имя: задать значение параметра myhostname или параметра mydomain.

Параметр myhostname

Чтобы установить значение параметра myhostname, отредактируйте файл /etc/postfix/main.cf. Откройте этот файл в любом редакторе и найдите строку myhostname. Затем введите полностью определенное доменное имя хоста:

```
myhostname = mail.example.com
```

Как только вы задали параметр myhostname, Postfix может автоматически получить значение mydomain. Postfix просто отбрасывает все до первой точки включительно. А поскольку мы задали параметру myhostname значение mail.example.com, Postfix установит в mydomain значение example.com – что нам и требовалось.

Параметр mydomain

Можно не использовать параметр myhostname, а ограничиться установкой mydomain. Такой вариант может оказаться очень удобным, если вам надо растиражировать конфигурацию на несколько машин.

```
mydomain = example.com
```

Как только вы задали параметр mydomain, Postfix может получить значение myhostname, объединив вывод команды uname -n на данном хосте со значением mydomain. Из этого следует, что, если в файле main.cf явно задан только параметр mydomain и вы копируете его на другие машины в этом же домене (example.com в нашем примере), Postfix самостоятельно сформирует правильное имя хоста.

Настройка домена, в который адресована почта

Для локальных клиентов Postfix будет служить ретранслятором – в том смысле, что он будет принимать почту для тех доменов, в которых он не является конечным или промежуточным сервером. Все, что вам нужно сделать в конфигурации с одним доменом, – это задать значение параметра mydestination. (Действия по настройке Postfix в качестве почтового ретранслятора для нескольких доменов рассмотрены в главах 13 и 14.)

Примечание

Устанавливая значение параметра mydestination, вы можете задать место назначения явно (например, mydestination = mail.example.com) или, используя обозначение $parameter, сослаться на значения других параметров. Жестко заданные значения сильно затрудняют изменение конфигурации, т. к. приходится редактировать множество параметров, что с учетом опечаток и других свойственных человеку ошибок делает такой способ ненадежным. Мы не рекомендуем явное задание значений.

В этой главе наша цель состоит в том, чтобы научить Postfix принимать любую почту, адресованную в домен example.com. В силу того, что мы уже присвоили это значение параметру mydomain, можем просто сослаться на него при задании значения mydestination в файле main.cf:

```
mydestination = $mydomain
```

Если вы хотите тем же способом заставить Postfix принимать почту для хоста, указанного в параметре myhostname, то просто добавьте этот параметр в список mydestination:

```
mydestination = $mydomain, $myhostname
```

Как видите, значения в списке разделяются запятыми, запятая в конце отсутствует. Продолжая в том же духе, вы можете добавить в список www.example.com и ftp.example.com, объединив имена хостов с параметром $mydomain:

```
mydestination =
   $mydomain,
   $myhostname,
   www.$mydomain,
   ftp.$mydomain
```

Этот пример также иллюстрирует другую форму записи. Если в параметре надо указать много значений, то каждое из них можно поместить в отдельной строке, но при этом каждая строка должна начинаться с пробела (иначе Postfix не сможет распознать значение). Вы можете проверить это в окне командного интерпретатора с помощью команды postconf mydestination.

Такой формат может быть использован для любого параметра Postfix, способного принимать несколько значений.

Настройка домена, добавляемого в исходящие сообщения

Когда локальная программа, такая как cron, at или работающий в командной строке почтовый клиент, отправляет почту, она обычно не указывает полный адрес отправителя или получателя, ограничиваясь именем пользователя. Это вполне допустимо для локальных адресатов, но при отправке сообщения на другой хост возникает проблема.

Выяснение того, откуда пришло сообщение, занимает довольно много времени, а в случае отсутствия на указанном хосте нужного адресата принимающий почтовый сервер не сможет вернуть сообщение.

У Postfix есть параметр, значение которого добавляется к адресу отправителя или получателя, если он указан не полностью: `myorigin`. Как и раньше, вы можете ссылаться на параметры, ранее определенные в `main.cf`:

```
myorigin = $mydomain
```

Как только этот параметр вступит в силу, Postfix будет добавлять значение из `mydomain` к любому адресу, если он задан не полностью. Например, сообщение от процесса `cron` пользователю `root` получит адрес `root@$mydomain`, что в нашем случае будет преобразовано в `root@example.com`.

Если вы не укажете значение `myorigin`, то по умолчанию будет подставляться значение параметра `myhostname`, что может быть удобно, если у вас есть несколько хостов, для которых сообщения от `root` должны доставляться на один адрес на центральном сервере. В таком случае вы всегда будете знать, от какого хоста получено сообщение; в сообщении `cron`, например, Postfix превратит `root` в `root@$myhostname`, что в нашем случае будет преобразовано в `root@mail.example.com`.

Перенаправление сообщений для root в другой почтовый ящик

Postfix доставляет почту непосредственно локальным пользователям, включая и пользователя `root`, но в процессе доставки *внешние* программы не смогут получить привилегии `root`. Это означает, что вы не можете использовать локальные агенты доставки (LDA – local delivery agents), такие как `procmail` или `maildrop`, для доставки почты пользователю `root`, т. к. Postfix не будет запускать их с привилегиями `root`. Вместо этого он запустит их с привилегиями `default_privs`, по умолчанию соответствующими привилегиям пользователя `nobody`. Такая мера предосторожности предусмотрена для того, чтобы никоим образом не скомпрометировать учетную запись суперпользователя, запуская с его привилегиями уязвимые внешние программы. Но это не значит, что пользователю `root` нельзя доставить адресованные ему сообщения. Решение заключается в создании отдельного пользователя с обычными низкими привилегиями и перенаправлении ему адресованных суперпользователю сообщений.

В нашем примере мы используем имя `admin` для учетной записи, под которой осуществляется администрирование хоста.[1] Чтобы Postfix

[1] Как раз недавно вирус (червь) использовал адрес отправителя `admin@$mydomain` для своего распространения в Интернете. Имя `admin` не самый лучший выбор для учетной записи.

доставлял почту для root пользователю admin, просто откройте файл /etc/postfix/aliases, созданный при установке по умолчанию[1], и замените postfix на admin. Результат должен выглядеть так:

```
root:   admin
```

Примечание

Если вы решили использовать для этих целей учетную запись admin, то вам надо удалить в файле aliases строку, перенаправляющую почту для admin пользователю root. Иначе получится зацикливание.

Когда вы отредактировали файл /etc/postfix/aliases[2] и добавили туда нужного пользователя, вам надо создать индексированную версию – обычно с именем /etc/postfix/aliases.db, – чтобы ускорить поиск. Для этого надо либо обработать файл /etc/postfix/aliases программой postalias, либо запустить newaliases без параметров. Чтобы воспользоваться возможностями Postfix, выполните такую команду:

```
# postalias hash:/etc/postfix/aliases
```

Предупреждение

Postfix не учитывает изменения в файле aliases до тех пор, пока не будет заменена его индексированная версия, т. к. он обращается только к ней.

Запуск Postfix и проверка доставки почты для root

Пришло время выполнить первые тесты. В предыдущем разделе мы добавили и изменили ряд параметров, и если будем двигаться дальше, не удостоверившись, что все работает правильно, то рискуем в дальнейшем столкнуться с трудностями при поиске ошибки.

Запуск Postfix

Прежде чем начать отправлять почту, мы должны запустить Postfix. Все, что для этого нужно, – ввести postfix start, на что Postfix ответит таким сообщением:

```
# postfix start
postfix/postfix-script: starting the Postfix mail system
```

Если же сообщение будет таким, как указано ниже, значит, Postfix уже работает:

[1] Файл aliases, входящий в дистрибутив Postfix, содержит все адреса, которые должны быть на почтовом сервере согласно различным RFC. В самом файле вы найдете указания, где искать более подробную информацию об этих требованиях.

[2] Входной и выходной форматы файла должны быть совместимы с Sendmail версии 8 и подходить для использования в качестве карт NIS.

```
# postfix start
postfix/postfix-script: fatal: the Postfix mail system is already running
```

Если Postfix был запущен в то время, когда вы занимались изменением конфигурации, то эти изменения не повлияют на его работу. Можно было бы остановить и снова запустить Postfix, чтобы заставить его прочитать конфигурацию, но для этого есть более элегантный способ. Просто введите `postfix reload`:

```
# postfix reload
postfix/postfix-script: refreshing the Postfix mail system
```

В этом случае Postfix загрузит только конфигурацию, что займет меньше времени и не прервет обслуживание клиентов.

Отправка тестового сообщения

Теперь, когда Postfix запущен, можно выполнить первый тест: доставить почту, адресованную `root`, в его почтовый ящик. Это можно сделать двумя очень простыми способами: отправить сообщение из командной строки или из сеанса telnet. Оба способа хороши тем, что позволяют исключить влияние других приложений, таких как сложные почтовые клиенты с графическим интерфейсом, и позволяют сконцентрироваться на Postfix в случае возникновения ошибок.

Отправка сообщения из Postfix-версии sendmail

Самый простой и надежный способ проверки базовой функциональности – использовать программу `sendmail`; в этом случае используются только компоненты Postfix. Эта утилита командной строки названа `sendmail` с целью сохранения совместимости – многие отправляющие почту приложения в UNIX содержат в себе жестко заданную ссылку на программу `sendmail` в виде `/usr/sbin/sendmail` или `/usr/lib/sendmail`. Туда же Postfix помещает свою программу `sendmail`, чтобы сделать переход от Sendmail к Postfix максимально простым.[1]

Для отправки сообщения пользователю `root` введите такую команду:

```
# echo foo |  /usr/sbin/sendmail -f root root && tail -f /var/log/maillog
```

Будет отправлен текст `foo` пользователю `root` с адресом отправителя `root`, а затем для проверки статуса доставки будет открыт почтовый журнал:

```
Aug 20 21:56:42 mail postfix/pickup[5160]: 848AD7247: uid=0 from=<root>
Aug 20 21:56:42 mail postfix/cleanup[5340]: 848AD7247:
  message-id=<20030820195642.848AD7247@mail.example.com>
Aug 20 21:56:42 mail postfix/nqmgr[5161]: 848AD7247:
```

[1] Здесь есть одна тонкость: если вы переходите с Sendmail на Postfix, то можете получить в конце концов две программы `sendmail`: одну от Postfix, вторую – оставшуюся от настоящего Sendmail. Используйте только ту, которая установлена с Postfix.

```
  from=<root@mail.example.com>, size=306, nrcpt=1 (queue active)
Aug 20 21:56:42 mail postfix/local[5343]: 848AD7247:
  to=<admin@mail.example.com>, orig_to=<root>, relay=local, delay=0,
  status=sent (mailbox)
```

Как видно из журнала, Postfix справился с доставкой этого сообщения в почтовый ящик. В этом можно убедиться с помощью команды `less /var/mail/admin`:

```
From root@mail.example.com  Wed Aug 20 21:56:42 2003
Return-Path: <root@mail.example.com>
X-Original-To: root
Delivered-To: admin@mail.example.com
Received: by mail.example.com (Postfix, from userid 0)
        id 848AD7247; Wed, 20 Aug 2003 21:56:42 +0200 (CEST)
Message-Id: <20030820195642.848AD7247@mail.example.com>
Date: Wed, 20 Aug 2003 21:56:42 +0200 (CEST)
From: root@mail.example.com (root)
To: undisclosed-recipients:;

foo
```

Примечание

Если вы не знаете точно, где искать почтовый ящик, введите команду `postconf mail_spool_directory`. Она подскажет вам, куда Postfix доставляет почту.

Пока все хорошо. Postfix может работать со своим собственным приложением.

Отправка сообщения из командной строки

Теперь проверим, сможем ли мы отправить сообщение пользователю `root` из почтового клиента, расположенного на `localhost`. Это второй простейший тест:

```
# mail admin
Subject: Test from command line
This is a test mail from command line.
.
```

Совет

Если вы не знакомы с программой `mail`, вот краткая инструкция:

1. Введите `mail` в командной строке.
2. Введите имя учетной записи, которой вы хотите адресовать сообщение, и нажмите RETURN.
3. Получив приглашение, введите тему и нажмите RETURN.
4. Введите текст сообщения.
5. Для отправки сообщения перейдите на новую строку и введите одну точку (`.`), затем нажмите RETURN.

Чтобы проверить, отправлено ли сообщение, снова выполните команду `less /var/mail/admin`:

```
# less /var/mail/admin
From root@mail.example.com  Wed Aug 20 20:55:11 2003
Return-Path: <root@mail.example.com>
X-Original-To: admin
Delivered-To: admin@mail.example.com
Received: by mail.example.com (Postfix, from userid 0)
        id 37DE07247; Wed, 20 Aug 2003 20:55:11 +0200 (CEST)
To: admin@mail.example.com
Subject: Test from command line
Message-Id: <20030820185511.37DE07247@mail.example.com>
Date: Wed, 20 Aug 2003 20:55:11 +0200 (CEST)
From: root@mail.example.com (root)

This is a test mail from command line.
```

Сообщение доставлено, мы убедились, что локальные пользователи могут посылать почту другим локальным пользователям. Пришло время проверить, может ли удаленный пользователь отправить сообщение пользователю `admin`.

Отправка сообщения из сеанса telnet

Простейший почтовый клиент – это клиент telnet, установивший соединение с портом 25, выделенным для SMTP. Мы пойдем таким сложным путем, т. к. хотим исключить побочные эффекты, возможные при использовании более комфортных (и содержащих больше ошибок) почтовых клиентов. Вот как отправляется сообщение с помощью telnet:

```
# telnet mail.example.com 25
Trying 172.16.0.1...
Connected to mail.example.com.
Escape character is '^]'.
220 mail.example.com ESMTP Postfix
HELO client.example.com
250 mail.example.com
MAIL FROM: <test@client.example.com>
250 Ok
RCPT TO: <root@example.com>
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Test mail from a telnet session.
.
250 Ok: queued as 69F1A7247
QUIT
221 Bye
```

В последний раз проверим доставку командой `less /var/mail/admin`:

```
From test@client.example.com  Wed Aug 20 21:25:16 2003
Return-Path: <test@client.example.com>
X-Original-To: root@example.com
Delivered-To: admin@mail.example.com
Received: from client.example.com (mail.example.com [172.16.0.1])
        by mail.example.com (Postfix) with SMTP id 2D89A7251
        for <root@example.com>; Wed, 20 Aug 2003 21:24:59 +0200 (CEST)
Message-Id: <20030820192459.2D89A7251@mail.example.com>
Date: Wed, 20 Aug 2003 21:24:59 +0200 (CEST)
From: test@client.example.com
To: undisclosed-recipients:;

Test mail from a telnet session.
```

И это сообщение было доставлено; мы убедились в том, что Postfix принимает сообщения, отправленные удаленными пользователями, и доставляет их локальным пользователям.

Сопоставление электронных адресов именам пользователей

Теперь, когда мы заложили основы, пришло время создать более сложные адреса электронной почты. По умолчанию Postfix будет доставлять сообщения только локальным пользователям почтового сервера. Однако имена пользователей (например, `y0000247`), часто используемые для аутентификации при получении почты, редко соответствуют именам, используемым людьми в общении друг с другом (таким, как `john.doe@example.com`). Чтобы Postfix мог доставлять почтовые сообщения с используемыми в реальности адресами соответствующим пользователям, надо создать синонимы, указывающие Postfix, куда следует доставлять поступающие сообщения.

Создание синонимов

Давайте представим, что в вашей компании Example Inc. появился новый сотрудник по имени Джон Доу (John Doe), и ваша задача – создать для него учетную запись электронной почты. Джон работает в отделе продаж, и предполагается, что он должен получать в один почтовый ящик сообщения, отправленные по адресам `john@example.com`, `john.doe@example.com` и `doe@example.com`. Туда же ему должны приходить сообщения, отправленные на адрес отдела `sales@example.com`, где он работает вместе с Сильвией и Кэрол, которые также получают всю почту, приходящую на адрес `sales@example.com`. Джон уже получил учетную запись `john`, которая дает ему доступ к его файлам.

Вам надо сопоставить все эти синонимы (`john@example.com`, `sales@example.com` и т. д.) его локальному имени пользователя. Это делается с помощью записей в файле `/etc/postfix/aliases`. В случае с Джоном для четырех синонимов достаточно трех записей. Для `<john@example.com>` синоним не нужен, поскольку все сообщения на этот адрес будут до-

ставляться Джону как владельцу учетной записи john. Надо добавить в файл /etc/postfix/aliases следующие записи:

```
# users
john.doe:        john
doe:             john
# groups
sales:           silvia, karol, john
```

В завершение вам надо будет выполнить команду postalias hash:/etc/postfix/aliases или newaliases для обновления файла aliases.db.

Примечание

Из приведенного примера видно, что слева указывается локальная часть адреса, а через двоеточие справа – имя пользователя (локальная часть электронного адреса – это все, что расположено до знака @). Каждое определение синонимов может содержать одно или несколько значений, разделенных запятыми. Можно указывать как имена пользователей, так и их электронные адреса. Адреса могут принадлежать пользователям других хостов, а это означает, что вы можете принимать почту на своем сервере и доставлять ее по совершенно другим адресам. Дополнительные сведения можно получить непосредственно из файла aliases или с помощью команды man 5 aliases.

Когда вы добавили все необходимые вам синонимы, надо протестировать эти почтовые ящики подобно тому, как вы делали это раньше.

Установка разрешений на пересылку почты из своей сети

Открытые почтовые серверы (*open relays*) – это ночной кошмар администратора почтового сервера. При любой установке по умолчанию ретрансляция в Postfix ограничена. В типовой конфигурации Postfix будет отправлять только сообщения, полученные от IP-адресов собственной сети. Сетевые адреса Postfix получает из настроенных вами на сервере интерфейсов.

Примечание

На сервере под Linux Postfix будет доверять всем подсетям, в которые входят интерфейсы компьютера. Выполните в Linux команду ifconfig, чтобы получить список всех подсетей, которым Postfix будет доверять по умолчанию.

Настройки по умолчанию действуют до тех пор, пока ваш сервер и хосты, которые используют установленный на нем Postfix, находятся в рамках одной подсети. По мере роста или усложнения структуры вашей сети появляется вероятность того, что эти настройки придется изменить. Например, вам может понадобиться запустить Postfix в демилитаризованной зоне (DMZ) в диапазоне IP-адресов, отличающемся от используемого вашими внутренними хостами. В такой ситуации Post-

fix, вероятно, не позволит вашим клиентам отправлять почту внешним адресатам, и вам придется изменить конфигурацию, установив полномочия на ретрансляцию.

Расширение или ограничение прав на ретрансляцию может быть групповым – при помощи параметра `mynetworks_style`, подходящего для вашей сетевой архитектуры, или же индивидуальным – вы вручную указываете список IP-адресов или их диапазонов в нотации CIDR (Classless Inter-Domain Routing – бесклассовая междоменная маршрутизация, см. приложение С) для `mynetworks`.

Оба метода требуют выполнения изменений конфигурации в файле `main.cf` вручную. Такие усилия по администрированию будут разумны для статических диапазонов IP-адресов, потому что они редко изменяются.

Примечание

Если же вы хотите разрешить ретрансляцию для хостов с динамическими IP-адресами (которые регулярно меняют свой IP-адрес), то ручное администрирование нерационально. Ручное внесение изменений вскоре превратится в скучную и утомительную работу. В главе 16 рассказывается и показывается, как автоматизировать этот процесс.

Групповые полномочия на ретрансляцию

Групповые полномочия на ретрансляцию задаются путем выбора значения параметра `mynetworks_style`: `class`, `subnet` или `host`.

`class`

Если выбрано значение `class`, Postfix распространит полномочия на ретрансляцию почты для всей IP-сети класса A/B/C, для которой сконфигурирован сервер. Например, если вы запускаете Postfix на компьютере с IP-адресом `192.0.34.166` и устанавливаете значение параметра `mynetworks_style = class`, то Postfix будет доверять всей сети класса C, `192.0.34.0/24`, и разрешит пересылку для хостов этого диапазона.

`subnet`

Значение `subnet` указывает, что Postfix разрешит ретрансляцию только для тех подсетей, для которых настроены сетевые интерфейсы сервера. Например, если вы запускаете Postfix на компьютере с IP-адресом `192.0.34.166/30` и задаете значение параметра `mynetworks_style = subnet`, то Postfix будет доверять всем хостам внутри данного диапазона.

`host`

При выборе значения `host` Postfix разрешит ретрансляцию только для того сервера, на котором работает Postfix. Например, если вы запускаете Postfix на компьютере с IP-адресами `192.0.34.166`

и 127.0.0.1 и указываете mynetworks_style = host, то Postfix будет доверять только этим хостам (IP-адреса 127.0.0.1 и 192.0.34.166).

Индивидуальные полномочия на ретрансляцию

Индивидуальные полномочия на ретрансляцию задаются в mynetworks путем создания списка всех хостов и сетей, для которых Postfix может пересылать сообщения (используется CIDR-нотация, значения разделяются запятыми).

Например, если Postfix обслуживает сеть, которая объединяет две подсети (192.168.100.0/24 и 192.168.200.0/24), и вы хотите разрешить серверу ретрансляцию для всех хостов демилитаризованной зоны, в которой он находится (10.0.0.0/30), а также для всех его собственных локальных интерфейсов (127.0.0.0/8), то следует указать такой список:

```
mynetworks = 127.0.0.0/8, 192.168.100.0/24, 192.168.200.0/24, 10.0.0.0/30
```

Примечание

Если у вас множество IP-адресов и диапазонов, то такое перечисление внутри main.cf может стать слишком сложным. Вместо этого вы можете указать путь к отдельному файлу (mynetworks = hash:/etc/postfix/mynetworks) и создать сложный список в этом файле. Однако в этом файле невозможно использование CIDR-нотации для сетей. Если вам необходима CIDR-нотация, укажите mynetworks = cidr:/etc/postfix/mynetworks.

4

Почтовый сервер с коммутируемым соединением для одного домена

Настройка почтового сервера на использование коммутируемого соединения требует внесения лишь незначительных изменений в базовую конфигурацию Postfix. Коммутируемый доступ в Интернет может стоить денег (в частности, в Европе, где действует плата за соединение), так что, возможно, вы не захотите, чтобы почтовый сервер инициировал соединение для каждого исходящего сообщения. Вместо этого сервер может накапливать определенное количество сообщений и затем отправлять их, что повышает экономическую эффективность процесса.

Когда коммутируемое соединение установлено, нужно, чтобы Postfix переслал стоящие в очереди сообщения при помощи хоста-ретранслятора вашего провайдера. Кроме того, вам может понадобиться поддержка SMTP-аутентификации. Также нужно будет автоматически извлекать сообщения, которые не могли быть доставлены локальным пользователям, пока сервер не был подключен к Интернету.

Отличия между сервером с коммутируемым соединением и базовой конфигурацией Postfix состоят в следующем:

Соединение

Почтовый сервер лишь временно соединен с сетью Интернет, поэтому, вероятно, его IP-адрес при каждом соединении будет новым.

Разрешение имен

Сервер не может искать имена хостов, когда он отключен от Интернета. Кроме того, собственная информация сервера в DNS меняется при каждом новом соединении, так что корректное обратное разрешение может оказаться недоступным.

Ограничения на доставку

Ваш интернет-провайдер требует, чтобы вы пересылали почту через его хост; более того, хост-ретранслятор может пересылать сообщения только для аутентифицированных пользователей.

Извлечение почты

Внешние почтовые серверы не могут доставить сообщения непосредственно вашему серверу, т. к. обычно он не подключен к Интернету. Ваш интернет-провайдер должен обрабатывать такую ситуацию, используя почтовый сервер для хранения вашей почты. Когда вам отправляют сообщение, почтовый сервер вашего провайдера принимает его и хранит до тех пор, пока вы не воспользуетесь программой `fetchmail` или другим POP/IMAP-клиентом для извлечения почты и передачи ее вашему локальному MTA.

Примечание

Извлечение почты посредством POP/IMAP и `fetchmail` (*http://catb.org/~esr/fetchmail*) в книге не рассматривается.

На рис. 4.1 изображена типичная сеть с коммутируемым соединением. Один или несколько компьютеров находятся в частной сети, при этом при необходимости доступа к услугам Интернета они используют ваш шлюз с коммутируемым соединением, на котором работает и Postfix-сервер.

Чтобы настроить Postfix для работы в качестве почтового сервера с коммутируемым соединением для одного домена, необходимо выполнить следующие действия (они будут описаны в следующих разделах):

1. Отменить разрешение имен.
2. Проверить разрешения на ретрансляцию.
3. Определить хост-ретранслятор.
4. Отложить передачу сообщений.

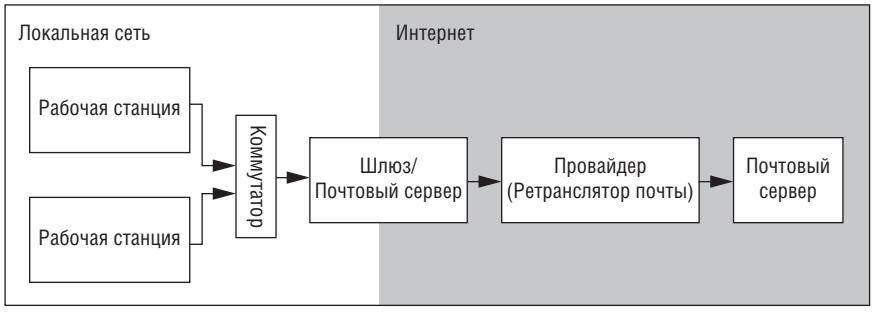

Рис. 4.1. *Типичная сеть с коммутируемым соединением*

5. Инициировать доставку сообщений.
6. Указать права на пересылку для хоста-ретранслятора.

Примечание

Описываемая процедура основывается на настройке, сделанной в главе 3. Вам необходимо сконфигурировать и проверить ваш сервер так, как указано в главе 3. Кроме того, у вашего сервера уже должна быть настроена процедура установления коммутируемого соединения (*http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/PPP-HOWTO.html*).

Отмена разрешения имен

Когда Postfix получает сообщение, которое должно быть доставлено в удаленный домен, он должен найти MX- или A-запись для домена назначения. Для поиска имени на DNS-серверах обычно требуется запрос, обращенный за пределы вашей сети, для чего необходимо подключение сервера к сети Интернет.

Поскольку вы хотите свести коммутируемые соединения к минимуму, следует сообщить Postfix, что поиск данных в DNS не должен начинаться до тех пор, пока сервер не установит соединение. Фактически Postfix вообще никогда не должен заниматься поиском удаленных доменов, т. к. вы хотите, чтобы сервер отправлял сообщения через хост-ретранслятор вашего интернет-провайдера, который сам в состоянии определить, куда отправлять сообщения.

Для того чтобы отменить обращение Postfix к DNS, установите параметр `disable_dns_lookups` в файле `main.cf`:

```
disable_dns_lookups = yes
```

Поиск MX- и A-записи DNS в клиентской программе smtp(8) и поиск A-записи в lmtp(8) будет отменен; в обоих случаях вместо этого будет использоваться вызов `gethostbyname()`. Не забывайте об этом, когда будете определять хост-ретранслятор далее в этой главе.

После установки параметра `disable_dns_lookups` перезагрузите конфигурацию Postfix, чтобы изменение вступило в силу.

Примечание

Сделанная настройка не отменяет использование DNS в серверной программе `smtpd`. Такие параметры, как `reject_unknown_sender_domain` и `permit_mx_backup` (см. главу 8), продолжают действовать вне зависимости от значения параметра `disable_dns_lookups`.

Изменение прав на ретрансляцию

Сервер с коммутируемым соединением обычно имеет динамический IP-адрес, который меняется при каждом соединении сервера с сетью

Интернет. Следовательно, единственным способом управления правами на ретрансляцию для сетевого интерфейса сервера с коммутируемым соединением является определение таких прав вручную при каждом соединении. К тому же кто, кроме спамеров, захочет выполнять ретрансляцию через хост на коммутируемом соединении?

Примечание

Даже если ваш хост только периодически имеет возможность соединения, никогда не следует разрешать пересылку для всего Интернета. К одному из коммутируемых почтовых серверов автора было обращено 56 (неудавшихся) попыток ретрансляции в течение 30 дней. Получается приблизительно по две в день, и это при том, что компьютер не был доступен в сети постоянно! К счастью, ничего не случилось, т. к. пересылка на нем была запрещена.

За исключением тех случаев, когда вы по какой-то странной причине хотите разрешить некоторым пользователям Интернета использовать ваш Postfix-сервер в качестве ретранслятора (см. главу 16), вам следует ограничить ретрансляцию интерфейсом локальной сети и интерфейсом обратной связи (loopback) в файле `main.cf`. Покажем, как это можно сделать для частной сети `192.168.0.0/24`:

```
mynetworks = 192.168.0.0/24, 127.0.0.0/8
```

Предупреждение

Не используйте настройку `mynetworks_style = class` для управления правами на ретрансляцию для сервера с коммутируемым соединением. Этот параметр использует все диапазоны IP-адресов, сконфигурированные для ваших сетевых интерфейсов, в том числе и сеть, в которую звонит ваш сервер. Следовательно, все клиенты сети вашего провайдера смогут использовать ваш почтовый сервер для пересылки сообщений!

Как и раньше, используйте команду `postfix reload` для перезагрузки конфигурации.

Определение хоста-ретранслятора провайдера

Прежде чем выполнять данный этап настройки, необходимо определить хост-ретранслятор вашего интернет-провайдера. Многие провайдеры блокируют исходящие соединения с портом 25 TCP (порт SMTP) для пользователей на коммутируемых линиях, т. к. спамеры злоупотребляют возможностями демонстрационного доступа по коммутируемому соединению.

Примечание

В дополнение к требованиям вашего интернет-провайдера существует множество причин, по которым не стоит просить Postfix доставлять сообщения напрямую конечному адресату. Например, в связи с тем, что источником значительной части спама являются компьютеры с коммутируемым доступом в Интернет,

в черные списки стали включать целые блоки сетей с коммутируемым доступом (аналоговых, ISDN и DSL), про которые известно, что их используют спамеры. Даже если ваше сообщение не является спамом, удаленный агент передачи сообщений (MTA) может отклонить его на основе списка коммутируемых пользователей (DUL – dial-up user list) просто потому, что оно отправлено из диапазона IP-адресов, принадлежащих коммутируемому пулу.

Например, если хост-ретранслятор – `relay.example.com`, используйте такую строку:

```
relayhost= [relay.example.com]
```

Поместив имя или адрес хоста в квадратные скобки, вы отмените поиск MX-записи для данного хоста.

После уже вошедшей в привычку команды `postfix reload` вы готовы двигаться дальше.

Отложенная передача сообщений

На данный момент сервер Postfix имеет конфигурацию, которая доставляет почту хосту ретрансляции без разрешения имен и запрещает пересылку через собственный сервер. Однако сервер все еще звонит интернет-провайдеру всякий раз, когда получает исходящее сообщение, адресованное в удаленную сеть. Для того чтобы изменить такое поведение и заставить Postfix ставить исходящие сообщения в очередь, отредактируйте файл `main.cf`, указав в параметре `defer_transports`, что следует отложить сеанс SMTP, как показано в следующем примере:

```
defer_transports = smtp
```

Примечание

Если вы используете UUCP вместо SMTP, то можете заменить `smtp` на `uucp`.

Как обычно, после внесения изменения выполняем `postfix reload` от имени `root`. После перезагрузки Postfix не будет доставлять сообщения по SMTP до тех пор, пока параметр `defer_transports` не будет изменен или отменен. В следующем разделе будет показано, как использовать эту функциональность для отправки сообщений при установлении соединения сервера с интернет-провайдером.

Инициирование отправки сообщений

Единственное, что нам осталось, – проинструктировать Postfix об отправке всех стоящих в очереди сообщений по SMTP при установлении соединения с Интернетом. Все, что необходимо сделать, – это автоматически реконфигурировать Postfix при переходе сервера в режим онлайн и вернуть его в прежнее состояние при отключении. Задача решается при помощи сценариев, которые запускаются после установле-

ния соединения. В Linux-системе, использующей протокол PPP, такие сценарии часто хранятся в каталоге /etc/ppp/ip-up.d.

Создайте сценарий с именем postfix в этом каталоге, так чтобы он был запущен *после* сценария, определяющего resolv.conf. Сценарий postfix будет выглядеть так:

```
## запустить или перезагрузить Postfix
# если Postfix запускается через chroot, скопировать
# resolv.conf в используемый Postfix файл resolv.conf
cp -p /etc/resolv.conf `postconf -h queue_directory`/etc/resolv.conf ❶
# отменить отложенную отправку и сделать так, чтобы Postfix это заметил
postconf -e "defer_transports ="
postfix reload
# заставить очередь выгрузить всю ожидающую отправки почту
postfix flush
```

❶ Строка, относящаяся к resolv.conf, имеет значение, только если Postfix запущен через chroot. Ее смысл в том, чтобы сервер изменил свой файл resolv.conf при установлении соединения с Интернетом. Postfix также должен знать текущие серверы имен, поэтому команда копирует новую версию в файл, соответствующий chroot, с тем чтобы Postfix мог его найти.

Аналогично при отключении от сети вам следует восстановить старый порядок работы с очередью. Создайте сценарий postfix в каталоге /etc/ppp/ip-down.d, который будет запущен после завершения соединения (опять-таки, строка для resolv.conf необходима, только если используется chroot):

```
## запустить или перезагрузить Postfix
# скопировать resolv.conf в resolv.conf, используемый
# Postfix (только для chroot)
cp -p /etc/resolv.conf `postconf -h queue_directory`/etc/resolv.conf
# включить отложенную отправку и сделать так, чтобы Postfix это заметил
postconf -e "defer_transports = smtp"
postfix reload
```

Назначение прав на ретрансляцию для хоста-ретранслятора

Многие провайдеры бесплатной электронной почты, особенно те из них, кто наряду с почтовым веб-интерфейсом предоставляет клиентам доступ по SMTP, требуют дополнительной авторизации, прежде чем разрешить клиенту использовать пересылку. Это необходимо, потому что большинство пользователей, соединяющихся с почтовым сервером, приходят от других провайдеров (т. е. из диапазонов IP-адресов, отличных от их собственного), так что определение прав на ретрансляцию на основе IP-адресов невозможно. Если бы провайдеры почтовых услуг открыли свои почтовые серверы широкому кругу IP-адресов, эти

серверы фактически превратились бы в открытые и уже через несколько минут стали добычей спамеров. Поэтому провайдеры почтовых услуг требуют аутентификацию «POP-before-SMTP» или SMTP.

POP–before–SMTP

Провайдер, запрашивающий аутентификацию типа «POP-before-SMTP» (POP перед SMTP, см. главу 15), принимает исходящие сообщения на пересылку только в том случае, если вы, прежде чем отправлять новые сообщения, получаете входящую почту. Другими словами, ваш компьютер должен аутентифицироваться на POP3- или IMAP4-сервере провайдера, прежде чем что-либо отправить. Когда ваш хост будет аутентифицирован, провайдер запомнит его текущий IP-адрес и в течение некоторого времени позволит отправку сообщений с этого IP-адреса через свой хост.

Postfix – это агент передачи сообщений, он *не* говорит на языке POP3 и IMAP4. Поэтому Postfix не может самостоятельно выполнить процедуру POP-before-SMTP. Но это не страшно, т. к. вы можете без труда настроить утилиту `fetchmail` (*http://catb.org/~esr/fetchmail*) так, чтобы она делала это для вас (`fetchmail` – это маленькая утилита командной строки, которая извлекает почту из практически любых почтовых систем в Интернете). Чтобы использовать ее для аутентификации POP-before-SMTP, выполните следующие действия:

1. Настройте Postfix так, как описано в этой главе.
2. Следуя инструкциям документации по `fetchmail`, создайте рабочую конфигурацию.
3. Добавьте триггер, который вызывает `fetchmail` перед реконфигурированием Postfix в вашем сценарии `/etc/ppp`.

В этом случае ваш сервер запустит `fetchmail` (клиент POP/IMAP) по крайней мере единожды перед началом этапа отправки сообщений из очереди Postfix (SMTP), так что почтовый провайдер разрешит вам отправку исходящих сообщений.

SMTP–аутентификация

Провайдер, который требует SMTP-аутентификацию, позволяет вашему клиенту или серверу пересылать сообщения через его хост ретрансляции только после авторизации, полученной в результате SMTP-диалога. Для использования SMTP-аутентификации в Postfix вам не потребуется никаких дополнительных сервисов и программ, так что этот вариант является предпочтительным по сравнению с POP-before-SMTP (особенно в тех случаях, когда вы хотите отправить сообщения, но ничего получать вам не нужно).

Подробная информация о настройке SMTP-аутентификации для Postfix на стороне клиента будет приведена в главе 16.

5

Анатомия Postfix

В этой главе рассказывается о том, как работает Postfix, чем занимается каждый элемент системы и как все компоненты связаны друг с другом. После знакомства с этим материалом у вас должна сложиться целостная картина Postfix, и далее вы сможете заняться решением своих собственных задач.

Postfix включает в себя несколько программ, взаимодействующих с пользовательскими процессами (`sendmail`, `postqueue`, `postsuper` и т. д.), и большое количество программ, работающих в фоновом режиме. Программы, работающие в фоновом режиме, управляются демоном `master`. Работа демона `master` состоит в том, чтобы определять, какую задачу следует выполнить, и запускать для ее выполнения соответствующую программу. Модульная структура обеспечивает высокий уровень безопасности, т. к. каждая программа работает с минимальным набором привилегий, необходимых для выполнения своей задачи.

Всю систему Postfix в целом можно воспринимать как маршрутизатор. На первый взгляд, такая мысль может показаться странной, но вспомните, что работа маршрутизатора заключается в том, чтобы посмотреть на IP-пакет, определить IP-адрес места назначения (и, возможно, источника) и выбрать правильный интерфейс для направления пакета к месту его назначения. Postfix делает то же самое с почтой (рис. 5.1): смотрит на место назначения сообщения (получателя конверта) и его источник (отправитель конверта), чтобы выбрать приложение, которое доставит сообщение ближе к его конечному адресату.

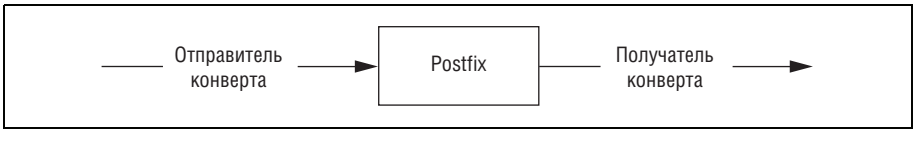

Рис. 5.1. *Postfix работает подобно маршрутизатору*

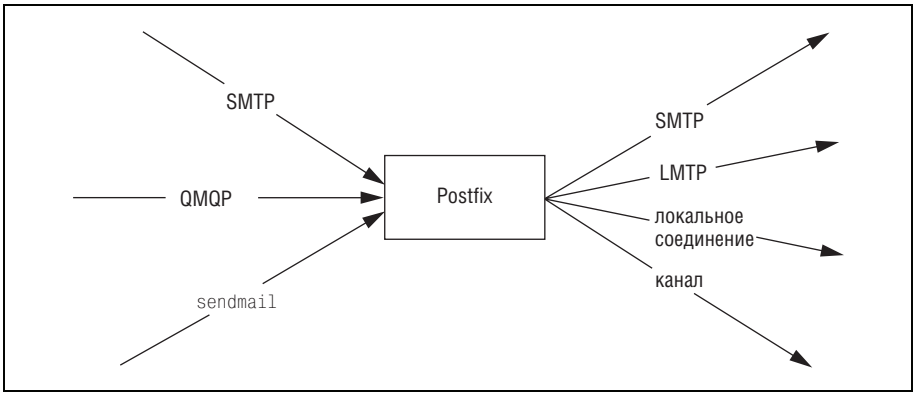

Рис. 5.2. *«Маршрутизатор» Postfix принимает и устанавливает все виды соединений*

Давайте теперь посмотрим на систему более внимательно. Настоящий маршрутизатор принимает IP-пакеты от множества интерфейсов и отправляет их дальше через те же интерфейсы. Postfix действует аналогично – он принимает сообщения от множества источников и передает почту множеству адресатов. Источником сообщения может быть локальная программа `sendmail` либо SMTP- или QMQP-соединение. Местом назначения может быть локальный почтовый ящик, исходящее SMTP- или LMTP-соединение, канал в программу и т. д. (рис. 5.2).

С источником и адресатом сообщения все достаточно понятно, но как Postfix выбирает способ доставки для конкретного места назначения? Маршрутизатор для определения пути использует таблицы маршрутизации, которые сопоставляют IP-адреса сетям. Postfix делает то же самое с электронными адресами.

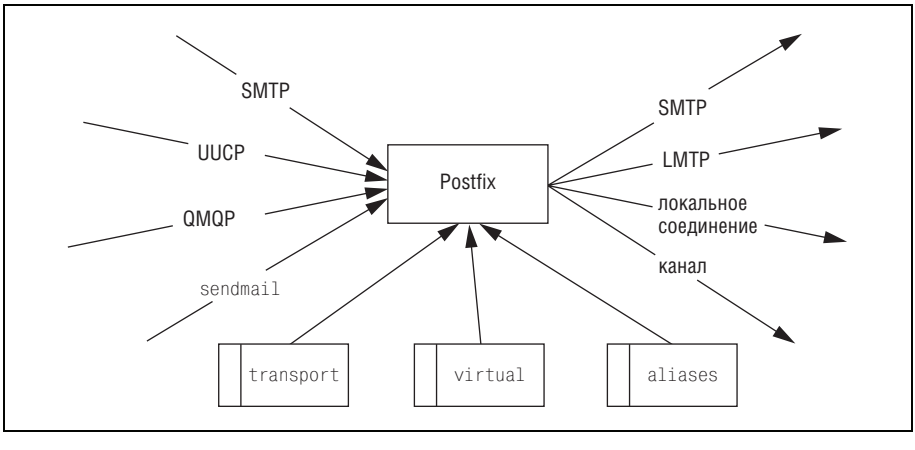

Рис. 5.3. *Карты – это таблицы поиска «маршрутизатора» Postfix*

В Postfix таблицы поиска называются *картами* (*maps*). Postfix использует карты не только для определения того, куда отправить почту, но и для введения ограничений для клиентов, отправителей и получателей, а также для проверки содержимого электронного письма на наличие определенных шаблонов. На рис. 5.3 показано, как карты вписываются в систему Postfix (в качестве примера взяты `aliases`, `virtual` и `transport`, в действительности их гораздо больше).

Демоны Postfix

На рис. 5.4 представлены демоны Postfix и схема отношений между ними.

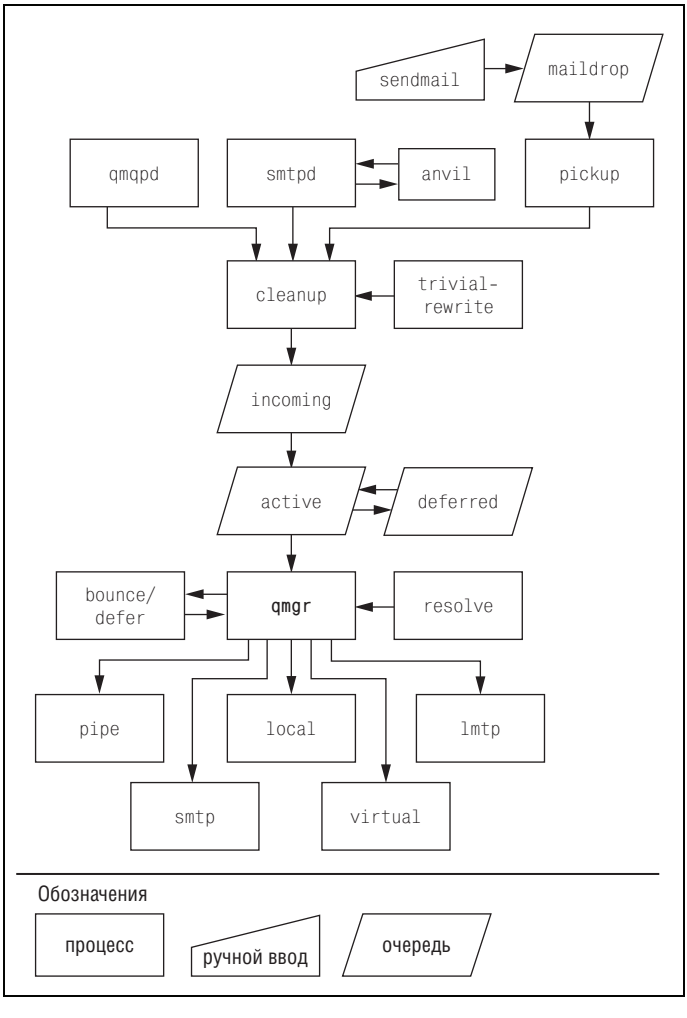

Рис. 5.4. Отношения между демонами Postfix

Примечание

Postfix постоянно находится в процессе развития. Предлагаемый перечень демонов относится к версии Postfix 2.1.[1]

`master`

Демон `master` – управляющая программа Postfix; он осуществляет контроль над всеми остальными демонами Postfix. Демон `master` ожидает поступления входящих заданий для передачи их подчиненным демонам. Если работы много, то `master` может вызвать несколько экземпляров демона. Вы можете задать количество одновременно работающих экземпляров демона, частоту их повторного использования и величину периода простоя, после которого произойдет остановка экземпляра.

Если вы работали с сервером `inetd` на компьютере с UNIX, то обнаружите много общего между ним и демоном `master`.

`bounce` **и** `defer`

Агент передачи сообщений должен уведомить отправителя о невозможности доставить сообщения. В Postfix эту задачу решают демоны `bounce` и `defer`, а инициируется она диспетчером очередей (`qmgr`). Существует два вида событий, которые могут вызвать отправку уведомлений – неисправимые ошибки и адресат, которого невозможно достичь в течение долгого периода времени. В последнем случае отправляется предупреждение о задержке.

`error`

Демон `error` является агентом доставки сообщений, как `local` или `smtp`. Этот агент доставки сообщений всегда вызывает возврат сообщений. Обычно он используется, только если вы хотите настроить домен как недоступный для доставки, направляя почту агенту доставки `error`. Если сообщение отправляется демону `error`, то он информирует демон `bounce` о необходимости отметить, что адресату невозможно доставить почту.

`trivial-rewrite`

Демон `trivial-rewrite` действует по запросу демона `cleanup` для преобразования нестандартных адресов в стандартную форму `user@fqdn`.

Этот демон также занимается разрешением имен адресатов по запросу от диспетчера очередей (`qmgr`). По умолчанию `trivial-rewrite` различает только локальных и удаленных адресатов.

`showq`

Демон `showq` выводит содержимое очереди сообщений Postfix – именно эта программа работает, когда выполняется команда `mailq`

[1] Самая свежая стабильная версия на март 2008 года – Postfix 2.5.1. – *Примеч. науч. ред.*

(sendmail -bp). Этот демон необходим, т. к. очередь Postfix доступна для чтения не всем; пользовательская программа без применения setuid не может вывести содержимое очереди (а программы Postfix не используют setuid).

flush

Демон flush пытается очистить очередь сообщений от зависших и отложенных сообщений. Используя список стоящей в очереди почты, разбитый по адресатам, он улучшает производительность SMTP-запроса Extended Turn (ETRN) и его эквивалента в командной строке – sendmail -qR destination. Список адресатов может содержаться в параметре fast_flush_domains в файле main.cf.

qmgr

Демон qmgr управляет очередями Postfix; это сердце почтовой системы Postfix. Он распределяет задачи доставки между демонами local, smtp, lmtp и pipe. После того как исполнитель задачи выбран, демон qmgr передает ему информацию об имени и пути файла из очереди, адрес отправителя сообщения, хост назначения (для сообщений удаленным пользователям) и один или несколько адресов получателей.

Алгоритм работы qmgr является хорошим примером того, как Postfix при обработке заданий избегает нехватки ресурсов и обеспечивает стабильность. Отметим две особенности:

- qmgr поддерживает маленькую очередь active, в которой ожидают отправки буквально несколько сообщений. Эта очередь фактически выполняет роль небольшого окна для потенциально больших очередей incoming и deferred, предотвращая нехватку памяти qmgr при большой загруженности.
- Если Postfix не может незамедлительно доставить сообщение, то qmgr помещает его в очередь deferred. Хранение временно недоставимых сообщений в отдельной очереди гарантирует, что нормальный доступ к очереди не будет тормозиться большим объемом почты, обработка которой не завершена.

qmgr использует демоны bounce и error для отказа в доставке сообщений тем получателям, сведения о которых содержатся в таблице relocated с данными о пользователях и доменах, более не существующих в системе.

proxymap

Клиентские процессы Postfix могут получать доступ (только на чтение) к картам посредством демона proxymap. За счет совместного использования одной открытой карты многими демонами Postfix proxymap обходит ограничения, налагаемые chroot, и уменьшает количество открытых поисковых таблиц.

`spawn`

Процесс `spawn` по запросу создает не-Postfix процессы. Он занимается прослушиванием порта TCP, сокета домена UNIX[1] или FIFO-очереди стандартного потока ввода, вывода и ошибок. В этой книге будет описан только один пример использования `spawn`: для внешней системы фильтрации сообщений по содержанию тела сообщения (глава 13).

`local`

Как понятно из названия, демон `local` отвечает за доставку в локальные почтовые ящики. Postfix-демон `local` может записывать данные в почтовые ящики форматов mbox и Maildir.[2] Кроме того, `local` может обращаться за данными в формате Sendmail к базам данных `alias` и пользовательским файлам `.forward`.

Примечание

Эти возможности делают `local` аналогом агента отправки сообщений Sendmail, их пользовательские интерфейсы также совпадают.

В качестве альтернативы демон `local` может передать доставку в почтовые ящики локальному агенту доставки (local delivery agent – LDA), который имеет более широкие возможности, например поддерживает фильтрацию. Общераспространенными являются такие LDA, как `procmail` (*http://www.procmail.org*) и `maildrop` (*http://www.flounder.net/~mrsam/maildrop*).

Postfix может запускать несколько экземпляров `local`.

`virtual`

Демон `virtual`, который иногда называют *виртуальным агентом доставки*, – это упрощенный вариант `local`, осуществляющий доставку только в почтовые ящики. Это самый надежный агент доставки Postfix; он не выполняет подстановок по файлам `alias` и `.forward`.

Этот агент доставки может доставлять почту для нескольких доменов, что делает его особенно удобным для поддержки ряда неболь-

[1] Подробнее о сокетах и коммуникационных доменах UNIX можно прочесть, например, по адресу *http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.doc/aixprggd/progcomc/skt_comms.htm*. – *Примеч. науч. ред.*

[2] Почтовые ящики могут иметь разную структуру, и важно, чтобы все программы – и те, которые записывают новые сообщения в почтовый ящик, и те, которые извлекают их из ящика и доставляют пользователю, – понимали эту структуру одинаково. Наиболее распространены форматы mbox (все сообщения хранятся в одном файле и отделены друг от друга пустой строкой) и maildir (каждое сообщение хранится в отдельном файле, подробнее см. *http://www.dataloss.nl/breakage/maildir/*). – *Примеч. науч. ред.*

ших доменов на одной машине (это так называемый POP-тостер[1]) без настройки отдельной почтовой системы для каждого домена или интерактивного доступа пользователей к системе.

`smtp`

Демон `smtp` – это клиентская программа Postfix, которая передает исходящие сообщения удаленным адресатам. SMTP-клиент ищет почтовые серверы назначения, сортирует список по предпочтениям и проверяет каждый адрес до тех пор, пока не найдет сервер, который ответит. При большой нагрузке в системе Postfix обычно работает несколько демонов `smtp` одновременно.

`lmtp`

Демон `lmtp` взаимодействует с локальным и удаленным серверами почтовых ящиков по протоколу LMTP[2] (Local Mail Delivery Protocol), который определен в RFC 2033 (*ftp://ftp.rfc-editor.org/in-notes/rfc2033.txt*). Часто используется совместно с сервером Cyrus IMAP (*http://asg.web.cmu.edu/cyrus/imapd*).

Преимущество использования клиента `lmtp` заключается в том, что управлением очередями занимается Postfix и одна Postfix-машина может по протоколу LMTP обслуживать много серверов почтовых ящиков (на которых должны работать демоны LMTP). Обратное также верно: несколько Postfix-машин могут обслуживать один сервер почтовых ящиков по LMTP. На таких серверах почтовых ящиков может, например, работать Cyrus IMAP.

`pipe`

Демон `pipe` обеспечивает клиентский интерфейс для исходящих соединений к другим почтовым транспортным механизмам. Он вызывает программы с параметрами и отправляет тело сообщения по каналу (pipe) на их стандартный ввод.

`pickup`

Демон `pickup` забирает сообщения, помещенные в очередь `maildrop` локальной клиентской программой отправки почты. После выполнения ряда проверок `pickup` передает сообщения демону `cleanup`.

---

[1] Классическое определение POP-тостера согласно *http://cr.yp.to/qmail/toaster.html*: POP-тостер – это система, содержащая три обязательных компонента, обеспечивающих:

а) хранение почты, подразумевающее запись входящих сообщений на диск;

б) выдачу почты пользователям, обращающимся за ней через сеть;

в) управление учетными записями пользователей, в том числе изменение паролей авторизованным пользователям. – *Примеч. науч. ред.*

[2] На самом деле LMTP расшифровывается как Local Mail Transfer Protocol; он может использоваться вместо SMTP в тех случаях, когда на принимающей стороне не запущен демон SMTP. – *Примеч. науч. ред.*

`smtpd`

Демон `smtpd` обеспечивает взаимодействие с сетевыми почтовыми клиентами, отправляющими сообщения через Postfix по SMTP. Демон `smtpd` проводит ряд проверок для защиты остальной части системы Postfix. Его можно настроить так, чтобы он занимался проверкой входящей почты на спам (локальные или сетевые черные списки, поиск в DNS, другие клиентские запросы и т. д.).

`smtpd` передает сообщение демону `cleanup`.

`cleanup`

Демон `cleanup` реализует последний этап обработки новых сообщений. Он добавляет недостающие заголовки, корректирует преобразование адреса и (если надо) извлекает адреса получателей из заголовков сообщений. Демон `cleanup` помещает результат в очередь `incoming`, затем уведомляет диспетчер очередей о прибытии нового сообщения.

`sendmail`

`sendmail` – это команда Postfix, которая подменяет собой MTA Sendmail Эрика Оллмана (Eric Allman). Ее задача заключается в предоставлении совместимого с Sendmail интерфейса для приложений, способных только вызывать `/usr/sbin/sendmail`. Она взаимодействует с программой `postdrop` для помещения сообщений в очередь `maildrop`, откуда их заберет `pickup`.

Примечание

Использование команды `sendmail` – это самый медленный способ помещения сообщений в систему очередей Postfix. Если вам нужно одновременно отправить большое количество сообщений, воспользуйтесь SMTP.

`qmqpd`

QMQP-сервер Postfix реализует протокол QMQP (Quick Mail Queuing Protocol – быстрый протокол постановки почтовых сообщений в очередь, см. *http://cr.yp.to/proto/qmqp.html*) с целью обеспечения совместимости Postfix с qmail и диспетчером списков `ezmlm`.[1]

`anvil`

Демон `anvil` – это предварительная защита от атак типа «отказ в обслуживании», при которых клиенты засыпают SMTP-сервер огромным количеством одновременных или последовательных попыток соединения. В нем предусмотрена возможность отмены ограничений для авторизованных клиентов. Демон `anvil` не включен в версию

[1] Возможно, вас заинтересует также то, что QMQP более быстр, чем SMTP, и специально создан для ситуации, когда несколько почтовых серверов, соединенных с сетью медленными каналами, отдают исходящие сообщения серверу QMQP, а тот, имея широкий канал в Сеть, занимается обработкой очереди и централизованной рассылкой сообщений. – *Примеч. науч. ред.*

Postfix 2.1, но доступен в экспериментальной версии Postfix 2.2. anvil будет оставаться экспериментальным до тех пор, пока не будет накоплен достаточный опыт ограничения частоты запросов.

Очереди Postfix

Postfix обрабатывает все очереди в каталоге, который определен в параметре queue_directory файла main.cf. Обычно очереди хранятся в каталоге /var/spool/postfix. Каждая очередь имеет свой собственный подкаталог, имя которого идентифицирует очередь. Все обрабатываемые Postfix сообщения находятся в этих каталогах до тех пор, пока Postfix не доставит их. Очередь определяет статус сообщения: incoming, maildrop, deferred, active, hold или corrupt.

`incoming`

Все новые сообщения, попадающие в систему очередей Postfix, отправляются сервисом cleanup в очередь incoming. Новые файлы очередей создаются от имени пользователя postfix с правами доступа 0600. Как только файл очереди готов к дальнейшей обработке, сервис cleanup изменяет права на 0700 и уведомляет диспетчер очередей о том, что поступила новая почта. Диспетчер очередей игнорирует незавершенные файлы очереди с правами 0600.

Диспетчер очередей просматривает очередь incoming, перемещая новые сообщения в очередь active, и следит за тем, чтобы не превысить ее предельный размер. По умолчанию очередь active может содержать не более 20 000 сообщений.

Предупреждение

Когда предельное количество сообщений в очереди active достигнуто, диспетчер очередей прекращает просмотр очередей incoming и deferred.

`maildrop`

Переданные командой sendmail сообщения, которые не были отправлены в первичные очереди Postfix сервисом pickup, ожидают обработки в очереди maildrop. Вы можете добавлять сообщения в очередь maildrop даже тогда, когда Postfix не запущен; он обратится к ним, как только будет запущен.

Однопоточный сервис pickup просматривает и забирает сообщения из очереди maildrop периодически, а также по уведомлению от программы postdrop. Программа postdrop – это setgid-помощник, который позволяет непривилегированной программе sendmail ставить сообщения в очередь maildrop и уведомлять службу pickup о поступлении сообщения (все сообщения, попадающие в основные очереди Postfix, делают это посредством службы cleanup).

`deferred`

Если каким-то из адресатов сообщения все еще не удается доставить его по какой-либо временной причине и всем тем получателям, для которых это возможно, сообщение уже доставлено, Postfix помещает такое сообщение в очередь `deferred`.

Диспетчер очередей периодически просматривает очередь `deferred`, чтобы перемещать отложенные сообщения обратно в очередь `active`. Интервал между просмотрами определяется параметром конфигурации `queue_run_delay`. Если оказывается, что просмотры очередей `deferred` и `incoming` должны состояться одновременно, то диспетчер очередей обращается к ним попеременно: по сообщению из каждой очереди.

`active`

Очередь `active` в некотором роде является аналогом очереди запуска процессов операционной системы. Сообщения в очереди `active` готовы к отправке, но не обязательно находятся в процессе отправки.

Диспетчер очередей – это планировщик агентов доставки, который обеспечивает быструю и четкую доставку почты всем адресатам в пределах выделенных ресурсов.

Примечание

Большая часть администраторов Postfix воспринимает очередь `active` как каталог на диске; на самом же деле очередь `active` – это набор структур данных в памяти диспетчера очередей.

`hold`

Администратор может определить политики `smtpd` access(5) и проверки тела и заголовков с помощью `cleanup` (см. главу 10), которые могут привести к тому, что сообщения будут автоматически исключены из нормальной обработки и помещены на неопределенное время в очередь `hold`. Сообщения остаются в очереди `hold` до тех пор, пока не вмешается администратор. Периодические попытки отсылки сообщений из очереди `hold` не предпринимаются. Используя команду `postsuper`, вы можете вручную поместить сообщения в очередь `hold` или вернуть их оттуда, переместив в очередь `deferred`.

Теоретически сообщения могут оставаться в очереди `hold` более долгий срок, чем указано параметром `maximal_queue_lifetime` для времени жизни файла очереди (после чего недоставленные сообщения возвращаются обратно отправителю). Если необходимо вернуть из очереди `hold` более старые сообщения, то можно использовать команду `postsuper -r` для перемещения их в очередь `maildrop`. В этом случае сообщение получит новую временную метку и еще один шанс быть доставленным.

Примечание

Очередь `hold` не оказывает значительного влияния на производительность Postfix. Внимание к ней вызвано скорее отслеживанием спама и вредоносных программ, чем вопросами производительности.

`corrupt`

Каталог `corrupt` содержит поврежденные файлы очередей. Вместо того чтобы удалять их, Postfix сохраняет их для того, чтобы человек – администратор почтовой системы – мог исследовать их с помощью `postcat`.

При запуске Postfix записывает в журнал предупреждение о каждом поврежденном файле.

Карты

Карты – это файлы и базы данных, которые Postfix использует для поиска информации. Карты могут иметь разное назначение, но у них есть общий признак: левая сторона (left-hand side, LHS), называемая *ключом*, и правая сторона (right-hand side, RHS), называемая *значением*.

Приведем несколько примеров ключей и значений:

Ключ	Значение
`postmaster:`	`John`
`postmaster@example.com`	`John`
`192.168.254.12`	`REJECT`
`spammer@example.com`	`REJECT`
`/^Subject: your account {25}[a-z]{8}/`	`REJECT Mimail Virus Detected`

Работа с картой происходит следующим образом: вы указываете ключ и получаете результат – соответствующее значение.

Примечание

Перечисленные в примере ключи и значения хранятся в разных файлах и не имели бы смысла, будучи собраны вместе в одном файле. Представленный выше список приведен только в качестве иллюстрации того, что записи всех карт имеют общую базовую форму.

Типы карт

Postfix может использовать различные виды карт. Доступные форматы зависят от того, каким образом Postfix был скомпилирован в вашей системе. Для того чтобы определить, какие форматы поддерживает ваш Postfix, выполните в командной строке команду `postconf -m`. Будет выведен перечень типов карт:

```
# postconf -m
btree
cdb
cidr
environ
hash
ldap
mysql
nis
pcre
proxy
regexp
sdbm
static
tcp
unix
```

Индексированные карты (hash, btree, dbm и другие)

Индексированные карты – это двоичные базы данных, созданные из обычных текстовых файлов посредством таких команд, как `newaliases`, `postalias` и `postmap`. Двоичные карты индексированы, так что Postfix может быстро извлекать значение, сопоставленное ключу. Для улучшения производительности демоны Postfix открывают такие карты при запуске и не перечитывают их заново до тех пор, пока не заметят изменения файлов карт в файловой системе. Для перезагрузки карты демон завершает свою работу, и демон `master` запускает новый демон.

Примечание

Если у вас есть часто изменяющиеся индексированные карты, то использующие эти карты демоны будут перезапускаться с такой же частотой. При высокой загрузке это может привести к проблемам с производительностью.

Чаще всего используются индексированные карты, созданные из текстовых файлов `aliases`, `virtual`, `transport`, `relocated` и `sasl_passwd`. Идентифицировать файл карты можно по его имени, состоящему из имени исходного файла и суффикса, обозначающего формат индекса. Например, файл карты `aliases`, созданный командой `postalias`, называется `aliases.db`.

Примечание

Когда вы создаете файл с целью построить на его основе индексированную карту, вам не нужно помещать ключи в каком-то определенном порядке. Программы преобразования, использующие индексированные карты, не требуют на входе определенного порядка. В действительности процесс преобразования устраняет упорядоченность.

Postfix запрашивает записи в предопределенном порядке, указанном в страницах руководства для таблиц access(5), transport(5), virtual(5),

aliases(5) и canonical(5). Другими словами, каждый просмотр карты фактически состоит из серии одиночных запросов (производных от исходного запроса) по отдельным ключам индексированной карты.

Линейные карты (PCRE, regexp, CIDR и обычные файлы)

Линейные карты – это обычные текстовые файлы. Postfix читает такие файлы сверху вниз, что отличает их от индексированных карт. Это весьма важное отличие, т. к. первое совпадение в файле определяет то действие, которое выполнит Postfix. Последующие записи Postfix игнорирует вне зависимости от того, удовлетворяют ли они запросу или нет.

Рассмотрим regexp-карту, в которой поиск `john.doe@example.com` возвращает `OK`, потому что первая строка соответствует условию запроса.

```
/john\.doe@example\.com/ OK
/example\.com/           REJECT
```

Однако если поменять местами строки такой карты, то первое совпадение будет найдено в другой строке, так что тот же поиск `john.doe@example.com` вернет `REJECT`:

```
/example\.com/           REJECT
/john\.doe@example\.com/ OK
```

Преобразовывать линейные карты в двоичную форму *не* нужно (на самом деле вам бы и не удалось это сделать). Демоны Postfix читают их при запуске и не замечают изменений в карте до тех пор, пока не будут перезапущены. Стандартными линейными картами Postfix являются `header_checks`, `body_checks` и `mime_header_checks` (см. главу 9).

Предупреждение

По мере роста линейных карт демоны Postfix будут тратить больше времени на их обработку. Это особенно верно в отношении проверок тела или заголовка, т. к. демон `cleanup` должен сравнить каждую строку тела (вплоть до `body_checks_size_limit`) и заголовка с каждой строкой карты.

Это вызывает значительное снижение скорости работы, особенно если у вас задано много параметров `*_checks`, которые используют карты типа regexp и PCRE (Perl-compatible regular expression – Perl-совместимое регулярное выражение) для предотвращения попадания в систему спама. Если такое случается, обычно это означает, что пришло время поручить комплексную фильтрацию спама внешнему приложению.

Для того чтобы демоны Postfix заметили изменения в линейных картах, выполните команду `postfix reload`. Если скорость работы не очень важна, то можно установить параметр `max_use` для определения времени жизни демонов. Как только демон обработает то количество задач, которое указано в данном параметре, он будет остановлен и перезапущен демоном `master`. После перезапуска все необходимые карты будут считаны заново.

Базы данных (MySQL, PostgreSQL, LDAP)

Postfix воспринимает базу данных так же, как индексированную карту. Результатом запроса базы данных является `Match` (вместе со значением, возвращаемым запросом) или `No match`. Основное отличие между картой–базой данных и индексированной картой состоит в том, что при изменении в базе данных вам не нужно перезапускать демон. Postfix не считает, что единственным, кто может изменять базу данных, является администратор почтовой системы.

Недостаток такого подхода в том, что база данных может не справиться с поступающими запросами, т. к. Postfix должен выполнить как минимум три запроса для каждого поиска в карте (см. ниже раздел «Как Postfix обращается к картам»). При высокой нагрузке компьютер базы данных может замедлиться или зависнуть, и ваш почтовый сервер окажется уязвимым для атак типа «отказ в обслуживании» и «саморазрушение» (self-induced meltdown). Этот факт не должен препятствовать использованию баз данных, но не забывайте о возможном риске.

Поиск в базе данных может стать проблемой для систем с высокой нагрузкой, но это не единственное, на что следует обратить внимание, – еще одной проблемой может стать задержка. Запросам к базе данных соответствует большая задержка, чем запросам к индексированным картам, т. к. Postfix должен подключиться к компьютеру базы данных, отправить запрос и дождаться результата. При работе с индексированной картой Postfix должен лишь обратиться к данным, которые уже загружены в память.

Если база данных становится вашим «узким местом», а ваша карта не слишком велика, то можно использовать промежуточную карту между базой данных и сервером Postfix. То есть вы можете с помощью полного запроса к базе данных создать индексированную карту, а затем запускать Postfix с полученной картой. Необходимо помнить об обновлении такой карты по мере необходимости, при этом можно использовать демон `proxymap` для значительного уменьшения количества одновременных соединений.

Определение количества одновременных соединений с базой данных

Демоны Postfix (`smtpd`, `smtp` и т. д.) работают с ограничением на количество одновременных процессов (задается параметром `default_process_limit`), равным 100. При пиковой загрузке получаем 100 действующих одновременно демонов `smtpd`, каждый из которых обращается к базе данных для одного поиска access(5) (например, если мы используем карту, чтобы проверить, входит ли клиент в наш личный черный список и надо ли запретить ему отправлять нам почту).

Согласно следующему разделу один поиск требует от одного запроса (в случае точного совпадения) и до числа, зависящего от количества «.» (точек) в доменной части (если совпадение не найдено) ключа поиска.

Поэтому в среднем можно принять количество запросов равным 3, тогда количество одновременных запросов к базе данных будет равно как минимум `default_process_limit` * 3 (т. е. 300 запросов с настройками по умолчанию), где `default_process_limit` – количество одновременных соединений. И это только запросы и соединения для демонов `smtpd`; другие демоны, такие как `local` и `qmgr`, могут заниматься другой работой, увеличивая количество открытых соединений и одновременных запросов.

Как Postfix обращается к картам

Карты можно применять для решения различных задач. Postfix включает в себя табличные механизмы, использующие карты (см. access(5), aliases(5), canonical(5) и transport(5)). В этих картах могут использоваться различные механизмы поиска (LDAP, NIS, SQL, btree, hash, regexp, cdb, cidr, pcre и т. д.).

1. *<localpart@domainpart>* Точное полное совпадение с указанным почтовым адресом.
2. *<domainpart>* Совпадение *domainpart* с доменной частью почтового адреса. Шаблону *domainpart* также соответствуют поддомены, но только в случае, если строка `smtpd_access_maps` указана в параметре конфигурации Postfix `parent_domain_matches_subdomains`. В противном случае указывайте *.domainpart* (с начальной точкой) для поиска совпадающих поддоменов.
3. *<localpart@>* Совпадает со всеми почтовыми адресами с указанной пользовательской частью (*localpart*) вне зависимости от того, какому домену они принадлежат.
4. Нет совпадения. Если совпадение не найдено, Postfix возвращает `no match found` и запрос завершается с ошибкой.

Примечание

В некоторых типах таблиц поиска невозможен поиск пустого адреса отправителя. По умолчанию в качестве ключа поиска для пустого адреса отправителя Postfix использует <>. Значение задается в параметре `smtpd_null_access_lookup_key` файла `main.cf`.

Такой порядок поиска подразумевает, что Postfix выполняет несколько поисков для каждого запроса. Это не представляет проблемы, если вы не пользуетесь картами с большой задержкой, такими как SQL или LDAP (и, конечно, следует понимать, что множество поисков потребует выполнения множества запросов). Не забывайте об этом, когда, поместив все свои карты в LDAP, вы будете жаловаться в списках рассылки `postfix-users` на то, что «Postfix такой медленный...».

Примечание

Описанный порядок поиска относится только к картам типа access(5).

Внешние источники

Postfix поддерживает источники информации, которые не встроены в Postfix и даже не находятся под вашим непосредственным контролем, такие как черные списки (DNSBL и RHSBL), списки DNS и другие внешние источники. Черные списки используются практически исключительно в параметрах `smtpd_*_restrictions` для отклонения почты, полученной от клиентов или отправителей, входящих в списки DNSBL или RHSBL (см. главу 7).

Как и в случае любого внешнего запроса, такой поиск может оказаться неудачным из-за проблем с установлением соединения, атак «отказ в обслуживании» против серверов черных списков и других проблем. В случае истечения времени ожидания или другой неполадки Postfix тем не менее может принять почту (игнорируя возможное ограничение), но запишет соответствующее предупреждение в журнал электронной почты.

Утилиты командной строки

Postfix поставляется с набором утилит командной строки, которые помогают решать административные задачи. Они выполняют разнообразные функции (обращение к картам, просмотр файлов очередей, постановка сообщений в очередь и извлечение из очереди, изменение конфигурации), но имеют одну общую характеристику – их имена начинаются с «post».

Примечание

Эти команды могут сделать гораздо больше, чем описано в книге. Мы сосредоточимся на тех возможностях, которые будут нужны вам в каждодневных операциях. Если вы не обнаружите здесь того, что вас интересует, то в первую очередь стоит обратиться за информацией к онлайновому руководству.

postfix

Команда `postfix` останавливает, запускает и перезагружает конфигурацию с помощью параметров `stop`, `start` и `reload`.

postalias

Команда `postalias` создает индексированную карту псевдонимов из файла псевдонимов. Она работает аналогично команде `postmap` (она будет описана ниже), при этом уделяя особое внимание нотации в файле псевдонимов (ключ и значение разделяются двоеточием). Команду `postalias` следует использовать для файлов псевдонимов.

postcat

Команда `postcat` выводит содержимое сообщения, находящегося в почтовой очереди.

Для того чтобы прочитать сообщение, находящееся в очереди, необходимо знать идентификатор очереди. Выполните команду `mailq` для получения списка идентификаторов очередей. Например, идентификатор очереди следующего сообщения равен `F2B9715C0B3`:

```
# mailq
F2B9715C0B3  2464 Mon Oct 13 15:29:39  markus.herrmann@example.com
            (connect to mail.example.com[217.6.113.151]: Connection timed out)
                                       torsten.hecke@example.net
-- 2 Kbytes in 1 Requests.
```

После получения идентификатора очереди укажите его в качестве параметра команды `postcat` для просмотра содержимого файла:

```
# postcat -q F2B9715C0B3
```

postmap

Главная задача команды `postmap` заключается в построении индексированных карт на основе обычных текстовых файлов. Например, для того чтобы создать карту `/etc/postfix/virtual.db` на основе `/etc/postfix/virtual`, выполните такую команду.

```
# postmap hash:/etc/postfix/virtual
```

Команда `postmap` способна и на большее. К ее наиболее полезным возможностям относится способность тестирования карт любого вида, поддерживаемых вашей конфигурацией Postfix. Это чрезвычайно полезно при отладке конфигурации, в которой поиск по карте не работает, когда вы не уверены, доступны ли в действительности серверу Postfix ключи и значения карты.

Отладка поиска записи в таблице поиска

Для того чтобы определить, может ли Postfix найти запись в карте, используйте команду `postmap -q`. Например, следующая команда возвращает значение, присвоенное ключу `<sender@example.com>` в карте `/etc/postfix/sender_access` (типа `hash`):

```
# postmap -q sender@example.com hash:/etc/postfix/sender_access
OK
```

Важно отметить, что команда `postmap` *не* ищет такие элементы, как `<sender@>`, `<example.com>` и `<com>`, хотя они есть на странице руководства access(5). Поиск таких элементов необходимо выполнить вручную:

```
# postmap -q sender@ hash:/etc/postfix/sender_access
# postmap -q example.com hash:/etc/postfix/sender_access
# postmap -q com hash:/etc/postfix/sender_access
```

postdrop

Команда `postdrop` считывает почту из стандартного ввода и записывает результат в каталог `maildrop`. Эта программа работает в связке с утилитой `sendmail`.

postkick

Команда postkick отправляет запрос демону Postfix по локальному транспортному каналу, делая межпроцессное взаимодействие Postfix доступным для сценариев оболочки и других программ.

Примечание

Команда postkick отправляет сообщения процессам демонов Postfix, поэтому Postfix должен быть запущен.

Повторное помещение сообщения в очередь

Следующий более сложный пример использования команды postkick показывает, как повторно поставить сообщение в очередь для незамедлительной доставки:

```
# cat queueidlist | postsuper -r -
postkick public pickup W
```

В приведенной последовательности команда postsuper -r – перемещает все выбранные сообщения, перечисленные в queueidlist, в очередь maildrop, где демон pickup будет обрабатывать их как любые другие сообщения. Таким образом вы устанавливаете фильтр содержимого в режим, соответствующий локальной обработке, и создаете дополнительный заголовок Received:.

Команда postkick вызывает немедленный просмотр очереди maildrop. В противном случае сообщения останутся в очереди maildrop максимум на 60 секунд. Демон pickup передаст сообщение демону cleanup, где оно получит новый идентификатор queueid и будет помещено в очередь incoming. Общая задача состоит в максимально быстром перемещении сообщения в очередь active.

postlock

Команда postlock предоставляет монопольный доступ к файлам mbox, в которые выполняет запись Postfix, а затем исполняет команду, удерживая блокировку. Получаемая от postlock блокировка совместима с агентом доставки Postfix local. Postfix не трогает файл во время исполнения команды. Приведем пример:

```
# postlock /var/mail/user from
```

Предупреждение

Старайтесь избегать команд, для завершения которых может потребоваться нажатие Ctrl-C. Прерывание postlock не гарантирует снятия блокировки; возможно, вам придется удалить файл блокировки, чтобы снова разрешить доставку в почтовый ящик. Для того чтобы проверить, остался ли файл блокировки, запустите postlock без указания команды. Если команда «висит» и время ожидания уже истекло, похоже на то, что у вас осталась незакрытая блокировка.

postlog

Команда postlog позволяет внешним программам, таким как сценарии командного интерпретатора, писать сообщения в журнал электронной почты. Это Postfix-совместимый интерфейс регистрации. По умолчанию он пишет текст из командной строки в виде одной записи. Рассмотрим очень простой пример:

```
# postlog This is a test
postlog: This is a test
# grep "This is a test" /var/log/mail.log
Feb 20 11:50:16 mail postlog: This is a test
```

postqueue

Команда postqueue — это пользовательский интерфейс для очередей Postfix, предоставляющий возможности, обычно доступные в рамках выполнения команды sendmail.

- Команда postqueue с параметром -f просит диспетчер очередей доставить всю стоящую в очереди почту вне зависимости от места назначения, что является эквивалентом postfix flush или sendmail -q:

  ```
  # postqueue -f
  ```

- Команда postqueue с параметром -p выводит содержимое очереди; эквивалент mailq:

  ```
  # postqueue -p
  ```

- Команда postqueue с параметром -s domain пытается доставить всю стоящую в очереди почту для домена domain; эквивалент sendmail -q domain:

  ```
  # postqueue -s example.com
  ```

Примечание

Команда postqueue отправляет сообщения процессам демонов Postfix, поэтому Postfix должен быть запущен.

postsuper

Команда postsuper обслуживает задания внутри очередей Postfix. В отличие от postqueue, эта команда доступна только суперпользователю, и она может быть выполнена, когда сервер не запущен. Некоторые свойства postsuper необходимы для проверки очереди перед запуском процессов демонов. В табл. 5.1 показано, что умеет делать команда postsuper.

Одним из наиболее распространенных случаев использования команды postsuper является удаление сообщения из очереди при помощи вызова postsuper -d queueid. Вручную делать это утомительно, особенно если нужно удалить много файлов. Следующий сценарий на Perl (delete-from-mailq) упрощает выполнение задачи:

```
#!/usr/bin/perl
$REGEXP = shift || die "no email-address given (regexp-style, e.g.
bl.*\@yahoo.com)!";
@data = qx</usr/sbin/postqueue -p>;
for (@data) {
  if (/^(\w+)(\*|\!)?\s/) {
     $queue_id = $1;
  }
  if($queue_id) {
    if (/$REGEXP/i) {
      $Q{$queue_id} = 1;
      $queue_id = "";
    }
  }
}
#open(POSTSUPER,"|cat") || die "couldn't open postsuper" ;
open(POSTSUPER,"|postsuper -d -") || die "couldn't open postsuper" ;
foreach (keys %Q) {
  print POSTSUPER "$_\n";
};
close(POSTSUPER);
```

Таблица 5.1. Возможности команды postsuper

Параметр	Действие
-d	Удаляет сообщение с указанным идентификатором очереди из указанной очереди (очередей) сообщений
-h	Помещает сообщение на удержание, так чтобы не предпринимались попытки его доставки
-H	Освобождает сообщения, в настоящее время находящиеся на удержании
-p	Очищает временные файлы, оставшиеся после аварий
-r	Повторно ставит в очередь сообщения с указанным идентификатором из указанной очереди сообщений
-s	Проверяет и исправляет структуру очереди

Вот как мы будем его использовать:

```
# mailq
C73A015C095     7509 Mon Oct 13 14:56:17  MAILER-DAEMON
         (connect to mx5.ancientaward.com[64.156.166.211]: Connection refused)

National_Nosepicking_Month@mx5.ancientaward.com
```

Обратите внимание на то, что отправитель здесь идентифицирован как <MAILER-DAEMON>. Для того чтобы удалить эти уведомления о невозможности доставки, запустите команду `delete-from-mailq` от имени пользователя `root`:

```
# delete-from-mailq MAILER-DAEMON
postsuper: C73A015C095: removed
postsuper: Deleted: 1 message
```

II

Контроль содержимого

Postfix включает в себя три набора функций, управляющих поступлением сообщений в почтовую систему и отправкой их из системы. При помощи этих функций вы можете самостоятельно управлять потоком сообщений на основе SMTP-диалога и содержимого сообщения или доверить управление содержимым внешним приложениям. Этим функциям соответствуют три отдельных группы параметров конфигурации: ограничения, проверки и фильтры.

Пособие для начинающих администраторов почтовой системы

Для того чтобы контролировать содержимое, необходимы знания об этом содержимом. Для того чтобы эффективно применять ограничения, проверки и фильтры, вы должны знать, что должно, могло бы и может входить в сообщение. Прочтите главу 6, чтобы проникнуть вглубь содержимого электронного письма.

Как работают ограничения на передачу сообщений

SMTP-взаимодействие регулируется ограничениями. В главе 7 будет объяснено, как работают ограничения. Не жалейте времени на ее изучение, это значительно упростит для вас дальнейшую реализацию ограничений.

Использование ограничений на передачу сообщений

В главе 8 мы покажем, как использовать ограничения на практике. Все они могут быть применены практически сразу же.

Как работают встроенные фильтры содержимого

К содержимому сообщений можно применять проверки. Как они работают? Глава 9 знакомит вас с проверками и представляет всю их теорию.

Использование встроенных фильтров содержимого

В главе 10 приведены разнообразные примеры, которые помогут вам быстро приступить к работе.

Как работают внешние фильтры содержимого

Внешние фильтры содержимого передают управление SMTP-взаимодействием и управление содержимым внешним приложениям. Для того чтобы понять, каким образом Postfix обрабатывает сообщения, проходящие через внешние фильтры содержимого, обратитесь к главе 11.

Использование внешних фильтров содержимого

Хотите увидеть несколько примеров реализации внешних фильтров содержимого? В главе 12 вы найдете примеры, которые помогут вам в вашей практике.

6

Пособие для начинающих администраторов почтовой системы

Термины *«конверт»* (*envelope*), *«заголовок»* (*header*), *«тело»* (*body*) и *«вложение»* (*attachment*) относятся к различным частям данных, которыми обмениваются агенты передачи сообщений (MTA). Если вы знаете, что они означают, то вам будет понятно, на какие части сообщений действуют параметры управления содержимым Postfix. Так же удобно, что имена и синтаксис параметров берут начало в документах RFC.

Эта глава представляет собой «букварь» по управлению содержимым сообщений. Прочтите ее внимательно и уделите достаточно времени знакомству с концепциями. После освоения основ вы без проблем добьетесь эффективного управления содержимым.

Основы передачи сообщений

Понятие передачи сообщений имеет два аспекта: SMTP-соединение, которое обеспечивает передачу, и передаваемые данные (которые обычно называют «электронным письмом» или «сообщением»). Термины, используемые для описания передачи сообщений, не были придуманы на пустом месте; они были позаимствованы у старинной, но хорошо известной и устоявшейся системы, которую люди прошлых столетий называли «почтой».

Когда речь идет об обычной почте, значение терминов *«доставщик»*, *«конверт»*, *«заголовок»*, *«тело»* и *«вложение»* хорошо известно. По отношению к электронной почте эти слова являются техническими терминами. На рис. 6.1 обычное письмо сравнивается с электронным, при этом выделены следующие составляющие:

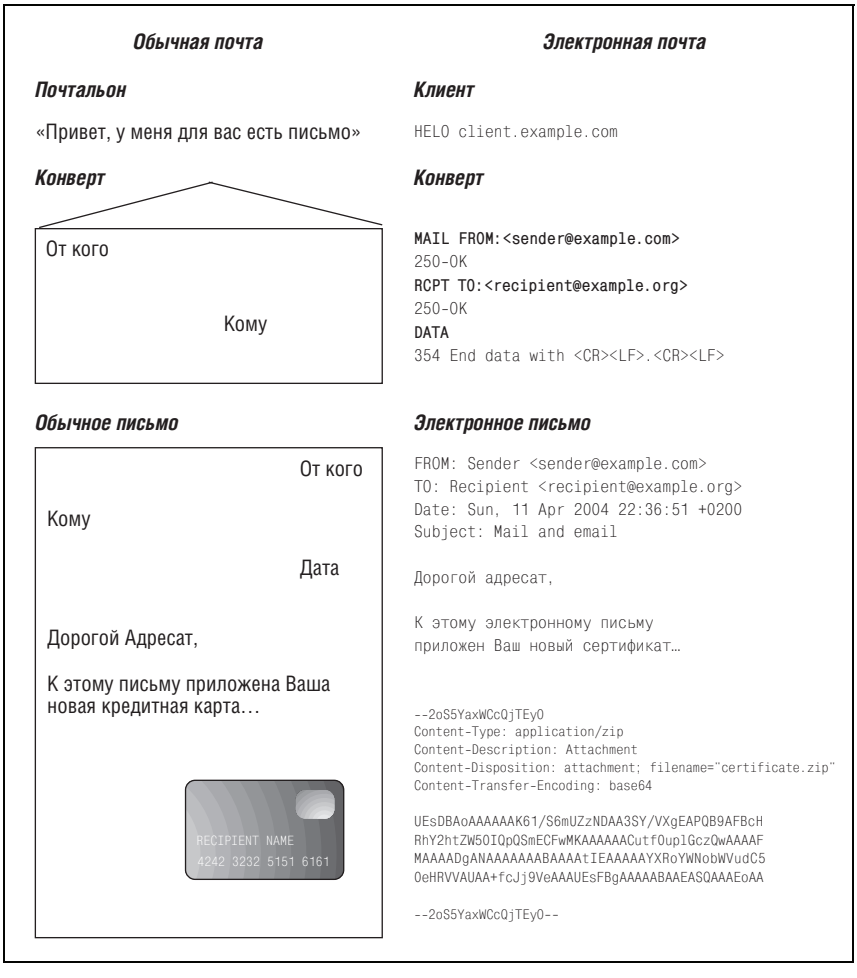

Рис. 6.1. *Сравнение обычного письма с электронным*

Доставщик

В обычной почте доставщик – это *почтальон* или *почтовый курьер*. В электронной почте доставщик – это *клиент*.

Конверт

В электронном письме, как и в обычном, конверт служит упаковкой, которая объясняет, как должно быть доставлено содержимое. На конверте указаны *отправитель конверта* и *получатель конверта*.

Заголовок

Заголовок предоставляет вам метаданные о сообщении. Как и в обычном письме, это информация об отправителе (заголовок From:), получателе (To:), дате и времени отправки (Date:) и теме сообщения

(Subject:). Кроме того, заголовки Received: электронного сообщения сообщают вам о пути, проделанном сообщением, и времени, в течение которого оно передавалось.

Тело

В теле электронного сообщения находится собственно его содержимое, совсем как в обычном письме.

Вложения

Если электронное сообщение содержит вложения, этот факт будет отмечен в теле письма, как это было бы сделано и в обычном письме. Вложения не являются обязательными и могут иметь разнообразные форматы.

Зачем вам это знать?

Все вышесказанное может показаться далекой от жизни теорией – при чем тут работа с Postfix? Во-первых, в электронном сообщении обычно больше информации, чем в обычном письме. Вам необходимо знать, что это за дополнительные элементы и в какой части сообщения они появляются. К тому же Postfix имеет три отдельных группы параметров управления содержимым, которые непосредственно связаны с различными частями сообщения:

`smtpd_*_restrictions`

Параметры `smtpd_*_restrictions` управляют клиентским соединением и конвертом в процессе передачи сообщений.

`*_checks`

Параметры `*_checks` контролируют заголовок, тело и вложения.

Фильтры

Postfix использует фильтры для передачи задач другим (внешним) отбраковывающим приложениям. Фильтры универсальны – они могут контролировать любую частью сообщения, от конверта до вложения.

Каждый из этих параметров имеет большое количество возможных значений; если вы не будете знать, на какую часть сообщения действует каждый из параметров, то ваш контроль содержимого не будет работать.

Управление SMTP-соединением (конверт)

В SMTP-соединении участвуют клиент (компьютер, который обращается к SMTP-серверу) и передаваемый клиентом конверт. Давайте рассмотрим ситуацию на примере, используя для подключения к серверу программу telnet на вашем компьютере.

Начнем с подключения в командной строке к порту 25 вашего почтового сервера:

```
$ telnet mail.example.com 25
220 mail.example.com
```

Возвращаемый сервером код 220 подтверждает имя хоста сервера. Теперь наша очередь представиться серверу:

```
HELO client.example.com
250 mail.example.com
```

«Рукопожатие» можно осуществить при помощи команды HELO (для SMTP) или EHLO (для ESMTP), указав имя хоста клиента в качестве параметра. В случае успешного выполнения команды вы получите код возврата 250, за которым будет следовать имя хоста сервера.

Давайте теперь отправим почту. Команда MAIL создает конверт, начиная с указания отправителя. Если сервер признал отправителя, вы снова получите код возврата 250:

```
MAIL FROM:<sender@example.com>
250 Ok
```

Следующий этап создания конверта – указание получателя конверта с помощью команды RCPT. Вы можете указать одного или нескольких получателей:

```
RCPT TO:<recipient@example.com>
250 Ok
RCPT TO:<recipient_2@example.com>
250 Ok
```

Примечание

Помните, что отправитель конверта и получатель конверта часто отличны от отправителя и получателя, указанных в заголовке сообщения (они указываются внутри команды DATA, о которой мы сейчас будем говорить). Если вы перепутаете разных получателей и отправителей, то ваше управление содержимым может дать сбой.

Для того чтобы отправить настоящее сообщение (включая все дополнительные заголовки, такие как Subject, To и Date), используйте команду DATA:

```
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: message
...

This is the message
...
.
250 Ok: queued as 92933E1C66
QUIT
```

Приведем теперь краткое описание только что рассмотренных понятий, представленное в RFC для электронной почты:

Клиент

Клиент – это компьютер, отправляющий почту; Postfix запишет имя и IP-адрес хоста или же «unknown» (если в результате поиска в DNS невозможно определить имя хоста). Прежде чем будет открыто SMTP-соединение, Postfix получает IP-адрес клиента из TCP/IP-стека ядра, а имя от DNS или из `/etc/hosts`. Это позволяет Postfix наложить ограничения в случае, если в SMTP-соединении возникнет несоответствие IP-адреса клиента и имени хоста.

Postfix всегда записывает IP-адрес клиента и имя хоста (если оно имеется) в почтовый журнал, а также добавляет эту информацию в последний заголовок сообщения.

Команда `HELO/EHLO`

Клиент должен представиться почтовому серверу, сообщив о себе следующие сведения: тип обслуживания и имя хоста.

Первая часть команды представления – это запрашиваемый клиентом тип обслуживания. `HELO` определяет обычное обслуживание, как описано в RFC 821 (*ftp://ftp.rfc-editor.org/in-notes/rfc821.txt*), а `EHLO` – расширенное, как описано в RFC 2821 (*ftp://ftp.rfc-editor.org/in-notes/rfc2821.txt*).

Вслед за запрашиваемым типом обслуживания должна быть идентифицирована личность клиента (предполагается, что клиент предоставит полное имя хоста).

Конверт

Конверт должен содержать как минимум два различных элемента: ровно одного отправителя конверта и как минимум одного получателя конверта. Клиент отправляет конверт, передавая сначала имя отправителя, затем имена получателей конверта.

Если получателей у конверта несколько, то клиент должен передавать их имена одно за другом, каждое с новой строки, ожидая ответа сервера после каждого нового имени.[1] В задачу сервера входит разрешение доставки всем или нескольким получателям.

Отправитель конверта

Отправитель конверта – это отправитель, которому Postfix ответит в случае ошибки: отложенной доставки или получении уведомления о возврате.

Получатель конверта

Получатель конверта – это подразумеваемый получатель (получатели) сообщения. Одно сообщение может иметь несколько получателей конверта (например, сообщение нескольким подписчикам списка рассылки).

---

[1] Конвейерная обработка команд ESMTP является исключением из этого правила.

Почтовый сервер требует указания хотя бы одного получателя конверта (в противном случае ему некому доставлять сообщение). Поэтому клиент не может указать пустого получателя конверта (<>).

Предупреждение

Когда вы будете накладывать ограничения на получателей сообщения, не обращайте внимания на получателей, указанных в поле заголовка To. Сообщение передается получателям, определенным в конверте, а не в заголовке сообщения.

Практически все данные из предыдущего списка могут быть сфальсифицированы, поэтому Postfix позволяет снизить риск подлога при помощи параметров `smtpd_*_restrictions`, которые определяют ответы на такие вопросы:

1. Откуда пришел клиент?
2. Кем клиент представился?
3. Есть ли у клиента специальные привилегии?
4. Кто является отправителем?
5. Кто является получателем?

Postfix также пытается дать ответы на более сложные вопросы:

1. В правильной ли форме клиент предоставляет информацию Postfix?
2. В правильном ли порядке клиент предоставляет информацию?
3. Всю ли информацию предоставил клиент?
4. Если клиент предоставил не всю необходимую информацию, попытается ли он отправить сообщение?
5. Можно ли определить, корректна ли информация?
6. Если возможно это определить, лжет ли клиент?

Postfix может дать ответы на эти вопросы, исследовав конверт сообщения и элементы SMTP-диалога.

Когда Postfix отвергает сообщение на основе ограничений для SMTP-конверта, он отвергает сообщение до его получения. Поэтому Postfix не отправляет уведомление о «недоставимости» сообщения по адресу отправителя, этим должен заниматься клиент.

Примечание

Если Postfix отвергает сообщение на основе ограничений для SMTP-конверта, Postfix не должен уведомлять об этом, т. к. он отказал клиенту в обслуживании. Этот подход помогает сохранить системные ресурсы, не увеличивает[1] трафик

[1] Не просто «не увеличивает», а кардинально уменьшает трафик: вместо того чтобы принять сообщение (возможно, с вложением в несколько мегабайт) и затем пытаться отправить обратно сообщение о недоставке, Postfix сразу отказывается принимать сообщение на основании ограничений. – *Примеч. науч. ред.*

и может быть чрезвычайно полезным в случае спам-атаки, которая потребовала бы тысяч возвратов, если бы сообщения изначально были приняты для дальнейшей передачи.

В главе 7 вы узнаете о том, какие ограничения существуют в Postfix и как они работают. В главе 8 предлагается ряд примеров, которые вы сможете использовать в своей собственной конфигурации.

Контроль содержимого сообщения

Сообщение электронной почты состоит из заголовка и тела. Тело также может включать в себя одно или несколько вложений в виде файла или другого сообщения, инкапсулированного в главное сообщение. На рис. 6.2 представлен общий вид простого сообщения с вложением.

На рис. 6.3 изображено сообщение, вложением в которое является другое сообщение.

Эти части сообщения можно найти, просматривая его в редакторе, работающем с обычным текстом. Вот, например, сообщение с файловым вложением:

```
Return-Path: <sender@example.com> ❶
X-Original-To: recipient@example.com
Delivered-To: recipient@example.com
Received: by mail.example.com (Postfix)
        id 9F71443F50; Mon, 26 Apr 2004 01:32:59 +0200 (CEST)
```

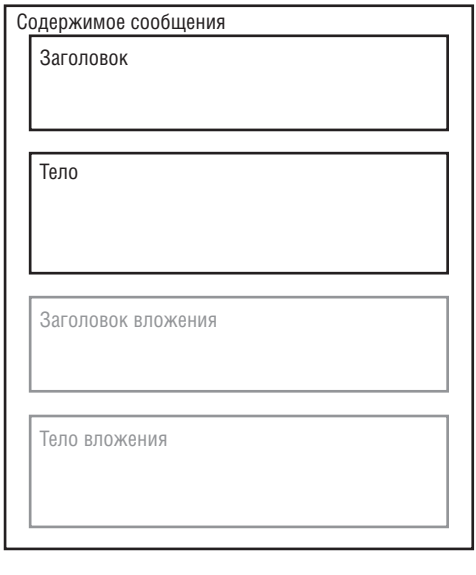

***Рис. 6.2.** Сообщение электронной почты с вложением*

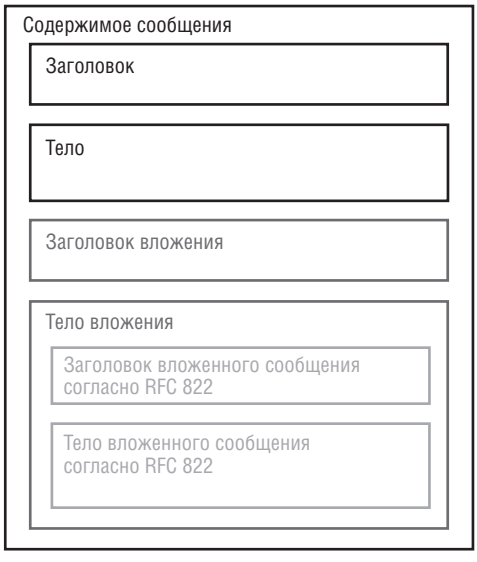

Рис. 6.3. *Сообщение электронной почты с другим сообщением в качестве вложения*

```
Delivered-To: recipient@example.com
Received: by mail.example.com (Postfix, from userid 500)
        id 2F23043F4F; Mon, 26 Apr 2004 01:32:59 +0200 (CEST)
Date: Mon, 26 Apr 2004 01:32:58 +0200
From: Sender <sender@example.com>
To: Recipient <recipient@example.com>
Subject: Elements of email content
Message-ID: <20040425233258.GA22383@mail.example.com>
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="/9DWx/yDrRhgMJTb"
Content-Disposition: inline
User-Agent: Mutt/1.5.4i

--/9DWx/yDrRhgMJTb ❷
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline

A blank line separates the body of a message from the headers. MIME-
encoded text and MIME-encoded attachments may appear in the body.

You may attach one or more files, including another email message.
A message within another message includes its own header and body.
Therefore, you may have nested headers.

Hope this helps,

sender

--/9DWx/yDrRhgMJTb ❸
```

```
Content-Type: application/x-zip-compressed
Content-Disposition: attachment; filename="attachment.zip"
Content-Transfer-Encoding: base64

UEsDBAoAAAAAAIILmjBOMx1uCwAAAAsAAAAOAAAAYXROYWNobWVudC50eHRhdHRhY2htZW50
ClBLAQIUAAoAAAAAAIILmjBOMx1uCwAAAAsAAAAOAAAAAAAAAAAAAAEAIAC2gQAAAABhdHRhY2ht
ZW50LnR4dFBLBQYAAAAAAQABADwAAAA3AAAAAAA=

--/9DWx/yDrRhgMJTb--
```

Сообщение состоит из следующих частей:

❶ Заголовки электронного сообщения.

❷ Начало тела сообщения.

❸ Начало вложения.

Postfix может произвести проверку каждого из этих элементов (`header_checks`, `body_checks`, `mime_header_checks`) в отдельности. Для того чтобы такие проверки были эффективными, вам необходимо знать, какие обязательные, рекомендованные и необязательные элементы может включать в себя сообщение.

Заголовки

Заголовок несет в себе метаданные о теле сообщения, такие как кодировка символов и дата отправки. RFC 2822 (*ftp://ftp.rfc-editor.org/in-notes/rfc2822.txt*) делит элементы заголовка на обязательные и рекомендованные.

Примечание

Поля заголовка *не* обязаны появляться в определенном порядке. Однако рекомендуется отправлять заголовки в следующем порядке: Return-Path, Received, Date, From, Subject, Sender, To, Cc и т. д. Подробную информацию о заголовках можно найти в документе «Reading Email Headers» (*http://www.stopspam.org/email/headers.html*).

Обязательные заголовки

Два элемента заголовка являются обязательными:

Date

Поле даты обычно содержит дату и время, в которое сообщение было составлено и отправлено. Если клиент отправителя пропустил этот заголовок, то Postfix добавит его.

From

Это поле идентифицирует личность человека, отправившего сообщение. Если клиент отправителя пропустил этот заголовок, то Postfix добавит его.

Рекомендованные заголовки

Приведем перечень рекомендованных элементов заголовка:

Message-Id

Это поле содержит уникальный идентификатор, который определяет текущую версию текущего сообщения. Клиент генерирует идентификатор сообщения и гарантирует его уникальность. Кроме того, идентификатор сообщения предназначен для компьютеров и не обязательно будет означать что-то, понятное для людей. Так как идентификатор сообщения соответствует ровно одному экземпляру конкретного сообщения, то любые последующие редакции сообщения получают новые идентификаторы.

Если клиент отправителя пропустил этот заголовок, то Postfix добавит его.

To

Это поле определяет основного получателя сообщения. Если клиент отправителя пропустит это поле, то Postfix вставит туда значение параметра конфигурации `undisclosed_recipients_header`.

Subject

Это поле должно содержать очень краткое описание сообщения.

Cc

Это поле указывает дополнительных получателей сообщения.

Reply-To

Это поле указывает, куда клиент отправителя должен отправлять ответ на сообщение.

Content-type

Это поле описано в RFC 1049 (*ftp://ftp.rfc-editor.org/in-notes/rfc1049.txt*), оно определяет структуру тела сообщения.

MIME-Version

Если такое поле заголовка присутствует, это означает, что тело сообщения было (предположительно) составлено в соответствии с RFC 1521 (*ftp://ftp.rfc-editor.org/in-notes/rfc1521.txt*).

Received

Каждый транспортный агент, который сталкивается с сообщением, добавляет одну такую строку заголовка для того, чтобы отметить, куда, когда и как поступало сообщение. Сведения из этих полей могут быть полезны при трассировке в случае проблем с передачей.

Return-Path

Это поле указывает отправителя конверта и используется для определения пути обратно к автору. Почтовый сервер вставляет это поле после доставки локальным агентом доставки, таким как демон `local`.

Необязательные заголовки (X-заголовки)

X-заголовок – это общий термин для дополнительных полей заголовка, имя которых начинается с заглавной буквы «X», за которой следует дефис. X-заголовки предназначены только для передачи нестандартных данных, и наоборот – любой нестандартный информативный заголовок должен быть X-заголовком.

Приведем несколько примеров X-заголовков (конечно же, существуют миллионы других):

```
X-Mailer: Ximian Evolution 1.4.3
X-Priority: 3
X-Spam-Checker-Version: SpamAssassin 2.53 (1.174.2.15-2003-03-30-exp)
X-Original-To: recipient@example.com
```

Тело

В теле помещается собственно сообщение, оно должно находиться после раздела заголовков. Тело может представлять собой открытый или закодированный текст. Тело также может включать вложения, закодированные так, чтобы не допустить искажений при передаче через Интернет (в старые времена многие агенты передачи сообщений не пропускали восемь битов, а отрезание восьмого бита искажает двоичный файл).

Вложения

Вложения – это файлы, конвертированные в текстовый формат без элементов форматирования (только печатаемые символы), пригодный для отправки в качестве электронного сообщения. В мозаике вложений много различных элементов, о них будет рассказано в последующих подразделах.

MIME-кодировки

MIME – это сокращение от Multipurpose Internet Mail Extensions (многоцелевые расширения электронной почты). Речь идет о системе переопределения формата сообщений, описанной в RFC 2045 (*http://www.rfc-editor.org/rfc/rfc2045.txt*). Для двоичных файлов широко используются две MIME-кодировки: quoted-printable и base64:

quoted-printable

Кодировка quoted-printable предназначена для представления данных, которые по большей части состоят из октетов, соответствующих печатаемым символам в наборе символов US-ASCII. Данные кодируются таким образом, что изменение получившихся октетов в процессе передачи почты маловероятно

base64

base64 – это система кодирования данных, которая определена в RFC 1421 (*ftp://ftp.rfc-editor.org/in-notes/rfc1421.txt*) и RFC 2045

(*ftp://ftp.rfc-editor.org/in-notes/rfc2045.txt*). Она предназначена для преобразования двоичных данных в печатаемые символы ASCII. По существу это кодировка передачи MIME-содержимого в Интернете, использующая только буквенно-цифровые символы (A–Z, a–z, цифры 0–9) и символы «+» и «/», при этом символ «=» является специальным кодовым суффиксом. Ручное кодирование и декодирование осуществляются при помощи утилит командной строки `mpack`, `munpack` и `uudeview`.

Все современные почтовые клиенты поддерживают MIME, и вложения обычно отправляются только в кодировке base64.

Обработка кодировки

Почтовый клиент занимается кодированием двоичного вложения, а также автоматически создает MIME-структуру, необходимую для того, чтобы встроить текст письма и закодированные вложения в форме, понятной другим поддерживающим MIME почтовым клиентам. Для этого в сообщении должны быть следующие поля:

MIME-Version

Наличие такого заголовка указывает на то, что сообщение представлено в MIME-формате. Значение, как правило, равно 1.0, так что заголовок обычно выглядит так:

```
MIME-Version: 1.0
```

Content-type

Этот заголовок указывает тип и подтип содержимого сообщения, например:

```
Content-type: text/plain
```

Сочетание типа (`text` в данном примере) и подтипа (`plain`) обычно называется MIME-типом, так что в нашем примере MIME-тип – это `text/plain`.

Очень многие форматы файлов имеют зарегистрированные MIME-типы. IANA ведет архив всех зарегистрированных типов (*ftp://ftp.isi.edu/in-notes/iana/assignments/media-types*). К тому же все текстовые типы имеют дополнительный необязательный параметр `charset`, который указывает кодировку символов. Очень большому количеству кодировок символов соответствуют зарегистрированные названия MIME `charset`.

Типы содержимого

В этом разделе будет представлено несколько MIME-типов, которые вам, вероятнее всего, встретятся. В дополнение MIME-тип `multipart-mime-message` (многоэлементное MIME-сообщение) позволяет сообщениям состоять из нескольких различных частей, организованных в древовидную структуру, где узлы-листья имеют не многоэлементный тип со-

держимого, а узлы, не являющиеся листьями, относятся к одному из многоэлементных типов. MIME-механизм поддерживает (среди прочих) следующие типы:

`text/plain`

Простые текстовые сообщения используют тип `text/plain`; это значение по умолчанию заголовка Content-type.

`multipart/mixed`

Этот тип указывает текст плюс вложения (`multipart/mixed` с элементом `text/plain` и другими нетекстовыми элементами). MIME-сообщение с вложенным файлом обычно указывает исходное имя файла в заголовке Content-disposition, так что тип файла определяется как MIME-типом содержимого, так и расширением имени файла (обычно зависящим от операционной системы).

Вирусы часто рассылаются как файлы, в которых заголовки Content-type и Content-disposition указывают разные типы файлов.

`message/rfc822`

Это ответ с вложенным исходным сообщением (`multipart/mixed` с элементом `text/plain` и с исходным сообщением в виде элемента `message/rfc822`). Postfix обычно возвращает сообщения таким образом (вложение `message/rfc822` – это исходное сообщение, которое было возвращено).

`multipart/alternative`

Этот тип определяет содержимое с двумя альтернативными способами просмотра, например сообщение, отправленное как простым текстом, так и в формате HTML (одно и то же содержимое в форматах `text/plain` и `text/html`). Outlook Express использует этот тип содержимого по умолчанию, т. к. отправляет почту одновременно в виде HTML и простого текста.

Структура кодирования

Заголовок Content-type многоэлементного MIME-сообщения включает в себя границу, которая помечена в сообщении как `boundary`, и эта граница не должна встречаться ни в одной из частей. Вместо этого она должна появиться между частями, а также в начале и в конце тела сообщения. Рассмотрим пример многоэлементного сообщения:

```
Return-Path: <sender@example.com>
X-Original-To: recipient@example.com
Delivered-To: recipient@example.com
Received: by mail.example.com (Postfix)
        id 9F71443F50; Mon, 26 Apr 2004 01:32:59 +0200 (CEST)
Delivered-To: root@example.com
Received: by mail.example.com (Postfix, from userid 500)
        id 2F23043F4F; Mon, 26 Apr 2004 01:32:59 +0200 (CEST)
Date: Mon, 26 Apr 2004 01:32:58 +0200
```

```
From: Sender <sender@example.com>
To: Recipient <recipient@example.com>
Subject: Elements of email content
Message-ID: <20040425233258.GA22383@mail.example.com>
Mime-Version: 1.0 ❶
Content-Type: multipart/mixed; boundary="/9DWx/yDrRhgMJTb" ❷
Content-Disposition: inline
User-Agent: Mutt/1.5.4i

--/9DWx/yDrRhgMJTb ❸
Content-Type: text/plain; charset=us-ascii ❹
Content-Disposition: inline

A blank line separates the body of a message from the headers. MIME-
encoded text and MIME-encoded attachments may appear in the body.

You may attach one or more files, including another email message.
A message within another message includes its own header and body.
Therefore, you may have nested headers.

Hope this helps,

sender

--/9DWx/yDrRhgMJTb ❺
Content-Type: application/x-zip-compressed ❻
Content-Disposition: attachment; filename="attachment.zip"
Content-Transfer-Encoding: base64 ❼

UEsDBAoAAAAAAIILmjBOMx1uCwAAAAsAAAAOAAAAYXROYWNobWVudC50eHRhdHRhY2htZW50
ClBLAQIUAAoAAAAAAIILmjBOMx1uCwAAAAsAAAAOAAAAAAAAAAAAAAEAIAC2gQAAAABhdHRhY2ht
ZW50LnR4dFBLBQYAAAAAAQABADwAAAA3AAAAAAA=

--/9DWx/yDrRhgMJTb-- ❽
```

Сообщение включает в себя следующие элементы:

❶ Заголовок MIME-версии.

❷ Заголовок, содержащий тип содержимого и граничную строку, которая используется для разделения различных частей сообщения.

❸ Первое вхождение граничной строки. Здесь начинается новый элемент многоэлементного сообщения.

❹ Эта часть является простым текстом.

❺ Второе вхождение граничной строки, указывающее на то, что предыдущая часть завершена и здесь начинается новая часть многоэлементного сообщения.

❻ Новая часть – это файл в формате Zip.

❼ Zip-файл представлен в кодировке base64.

❽ Это последнее использование граничной строки, обозначающее конец части и сообщения.

7

Как работают ограничения на передачу сообщений

Для того чтобы понять, что можно ограничивать, нужно знать, что такое «что» и чем оно должно быть...

– Патрик в попытке понять, что Ральф рассказывает ему об ограничениях...

Эта глава представляет теоретические сведения об ограничениях. Ограничения позволяют вашему почтовому серверу принять или отвергнуть сообщения на основе данных SMTP-соединения между клиентом и сервером. Информация, полученная из этого диалога, позволяет Postfix наложить или отменить ограничения на клиент, отправителя и получателя.

Слово «ограничить» (restrict) обычно означает, что вы устанавливаете какие-то пределы, но в Postfix термин «ограничение» (ограничение) может также означать и обратное – при помощи ограничений вы можете явно разрешить нечто.

Триггеры ограничений

Ограничения – это мощный инструмент. Чтобы эффективно им пользоваться, необходимо понимать, каким образом происходит SMTP-взаимодействие и какими средствами Postfix это взаимодействие анализирует. Мы уже говорили об SMTP-взаимодействии в главе 6, сейчас посмотрим на него под другим углом. Теперь нас будут интересовать этапы SMTP-диалога, обозначенные командами, выдаваемыми клиентом (рис. 7.1).

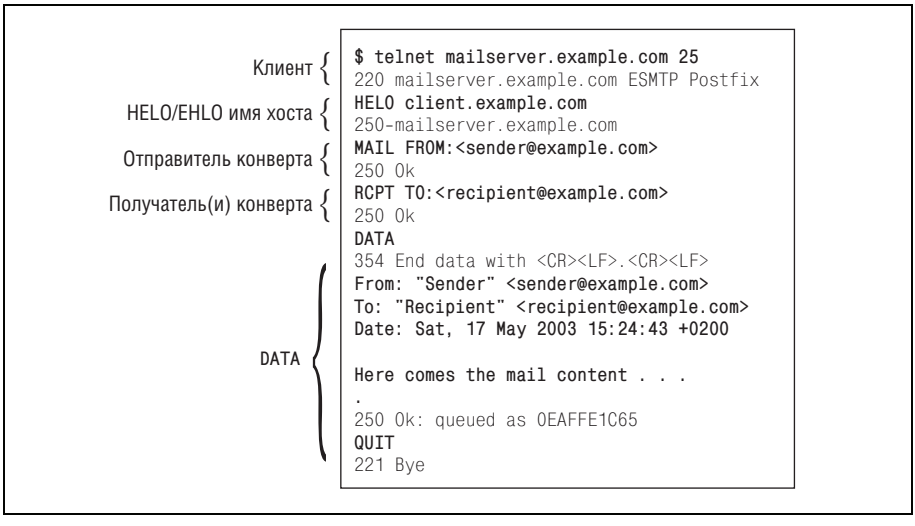

***Рис. 7.1.** Этапы SMTP-взаимодействия и стандартный клиентский ввод*

Каждый новый шаг на рис. 7.1 отмечает момент, когда демон `smtpd` получает очередную порцию информации о клиенте и сообщении, которое он хочет передать. Postfix использует эти этапы для инициирования триггеров ограничений; для каждого этапа существует свой собственный параметр ограничения, имя которого образовано из названия активного демона, названия этапа и его предназначения. Поэтому названия триггеров ограничений строятся по такому шаблону: `smtpd_`*`название_этапа`*`_restrictions`.

Приведем список всех триггеров ограничений с указанием их поведения по умолчанию:

`smtpd_client_restrictions`

Этот триггер применяется к IP-адресу или имени хоста клиента либо к ним обоим. По умолчанию Postfix разрешает подключение любому клиенту.

`smtpd_helo_restrictions`

Этот триггер применяется к аргументу `HELO/EHLO` клиента и к IP-адресу и (или) имени хоста клиента. По умолчанию допускается любой аргумент `HELO/EHLO`.

`smtpd_sender_restrictions`

Это первый набор триггеров, который относится к частям конверта. Postfix применяет его к отправителю конверта, аргументу `HELO/EHLO` и клиенту. По умолчанию любому отправителю конверта разрешено отправлять сообщения.

`smtpd_recipient_restrictions`

Этот триггер применяется к получателям конверта, отправителю конверта, аргументу `HELO/EHLO` и к IP-адресу и (или) имени хоста клиента. По умолчанию Postfix допускает любых получателей для клиентов, которые определены в параметре конфигурации `mynetworks`, для остальных же разрешены получатели в доменах из `relay_domains` и `mydomains`. Это не дает Postfix возможности превратиться в открытый почтовый сервер.

`smtpd_data_restrictions`

Этот триггер выявляет клиенты, которые отправляют содержимое письма прежде, чем Postfix ответит на команду `DATA`. Postfix делает это посредством трассировки команды `DATA`, когда клиент отправляет команду на сервер. По умолчанию ограничения нет.

`smtpd_etrn_restrictions`

Этот специальный триггер может ограничить клиенты, которые могут запрашивать у Postfix очистку очереди сообщений. По умолчанию всем клиентам разрешено выдавать команду `ETRN`.

Каждый триггер ограничений соответствует набору ограничений; вы можете воспринимать триггеры как пустые коробки. Для того чтобы от них была какая-то польза, следует вложить в них ограничения.

Типы ограничений

В Postfix существуют несколько видов ограничений, которые можно разбить на четыре группы:

- Общие ограничения
- Переключаемые ограничения
- Настраиваемые ограничения
- Дополнительные параметры контроля спама

Общие ограничения

Ограничения первой группы ничего в SMTP-диалоге не проверяют, они просто выполняют команды:

`permit`

Разрешает запрос.

`defer`

Откладывает запрос.

`reject`

Отвергает запрос.

`warn_if_reject`

Содействует последующим ограничениям; если ограничение после `warn_if_reject` решает отвергнуть запрос, то Postfix на самом деле не отвергает сообщение, а вместо этого пишет в журнал сообщение `reject_warning`.

`reject_unauth_pipelining`

Отвергает запрос, когда клиент отправляет команды SMTP раньше времени, еще не зная о том, действительно ли Postfix поддерживает конвейерную обработку команд ESMTP. Тем самым достигается противодействие программам массовой рассылки, которые некорректно используют конвейерную обработку команд ESMTP для ускорения доставки.

Переключаемые ограничения

Ограничения второго типа работают как переключатели. Их можно включить или выключить и при активации они проверяют выполнение некоторого условия. Приведем их неполный список:

`smtpd_helo_required`

Это ограничение требует от клиентов отправки команды `HELO` (или `EHLO`) в начале сеанса SMTP. Наличия команды `HELO/EHLO` требуют RFC 821 и RFC 2821.

`strict_rfc821_envelopes`

Это ограничение регулирует степень терпимости Postfix к ошибкам в адресах, указанных в команде `MAIL FROM` или `RCPT TO`. К сожалению, широко используемая программа Sendmail[1] допускает нестандартное поведение, и в результате существует множество программ, которые ожидают, что им это сойдет с рук. Применение строгих мер может остановить нежелательную почту, но может также и заблокировать легальные сообщения от плохо написанных клиентов.

`disable_vrfy_command`

SMTP-команда `VRFY` позволяет клиентам проверять существование получателя. Это ограничение позволяет отменить команды `VRFY`.

`allow_percent_hack`

Это ограничение регулирует преобразование из формы `user%domain` в `user@domain`.

`swap_bangpath`

Это ограничение контролирует преобразование из формы `site!user` в `user@site`. Оно необходимо, если ваш компьютер подключен к сети UUCP.

---

[1] Имеется в виду популярный демон sendmail (*www.sendmail.org*), а не одноименная программа из комплекта поставки Postfix. – *Примеч. науч. ред.*

Настраиваемые ограничения

Настраиваемые ограничения – это карты, которые работают как фильтры. В каждой записи карты ключ является фильтром, а значение – тем действием, которое необходимо выполнить при совпадении (список возможных действий приведен в разделе «Общие ограничения»). Рассмотрим несколько видов настраиваемых ограничений:

`HELO (EHLO)` имя хоста

Эти ограничения относятся к именам хостов, которые клиенты могут отправлять с командой `HELO` или `EHLO`.

Имя хоста/адрес клиента

Этими ограничениями определяются клиенты, которые могут устанавливать SMTP-соединения с почтовым сервером.

Адрес отправителя

Эти ограничения определяют адреса отправителей (конвертов), которые Postfix разрешает для использования в командах `MAIL FROM`.

Адрес получателя

Эти ограничения определяют адреса получателей (конвертов), которые Postfix разрешает для использования в командах `RCPT TO`.

ETRN-команды

Ограничение накладывается на клиенты, которые могут выдавать команды `ETRN`.

Проверка заголовка

Ограничение регулирует заголовки сообщений. Шаблоны применяются ко всему логическому заголовку целиком, даже в том случае, когда логический заголовок занимает несколько физических строк текста.

Проверка тела

Ограничения накладываются на текст, который может появляться в строках тела сообщения.

Черные списки DNSBL

Эти черные списки ограничивают соединения от IP-адресов (клиентов), которые включены в черные списки DNSBL.

Черные списки RHSBL

Эти черные списки запрещают те домены отправителей (конверта), которые присутствуют в черных списках RHSBL.

Дополнительные параметры контроля спама

Дополнительные параметры контроля спама поддерживают другие ограничения или возможности, не входящие в функциональность Postfix по умолчанию. Приведем лишь несколько таких ограничений:

`default_rbl_reply`

Создает шаблон ответа по умолчанию, который будет использоваться при блокировании запроса SMTP-клиента ограничением `reject_rbl_client` или `reject_rhsbl_sender`.

`permit_mx_backup_networks`

Ограничивает использование функции контроля за пересылкой `permit_mx_backup` теми адресатами, у которых основные хосты MX входят в указанный список сетей.

`rbl_reply_maps`

Определяет таблицы поиска и шаблоны ответов DNSBL, индексированные по имени домена DNSBL. Если шаблон не найден, Postfix будет использовать шаблон `default_rbl_reply`.

`relay_domains`

Указывает Postfix на необходимость приема почты для этих доменов несмотря на то, что данный сервер не является местом их конечного назначения.

`smtpd_sender_login_maps`

Определяет пользователя, которому разрешено использовать определенный адрес `MAIL FROM` (отправитель конверта). Для того чтобы Postfix мог использовать это ограничение, ему необходимо знать имя пользователя, так что клиент должен идентифицироваться посредством SMTP-аутентификации.

Области применения

Для того чтобы правильно применять ограничения, необходимо понимать, на каком этапе SMTP-взаимодействия их можно использовать. Некоторые ограничения не имеют никакого смысла на определенных этапах. В табл. 7.1 показано, на каком этапе какие ограничения следует использовать.

Таблица 7.1. Области применения ограничений

Этап	Ограничение

Клиент (IP-адрес и/или имя хоста)	`check_client_access`
	`reject_rbl_client`
	`reject_rhsbl_client`
	`reject_unknown_client`

Команда `HELO/EHLO` *имя_хоста*	`check_helo_access`
	`permit_naked_ip_address`
	`reject_invalid_hostname`
	`reject_non_fqdn_hostname`
	`reject_unknown_hostname`

Этап	Ограничение
Отправитель конверта	check_sender_access reject_non_fqdn_sender reject_rhsbl_sender reject_unknown_sender_domain reject_unverified_sender
Получатель конверта	check_recipient_access permit_auth_destination permit_mx_backup reject_non_fqdn_recipient reject_unauth_destination reject_unknown_recipient_domain reject_unverified_recipient
Команда DATA	reject_unauth_pipelining

Создание ограничений

Ограничения могут стать очень сложными, и вы рискуете бесконечными усложнениями просто испортить свой почтовый сервер, запутавшись в попытках настроить ограничения. При создании ограничений старайтесь придерживаться следующих правил:

- Некорректная запись сделает ваши ограничения бесполезными.
- Имеет значение то, на каком этапе происходит оценка ограничений.
- Имеет значение порядок появления ограничений в триггере. Предыдущие действия влияют на оценку последующих ограничений.

Запись ограничений

Как мы уже говорили, триггеры ограничений подобны пустым коробкам. Однако это не означает, что для их наполнения достаточно набросать туда ограничений.

```
триггер_ограничения = условное_ограничение, настраиваемое_ограничение \
maptype:/path/to/the/map, общее_ограничение
```

Одно ограничение может иметь размер, превышающий ширину строки, поэтому можно добавить пробел в начало каждой строки продолжения, чтобы Postfix распознал строки как относящиеся к одному параметру.

Кроме того, запятые, разделяющие ограничения, являются необязательными. Так что приведенная ниже запись эквивалента предыдущей (и гораздо удобнее для чтения):

```
триггер_ограничения =
   условное_ограничение
   настраиваемое_ограничение maptype:/path/to/the/map
   общее_ограничение
```

Момент оценки

Вообще говоря, Postfix не осуществляет оценку и выполнение ограничений на основе триггера ограничений сразу же после наступления соответствующего этапа SMTP-взаимодействия. Postfix ждет того момента, когда клиент отправит данные первого получателя конверта. Наличие этой задержки объясняется тем, что некоторые почтовые клиенты продолжают пытаться отправить свое сообщение, если сервер отвергает команду прежде, чем закончена передача данных по крайней мере одного получателя конверта.

Для того чтобы изменить такое поведение по умолчанию, установите параметр `smtpd_delay_reject` в значение `no`.

Однако, несмотря на то, что можно отследить такие клиенты и создать список исключений, чтобы гарантировать, что они не будут прерваны, лучше всего подождать завершения всех этапов и уже затем применять ограничения. Тем самым вы не только уменьшите сложность вашей почтовой системы, но и соберете больше сведений о попытке доставки сообщений.

Для того чтобы понять, как параметр `smtpd_delay_reject` влияет на оценку ограничений, посмотрите на рис. 7.2.

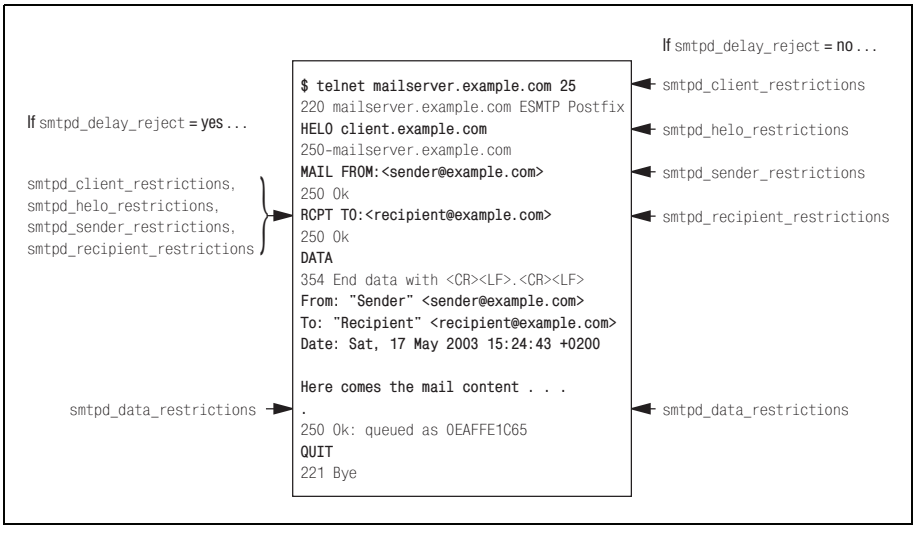

Рис. 7.2. *Влияние параметра smtpd_delay_reject на оценку ограничений*

Влияние действий на оценку ограничений

Как вы уже знаете из раздела «Настраиваемые ограничения», настраиваемые ограничения используют карты. Когда Postfix находит

ключ в карте ограничений, выполняется значение (действие), соответствующее этому ключу. Карта может выглядеть так:

```
10.0.0.1          PERMIT Private IP from VPN transfer tunnel
172.16.0          REJECT Private IP address cannot come from outside
168.100.1.3       DUNNO
192.0.34.166      OK
```

Приведенная карта содержит четыре различных действия для настраиваемых ограничений: `PERMIT`, `REJECT`, `DUNNO` и `OK`. Эти значения указывают Postfix, как следует поступить с клиентом, отправителем или получателем. Приведем лишь самые распространенные действия (полный перечень вы сможете найти на странице руководства access(5)):

`OK`

Никаких возражений против клиента и сообщения. Postfix прекращает оценку ограничений в текущем наборе ограничений и переходит к следующему набору.

`PERMIT`

Аналог `OK`.

`REJECT`

Незамедлительно отвергает сообщение, игнорируя все последующие ограничения. Сообщение отвергается.

`DUNNO`

Прекращает оценку текущего ограничения, но приступает к следующему ограничению текущего набора ограничений.

Порядок ограничений внутри набора очень важен, т. к. первое совпадение, возвращающее `OK` или `REJECT`, сразу же останавливает оценку ограничений текущего набора (при этом действие `REJECT` означает, что клиент, получатель или отправитель безоговорочно отвергается). Postfix читает и применяет ограничения сверху вниз или, если вы пишете их в одну строку, слева направо. Для сложных ограничений проще использовать многострочную форму: представьте себе, как бы вы читали такое ограничение, если бы оно было записано в одну строку!

```
smtpd_recipient_restrictions =
  check_recipient_access hash:/etc/postfix/recipients_restrictions,
  permit_sasl_authenticated,
  permit_mynetworks,
  reject_unauth_destination,
  reject_unauth_pipelining,
  reject_rbl_client relays.ordb.org
  permit
```

На рис. 7.3 изображен процесс оценки ограничений и показаны действия для всех четырех значений.

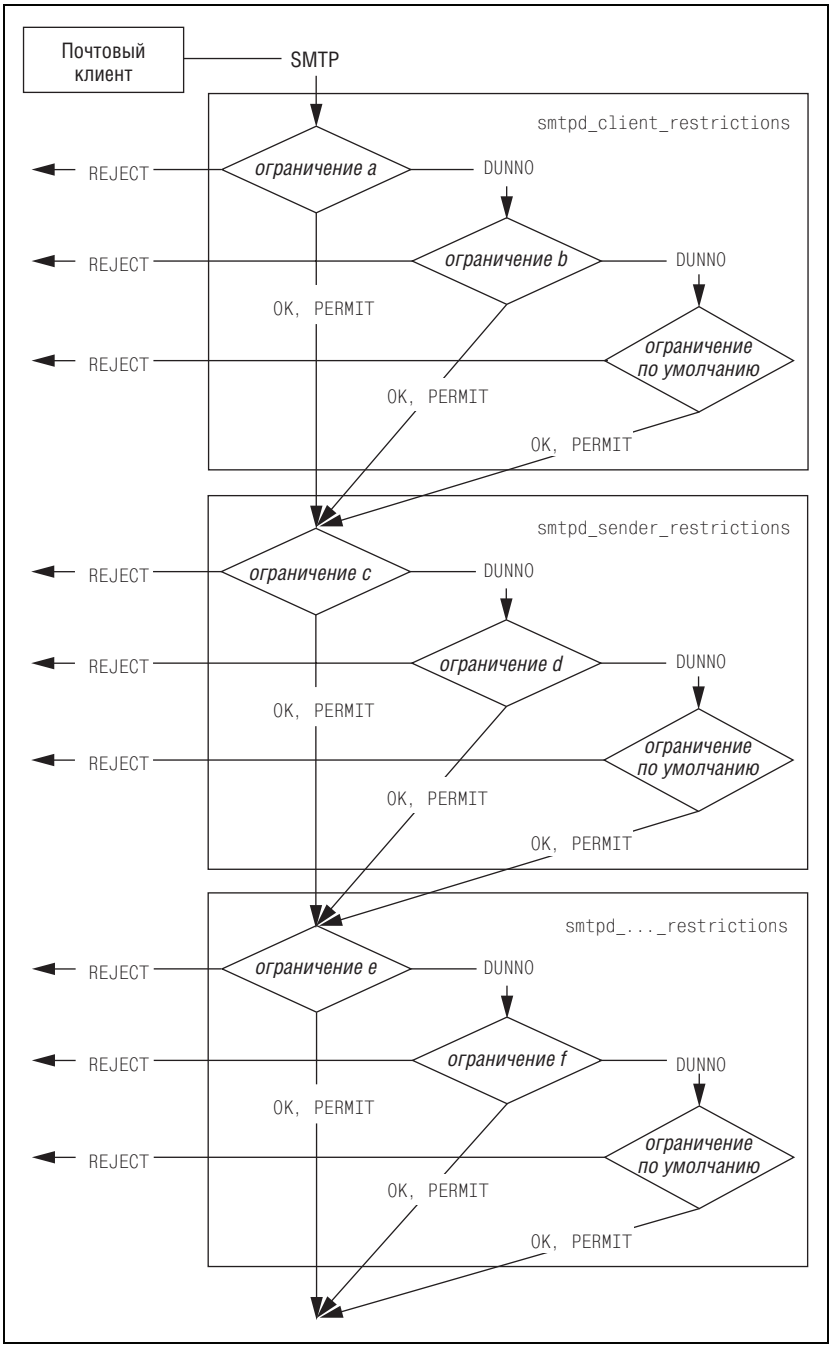

Рис. 7.3. *Процесс оценки ограничений*

Замедление плохих клиентов

Для любого клиента, вызывающего много ошибок при общении с `smtpd` (например, инициирующего `REJECT` в ограничении или приводящего к синтаксической ошибке в аргументах), `smtpd` делает небольшую паузу, прежде чем принимать последующие команды в том же сеансе. Такое поведение служит защитой от некорректно работающего клиентского программного обеспечения.

Существует ряд настроек, регулирующих подобные случаи. Параметр `smtpd_error_sleep_time` определяет длительность паузы (в секундах) после каждой ошибки (по умолчанию – одна секунда). Параметр `smtpd_soft_error_limit` служит своего рода «замедлителем»: когда удаленный SMTP-клиент делает несколько ошибок, SMTP-сервер Postfix может добавлять дополнительные задержки перед ответом. Наконец, вы можете прервать сеанс, используя параметр `smtpd_hard_error_limit`.

Эти три параметра работают вместе следующим образом:

- Если клиент вызывает ошибки и общее количество ошибок в текущем SMTP-сеансе меньше или равно значению параметра `smtpd_soft_error_limit`, то каждая ошибка приводит к задержке на значение `smtpd_error_sleep_time`.
- Если клиент вызывает ошибки и общее количество ошибок в SMTP-сеансе превышает значение `smtpd_soft_error_limit`, то каждая ошибка вызывает дополнительную задержку на количество секунд, равное числу ошибок сверх значения параметра `smtpd_soft_error_limit`.
- Если количество ошибок клиента превышает значение `smtpd_hard_error_limit`, то Postfix завершает сеанс.

Например, давайте представим, что параметры определены так:

```
smtpd_soft_error_limit = 5
smtpd_hard_error_limit = 10
smtpd_error_sleep_time = 1s
```

Если клиент вызвал 11 ошибок в одном сеансе, то Postfix делает паузы на 1, 1, 1, 1, 1, 2, 3, 4, 5, и 6 секунд соответственно, а после 11-й ошибки сервер отключается.

Классы ограничений

Класс ограничений – это специальная форма триггера ограничений, который не предопределен и не связан ни с каким конкретным этапом SMTP-взаимодействия. Вы определяете их по мере необходимости и инициируете, ссылаясь на них в картах настраиваемых ограничений.

Пусть, например, у вас есть карта настраиваемого ограничения для проверки адресов отправителей конвертов, и вы хотите инициировать другой набор ограничений в случае, если отправитель конверта соответствует `example.com`. В этом случае вы можете поместить этот новый

набор ограничений в новый класс с именем `check_if_example.com_sender`. Сначала объявляем новый класс в файле `main.cf`.

```
smtpd_restriction_classes =
   check_if_example.com_sender
```

Теперь, также внутри `main.cf`, добавляем в новый класс несколько ограничений:

```
check_if_example.com_sender =
   check_sender_access hash:/etc/postfix/bounces
   check_sender_access hash:/etc/postfix/valid_example.com_senders
   check_sender_access regexp:/etc/postfix/nice_reject.regexp
```

Эти новые ограничения анализируют отправителя конверта (но здесь могли бы использоваться любые ограничения, подходящие для текущего этапа SMTP-диалога).

Не беспокойтесь о картах для данных ограничений – далее вы увидите, как их определить. Однако нам до сих пор не хватает чего-то важного... Как инициировать `check_if_example.com_sender`?

Для этого необходимо наличие ограничения `check_sender_access` в наборе `smtpd_*_restrictions`. Давайте будем считать, что у вас уже есть такой набор ограничений с картой, которая разрешает отправителей из `foo.com` и отвергает отправителей из `bar.org` (возможные действия описаны в разделе «Влияние действий на оценку ограничений» ранее в этой главе):

```
foo.com                    OK
bar.org                    REJECT
```

Для добавления нового класса ограничений расширьте карту следующим образом:

```
foo.com                    OK
bar.org                    REJECT
example.com                check_if_example.com_sender
```

Как видите, главное в использовании классов ограничений – это найти правильное место для их вставки в карту настраиваемого ограничения.

8

Использование ограничений на передачу сообщений

Спам – это война. Правила RFC не действуют.

– Витсе Венема

Ограничения управляют потоком сообщений, принимая решения на основе информации, передаваемой клиентом в процессе SMTP-диалога. Количество случаев, в которых можно применять ограничения, огромно, поэтому перечислять все ограничения и все их возможные параметры в этой главе мы не будем, а опишем ситуации, которые часто встречаются в списках рассылки Postfix и в повседневной работе. Для каждой ситуации будут подробно рассмотрены ограничения и их параметры, с тем чтобы вы поняли, как их реализовывать и почему они реализованы именно так.

Создание и тестирование ограничений

Прежде чем приступать к изменению ограничений по умолчанию, вы должны знать, что именно вы пытаетесь ограничить. Если вы просто включаете или выключаете булево ограничение, в этом нет ничего сложного, но если речь идет об отказе в приеме сообщений хостам, скрывающим свое происхождение, задача несколько усложняется.

В списке рассылки Postfix популярно следующее изречение: «Журнал твой – друг твой». Может быть, вам нелегко представить журнал в роли друга, но он на самом деле очень полезен при сборе информации для ограничения потока сообщений. Дело в том, что почтовый журнал хранит большую часть информации, необходимой для создания эффективных ограничений. Давайте посмотрим на записи журнала для входящего сообщения:

```
Apr 14 21:14:48 mail postfix/smtpd[31840]: 4F2A643F30:
  client=unknown[172.16.0.1] ❶
Apr 14 21:14:48 mail postfix/cleanup[31842]: 4F2A643F30:
  message-id=<002101c42254$792c2530$010010ac@stateofmind.de> ❷
Apr 14 21:14:48 mail postfix/nqmgr[31836]: 4F2A643F30:
  from=<test@example.com>, ❸
  size=666, nrcpt=1 ❹ (queue active)
Apr 14 21:14:48 mail postfix/smtpd[31840]: disconnect from unknown[172.16.0.1]
Apr 14 21:14:48 mail postfix/smtp[31844]: 4F2A643F30:
                                          to=<p@state-of-mind.de>, ❺
  relay=mail.state-of-mind.de[212.14.92.89], ❻
  delay=0, status=sent (250 Ok: queued as 97E70E1C65) ❼
```

Сообщение состоит из следующих частей:

❶ Клиент (IP-адрес и имя хоста), который доставил сообщение.

❷ Заголовок Message-Id.

❸ Отправитель конверта (команда `MAIL FROM` в SMTP-диалоге).

❹ Количество получателей.

❺ Получатель (получатели) конверта (команда `RCPT TO` в SMTP-диалоге).

❻ Где побывало сообщение.

❼ Идентификатор очереди, который присвоил сообщению удаленный сервер Postfix.

Если вам необходимо ограничить передачу сообщения и нужны данные для того, чтобы разобраться, с чем вы имеете дело, журнал – это то место, где вы можете лучше узнать своего «противника».

Моделирование работы ограничений

С первой попытки редко удается найти удачный набор ограничений. Обычно, чтобы получить желаемый результат, вы проходите через последовательность проб и ошибок. Для проверки ваших ограничений вам необходимы сообщения, к которым их можно было бы применить, при этом вполне вероятно, что в вашем распоряжении нет тестовой машины и вы создаете свои ограничения на рабочем сервере. К сожалению, это порождает риск получения ложноположительных результатов и потери важных сообщений.

Для решения проблемы Postfix предлагает для тестирования ограничений параметр `warn_if_reject`, который аналогичен действию `WARN` в проверках. Если поставить этот параметр перед ограничением, которое вы хотите проверить, то Postfix будет записывать в журнал результаты действия ограничения, но сообщения отвергать не будет. Вот как следует использовать данный параметр для проверки ограничения `reject_unknown_sender_domain`:

```
smtpd_recipient_restrictions =
   permit_mynetworks
```

```
    reject_unauth_destination
    warn_if_reject reject_unknown_sender_domain
    permit
```

Как только параметр установлен, в журнале сообщений появляются записи о мнимом отклонении сообщений:

```
Jun 25 16:10:52 mail postfix/smtpd[32511]: 8075015C02F: reject_warning: RCPT
   from sccrmhc11.comcast.net[204.127.202.55]: 550 <DickinsL@newfaces.gr>:
   Sender address rejected: Domain not found; from=<DickinsL@newfaces.gr>
   to=<example@charite.de> proto=ESMTP helo=<sccrmhc11.attbi.com>
```

После того как вы убедитесь в том, что ограничения работают, можно удалить параметр `warn_if_reject` из ограничения. Последующие журнальные записи сообщат вам об успешном отклонении сообщений:

```
Jun 25 16:11:23 mail postfix/smtpd[32511]: 8075015C02F: reject: RCPT from
   sccrmhc11.comcast.net[204.127.202.55]: 550 <DickinsL@newfaces.gr>: Sender
   address rejected: Domain not found; from=<DickinsL@newfaces.gr>
   to=<recipient@example.com> proto=ESMTP helo=<sccrmhc11.attbi.com>
```

Немедленное введение ограничений в действие

Postfix состоит из множества различных демонов, которые при запуске загружают свои конфигурации. Некоторые демоны работают лишь небольшой период времени и завершают свою работу, чтобы не создавать излишнюю нагрузку. Однако есть демоны, которые *не* перезапускаются до тех пор, пока вы не укажете Postfix на необходимость их перезапуска.

Эти долго работающие демоны, `qmgr` и `nqmgr` (он назывался `nqmgr` только в старых версиях, в новых версиях Postfix по умолчанию используется новый диспетчер очередей, который здесь носит имя `qmgr`, при этом старый диспетчер очередей называют `oqmgr`), играют важную роль в ограничении потока электронной почты и *не* замечают изменений конфигурации до тех пор, пока вся система не будет перезапущена или вы не вмешаетесь вручную. Поэтому вам необходимо помнить о том, что каждый раз, когда вы изменяете файл `main.cf` или `master.cf`, необходимо выполнить команду `postfix reload`, чтобы диспетчер очередей перезагрузил конфигурацию.

Примечание

В принципе, со временем изменения будут замечены, т. к. демоны умирают и перезапускаются по достижении значения `max_use` (конечно, за исключением `qmgr`, который никогда не умирает). Изменения параметров `qmgr` требуют обязательного выполнения команды `postfix reload`. Если же разрешить принятие изменений «со временем», то может получиться так, что одни демоны будут использовать старую конфигурацию, а другие – новую, что вряд ли будет хорошо.

Ограничения по умолчанию

Postfix поставляется с надежным набором ограничений по умолчанию, которые не дают вашему компьютеру превратиться в открытый почтовый сервер.[1] Для того чтобы познакомиться с ограничениями по умолчанию, необходимо попросить `postconf` вывести эти значения для `smtpd_recipient_restrictions`:

```
# postconf -d smtpd_recipient_restrictions
smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination
```

Postfix оценивает ограничения в том порядке, в котором они приведены. В данном случае, если клиент хочет переслать сообщение, Postfix проверяет, исходит ли соединение от хоста, указанного в `mynetworks`. Если это так (ограничение `permit_mynetworks` оценивается как `OK`), то Postfix принимает сообщение для доставки.

Если клиент не относится к сети из `mynetworks`, Postfix оценивает `reject_unauth_destination`. Это ограничение отклоняет попытки пересылки, проверяя, относится ли получатель сообщения к доменам места назначения и доменам пересылки, указанным в ваших настройках. Если получатель не относится к этим доменам, то `reject_unauth_destination` возвращает `REJECT`, и Postfix сообщает клиенту о том, что пересылка невозможна.

Если место назначения сообщения входит в зону ответственности Postfix, ограничение `reject_unauth_destination` возвращает `DUNNO`, и Postfix переходит к оценке следующего ограничения. Если же в списке больше нет ограничений, то Postfix считает, что по умолчанию подразумевается `permit`, и принимает сообщение.

Эти два ограничения – та основа, которая защищает сервер от превращения в открытый почтовый сервер, но они не защищают пользователей от спама и не заставляют клиенты вести себя корректно. В оставшейся части главы будет показано, как сделать ограничения более строгими.

Требование соответствия RFC

Требование надлежащего поведения (в соответствии с RFC) от локальных и удаленных клиентов – это первый шаг к тому, чтобы управляемый вами корабль не дал течи. Это не только гарантирует, что ваш почтовый сервер передает другим почтовым серверам корректные со-

[1] Англоязычный термин «open relay» здесь и далее вернее переводить как «открытый ретранслятор», и поскольку в тексте встречаются оба термина, просто помните, что «открытый почтовый сервер» и «открытый ретранслятор» – это одно и то же. И одинаково неприемлемы в качестве рабочей (не тестовой) системы. – *Примеч. науч. ред.*

общения, но и заставляет удаленные клиенты вести себя правильно. Такое требование полезно для защиты от спамеров, которые всегда спешат, не следуют правилам и фальсифицируют идентификационную информацию.

В этом разделе будет показано, как накладывать ограничения на имя хоста, отправителя и получателя конверта, чтобы добиться соответствия RFC.

Примечание

Приведенные ограничения будут использованы в файле `main.cf` не в том порядке, в котором они поясняются здесь. Это сделано специально, и вы увидите, зачем, в разделе «Порядок обработки RFC-ограничений» ниже в этой главе. Пока же просто добавляйте ограничения в том порядке, в котором они появляются в листингах примеров.

Ограничения на имя хоста в команде HELO/EHLO

Хорошей отправной точкой будет требование Postfix к клиентам относительно их корректного приветствия, если они хотят, чтобы их сообщения были отправлены на ваш сервер или переданы через него. Существует целый ряд ограничений, которые могут быть наложены на `HELO/EHLO`-часть SMTP-диалога, начиная с простого требования отправки имени хоста до требования отправки достоверного имени хоста.

Требование указания имени хоста

Параметр `smtpd_helo_required` требует, чтобы все клиенты, открывающие SMTP-соединение, выполняли команду `HELO` или `EHLO`. Такого обязательного приветствия требует как RFC 821 (*ftp://ftp.rfc-editor.org/in-notes/rfc821.txt*), так и RFC 2821 (*ftp://ftp.rfc-editor.org/in-notes/rfc2821.txt*), но Postfix по умолчанию устанавливает этот параметр в значение `no`. Для того чтобы включить данное требование, добавьте в файл `main.cf` такую строку:

```
smtpd_helo_required = yes
```

После перезагрузки конфигурации Postfix будет отклонять сообщения любого клиента, который не представится должным образом. Для того чтобы проверить это, подключитесь к вашему серверу и попробуйте инициировать передачу сообщения без команды `HELO`. Вот как отреагирует Postfix, требующий указания имени хоста:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
MAIL FROM: <sender@example.com>
503 Error: send HELO/EHLO first
QUIT
221 Bye
```

Требование указания полностью определенного доменного имени хоста

Команда HELO/EHLO – это, конечно, хорошо, но клиенты также должны передать вместе с приветствием полное имя хоста (например, HELO client.example.com). Более того, RFC требуют, чтобы указывалось полностью определенное доменное имя хоста (FQDN).

Примечание

FQDN не обязательно присутствует в записях DNS.

Postfix будет отвергать сообщения от любого клиента, не предоставившего полностью определенное доменное имя хоста, если вы установите параметр reject_non_fqdn_hostname внутри ограничения smtpd_recipient_restrictions.

Предупреждение

Будьте аккуратны с этим ограничением. Некоторые почтовые клиенты, такие как Microsoft Outlook, по умолчанию используют только локальную часть имени (например, client), если только вы не настроите операционную систему так, чтобы она передавала своим приложениям полностью определенное доменное имя.

Когда вы добавите параметр reject_non_fqdn_hostname в список smtpd_recipient_restrictions, он будет иметь такой вид в файле main.cf:

```
smtpd_recipient_restrictions =
  permit_mynetworks
  reject_unauth_destination
  reject_non_fqdn_hostname
  permit
```

Проверьте ограничение, подключившись к своему почтовому серверу и указав простое имя хоста, например:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
HELO client
250 mail.example.com
MAIL FROM: <sender@example.com>
250 Ok
RCPT TO: <recipient@example.com>
504 <client>: Helo command rejected: need fully-qualified hostname
QUIT
221 Bye
```

Требования к символам, составляющим имя хоста

RFC определяет, что не только имена хостов, отправляемые с командой HELO/EHLO, должны быть полностью определенными доменными именами, но и используемые для составления таких имен символы

должны подчиняться требованиям к построению имени хоста. Корректное доменное имя должно включать в себя как минимум следующие элементы:

- Домен верхнего уровня, такой как `com`
- Имя домена, например `example`
- Точку (.), разделяющую домен верхнего уровня и доменное имя

Любое другое имя хоста, скорее всего, не будет корректно разрешено, что затруднит (или даже сделает невозможным) взаимодействие между сервером и клиентом. Используя параметр `reject_invalid_hostname` в списке `smtpd_recipient_restrictions`, вы можете указать Postfix, что не следует общаться с такими клиентами. Вот пример, показывающий, куда можно вставить этот параметр:

```
smtpd_recipient_restrictions =
  permit_mynetworks
  reject_unauth_destination
  reject_non_fqdn_hostname
  reject_invalid_hostname
  permit
```

Как и прежде, проверяем ограничение, подключаясь к своему почтовому серверу от удаленного хоста и предоставляя недействительное имя хоста. В данном примере клиент представляется как «.».

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
HELO .
250-mail.example.com
MAIL FROM:<sender@example.com>
250 Ok
RCPT TO:<recipient@example.com>
501 <.>: Helo command rejected: Invalid name
QUIT
221 Bye
```

Ограничения на отправителя конверта

Доменная часть имени отправителя конверта должна содержать полностью определенное доменное имя (FQDN), и конверт должен принадлежать к существующему домену. Такие отправители конверта, как `sender` и `sender@example`, не указывают полностью определенное доменное имя. Примером полного имени отправителя конверта может быть `sender@example.com`. Неполные адреса могут вызвать путаницу, т. к. адрес отправителя выглядит так, как будто сообщение создано на данном сервере. Возможны два варианта нежелательного развития событий:

- Агент передачи сообщений, который должен возвратить сообщение с неполным именем отправителя конверта, будет возвращать его локальным пользователям. Возврат не дойдет до исходного отправителя.

- Postfix может попытаться «исправить» неправильный адрес, что только усложнит дело. Так как Postfix знает, что отправитель конверта должен иметь полностью определенное доменное имя, он запустит демон `trivial-rewrite` для канонизации адреса путем добавления `$myorigin` для отправителя `sender` (в результате получится `sender@$myorigin`) и `$mydomain` — для `sender@example` (получится `sender@example.$mydomain`). Следовательно, отправитель конверта для сообщений, полученных от удаленного сервера, будет абсолютно неверным.

Для того чтобы избежать подобных ситуаций, добавьте параметр `reject_non_fqdn_sender` в список `smtpd_recipient_restrictions`, например:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_sender
  permit_mynetworks
  reject_unauth_destination
  reject_non_fqdn_hostname
  reject_invalid_hostname
  permit
```

Проверьте ограничение, подключившись с удаленной машины к вашему почтовому серверу и указав некорректное имя отправителя конверта. Покажем, как ограничения заставят Postfix отклонить сообщения такого отправителя:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
HELO client.example.com
250 mail.example.com
MAIL FROM: <sender>
250 Ok
RCPT TO: <recipient@example.com>
504 <sender>: Sender address rejected: need fully-qualified address
```

Сообщение из несуществующих доменов

Ответственный почтовый сервер не принимает сообщения от получателей, домены которых не существуют, т. к. в случае невозможности доставки сообщения он не сможет сообщить об этом отправителю. В других конфигурациях возникает двойной возврат, как только агент передачи сообщений пытается уведомить отправителя, и в конце концов сообщение с несуществующим доменом отправителя окажется в почтовом ящике администратора почтовой системы.

Примечание

Почтовым серверам приходится иметь дело с несуществующими доменами, так как пользователи иногда неправильно набирают свои почтовые адреса при настройке своих почтовых клиентов; кроме того, несуществующие домены используют спамеры, чтобы скрыть фактический источник своих сообщений.

Для защиты получателей и администраторов почтовой системы от двойного возврата и неправильно составленных сообщений добавьте параметр `reject_unknown_sender_domain` в список `smtpd_recipient_restrictions`, например:

```
smtpd_recipient_restrictions =
  reject_unknown_sender_domain
  permit_mynetworks
  reject_unauth_destination
  reject_non_fqdn_hostname
  reject_invalid_hostname
  permit
```

Следующий пример показывает, как можно проверить ограничение (ищем код ошибки 450, который Postfix отправляет в качестве ответа команде `MAIL FROM`):

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
HELO client.example.com
250 mail.example.com
MAIL FROM: <sender@domain.invalid>
250 Ok
RCPT TO: <recipient@example.com>
450 <sender@domain.invalid>: Sender address rejected: Domain not found
```

Ограничения на получателя конверта

В качестве последнего действия по приведению входящих соединений в соответствие RFC вы можете отвергать сообщения, в которых для получателя конверта указан несуществующий домен или пользователь.

Почтовый сервер не должен принимать никакие сообщения для несуществующего домена, т. к. их невозможно доставить. Если почтовый сервер примет сообщение, а затем вернет его, то пользователь может подумать, что у сервера какие-то проблемы, ведь изначально сообщение было принято.

Если настроить почтовый сервер так, чтобы сообщения в несуществующие домены отклонялись, то ваши проблемы станут проблемами того клиента или пользователя, которые отправили сообщение. Для введения такого ограничения используйте параметр `reject_unknown_recipient_domain` в списке `smtpd_recipient_restrictions`, например:

```
smtpd_recipient_restrictions =
  reject_unknown_recipient_domain
  permit_mynetworks
  reject_unauth_destination
  reject_non_fqdn_hostname
  reject_invalid_hostname
  permit
```

Как обычно, вы можете протестировать ограничение, отправив письмо в несуществующий домен получателя при ручном подключении к серверу. Покажем на примере, как Postfix отвергает сообщение из-за того, что домен `invalid.domain` не существует:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
HELO client.example.com
250 mail.example.com
MAIL FROM: <sender@example.com>
250 Ok
RCPT TO: <recipient@domain.invalid>
450 <recipient@domain.invalid>: Recipient address rejected: Domain not found
```

Сообщения неизвестным получателям

Вы можете настроить Postfix так, чтобы сообщения для неизвестного пользователя из вашего домена доставлялись администратору почтовой системы. На первый взгляд такая идея кажется весьма удачной, т. к. администратор может изучить сообщение и при наличии возможности доставить его вручную.

Но несмотря на то, что в теории подобная конфигурация могла бы обеспечить идеальное обслуживание клиентов, использование адресата по умолчанию может привести к DoS-атакам (Denial-of-service – отказ в обслуживании) на ваш сервер, как только он станет целью для спамера или червя, использующего *атаку по словарю*. Такая атака заключается в попытке отправить сообщение существующим получателям за счет рассылки сообщений по адресам, составленным из всех возможных сочетаний букв. Например, атакующий может начать с адреса `aa@yourdomain.com`, затем попробовать `ab@yourdomain.com` и так пройти через все двухбуквенные комбинации вплоть до `zz@yourdomain.com`.

Дело не только в сложности отделения законных сообщений от сообщений, созданных в процессе подобной атаки, но и в том, что сервер подвергается риску в связи с исчерпанием пропускной способности канала, производительности процессора, памяти и дискового пространства, и в результате сдается и останавливает обслуживание запросов на передачу сообщений. Например, вирус Sobig.F привел к перегрузке многих почтовых серверов в августе 2003 года.

Помните, что для Postfix приоритетом является надежность обслуживания. Надежность подразумевает согласованность, и поэтому сервер отвергает почту, адресованную неизвестным пользователям, по умолчанию без какого-либо ручного вмешательства. Это замечательно для автономного сервера Postfix, но полезно и для сервера Postfix, работающего на интеллектуальном хосте, который защищает остальные почтовые серверы.

Postfix определяет корректность адресов получателей, сверяясь с картами. Существуют два параметра конфигурации, которые указывают

Postfix, где следует искать такую информацию: local_recipient_maps и relay_recipient_maps. В обоих параметрах указывается одна или несколько карт, содержащих действительных получателей. Параметр local_recipient_maps определяет действительных локальных получателей, как показано в примере, где получатели определены в файле паролей UNIX и картах псевдонимов:

```
# postconf -d local_recipient_maps
local_recipient_maps = proxy:unix:passwd.byname $alias_maps
```

В то же время параметр relay_recipient_maps определяет получателей, для которых Postfix пересылает сообщения к конечному месту назначения (серверу почтовых ящиков):

```
# postconf -d relay_recipient_maps
relay_recipient_maps = hash:/etc/postfix/relay_recipients
```

При использовании relay_recipient_maps позаботьтесь о том, чтобы Postfix знал всех действительных получателей в тех системах, куда он осуществляет пересылку. Если место назначения – это сервер Microsoft Exchange, обратитесь к главе 13 за сведениями о том, как можно извлечь карту пользователей.

Предупреждение

Использование luser_relay отменяет параметр local_recipient_maps, т. к. делает действительными всех локальных получателей. Аналогично запись с групповым именем catchall в списке virtual_alias_maps отменяет отклонение почты, адресованной несуществующим получателям, т. к. групповое имя делает действительными всех получателей. Например, следующая запись карты делает действительными всех получателей в домене example.com:

```
@example.com    catchall@localhost
```

Сообщения получателям с неполным именем

Неполностью указанный адрес, такой как recipient, содержит только локальную часть электронного адреса. С приемом таких сообщений для локальных пользователей на компьютере, получающем почту только для одного домена, проблем нет, но трудности появляются, если ваш почтовый сервер получает сообщения и для других доменов.

Есть указана одна лишь локальная часть, то остается слишком много простора для интерпретации почтового адреса.

Пусть, например, вы работаете интернет-провайдером для двух конкурирующих компаний, example.com и example.net. Если вы получите сообщение для получателя sales, куда оно будет отправлено? По какому адресу оно должно попасть – sales@example.com или sales@example.net, если один и тот же сервер обслуживает оба этих электронных адреса?

В силу вышесказанного вам следует отклонять сообщения с неполными адресами. Не принимайте на себя чужую ответственность. Подго-

товка сообщения для корректной доставки – это задача отправителя, и он должен определить получателя уникальным способом.

Примечание

Существует всего одно исключение: вы должны принимать сообщения для postmaster при неполностью указанном адресе. На адрес postmaster не накладываются вообще никакие ограничения, действующие для получателей (включая ограничения на отправителя, команду helo и клиент).

Postfix будет отвергать сообщения для получателей с неполным именем, если вы добавите параметр reject_non_fqdn_recipient в ваш список smtpd_recipient_restrictions, как в следующем примере:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_unknown_recipient_domain
  permit_mynetworks
  reject_unauth_destination
  reject_non_fqdn_hostname
  reject_invalid_hostname
  permit
```

Проверим ограничение, подключившись к своему почтовому серверу с удаленного компьютера и отправив сообщение с неполным адресом получателя. Для проверки работы ограничения достаточно будет такого сеанса:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
HELO client.example.com
250 mail.example.com
MAIL FROM: <sender@example.com>
250 Ok
RCPT To: <recipient>
504 <recipient>: Recipient address rejected: need fully-qualified address
```

Обеспечение соответствия RFC

Возможно, вы уже обратили внимание на то, что ограничения могут стать довольно сложными. И чем сложнее становятся ограничения, тем выше вероятность того, что среди них появится такое, которое приведет к неправильной работе вашей почтовой системы (или вообще сделает ее абсолютно бесполезной), отвергая сообщения, которые должны быть приняты при любых обстоятельствах. Последующие разделы покажут вам, как избежать непредумышленной блокировки некоторых или всех отправителей. Это важно, т. к. вы можете случайно исключить отправителей, которые могли бы сообщить о недостатках вашей конфигурации.

Пустое имя отправителя конверта

Во-первых, никогда не блокируйте пустого отправителя конверта (<>). Этот адрес принадлежит MAILER-DAEMON, почтовый сервер использует его при отправке возвратов и уведомлений о состоянии. Если заблокировать этот адрес, то удаленные серверы не смогут сообщить вашим пользователям о том, что с отправленными ими сообщениями возникли проблемы.

Предупреждение

Черные списки, такие как `dsn.rfc-ignorant.org`, приводят перечень почтовых серверов, которые категорически отказываются принимать почту от отправителей конвертов с пустым именем, так что использующие эти черные списки почтовые серверы не принимают почту от перечисленных там серверов (мы вернемся к этому вопросу в разделе «Отказ доменам отправителей из черных списков»).

Все, что вам нужно, – это рассматривать пустой адрес отправителя конверта как любой другой допустимый адрес и создать хорошие ограничения (противодействующие спаму) для защиты ваших получателей. Пусть ограничения выполняют свою работу, и если вы получите сообщение с пустым именем отправителя, примените его. В конце концов, любой адрес отправителя может оказаться подделкой...

Специальные учетные записи

На почтовом сервере имеются два адреса, для которых вы всегда должны принимать сообщения; они необходимы для того, чтобы работа почтового сервера соответствовала RFC:

`postmaster`

Всегда принимайте почту, адресованную администратору почтовой системы `postmaster`, – это центр обработки информации для вопросов, связанных с электронными сообщениями. Пользователи должны иметь возможность обратиться к администратору за помощью (см. RFC 2821 по адресу *http://www.rfc-editor.org/rfc/rfc2821.txt*).

`abuse`

Прием почты для адресата `abuse` гарантирует, что пользователи смогут уведомить вас о возможных почтовых злоупотреблениях, исходящих от вашего сервера (см. RFC 2142 по адресу *http://www.rfc-editor.org/rfc/rfc2142.txt*).

Дополнительно (но не обязательно) вы можете принимать сообщения для следующих адресатов, если поддерживаете соответствующие серверы (см. RFC 2142; *http://www.rfc-editor.org/rfc/rfc2142.txt*):

`webmaster`

Принимайте почту для `webmaster`, если у вас работает веб-сервер.

`hostmaster`

Принимайте почту для `hostmaster`, если у вас работает сервер имен.

Вы можете настроить прием сообщений для этих получателей, используя параметр `check_recipient_access` в сочетании с картой, такой как `/etc/postfix/roleaccount_exceptions`, где перечислены получатели, сообщения для которых должны быть приняты. Такая карта может выглядеть следующим образом (значение `OK` для каждого ключа карты сообщает Postfix о том, что следует принимать сообщения для данного получателя без учета ограничений для получателей):

```
# addresses that you must always accept
postmaster@    OK
abuse@         OK
# addresses that you should accept if you run DNS and WWW servers
hostmaster@    OK
webmaster@     OK
```

После создания этого файла преобразуйте его в карту командой `postmap hash:/etc/postfix/roleaccount_exceptions`. Затем укажите карту в качестве значения параметра `check_recipient_access` в списке ограничений файла `main.cf`, например:

```
smtpd_recipient_restrictions =
   reject_non_fqdn_recipient
   reject_non_fqdn_sender
   reject_unknown_sender_domain
   reject_unknown_recipient_domain
   permit_mynetworks
   reject_unauth_destination
   check_recipient_access hash:/etc/postfix/roleaccount_exceptions
   permit
```

После перезагрузки конфигурации вы можете спокойно переходить к созданию более сложных правил. Карта с исключениями запрашивается после того, как Postfix проводит проверки на неавторизованную пересылку, так что использование адреса `postmaster@` будет безопасным.

Порядок обработки RFC-ограничений

Возможно, вы обратили внимание на то, что параметры, добавляемые в список `smtpd_recipient_restrictions` в предыдущих разделах, указывались не в том порядке, как сами разделы. Это объясняется тем, что параметры ограничений могут влиять один на другой и мешать друг другу, если будут указаны не в нужном порядке. Давайте, например, посмотрим на такой список:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_non_fqdn_sender
```

```
reject_unknown_sender_domain
reject_unknown_recipient_domain
permit_mynetworks
reject_unauth_destination
check_recipient_access hash:/etc/postfix/roleaccount_exceptions
reject_non_fqdn_hostname
reject_invalid_hostname
permit
```

Параметр `permit_mynetworks` обозначает важную границу между клиентами вашей внутренней сети и внешними клиентами. Параметры, заданные выше этого элемента (включая его самого), относятся как к внутренним, так и к внешним клиентам, в то время как параметры ниже `permit_mynetworks` применяются только к внешним клиентам.

Параметры, предшествующие `permit_mynetworks`, требуют базового соблюдения RFC от всех клиентов как внутри вашей сети, так и вне ее.

Параметр `reject_unauth_destination` не дает вашему серверу превратиться в открытый почтовый сервер. Лучше не указывать никакие параметры, разрешающие пересылку сообщений, пока не задан параметр `permit_mynetworks`. Далее как можно раньше следует указать `reject_unauth_destination`, чтобы быть уверенным в том, что неавторизованный хост никоим образом не сможет использовать ваш сервер как открытый почтовый сервер.

Проверка на SMTP-аутентификацию должна находиться между `reject_unauth_destination` и `permit_mynetworks`. Затем, прежде чем указывать еще какие-то параметры отказа от сообщений, используйте параметр `check_recipient_access` для разрешения безусловной доставки специальным почтовым ящикам вашей системы.

Наконец, после отражения возможных попыток возврата спама множественным получателям и отказа в приеме сообщениям с фальшивыми именами хостов получателей конверта вы можете принимать сообщения, используя параметр `permit`.

Меры борьбы со спамом

Спамерам необходимо замаскировать источник своих сообщений, если они не хотят судебного преследования.[1] Обычно они подделывают адрес отправителя конверта или пытаются усыпить бдительность принимающего сервера, сообщая ему, что их клиенту можно доверять – как

[1] По состоянию на март 2008 года законодательство РФ не предусматривает, к сожалению, никакой ответственности за рассылку спама; единственным ограничением для спамера является типовой договор клиента с провайдером, в котором клиент обычно обязуется не использовать предоставленный доступ в Интернет для осуществления массовых непрошеных рассылок. – *Примеч. науч. ред.*

будто он принадлежит к локальной сети. Ограничения могут проверить и отклонить такие сообщения. Более того, они могут запрашивать черные списки, в которых перечислены спамеры и другие адресаты, сообщения от которых вы не хотите принимать. В этом разделе будет показано, как провести такие ограничения в жизнь.

Предотвращение явных фальсификаций

Некоторые спам-программы пытаются скрыть источник сообщения, используя имя хоста вашего почтового сервера как свое собственное в приветствии HELO/EHLO. Postfix воспринимает ситуацию как парадоксальную, ведь единственный хост, который может использовать имя хоста сервера, – это сам сервер. Однако Postfix никогда не стал бы подключаться к своему демону smtpd для отправки почты самому себе, если только не была сделана ошибка конфигурации, приведшая к возникновению петли.

Добавление ограничений после строки permit_mynetworks сделает их применимыми только к внешним клиентам, но не к прокси-фильтрам или локальным клиентам с неполной реализацией SMTP.

Поэтому вы можете отказывать в SMTP-соединении любому клиенту, который приветствует ваш почтовый сервер с именем хоста этого сервера. Для этого сначала создайте файл карты с именем /etc/postfix/helo_checks, содержащий различные вариации имени вашего хоста. Приведем несколько примеров для имени хоста, IP-адреса хоста и IP-адреса в скобках, которые не должны использоваться внешними клиентами:

```
/^mail\.example\.com$/        550 Don't use my hostname
/^192\.0\.34\.166$/           550 Don't use my IP address
/^\[192\.0\.34\.166\]$/       550 Don't use my IP address
```

Документ RFC 2821 указывает, что сам по себе IP-адрес не является разрешенным аргументом для команды HELO. IP-адрес разрешен, если он задан в форме [ipv4address] (в квадратных скобках) или как IPv6-адрес, [ipv6:ipv6address], опять-таки в квадратных скобках. Чтобы соблюсти все формальности и отказать в обслуживании клиентам, которые отправляют IP-адрес без квадратных скобок, добавьте такую строку:

```
/^[0-9.]+$/                   550 Your client is not RFC 2821 compliant
```

Для того чтобы привести карту в действие, укажите ее (и ее тип) как аргумент для параметра check_helo_access в списке smtpd_recipient_restrictions, например:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_non_fqdn_sender
  reject_unknown_sender_domain
  reject_unknown_recipient_domain
```

```
permit_mynetworks
reject_unauth_destination
check_recipient_access hash:/etc/postfix/roleaccount_exceptions
reject_non_fqdn_hostname
reject_invalid_hostname
check_helo_access pcre:/etc/postfix/helo_checks
permit
```

Для проверки ограничения подключитесь к своему почтовому серверу и укажите собственное имя в приветствии HELO. Вы должны получить отказ, как показано в примере:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
HELO mail.example.com
250 mail.example.com
MAIL FROM: <sender@example.com>
250 Ok
RCPT TO: <recipient@example.com>
550 <mail.example.com>: Helo command rejected: Don't use my hostname
QUIT
221 Bye
```

Фиктивные записи сервера имен

Postfix может отвергать сообщения, если очевидно, что записи сервера имен для домена HELO, доменов отправителя и получателя отсутствуют или не позволяют обеспечить корректную передачу сообщения. Приведем ряд обстоятельств, которые могут показаться подозрительными в записях DNS:

Фальшивые сети

Некоторые почтовые серверы утверждают, что относятся к сетям, недостижимым для Postfix, в том числе неиспользуемым вами частным сетям (см. RFC 1918, *ftp://ftp.rfc-editor.org/in-notes/rfc1918.txt*), сети обратной связи (loopback), широковещательным сетям или сетям многоадресной рассылки.

Пристанища спамеров

Пристанища спамеров (spam havens) – это сети, про которые известно, что они принадлежат спамерам или предоставляют услуги спамерам. Можно отклонять все сообщения из таких доменов. Информацию о пристанищах спамеров и их деятельности вы можете найти в ROKSO (Register of Known Spam Operations – список известных спамеров, *http://www.spamhaus.org/rokso/index.lasso*).

«Безразличные» агенты передачи сообщений

«Безразличные» (wildcard) агенты передачи сообщений заявляют, что они ответственны за все домены, даже за несуществующие. Казалось бы, это не должно стать проблемой, ведь вы можете отказать в доступе в случае неизвестных доменов получателя и отправителя.

К сожалению, некоторые регистраторы доменов прониклись гениальной идеей перенаправлять неизвестные доменные имена в свой собственный домен. В результате неизвестные домены получают действительную A-запись, что делает бесполезными параметры ограничений `reject_unknown_sender_domain` и `reject_unknown_recipient_domain`.

Примечание

Первым доменным регистратором, перенаправившим неизвестные домены, стал VeriSign (*http://www.verisign.com*) в 2003 году. VeriSign злоупотребил своей властью над пространствами имен `.net` и `.com` и пересылал все несуществующие домены `.com` и `.net` на собственный сайт (`sitefinder.verisign.com`). Кроме того, VeriSign создал для неизвестных доменов собственную почтовую службу, что сделало невозможным отклонение сообщений из неизвестных доменов. Это явное приглашение для спамеров, и вы можете отклонять сообщения от «безразличных» агентов передачи сообщений, блокируя MX-хосты в таких доменах.

Во всех описанных случаях используются либо фальшивые записи сервера, либо поддержка спамеров. Для отклонения почты из таких доменов и сетей вы можете создать в файле `/etc/postfix/bogus_mx` карту, содержащую IP-адреса вместе с типом ответа, который вы хотите им дать (полный перечень таких ответов приведен в приложении C). Рассмотрим пример файла карты:

```
# bogus networks
0.0.0.0/8          550 Mail server in broadcast network
10.0.0.0/8         550 No route to your RFC 1918 network
127.0.0.0/8        550 Mail server in loopback network
224.0.0.0/4        550 Mail server in class D multicast network
192.168.0.0/16     550 No route to your RFC 1918 network
# spam havens
69.6.0.0/18        550 REJECT Listed on Register Of Known Spam Operations ❶
# wild-card MTA
64.94.110.11/32    550 REJECT VeriSign Domain wildcard ❷
```

❶ На момент написания книги эта сеть упоминалась в списке на сайте *spamhaus.org* (*http://www.spamhaus.org/sbl/sbl.lasso?query=SBL6636*) как известная своей спамерской деятельностью.

❷ На момент написания книги было известно, что этот хост действует как «безразличный» MTA.

Мы редактируем карту типа CIDR, которая относится к последовательному типу (см. главу 5), поэтому нам не нужно и мы не можем преобразовывать ее при помощи `postmap`. Postfix будет использовать файл «как есть». Просто добавьте параметр `check_sender_mx_access`, указав карту в качестве аргумента, в свой список `smtpd_recipient_restrictions`, например:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_non_fqdn_sender
  reject_unknown_sender_domain
  reject_unknown_recipient_domain
  permit_mynetworks
  reject_unauth_destination
  check_recipient_access hash:/etc/postfix/roleaccount_exceptions
  reject_non_fqdn_hostname
  reject_invalid_hostname
  check_helo_access pcre:/etc/postfix/helo_checks
  check_sender_mx_access cidr:/etc/postfix/bogus_mx
  permit
```

Ограничение вступит в силу после перезагрузки параметров. Его действие найдет отражение в почтовом журнале:

```
Sep 17 12:19:23 mail postfix/smtpd[3323]: A003D15C021: reject: RCPT from
  unknown[61.238.134.162]:
  554 <recipient@example.com>: Sender address rejected: VeriSign Domain
  wildcard;
  from=<alli.k_lacey_mq@joymail.com> to=<recipient@example.com> proto=ESMTP
  helo=<example.com>
```

Вы можете проверить IP при помощи команды `host`:

```
# host -t mx joymail.com
# host -t a joymail.com
joymail.com has address 64.94.110.11
```

Примечание

Этот домен действительно существует в настоящее время; судя по всему, он был зарегистрирован в октябре 2003 года.

Возврат множеству получателей

В разделе «Пустое имя отправителя конверта» вы узнали о том, что не следует блокировать сообщения с пустым именем отправителем конверта. Это правило имеет одно исключение – необходимо блокировать сообщения с пустым именем отправителя конверта, отправленные множеству получателей. Дело в том, что в настоящее время нет оснований для легальной рассылки уведомлений о состоянии многочисленным получателям, так что любые такие сообщения, вероятно, являются запрещенными.

Для отказа в приеме сообщениям с пустым именем отправителя конверта, предназначенным нескольким получателям, добавьте параметр `reject_multi_recipient_bounce` в свой список `smtpd_recipient_restrictions`. Этот параметр может быть добавлен практически в любое место списка ограничений; в данном примере он поставлен в `smtpd_data_restrictions`:

```
smtpd_data_restrictions =
  reject_multi_recipient_bounce
```

В документации говорится, что параметр `reject_multi_recipient_bounce` может надежно использоваться только в `smtpd_data_restrictions`, когда известны все получатели.

Проверить это ограничение вы, как и раньше, можете посредством ручного подключения к почтовому серверу. Передача пустого имени отправителя конверта и нескольких получателей приведет к отказу, как показано в следующем примере:

```
$ telnet localhost 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250 8BITMIME
MAIL FROM:<>
250 Ok
RCPT TO: <recipient1@example.com>
RCPT TO: <recipient2@example.com>
550 : Recipient address rejected: Multi-recipient bounce
QUIT
221 Bye
```

Использование черных списков DNS

DNS-сервер черных списков – это сервер, который сообщает вам о ресурсах (IP-адресах, отправителях конвертов и доменах), которым, вероятно, не стоит доверять. Правильно выбранные черные списки могут быть чрезвычайно полезны для блокирования почты, отправленной клиентами на ваш сервер. Однако неверный выбор черного списка может заставить ваш сервер отвергать почту, которую вы считаете допустимой. Обязательно проверяйте принципы составления черного списка, прежде чем им воспользоваться. Любой сайт, ведущий черный список, должен представлять перечень условий, по которым ресурс вносится в черный список; кроме того, должна быть опубликована процедура удаления из списка ресурса, который более не должен там находиться.

Если вы ищете черный список, то можете начать с сайта *dmoz.org* (*http://dmoz.org/Computers/Internet/Abuse/Spam/Blacklists*).

Предупреждение

Основой всех черных списков является служба доменных имен, т. е. Postfix должен выполнять поиск в DNS. Некэшированный поиск в DNS может занять около секунды, и в случае тайм-аута скорость, с которой сервер может прини-

мать сообщения, значительно уменьшается. Поэтому проверки по черным спискам достаточно затратны с точки зрения времени отклика. Вы должны использовать их в своем списке ограничений только в качестве последнего средства.

Отказ клиентам из черных списков

Вы можете отвергать занесенные в черный список клиенты при помощи списков DNSBL (DNS-based Blackhole List). В Postfix есть параметр `reject_rbl_client`, который принимает в качестве аргумента полное имя хоста сервера черных списков. Приведем пример использования этого параметра:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_non_fqdn_sender
  reject_unknown_sender_domain
  reject_unknown_recipient_domain
  permit_mynetworks
  reject_unauth_destination
  check_recipient_access hash:/etc/postfix/roleaccount_exceptions
  reject_non_fqdn_hostname
  reject_invalid_hostname
  check_helo_access pcre:/etc/postfix/helo_checks
  reject_rbl_client relays.ordb.org
  permit
```

Новый параметр вступит в силу после перезагрузки параметров.

Примечание

Для того чтобы проверить, упомянут ли клиент в списке DNSBL, измените на обратный порядок четырех октетов IP-адреса клиента (т. е. замените `a.b.c.d` на `d.c.b.a`), добавьте в конец `rbl.domain` (например, `relays.ordb.org`) и ищите в списке полученное значение. Если хост занесен в черный список, то вы получите ответ, указывающий на исходный IP-адрес, как в следующем примере:

```
$ host 2.0.0.127.relays.ordb.org
2.0.0.127.relays.ordb.org A 127.0.0.2
```

Многозначные результаты

Postfix может обработать дополнительную информацию, когда известно не только то, что хост занесен в черный список: возвращенный IP-адрес дает возможность определить, почему он туда занесен. Например, следующая конфигурация будет отклонять сообщения от любого хоста, который соответствует A-записи `127.0.0.2` в нашем воображаемом черном списке `domain.tld`:

```
reject_rbl_client domain.tld=127.0.0.2
```

Отказ отправителям из доменов черных списков

В дополнение к запрещению почты от определенных IP-адресов вы можете блокировать сообщения тех отправителей, домены которых занесены в черные списки. Такие списки называются RHSBL (Right-Hand-Side Blacklist – черный список правых частей)). Настройка Postfix для использования RHSBL требует выполнения тех же операций, что и для DNSBL. В качестве примера в этом разделе будет использоваться специальный список с сайта *dsn.rfc-ignorant.org*:

> **Программное заявление** *www.rfc-ignorant.org*:
>
> Мы ведем ряд списков (в настоящее время `dsn`, `abuse`, `postmaster`, `bogusmx` и `whois`), содержащих домены, администраторы которых решили не подчиняться требованиям RFC – стандартным правилам Сети.
>
> Важно отметить, что НИЧТО и НИКОГО не заставляет соответствовать RFC (Request for Comments – запрос на комментарии), однако возможность совместной работы, достигнутая в Сети, основывается на том, что у всех есть один и тот же свод правил и все ему следуют. Присутствие в списке означает только то, что для домена было решено не реализовывать условия, описанные в определенном RFC. Естественно, решение относительно того, взаимодействовать ли, например, с хостами доменов, не соответствующими RFC 2142 и имеющими работающий адрес `<abuse@domain>`, остается исключительно за вами.
>
> – dredd, *www.rfc-ignorant.org*

Существует множество агентов передачи сообщений, которые принимают почту не так, как этого требуют RFC (например, они могут отклонять сообщения с пустым именем отправителя конверта), по ряду ошибочных причин, включая такие:

- запрещение возвратов сообщений от анонимных отправителей;
- запрещение пустого имени отправителя (для борьбы со спамом).

Прокомментируем такое ошибочное поведение не соответствующих RFC почтовых серверов: кто угодно может подделать любой электронный адрес. Вы можете отправлять сообщения от имени `president@whitehouse.gov`, и они будут такими же анонимными, как и сообщения с пустым именем отправителя.

Спам может отправляться произвольными отправителями, но возвраты могут отправляться *только* с пустым именем отправителя конверта.

Любой почтовый сервер, блокирующий пустых отправителей конвертов, не дает своим пользователям возможности узнать о том, что их сообщения, возможно, были отклонены другим почтовым сервером: возвраты, отправляемые другим соответствующим RFC сервером, будут отклонены, т. к. для возврата используется пустое имя отправителя, как это и описано в RFC.

RFC 2821 явно определяет, что агент передачи сообщений *должен* принимать сообщения с пустым обратным путем (адресом отправителя конверта), т. к. использование пустого имени отправителя при отправке уведомления о возврате предотвращает бесконечные скитания недоставимого уведомления от одной системы к другой и обратно.

Postfix имеет параметр `reject_rhsbl_sender`, который отделяет локальную часть электронного адреса и использует доменную часть для обращения к черному списку (такому как `dsn.rfc-ignorant.org`). Если домен отправителя конверта внесен в черный список, то Postfix отклоняет входящее сообщение. Как и другие параметры черных списков, этот параметр должен быть помещен в конец списка ограничения `smtpd_recipient_restrictions`, например:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_non_fqdn_sender
  reject_unknown_sender_domain
  reject_unknown_recipient_domain
  permit_mynetworks
  reject_unauth_destination
  reject_rbl_client relays.ordb.org
  check_recipient_access hash:/etc/postfix/roleaccount_exceptions
  reject_non_fqdn_hostname
  reject_invalid_hostname
  check_helo_access pcre:/etc/postfix/helo_checks
  reject_rhsbl_sender dsn.rfc-ignorant.org
  permit
```

После перезагрузки изменения вступят в силу, и вы сможете проверить ограничение, подключившись к вашему серверу и используя отправителя конверта из домена, входящего в список `.rfc-ignorant.org`, как в следующем примере (`sender@example.com` – это официальный адрес для проверок):

```
$ telnet localhost 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250 8BITMIME
MAIL FROM:<sender@example.com>
250 Ok
RCPT TO: <recipient@example.com>
554 Service unavailable; Sender address [sender@example.com] blocked \
    using dsn.rfc-ignorant.org; Not supporting null originator (DSN)
QUIT
221 Bye
```

Проверка вручную на вхождение в черный список

Проверка домена на вхождение в RHSBL аналогична процедуре проверки IP-адреса, с тем лишь отличием, что не нужно менять порядок элементов. Просто добавьте имя сервера черных списков после имени домена, который вы хотите проверить, и выполните поиск в DNS.

Здесь проверка была проведена для домена, которого не оказалось в черном списке:

```
$ host postfix-book.com.dsn.rfc-ignorant.org
Host postfix-book.com.dsn.rfc-ignorant.org not found: 3(NXDOMAIN)
```

Если же домен найден в черном списке, то результат будет таким:

```
$ host example.com.dsn.rfc-ignorant.org
example.com.dsn.rfc-ignorant.org has address 127.0.0.2
```

Исключения для доменов отправителя из черных списков

Если вы хотите отклонять сообщения от почтовых серверов, которые не следуют правилам, но при этом вам необходимо поддерживать связь с некоторым доменом, который попадает под ограничения (так что почта из него была бы отклонена), можно создать список исключений. Для реализации исключений используйте параметр `check_sender_access` и карту исключений.

Сначала создайте файл, например `/etc/postfix/rhsbl_sender_exceptions`, содержащий пользователей и домены, от которых вы хотите принимать сообщения. Например, следующий файл разрешает прием сообщений от всех пользователей `example.com` и одного пользователя `sender@example.org`:

```
example.com                OK
sender@example.org         OK
```

Используйте команду `postmap hash:/etc/postfix/rhsbl_sender_exceptions` для создания карты. Затем добавьте параметр `check_sender_access` сразу же после параметра `reject_rhsbl_sender`, например:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_non_fqdn_sender
  reject_unknown_sender_domain
  reject_unknown_recipient_domain
  permit_mynetworks
  reject_unauth_destination
  reject_rbl_client relays.ordb.org
  check_recipient_access hash:/etc/postfix/roleaccount_exceptions
  reject_non_fqdn_hostname
  reject_invalid_hostname
  check_helo_access pcre:/etc/postfix/helo_checks
  check_sender_access hash:/etc/postfix/rhsbl_sender_exceptions
  reject_rhsbl_sender dsn.rfc-ignorant.org
  permit
```

Примечание

В примере использован параметр `check_sender_access`, всего же для исключений могут использоваться четыре параметра:

- `check_sender_access`
- `check_client_access`
- `check_helo_access`
- `check_recipient_access`

Вы уже встречались с параметром `check_helo_access` в разделе «Предотвращение явных фальсификаций». Сведения об оставшихся двух параметрах вы найдете в документации Postfix.

Проверка отправителя

Бриллиант в короне средств Postfix борьбы со спамом – это проверка адреса отправителя, в ходе которой проверяется, существует ли в домене отправителя адрес отправителя, и если такого отправителя не существует, то Postfix не принимает сообщение.

К сожалению, эта функциональность является весьма дорогостоящей, т. к. проверка занимает много времени и требует дополнительных системных ресурсов. Рассмотрим ее пошагово:

1. Клиент передает данные отправителя конверта.
2. Postfix формирует и ставит в очередь пробное сообщение отправителю конверта.
3. Postfix ищет MX- или A-запись домена отправителя конверта.
4. Postfix пытается подключиться к почтовому серверу отправителя. Если подключиться к удаленному серверу не удается, то `smtpd` откладывает решение о принятии сообщения, возвращая клиенту код временной ошибки 450. Тем временем Postfix продолжает пытаться проверить адрес.
5. Postfix инициирует сеанс SMTP с удаленным сервером.
6. Postfix передает данные отправителя конверта удаленному почтовому серверу в качестве сведений о получателе конверта.
7. В зависимости от ответа удаленного сервера Postfix делает одно из двух:
   - Если удаленный почтовый сервер принимает получателя (отправителя исходного конверта), то Postfix отключается, уничтожает пробное сообщение и принимает сообщение от исходного клиента.
   - Если удаленный почтовый сервер отклоняет получателя (отправителя исходного конверта), то Postfix отключается, уничтожает пробное сообщение и отклоняет сообщение от клиента.

При включенной проверке адресов отправителей сообщения обычно будут испытывать задержку до 9 секунд на проверку адреса, встретив-

шегося *впервые*. Однако затем Postfix кэширует статус адреса, так что последующие сообщения не будут задерживаться.

Если проверка длится более 9 секунд, то `smtpd` отвергает сообщение клиента (отправляющего компьютера) с кодом 450. Обычные почтовые клиенты повторят попытку через некоторое время, а захваченные (hijacked) прокси-серверы – нет, т. к. они занимаются только пересылкой команд SMTP, и человек, управляющий таким прокси-сервером, не захочет терять дополнительное время.

Настройка проверки адреса отправителя

Для включения проверки адреса отправителя добавьте параметр `reject_unverified_sender` в свой список ограничений `smtpd_recipient_restrictions`, например:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_non_fqdn_sender
  reject_unknown_sender_domain
  reject_unknown_recipient_domain
  permit_mynetworks
  reject_unauth_destination
  reject_rbl_client relays.ordb.org
  check_recipient_access hash:/etc/postfix/roleaccount_exceptions
  reject_non_fqdn_hostname
  reject_invalid_hostname
  check_helo_access pcre:/etc/postfix/helo_checks
  check_sender_access hash:/etc/postfix/rhsbl_sender_exceptions
  reject_rhsbl_sender dsn.rfc-ignorant.org
  reject_unverified_sender
  permit
```

Кроме `reject_unverified_sender` существуют и другие параметры, которые можно добавить в ограничения. Однако параметры обладают разумными значениями по умолчанию, и служат они скорее для регулировки проверки адреса отправителя, чем для ее настройки. Следующие подразделы описывают наиболее часто используемые изменения проверки адреса отправителя. Дополнительные параметры настройки вы сможете найти в файле `ADDRESS_VERIFICATION_README`, который присутствует в дистрибутиве Postfix.

Отправитель пробного конверта

Когда Postfix формирует пробное сообщение для проверки отправителя, он сам должен представиться удаленному серверу – указать своего отправителя конверта. Вы можете задать этот адрес в параметре `address_verify_sender`. Значение по умолчанию – `postmaster@$myorigin`.

При желании указать другого отправителя конверта для пробного сообщения добавьте параметр `address_verify_sender` в файл `main.cf`:

```
address_verify_sender = sender@example.com
```

Конечно, такой адрес отправителя должен существовать; не забывайте о том, что другие серверы также могут использовать по отношению к вам механизм проверки адреса отправителя.

Примечание

На адрес получателя, указанный в параметре `address_verify_sender`, не накладываются никакие ограничения.

Кэширование

По умолчанию Postfix хранит проверенные адреса отправителей в оперативной памяти. Когда вы перезагружаете параметры или перезапускаете Postfix, то теряете их, если только не будете использовать (что не обязательно) дополнительную базу данных для постоянного хранения адресов. Для того чтобы работать с базой данных, задайте путь к ней в параметре `address_verify_map` (убедившись, что в выбранной файловой системе достаточно *много* свободного места), например:

```
address_verify_map = btree:/var/spool/postfix/verified_senders
```

После перезагрузки Postfix создаст базу данных и будет добавлять туда результаты как положительных, так и отрицательных проверок. Если вы хотите отменить сбор отрицательных данных, задайте параметр `address_verify_negative_cache` в файле `main.cf`:

```
address_verify_negative_cache = no
```

Выборочная проверка адресов отправителей

По мере возрастания нагрузки на ваш почтовый сервер процедура проверки адресов получателей, вероятнее всего, приведет к нехватке ресурсов. В этот момент следует перейти к выборочной проверке адресов отправителей.

Выборочная проверка адресов отправителей работает на основе карты обычно используемых спамерами доменов отправителей конвертов. Если домен отправителя входящего сообщения присутствует в карте, то Postfix проверяет отправителя, иначе – не беспокоится. Вам нужно создать файл карты, например `/etc/postfix/common_spam_senderdomains`, и указать параметр `reject_unverified_sender` в качестве действия, которое следует предпринять в случае совпадения с доменом отправителя конверта. Приведем пример того, как может выглядеть такой файл:

```
hotmail.com reject_unverified_sender
web.de      reject_unverified_sender
msn.com     reject_unverified_sender
mail.ru     reject_unverified_sender
```

Страница руководства access(5) поясняет, что правая часть карты – это имя действующего ограничения или класса `smtpd_restriction_class`.

В данном примере при инициировании клиентом передачи сообщения Postfix делает одно из двух:

- Если домену отправителя соответствует запись в common_spam_senderdomains, то просмотр карты возвращает действие reject_unverified_sender, и Postfix проверяет отправителя конверта. В случае успешной проверки reject_unverified_sender возвращает DUNNO, и Postfix переходит к оценке следующего ограничения. Если же адрес недействителен, то Postfix отклоняет сообщение.
- Если домену отправителя не соответствует никакая запись в common_spam_senderdomains, то поиск в карте не дает результата, и выборочный оценщик возвращает DUNNO, а Postfix оценивает следующее ограничение, не проверяя адрес отправителя.

После создания карты преобразуйте ее в базу данных, используя команду postmap hash:/etc/postfix/common_spam_senderdomains. Наконец, замените существующий параметр reject_unverified_sender на параметр check_sender_access с картой в качестве аргумента. Приведем пример, в котором используется карта hash:/etc/postfix/common_spam_senderdomains:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_non_fqdn_sender
  reject_unknown_sender_domain
  reject_unknown_recipient_domain
  permit_mynetworks
  reject_unauth_destination
  check_recipient_access hash:/etc/postfix/roleaccount_exceptions
  reject_non_fqdn_hostname
  reject_invalid_hostname
  check_helo_access pcre:/etc/postfix/helo_checks
  check_sender_access hash:/etc/postfix/rhsbl_sender_exceptions
  reject_rhsbl_sender dsn.rfc-ignorant.org
  check_sender_access hash:/etc/postfix/common_spam_senderdomains
  permit
```

Вы можете пойти дальше и ввести, помимо отправителя конверта, дополнительные критерии проверки, например содержимое сообщения. Создайте новую карту с именем common_spam_senderdomain_keywords – она будет содержать ключевые слова из имен доменов, которые будут запускать проверку адреса отправителя, например:

```
/sex/   reject_unverified_sender
/girl/  reject_unverified_sender
/sell/  reject_unverified_sender
/sale/  reject_unverified_sender
/offer/ reject_unverified_sender
/power/ reject_unverified_sender
```

Затем добавьте еще один параметр check_sender_access, указывающий на новую карту:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_non_fqdn_sender
  reject_unknown_sender_domain
  reject_unknown_recipient_domain
  permit_mynetworks
  reject_unauth_destination
  check_recipient_access hash:/etc/postfix/roleaccount_exceptions
  reject_non_fqdn_hostname
  reject_invalid_hostname
  check_helo_access pcre:/etc/postfix/helo_checks
  check_sender_access hash:/etc/postfix/rhsbl_sender_exceptions
  reject_rhsbl_sender dsn.rfc-ignorant.org
  check_sender_access hash:/etc/postfix/common_spam_senderdomains
  check_sender_access regexp:/etc/postfix/common_spam_senderdomain_keywords
  permit
```

Порядок введения ограничений

Борьба со спамом весьма затратна с точки зрения системных ресурсов. Рассмотрим пример того, как следует упорядочивать параметры, предотвращающие прием спама:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_recipient
  reject_non_fqdn_sender
  reject_unknown_sender_domain
  reject_unknown_recipient_domain
  permit_mynetworks ❶
  (permit_sasl_authenticated)
  (pop-before-smtp)
  reject_unauth_destination
  check_recipient_access hash:/etc/postfix/roleaccount_exceptions
  check_helo_access pcre:/etc/postfix/helo_checks ❷
  reject_non_fqdn_hostname
  reject_invalid_hostname
  check_sender_mx_access cidr:/etc/postfix/bogus_mx ❸
  check_sender_access hash:/etc/postfix/rhsbl_sender_exceptions ❹
  reject_rhsbl_sender dsn.rfc-ignorant.org ❺
  check_sender_access hash:/etc/postfix/common_spam_senderdomains ❻
  check_sender_access regexp:/etc/postfix/common_spam_senderdomain_keywords
  permit
```

Общая идея заключается в том, чтобы «дешевые» ограничения предшествовали «дорогостоящим»:

❶ Поместите все параметры борьбы со спамом после `permit_mynetworks`, с тем чтобы они относились только к внешним клиентам (т. е. клиентам, не перечисленным в `mynetworks`).

❷ Вы можете без промедлений отвергать любой клиент, который использует имя хоста вашего сервера. Уже не имеет значения, использует ли он полное имя хоста и действительно ли это имя.

❸ С этого параметра начинаются дорогостоящие ограничения. Для `check_sender_mx_access` требуется один или два поиска в DNS. Если у вас работает кэширующий сервер имен, вы можете разрешать запросы к DNS локально.

❹ Эта карта стоит перед параметрами черных списков, т. к. она содержит исключения для пользователей и доменов, которые в противном случае могли бы быть отвергнуты.

❺ Этот параметр требует запроса к удаленной системе (DNS-серверу для `dsn.rfc-ignorant.org`), которая может быть сильно загружена или временно не работать. Параметр дорогостоящий, поэтому он используется ближе к концу списка ограничений.

❻ Последними вводятся два самых дорогостоящих действия. Если они запущены, Postfix должен создать фиктивное сообщение, попытаться его доставить и записать результат. Очень затратно, поэтому используется в последнюю очередь.

Использование классов ограничений

В примере, рассматриваемом в данном разделе, ограничения на отправителей конверта накладываются с двух сторон. Во-первых, мы требуем, чтобы у сообщений, приходящих извне, адрес отправителя *не* относился к нашему домену, и во-вторых, чтобы сообщения от внутренних клиентов *имели* адрес отправителя, относящийся к нашему домену.

Идея заключается в том, чтобы Postfix сначала проверял, относится ли входящее клиентское соединение к вашей сети:

1. Если клиент находится в вашей сети, Postfix отправляет его в класс ограничений. Этот класс содержит проверку адреса отправителя конверта:
   - Если отправитель конверта соответствует шаблону вашего домена, проверка возвращает `OK`. Postfix прекращает оценку ограничений и позволяет клиенту продолжать.
   - Если отправитель конверта не соответствует шаблону вашего домена, то следующий параметр ограничений – это `reject`, так что Postfix отказывает клиенту в обслуживании.
2. Если клиент не относится к вашей сети, Postfix не использует класс ограничений. Вместо этого он переходит к следующему ограничению, в котором проверяется адрес получателя конверта:
   - Если клиент использует имя вашего домена в адресе отправителя конверта, то Postfix отказывает в обслуживании.

- Если клиент не использует имя вашего домена в адресе отправителя конверта, то тест пройден, и Postfix переходит к следующему ограничению.

Для реализации описанного алгоритма создадим файл карты, содержащий список IP-адресов и сетей внутри вашей сети. Назвать файл можно `/etc/postfix/internal_networks`; выглядеть он будет примерно так:

```
192.0.34        has_our_domain_as_sender
192.168         has_our_domain_as_sender
192.168.1       has_our_domain_as_sender
```

Затем создадим другой файл карты `/etc/postfix/our_domain_as_sender`, содержащий шаблон вашего домена и пустое имя отправителя конверта (помните, что ваш сервер должен без вопросов принимать сообщения такого отправителя); это будет список доменов отправителей конверта, которые могут использовать внутренние клиенты. Этот файл карты может быть таким:

```
example.com     OK
<>              OK
```

Теперь создадим файл карты, содержащий домены, которые внешние клиенты не могут использовать для отправителя конверта. В нашем примере файл будет называться `/etc/postfix/not_our_domain_as_sender` и содержать всего одну строку:

```
example.com     554 Do not use my domain in your envelope sender
```

После создания при помощи команды `postmap` карт на основе этих файлов задайте класс ограничений и необходимые параметры ограничений в файле `main.cf`:

```
smtpd_restriction_classes =
  has_our_domain_as_sender
has_our_domain_as_sender =
  check_sender_access hash:/etc/postfix/our_domain_as_sender
  reject
smtpd_recipient_restrictions =
  check_client_access hash:/etc/postfix/internal_networks
  check_sender_access hash:/etc/postfix/not_our_domain_as_sender
  reject_unauth_destination
  ...
  permit
```

Как обычно, для того чтобы изменения вступили в силу, необходимо перезагрузить конфигурацию Postfix.

9

Как работают встроенные фильтры содержимого

Проверки исследуют содержимое сообщения и на основе этого исследования выполняют предопределенное действие. В этой главе рассказывается о том, какими могут быть проверки и какие действия предпринимает Postfix для контроля над содержимым.

Проверки дополняют ограничения. *Ограничения* следят за SMTP-диалогом, в то время как *проверки* занимаются содержимым сообщения. На первый взгляд может показаться, что проверки очень сильно отличаются от ограничений. Проверки легко активируются, но синтаксис, используемый для создания шаблонов поиска, может быть весьма сложным, потому что в шаблонах применяются регулярные выражения.

Postfix – это агент передачи сообщений, и встроенные проверки не пытаются подменить собой полнофункциональное средство анализа содержимого, они лишь помогают решению простых задач. Вот что вы можете делать с их помощью:

- Блокировать сообщения, сформированные некоторыми программами, например вашим почтовым шлюзом SAP.
- Блокировать сообщения с определенными строками в заголовке Subject.
- Уничтожать сообщения, содержащие потенциально вредные вложения.
- Удалять фрагменты данных из заголовков сообщений.

Примечание

При отсутствии опыта работы с регулярными выражениями обратитесь к отличной книге на эту тему: Джеффри Фридл (Jeffrey E. F. Friedl) «Mastering Regular Expressions» («Регулярные выражения», 3-е издание, Символ-Плюс, 2008).

Как работают проверки?

Проверки просматривают сообщения, выполняя поиск заданных образцов. В случае совпадения какого-то элемента содержимого с образцом выполняется некоторое действие. Postfix может применять разные фильтры к разным разделам сообщения. В настоящее время Postfix поддерживает следующие разделы:

- Заголовки сообщений
- MIME-заголовки
- Тело сообщения, включая вложения
- Заголовки вложенных сообщений

Для создания набора проверок вы определяете отдельные шаблоны в отдельных картах, а затем назначаете эти карты различным параметрам проверок, которые применяются к разным разделам. Для исследования содержимого сообщения Postfix использует карты со встроенной MIME-программой синтаксического анализа. Эта программа работает как команда `egrep`; она может распознавать только простой текст и только построчно. Посмотрим, как все это работает:

1. Синтаксический анализатор проходит сообщение строка за строкой.
2. Синтаксический анализатор определяет, к какому разделу сообщения относится текущая строка.
3. Если для раздела существует проверка, Postfix использует соответствующую карту для сравнения содержимого с шаблоном карты.
4. Если соответствие найдено, то Postfix инициирует действие, оставшиеся проверки не выполняются. То есть «кто первый встал, того и тапки».

Возможно, вы уже догадались, что проверки интенсивно используют процессор, поэтому решающее значение может иметь порядок шаблонов поиска в карте, т. к. чем раньше найдено совпадение, тем меньше процессорного времени использует процесс проверки.

Применение проверок к отдельным разделам сообщения

Postfix использует отдельный параметр конфигурации для каждого известного ему раздела сообщения. Приведем перечень параметров проверок (имейте в виду, что по умолчанию в файле `main.cf` они не включены).

`header_checks`

Эти проверки применяются к заголовку сообщения, т. е. ко всему от первой строки сообщения до первой пустой строки, включая заголовки, которые физически занимают несколько строк.

`body_checks`

Эти проверки применяются к телу сообщения; синтаксический анализатор воспринимает как тело все, что находится между заголовками.

Предупреждение

Использование большого количества шаблонов `body_checks` очень сильно загружает процессор, значительно замедляя работу компьютера, т. к. при проверке тела анализируется каждая строка сегмента тела и каждая из них сравнивается с каждым регулярным выражением, определенным в карте `body_checks`.

Чтобы не создавать излишней нагрузки на процессор, Postfix по умолчанию проверяет только первые 51 200 байт текущего сегмента тела. Вы можете увеличить это количество, используя параметр `body_checks_size_limit`. Для того чтобы справиться с возросшей нагрузкой, можно передать контроль содержимого другому приложению, работающему на отдельном компьютере, используя функциональность `content_filter`, описанную в главе 11.

`mime_header_checks`

Эти проверки применяются к MIME-заголовкам в заголовках сообщений верхнего уровня, заголовках MIME-элементов и к MIME-заголовкам в заголовках вложенных сообщений в соответствии с RFC 822 (см. рис. 6.3 на с. 90).

`nested_header_checks`

Эти проверки применяются к заголовкам вложенных сообщений, за исключением MIME-заголовков. Действуют только на заголовки вложенных сообщений `message/rfc822`, за исключением MIME-заголовков, перечисленных ранее в описании параметра `mime_header_checks`.

Что особенного в этих параметрах?

Postfix 2.*x* обрабатывает тело сообщения как состоящее из *n* сегментов, при этом каждый сегмент помечен MIME-заголовком. Такая MIME-обработка включена по умолчанию, но вы можете отменить ее, указав `disable_mime_input_processing = yes` в своем файле `main.cf`.

Синтаксический анализатор MIME исследует каждую прочитанную строку, выясняя, к заголовку или к сегменту тела она относится. От ответа на этот вопрос зависит то, какие проверки выполняет Postfix. Если сегмент сообщения имеет почтовые заголовки (т. е. является вложенным сообщением типа `message/rfc822`), такие заголовки оцениваются параметром `nested_header_checks`.

Все, что следует внутри сегмента за вложенными заголовками, оценивается параметром `body_checks`, при этом анализируемый объем определяется параметром `body_checks_size_limit`. Например, если у вас есть сообщение из пяти 100-килобайтных MIME-сегментов (или вложений), то Postfix проверяет в *каждом* сегменте первые `body_checks_size_limit` байт.

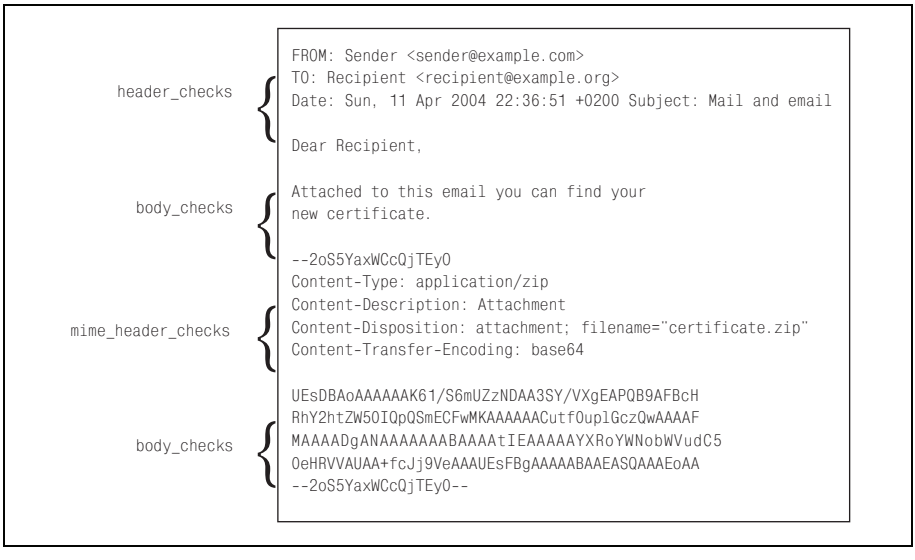

***Рис. 9.1.** Postfix принимает решение о том, какую проверку использовать, отдельно для каждой строки сообщений*

Postfix использует параметр `mime_header_checks` для оценки каждого MIME-заголовка (начала каждого нового сегмента). Если после какого-то MIME-заголовка есть почтовые заголовки, они оцениваются параметром `nested_header_checks` в каждом сегменте.

На рис. 9.1 показано, какие проверки применяются для каждой строки сообщения.

Когда Postfix применяет проверки?

Клиент передает сообщение после успешного завершения первоначального SMTP-диалога. То есть Postfix обрабатывает параметры `*_checks` после обработки параметров ограничений `smtpd_*_restrictions`. На рис. 9.2 показано, когда демон `cleanup` начинает заниматься проверками.

Какие действия могут вызвать проверки?

Для каждого шаблона поиска вы можете определить только одно соответствующее действие. В настоящее время Postfix поддерживает следующие действия:

`REJECT [необязательный текст...]`

Отказывает сообщению в приеме. Необязательный текст будет отправлен клиенту, пытающемуся передать сообщение. Postfix также запишет этот текст в почтовый журнал.

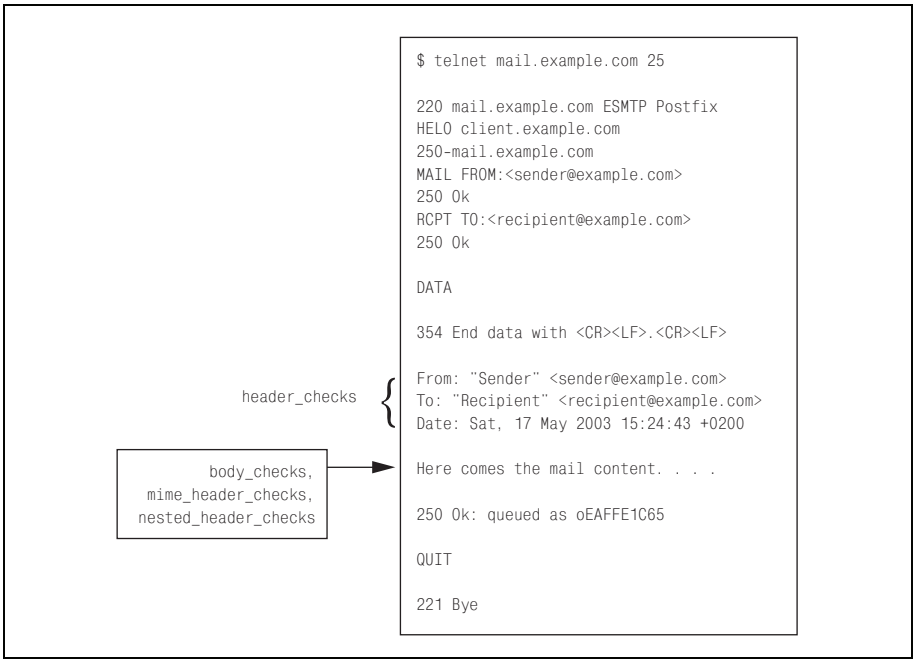

Рис. 9.2. *Проверки применяются после ограничений и только к содержимому сообщения*

`IGNORE`

Удаляет из сообщения строку, которая совпадает с шаблоном поиска.

`WARN [необязательный текст...]`

Заставляет Postfix записать в почтовый журнал предупреждение. Если указан необязательный текст, он также будет записан в журнал. При этом Postfix доставит сообщение без каких-либо изменений.

`HOLD [необязательный текст...]`

Помещает сообщение в очередь отложенных сообщений, где оно будет находиться до тех пор, пока администратор почтовой системы не исследует его и не решит, что с ним делать. Postfix записывает в журнал строку тела или заголовка, в которой было обнаружено совпадение с шаблоном, а также необязательный текст.

`DISCARD [необязательный текст...]`

Сообщает почтовому клиенту об успешной доставке сообщения, но молча удаляет сообщение вместо того, чтобы передавать его в место конечного назначения. Если указан необязательный текст, то Postfix записывает его вместе со строкой, в которой найдено совпадение с шаблоном, в журнал электронной почты.

`FILTER transport:nexthop`

Отправляет сообщение фильтру (службе, определенной в файле `master.cf`, которая занимается передачей сообщений другой системе обработки данных, например антивирусному сканеру). Более подробно об определении фильтров рассказывается в главе 11.

`REDIRECT user@domain`

Перенаправляет сообщение по указанному адресу вместо того, чтобы доставить его первоначальному получателю (получателям). Подменяет любое действие `FILTER`.

10

Использование встроенных фильтров содержимого

Как вы знаете из главы 9, Postfix может исследовать содержимое сообщения на основе таблиц шаблонов и действий. В этой главе будет показано, как применять эти шаблоны и действия. Помните, что проверки предназначены только для простой фильтрации содержимого. Для решения более сложных задач обратитесь к главе 11.

Проверки ищут в сообщениях определенные символы, а также могут изменять сообщения. Имена параметров конфигурации, включающих проверки, заканчиваются на `_checks`, и все они — `header_checks`, `body_checks`, `mime_header_checks` и `nested_header_checks` — действуют по одой схеме:

1. Postfix анализирует сообщение строку за строкой, сравнивая их с картой шаблонов, составленных из регулярных выражений (regexp) или регулярных выражений Perl (PCRE).
2. Если строка соответствует регулярному выражению, то Postfix предпринимает действие, определенное для данного выражения, *и* переходит к исследованию следующей строки ввода.

Предупреждение

В примерах этой главы повсеместно используется синтаксис продолжаемых строк, поддерживаемый Postfix для улучшения читаемости на бумаге. Речь идет о том, что строка, начинающаяся с символов пробела, является продолжением предыдущей строки.

Проверка Postfix на поддержку проверок

Postfix по умолчанию поддерживает фильтрацию заголовков и тел сообщений, но т. к. он может использовать карты regexp и (или) PCRE,

вам нужно узнать, поддерживает ли Postfix оба типа регулярных выражений или же только regexp. Чтобы проверить, какие карты поддерживает ваша версия Postfix, выполните команду `postconf -m` – будут выведены все типы карт, поддерживаемых системой. В приведенном примере Postfix поддерживает regexp и PCRE, а также ряд других карт:

```
$ postconf -m

btree
cidr
environ
hash
nis
pcre
proxy
regexp
static
tcp
unix
```

Все системы должны по умолчанию поддерживать таблицы типа regexp. Если у вашей системы при использовании карт регулярных выражений возникают проблемы с производительностью (или, еще хуже, если реализация regexp в вашей системе содержит ошибки), то вы можете установить библиотеки и заголовочные файлы PCRE и пересобрать Postfix с поддержкой PCRE.

Сборка Postfix с поддержкой карт PCRE

Для того чтобы собрать Postfix с поддержкой PCRE, вам понадобятся библиотеки и заголовочные файлы PCRE. Вы найдете их в инструментальном пакете вашего дистрибутива или можете получить исходные тексты PCRE с сайта *http://www.pcre.org* и установить их вручную.

Для настройки поддержки PCRE в Postfix необходимо добавить `-DHAS_PCRE` и флаг препроцессора `-I` для каталога включаемых файлов PCRE в `CCARGS` и флаги для библиотеки PCRE и пути в `AUXLIBS`. Например, пусть `pcre.h` находится в каталоге `/usr/local/include`, а `pcre.a` – в каталоге `/usr/local/lib`:

```
$ CCARGS="-DHAS_PCRE -I/usr/local/include" \
  AUXLIBS="-L/usr/local/lib -lpcre" \
  make makefiles
$ make
```

Примечание

Для Solaris также необходимо указать `-R/usr/local/lib`.

Безопасная реализация фильтрации заголовка или тела сообщения

Регулярные выражения могут стать очень сложными, и может получиться так, что ваш шаблон не будет работать, ему будет соответствовать больше строк, чем предполагалось, или вы вообще перестанете понимать, что происходит.

Для отладки шаблонов Postfix предлагает действие `WARN`. Если вы используете это действие в правой части своего шаблона, то Postfix при совпадении с выражением будет доставлять сообщение и одновременно записывать комментарий в журнал электронной почты. После того как вы убедитесь, что шаблон работает, просто замените `WARN` на нужное действие.

Безопасная процедура добавления проверок выглядит следующим образом:

1. Добавляете в карту свой шаблон с соответствующим ему действием `WARN`.
2. Создаете файл, содержащий выражение, совпадающее с шаблоном фильтра.
3. Проверяете, совпадает ли шаблон карты с тестовым шаблоном.
4. Включаете проверку в основном файле конфигурации Postfix.
5. Тестируете фильтр на реальных сообщениях.

Добавление регулярного выражения и определение действия WARN

Первый этап – это добавление шаблона, который вы хотите проверять, в карту, и определение действия `WARN` на тот случай, когда содержимое сообщения будет совпадать с тестовым шаблоном. В этом разделе мы будем рассматривать пример, проверяющий на соответствие шаблону заголовок Subject, но вы сможете использовать эту процедуру и для других заголовков и других параметров `*_checks`.

Добавьте шаблон фильтра в файл `/etc/postfix/header_checks`. Этот файл хранит карту для параметра `header_checks`, например:

```
/^Subject: FWD: Look at pack from Microsoft/
  WARN Unhelpful virus warning
```

Создание тестового шаблона

Все, что вам нужно сделать для создания тестового шаблона, – это поместить совпадающее с шаблоном сообщение в файл, например `/tmp/testpattern`. В нашем примере это будет такое сообщение:

```
From: dingdong@example.com
Subject: FWD: Look at pack from Microsoft
blah blah
```

Соответствует ли регулярное выражение тестовому шаблону?

Проверим шаблон фильтра, передав карту проверки и тестовый шаблон команде postmap. Выполним, например, такую команду:

```
$ postmap -q - regexp:/etc/postfix/header_checks < /tmp/testpattern
```

В случае совпадения команда печатает соответствующую строку тестового шаблона, например:

```
Subject: FWD: Look at pack from Microsoft        WARN Unhelpful virus warning
```

Если соответствие шаблону не найдено, то команда postmap ничего не выводит.

Определение проверки в основной конфигурации

Если пока все работает, вы можете отредактировать свой файл main.cf, использовав только что созданный и протестированный файл, содержащий header_checks:

```
header_checks = regexp:/etc/postfix/header_checks
```

Перезагрузите конфигурацию и отправьте тестовое сообщение, содержащее тот же самый тестовый шаблон.

Тестирование на реальном сообщении

Для проверки фильтра на реальном сообщении выполним команду для передачи нашего теста серверу Postfix:

```
$ /usr/sbin/sendmail recipient@example.com < /tmp/testpattern
```

Теперь посмотрим на почтовый журнал, чтобы проверить, записал ли Postfix сообщение для тестового шаблона. Вторая строка данного фрагмента журнала содержит предупреждение:

```
Mar 30 17:17:52 mail postfix/pickup[2455]: 53CAB633B3: uid=7945
   from=<sender@example.com>
Mar 30 17:17:52 mail postfix/cleanup[2461]: 53CAB633B3:
   warning: header Subject: FWD: Look at pack from Microsoft from local;
   from=<sender@example.com> to=<recipient@example.com>:
   Unhelpful virus warning
Mar 30 17:17:52 mail postfix/cleanup[2461]: 53CAB633B3:
   message-id=<20040330151752.53CAB633B3@mail.example.com>
Mar 30 17:17:52 mail postfix/qmgr[2456]: 53CAB633B3:
   from=<sender@example.com>, size=346, nrcpt=1 (queue active)
```

После того как вы убедитесь, что шаблон фильтра работает, вы можете без риска заменить действие WARN на то действие, которое на самом деле что-то делает, такое как REJECT или DISCARD.

Проверка заголовков

По результатам проверки заголовков Postfix может выполнять разнообразные действия, такие как отклонение или удержание сообщения, удаление заголовков, отбраковка, перенаправление или фильтрация сообщений. В этом разделе будет рассказано о том, как реализовать эти действия.

Отклонение сообщений

Postfix может отклонять сообщения, используя действие `REJECT`. Вы можете применять это действие для блокировки сообщений, которые соответствуют шаблону, например, содержащих определенный заголовок Subject.

Отклонение предотвращает попадание сообщений в систему, не подпуская их к требующему большого количества вычислений антивирусному средству, программе выявления спама или (что, возможно, еще хуже) к вашим пользователям. Рассмотрим несколько примеров.

Этот шаблон отклоняет бесполезные предупреждения о вирусах, формируемые программой ScanMail (которая всегда предупреждает отправителя, даже если вирус подделал адрес отправителя):

```
/^Subject: ScanMail Message: To Sender, sensitive content found and action/
   REJECT Unhelpful virus warning
```

Если вы хотите блокировать сообщения с некорректным заголовком Undisclosed-recipients, то можете использовать следующий шаблон. Этому шаблону будет соответствовать ситуация, когда в заголовках встречается `To: <Undisclosed Recipients>` (в скобках или без них, с буквой «s» в конце или без нее). Корректный же заголовок Undisclosed-recipients должен выглядеть так: `To: undisclosed-recipients:;`.

```
/^To:.*<?Undisclosed Recipients?>?$/
   REJECT Wrong undisclosed recipients header
```

Следующий шаблон отлично описан в его собственном комментарии и сопроводительном сообщении:

```
#
# Спам, который содержит Subject: something     565876
#
/^Subject:.*[[:space:].]{5,}\(?#?[[:digit:]]{2,}\)?$/
   REJECT More than 5 whitespaces and a number follow the Subject:
```

Нам никогда не встречалось `To:...<>` в заголовках правомерных сообщений:

```
/^To:.*<>/
   REJECT To: <> in headers
```

Наконец, некоторые строки темы сообщения как две капли воды похожи на предупреждения о спаме. Для того чтобы понять, о чем речь, по-

смотрите на четыре предложенных шаблона (использование разных цифр для предупреждений облегчает отладку ошибок).

```
#
# Некоторые строки заголовка Subject являются признаками спама.
#
/^Subject:.*is NOT being SEEN/            REJECT fraud spam #1
/^Subject:.*URGENT BUSINESS RELATIONSHIP/ REJECT fraud spam #2
/^Subject:.*Confidential Proposal/        REJECT fraud spam #3
/^Subject: SEX-FLATRATE/                  REJECT fraud spam #4
```

Приостановка доставки

Postfix может задержать доставку сообщений при помощи действия `HOLD`. Его можно использовать для помещения подозрительных сообщений «на удержание» для дальнейшей проверки. Для просмотра сообщений используйте команду `postcat`, а для отправки сообщения – команду `postsuper -H`. Для того чтобы удалить сообщение из очереди Postfix, используйте команду `postsuper -d`.

Рассмотрим шаблон, соответствующий любому сообщению, заголовок Subject: которого начинается с `Subject: [список_имен]`. Одним из применений такого шаблона является удержание сообщений, предназначенных всем пользователям внутренних списков рассылки, до окончания рабочего времени, когда уменьшится загруженность системы:

```
/^Subject: \[список_имен\]/
   HOLD
```

Следующий шаблон удерживает сообщения, в MIME-заголовках которых используется один символ возврата каретки. Большую часть таких сообщений составляют вирусы и спам, а незначительное количество – результат неправильной установки SquirrelMail под Windows.

```
/\r/
   HOLD Lone CR in headers indicates virus or spam!
```

Удаление заголовков

Если вы хотите удалить строки из заголовков, используйте действие `IGNORE`. Вы можете использовать его для сокрытия сведений, записанных в заголовках (например, тип используемого вами почтового клиента) или для того, чтобы удалить заголовки Received, которые могли добавить ваши внутренние почтовые серверы, брандмауэры и антивирусные средства. Рассмотрим шаблон, применяемый для удаления заголовков Received, добавленных программой, которой Postfix поручил обработку почты посредством директивы `content_filter` (например, из amavisd-new):

```
/^Received: from localhost /
   IGNORE
```

В следующем примере удаляется заголовок Sender: некоторые версии Outlook странно ведут себя, отвечая на сообщение, которое содержит такой заголовок:

```
/^Sender:/
   IGNORE
```

Отбраковывание сообщений

Postfix может молча отбраковывать сообщения, используя действие DISCARD. Например, может понадобиться удалить сообщения с определенной строкой в поле темы так, чтобы никто этого не заметил. Рассмотрим глупый пример:

```
/^Subject:.*deadbeef/
   DISCARD No dead meat!
```

Удаляя сообщение, Postfix, как обычно, записывает действие в журнал. Например, вы можете увидеть в журнале следующее:

```
Apr  9 23:14:28 mail postfix/cleanup[11580]: BB92B15C009: discard: header
   Subject: deadbeef from client.example.com[10.0.0.1];
   from=<sender@example.com> to=<recipient@example.com>
   proto=ESMTP helo=<client.example.com>: No dead meat!
```

Перенаправление сообщений

Postfix может перенаправить сообщение другим получателям, используя действие REDIRECT в случае соответствия шаблону в заголовках и в теле сообщения. Рассмотрим пример перенаправления (на самом деле мы бы не рекомендовали его использовать):

```
/Subject:.*deadbeef/
   REDIRECT bigbrotheriswatchingyou@example.com
```

В журнале запись о перенаправлении будет выглядеть так:

```
Apr  9 23:20:38 mail postfix/smtpd[11873]: 9305215C009:
   client=client.example.com[10.0.0.1]
Apr  9 23:20:38 mail postfix/cleanup[11865]: 9305215C009: redirect: header
   Subject: deadbeef from client.example.com[10.0.0.1];
   from=<sender@example.com> to=<recipient@example.com>
   proto=ESMTP helo=<client.example.com>: bigbrotheriswatchingyou@example.com
Apr  9 23:20:38 mail postfix/cleanup[11865]: 9305215C009:
   message-id=<20040409212038.GK3406@example.com>
Apr  9 23:20:38 mail postfix/qmgr[11857]: 9305215C009:
   from=<sender@example.com>, size=1111, nrcpt=1 (queue active)
Apr  9 23:21:08 mail postfix/smtp[11874]: 9305215C009:
   to=<bigbrotheriswatchingyou@example.com>,
   orig_to=<recipient@example.com>, relay=none, delay=30, status=deferred
   (connect to example.com [192.0.34.166]: Connection timed out)
```

Фильтрация сообщений

Postfix может направлять сообщения механизму content_filter (см. главу 11), используя действие FILTER. Например, вы можете перенаправлять некоторые классы сообщений транспортам различных типов в зависимости от их заголовков.

Это действие перекрывает настройки content_filter из файла main.cf и требует, чтобы вы убедились в том, что те же самые *_checks не используются при возврате сообщений после фильтрации. Параметры проверок header_checks и body_checks для второго сервера cleanup должны быть отключены во избежание зацикливания! Подробная информация по данному вопросу приведена в главе 12 (ищите описания no_header_body_checks и receive_override_options). Первый из приведенных ниже шаблонов не отправляет сообщение фильтру, а второй – отправляет.

```
/^To:.*@example\.org/   FILTER nofilter:dummy
/^To:.*@example\.com/   FILTER virusfilter:dummy
```

Примечание

Помните, что это всего лишь пример, не следует применять его на рабочем сервере! Одно сообщение, предназначенное получателям в обоих доменах, совпадет с первым регулярным выражением и соответственно (выигрывает всегда первое совпадение) никогда не будет отправлено фильтру; второе действие никогда не будет выполнено.

Фильтрация сообщений оставляет такие записи в почтовом журнале:

```
Apr  9 23:34:12 mail postfix/cleanup[12543]: 2B97315C00D: filter: header To:
   nofilter@example.org from client.example.com[10.0.0.1];
   from=<sender@example.com> to=<nofilter@example.org> proto=ESMTP
   helo=<client.example.com>: nofilter:dummy
Apr  9 23:38:00 mail postfix/cleanup[12543]: 2299815C00E: filter: header To:
   virusfilter@example.com from client.example.com[10.0.0.1];
   from=<sender@example.com> to=<virusfilter@example.com> proto=ESMTP
   helo=<client.example.com>: virusfilter:dummy
```

Проверка MIME-заголовков

MIME-заголовки прилагаются к файлам, вложенным в сообщение. По умолчанию для просмотра MIME-заголовков на предмет соответствия шаблонам будет использоваться карта header_checks, если вы не определите отдельную карту и не укажете Postfix на необходимость ее использования в параметре mime_header_checks.

Примечание

Определение отдельных карт имеет смысл, когда вы хотите минимизировать размер карты mime_header_checks, используя шаблоны для заголовков MIME только в том случае, если Postfix выявит наличие вложения в сообщении.

Сначала следует создать карту для хранения шаблонов MIME-заголовков. Пусть это будет файл /etc/postfix/mime_header_checks, содержащий следующие проверки:

```
# Files blocked by their suffix
/name=\"(.*)\.(386|bat|bin|chm|cmd|com|do|exe|hta|jse|lnk|msi|ole)\"$/
   REJECT Unwanted type of attachment $1.$2
/name=\"(.*)\.(pif|reg|rm|scr|shb|shm|shs|sys|vbe|vbs|vxd|xl|xsl)\"$/
   REJECT Unwanted type of attachment $1.$2
```

В данном примере Postfix ищет MIME-заголовки, содержащие строку name=", за которой следует произвольное количество символов, а затем – точка (.). Далее идет заключенный в скобки большой список, который содержит несколько запрещенных расширений, разделенных вертикальной чертой (|). Регулярное выражение заканчивается кавычкой (\"), которая также должна присутствовать в конце строки ($).

С правой стороны определено действие, использующее дополнительный текст, указанный после REJECT. В нашем примере в нем указаны ссылки на два фрагмента совпадающего текста: $1 (первый элемент для совпадения – (.*)) и $2 (расширение имени файла).

Если кто-то отправит вложение с именем image.pif, то строка MIME-заголовка в сообщении будет выглядеть так:

```
filename="image.pif"
```

и для него будет создано следующее сообщение об ошибке:

```
Unwanted type of attachment image.pif
```

так как $1 соответствует image, а $2 соответствует pif.

Теперь добавьте параметр mime_header_checks в файл main.cf, указав путь к вашей карте:

```
mime_header_checks = pcre:/etc/postfix/mime_header_checks
```

После перезагрузки конфигурации параметр mime_header_checks начнет действовать.

Проверка заголовков во вложенных сообщениях

Postfix может выполнять особые действия для заголовков сообщений, вложенных в сообщение. По умолчанию на них будет действовать любой из параметров header_checks, но если вы хотите создать отдельную карту (для экономии процессорного времени или для создания исключений), то можете определить ее в параметре nested_header_checks.

Как и для других видов проверок, для хранения шаблонов вам нужно создать отдельный файл карты, например /etc/postfix/nested_header_checks. В нашем примере в журнал будет записываться предупреждение с идентификатором сообщения из вложенного заголовка:

```
/^Message-Id:/   WARN Nested Message-Id:
```

Теперь добавим параметр `nested_header_checks` в файл `main.cf`:

```
nested_header_checks = pcre:/etc/postfix/nested_header_checks
```

После перезагрузки конфигурации в журнале должны появиться записи такого вида:

```
Apr 14 13:17:55 mail postfix/cleanup[32397]: 59C3115CO2A: warning: header
   Message-ID: <DIDL27HL1L4H87CA@example.com>
   from mgate22.so-net.ne.jp[210.139.254.169];
   from=<> to=<recipient@example.com> proto=ESMTP
   helo=<mgate22.so-net.ne.jp>: Nested Message-Id:
```

Примечание

На самом деле приведенный пример бесполезен – ни здесь, ни в списке рассылки невозможно привести реальный сценарий. Если вам необходим параметр `nested_header_checks`, вы, вероятно, об этом узнаете.

Проверка тела сообщения

Просмотр частей тела письма полезен, если вам необходимо найти шаблон внутри тела для инициирования действия. Как и в случае с другими проверками, вы исследуете содержимое на соответствие заданному шаблону, используя параметр `body_checks` в сочетании с картой, хранящей шаблоны и вызываемые действия.

Предупреждение

Проверки тела применяются ко всем сообщениям, входящим и исходящим, и ко всем отправителям и получателям. Поэтому они действуют и для сообщений, отправленных на адрес `abuse` и `postmaster`.

При реализованной проверке на спам пользователи, жалующиеся на спам, который предположительно был отправлен из вашей сети, не смогут сообщить об этом адресатам `abuse` и `postmaster`, если в их жалобе содержится само нежелательное сообщение. В текущей версии Postfix нет возможности отменить проверки для отдельных пользователей.

Начинаем с обычного файла карты, например `/etc/postfix/body_checks`, пусть он содержит такие шаблоны и действия:

```
# Пропустить блоки, закодированные в base 64. Это сэкономит много
процессорного времени.
# Автор Liviu Daia; изменения Victor Duchovni.
~^[[:alnum:]+/]{60,}\s*$~       OK
```

Шаблон соответствует блокам в кодировке base64. Обратите внимание на то, что для определения границ регулярного выражения использована тильда (~), а не привычная косая черта (/), так что нет необходимости в «экранировании» косой черты внутри регулярных выражений.

Приведем несколько шаблонов спама, как известных, так и уникальных, которые будут отвергнуты Postfix:

```
# SPAM
/(AS SEEN ON NATIONAL TV|READ THIS E-MAIL TO THE END)/
    REJECT Spam #1
/We are shanghai longsun electrical alloy/
    REJECT Chinese spammer from hell
/Do you want EVERYONE to know your business/
    REJECT Spam #2
/(Zainab|San?ni) Abacha/
    REJECT Nigeria spam
/MILITARY HEAD OF STATE IN NIGERIA/
    REJECT Nigeria fraud spam
/antivirus\.5xx\.net/
    REJECT Virus hoax (0190-dialer)
/MOSE CHUKWU/
    REJECT Business fraud spam #1
/Ahmed Kabbah/
    REJECT Business fraud spam #2
/Godwin Igbunu/
    REJECT Business fraud spam #3
/I PRESUME THIS EMAIL WILL NOT BE A SURPRISE TO YOU/
    REJECT Business fraud spam #4
/http:\/\/www\.a1-opportunity4u\.com\/euro2/
    REJECT Business fraud spam #5
/http:\/\/66.151.240.30\//
    REJECT Spam of the worst kind
/http:\/\/members.tripod.com.br\/lev3irkd/
    REJECT Spam of the worst kind II
```

Сообщения, содержащие следующие шаблоны, будут отклонены; отправитель конверта получит уведомление о возврате с указанием на базу данных подложных сообщений (hoax database).

```
# Hoaxes
/jdbgmgr\.exe/
    REJECT Virus hoax!
/ready to dictate a war/
    REJECT Hoax: http://www.tu-berlin.de/www/software/hoax/unicwash.shtml
/inquiries@un\.org/
    REJECT Hoax: http://www.tu-berlin.de/www/software/hoax/unicwash.shtml
/UNO is ready to receive signatures/
    REJECT Hoax: http://www.tu-berlin.de/www/software/hoax/unicwash.shtml
/Third World War/
    REJECT Hoax: http://www.tu-berlin.de/www/software/hoax/unicwash.shtml
```

В некоторых случаях проверка `body_checks` может быть полезна в качестве оперативного средства для отклонения вредоносных сообщений, если ваш антивирус еще не научился распознавать их. Не забудьте удалить шаблон, как только ваша программа сможет разобраться с вирусами:

```
## Virus
# Win32.Netsky.V
/The processing of this message can take a few minutes\.\.\./
    REJECT Win32.Netsky.V
/Converting message. Please wait\.\.\./
    REJECT Win32.Netsky.V
/Please wait while loading failed message\.\.\./
    REJECT Win32.Netsky.V
/Please wait while converting the message\.\.\./
    REJECT Win32.Netsky.V
```

Когда файл карты готов, добавляем параметр `body_checks` **в файл** `main.cf`**:**

```
body_checks = regexp:/etc/postfix/body_checks
```

Как и ранее, вам необходимо перезагрузить конфигурацию Postfix для активации проверки тела сообщений.

11

Как работают внешние фильтры содержимого

Будьте либералом при приеме и консерватором при отправке.

– Джон Постел, пионер Интернета (1943–1998)

Описанные в предыдущих главах встроенные фильтры предназначены для решения простых проблем; более сложную фильтрацию поручают внешним программам.

Postfix может запускать приложения проверки содержимого до или после постановки сообщений в очередь. Если почта фильтруется до постановки в очередь, то Postfix может оставить уведомление отправителей на усмотрение клиента. Если же почта фильтруется после постановки в очередь, ответственность за уведомления несет Postfix.

В этой главе в общих чертах описан процесс передачи полномочий. Вы увидите, как настроить архитектуру демонов Postfix для отправки сообщений внешним механизмам фильтрации и как позволить сообщениям вернуться в систему Postfix после успешной фильтрации для окончательной доставки.

Внешние фильтры содержимого вступают в дело, когда со сцены уходят встроенные фильтры заголовков и тела сообщения; внешние приложения не только исследуют и отвергают сообщения, но и могут изменять их содержимое. Приведем несколько стандартных задач для фильтров:

- Добавление «отказа от ответственности» (disclaimer)
- Проверка на наличие вирусов и червей
- Выявление спама
- Архивирование почты

Postfix поддерживает два механизма фильтрации, `content_filter` и `smtpd_proxy_filter`, которые близки по духу, но отличаются своими возможностями и способами обработки содержимого. Различия между фильтрами показаны в табл. 11.1

Таблица 11.1. Различия между фильтрами

Название фильтра	Транспорт	Момент отклонения
`content_filter`	`smtp`, `lmtp`, `pipe`	После постановки в очередь
`smtpd_proxy_filter`	`smtp`	До постановки в очередь

В данной главе эти отличия будут рассмотрены подробно, что поможет вам решить, какой из фильтров лучше подходит в вашей конкретной ситуации.

Наилучший момент для фильтрации содержимого

Стандарты RFC говорят о том, что почтовый сервер должен решить, принять или отвергнуть сообщение, не позднее, чем на этапе команды `DATA` в диалоге SMTP. К сожалению, такое требование оставляет почтовому серверу мало времени на анализ содержимого сообщения, т. к. в почтовых клиентах реализован достаточно короткий тайм-аут с тем, чтобы не «зависнуть» в общении с неисправным почтовым сервером. Например, тайм-аут для SMTP-клиента Postfix определяется параметром `smtp_data_done_timeout`, который весьма толерантен и по умолчанию установлен в 600 секунд.

Если почтовый сервер завершает просмотр содержимого *до* истечения клиентского тайм-аута, все работает отлично, т. к. у сервера есть время на уведомление клиента о своем решении в отношении приема сообщения. Однако если сервер работает слишком медленно, то клиент заканчивает соединение и повторяет попытку позже, при этом шансы на успех при следующей попытке невелики.

Имеющийся в Postfix механизм `content_filter` позволяет избежать таких проблем благодаря специальной организации исследования содержимого:

1. Почтовый клиент отправляет содержимое на этапе `DATA`.
2. Сервер Postfix принимает сообщение и ставит его в очередь. Клиент предполагает, что передача была успешной.
3. Диспетчер очередей анализирует почту и составляет расписание доставки согласно записям `content_filter`.
4. Postfix передает сообщение внешнему приложению.
5. Внешнее приложение берет на себя доставку сообщения. Оно может выполнить для сообщения одно из следующих действий:

- Принять сообщение и передать его обратно Postfix для доставки.
- Принять сообщение и передать его другому приложению или серверу.
- Удалить или вернуть сообщение.

Второй фильтр (`smtpd_proxy_filter`) обращается с сообщениями по-другому:

1. Почтовый клиент отправляет содержимое на этапе `DATA`.
2. Postfix-демон `smtpd` передает команды SMTP и содержимое сообщения внешнему приложению.
3. Внешнее приложение отправляет SMTP-ответы обратно демону `smtpd`, а тот передает их почтовому клиенту.

У этого фильтра могут возникнуть проблемы из-за тайм-аутов почтового клиента, кроме того, он не масштабируется для большого количества одновременных соединений с клиентами. Дело в том, что в `smtpd_proxy_filter` отсутствует механизм очередей для планирования фильтрации содержимого. А без такого механизма внешнему приложению приходится начинать работу сразу же при получении Postfix каждого нового сообщения. В результате может наблюдаться значительное замедление работы, если внешнее приложение не сможет работать в темпе поступления сообщений. Такая проблема может возникнуть с приложениями, выявляющими спам или проверяющими наличие вирусов, которым часто приходится выполнять требующие больших затрат времени операции распаковки и раскодирования вложений.

Оба подхода имеют свои недостатки:

- `content_filter` генерирует дополнительный трафик, т. к. Postfix первоначально принимает сообщения перед обработкой. Это может впоследствии привести к необходимости возврата, если фильтрующее приложение решит, что сообщение должно быть отклонено.
- `smtpd_proxy_filter` сразу же отвергает нежелательное содержимое, но он плохо масштабируется и может оказаться недостаточно производительным.

Фильтры и перезапись адресов

При перезаписи адресов в заголовке сообщения вам необходимо подумать о том, как применять фильтры. В частности, следует решить, когда вы будете просить Postfix перезаписывать адреса (например, посредством `virtual_alias_maps`) – до или после фильтрации.

Если вы примете решение перезаписывать адреса перед фильтрацией, то появится риск использования внутренних адресов для возвратов и предупреждений. Например, предупреждение, вызванное сообщением адресату `moe_helden@example.com`, может быть возвращено с адресом `mh123@mailbox.example.com`.

Поэтому с нашей точки зрения следует перезаписывать адреса (используя `virtual_alias_maps` или `canonical_maps`) *после* возвращения сообщений обратно в очередь Postfix для финальной доставки. Это позволит внешнему приложению (например, антивирусной программе) видеть исходные адреса и формировать соответствующие предупреждения до того, как Postfix их заменит.

Существуют два способа отмены преобразования адресов (расширения виртуального псевдонима, канонического преобразования, трансляции адреса и т. д.) перед фильтрацией. Первый заключается в установке специального параметра в файле `main.cf`:

```
receive_override_options = no_address_mappings
```

Вы также можете отключить перезапись адресов в файле `master.cf` только для принимающего сообщения из сети демона (обычно это `smtpd`):

```
smtp      inet n       -       n       -       -       smtpd
  -o content_filter=foo:[127.0.0.1]:54321
  -o receive_override_options=no_address_mappings
  ...
```

После того как фильтр обрабатывает сообщение, оно обычно вставляется обратно в очередь Postfix. Это как раз подходящий момент для манипуляций с адресами; вам потребуется дополнительный демон `smtpd`, принимающий отфильтрованную почту. Вместо того чтобы использовать настройку `receive_override_options=no_address_mappings`, этот дополнительный `smtpd` будет использовать `receive_override_options=no_unknown_recipient_checks`. В последующих разделах мы рассмотрим работу фильтров `content_filter` и `smtpd_proxy_filter` более подробно.

content_filter: сначала постановка в очередь, затем фильтрация

Для настройки почтовой службы с использованием механизма `content_filter` вам потребуются два экземпляра `smtpd` (рис. 11.1). Первый демон `smtpd` принимает нефильтрованные сообщения и использует `content_filter` для передачи сообщений внешнему фильтрующему приложению. Второй экземпляр `smtpd` занимается прослушиванием соединений от внешних приложений, чтобы сообщения могли вновь попасть в систему очередей Postfix для последующей обработки.

Предупреждение

При настройке второго экземпляра учтите, что он не должен использовать указанное в `content_filter` приложение, запускаемое первым экземпляром. Возник бы бесконечный цикл, в котором Postfix отправлял бы сообщение фильтрующему приложению, и это сообщение возвращалось бы на то же место, что и раньше, в очереди Postfix.

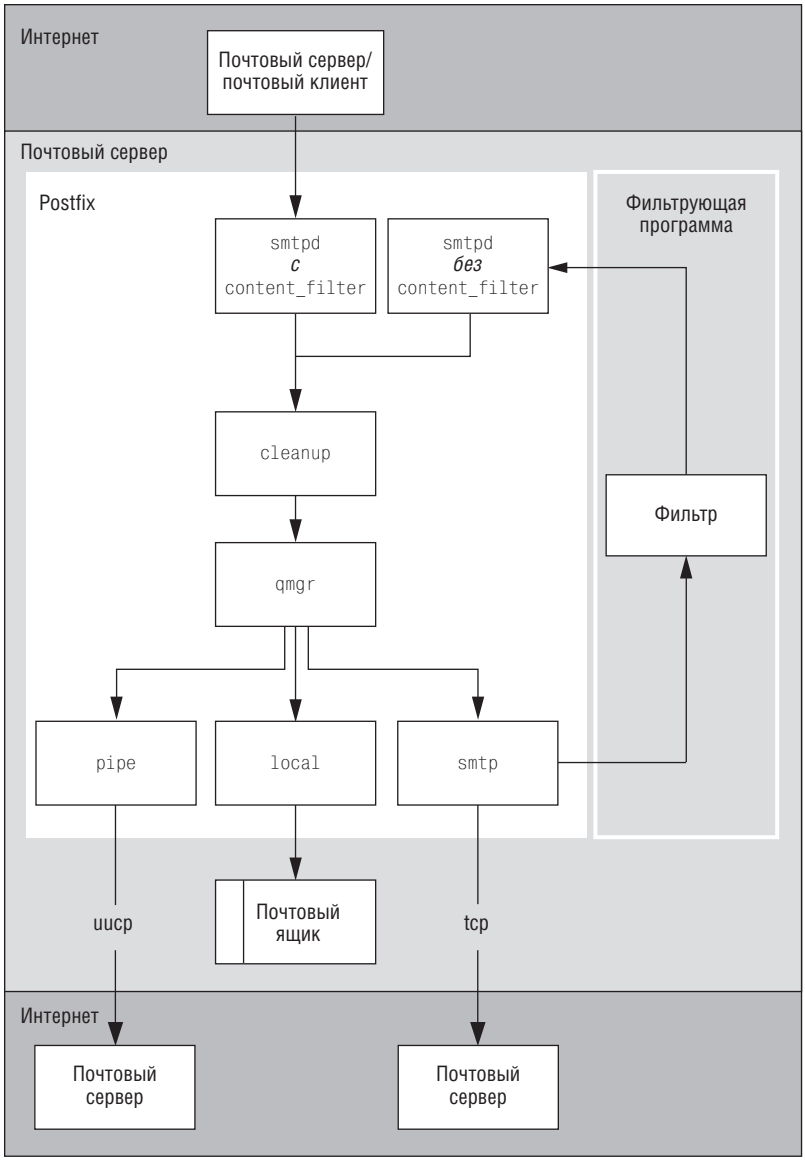

***Рис. 11.1.** Процесс доставки с использованием механизма content_filter*

Вот как это работает:

1. Экземпляр `smtpd`, настроенный с `content_filter`, передает сообщение демону `cleanup`, который передает его диспетчеру очередей.
2. Диспетчер очередей передает сообщение демону `smtp`, `lmtp` или `pipe` для доставки его внешнему фильтрующему приложению.

Примечание

На рис. 11.1 изображен один из трех возможных сценариев работы, когда диспетчер очередей `qmgr` передает почту демону `smtp`.

3. Внешнее фильтрующее приложение принимает сообщение и обрабатывает его.
4. Если фильтрующая программа передает сообщение обратно в систему Postfix, она подключается к Postfix-демону `smtpd`, настроенному для работы без использования `content_filter`.
5. Второй экземпляр `smtpd` передает сообщение диспетчеру очередей.
6. Postfix доставляет сообщение локально или пересылает его на другой почтовый сервер.

В дополнение к сообщению Postfix может отсылать внешнему фильтрующему приложению дополнительную информацию. Какая именно информация передается, зависит от того, какой демон (`lmtp`, `smtp` или `pipe`) Postfix вызывает для отправки сообщения приложению.

Демоны, передающие сообщения фильтрам

При работе с механизмом `content_filter` в вашем распоряжении имеются три основных демона, которые могут передавать сообщения внешнему фильтрующему приложению. Демоны отличаются друг от друга тем, что они могут делать и передавать:

`pipe`

Демон `pipe` отправляет сообщения сценариям и другим исполняемым программам. Они могут выполнять практически любые вообразимые действия, начиная с архивирования сообщений и заканчивая различными видами автоматической обработки содержимого сообщения, как, например, выявление вирусов.

Неограниченный диапазон задач, решаемых этими программами, вызывает необходимость передачи фильтрующей программе (наряду с сообщением) различных аргументов и флагов. Об этих аргументах вы можете прочитать на странице руководства pipe(8).

`smtp`

Вы можете использовать Postfix-демон `smtp` для передачи сообщения фильтрующему приложению по SMTP или ESMTP (например, другому агенту передачи сообщений). Информация, которая может отправляться вместе с сообщением, определяется протоколом; более подробные сведения вы найдете на странице руководства smtp(8).

`lmtp`

Postfix-демон `lmtp` может отправлять сообщения для фильтрации по протоколу LMTP. Как и в случае SMTP-клиента, протокол LMTP накладывает ограничение на объем дополнительной информации,

которая может быть передана с сообщением (вы можете прочитать об этом на странице руководства lmtp(8)).

Примечание

В отличие от SMTP-клиента Postfix, в котором в настоящее время не реализована нотификация о статусе доставки (DSN – Delivery Status Notification) для формирования отдельных уведомлений, протокол LMTP позволяет серверу отправлять каждому получателю отчеты о состоянии сообщения (т. е. отчеты о том, отвергнуто или принято сообщение для каждого получателя). Это позволяет избежать путаной нотификации о состоянии для нескольких получателей в случае, когда сообщение принимается для некоторых, но не для всех адресатов.

Основы настройки content_filter

Для отправки сообщений внешней программе фильтрации при помощи механизма `content_filter` вам необходимо изменить поведение всех демонов, которые занимаются входящей почтой. В частности, вам предстоит сделать следующее:

1. Определить `content_filter` в качестве транспорта в вашем файле `main.cf`.
2. Настроить транспорт в `master.cf`.
3. Настроить в `master.cf` дополнительный путь для возвращения сообщений в систему после фильтрации.

Определение транспорта

Для того чтобы сообщить серверу Postfix о необходимости передачи сообщений внешнему приложению, используйте `content_filter` в файле `main.cf`. Надо сообщить серверу Postfix, что он должен отправлять все сообщения своей службе (которую еще предстоит создать), которая будет передавать сообщения фильтрующему приложению. Например, следующая строка указывает серверу Postfix, что следует отправить сообщения транспорту с именем `foo` через порт 54 321 локального хоста (`127.0.0.1`). Помните, что использование квадратных скобок препятствует поиску MX-записи для хоста `127.0.0.1`:

```
content_filter = foo:[127.0.0.1]:54321
```

Настройка транспорта

Теперь нужно настроить транспортную службу, которая только что была создана в файле `main.cf`. Файлом конфигурации транспортной службы является `master.cf`, т. к. именно демон `master` должен иметь всю информацию о доступных службах. Вы должны предоставить демону `master` следующие сведения о новой транспортной службе:

1. Имя службы.
2. Имя демона Postfix, который будет осуществлять транспорт.

3. Параметры и другие данные, которые необходимы программе для работы.

Приведем пример создания транспорта `foo`:

```
#==========================================================
# service type        private unpriv chroot wakeup maxproc command
#                      (yes)  (yes)   (yes) (never) (100)
# ==========================================================
...
foo       unix        -      -        n     -      2       smtp
  -o smtp_data_done_timeout=1200s
  -o disable_dns_lookups=yes
...
```

Строка, начинающаяся с `foo`, – это основная конфигурация, содержащая восемь столбцов. Первый должен соответствовать имени транспорта, определенному в файле `main.cf`. Столбец команды содержит команду, которая будет отправлять сообщения фильтрующему приложению (в данном случае это SMTP-клиент Postfix). Две последующие строки содержат параметры команды.

Предупреждение

Синтаксис параметров команды: при перечислении дополнительных параметров команды добавляйте пробел в начало каждой новой строки, содержащей параметры, т. к. строка, начинающаяся с пробела, продолжает логическую строку. Однако от пробелов между параметрами и значениями (как в примере между `smtp_data_done_timeout` и `1200s`) вы должны избавиться; в противном случае они не будут распознаны и восприняты демоном `master`.

Пока все отлично – вы можете передавать сообщения внешним приложениям. Однако если приложение должно отправлять сообщения обратно серверу Postfix, вам следует настроить путь для возврата сообщений.

Настройка дополнительного пути для возврата сообщений

Путь возврата сообщений – это просто локальный метод Postfix, возвращающий сообщения в систему (как SMTP, LMTP или локальная передача посредством `sendmail`) без применения `content_filter`. Обычно используется второй экземпляр демона `smtpd`, запущенный со специальными параметрами, указанными в файле `main.cf`. Например, если вы хотите, чтобы дополнительный демон возврата `smtpd` прослушивал порт 10 025, сделайте следующие настройки в файле `master.cf`:

```
#==========================================================
# service type        private unpriv chroot wakeup maxproc command
#                      (yes)  (yes)   (yes) (never) (100)
# ==========================================================
...
127.0.0.1:10025  inet n       -      n      -      -       smtpd
  -o content_filter=
```

```
-o receive_override_options=no_unknown_recipient_checks
-o smtpd_recipient_restrictions=permit_mynetworks,reject
-o mynetworks=127.0.0.0/8
...
```

Интернет-службы могут быть заданы в виде `хост:порт` без указания транспорта в файле `main.cf` (хост может быть задан именем или IP-адресом, определенным в файле `/etc/hosts`, а порт может быть представлен числом или именем службы, определенной в файле `/etc/services`).

Вы можете пропустить часть `хост:`, но тогда служба будет доступна всем сетевым интерфейсам, определенным в `inet_interfaces`. Для того чтобы свести к минимуму риск создания открытого почтового сервера посредством настройки пути возврата, вы должны ограничить количество прослушивающих сетевых интерфейсов и разрешить прослушивание лишь тем, которые вам необходимы, а в данном случае вам необходим лишь `localhost/127.0.0.1`.

У команды имеются дополнительные параметры:

- Явно заданный пустой параметр `content_filter` отменяет использование фильтра в файле `main.cf`, так что вы не попадете в бесконечный цикл фильтрации.
- Параметр `receive_override_options` отменяет проверку получателей по картам `local_recipient_maps` и `relay_recipient_maps`; эти проверки уже были проведены демоном `smtpd`, принявшим почту из Интернета, так что нет необходимости в их повторении.
- Два последних параметра работают вместе, сначала разрешая поступление сообщений только от хостов из параметра `mynetworks`, а затем явно устанавливая значение `mynetworks` в `127.0.0.0/8`. Это дополнительная защита от внешних хостов, пытающихся воспользоваться вашим путем возврата сообщений в систему как открытым почтовым сервером.

smtpd_proxy_filter: сначала фильтрация, затем постановка в очередь

Для использования механизма `smtpd_proxy_filter` вам необходимо изменить существующий демон `smtpd` (`smtpd` «до фильтрации») для передачи соединений от почтовых клиентов внешним фильтрующим программам (рис. 11.2).

Примечание

Postfix-демон `smtpd` защищает внешнее приложение, исключая такие потенциально опасные эффекты, как конвейеризация, длинные аргументы и неразрешенные символы, которые могут появиться из соединения.

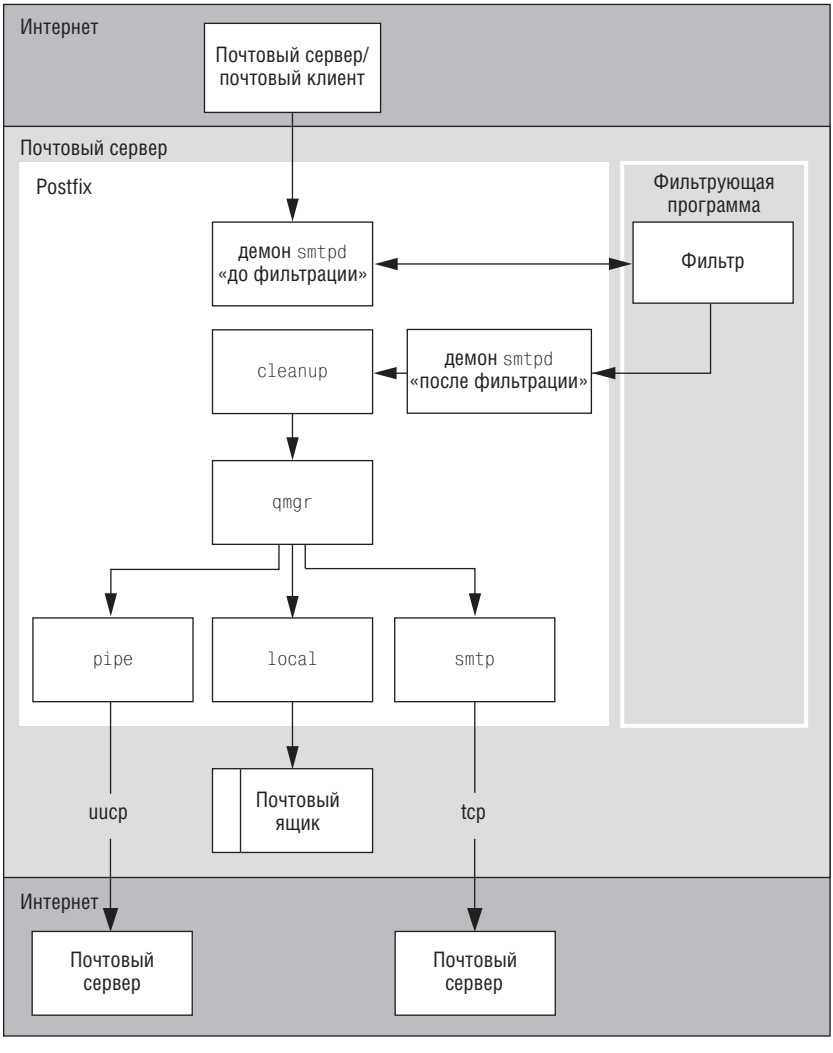

Рис. 11.2. *Процесс доставки с проходом через фильтр-посредник*

В зависимости от используемого фильтрующего приложения вам может потребоваться создание второго экземпляра `smtpd` (`smtpd` «после фильтрации»), который будет прослушивать соединения, ожидая сообщений, отправленных обратно внешним фильтрующим приложением.

Посмотрим, как работает этот механизм:

1. Демон `smtpd` «до фильтрации» подключается к внешнему приложению.
2. Демон `smtpd` передает входящие команды SMTP и данные внешнему приложению.

3. Внешнее фильтрующее приложение держит соединение открытым, пока обрабатывает содержимое сообщения.
4. Если фильтрующее приложение принимает сообщение, оно может передать его демону `smtpd` «после фильтрации» или отослать другому приложению.
5. После того как внешнее фильтрующее приложение определится, принять ему сообщение или отклонить, оно отправляет SMTP-ответы (такие как `250 OK` или `554 Reject`) демону `smtpd` «до фильтрации».
6. Демон `smtpd` «до фильтрации» передает эти ответы исходному почтовому клиенту.

Некоторые соображения о фильтрах-посредниках

При работе с `smtpd_proxy_filter` помните о следующих обстоятельствах:

ESMTP-взаимодействие

Postfix отправляет сообщения фильтрующему приложению по ESMTP, но не использует конвейеризацию команд. Postfix-демон `smtpd` генерирует собственные команды `EHLO`, `XFORWARD` (для записи в журнал IP-адреса почтового клиента вместо `localhost[127.0.0.1]`), `DATA` и `QUIT`. В остальном Postfix просто пересылает неизмененные копии команд `MAIL FROM` и `RCPT TO`, которые демон `smtpd` «до фильтрации» получил от удаленного почтового клиента.

Требования к внешнему приложению

Фильтр (который *должен* понимать SMTP) должен использовать тот же синтаксис команд `MAIL FROM` и `RCPT TO`, что и демон `smtpd` в Postfix.

Возврат сообщений в систему

Предполагается, что фильтрующее приложение передаст неизмененные команды SMTP от `smtpd` «до фильтрации» экземпляру `smtpd` «после фильтрации» (который обычно прослушивает нестандартный порт для возврата сообщений по пути, не использующему тот же самый фильтр; все происходит аналогично случаю применения механизма `content_filter`, который обсуждался ранее в этой главе).

Отклонение содержимого

Если фильтр отвергает содержимое, он должен отправить отрицательный SMTP-ответ (код 5xx) обратно демону `smtpd` «до фильтрации», а затем прервать соединение с демоном `smtpd` «после фильтрации», не завершая SMTP-диалог с ним. В противном случае демон «после фильтрации» может случайно доставить сообщение.

Основы настройки smtpd_proxy_filter

Для отправки сообщений внешнему фильтру при помощи механизма `smtpd_proxy_filter` вам нужно изменить поведение демона `smtpd`. Следует выполнить два шага:

1. Изменить существующий демон `smtpd` (будем называть его демоном «до фильтрации»).
2. Настроить дополнительный экземпляр `smtpd` для возврата отфильтрованной почты обратно в очередь Postfix; это будет демон «после фильтрации».

Изменение существующего smtpd (демон «до фильтрации»)

Для того чтобы существующий демон `smtpd` передавал соединения фильтрующему приложению, добавьте параметр `smtpd_proxy_filter` в определение службы `smtpd` в файле `master.cf`. Необходимо указать IP-адрес или полностью определенное доменное имя (FQDN) и порт фильтра-посредника.

В следующем примере используется порт 10 024 локального хоста:

```
#==========================================================
# service type        private unpriv chroot wakeup maxproc command
#                      (yes)  (yes)   (yes) (never) (100)
# ==========================================================
...
smtp      inet           n      -       n      -      20     smtpd
   -o smtpd_proxy_filter=localhost:10024
```

Настройка дополнительного экземпляра smtpd (демон «после фильтрации»)

Для создания еще одного экземпляра `smtpd`, который будет принимать на локальном хосте отфильтрованные сообщения, вам необходимо добавить еще одну строку в файл `master.cf`. Все настройки выполняются аналогично первому `smtpd`, только второй будет прослушивать другой порт и для него не будет задан параметр передачи сообщений фильтру, как для демона «до фильтрации». Приведем пример настройки `smtpd` «после фильтрации», прослушивающего порт 10 025:

```
#==========================================================
# service type        private unpriv chroot wakeup maxproc command
#                      (yes)  (yes)   (yes) (never) (100)
# ==========================================================
...
127.0.0.1:10025    inet n       -      n      -      -       smtpd
    -o smtpd_authorized_xforward_hosts=127.0.0.0/8
    -o smtpd_client_restrictions=
    -o smtpd_helo_restrictions=
    -o smtpd_sender_restrictions=
    -o smtpd_recipient_restrictions=permit_mynetworks,reject
    -o mynetworks=127.0.0.0/8
    -o receive_override_options=no_unknown_recipient_checks
    -o content_filter=
```

12

Использование внешних фильтров содержимого

Если вам нужно, я знаю, где это можно достать.

– Первый тюремщик в фильме «Жизнь Брайана по Монти Пайтону» («Monty Python's Life of Brian»)

У каждого инструмента – свое назначение. Представьте, что вы пытаетесь создать молоток, который можно было бы использовать и для полировки. Скорее всего, в итоге получился бы плохой молоток, который бы плохо полировал. Именно по этой причине Postfix не занимается фильтрацией спама, архивированием или очисткой почты. Вместо этого он предоставляет вам возможность подключить лучший из доступных внешний фильтр к лучшему из имеющихся агенту передачи сообщений. В предыдущих главах вы узнали о том, что Postfix предлагает несколько подходов к фильтрации почты, отличающихся тем моментом, когда обрабатываются входящие сообщения. В данной главе обсуждается применение этих приемов на практике.

В частности, вы узнаете о том, как присоединять к сообщениям отказы от ответственности (disclaimer), передавая сообщения на обработку сценариям (через pipe), и как проверять сообщения на вирусы, отправляя их программе amavisd-new посредством механизма `content_filter` или `smtpd_proxy_filter`.

Присоединение к сообщению отказа от ответственности при помощи сценария

Одной из многочисленных задач, которые можно решить посредством `content_filter`, является добавление отказа от ответственности во все

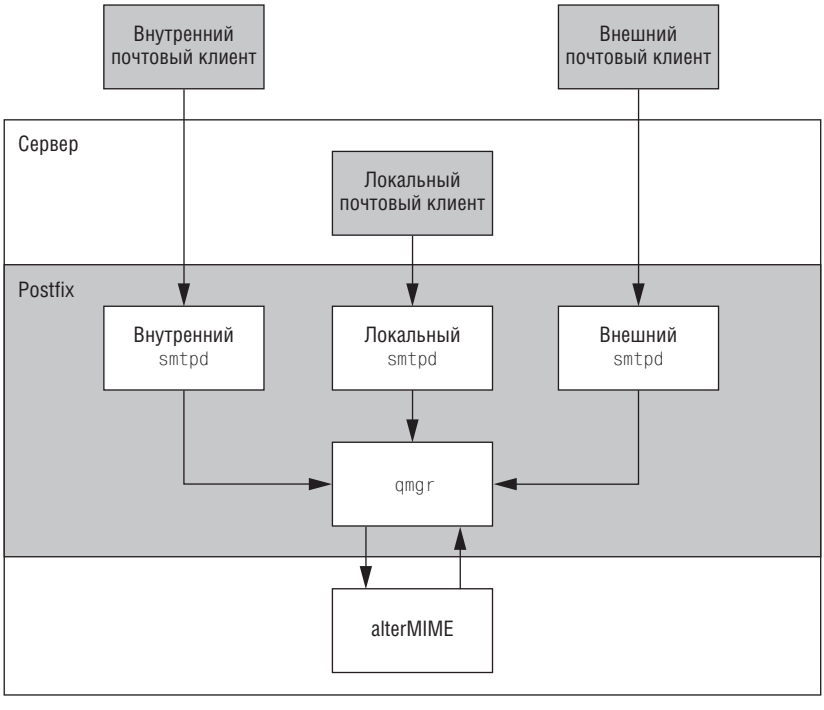

Рис. 12.1. *Интеграция alterMIME в систему Postfix*

исходящие сообщения. В следующем примере для добавления отказа от ответственности в каждое сообщение, отправляемое от внутренних клиентов, используется alterMIME – небольшая программа, изменяющая сообщения в MIME-кодировке. На рис. 12.1 показано, как alterMIME интегрируется в процесс передачи сообщения.

Для того чтобы добавить отказы от ответственности в исходящие сообщения, не затрагивая входящие и локальные сообщения, необходимо разделить трафик по направлениям. Будем считать, что ваш почтовый сервер поддерживает отдельные сетевые интерфейсы для внешней и внутренней сетей. Это означает, что вам нужно будет создать три отдельных экземпляра демона `smtpd` и связать их с сетевыми интерфейсами внешней и внутренней сети и локального хоста. В следующем примере показано, как передача сообщения из вашей внутренней сети удаленному получателю будет обработана в случае наличия отдельных экземпляров `smtpd` для разных сетевых интерфейсов.

1. Когда сообщение покидает вашу сеть, почтовый клиент подключается к экземпляру `smtpd`, прослушивающему внутренний интерфейс.
2. Этот внутренний `smtpd` принимает сообщение и отправляет его диспетчеру очередей `qmgr`.
3. `qmgr` отправляет сообщение службе `content_filter`.

4. Служба `content_filter` использует демон `pipe` для передачи сообщений сценарию.
5. Сценарий добавляет отказ от ответственности.
6. Сценарий возвращает сообщение экземпляру `smtpd`, прослушивающему локальный интерфейс.
7. Локальный `smtpd` отправляет заново полученное сообщение диспетчеру очередей `qmgr`.
8. `qmgr` отправляет сообщение `smtpd` и далее в Интернет.

Однако, прежде чем настраивать транспорт, вы должны создать сценарий, который будет вызывать alterMIME из Postfix.

Установка alterMIME и создание сценария фильтра

Сценарий будет запускать программу alterMIME (*http://www.pldaniels.com/altermime*) для изменения исходящего сообщения. Если в вашем распоряжении нет alterMIME (как нет и бинарного пакета для вашей операционной системы), скачайте ее, распакуйте архив, перейдите в каталог исходных текстов и запустите `make` и `make install`. В результате вы должны получить исполняемый файл alterMIME в каталоге `/usr/local/bin/altermime`.

Создание окружения alterMIME

Вы должны запускать alterMIME от имени непривилегированного пользователя системы. Например, если вы бы хотели использовать на своей машине имя пользователя `filter`, то для создания пользователя выполните такие команды:

```
# groupadd filter
# useradd -d /var/spool/altermime -G filter filter
```

Создание каталога сценариев

Загромождать свой каталог `/etc/postfix` сценариями — не слишком удачная мысль. Создайте как суперпользователь отдельный подкаталог для хранения своих сценариев и сделайте этот подкаталог доступным только для пользователей `filter` и `root`.

Например, предложенная последовательность команд создает каталог с требуемыми привилегиями и владельцем:

```
# mkdir /etc/postfix/filter
# chown root /etc/postfix/filter
# chgrp filter /etc/postfix/filter
# chmod 770 /etc/postfix/filter
```

Создание сценария

Следующий сценарий, названный `/etc/postfix/filter/add_disclaimer.sh`, вызывает alterMIME для исходящего сообщения, полученного из Post-

fix (отправленного из демона pipe). Программа alterMIME добавляет в сообщение отказ от ответственности и возвращает его обратно в очередь Postfix. Программе alterMIME требуется место для записи временного файла, она не может работать с stdin.

```
#!/bin/sh
# настройки, зависящие от системы
ALTERMIME=/usr/local/bin/altermime
ALTERMIME_DIR=/var/spool/altermime
SENDMAIL=/usr/sbin/sendmail
# Ожидается, что коды завершения команд, вызываемых Postfix,
# будут придерживаться правил, определенных в <sysexits.h>.
TEMPFAIL=75
UNAVAILABLE=69
# Перейти в рабочий каталог alterMIME и уведомить
# Postfix в случае неудачи 'cd'.
cd $ALTERMIME_DIR || { echo $ALTERMIME_DIR does not exist; exit $TEMPFAIL; }
# Очистка по завершении или прерывании работы.
trap "rm -f in.$$" 0 1 2 3 15
# Запись почты во временный файл.
# Уведомление Postfix в случае неудачи.
cat >in.$$ || { echo Cannot write to $ALTERMIME_DIR; exit $TEMPFAIL; }
# Вызываем alterMIME, передаем ей сообщение и говорим, что с ним делать.
$ALTERMIME --input=in.$$ \
           --disclaimer=/etc/postfix/filter/disclaimer.txt \
           --disclaimer-html=/etc/postfix/filter/disclaimer.txt \
           --xheader="X-Copyrighted-Material: Please visit http:// \
           www.example.com/message_disclaimer.html" || \
           { echo Message content rejected; exit $UNAVAILABLE; }
# Вызов sendmail для возврата сообщения обратно в Postfix
$SENDMAIL -i "$@" <in.$$
# Используем код завершения sendmail, чтобы сообщить
# Postfix, как все прошло.
exit $?
```

После создания сценария предоставляем доступ на запись только пользователю root, но даем права на исполнение пользователю filter:

```
# chown root add_disclaimer.sh
# chgrp filter add_disclaimer.sh
# chmod 750 add_disclaimer.sh
```

Естественно, теперь нам нужно создать отказ от ответственности, на который мы ссылаемся в сценарии.

Создание отказа от ответственности

Если у вас уже есть готовый отказ от ответственности, поместите его текст в файл /etc/postfix/filter/disclaimer.txt. Если же вы только ищете подходящий текст, то можете посетить сайт *http://www.email-disclaimers.com*, специально посвященный отказам от ответственности и законам, связанным с электронной почтой. В следующем примере будет использован макет текста, взятый с сайта *http://www.lipsum.com*:

```
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nam commodo
lobortis magna. Quisque neque. Etiam aliquam. Nulla tempor vestibulum.
```

Когда текст готов, позволяем чтение нашего отказа от ответственности только группе `filter`:

```
# chgrp filter disclaimer.txt
# chmod 640 disclaimer.txt
```

Сценарий фильтра готов. Теперь нам следует настроить Postfix для использования сценария.

Настройка Postfix: сценарий отказа от ответственности

Настройка Postfix для вызова сценария состоит из двух этапов:

1. Определение параметра `content_filter` для соответствующего демона `smtpd` в файле `master.cf`.
2. Определение транспорта в файле `master.cf`.

Определение параметра content_filter

Как уже говорилось в главе 11, теперь вам следует добавить параметр `content_filter` в файл `main.cf` и задать имя транспорта. Однако это означало бы, что `content_filter` определен глобально, и фильтр применялся бы ко всем процессам, обрабатывающим входящую почту. В данном случае нам это не подходит – мы хотим, чтобы фильтр применялся только к сообщениям, поступающим от внутреннего сетевого интерфейса.

Для сопоставления фильтра только сообщениям, поступающим от внутреннего сетевого интерфейса, добавьте параметр `content_filter` только одному экземпляру `smtpd` в файле `master.cf`. В следующем примере `master.cf` внутренний интерфейс имеет адрес `172.16.0.1`:

```
127.0.0.1:smtp      inet    n       -       n       -       -       smtpd ❶
172.16.0.1:smtp     inet    n       -       n       -       -       smtpd ❷
    -o content_filter=disclaimer:
192.0.34.166:smtp   inet    n       -       n       -       -       smtpd ❸
```

❶ Это локальный экземпляр `smtpd`.

❷ Это экземпляр `smtpd`, который прослушивает сетевой интерфейс внутренней сети.

❸ Это экземпляр `smtpd`, который прослушивает внешнюю сеть.

Обратите внимание на то, что имя транспорта фильтра – `disclaimer` – это не имя сценария. Мы определим этот транспорт в следующем разделе.

Определение транспорта

Теперь определяем транспорт `disclaimer` в файле `master.cf`. Создаем экземпляр транспорта `pipe`, который запускает сценарий `add_disclaimer.sh`. Вот как можно сделать это для приведенного выше сценария:

```
disclaimer  unix  -       n       n       -       -       pipe
   flags=Rq user=filter argv=/etc/postfix/filter/add_disclaimer.sh
   -f ${sender} -- ${recipient}
```

Данное определение запускает демон `pipe` от имени пользователя `filter`, вызывая `add_disclaimer.sh` при поступлении сообщения. Кроме того, сценарию передается отправитель и получатель конверта. Флаг `R` предваряет заголовок сообщения `Return-Path` с адресом отправителя конверта, а флаг `q` «экранирует» пробелы и другие специальные символы в аргументах командной строки `$sender` и `$recipient`.

Примечание

Полный перечень флагов и параметров приведен на странице руководства pipe(8).

Тестирование фильтра

Для проверки фильтра вам нужно будет выполнить перечисленные ниже действия, которые будут подробно рассмотрены в последующих разделах:

1. Отправить сообщение удаленному пользователю через внутренний сетевой интерфейс.
2. Проверить записи о действиях фильтра в почтовом журнале.
3. Проверить наличие отказа от ответственности в отправленном сообщении.

Отправка сообщения удаленному пользователю

Для того чтобы сформировать сообщение, которое Postfix отправит фильтру, используйте telnet для подключения к внутреннему сетевому интерфейсу (где экземпляр `smtpd` должен использовать фильтр). Вот пример сеанса:

```
$ telnet 172.16.0.1 25
Trying 172.16.0.1...
Connected to 172.16.0.1.
Escape character is '^]'.
220 mail.example.com ESMTP Postfix
HELO client.example.com
250 mail.example.com
MAIL FROM: <sender@example.com>
250 Ok
RCPT TO: <recipient@remote-example.com>
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
FROM: Sender <sender@example.com>
TO: Recipient <recipient@remote-example.com>
Subject: Testing disclaimer
```

```
This is a test. There should be text at the bottom of this message
added by a disclaimer script.
.
250 Ok: queued as 3C4D043F2F
QUIT
221 Bye
```

Проверка почтового журнала

Почтовый журнал должен содержать подтверждения деятельности фильтра, как в следующем примере:

```
Mar 12 01:59:53 mail postfix/smtpd[30206]: connect from
  client.example.com[172.16.0.2]
Mar 12 02:00:21 mail postfix/smtpd[30206]: 3C4D043F2F:
  client=client.example.com[172.16.0.2]
Mar 12 02:01:53 mail postfix/cleanup[30209]: 3C4D043F2F:
  message-id=<20040312010021.3C4D043F2F@mail.example.com>
Mar 12 02:01:53 mail postfix/nqmgr[30193]: 3C4D043F2F:
  from=<sender@example.com>, size=444, nrcpt=1 (queue active)
Mar 12 02:01:53 mail postfix/pipe[30213]: 3C4D043F2F:
  to=<recipient@remote-example.com>, relay=disclaimer, delay=92,
  status=sent (mail.example.com) ❶
Mar 12 02:01:53 mail postfix/pickup[30192]: 8421143F2F: uid=100
  from=<sender@example.com> ❷
Mar 12 02:01:53 mail postfix/cleanup[30209]: 8421143F2F:
  message-id=<20040312010021.3C4D043F2F@mail.example.com>
Mar 12 02:01:53 mail postfix/nqmgr[30193]: 8421143F2F:
  from=<sender@example.com>, size=977, nrcpt=1 (queue active)
Mar 12 02:01:55 mail postfix/smtpd[30206]: disconnect from
  client.example.com[172.16.0.2]
Mar 12 02:02:03 mail postfix/smtp[30220]: 8421143F2F:
  to=<recipient@remote-example.com>,
  relay=mail.remote-example.com[212.14.92.89],
  delay=10, status=sent (250 Ok: queued as   56851E1C65) ❸
```

❶ Демон `pipe` использует транспорт `disclaimer` для отправки сообщений сценарию.

❷ Сценарий возвращает сообщение обратно с исходным отправителем конверта.

❸ Демон `smtp` успешно доставляет сообщение получателю конверта.

Проверка наличия в сообщении отказа от ответственности

В качестве последней (и, в общем-то, очевидной) проверки извлекаем сообщение и смотрим, содержит ли оно X-заголовок и отказ от ответственности, которые программа alterMIME должна была добавить в исходящие сообщения. В примере видно, что оба они присутствуют:

```
Return-Path: <sender@example.com>
X-Original-To: recipient@remote-example.com
```

```
Delivered-To: recipient@remote-example.com
Received: from mail.example.com (mail.example.com [192.0.34.166])
  by mail.remote-example.com (Postfix) with ESMTP id 56851E1C65
  for <recipient@remote-example.com>; Fri, 12 Mar 2004 02:01:25 +0100 (CET)
Received: by mail.example.com (Postfix, from userid 100)
  id 8421143F2F; Fri, 12 Mar 2004 02:01:53 +0100 (CET)
Received: from client.example.com (client.example.com [172.16.0.2])by
  mail.example.com+(Postfix) with SMTP id 3C4D043F2Ffor
  <recipient@remote-example.com>; Fri, 12 Mar 2004+02:00:21 +0100 (CET)
From: Sender <sender@example.com>
To: Recipient <recipient@remote-example.com>
Subject: Testing disclaimer
Message-Id: <20040312010021.3C4D043F2F@mail.example.com>
Date: Fri, 12 Mar 2004 02:00:21 +0100 (CET)
X-Copyrighted-Material: Please visit http://www.example.com/
message_disclaimer.html

This is a test. There should be text at the bottom of this message
added by a disclaimer script.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nam commodo
lobortis magna. Quisque neque. Etiam aliquam. Nulla tempor vestibulum.
```

Проверка на вирусы посредством content_filter и amavisd-new

В этом разделе будет описано использование механизма `content_filter`, представленного в главе 11, для интегрирования в Postfix популярной программы amavisd-new. Эта программа связывает почтовый агент с одним или несколькими средствами обнаружения вирусов или спама, такими как SpamAssassin. Программа активно развивается и рекомендована многочисленными администраторами почтовых систем в списке рассылки Postfix.

Примечание

Для обеспечения функциональности обнаружения вирусов вам в дополнение к amavisd-new потребуется по крайней мере одна поддерживаемая антивирусная программа; обратитесь к документации за списком поддерживаемых продуктов.

На рис. 12.2 показано, как Postfix и amavisd-new работают совместно с другими приложениями, такими как средства обнаружения спама и вирусов. Сообщение проделывает следующий путь:

1. Почтовый клиент отправляет сообщение Postfix.
2. `smtpd` принимает сообщение.
3. `smtpd` отправляет сообщение `qmgr`.
4. `qmgr` отправляет сообщение amavisd-new.
5. amavisd-new отправляет сообщение другим приложениям (в данном случае антивирусным программам).

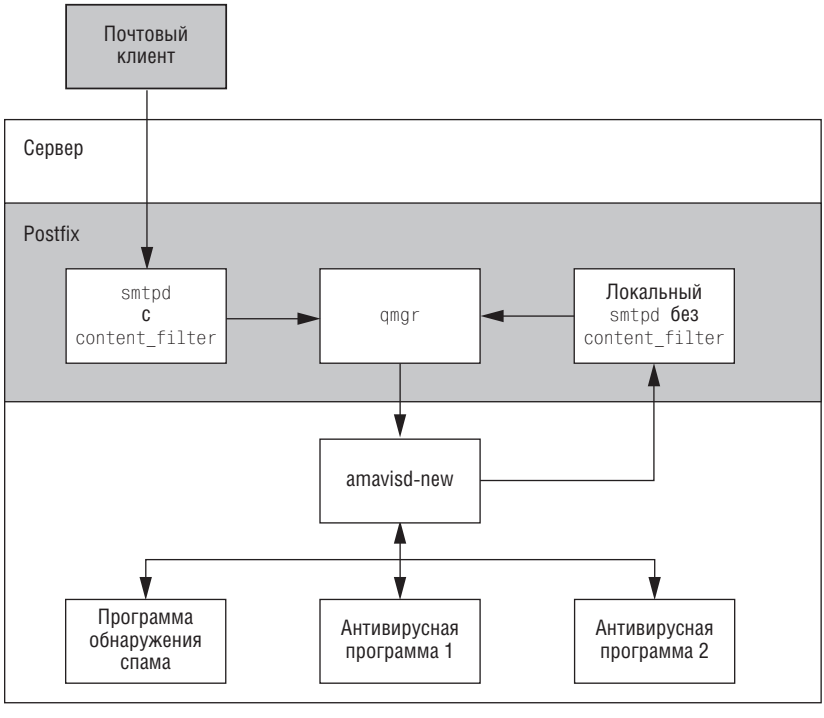

***Рис. 12.2.** Интеграция amavisd-new в систему Postfix при помощи content_filter*

6. amavisd-new возвращает сообщение обратно локальному экземпляру `smtpd`.
7. Локальный `smtpd` отправляет сообщение `qmgr`.
8. `qmgr` или отправляет уведомление о возврате, или доставляет сообщение.

Установка amavisd-new

Вы можете скачать amavisd-new с одного из зеркал, указанных на домашнем сайте (*http://www.ijs.si/software/amavisd*). После распаковки архива следуйте инструкциям в файле `INSTALL` по установке amavisd-new для Postfix.

Вам также следует прочитать файл `README.postfix` (*http://www.ijs.si/software/amavisd/README.postfix*), содержащий обновленные инструкции и специальные замечания для Postfix.

Совет

Вам нужно собрать лишь демон amavisd-new. Вспомогательные приложения, такие как amavis(.c) и amavisd-milter(.c), не требуются для использования в системе Postfix.

Установка модулей Perl для amavisd-new из CPAN

Программе amavisd-new для корректной работы требуется некоторое количество модулей Perl. Документ `INSTALL` в каталоге `SOURCE` для amavisd-new содержит полный перечень таких модулей. При установке модулей обычно предлагается выбор между пакетом, предоставленным создателями вашего дистрибутива, или загрузкой модулей непосредственно из архива CPAN (Comprehensive Perl Archive Network, *http://www.cpan.org*).

Примечание

Обычно CPAN является лучшим источником самых последних версий, но вы можете выбрать и пакеты своей операционной системы для поддержания совместимости.

Для установки таких модулей, как `Compress::Zlib`, из CPAN вам следует запустить Perl с модулем CPAN следующим образом:

```
# perl -MCPAN -e shell;
cpan shell -- CPAN exploration and modules installation (v1.76)
ReadLine support enabled
cpan> install Compress::Zlib
Running install for module Compress::Zlib
Running make for P/PM/PMQS/Compress-Zlib-1.33.tar.gz
Fetching with LWP:
  ftp://ftp-stud.fht-esslingen.de/pub/Mirrors/CPAN/authors/id/P/PM/PMQS/
Compress-Zlib-1.33.tar.gz
CPAN: Digest::MD5 loaded ok
Fetching with LWP:
  ftp://ftp-stud.fht-esslingen.de/pub/Mirrors/CPAN/authors/id/P/PM/PMQS/
CHECKSUMS
Checksum for /root/.cpan/sources/authors/id/P/PM/PMQS/Compress-Zlib-
1.33.tar.gz ok
Scanning cache /root/.cpan/build for sizes
Compress-Zlib-1.33/
... lots of building output ...
All tests successful, 1 test skipped.
Files=6, Tests=287,  2 wallclock secs ( 0.73 cusr +  0.11 csys =  0.84 CPU)
  /usr/bin/make test -- OK
Running make install
Installing /usr/lib/perl5/site_perl/5.6.0/i386-linux/auto/Compress/Zlib/
Zlib.so
Files found in blib/arch: installing files in blib/lib into architecture
dependent library tree
```

```
Installing /usr/lib/perl5/site_perl/5.6.0/i386-linux/Compress/Zlib.pm
Installing /usr/man/man3/Compress::Zlib.3pm
Writing /usr/lib/perl5/site_perl/5.6.0/i386-linux/auto/Compress/Zlib/.packlist
Appending installation info to /usr/lib/perl5/site_perl/5.6.0/i386-linux/
perllocal.pod
  /usr/bin/make install  -- OK
```

После того как необходимые модули получены и amavisd-new установлена, ее необходимо проверить.

Тестирование amavisd-new

Для того чтобы протестировать amavisd-new автономно, прежде чем пытаться обеспечить ее взаимодействие с Postfix, выполните следующие действия:

1. Запустите программу amavisd-new в режиме отладки, чтобы посмотреть, корректно ли она запускается.
2. Выполните сетевой тест, чтобы посмотреть, прослушивает ли она сетевой порт.

Запуск amavisd-new в режиме отладки

Запуск amavisd-new в режиме отладки сразу же дает вам ответы на следующие вопросы:

- Работает ли она? Для запуска amavisd-new должны быть установлены все обязательные модули Perl. Если какого-то модуля не хватает, вы получите сообщение об ошибке, указывающее на отсутствующий модуль.
- Можете ли вы запустить ее от имени непривилегированного пользователя? Для amavisd-new требуется новая группа (по умолчанию – `vscan`) и пользовательская учетная запись в этой группе (также `vscan` по умолчанию).
- Находит ли она необязательные модули Perl, реализующие дополнительную функциональность, такие как SpamAssassin, LDAP и SQL?
- Надлежащую ли версию Perl она использует? Если у вас установлены несколько версий Perl, то может оказаться, что не все модули установлены для той конкретной версии Perl, которую вы пытаетесь использовать.
- Находит ли она вспомогательные программы, такие как антивирусные сканеры?
- Какой файл конфигурации используется? Обычно это `/etc/amavisd.conf`, но вы можете указать и какой-то другой, если точно знаете, что делаете.
- Может ли она связываться с портами, указанными в файле конфигурации?

Для первой попытки лучше всего запустить amavisd-new интерактивно, чтобы она была связана с терминалом. Для этого переключитесь на пользователя `vscan` и запустите amavisd-new с параметром `debug`. Рассмотрим пример ожидаемых выходных данных:

```
# su - vscan
$ /usr/local/sbin/amavisd debug
Jan 28 11:10:43 mail amavisd[29188]: starting.  amavisd at mail \
  amavisd-new-20030616-p6
Jan 28 11:10:43 mail amavisd[29188]: Perl version              5.006 ❶
Jan 28 11:10:43 mail amavisd[29188]: Module Amavis::Conf       1.15
Jan 28 11:10:43 mail amavisd[29188]: Module Archive::Tar       1.08
Jan 28 11:10:43 mail amavisd[29188]: Module Archive::Zip       1.09
Jan 28 11:10:43 mail amavisd[29188]: Module Compress::Zlib     1.33
Jan 28 11:10:43 mail amavisd[29188]: Module Convert::TNEF      0.17
Jan 28 11:10:43 mail amavisd[29188]: Module Convert::UUlib     1.0
Jan 28 11:10:43 mail amavisd[29188]: Module MIME::Entity       5.404
Jan 28 11:10:43 mail amavisd[29188]: Module MIME::Parser       5.406
Jan 28 11:10:43 mail amavisd[29188]: Module MIME::Tools        5.411
Jan 28 11:10:43 mail amavisd[29188]: Module Mail::Header       1.60
Jan 28 11:10:43 mail amavisd[29188]: Module Mail::Internet     1.60
Jan 28 11:10:43 mail amavisd[29188]: Module Mail::SpamAssassin  2.63
Jan 28 11:10:43 mail amavisd[29188]: Module Net::Cmd           2.24
Jan 28 11:10:43 mail amavisd[29188]: Module Net::DNS           0.40
Jan 28 11:10:43 mail amavisd[29188]: Module Net::SMTP          2.26
Jan 28 11:10:43 mail amavisd[29188]: Module Net::Server        0.86
Jan 28 11:10:43 mail amavisd[29188]: Module Time::HiRes        1.55
Jan 28 11:10:43 mail amavisd[29188]: Module Unix::Syslog       0.99
Jan 28 11:10:43 mail amavisd[29188]: Found myself: /usr/sbin/amavisd
  -c /etc/amavisd.conf
Jan 28 11:10:43 mail amavisd[29188]: Lookup::SQL code       NOT loaded ❷
Jan 28 11:10:43 mail amavisd[29188]: Lookup::LDAP code      NOT loaded
Jan 28 11:10:43 mail amavisd[29188]: AMCL-in protocol code  loaded
Jan 28 11:10:43 mail amavisd[29188]: SMTP-in protocol code  loaded
Jan 28 11:10:43 mail amavisd[29188]: ANTI-VIRUS code        loaded
Jan 28 11:10:43 mail amavisd[29188]: ANTI-SPAM  code        loaded ❸
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: 2004/01/28-11:10:43
  Amavis (type Net::Server::PreForkSimple) starting! pid(29188)
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: Binding to UNIX
  socket file /var/amavis/amavisd.sock using SOCK_STREAM
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: Binding to TCP
  port 10024 on host 127.0.0.1
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: Setting gid
  to "54322 54322"
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: Setting uid to "7509"
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: Setting
  up serialization via flock
Jan 28 11:10:43 mail amavisd[29188]: Found $file       at /usr/bin/file
Jan 28 11:10:43 mail amavisd[29188]: Found $arc        at /usr/bin/arc
Jan 28 11:10:43 mail amavisd[29188]: Found $gzip       at /usr/bin/gzip
Jan 28 11:10:43 mail amavisd[29188]: Found $bzip2      at /usr/bin/bzip2
```

```
Jan 28 11:10:43 mail amavisd[29188]: Found $lzop       at /usr/local/bin/lzop
Jan 28 11:10:43 mail amavisd[29188]: Found $lha        at /usr/bin/lha
Jan 28 11:10:43 mail amavisd[29188]: Found $unarj      at /usr/bin/unarj
Jan 28 11:10:43 mail amavisd[29188]: Found $uncompress at /usr/bin/uncompress
Jan 28 11:10:43 mail amavisd[29188]: Found $unfreeze
  at /usr/local/bin/unfreeze
Jan 28 11:10:43 mail amavisd[29188]: Found $unrar      at /usr/bin/rar
Jan 28 11:10:43 mail amavisd[29188]: Found $zoo        at /usr/bin/zoo
Jan 28 11:10:43 mail amavisd[29188]: Found $cpio       at /bin/cpio ❹
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner:
  KasperskyLab AntiViral Toolkit Pro (AVP)
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner:
  KasperskyLab AVPDaemonClient
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: H+BEDV AntiVir
  or CentralCommand Vexira Antivirus
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Command
  AntiVirus for Linux
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner:
  Symantec CarrierScan via Symantec CommandLineScanner
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner:
Symantec AntiVirus Scan Engine
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Dr.Web
  Antivirus for Linux/FreeBSD/Solaris
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner:
  F-Secure Antivirus
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: CAI InoculateIT
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: MkS_Vir
  for Linux (beta)
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: MkS_Vir daemon
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: ESET
  Software NOD32
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: ESET
  Software NOD32 - Client/Server Version
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Norman
  Virus Control v5 / Linux
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Panda
  Antivirus for Linux
Jan 28 11:10:43 mail amavisd[29188]: Found primary av scanner NAI
  McAfee AntiVirus (uvscan) at /usr/local/bin/uvscan ❺
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: VirusBuster
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: CyberSoft VFind
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Ikarus
  AntiVirus for Linux
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: BitDefender
Jan 28 11:10:43 mail amavisd[29188]: No secondary av scanner: Clam
  Antivirus - clamscan
Jan 28 11:10:43 mail amavisd[29188]: No secondary av scanner: FRISK
  F-Prot Antivirus
Jan 28 11:10:43 mail amavisd[29188]: No secondary av scanner: Trend
  Micro FileScanner
Jan 28 11:10:43 mail amavisd[29188]: SpamControl: initializing
  Mail::SpamAssassin ❻
```

❶ Эта строка указывает версию Perl.

❷ Эта и следующая строки показывают, что код SQL и LDAP отсутствует.

❸ Эта строка указывает, что был загружен код борьбы со спамом, а это возможно, только если доступны SpamAssassin или dspam.

❹ Эта и предыдущая строка показывают, что были обнаружены различные внешние распаковщики, т. е. amavisd-new имеет возможность распаковывать вложения, сжатые посредством этих упаковщиков.

❺ Эта строка говорит о том, что присутствует McAfee AntiVirus.

❻ amavisd-new готова к работе.

Тестирование возможности сетевого взаимодействия

Теперь, когда мы знаем, что amavisd-new работает автономно, пришло время проверить, принимает ли она соединения. Используйте сеансы telnet для тестирования как ESMTP-, так и LMTP-взаимодействия.

Проверка доступности ESMTP

Открываем telnet-соединение с локальным портом 10 024 (порт по умолчанию для amavisd-new). Вы должны убедиться, что программа прослушивает порт и отвечает на ESMTP-команды. В следующем примере сеанса программа amavisd-new отвечает на команду `EHLO` набором доступных команд:

```
# telnet localhost 10024
220 [127.0.0.1] ESMTP amavisd-new service ready
EHLO mail.example.com
250-[127.0.0.1]
250-PIPELINING
250-SIZE
250-8BITMIME
250 ENHANCEDSTATUSCODES
QUIT
221 2.0.0 [127.0.0.1] (amavisd) closing transmission channel
```

Проверка доступности LMTP

Затем следует проверить, прослушивает ли amavisd-new локальный порт 10 024 (порт по умолчанию для LMTP) и отвечает ли на команды LMTP. Вы должны суметь запустить команду `LHLO`, как в данном сеансе:

```
# telnet localhost 10024
220 [127.0.0.1] ESMTP amavisd-new service ready
LHLO mail.example.com
250-[127.0.0.1]
250-PIPELINING
250-SIZE
250-8BITMIME
250 ENHANCEDSTATUSCODES
```

```
QUIT
221 2.0.0 [127.0.0.1] (amavisd) closing transmission channel
```

Оптимизация производительности amavisd-new

Если у вас очень много сообщений, вы, возможно, примете решение улучшить производительность amavisd-new. Производительность amavisd-new может зависеть от скорости и задержки дисковых операций ввода-вывода, т. к. она активно использует файловую систему для подготовки сообщений к дальнейшему анализу. Вы можете значительно улучшить скорость операций ввода-вывода, передав эту подготовку файловой системе на базе оперативной памяти. Эта процедура, описанная в последующих разделах, использует временную файловую систему Linux (temporary filesystem – tmpfs).

Безопасно ли это?

Благодаря способу интеграции amavisd-new в Postfix вы можете быть уверены в том, что в процессе фильтрации никакие сообщения потеряны не будут. Давайте посмотрим, что происходит в процессе фильтрации:

1. По получении нового сообщения диспетчер очередей Postfix отправляет запрос на доставку сообщения SMTP- или LMTP-клиенту Postfix. Клиент передает сообщение программе amavisd-new.
2. Программа amavisd-new начинает работать над сообщением (исследует, проверяет на спам и т. д.), но не подтверждает получение сообщения сразу же.
3. Ожидая ответа amavisd-new, сервер Postfix держит сообщение в очереди – amavisd-new должна сказать, что сообщение должным образом получено.
4. После того как amavisd-new закончит свою работу, она возвращает сообщение обратно в очередь Postfix.
5. Postfix-демон `smtpd`, который занимается возвратом сообщений в очередь, принимает сообщение от amavisd-new.
6. Получив подтверждение от экземпляра `smtpd`, возвращающего сообщения в очередь, amavisd-new подтверждает успешную передачу исходному LMTP- или SMTP-клиенту Postfix, который сообщает диспетчеру очередей о том, что сообщение доставлено.

Как видите, amavisd-new сообщает предфильтровой составляющей Postfix о том, что она получила сообщение, лишь после того, как `smtpd` «после фильтрации» примет обработанное сообщение. Таким образом, вы никак не можете потерять почту в amavisd-new.

Определение размера временной файловой системы

Для вычисления необходимого размера tmpfs подумайте о следующем: если вы запустите *n* экземпляров amavisd-new и каждый из них

будет принимать сообщения с максимальным размером предельный_размер_сообщения, то вам потребуется столько места:

```
n * (1 + максимально_ожидаемый_коэффициент_расширения) *
        предельный_размер_сообщения * 7/8
```

Элемент коэффициент_расширения кажется очень сложным, но фактически отлично подходит коэффициент 2 (сжатое сообщение, такое как файл *.zip или *.rar, может увеличиться вдвое по отношению к исходному размеру).

Например, если у вас пять экземпляров amavisd-new, а предельный размер сообщения установлен равным 10 Мбайт, то вы получите следующий результат для размера временной файловой системы:

```
5 * (1 + 2) * 10MB * 7/8 = 131.25MB
```

Примечание

Убедитесь в том, что у вас достаточно физической памяти для хранения tmpfs; в противном случае ваш компьютер начнет свопировать память на диск, и в итоге производительность будет хуже, чем у обычной файловой системы.

Настройка оптимизации

Настройка amavisd-new для использования tmpfs состоит из нескольких этапов:

1. Найти параметр $TEMPBASE программы amavisd-new.
2. Создать временную файловую систему на базе оперативной памяти (tmpfs).
3. Остановить amavisd-new.
4. Смонтировать tmpfs.
5. Запустить amavisd-new.
6. Убедиться в том, что amavisd-new работает.

Сначала необходимо узнать, где определен параметр amavisd-new $TEMPBASE. Это точка монтирования для создаваемой файловой системы tmpfs. По умолчанию значение $TEMPBASE совпадает с $MYHOME, которое по умолчанию равно /var/amavis. Чтобы знать наверняка, используйте grep для вашего файла конфигурации. В данном примере параметр $TEMPBASE установлен в $MYHOME:

```
# grep TEMPBASE /etc/amavisd.conf
$TEMPBASE = $MYHOME;            # (должен быть задан, если его используют
                                # другие переменные конфигурации)
$ENV{TMPDIR} = $TEMPBASE;       # разумно, но обычно не требуется
    "-f=$TEMPBASE {}", [0,8], [3,4,5,6], qr/infected: ([^\r\n]+)/ ],
    # adjusting /var/amavis above to match your $TEMPBASE.
    #   directory $TEMPBASE specifies) in the 'Names=' section.
```

Запускаем grep еще раз для определения значения $MYHOME. В примере видно, что определение $MYHOME закомментировано, так что используется значение по умолчанию:

```
# grep MYHOME /etc/amavisd.conf
# $MYHOME служит значением по умолчанию для некоторых
# других параметров конфигурации.
# $MYHOME не используется программой напрямую. Нет завершающей косой черты!
#$MYHOME = '/var/lib/amavis';  # (умолчание - это '/var/amavis')
$TEMPBASE = $MYHOME;          # (должен быть задан, если его используют
                              # другие переменные конфигурации)
#$TEMPBASE = "$MYHOME/tmp";   # предпочитает оставить домашний
                              # каталог /var/amavis чистым?
#$helpers_home = $MYHOME;     # (по умолчанию равен $MYHOME)
#$daemon_chroot_dir = $MYHOME; # (по умолчанию undef, т. е. chroot
                              # не требуется)
#$pid_file  = "$MYHOME/amavisd.pid";  # (по умолчанию "$MYHOME/amavisd.pid")
#$lock_file = "$MYHOME/amavisd.lock"; # (по умолчанию "$MYHOME/amavisd.lock")
#$forward_method = "bsmtp:$MYHOME/out-%i-%n.bsmtp";
$unix_socketname = "$MYHOME/amavisd.sock"; # сокет вспомогательного
протокола amavis (обычно устанавливается в $MYHOME/amavisd.sock)
$LOGFILE = "$MYHOME/amavis.log";  # (по умолчанию пуст, журнал не ведется)
    "{} -ss -i '*' -log=$MYHOME/vbuster.log", [0], [1],
```

Теперь необходимо создать запись для tmpfs в файле /etc/fstab, используя вычисленный в предыдущем разделе размер файловой системы. В следующем примере указан размер 150 Мбайт, а доступ ограничен определенным пользователем и группой. В данном случае идентификатор пользователя равен 7509, а GID – 54322, что соответствует пользователю и группе vscan в файлах /etc/passwd и /etc/group; но помните, что в вашей системе номера практически наверняка будут другими, так что вам нужно будет найти их самостоятельно:

```
/dev/shm  /var/amavis  tmpfs defaults,size=150m,mode=700,uid=7509,gid=54322 0 0
```

Прежде чем монтировать /var/amavis, остановите amavisd-new командой:

```
# /etc/init.d/amavisd-new stop
```

Затем монтируем /var/amavis (помните, что это файловая система tmpfs, которую вы только что определили в /etc/fstab):

```
# mount /var/amavis
```

Теперь заново запускаем amavisd-new:

```
# /etc/init.d/amavisd-new start
```

Проверяем, все ли хорошо, просматривая журналы и исследуя вывод df -h. В следующем примере размер /var/amavis равен 100 Мбайт, а используются в настоящее время всего 76 Кбайт:

```
# df -h /var/amavis
```

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/shm              100M   76k   99M   1% /var/amavis
```

Примечание

Иногда amavisd-new оставляет устаревшие файлы в своем каталоге `$TEMPBASE`. Для того чтобы предотвратить заполнение `$TEMPBASE` такими файлами, вы можете каждый день останавливать amavisd-new, удалять устаревшие файлы и перезапускать программу. Эту задачу может выполнить сценарий, ежедневно запускаемый посредством `cron`:

```
#!/bin/bash
/etc/init.d/amavisd stop
rm -Rf /var/amavis/amavis-200*
/etc/init.d/amavisd start
```

Настройка Postfix для использования amavisd-new

На настоящий момент Postfix и amavisd-new работают независимо друг от друга. Следовательно, вам нужно настроить сервер Postfix так, чтобы он отправлял сообщения amavisd-new, и создать еще один экземпляр `smtpd` для возвращения сообщений в очередь Postfix. В дальнейших разделах будут описаны следующие этапы интегрирования amavisd-new в Postfix:

1. Создание транспорта.
2. Настройка транспорта.
3. Настройка пути возврата сообщений.

Примечание

Отфильтрованная почта должна каким-то образом вернуться в очередь Postfix без повторного сканирования, поэтому вам необходим отдельный экземпляр `smtpd`, не использующий `content_filter`. Это позволит amavisd-new возвращать сообщения в систему, не создавая бесконечных петель. Порт 25 уже занят, так что новый экземпляр `smtpd` должен прослушивать нестандартный порт. В нашем примере используется порт 10 025 локального хоста.

Кроме того, amavisd-new также необходим порт для прослушивания. Тут вполне подходит значение по умолчанию – порт 10 024 локального хоста.

Создание транспорта при помощи content_filter в main.cf

Первый шаг в передаче обработки содержимого внешней программе – это определение транспорта, который отправляет сообщения фильтрующей программе. Postfix использует параметр `content_filter` в файле `main.cf`. Параметр должен быть записан в виде `имя_транспорта:следующий_узел:порт`.

В рассматриваемом примере программа amavisd-new работает на том же компьютере, что и Postfix, так что вы можете обращаться к ней че-

рез порт 10 024 локального хоста (127.0.0.1). Для того чтобы Postfix подключился к amavisd-new, необходимо определить в файле main.cf такой параметр content_filter:

```
content_filter = amavisd-new:[127.0.0.1]:10024
```

Запуск amavisd-new на другом хосте

Если вы чувствуете, что создаваемая фильтрацией нагрузка слишком велика для одного компьютера, то можете запускать amavisd-new на нескольких компьютерах. Составляющая следующий_узел параметра имя_транспорта:следующий_узел:порт позволяет вам без труда указать другой хост для фильтра. Например, в следующем примере использован vscanners.example.com:

```
content_filter = amavisd-new:vscanners.example.com:10024
```

Имя vscanners.example.com может относиться к одному из следующих объектов:

- одному компьютеру (через одну A-запись);
- нескольким компьютерам (через несколько циклических A-записей);
- нескольким компьютерам (один или несколько компьютеров с MX-записями разных приоритетов).

Определение транспорта в файле master.cf

Затем вам следует определить демон, который будет подключаться к amavisd-new, и создать его окружение. Помните о том, что это может быть демон smtp, lmtp или pipe. Вы уже видели пример использования pipe выше в этой главе, теперь пришло время обратиться к оставшимся двум демонам.

Определение ESMTP-транспорта

Если вы хотите использовать протокол ESMTP для отправки сообщений amavisd-new, то добавьте следующие записи в файл master.cf:

```
#==========================================================================
# service type          private  unpriv  chroot   wakeup   maxproc  command
#                       (yes)    (yes)   (yes)    (never)  (100)
# ==========================================================================
...
amavisd-new    unix     -        -       n        -        2        smtp
  -o smtp_data_done_timeout=1200s
  -o disable_dns_lookups=yes
```

Кое-что в приведенных записях необходимо отметить особо:

- Специальный транспорт amavisd-new – это копия обычного транспорта smtp. Его имя должно совпадать с именем транспорта, указанным для параметра content_filter, который вы определили в файле main.cf.

- Программа amavisd-new требует достаточно большого объема ресурсов. Поэтому, если у вас не слишком быстрый компьютер, вы, возможно, захотите ограничиться максимум двумя одновременными экземплярами.
- Параметр `smtp_data_done_timeout` является первой из двух дополнительных настроек, которые изменяют поведение демона. Обработка входящего сообщения может занять у amavisd-new значительное время, и увеличение тайм-аута после отправки демоном `smtp` сообщения не дает Postfix прекратить ожидание, пока amavisd-new не завершит работу.
- Вероятно, вы пока имеете дело лишь с локальными машинами, так что параметр `disable_dns_lookups` отменит ненужный поиск в DNS для клиента `smtp`.

Примечание

Вам не обязательно понадобится выделенный SMTP-транспорт, т. к. существующий по умолчанию `smtp` не вызывает нареканий. Однако из соображений производительности (и из-за достаточно длинного тайм-аута amavisd-new) может иметь смысл выделение отдельного транспорта специально для amavisd-new.

Определение LMTP-транспорта

Если вы решили использовать протокол LMTP (вместо SMTP) для передачи сообщений программе amavisd-new, то добавьте следующую запись в свой файл `master.cf`:

```
#==========================================================================
# service type          private  unpriv  chroot   wakeup   maxproc   command
#                       (yes)    (yes)   (yes)    (never)  (100)
#==========================================================================
...
amavisd-new    unix     -        -       n        -        2         lmtp
  -o lmtp_data_done_timeout=1200s
  -o disable_dns_lookups=yes
```

Настройка пути возврата сообщений в Postfix

Наконец, вам необходимо создать путь возврата, который позволит amavisd-new передавать сообщения обратно в очередь Postfix. Важно, что этот путь возврата игнорирует транспорт amavisd-new. В противном случае сообщение попало бы в замкнутый круг: Postfix отправлял бы сообщение amavisd-new, затем оно возвращалось бы в очередь Postfix, а оттуда снова отправлялось бы в amavisd-new.

Путь возврата, игнорирующий любой определенный ранее параметр `content_filter`, выглядит в файле `master.cf` следующим образом:

```
#==========================================================================
# service type          private  unpriv  chroot   wakeup   maxproc   command
#                       (yes)    (yes)   (yes)    (never)  (100)
# ==========================================================================
...
127.0.0.1:10025  inet   n        -       n        -        -         smtpd
  -o content_filter=
  -o local_recipient_maps=
  -o relay_recipient_maps=
  -o smtpd_restriction_classes=
  -o smtpd_client_restrictions=
  -o smtpd_helo_restrictions=
  -o smtpd_sender_restrictions=
  -o smtpd_recipient_restrictions=permit_mynetworks,reject
  -o mynetworks=127.0.0.0/8
  -o strict_rfc821_envelopes=yes
```

Из всех параметров предыдущей записи *абсолютно* необходим пустой параметр `content_filter`. Эта настройка перекрывает параметр `content_filter` в файле `main.cf`. Остальные параметры также подменяют соответствующие параметры `main.cf`, включая параметры для отмены ограничений, которые не имеют смысла для транспорта, прослушивающего лишь локальный сетевой интерфейс.

Когда все настройки сделаны, можно приступать к тестированию фильтра. Не забывайте о том, что для вступления в силу изменений `master.cf` требуется перезагрузка конфигурации Postfix.

Тестирование фильтра amavisd-new в Postfix

Для тестирования совместной работы Postfix и amavisd-new вам следует проверить, может ли Postfix отправлять почту в amavisd-new и может ли amavisd-new возвращать сообщения обратно в очередь Postfix. Тестирование будет состоять из следующих этапов:

1. Проверяем, прослушивает ли Postfix путь возврата.
2. Отправляем сообщение серверу Postfix и проверяем действия сервера: он должен направить сообщение в amavisd-new, а затем оно вернется обратно в очередь Postfix.
3. Проверяем, обнаружит ли антивирусный сканер тестовый образец.

Проверка пути возврата

Изменив файл `master.cf`, выполните команду `postfix reload`, чтобы Postfix прочитал измененный файл, а затем просмотрите файл журнала: нет ли там каких-нибудь жалоб. Затем проверьте, прослушивает ли демон возврата `smtpd` порт `10025` хоста `localhost`, как в следующем сеансе:

```
$ telnet 127.0.0.1 10025
220 mail.example.com ESMTP Postfix
```

```
EHLO 127.0.0.1
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
QUIT
221 Bye
```

Отправка тестового сообщения серверу Postfix

Postfix должен пропустить «неинфицированное» сообщение через всю систему. Отправляем сообщение из командной строки и следим за его судьбой по сообщениям в журнале, оставляемым там сервером Postfix и программой amavisd-new. Например, вы могли бы использовать такую команду для отправки своего файла `main.cf` по адресу `recipient@example.com`:

```
# sendmail -f sender@example.com recipient@example.com < /etc/postfix/main.cf
```

Обратимся к журналу. В нижней части файла должен присутствовать сеанс, начинающийся как приведенный ниже, где Postfix присваивает сообщению идентификатор, по которому мы и будем его отслеживать:

```
Jan 31 10:45:08 mail postfix/pickup[10096]: 2788029AB29: uid=0
  from=<sender@example.com>
Jan 31 10:45:08 mail postfix/cleanup[10652]: 2788029AB29:
  message-id=<20040131094508.2788029AB29@mail.example.com>
```

Следующий набор записей журнала должен показывать, что Postfix передает сообщение локальному хосту `localhost` для обработки программой amavisd-new (к сожалению, Postfix не пишет в журнал номер порта и имя транспорта):

```
Jan 31 10:45:08 mail postfix/qmgr[10097]: 2788029AB29:
  from=<sender@example.com>, size=1271, nrcpt=1 (queue active)
Jan 31 10:45:08 mail postfix/smtp[10660]: 2788029AB29:
  to=<recipient@example.com>, relay=localhost[127.0.0.1], delay=0,
  status=sent (250 2.6.0 Ok, id=25809-04, from MTA: 250 Ok:
  queued as 377D829AB2A)
```

Теперь amavisd-new анализирует сообщение и пишет в журнал, что сообщение пропущено (passed):

```
Jan 31 10:45:08 mail amavis[25809]: (25809-04) Passed,
  <sender@example.com> -> <recipient@example.com>, Message-ID:
  <20040131094508.2788029AB29@mail.example.com>, Hits: -
```

Затем сообщение попадает обратно в Postfix из amavisd-new для возвращения в очередь. Обратите внимание на то, что второй экземпляр `smtpd` также записывает идентификатор сообщения:

```
Jan 31 10:45:08 mail postfix/smtpd[10658]: connect from localhost[127.0.0.1]
Jan 31 10:45:08 mail postfix/smtpd[10658]: 377D829AB2A:
  client=localhost[127.0.0.1]
Jan 31 10:45:08 mail postfix/cleanup[10652]: 377D829AB2A:
  message-id=<20040131094508.2788029AB29@mail.example.com>
Jan 31 10:45:08 mail postfix/qmgr[10097]: 377D829AB2A:
  from=<sender@example.com>, size=1723, nrcpt=1 (queue active)
Jan 31 10:45:08 mail postfix/smtpd[10658]: disconnect
  from localhost[127.0.0.1]
```

Наконец, Postfix пересылает сообщение другому хосту для доставки (если же оказывается, что этот сервер является местом конечного назначения, то Postfix выполняет локальную доставку):

```
Jan 31 10:45:08 mail postfix/smtp[10655]: 377D829AB2A:
  to=<recipient@example.com>, relay=relayhost[10.0.0.1], delay=0,
  status=sent (250 OK id=1AmrgY-00073g-00)
```

Проверка тестового вирусного образца

Последней проверкой будет имитация заражения сообщения вирусом. Вы можете сделать это, получив тестовый образец вируса EICAR (*http://www.eicar.org*) и отправив его Postfix. Любые средства поиска вирусов, в которых этот образец специально не отключен, должны его распознавать. Например, следующая команда отправит вирус по адресу `recipient@example.com`:

```
# sendmail -f sender@example.com recipient@example.com < eicar.com
```

Журнальные сообщения выглядят как и раньше до тех пор, пока amavisd-new не начинает проверять сообщение:

```
Feb  6 15:48:54 mail postfix/pickup[30051]: 13B9E29AB29: uid=0
  from=<sender@example.com>
Feb  6 15:48:54 mail postfix/cleanup[30741]: 13B9E29AB29:
  message-id=<20040206144854.13B9E29AB29@mail.example.com>
Feb  6 15:48:54 mail postfix/qmgr[19295]: 13B9E29AB29:
  from=<sender@example.com>, size=347, nrcpt=1 (queue active)
Feb  6 15:48:54 mail postfix/smtp[30744]: 13B9E29AB29:
  to=<recipient@example.com>, relay=localhost[127.0.0.1], delay=0,
  status=sent (250 2.5.0 Ok, id=10217-07, BOUNCE)
...
Feb  6 15:48:54 mail amavis[10217]: (10217-07) INFECTED
  (Eicar-Test-Signature), <sender@example.com> -> <recipient@example.com>,
  quarantine virus-20040206-154854-10217-07, Message-ID:
  <20040206144854.13B9E29AB29@mail.example.com>, Hits: -
```

Увидев, что сообщение содержит вирус, amavisd-new предупреждает `virusalert@example.com` и возвращает сообщение обратно отправителю:

```
Feb  6 15:48:54 mail postfix/smtpd[30747]: connect from localhost[127.0.0.1]
Feb  6 15:48:54 mail postfix/smtpd[30747]: 639A729AB2A:
  client=localhost[127.0.0.1]
Feb  6 15:48:54 mail postfix/cleanup[30741]: 639A729AB2A:
  message-id=<VA10217-07@mail>
Feb  6 15:48:54 mail postfix/qmgr[19295]: 639A729AB2A: from=<>, size=1463,
  nrcpt=1 (queue active)
Feb  6 15:48:54 mail postfix/local[30749]: 639A729AB2A:
  to=<virusalert@example.com>, relay=local, delay=0, status=sent
  (forwarded as 8484829AB2C)
Feb  6 15:48:54 mail postfix/smtpd[30747]: disconnect from
  localhost[127.0.0.1]
...
Feb  6 15:48:54 mail postfix/smtpd[30747]: connect from localhost[127.0.0.1]
Feb  6 15:48:54 mail postfix/smtpd[30747]: 7A2FD29AB2B:
  client=localhost[127.0.0.1]
Feb  6 15:48:54 mail postfix/cleanup[30741]: 7A2FD29AB2B:
  message-id=<VS10217-07@mail>
Feb  6 15:48:54 mail postfix/qmgr[19295]: 7A2FD29AB2B: from=<>,
  size=2554, nrcpt=1 (queue active)
Feb  6 15:48:55 mail postfix/smtp[30744]: 7A2FD29AB2B:
  to=<sender@example.com>, relay=relayhost[10.0.0.1], delay=1,
  status=sent (250 OK id=1Ap7Ho-00014I-00)
```

Возврат сообщения отправителю – не очень хорошая мысль, т. к. при рассылке вирусов адрес отправителя практически всегда сфальсифицирован, но, к сожалению, именно так amavisd-new ведет себя по умолчанию.

Поиск вирусов при помощи smtpd_proxy_filter и amavisd-new

Существует другой, более новый подход к фильтрации содержимого в Postfix – исследование входящих сообщений до постановки их в очередь. Этот вид фильтра называется `smtpd_proxy_filter`. Вы можете использовать его совместно с amavisd-new, как показано на рис. 12.3.

При использовании `smtpd_proxy_filter` путь сообщения будет таким:

1. Почтовый клиент отправляет сообщение Postfix-демону `smtpd`.
2. `smtpd` (с включенным `smtpd_proxy_filter`) передает сообщение программе amavisd-new. Обратите внимание на отличие от случая с `content_filter`, где диспетчер очередей просит `lmtp`- или `smtp`-клиент Postfix отправить сообщение amavisd-new.
3. amavisd-new отправляет сообщение другим приложениям (в данном примере – двум антивирусным программам).
4. amavisd-new сообщает `smtpd` о том, принимает она или отвергает сообщение. Если сообщение принято, то оно возвращается обратно второму экземпляру `smtpd`, если же оно отвергнуто, то действия определяются параметрами конфигурации.

5. Исходный экземпляр `smtpd` ожидает ответа amavisd-new, принимая или отвергая сообщение клиента.

Примечание

`smtpd_proxy_filter` – это демон `smtpd`, разбитый на две части:

- первая часть очищает входящие сообщения при помощи фильтра;
- вторая часть занимается постановкой в очередь.

В этом разделе будет рассказано о том, как настроить amavisd-new для работы с `smtpd_proxy_filter`, используя в качестве основы общие операции, описанные в главе 11. Вам нужно будет выполнить следующие действия:

1. Установите amavisd-new (см. ранее в этой главе раздел «Установка amavisd-new»).
2. Протестируйте amavisd-new (см. ранее в этой главе раздел «Тестирование amavisd-new»).
3. Настройте Postfix для работы с amavisd-new.
4. Протестируйте конфигурацию.

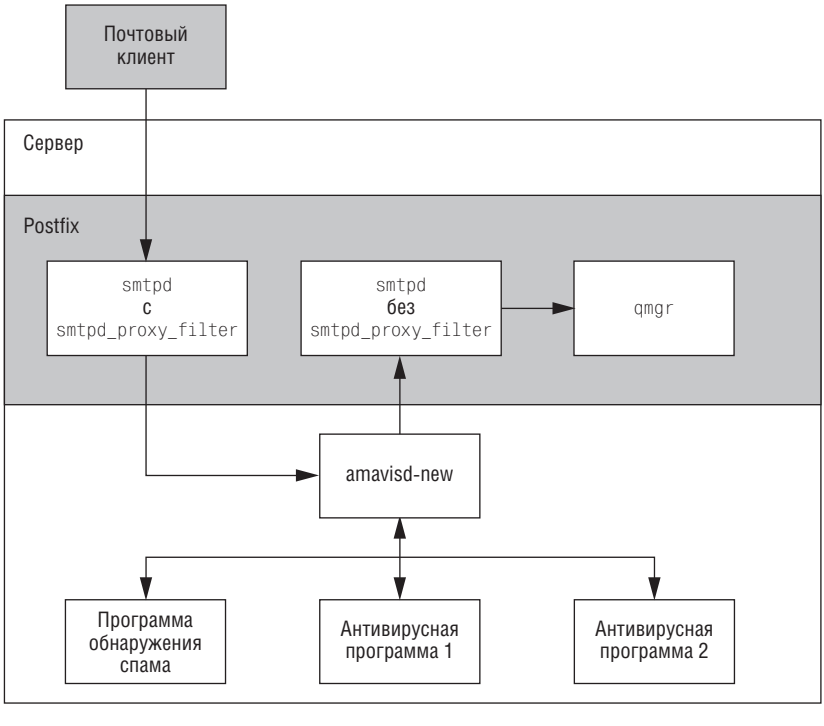

***Рис. 12.3.** Интеграция amavisd-new с Postfix посредством smtpd_proxy_filter*

Настройка Postfix для использования amavisd-new с smtpd_proxy_filter

Первым делом необходимо определить транспорты, по которым сообщения будут попадать в фильтрующую программу. Вам нужно выполнить следующее:

1. Изменить существующий транспорт `smtpd` для пересылки сообщений amavisd-new.
2. Создать дополнительный экземпляр `smtpd` для возврата сообщений обратно в Postfix в обход глобального параметра `smtpd_proxy_filter`.
3. Протестировать конфигурацию, как было показано в предыдущем разделе.

Изменение существующего smtpd

Для того чтобы `smtpd` передавал сообщения amavisd-new, добавьте параметр `smtpd_proxy_filter` в существующую службу `smtp` в файле `master.cf`. Например, следующая запись заставляет `smtpd` отправлять сообщения на порт 10 024 хоста `localhost` (если помните, это настройки по умолчанию для amavisd-new):

```
#==========================================================================
# service type        private   unpriv   chroot   wakeup   maxproc  command
#                               (yes)    (yes)    (yes)    (never)  (100)
# ==========================================================================
...
smtp    inet          n         -        n        -        20       smtpd
   -o smtpd_proxy_filter=localhost:10024
   -o smtpd_client_connection_count_limit=10
```

Обратите внимание на настройку `-o smtpd_client_connection_count_limit=10`, которая не дает одному SMTP-клиенту использовать все 20 серверных процессов SMTP, определенных в столбце `maxproc`. Это ограничение не обязательно, если вы получаете сообщения только от заслуживающего доверия хоста пересылки.

В отличие от случая использования механизма `content_filter`, вы не определяете глобальный параметр в файле `main.cf`, так что нет необходимости в его явной подмене для транспорта возврата сообщений в Postfix.

Создание дополнительного экземпляра smtpd для возврата сообщений

Для того чтобы сообщения могли вернуться в очередь Postfix не по передающему сообщения экземпляру `smtpd`, вам нужен дополнительный специальный экземпляр `smtpd` в файле `master.cf`. Рассмотрим пример создания второго экземпляра для порта 10 025 хоста `localhost`:

```
#==========================================================================
# service type        private  unpriv  chroot   wakeup   maxproc   command
#                              (yes)   (yes)    (yes)    (never)   (100)
# ==========================================================================
...
127.0.0.1:10025 inet  n        -       n        -        -         smtpd
   -o smtpd_authorized_xforward_hosts=127.0.0.0/8
   -o smtpd_client_restrictions=
   -o smtpd_helo_restrictions=
   -o smtpd_sender_restrictions=
   -o smtpd_recipient_restrictions=permit_mynetworks,reject
   -o mynetworks=127.0.0.0/8
   -o receive_override_options=no_unknown_recipient_checks
```

Настройка `-o smtpd_authorized_xforward_hosts=127.0.0.0/8` позволяет экземпляру `smtpd` «после фильтрации» получать информацию удаленных SMTP-клиентов от экземпляра `smtpd` «до фильтрации». А именно, экземпляр «после фильтрации» будет принимать любые команды `XFORWARD`, отправленные хостом из списка `smtpd_authorized_xforward_hosts`. Это чрезвычайно полезно для отладки, т. к. `smtpd` будет использовать исходный клиентский IP-адрес вместо `localhost[127.0.0.1]`.

Остальные параметры уменьшают нагрузку на экземпляр `smtpd` «после фильтрации», т. к. работа уже сделана демоном «до фильтрации».

III

Сложные конфигурации

В этой части книги будут рассмотрены распространенные случаи взаимодействия Postfix с другими приложениями, такими как SQL-серверы, Cyrus SASL, OpenSSL и OpenLDAP. Приведем краткий обзор глав данной части:

Почтовые шлюзы

Почтовые шлюзы передают сообщения от имени других почтовых серверов или клиентов. В большинстве случаев почтовые шлюзы соединены с Интернетом, в то время как другие серверы находятся в безопасности под защитой межсетевого экрана. В главе 13 вы узнаете, как из простого хоста-ретранслятора сделать интеллектуальный хост.

Почтовый сервер для нескольких доменов

В главе 14 описаны два способа, которыми Postfix может обрабатывать сообщения для нескольких доменов. Кроме того, вы узнаете, как настраивать Postfix для обращения к серверу SQL вместо просмотра статических карт.

Введение в SMTP-аутентификацию

SMTP-аутентификация – это система аутентификации почтовых клиентов, выполняемой перед тем, как они будут пересылать сообщения. SMTP-аутентификация в Postfix реализована на основе Cyrus SASL, поэтому глава 15 расскажет вам о том, как настраивать библиотеки Cyrus SASL для совместной работы с Postfix.

SMTP-аутентификация

Продолжая разговор о SMTP-аутентификации, глава 16 показывает, как настраивать Postfix для аутентификации в качестве сервера и в качестве клиента.

Протокол TLS (Transport Layer Security)

Протокол безопасности транспортного уровня TLS обеспечивает шифрование на уровне соединения между Postfix и другими хостами. В реализации TLS для Postfix используется OpenSSL, так что глава 17 рассказывает не только о том, как работает TLS, но и как подготовить необходимые сертификаты.

Использование TLS

В главе 18 показано, как настроить сервер Postfix так, чтобы он предоставлял услуги шифрования для других хостов, и как сделать так, чтобы клиент Postfix использовал его в то время, когда другие серверы обеспечивают TLS. Вы также узнаете, как работает пересылка на основе сертификатов.

Корпоративный почтовый сервер

Глава 19 объясняет, как настроить Postfix для обращения к серверу LDAP. При этом вы поручаете локальную доставку агенту доставки сообщений и настраиваете базовый сервер Courier IMAP. К концу главы у вас будет готова полная почтовая система, получающая пользовательские данные от сервера OpenLDAP.

Запуск Postfix в chroot-окружении

В главе 20 показано, как настроить сервер Postfix в случае запуска с помощью chroot. В ней объясняется, почему некоторые демоны не могут работать из-под chroot, и приводится пример работы запущенного из chroot сервера Posfix совместно с SASL.

13

Почтовые шлюзы

Почтовый шлюз (также называемый «интеллектуальным хостом») – это сервер, который соединяет логически разделенные сети. Обычно почтовый шлюз является местом конечного назначения в DNS-записях для других почтовых серверов, и серверы-отправители даже не подозревают, что за шлюзом есть другие почтовые серверы. В этой главе будет рассказано о том, как настроить почтовый шлюз и какими характеристиками обладает *настоящий* интеллектуальный хост.

Компании и интернет-провайдеры используют почтовые шлюзы для управления входящим и исходящим SMTP-трафиком своей сети. Обычно сетевые настройки разрешают трафику, приходящему на порт 25, достичь лишь почтового шлюза и требуют от клиентов внутри сети использовать эту машину для исходящей почты. Межсетевой экран блокирует порты, а Postfix занимается пересылкой почты.

На рис. 13.1 изображен сервер рабочей группы, который пересылает все сообщения почтовому шлюзу и принимает все сообщения от него. Почтовый шлюз защищает сервер от внешних атак: внешние клиенты и серверы не могут подключиться напрямую к серверу рабочей группы.

Совет

Вы можете расширить функциональность почтового шлюза за счет интеграции антивирусной программы или централизованного спам-фильтра. Тем самым вы защитите внутреннюю сеть[1] не только от вредоносных соединений, но и от вредоносного содержимого.

[1] «Внутренняя сеть» здесь означает сеть, скрытую для внешних отправителей почты. Это могут быть все клиенты провайдера, если интеллектуальный хост – это почтовый сервер провайдера, или все пользователи внутренней сети организации, для которой данный почтовый сервер принимает письма. – *Примеч. науч. ред.*

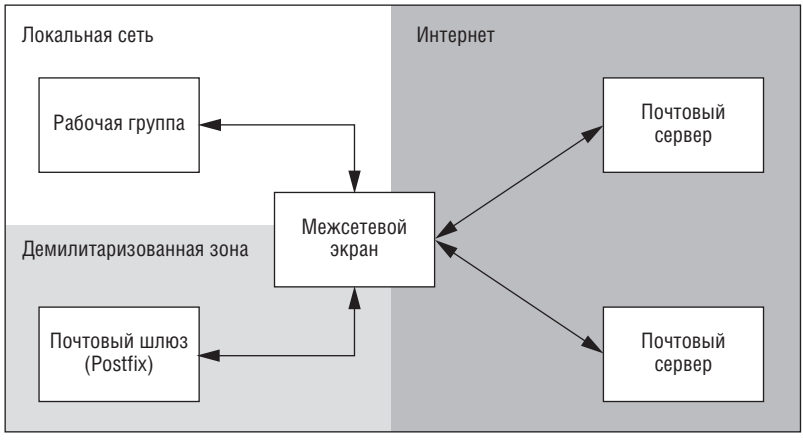

Рис. 13.1. *Postfix в качестве почтового шлюза для сервера рабочей группы*

Если вы предоставляете клиентам услуги пересылки, эта глава поможет вам выполнить базовую настройку почтового шлюза. Можно расширить его возможности, добавив поддержку SMTP-аутентификации (см. главу 16) и протокола TLS (см. главу 18).

Базовая настройка

Базовая настройка почтового шлюза позволяет Postfix работать на внешнем почтовом сервере и ретранслировать сообщения, предназначенные для определенных доменов, другому (внутреннему) почтовому серверу. Для создания такого почтового шлюза вам надо выполнить на сервере-ретрансляторе следующие действия:

1. Разрешить внутреннему серверу использовать шлюз в качестве ретранслятора.
2. Указать домены, от которых сообщения будут приниматься для ретрансляции внутренним серверам (`relay_domains`).
3. Указать внутренний хост, на который будут ретранслироваться сообщения (`transport_maps`).
4. Определить получателей, сообщения для которых будут приниматься для ретрансляции внутренним серверам (`relay_recipient_maps`).

Определение прав на ретрансляцию для шлюза

Наш первый шаг состоит в том, чтобы разрешить ретрансляцию на почтовом шлюзе для нашего «невидимого» почтового сервера. Добавляем IP-адрес внутреннего почтового сервера в список серверов параметра `mynetworks`. Например, если адрес внутреннего сервера – `172.16.1.1`, то вы можете добавить такую строку в файл `main.cf` на почтовом шлюзе:

```
mynetworks = 127.0.0.0/8 172.16.1.1/32
```

Доступ на ретрансляцию ограничен адресом localhost почтового шлюза (127.0.0.1) и внутреннего почтового сервера (в нашем примере это 172.16.1.1), так что другие хосты внутри вашей сети не могут использовать шлюз для пересылки.

Определение домена ретрансляции для шлюза

На следующем шаге мы указываем Postfix на необходимость приема сообщений из внешней сети для хоста внутренней сети. С помощью параметра relay_domains Postfix определяет список доменов, для которых он осуществляет пересылку, даже в том случае, если не является местом конечного назначения для этих доменов. Например, если вы хотите пересылать почту для example.com, используйте такую настройку:

```
relay_domains = example.com
```

Определение внутреннего почтового хоста для шлюза

Теперь, когда шлюз знает о том, что должен принимать сообщения для некоторого домена, вы должны сказать ему, куда ретранслировать входящие сообщения, направляющиеся в этот домен. Для этого создаем карту транспорта в файле /etc/postfix/transport. Например, если вы хотите пересылать сообщения для домена example.com на хост mail.office.example.com, файл карты может быть таким:

```
example.com    smtp:[mail.office.example.com]
```

В этой строке smtp означает, что Postfix должен использовать тип транспорта smtp, определенный в файле master.cf. Квадратные скобки имеют важное значение, т. к. они отменяют поиск MX для mail.office.example.com. Без таких скобок сервер Postfix стал бы искать MX-запись для mail.office.example.com. А так как запись, вероятно, принадлежит самому хосту сервера, он пытался бы доставить почту себе самому, и входящие сообщения попали бы в бесконечный цикл.

Теперь нам нужно создать индексированный файл следующей командой:

```
# postmap hash:/etc/postfix/transport
```

Наконец, задаем параметр transport_maps parameter в файле main.cf (и перезагружаем конфигурацию):

```
transport_maps = hash:/etc/postfix/transport
```

Определение получателей для пересылки

Что делает шлюз «интеллектуальным»? Обычный шлюз принимает *любые* сообщения для домена ретрансляции, в том числе для недействительных получателей, не существующих на внутреннем почтовом сервере, который, в конечном счете, доставляет сообщения.

Учитывая огромное количество спама и вредоносных программ, распространяемых в наши дни через Интернет, и принимая во внимание то, что некоторым получателям может быть не разрешено получать сообщение извне (например, совместно используемым папкам, внутренним спискам рассылки и, возможно, даже каким-то реальным пользователям), можно сказать, что обычный шлюз, принимающий все сообщения, может вызвать проблемы. В частности, он будет ретранслировать сообщения, спам и вирусы несуществующим и неавторизованным пользователям.

Кроме того, принимая почту для несуществующего адресата, шлюз принимает на себя ответственность за уведомление отправителя в случае невозможности доставки сообщения. Тем самым «загрязняется» очередь с сообщениями MAILER-DAEMON, и по всей сети Интернет рассылаются уведомления людям, которые ничего подобного не отправляли.

Когда Postfix не принимает сообщение, ответственность за него остается на клиенте. Если клиент представляет собой взломанную систему Windows, то уведомления о недоставке вообще не будут рассылаться отправителям.[1]

Интеллектуальный хост умеет отделять зерна от плевел, т. к. у него есть список действительных получателей для внутренних серверов. Используйте параметр `relay_recipient_maps` для определения и активации списка действительных получателей. Например, если имя вашей карты – `/etc/postfix/relay_recipients`, то добавьте в файл `main.cf` такую строку:

```
relay_recipient_maps = hash:/etc/postfix/relay_recipients
```

Предупреждение

Если вы определяете этот параметр, карта должна содержать список действительных получателей для пересылки. В противном случае в действиях вашего шлюза не будет никакой последовательности. Если вы не можете предоставить список, отмените карту следующим образом: `relay_recipient_maps =`.

Естественно, если вы сообщаете Postfix, где он должен искать карту, то должны действительно предоставить список. Если карта описана как в предыдущем примере, создайте текстовый файл с именем `/etc/postfix/relay_recipients`, содержащий действительных получателей. Например, следующий файл разрешает пересылку для адресов `john@example.com` и `linda@example.com`:

[1] …поскольку письмо от такой системы наш интеллектуальный хост не примет, т. к. он не принимает почту для кого попало. А если спамерская программа на взломанной машине отгадает адрес настоящего получателя, то письмо будет доставлено, и уведомления о недоставке опять же не будет. – *Примеч. науч. ред.*

```
john@example.com        OK
linda@example.com       OK
```

Если же вы хотите явно запретить доступ к определенному получателю, удалите запись из карты.

Как всегда при работе с картами, вам необходимо преобразовать карту к индексированной базе того типа, который определен в параметре `relay_recipient_maps`. Для этого выполните команду `postmap hash:/etc/postfix/relay_recipients`.

Примечание

Список действительных пользователей легко составить вручную, если у вас всего несколько пользователей на удаленном почтовом сервере, которые редко меняются. Однако такая ситуация маловероятна; скорее всего, у вас будет много часто меняющихся пользователей, и вы можете даже не знать их всех. Обратитесь к разделу «Экспорт действительных получателей из Active Directory» ниже в этой главе, чтобы узнать, как можно автоматизировать процесс. В частности, в указанном разделе описано, как получить список действительных адресатов от сервера Microsoft Exchange 2003.

Расширенная настройка шлюза

Усовершенствованный шлюз не только пересылает почту другим серверам, но и защищает от локальных почтовых атак и автоматизирует процесс обновления списка действительных получателей домена. В последующих разделах описаны некоторые приемы повышения качества услуг, предоставляемых вашим почтовым шлюзом.

Повышение безопасности почтового шлюза

Пока что ваша конфигурация Postfix пересылает все сообщения с адресом `example.com` внутреннему почтовому серверу, `mail.office.example.com`. Если это единственная задача, которую должен решать ваш интеллектуальный хост (т. е. если интеллектуальный хост не получает сообщения от локальных пользователей), вам нужно отменить локальную доставку, чтобы он стал неуязвим для вредоносных сообщений, адресованных его локальным пользователям.

Для отмены локальной доставки нужно выполнить следующие действия:

1. Обнулить параметр локальной доставки. Сначала сообщите Postfix, что данный сервер не является конечным местом назначения, установив параметр `mydestination` в пустое значение:

```
mydestination =
```

2. Отключить локальных получателей. Установите пустое значение параметра `local_recipient_maps`, чтобы сервер Postfix не мог найти никакого локального получателя:

```
local_recipient_maps =
```

3. Переадресовать обязательных локальных получателей. Когда вы задаете пустое значение для параметра `local_recipient_maps`, сообщения всем локальным получателям блокируются. Однако вам нужно, чтобы шлюз соответствовал RFC, поэтому надо установить переадресацию сообщений для `postmaster` и `abuse` на ваш внутренний почтовый сервер.

   Создайте карту, на которую будет ссылаться параметр `virtual_alias_maps` (файл `/etc/postfix/virtual` вполне подойдет), и укажите в ней новые адреса этих двух получателей на внутреннем почтовом сервере. Файл карты может выглядеть так:

```
postmaster        postmaster@example.com
abuse             abuse@example.com
```

   Теперь строим на основе файла индексированную карту, используя команду `postmap hash:/etc/postfix/virtual`, и ссылаемся на нее в файле `main.cf`:

```
virtual_alias_maps = hash:/etc/postfix/virtual
```

4. Создать сообщение об ошибке локальной доставки. Отключив локальную доставку, вы должны сообщать клиентам, пытающимся отправить сообщение интеллектуальному хосту, что вы запретили доставку локальным получателям. Для этого определяем в параметре `local_transport` специальный локальный транспорт, который передает сообщение об ошибке. Например, следующая строка отправляет все локальные сообщения демону `error`, который будет выдавать соответствующее сообщение об ошибке:

```
local_transport = error:local mail delivery is disabled
```

5. Перенаправить ответы на сообщения локальных служб. Если вы уже проделали все перечисленные ранее действия, то теперь ваш сервер Postfix не будет принимать почту для локальных пользователей, за исключением `postmaster` и `abuse`. Однако локальные службы, такие как `cron`, которые используют Postfix для отправки отчетов о состоянии администраторам и пользователям, продолжают отправлять почту, используя адреса отправителей, соответствующие имени хоста. Это вызовет проблемы, т. к. вы не сможете ответить на такие сообщения.

   Для того чтобы пользователи не отправляли ответы таким приложениям, измените значение параметра `myorigin`, которое Postfix присоединяет в качестве доменной части адреса. Присвойте параметру `myorigin` имя домена, который на самом деле имеет почтовый сервер, а также почтовые ящики или псевдонимы для этих отпра-

вителей. Например, если внутренний почтовый сервер, являющийся местом конечного назначения для example.com, удовлетворяет этим требованиям, то вы можете указать:

```
myorigin = example.com
```

6. Отключить локальный агент доставки. В заключение вы можете запретить демону master запускать локальный агент доставки – это будет означать отключение локального агента доставки, т. к. на данной машине получателей нет. Отредактируйте файл master.cf и закомментируйте строку, содержащую локальную службу, поместив в начало строки знак решетки (#):

```
#
==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
#
==========================================================================
smtp      inet  n       -       n       -       -       smtpd
# local   unix  -       n       n       -       -       local
virtual   unix  -       n       n       -       -       virtual
lmtp      unix  -       -       n       -       -       lmtp
anvil     unix  -       -       n       -       1       anvil
```

После перезагрузки сервер Postfix не будет принимать сообщения для локальных получателей.

Использование Postfix с Microsoft Exchange Server

Microsoft Exchange Server, без сомнения, является мощнейшим сервером для рабочих групп, но вот его безопасность и устойчивость при возможных атаках уже не так высоки. Поэтому многие администраторы почтовых систем наращивают его функциональность за счет шлюза Postfix. В этом разделе будет рассказано о том, как предоставить хосту шлюза Postfix список действительных получателей и как автоматизировать эту процедуру.

Самый простой и распространенный способ организации совместной работы Postfix и Exchange Server заключается в обращении хоста ретрансляции Postfix к Exchange Server посредством LDAP (Lightweight Directory Access Protocol – облегченный протокол службы каталогов). Хост-ретранслятор будет запрашивать Exchange Server при каждом поступлении сообщения с целью определить, действителен ли получатель. Однако у этого подхода есть свои недостатки и ограничения. При альтернативном подходе Exchange Server «проталкивает» список получателей серверу Postfix, что предпочтительнее по следующим причинам:

Безопасность

Вне зависимости от того, какие программы работают на внутреннем почтовом сервере, вы захотите, чтобы его безопасности ничего не

угрожало. Поэтому в первую очередь вы помещаете его за межсетевым экраном. Одно из основных правил безопасности состоит в следующем: разрешать *только* то, что должно быть разрешено, и запрещать *все* остальное.

Первым порывом многих системных администраторов является использование Postfix-сервером LDAP-запросов для получения от удаленного сервера Exchange Server списка действительных получателей. Для этого вам необходимо открыть порт 389 (TCP/UDP) сервера Exchange Server, чтобы разрешить соединения от Postfix. Это достаточно легко, но порт оказывается открытым для внутренней локальной сети.

Повысить надежность можно, изменив направление, с тем чтобы Exchange предоставлял Postfix список действительных получателей только при изменении такого списка. Администратор Exchange Server посылает этот список интеллектуальному хосту посредством утилиты `scp` или `rsync`, исключая необходимость открытия порта в демилитаризованной зоне локальной сети.

Производительность

LDAP-запросы медленнее, чем используемые сервером Postfix индексированные карты. Если вы предоставляете Postfix статический список действительных получателей, интеллектуальный хост может обработать сообщения очень быстро.

Устойчивость

Интеллектуальный хост существует для защиты внутреннего почтового сервера, а получается все наоборот: атакуемый интеллектуальный хост вредит внутреннему серверу. Такое может случиться, когда спамеры используют атаки по словарю для одновременной рассылки сообщений большому количеству получателей: при этом почтовый шлюз, использующий LDAP, будет отправлять большое количество запросов серверу, который он должен был бы защищать, запрашивая сведения о действительных получателях. Это замедлит работу службы Active Directory (если не выведет ее из строя) и соответственно Exchange Server, что превратит атаку по словарю в DoS-атаку. Если почтовый сервер должен выйти из строя, пусть это будет внешний интеллектуальный хост, а не внутренний почтовый сервер.

В этом разделе мы передадим список действительных получателей от сервера Exchange 2003 Server почтовому шлюзу Postfix, выполнив следующие действия:

1. Экспорт списка всех действительных получателей.
2. Копирование списка на почтовый шлюз.
3. Извлечение действительных получателей из списка.
4. Создание карты получателей хоста-ретранслятора.

5. Индексирование карты получателей.
6. Автоматизация данных операций.

Экспорт действительных получателей из Active Directory

Microsoft использует в своей службе Active Directory атрибут `proxyAddresses`, в котором хранятся адреса действительных получателей для Exchange. Существует простой способ экспортирования `proxyAddresses` из Microsoft Active Directory – использование `csvde`, утилиты командной строки, доступной на любом сервере Exchange Server, – вам не потребуется писать собственный сценарий. Например, для того чтобы экспортировать значения `proxyAddresses` в файл `C:\export\example_com_recipients.txt`, вы можете просто использовать такую команду:

```
C:\> csvde -m -n -g -f "C:\export\example_com_recipients.txt" \
    -r "(|(&(objectClass=user)(objectCategory=person)) \
    (objectClass=groupOfNames) (objectClass=msExchDynamicDistributionList))" \
    -l proxyAddresses
```

Совет

Существуют тысячи способов организации и структурирования каталога Active Directory, так что поиск имен объектов для экспорта из вашего каталога может оказаться сложной задачей.

Вместе с сервером Exchange можно установить несколько средств поддержки, включая модуль ADSI Edit. Добавьте его в свою консоль управления Microsoft (MMC – Microsoft Management Console). Затем запустите `mmc.exe` из командной строки для получения полного доступа к именам объектов в Active Directory.

Приведенная выше команда выводит гораздо больше информации, чем требуется серверу Postfix. Вот каким может быть выходной файл:

```
DN,proxyAddresses
"CN=Administrator,CN=Users,DC=example,DC=com",smtp:abuse@example.com;SMTP:\
  Administrator@example.com; X400:c=DE\;a= \;p=Example Corporat\;
  o=Exchange\;s=Administrator\;;smtp:postmaster@example.com
"CN=Gast,CN=Users,DC=example,DC=com",
"CN=SUPPORT_388945a0,CN=Users,DC=example,DC=com",
"CN=krbtgt,CN=Users,DC=example,DC=com",
"CN=IUSR_MAIL,CN=Users,DC=example,DC=com",
"CN=IWAM_MAIL,CN=Users,DC=example,DC=com",
"CN=Wilma Pebble,OU=purchasing,DC=example,DC=com",\
  smtp:wilmapebble@example.com;smtp:wilma@example.com;\
  smtp:wilma.pebble@example.com;SMTP:w.pebble@example.com;smtp:\
  pebble@example.com; X400:c=DE\;a= \;p=Example Corporat\;
  o=Exchange\;s=Pebble\;g=Wilma\;
"CN=Betty McBricker,OU=purchasing,DC=example,\
  DC=com",smtp:mcbricker@example.com;smtp:\bettymcbricker@example.com;\
  smtp:betty@example.com;smtp:betty.mcbricker@example.com;\
  SMTP:b.mcbricker@example.com;X400:c=DE\;a= \;p=Example\
  Corporat\;o=Exchange\;s=McBricker\;g=Betty\;
"CN=Fred Flintstone,OU=sales,DC=example,DC=com",\
```

```
  smtp:fredflintstone@example.com;SMTP:fred.flintstone@example.com;\
  smtp:f.flintstone@example.com;smtp:fred@example.com;\
  smtp:flintstone@example.com;X400:c=DE\;a= \;p=Example Corporat\;
  o=Exchange\;s=Flintstone\;g=Fred\;
"CN=Barney Rubble,OU=sales,DC=example,DC=com",\
  SMTP:barney.rubble@example.com;smtp:barneyrubble@example.com;\
  smtp:rubble@example.com;smtp:barney@example.com;smtp:\
  b.rubble@example.com;X400:c=DE\;a= \;p=Example Corporat\;
  o=Exchange\;s=Rubble\;g=Barney\;
"CN=Bamm Bamm,OU=it,DC=example,DC=com",smtp:bammbamm@example.com;smtp:\
  bamm@example.com;smtp:bamm.bamm@example.com;SMTP:b.bamm@example.com;\
  X400:c=DE\;a= \;p=Example Corporat\;o=Exchange\;s=Bamm\;g=Bamm\;
"CN=SystemMailbox{C5C3EAFC-A32F-4925-85A5-3C08709DE617},\
  CN=Microsoft Exchange System\Objects,DC=example,DC=com",\
  SMTP:SystemMailbox{C5C3EAFC-A32F-4925-85A5-3C08709DE617}\
  @example.com;X400:c=DE\;a= \;p=Example\Corporat\;\
  o=Exchange\;s=SystemMailbox? C5C3EAFC-A32F-4925-85A5-3C\;
"CN=it-department,OU=it,DC=example,DC=com",SMTP:it-department@example.com;\
  X400:c=DE\;a= \;p=Example Corporat\;o=Exchange\;s=it-department\;
"CN=purchasing-department,OU=purchasing,DC=example,DC=com",\
  SMTP:purchasing-department@example.com;X400:c=DE\;a= \;p=Example\
  Corporat\;o=Exchange\;s=purchasing-department\;
"CN=sales-department,OU=sales,DC=example,DC=com",\
  SMTP:sales-department@example.com;\X400:c=DE\;a= \;p=Example Corporat\;
  o=Exchange\;s=sales-department\;
```

Действительным получателям соответствуют значения, помеченные `smtp` (псевдонимы) и `SMTP` (исходные адреса). Вам необходимо извлечь такие значения для создания списка, который мог бы использовать интеллектуальный хост Postfix. Со временем вы бы добились этого при помощи сценария на интеллектуальном хосте, но сейчас вам просто нужно передать список интеллектуальному хосту.

Отправка списка получателей ретранслятору почты

Копирование файла с сервера Exchange Server на интеллектуальный хост можно осуществить множеством способов, но один из лучших – использование утилиты `scp`, обеспечивающей шифрование и автоматизацию и поддерживаемой Windows и UNIX.

Автоматизация передачи файлов интеллектуальному хосту складывается из следующих этапов:

1. Установка клиента `scp` (защищенного копирования) для Windows, например PuTTY.
2. Создание пользователя для копирования на интеллектуальном хосте.
3. Создание ключей аутентификации.
4. Добавление открытых ключей к авторизованным.
5. Копирование секретного ключа на хост Windows.
6. Преобразование SSH-ключа к формату ключей PuTTY.
7. Копирование экспортного файла на интеллектуальный хост.

Получение клиента защищенного копирования для Windows

Одним из множества клиентов, позволяющих использовать `scp` для копирования файлов с хоста Windows, является PuTTY, бесплатный клиент telnet и SSH. Вы можете найти его по адресу *http://www.chiark.greenend.org.uk/~sgtatham/putty*.

Для выполнения необходимых нам операций скачайте файлы `pscp.exe` и `puttygen.exe` из данного пакета. Скопируйте исполняемые файлы в каталог, указанный в переменной окружения `path`, например `C:\Windows`.

Создание пользователя для копирования на интеллектуальном хосте

Для организации передачи файлов создаем на интеллектуальном хосте пользователя. Эта учетная запись будет использоваться только для получения экспортированного списка получателей. Например, вы можете создать пользователя `e3k` такой командой:

```
# useradd e3k
```

После создания пользователя определите для него пароль командой `passwd`. Вы будете использовать пароль в процессе настройки, а затем, когда все заработает, можете отключить его.

Создание ключей аутентификации

Следующим шагом будет создание набора ключей аутентификации, с тем чтобы передача файлов от сервера Windows интеллектуальному хосту не требовала использования пароля. От имени пользователя `root` на интеллектуальном хосте выполните команду `su - e3k` для переключения на пользователя `e3k` и выполните команду `ssh-keygen` для создания ключей, например:

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/e3k/.ssh/id_rsa):
Created directory '/home/e3k/.ssh'.
Enter passphrase (empty for no passphrase): ❶
Enter same passphrase again:
Your identification has been saved in /home/e3k/.ssh/id_rsa.
Your public key has been saved in /home/e3k/.ssh/id_rsa.pub.
The key fingerprint is:
17:7e:78:9e:39:0e:04:b7:ee:6d:39:28:c6:21:e4:84 e3k@mail.example.com
```

❶ Не вводите идентификационную[1] фразу, если хотите, чтобы процесс копирования был автоматическим. Если вы введете идентификацион-

[1] Строго говоря, это не идентификационная фраза, а ключевое слово, которое используется для шифрации ключа; если ключевое слово не вводить, то ключ будет храниться незашифрованным. Ничего страшного в этом нет, надо только гарантировать, что он ни к кому кроме вас не попадет. – *Примеч. науч. ред.*

ную фразу, то вам придется использовать ее при каждом копировании экспортного файла на интеллектуальный хост.

Предыдущая команда создает два файла: `.ssh/id_rsa` и `.ssh/id_rsa.pub`. Первый из них является секретным ключом, не выпускайте его из виду (он не должен находиться ни на одном хосте, кроме вашей Windows-машины).

Добавление открытого ключа к списку авторизованных ключей

Теперь, когда ключи готовы, вам нужно сообщить серверу SSH о только что созданном открытом ключе. Чтобы добавить открытый ключ из файла `id_rsa.pub` в список ключей в файле `$HOME/.ssh/authorized_keys` для пользователя, созданного с целью копирования (`e3k`), выполните следующую команду.

```
$ cd .ssh
$ cat id_rsa.pub >> authorized_keys
```

После создания файла `authorized_keys` убедитесь в том, что для него корректно определены права доступа, в противном случае сервер SSH откажется использовать файл для аутентификации:

```
$ chmod 644 authorized_keys
$ ls -l authorized_keys
-rw-r--r--    1 e3k      e3k           230 May 13 10:38 authorized_keys
```

Копирование секретного ключа в Windows

Затем вам нужно скопировать секретный ключ с интеллектуального хоста на ваш Windows-хост. Проще всего сделать это, используя команду `pscp.exe` на Windows-машине. Если IP-адрес вашего интеллектуального хоста `172.16.1.1`, то выполните такую команду:

```
> C:\export> pscp e3k@172.16.1.1:/home/e3k/.ssh/id_rsa .
e3k@172.16.1.1's password:
id_rsa                    |          0 kB |   0.9 kB/s | ETA: 00:00:00 | 100%
```

Используйте пароль, который вы создали в разделе «Создание пользователя для копирования на интеллектуальном хосте».

Преобразование SSH-ключа к формату ключа PuTTY

PuTTY использует формат, отличный от используемого большинством пакетов SSH в UNIX для хранения открытых и секретных ключей, так что для работы с секретным ключом вам придется преобразовать его в собственный формат PuTTY. Преобразовать формат ключа может утилита `puttygen`; запустите ее из командной строки следующим образом:

```
C:\export> puttygen id_rsa
```

Эта команда запускает GUI-клиент, который загружает секретный ключ. Вы должны увидеть диалоговое окно, показанное на рис. 13.2.

Нажмите `OK` для подтверждения. Откроется диалоговое окно `PuTTY Key Generator` (рис. 13.3). Укажите имя для преобразованного ключа (на-

пример, `example_com.ppk`) и нажмите кнопку Save private key (Сохранить секретный ключ).

Key Generator предупредит вас (рис. 13.4) о том, что в ключе задана пустая идентификационная фраза. Нажмите кнопку Yes для подтверждения и для сохранения секретного ключа. Теперь ваш хост Windows готов к использованию утилиты защищенного копирования для передачи файлов интеллектуальному хосту при помощи ключа аутентификации.

Рис. 13.2. *Успешный импорт ключа в PuTTYgen*

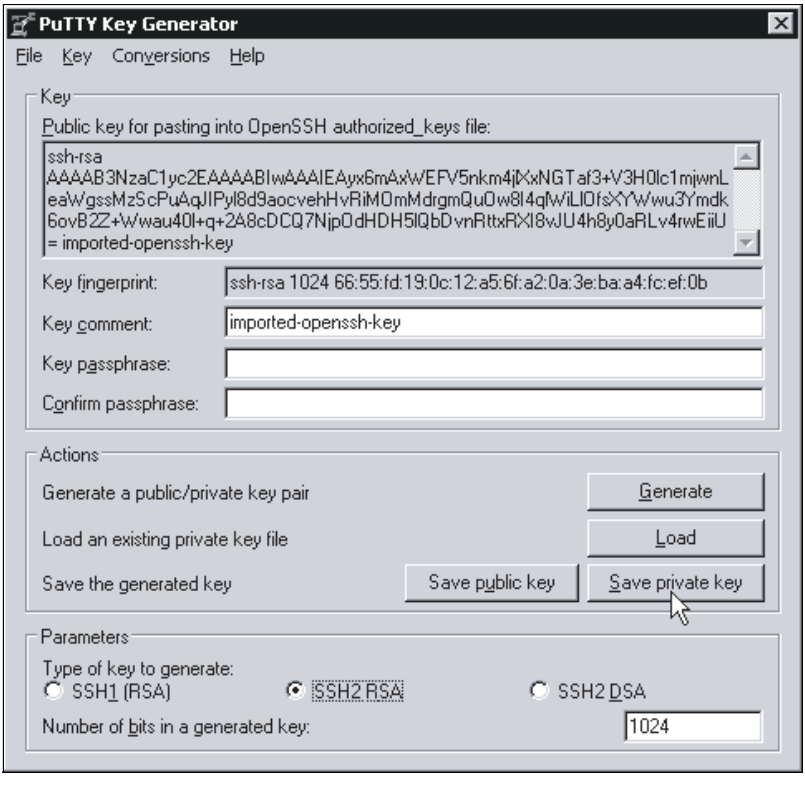

Рис. 13.3. *Сохранение секретного ключа PuTTYgen*

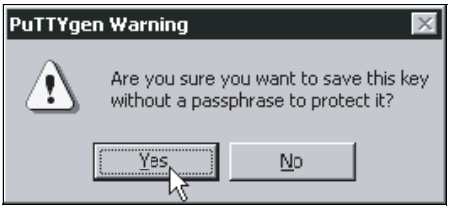

***Рис. 13.4.** Предупреждение PuTTY о пустой идентификационной фразе ключа*

Копирование списка получателей на интеллектуальный хост

Не закрывайте командное окно – оно понадобится для копирования файла с вашего Windows-хоста на интеллектуальный хост утилитой `pscp.exe` (это программа защищенного копирования, входящая в PuTTY). При запуске `pscp` вам необходимо указать секретный ключ, файл для копирования и пользователя, который осуществляет копирование. В нашем примере секретный ключ находится в файле `example_com.ppk`, копируемый файл – `example_com_recipients.txt`, а пользователь – `e3k`. Для копирования файла в каталог `/home/e3k` используем такую команду:

```
C:\export> pscp -i example_com.ppk example_com_recipients.txt
           e3k@172.16.1.1:/home/e3k/
Authenticating with public key "rsa-key-20040512"
example_com_recients.txt  |          2 kB |   2.4 kB/s | ETA: 00:00:00 | 100%
```

После успешного копирования файла на интеллектуальный хост вы можете сохранить команду экспортирования `csvde` из раздела «Экспорт действительных получателей из Active Directory» и команду `pscp` в командный файл с именем `export_valid_recipients.bat`. Впоследствии вы сможете запускать его щелчком мыши при создании, изменении или удалении получателя. Файл должен выглядеть примерно так:

```
csvde -m -n -g -f "C:\export\example_com_recipients.txt" \
    -r "(|(&(objectClass=user)(objectCategory=person)) \
    (objectClass=groupOfNames) (objectClass=msExchDynamicDistributionList))" \
    -l proxyAddresses
pscp -i example_com.ppk example_com_recients.txt e3k@172.16.1.1:/home/e3k/
```

После проверки работы этого командного файла вы можете закрыть доступ пользователю `e3k` к интеллектуальному хосту, используя команду вроде `usermod -L e3k`; тогда удаленный доступ к учетной записи `e3k` будет возможен только при наличии ключа аутентификации.

Создание карты получателей

Теперь на вашем интеллектуальном хосте есть файл экспорта Active Directory, и при помощи сценария вы можете извлечь из него получателей. При этом необходимо помнить о двух обстоятельствах:

- Для обозначения адресов получателей Microsoft использует как SMTP, так и smtp, так что сценарий должен отслеживать оба варианта.
- Ваш сценарий должен уметь исключать тех абонентов, которые не должны получать сообщения извне (примером может служить почтовый ящик SystemMailbox, используемый сервером Exchange для внутренней связи).

Следующий сценарий, extract_valid_recipients, извлекает всех действительных получателей и помещает их в файл, но не включает туда получателей, перечисленных в файле blacklist.

```
#!/bin/sh
# Извлекаем все адреса, которые начинаются с SMTP или smtp, из файла экспорта
# Active Directory, за исключением перечисленных в файле blacklist
cat $1 | tr -d \" | tr , \\n | tr \; \\n | awk -F\: '/(SMTP|smtp):/ \
  {printf("%s\tOK\n",$2)}' | \
  grep -v -f blacklist > $2
```

Файл blacklist выглядит так:

```
Administrator
SystemMailbox
```

Выполняем команду extract_valid_recipients для запуска сценария. Он выведет действительных абонентов в файл relay_recipients.

```
extract_valid_recipients /home/e3k/example_com_recipients.txt
relay_recipients
```

Результат должен выглядеть так:

```
abuse@example.com        OK
postmaster@example.com        OK
wilmapebble@example.com        OK
wilma@example.com        OK
wilma.pebble@example.com        OK
w.pebble@example.com        OK
pebble@example.com        OK
mcbricker@example.com        OK
bettymcbricker@example.com        OK
betty@example.com        OK
betty.mcbricker@example.com        OK
b.mcbricker@example.com        OK
fredflintstone@example.com        OK
fred.flintstone@example.com        OK
f.flintstone@example.com        OK
fred@example.com        OK
flintstone@example.com        OK
barney.rubble@example.com        OK
barneyrubble@example.com        OK
rubble@example.com        OK
barney@example.com        OK
b.rubble@example.com        OK
```

```
bammbamm@example.com        OK
bamm@example.com        OK
bamm.bamm@example.com        OK
b.bamm@example.com        OK
it-department@example.com        OK
purchasing-department@example.com        OK
sales-department@example.com        OK
```

Если список выглядит правильно, преобразуем его при помощи команды `postmap` (например, `postmap hash:relay_recipients`) и поместим в каталог, на который указывает параметр `relay_recipient_maps` (об этом мы говорили ранее в разделе «Определение получателей для пересылки»). Например, вы можете использовать такую команду:

```
# mv relay_recipients.db /etc/postfix/relay_recipients.db
```

Предупреждение

Параметр `relay_recipient_maps` не должен напрямую указывать на вашу только что созданную карту `relay_recipients` (например, `hash:/home/e3k/relay_recipients`)! Postfix прекратит работу, если карту не удастся преобразовать. Безопаснее сначала преобразовать карту, и только когда это удастся, переместить ее в каталог, обозначенный параметром `relay_recipient_maps`.

Создание карты доступа отправителей

В качестве приятного дополнения ваш файл экспорта Active Directory может предоставить Postfix список отправителей, которым разрешено отправлять почту во внешний мир. Для этого вы можете написать сценарий, подобный сценарию из раздела «Создание карты получателей», но с одним отличием: Microsoft использует `SMTP` для обозначения действительных адресов отправителей[1], так что сценарий извлечения адресов должен обрабатывать только элементы с такой пометкой.

Примечание

Это дает возможность не позволить вирусам использовать вашу папку контактов Outlook для создания фальшивых адресов отправителей и рассылки сообщений за пределы вашей сети.

Вы можете назвать сценарий `extract_valid_senders`, а выглядеть он будет так:

[1] В данном контексте «действительных адресов» следует читать как «адресов по умолчанию» или «основных адресов», поскольку у каждого владельца почтового ящика в Microsoft Exchange может быть несколько адресов, и только один из них – основной. Часто основной адрес – это адрес для «внешних» корреспондентов, в то время как остальные адреса могут использоваться для пересылки писем между владельцами почтовых ящиков одного Exchange-сервера. – *Примеч. науч. ред.*

```
#!/bin/bash
# Извлекаем все адреса, которые начинаются с SMTP, из файла экспорта
# Active Directory, за исключением перечисленных в файле blacklist
cat $1 | tr -d \" | tr , \\n| tr \; \\n | awk -F\: '/SMTP:/ \
{printf("%s\tOK\n",$2)}' | grep -v -f blacklist > $2
```

На этот раз, запустив следующую команду, вы должны будете получить более короткий, чем ранее, список, т. к. в нем нет псевдонимов:

```
# ./extract_valid_senders /home/e3k/example_com_recipients.txt
   example_com_senders
```

Вывод должен выглядеть так (используем в качестве основы пример из предыдущего раздела):

```
w.pebble@example.com        OK
b.mcbricker@example.com       OK
fred.flintstone@example.com       OK
barney.rubble@example.com       OK
b.bamm@example.com       OK
it-department@example.com       OK
purchasing-department@example.com       OK
sales-department@example.com       OK
```

В предыдущем примере мы перенаправляли вывод в файл `example_com_senders`. Теперь создадим из него индексированную базу данных при помощи команды `postmap hash:example_com_senders`. Затем создадим ограничение (см. главу 8), которое проверяет следующие условия для отправителей конвертов:

- Если сообщение приходит от внутреннего сервера, то отправитель конверта должен иметь действительный адрес.
- Если сообщение приходит не от внутреннего сервера, то ограничение не действует.

Чтобы настроить Postfix для применения данного условного ограничения, определите класс ограничения, который инициирует ограничение на отправителя конверта при поступлении сообщения от внутреннего сервера. Например, ваш файл `main.cf` может содержать следующее определение:

```
smtpd_restriction_classes =
    must_be_valid_sender ❶
must_be_valid_sender = ❷
    check_sender_access hash:/etc/postfix/example_com_senders
    reject
smtpd_recipient_restrictions =
    check_client_access hash:/etc/postfix/example_com_ip ❸
    reject_unauth_destination
    ...
```

❶ `must_be_valid_sender` — это имя ограничения, содержащего подмножество ограничений, которые применяются при поступлении сообще-

ний от внутреннего почтового сервера; в этой строке просто перечислены классы ограничений.

❷ Определение этого ограничения содержит процедуру для сообщений, поступающих от внутреннего сервера: проверка списка действительных отправителей конвертов и отклонение всех прочих отправителей.

❸ Параметр `check_client_access` запускает исполнение класса ограничений `must_be_valid_sender`.

Наконец, вам нужно добавить IP-адрес вашего внутреннего почтового сервера в файл `/etc/postfix/example_com_ip` вместе с действием, которое следует выполнить по ограничению. Например, следующая строка говорит, что, если сообщение придет с адреса `172.16.1.1`, Postfix должен применить ограничение `must_be_valid_sender`:

```
172.16.1.1   must_be_valid_sender
```

После добавления этих параметров вам следует перезагрузить конфигурацию, чтобы изменения вступили в силу.

Автоматизация процесса создания карты

Вы можете автоматизировать процесс создания карты на интеллектуальном хосте. Следующий пример использует Makefile, который вы можете загрузить с сайта *http://www.postfix-book.com*.

```
## Makefile для автоматизации создания карты
## параметры конфигурации
# местоположение файла, из которого извлекаем данные
ADS_DUMP=/home/e3k/example_com_recipients.txt
# местоположение файлов .proto
PROTO_PATH=relay_recipients
PROTO_PATH2=valid_senders
# место сохранения успешно созданных карт
MAP_PATH=/etc/postfix/relay_recipients
MAP_PATH2=/etc/postfix/valid_senders
# тип и суффикс создаваемых карт
DB_TYPE=hash
DB_SUFFIX=db
## параметры Makefile
#
# создаем все карты
all:                        $(MAP_PATH).$(DB_SUFFIX) $(MAP_PATH2)
                            .$(DB_SUFFIX) blacklist
# извлекаем действительных получателей из $(ADS_DUMP) в $(PROTO_PATH).proto
$(PROTO_PATH).proto:        $(ADS_DUMP)
                            ./extract_valid_recipients $(ADS_DUMP)
                            $(PROTO_PATH).proto
# извлекаем действительных отправителей из $(ADS_DUMP) в $(PROTO_PATH2).proto
$(PROTO_PATH2).proto:       $(ADS_DUMP)
                            ./extract_valid_senders $(ADS_DUMP)
                             $(PROTO_PATH2).proto
```

```
# создаем карту действительных получателей из $(PROTO_PATH).proto
$(MAP_PATH).$(DB_SUFFIX):   $(PROTO_PATH).proto
                            /usr/sbin/postmap -w $(DB_TYPE):
                            $(PROTO_PATH).proto && \
                            mv $(PROTO_PATH).proto.$(DB_SUFFIX)
                            $(MAP_PATH).$(DB_SUFFIX)
# создаем карту действительных отправителей из $(PROTO_PATH2).proto
$(MAP_PATH2).$(DB_SUFFIX):  $(PROTO_PATH2).proto
                            /usr/sbin/postmap -w $(DB_TYPE):\
                            $(PROTO_PATH2).proto && \
                            mv $(PROTO_PATH2).proto.$(DB_SUFFIX)\
                            $(MAP_PATH2).$(DB_SUFFIX)
# удаляем все промежуточные карты
clean:
                                rm -f $(PROTO_PATH).* $(PROTO_PATH2).* *~
```

После того как вы один раз запустили команду `make` для проверки работоспособности преобразования, можно создать задание `cron` для автоматического запуска. Например, следующее задание в файле `crontab` будет запускаться каждые 15 минут:

```
0,15,30,45 * * * * cd /root/relay_recipients && /usr/bin/make
```

Настройка взаимодействия Exchange и Postfix

В этом разделе будет рассказано о том, как настроить сервер Microsoft Exchange для ретрансляции всех сообщений через ваш шлюз Postfix, а также о том, как настроить сервер Exchange так, чтобы он случайно не завалил сообщениями сервер Postfix.

По умолчанию сервер Exchange не пересылает исходящие сообщения на шлюз. Для инициирования пересылки запустите Exchange System Manager из меню Programs и выполните следующие действия:

1. Выберите пункт Server в меню слева.
2. Выберите в поддереве свой почтовый хост.
3. Выберите пункт Protocols в меню Hosts.
4. Выберите протокол SMTP.
5. Нажав правую кнопку мыши на пункте SMTP, выберите Properties в записи Default Virtual SMTP Server.
6. Выберите в открывшемся окне свойств вкладку Delivery.

Теперь вы готовы завершить настройку, чему и будут посвящены последующие разделы.

Задание сервера Postfix в качестве интеллектуального хоста

Первое, что нужно сделать, – настроить сервер Exchange так, чтобы все исходящие сообщения отправлялись на ваш шлюз Postfix. Выберите Advanced Delivery на вкладке Delivery и введите полностью опреде-

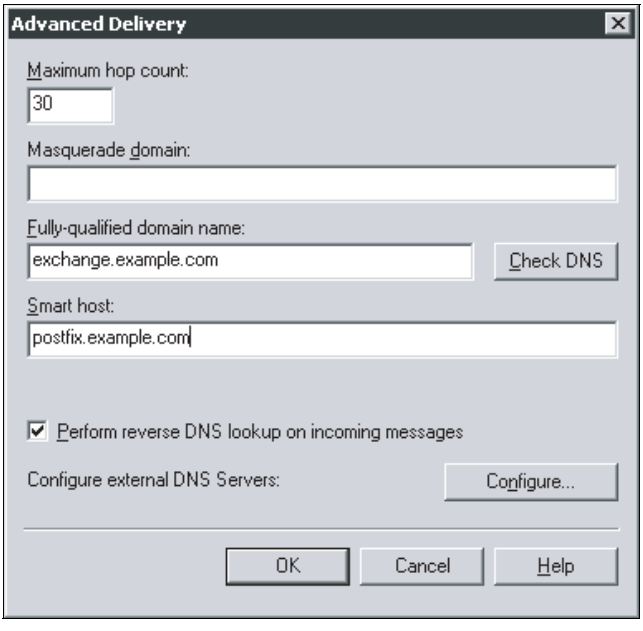

Рис. 13.5. *Настройка местоположения интеллектуального хоста в Exchange 2003*

ленное доменное имя (FQDN) вашего интеллектуального хоста Postfix (например, `postfix.example.com`), как показано на рис. 13.5.

Примечание

Сервер Exchange не принимает IP-адрес в качестве значения для поля Smart host. Добавьте интеллектуальный хост в DNS (внутренний), к которому обращается сервер Exchange, или задайте его статически при помощи файла `hosts` на вашем сервере Exchange.

После ввода полного имени интеллектуального хоста вам необходимо перезапустить виртуальный SMTP-сервер по умолчанию вашего сервера Exchange, чтобы изменения вступили в силу.

Ограничение исходящих соединений

Мы приложили массу усилий для защиты внутреннего сервера от грубого поведения со стороны интеллектуального хоста. Теперь следует защитить Postfix от перегрузки сообщениями, поступающими с вашего сервера Exchange. Нужно просто ограничить количество одновременных исходящих соединений.

Примечание

По умолчанию в Exchange разрешены 1000 одновременных соединений. Если исходящих сообщений будет так много (а это возможно в большой сети после

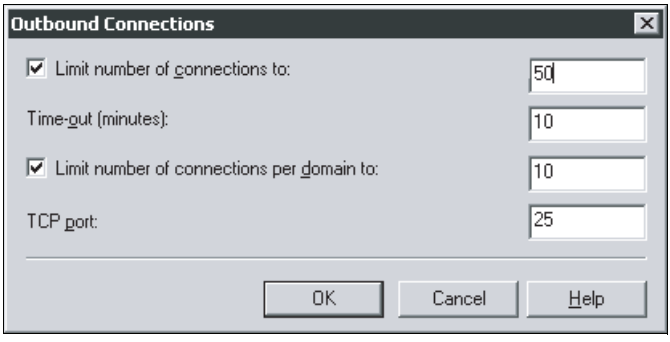

***Рис. 13.6.** Ограничение количества исходящих соединений в Exchange*

перезапуска SMTP-служб Exchange), эта огромная нагрузка за несколько минут будет переложена на интеллектуальный хост, что может потребовать от него слишком большого объема ресурсов, особенно если он обслуживает не один внутренний почтовый сервер.

В данном случае предпочтительно, чтобы сервер Exchange обрабатывал свою входящую почту более медленно. Если у вас нет доступа к серверу, вы также можете использовать Postfix-механизмы ограничения, описанные в главе 21, для наложения ограничения на количество соединений от хоста.

Выберите Outbound Connections на вкладке Delivery окна Default SMTP Virtual Server и установите максимальное количество одновременных соединений, равное 50, как показано на рис. 13.6 (проверено, что в обычных условиях это вполне подходящее значение).

Настройка NAT

Почтовый сервер Postfix, расположенный за шлюзом NAT, испытывает трудности, поскольку такой шлюз изменяет IP-пакеты (заменяя адрес назначения), прежде чем передать их в Postfix. Это означает, что `smtpd` будет прослушивать только частный IP-адрес шлюза, в то время как шлюз NAT принимает соединения на «официальный» IP-адрес.

Проблема возникает только из-за того, что согласно требованиям RFC почтовый сервер должен принимать сообщения, отправленные адресату `postmaster@[адрес]`, где `адрес` – это IP-адрес. Некоторые черные списки отправляют информацию об исключении из списка только по этому адресу.

Вы можете настроить Postfix для приема почты на адрес `postmaster@[адрес]`, используя параметр `proxy_interfaces`. Даже если Postfix будет прослушивать только внутренний IP-адрес, он будет принимать сообщения, адресованные `user@[адрес]`. Если адрес вашего шлюза NAT – `192.0.34.166`, используйте такую настройку:

```
proxy_interfaces = 192.0.34.166.
```

14

Почтовый сервер для нескольких доменов

Postfix может отправлять, получать и хранить сообщения для нескольких доменов двумя разными способами. Первый способ использует *домены виртуальных псевдонимов*, что просто увеличивает количество доменов, для которых сервер является местом конечного назначения. Второй подход, использующий *домены виртуальных почтовых ящиков*, более совершенен, т. к. домены виртуальных почтовых ящиков не требуют локальных учетных записей.

В этой главе будет рассказано о том, как реализовать оба метода, предоставляя службы SMTP более чем одному домену.

Домены виртуальных псевдонимов

Обычно сервер Postfix идентифицирует себя как место конечного назначения только для доменов, имена которых определены в параметре `mydestination`.[1] Домены, перечисленные в параметре `mydestination`, называются каноническими доменами, т. к. обычно они представляют собой имена локального компьютера (и иногда имя его родительского домена).

В этой главе мы покажем несколько способов сделать сервер Postfix местом конечного назначения для дополнительных доменов. Эти дополнительные домены называются виртуальными, т. к. у них нет ничего общего с собственным именем компьютера.

Для настройки базовых служб в домене виртуальных псевдонимов вы должны выполнить следующие действия:

---

[1] Postfix также идентифицирует себя как место конечного назначения для адресов в форме `user@[IP-адрес]`, где указан один из IP-адресов Postfix.

1. Задать имя домена виртуальных псевдонимов.
2. Создать карту адресов получателей.
3. Настроить Postfix для получения сообщений в домене виртуальных псевдонимов.
4. Протестировать новую конфигурацию.
5. Определить сложные отображения.

Эти действия будут описаны в последующих разделах.

Определение имени домена виртуальных псевдонимов

Сначала сообщаем серверу Postfix, что он является местом конечного назначения для некоторого домена в дополнение к системной настройке по умолчанию. Для определения карты виртуальных доменов Postfix использует параметр `virtual_alias_domains`. Для использования этого параметра создайте файл карты, например `/etc/postfix/virtual_alias_domains`, содержащий виртуальные домены:

```
# домены виртуальных псевдонимов
postfix-book.com        20021125
```

В этом примере число с правой стороны – это дата создания домена, но вы можете поместить туда все, что угодно. Postfix не использует правую часть при просмотре карты для параметра `virtual_alias_domains`, но карты Postfix всегда должны содержать как левую, так и правую часть.

После создания файла преобразуйте его в индексированную карту такой командой:

```
# postmap hash:/etc/postfix/virtual_alias_domains
```

Примечание

Если вы указываете домен в списке виртуальных, не используйте его в качестве значения параметра `mydestination`, т. к. возможны неожиданности. Postfix не будет знать, как ему поступить: доставлять ли почту локально или же отправлять ее на виртуальное преобразование. Поэтому Postfix в ответ на такую конфигурацию будет сильно ругаться в журнале.

Создание карты адресов получателей

Следующий этап настройки домена виртуальных псевдонимов состоит в создании файла `/etc/postfix/virtual_alias_maps` для отображения адресов получателей домена виртуальных псевдонимов на локальные адреса получателей. Следующий пример охватывает случаи отправки как одному, так и нескольким получателям.

```
# postfix-book.com
postmaster@postfix-book.com  ralf@example.com
abuse@postfix-book.com       abuse@example.com, patrick@example.com
ralf@postfix-book.com        ralf@example.com
patrick@postfix-book.com     patrick@example.com
```

Убедитесь в том, что задано соответствие для адресов postmaster и abuse, т. к. RFC требуют, чтобы все домены имели получателей для этих адресов.

Предупреждение

Всегда используйте полностью определенные доменные имена (FQDN) в адресах получателей в правой части файла virtual_alias_maps. В противном случае возможна неоднозначная интерпретация. Если вы укажете только локальную составляющую имени (например, ralf), то Postfix-демон trivial-rewrite добавит к адресу доменную часть, определяемую параметром myorigin. Имя пользователя ralf@$myorigin может оказаться некорректным (в зависимости от значений параметров myorigin и mydestination.).

После создания файла преобразуйте его в индексированную карту такой командой:

```
# postmap hash:/etc/postfix/virtual_alias_maps
```

Настройка Postfix для получения почты в доменах виртуальных псевдонимов

Теперь обе карты готовы, и вам нужно настроить Postfix на получение почты для вашего домена виртуальных псевдонимов согласно правилам, установленным в карте получателей. Необходимо задать в файле main.cf параметры virtual_alias_domains и virtual_alias_maps. Если использовать имена файлов из предыдущего раздела, то параметры будут такими:

```
virtual_alias_domains = hash:/etc/postfix/virtual_alias_domains
virtual_alias_maps = hash:/etc/postfix/virtual_alias_maps
```

Перезагрузите конфигурацию и протестируйте домены виртуальных псевдонимов, как описано в следующем разделе.

Проверка настроек доменов виртуальных псевдонимов

Вы можете проверить настройки вашего домена виртуальных псевдонимов, отправив сообщение существующему и неизвестному получателю в обоих доменах.

Отправка по действительному адресу в домене виртуальных псевдонимов

Отправляем сообщение действительному получателю (postmaster):

```
$ echo test | /usr/sbin/sendmail postmaster@postfix-book.com
```

Проверяем, куда попало сообщение, обратившись к файлу журнала. Вы должны увидеть там записи, подобные следующим:

```
Apr 19 11:20:50 mail postfix/pickup[17850]: B8C4629AB38: uid=0 from=<root>
Apr 19 11:20:50 mail postfix/cleanup[17863]: B8C4629AB38:
```

```
    message-id=<20040419092050.B8C4629AB38@mail.example.com>
  Apr 19 11:20:50 mail postfix/qmgr[17851]: B8C4629AB38:
    from=<root@mail.example.com>, size=282, nrcpt=1 (queue active)
  Apr 19 11:20:50 mail postfix/local[17866]: B8C4629AB38:
  to=<ralf@example.com>,
    orig_to=<postmaster@postfix-book.com>, relay=local, delay=0, status=sent
    (mailbox)
```

Тестовое сообщение сначала поступило на `postmaster@postfix-book.com`; затем согласно записям карты `virtual_alias_maps` почта для `postmaster@postfix-book.com` была отправлена на `ralf@example.com`. Сообщение было доставлено локально пользователю `ralf`, т. к. `example.com` — это «настоящий» домен.

Отправка на недействительный адрес в домене виртуальных псевдонимов

Вот как вы можете отправить сообщение недействительному получателю (`nouser`):

```
$ echo test | /usr/sbin/sendmail nouser@postfix-book.com
$ tail -f /var/log/mail.log
Apr 19 11:21:23 mail postfix/pickup[17850]: 9B61F29AB38: uid=0 from=<root>
Apr 19 11:21:23 mail postfix/cleanup[17863]: 9B61F29AB38:
  message-id=<20040419092123.9B61F29AB38@mail.example.com>
Apr 19 11:21:23 mail postfix/qmgr[17851]: 9B61F29AB38:
  from=<root@mail.example.com>, size=282, nrcpt=1 (queue active)
Apr 19 11:21:23 mail postfix/error[17887]: 9B61F29AB38:
  to=<nouser@postfix-book.com>, relay=none, delay=0, status=bounced
  (user unknown in virtual alias table)
```

Сообщение было адресовано `nouser@postfix-book.com`. Так как соответствующей записи в `virtual_alias_maps` не существует, сообщение для `nouser@postfix-book.com` возвращается с ошибкой «user unknown in virtual alias table» («пользователь не найден в таблице виртуальных псевдонимов»).

Сложные отображения

Чем больше доменов виртуальных псевдонимов вы добавляете, тем больше вероятность того, что вам придется добавлять в карту одинаковые записи. Вам на помощь могут прийти универсальные адреса (заданные посредством записей с регулярными выражениями и соответствий в неявном виде), о которых мы о поговорим в последующих разделах.

Универсальные адреса

В некоторых ситуациях может понадобиться, чтобы сообщение для неизвестного пользователя в домене виртуальных псевдонимов отправлялось на универсальный адрес (catchall address). На странице руководства virtual(5) приводится ряд способов, которыми вы можете сде-

лать это в записи карты virtual_alias_maps. Например, запись с наименьшим приоритетом может быть такой:

```
@postfix-book.com    catchall@example.com
```

Согласно этой записи, если ваш сервер Postfix не может найти совпадения для *неизвестный_пользователь*@postfix-book.com в карте виртуальных псевдонимов для postfix-book.com, то Postfix отображает этот адрес на catchall@example.com.

Записи с регулярными выражениями

Вы можете использовать в карте virtual_alias_maps регулярные выражения для отображения сообщений множества неизвестных пользователей в домене виртуальных псевдонимов на учетную запись catchall. Кроме того, вы можете заменить образец в левой части на адрес назначения в правой части (как показано в следующем примере). Это может быть удобно в случае, если вы отправляете найденные адреса в программу, которая указана в записи alias_maps.

Для того чтобы понять, как это работает, давайте посмотрим на такую запись virtual_alias_maps:

```
/^(.*)@postfix-book\.com$/    catchall+$1@example.com
```

При прибытии сообщения для неизвестного пользователя происходит следующее:

1. Сообщение для ***unknownuser***@postfix-book.com отображается на catchall+***unknownuser***@example.com.
2. Postfix доставляет сообщение локальному существующему получателю catchall@example.com, но в процессе доставки устанавливает переменную окружения $EXTENSION в значение *unknownuser*, как описано на странице руководства local(8). (Параметр recipient_delimiter задает разделитель для расширения; по умолчанию это +.)
3. Если программа обрабатывает почту для универсального адреса, она может использовать переменную окружения $EXTENSION для определения предполагаемого получателя и создать информационное сообщение для отправки обратно отправителю. Пример такой программы с именем fuzzy вы найдете по адресу *http://www.stahl.bau.tu-bs.de/~hildeb/postfix*.

Существует множество других способов использования регулярных выражений в карте virtual_alias_maps. В частности, очень удобно использовать их при отображении адресов, соответствующих некоторому шаблону, на одного получателя. Пусть у вас есть несколько адресов admin-имя@example.com, сообщения для которых должны попадать в один почтовый ящик. Вы можете использовать такую запись:

```
/^admin-.*@postfix-book\.com$/    mailbox@example.com
```

Это гораздо изящнее, чем указывать соответствие для каждого адреса вручную:

```
admin-firewall@postfix-book.com  mailbox@example.com
admin-mail@postfix-book.com      mailbox@example.com
admin-web@postfix-book.com       mailbox@example.com
...
```

Неявное соответствие для нескольких доменов

Иногда может быть удобно создать общее отображение, применяемое к нескольким доменам. Например, вы можете создать общего получателя – администратора почтовой системы, которому будут соответствовать все администраторы, вне зависимости от того, сколько доменов виртуальных псевдонимов вы поддерживаете. Для этого можно добавить в карту виртуальных псевдонимов такую запись:

```
postmaster   postmaster@example.com
```

В этом случае все сообщения, адресованные получателю, у которого локальная часть имени – `postmaster`, будут отправлены по адресу `postmaster@example.com`. Сервер Postfix будет принимать почту для перечисленных ниже адресов и доставлять ее по адресу `postmaster@example.com`:

- `postmaster@[$proxy_interfaces]`
- `postmaster@[$inet_interfaces]`
- `postmaster@$mydestination`

Обратите внимание, что все это домены виртуальных псевдонимов и что эти три домена необязательно покрывают все ваши домены виртуальных псевдонимов. Обратитесь к странице руководства virtual(5), на которой подробно описывается порядок поиска. Если вы хотите, чтобы все ваши домены виртуальных псевдонимов имели общий адрес для администратора почтовой системы, напишите сценарий, который добавит их в `virtual_alias_maps`.

Предупреждение

В карте нельзя использовать переменные конфигурации (такие как `$myorigin`). Postfix не будет подставлять значение переменной. Здесь такая запись приведена только для иллюстрации.

Домены виртуальных почтовых ящиков

Домены виртуальных почтовых ящиков – это домены для пользователей, у которых нет локальной учетной записи (т. е. для пользователей, которых нет в каталоге `/etc/passwd`). Появившись изначально в виде патча, включавшего в себя отдельный демон агента доставки, функция домена виртуальных почтовых ящиков превратилась в стандартный компонент Postfix.

Агент доставки `virtual` в Postfix базируется на агенте доставки `local`. В отличие от агента `local`, агент доставки `virtual` не имеет доступа

к системной информации о локальных пользователях (например, в /etc/passwd) для поиска имен получателей. Вместо этого агент доставки virtual полностью полагается на типы карт, которые не имеют ничего общего с вашей системой.

Существуют две причины, по которым следует запретить агенту доставки virtual знать что бы то ни было о системных учетных записях:

Масштабируемость

В Linux использование локальных учетных записей, определенных в файле /etc/passwd, ограничивает количество получателей для почтовых серверов значением, приблизительно равным 65 536. В Solaris и *BSD такого ограничения нет. Они имеют гораздо более длинные UID. Агент доставки virtual не связан такими ограничениями.

Безопасность

Вероятность взлома системы значительно уменьшается, если для простой отправки и получения почты локальным пользователям не требуются имена учетных записей и пароли.

Кроме того, виртуальный агент доставки не исполняет пользовательские команды интерпретатора и не присоединяет сообщения к пользовательским локальным файлам.

Так как агент доставки virtual ничего не знает о вашей системе, он не может обрабатывать такие файлы, как $HOME/.forward, а также использовать такие приложения, как procmail и vacation. Агент доставки virtual занимается только доставкой сообщений в почтовые ящики.

Проверка поддержки виртуального агента доставки в Postfix

Для использования доменов виртуальных почтовых ящиков демон master должен иметь возможность запускать демон virtual. Проверьте настройки в своем файле master.cf; по умолчанию демон включен, как в следующем примере:

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
# ==========================================================================
smtp      inet  n       -       n       -       -       smtpd
...
local     unix  -       n       n       -       -       local
virtual   unix  -       n       n       -       -       virtual
...
```

Примечание

Убедитесь в том, что демон virtual запущен не из-под chroot (см. пятый столбец предыдущего примера).

Базовая конфигурация

Для настройки базового виртуального домена почтовых ящиков вам следует сделать так, чтобы агент доставки `virtual` сохранял все сообщения, используя одни и те же UID и GID в плоской иерархии. Вам необходимо выполнить следующие действия:

1. Определение имени домена виртуальных почтовых ящиков.
2. Определение владельца файла для агента доставки `virtual`.
3. Определение базового каталога для почтовых ящиков домена.
4. Создание карты получателей.
5. Создание карты псевдонимов.

Определение имени домена виртуальных почтовых ящиков

Сначала необходимо сообщить серверу Postfix о том, что он является местом конечного назначения для одного или нескольких доменов виртуальных почтовых ящиков, указав список доменов в параметре `virtual_mailbox_domains` в файле `main.cf`. Например, если вы хотите создать домен виртуальных почтовых ящиков для `example.com`, используйте такую запись:

```
virtual_mailbox_domains = example.com
```

Определение владельца файла

Несмотря на то, что домены виртуальных почтовых ящиков не требуют, чтобы каждый почтовый ящик имел уникального пользователя, вам все же нужен хотя бы один идентификатор пользователя (UID) и один идентификатор группы (GID) для предоставления агенту доставки `virtual` доступа к почтовым ящикам. Для этого вам необходимо определить карты владельцев при помощи параметров `virtual_uid_maps` и `virtual_gid_maps`.

Определение пользователя

Для определения владельца почтового ящика вы должны создать локального пользователя для почтовых ящиков, если это еще не сделано. Чтобы создать пользователя почтовых ящиков с именем `vuser` и идентификатором пользователя `1000`, выполните такую команду:

```
# useradd vuser -u 1000
```

Предупреждение

По умолчанию владелец почтового ящика не может иметь UID меньше `100`. Это мера безопасности, задаваемая параметром `virtual_minimum_uid` для предотвращения перезаписи агентом `virtual` чувствительных файлов, принадлежащих системным учетным записям. Вы можете задать другое граничное значение в параметре `virtual_minimum_uid` своего файла `main.cf`.

Когда владелец почтового ящика определен, вы должны сказать агенту virtual, что он должен использовать этот UID при записи сообщений в файловую систему. Укажите UID в параметре virtual_uid_maps своего файла main.cf, как это сделано в следующем примере для UID, равного 1000:

```
virtual_uid_maps = static:1000
```

Примечание

Ключевое слово static для UID необходимо, чтобы демон virtual использовал исключительно этот UID. Вы также можете применять UID динамически, о чем мы поговорим в разделе «Тонкая настройка».

Определение группы

В дополнение к только что заданному пользователю вам необходима локальная группа. При базовой настройке создаем GID с тем же номером, что и UID в предыдущем разделе. Возможно, что ваша команда useradd уже сделала это за вас (проверьте свой файл /etc/group), если же нет, используйте такую команду:

```
# groupadd vuser -g 1000
```

Теперь задайте параметр virtual_gid_maps в своем файле main.cf так же, как и для пользователя виртуального почтового ящика:

```
virtual_gid_maps = static:1000
```

Определение базового каталога домена виртуальных почтовых ящиков

Агент доставки virtual должен знать, где искать почтовые ящики для своих получателей. Обычно переменные окружения и файлы конфигурации операционной системы сообщают приложениям системные настройки по умолчанию. Агент доставки virtual не распознает переменные окружения, поэтому вам необходимо явно сообщить ему, куда помещать сообщения, которые он должен доставить пользователям.

Установите параметр virtual_mailbox_base в своем файле main.cf, для того чтобы указать, где сохранять входящие сообщения, например:

```
virtual_mailbox_base = /var/spool/virtual_mailboxes
```

Примечание

Полный путь индивидуального виртуального почтового ящика состоит из значения virtual_mailbox_base и значения в карте поиска (она будет описана в следующем разделе). Другими словами, это $virtual_mailbox_base/$mailboxname.

После установки параметра virtual_mailbox_base не помешает на самом деле создать каталог и сделать его доступным пользователю, определенному в предыдущих разделах:

```
# mkdir /var/spool/virtual_mailboxes
# chown vuser:vuser /var/spool/virtual_mailboxes
# chmod 700 /var/spool/virtual_mailboxes
```

Создание карты получателей

Вы должны определить получателей домена виртуальных почтовых ящиков в карте. Например, вы можете создать файл /etc/postfix/virtual_mailbox_recipients с полными именами получателей с левой стороны и именами почтовых ящиков с правой стороны. Вот как он мог бы выглядеть:

```
wilma.pebble@example.com          wilmapebble
betty.mcbricker@example.com       bettymcbricker
fred.flintstone@example.com       fredflintstone
barney.rubble@example.com         barneyrubble
bamm.bamm@example.com             bammbamm/
```

Демон virtual добавляет значение параметра virtual_mailbox_base перед именем почтового ящика для образования полного имени файла почтового ящика. По умолчанию почтовые ящики имеют формат mbox, но вы можете задать формат Maildir, добавив косую черту (/) в конец имени почтового ящика, как в предыдущей записи для bamm.bamm@example.com.

Разобравшись с файлом, вы должны создать индексированную версию карты, выполнив такую команду:

```
# postmap hash:/etc/postfix/virtual_mailbox_recipients
```

Затем вы можете сообщить Postfix, где искать карту, задав параметр virtual_mailbox_maps в файле main.cf, например:

```
virtual_mailbox_maps = hash:/etc/postfix/virtual_mailbox_recipients
```

Ограничения карт получателей

По соображениям безопасности на карты получателей накладываются следующие ограничения:

- Получатели доменов виртуальных почтовых ящиков не могут использовать расширение адреса, такое как user+extension@domain.tld.
- Демон virtual, в отличие от демона local, не может вызывать внешние программы.
- Карты с регулярными выражениями разрешены, но вы не можете использовать подстановку выражений (т. е. вы не можете использовать $1 в правой части).
- Вы не можете выполнять просмотры таблиц посредством демона proxymap.

Создание карты псевдонимов

У вас могут быть псевдонимы для домена виртуальных почтовых ящиков, но вы должны поместить их в отдельную карту, например /etc/postfix/virtual_mailbox_aliases. С левой стороны должно стоять полное имя псевдонима, а справа – полное имя почтового ящика, например:

```
wilma@example.com          wilma.pebble@example.com
pebble@example.com         wilma.pebble@example.com
...
postmaster@example.com     bamm.bamm@example.com
abuse@example.com          bamm.bamm@example.com
```

Как и для других карт, вам следует создать индексированную версию такой командой:

```
# postmap hash:/etc/postfix/virtual_mailbox_aliases
```

Наконец, мы говорим Postfix, что нужно использовать карту псевдонимов, установив параметр virtual_alias_maps в файле main.cf, например:

```
virtual_alias_maps = hash:/etc/postfix/virtual_mailbox_aliases
```

После перезагрузки конфигурации ваш почтовый сервер будет принимать сообщения для получателей домена виртуальных почтовых ящиков.

Тонкая настройка

Если вам нужно предоставлять почтовые услуги нескольким доменам виртуальных почтовых ящиков, существует вероятность того, что хранение всех сообщений в одном каталоге может вызвать проблемы, т. к. возможно существование двух пользователей с одним именем. Кроме того, очень сложным становится раздельное резервное копирование данных. Для решения таких вопросов вы можете настроить домены виртуальных почтовых ящиков так, чтобы сообщения для разных доменов хранились в разных каталогах. Вы также можете использовать разные идентификаторы пользователей (UID) и групп (GID).

Для такой расширенной настройки вам необходимо выполнить следующие действия:

1. Определить имена доменов виртуальных почтовых ящиков.
2. Определить владельцев файлов для виртуального агента доставки.
3. Определить базовый каталог для почтовых ящиков домена.
4. Создать карту получателей.
5. Создать карту псевдонимов.
6. Определить права доступа на запись и чтение.

Определение имен виртуальных доменов

Как уже говорилось в разделе «Определение имени домена виртуальных почтовых ящиков», мы задаем имена доменов виртуальных ящиков в параметре `virtual_mailbox_domains`. Приведем пример для двух доменов:

```
virtual_mailbox_domains = example.com, postfix-book.com
```

Определение владельцев файлов

При расширенной настройке вам необходимо создать пары UID и GID для каждого домена виртуальных получателей. Допустим, что вы хотите использовать имя пользователя и имя группы `example` для домена `example.com` и `pfxbook` – для домена `postfix-book.com`. Для создания пользователей (см. раздел «Определение владельца файла» выше в этой главе) можно использовать такие команды:

```
# useradd example -u 1001
# useradd pfxbook -u 1002
# groupadd example -g 1001
# groupadd pfxbook -g 1002
```

Обратите внимание на эти UID и GID; вскоре мы снова будем их использовать при создании карт поиска в разделе «Определение прав на запись и чтение».

Определение базового каталога для доменов виртуальных почтовых ящиков

Вам необходимо задать параметр `virtual_mailbox_base`, чтобы сообщить демону `virtual`, где он должен хранить сообщения, подобно тому, как это делалось в разделе «Определение базового каталога домена виртуальных почтовых ящиков».

Давайте используем ту же настройку, что и в том разделе:

```
virtual_mailbox_base = /var/spool/virtual_mailboxes
```

Однако отличие между созданной ранее базовой конфигурацией и тем, что мы делаем сейчас, состоит в том, что нам следует изменить привилегии для `virtual_mailbox_base`. В противном случае демон `virtual`, используя разные UID и GID для каждого пользователя и домена при сохранении сообщения, не будет иметь права на запись в подкаталоги:

```
# mkdir /var/spool/virtual_mailboxes
# chown vuser:vuser /var/spool/virtual_mailboxes
# chmod 775 /var/spool/virtual_mailboxes
```

Теперь вам следует создать подкаталоги (например, `example.com` и `postfix-book.com`), т. к. демон `virtual` создаст только почтовый ящик mbox или Maildir для получателя, но не родительский каталог для его домена.

```
# mkdir example.com
# chown example example.com/
# chgrp example example.com/
# chmod 700 example.com/
```

Предупреждение

Postfix версии 2.0 и ранее не будет создавать почтовые ящики типа Maildir в общедоступных родительских каталогах; вам нужно будет создать почтовый ящик Maildir заранее.

Если доставить почту не удается из-за каких-либо проблем с привилегиями, то в почтовом журнале должны появиться подобные сообщения:

```
May 26 12:04:33 mail postfix/virtual[14196]: warning: maildir access problem
  for UID/GID=1002/1002: create /var/spool/mailboxes/postfix-book.com/
  patrick/tmp/1085565873.P14196.mail.example.com: Permission denied
May 26 12:04:33 mail postfix/virtual[14196]: warning: perhaps you need to
  create the maildirs in advance
```

Создание карты получателей

Теперь вам нужно создать карту действительных получателей в ваших доменах виртуальных почтовых ящиков. Процесс аналогичен описанному в предыдущем разделе «Создание карты получателей» с тем лишь отличием, что имя почтового ящика должно предваряться именем каталога. В этом случае сообщения для разных доменов попадут в разные каталоги, и вам не придется беспокоиться о конфликте имен.

Например, вы можете создать файл `/etc/postfix/virtual_mailbox_recipients` следующим образом:

```
wilma.pebble@example.com          example.com/wilmapebble/
betty.mcbricker@example.com       example.com/bettymcbricker/
fred.flintstone@example.com       example.com/fredflintstone/
barney.rubble@example.com         example.com/barneyrubble/
bamm.bamm@example.com             example.com/bammbamm/
ralf@postfix-book.com             postfix-book.com/ralf/
patrick@postfix-book.com          postfix-book.com/patrick/
```

Помните, что после создания карты вам необходимо построить ее индексированную версию:

```
# postmap hash:/etc/postfix/virtual_mailbox_recipients
```

Как и ранее, присваиваем параметру `virtual_mailbox_maps` имя созданной карты в файле `main.cf`:

```
virtual_mailbox_maps = hash:/etc/postfix/virtual_mailbox_recipients
```

Таким образом обслуживаются все получатели, за исключением входящих в карты псевдонимов, созданных нами ранее в разделе «Создание карты псевдонимов».

Определение прав на запись и чтение

Вы *не можете* определить разные права на запись и чтение для разных доменов виртуальных почтовых ящиков, как было описано ранее в разделе «Определение владельцев файлов». Вместо этого вы должны создать карты, сопоставляющие почтовые ящики идентификаторам пользователя и группы.

Теперь вам понадобятся UID и GID, которые мы создали в разделе «Определение владельцев файлов». Вы должны присвоить UID каждому почтовому ящику в карте, указанной в параметре `virtual_uid_maps`. Например, следующей строкой в файле `main.cf` вы можете дать карте имя `hash:/etc/postfix/virtual_uid_map`:

```
virtual_uid_maps = hash:/etc/postfix/virtual_uid_map
```

Теперь помещаем в эту карту получателей, указывая полный адрес получателя слева и UID справа, например:

```
wilma.pebble@example.com          1001
betty.mcbricker@example.com       1001
fred.flintstone@example.com       1001
barney.rubble@example.com         1001
bamm.bamm@example.com             1001
ralf@postfix-book.com             1002
patrick@postfix-book.com          1002
```

Примечание

Не забывайте о создании индексированной версии карты при помощи команды `postmap`. Еще более короткая версия будет такой:

```
@example.com 1001
@postfix-book.com 1002
```

Отображение GID работает точно так же, как отображение UID. Приведем пример карты, которую вы можете использовать для данного примера, в файле `/etc/postfix/virtual_gid_map`:

```
wilma.pebble@example.com          1001
betty.mcbricker@example.com       1001
fred.flintstone@example.com       1001
barney.rubble@example.com         1001
bamm.bamm@example.com             1001
ralf@postfix-book.com             1002
patrick@postfix-book.com          1002
```

Указываем этот файл в параметре `virtual_gid_maps`:

```
virtual_gid_maps = hash:/etc/postfix/virtual_gid_map
```

Совет

Так как записи для параметра `virtual_gid_maps` в данном примере полностью совпадают с записями для параметра `virtual_uid_maps`, вы можете пропустить

этап создания файла карты GID и просто сослаться на карту UID в своем файле `main.cf`:

```
virtual_gid_maps = $virtual_uid_maps
```

После завершения работы с картами и файлом конфигурации перезагрузите Postfix, чтобы демон `virtual` мог доставить сообщения в подкаталоги, названные в соответствии с доменами получателей.

Формирование карт посредством сценариев

Как вы могли заметить, демону `virtual` нужны как минимум три карты для поиска получателей, почтовых ящиков, прав доступа для владельцев и групп. Вы можете значительно упростить себе жизнь, сделав так, чтобы сценарий создавал все карты на основе одного исходного файла (например, `/etc/postfix/virtual_build_map_source`), который содержит всю необходимую информацию. Пусть исходный файл содержит следующие строки:

```
wilma.pebble@example.com          example.com/wilmapebble/      1001    1001
betty.mcbricker@example.com       example.com/bettymcbricker/   1001    1001
fred.flintstone@example.com       example.com/fredflintstone/   1001    1001
barney.rubble@example.com         example.com/barneyrubble/     1001    1001
bamm.bamm@example.com             example.com/bammbamm/         1001    1001
ralf@postfix-book.com             postfix-book.com/ralf/        1002    1002
patrick@postfix-book.com          postfix-book.com/patrick/     1002    1002
```

Следующий сценарий (назовем его `/etc/postfix/build_virtual_maps`) читает данные из исходного файла и создает три необходимые карты:

```
# !/bin/bash
#
# Создание всех необходимых карт виртуальных почтовых ящиков из одного
источника
# раздел: пути
SOURCE=/etc/postfix/virtual_build_map_source
VMAP=/etc/postfix/virtual_mailbox_recipients
VUID=/etc/postfix/virtual_uid_map
VGID=/etc/postfix/virtual_gid_map
AWK=/usr/bin/awk
POSTMAP=/usr/sbin/postmap
# раздел: создание
# создание $virtual_mailbox_maps
$AWK '{printf("%s %s\n",$1,$2)}' $SOURCE > $VMAP
$POSTMAP hash:$VMAP
# создание $virtual_uid_maps
$AWK '{printf("%s %s\n",$1,$3)}' $SOURCE > $VUID
$POSTMAP hash:$VUID
# создание $virtual_gid_maps
$AWK '{printf("%s %s\n",$1,$4)}' $SOURCE > $VGID
$POSTMAP hash:$VGID
```

Примечание

Вы можете загрузить сценарий с сайта *http://www.postfix-book.com* (возможно, вам понадобится изменить пути в начале сценария).

После запуска сценария все карты, кроме исходного файла и карты виртуальных псевдонимов, должны иметь одинаковую дату и время:

```
-rw-r--r--   1 root   root   532 May 26 12:12 virtual_build_map_source
-rw-r--r--   1 root   root   251 May 26 13:21 virtual_gid_map
-rw-r--r--   1 root   root   12288 May 26 13:21 virtual_gid_map.db
-rw-r--r--   1 root   root   394 May 26 13:21 virtual_mailbox_recipients
-rw-r--r--   1 root   root   12288 May 26 13:21
                                    virtual_mailbox_recipients.db
-rw-r--r--   1 root   root   251 May 26 13:21 virtual_uid_map
-rw-r--r--   1 root   root   12288 May 26 13:21 virtual_uid_map.db
```

Управляемые базой данных домены виртуальных почтовых ящиков

Если вам нужен агент передачи сообщений для предприятия или интернет-провайдера, то вы можете сделать так, чтобы сервер Postfix обращался за информацией о доменах виртуальных почтовых ящиков к базе данных. Эта способ обеспечивает особую гибкость, т. к. вы сможете передавать администрирование пользователей другим людям, не предоставляя им доступ с правами `root` к серверу. Если вы предоставите развитый веб-интерфейс, ваши клиенты смогут управлять собственными данными (добавлять псевдонимы, изменять свои пароли SMTP AUTH, POP3 и IMAP и т. д.). Более того, база данных обновляется сразу же, так что вам не придется перезагружать Postfix при каждом изменении пользовательских данных.

С другой стороны, доступ к индексированным картам является более быстрым, и просмотры карт потребляют не так много системных ресурсов, как SQL-запросы, т. к. не требуют работы сервера базы данных. Более того, использующее базу данных решение может оказаться более сложным.

Если у вас возникнут проблемы производительности при поиске в базе данных, вы можете использовать выделенный сервер базы данных, который может быть доступен многим серверам Postfix (и другим службам) в вашей сети. Применяя механизмы распределения нагрузки, такие как круговой метод обслуживания (round robin) и специальное оборудование, с помощью базы данных вы можете создать высокопроизводительную почтовую службу.

В этом разделе будет рассказано о том, как реализовать управляемые базой данных домены виртуальных почтовых ящиков, используя в качестве базы данных MySQL. Вот что вам нужно сделать:

1. Проверить поддержку карт MySQL в Postfix.
2. Собрать Postfix с поддержкой карт MySQL.
3. Настроить базу данных.
4. Протестировать управляемые базой данных домены виртуальных почтовых ящиков.

Примечание

Управляемые базой данных карты могут использоваться не только в доменах виртуальных почтовых ящиков, но и для других целей. Postfix также поддерживает запросы к PostgreSQL и LDAP (конфигурация PostgreSQL практически идентична MySQL; о LDAP мы поговорим в главе 19).

Проверка Postfix на поддержку карт MySQL

Прежде чем настраивать Postfix для обращения к MySQL, следует убедиться, что установленный у вас сервер Postfix действительно поддерживает этот тип карт. Используйте команду `postconf -m` для вывода поддерживаемых типов карт. Если MySQL поддерживается, то в списке вы увидите `mysql`, как в следующем примере:

```
# postconf -m
btree
cidr
environ
hash
ldap
mysql
nis
pcre
proxy
regexp
sdbm
static
unix
```

Если же MySQL не поддерживается, вам нужно или установить пакет Postfix с поддержкой MySQL из дистрибутива для вашей операционной системы, или собрать его самостоятельно вручную и установить новую версию (эти действия будут описаны в следующем разделе).

Сборка Postfix с поддержкой карт MySQL

Для сборки Postfix с поддержкой таблиц MySQL сначала определите, где находятся заголовочные файлы и библиотеки, которые нужны для сборки Postfix. Для поиска каталога заголовочных файлов используйте такую команду:

```
# find /usr -name 'mysql.h'
/usr/include/mysql/mysql.h
```

Вывод предыдущей команды показывает, что заголовочные файлы данной конкретной системы находятся в каталоге `/usr/include/mysql`. Для поиска клиентских библиотек MySQL выполните такую команду:

```
# find /usr -name 'libmysqlclient.*'
/usr/lib/mysql/libmysqlclient.so.10
/usr/lib/mysql/libmysqlclient.so.10.0.0
/usr/lib/mysql/libmysqlclient.so
/usr/lib/mysql/libmysqlclient.a
```

Вывод показывает, что библиотеки находятся в каталоге `/usr/lib/mysql`.

Теперь вы знаете пути и можете задать переменные для процесса сборки Postfix с помощью Makefile. Для настройки и сборки Postfix в нашем примере для полученных путей используйте такую команду:

```
$ make tidy
$ make makefiles CCARGS='-DHAS_MYSQL -I/usr/include/mysql'
  AUXLIBS='-L/usr/lib/mysql -lmysqlclient -lz -lm'
$ make
```

После завершения сборки и установки Postfix проверьте, действительно ли MySQL поддерживается (см. предыдущий раздел).

Настройка базы данных

Когда вы будете готовы к созданию базы данных MySQL для хранения информации о ваших виртуальных доменах, подключитесь к MySQL как пользователь `root` и создайте базу данных. Следующая команда создает базу данных с именем `mail`:

```
mysql> CREATE DATABASE `mail`;
```

Для того чтобы вносить записи в базу данных `mail` так, как показано ниже, следует выбрать эту базу перед началом создания таблиц:

```
mysql> use mail;
```

Совет

Вы можете скачать полный набор команд SQL, использованных в данном разделе, с сайта *http://www.postfix-book.com*. Если вам необходимо сначала получить какую-то базовую информацию о командах SQL, обратитесь к введению в SQL («A Gentle Introduction to SQL») на сайте *http://sqlzoo.net*.[1]

[1] Описание языка SQL и администрирование mysql см. по адресу *www.mysql.com*. – *Примеч. науч. ред.*

Создание таблицы доменов получателей

Создаем таблицу `virtual_mailbox_domains` для хранения доменов, по отношению к которым сервер Postfix будет считать себя местом конечного назначения. Можно использовать такую команду:

```
mysql> CREATE TABLE `virtual_mailbox_domains` (
mysql>   `Id` int(10) unsigned NOT NULL auto_increment,
mysql>   `domain` varchar(255) default NULL,
mysql>   PRIMARY KEY  (`Id`),
mysql>   FULLTEXT KEY `domains` (`domain`)
mysql> ) TYPE=MyISAM COMMENT='Postfix virtual aliases';
```

Если команда выполнена успешно, то вы можете добавлять в таблицу свои виртуальные домены. Например, для добавления в таблицу домена `example.com` выполните такую команду SQL, вставляющую строку в таблицу:

```
mysql> INSERT INTO virtual_mailbox_domains VALUES (1,'example.com');
```

Добавление пользователей

Пришло время создать таблицу, в которой каждая строка содержит получателя, имя почтового ящика, UID и GID. Вы можете назвать ее `virtual_users`; структура таблицы будет аналогична столбцам файла `/etc/postfix/virtual_build_map_source`, который мы использовали ранее в разделе «Формирование карт посредством сценариев».

Примечание

Следующая таблица создается на основе таблицы SMTP AUTH, описанной в главе 18. Она содержит пароли и другую информацию, так что вы можете использовать ее как источник данных для SMTP-аутентификации, а также для доменов виртуальных почтовых ящиков.

Выполните такую команду для создания таблицы `virtual_users`:

```
mysql> CREATE TABLE `virtual_users` (
mysql>   `id` int(11) unsigned NOT NULL auto_increment,
mysql>   `username` varchar(255) NOT NULL default '0',
mysql>   `userrealm` varchar(255) NOT NULL default 'mail.example.com',
mysql>   `userpassword` varchar(255) NOT NULL default '1stP@ss',
mysql>   `auth` tinyint(1) default '1',
mysql>   `active` tinyint(1) default '1',
mysql>   `email` varchar(255) NOT NULL default '',
mysql>   `virtual_uid` smallint(5) default '1000',
mysql>   `virtual_gid` smallint(5) default '1000',
mysql>   `virtual_mailbox` varchar(255) default NULL,
mysql>   PRIMARY KEY  (`id`),
mysql>   UNIQUE KEY `id` (`id`),
mysql>   FULLTEXT KEY `recipient` (`email`)
mysql> ) TYPE=MyISAM COMMENT='SMTP AUTH and virtual users';
```

Поле active является необязательным; вы можете использовать его для включения или отключения почтового ящика получателя (удобно в том случае, если клиент не платит, и вы хотите прервать его обслуживание, но не уничтожать учетную запись).

Когда таблица готова, следует добавить в нее данные для тестирования. Вставим пробную строку такой командой:

```
mysql> INSERT INTO virtual_users VALUES (5,'bamm.bamm',
        'mail.example.com','1stP@ss',1,1,
mysql> 'bamm.bamm@example.com',1001,1001,'example.com/bammbamm/');
```

Создание таблицы для виртуальных псевдонимов

Последняя таблица, которую нам нужно создать, будет хранить виртуальные псевдонимы. Как и в других таблицах псевдонимов, строки будут содержать имя псевдонима и реальный адрес получателя. Создаем таблицу (в данном примере – с именем virtual_aliases):

```
mysql> CREATE TABLE `virtual_aliases` (
mysql>   `Id` int(10) unsigned NOT NULL auto_increment,
mysql>   `alias` varchar(255) default NULL,
mysql>   `virtual_user_email` text,
mysql>   PRIMARY KEY  (`Id`),
mysql>   FULLTEXT KEY `aliases` (`alias`,`virtual_user_email`)
mysql> ) TYPE=MyISAM COMMENT='Postfix virtual recipients';
```

Теперь нашу таблицу, как и любую другую, надо наполнить данными. Начнем с псевдонимов, наличия которых требуют RFC:

```
mysql> INSERT INTO virtual_aliases VALUES (1,'postmaster@example.com',
        'bamm.bamm@example.com');
mysql> INSERT INTO virtual_aliases VALUES (2,'abuse@example.com',
        'bamm.bamm@example.com');
```

Создание пользователя MySQL для Postfix

Наша последняя задача при настройке базы данных состоит в создании пользователя MySQL для обращения к таблицам. Вы должны ограничить права пользователя, чтобы сервер Postfix не мог изменить данные.

Следующая команда создает нового пользователя с именем postfix, который может подключаться с хоста localhost:

```
mysql> CONNECT mysql;
mysql> INSERT INTO user VALUES
       ('localhost','postfix','','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y'
        ,'Y','Y','Y');
mysql> UPDATE mysql.user SET password=PASSWORD("Yanggt!") WHERE
        user='postfix' AND host='localhost';
mysql> FLUSH PRIVILEGES;
```

Этой учетной записи следует разрешить доступ только на чтение (SELECT). Сервер Postfix не должен иметь возможности изменять или создавать таблицы. Используем команду GRANT:

```
mysql> GRANT USAGE ON *.* TO 'postfix'@'localhost' IDENTIFIED BY
        PASSWORD '2fc879714f7d3e72';
mysql> GRANT SELECT ON mail.virtual_aliases TO 'postfix'@'localhost';
mysql> GRANT SELECT ON mail.virtual_users TO 'postfix'@'localhost';
mysql> GRANT SELECT ON mail.virtual_mailbox_domains TO
        'postfix'@'localhost';
```

Настройка Postfix для использования базы данных

При настройке системы с SQL-запросами в Postfix вам нужно задать в специальном файле ряд параметров. Сервер Postfix подставит эти параметры в последовательность команд SQL, которая завершается командой SELECT:

`user`

Имя пользователя, который подключается к базе данных.

`password`

Пароль пользователя базы данных. Должен быть предоставлен в виде простого текста.

`hosts`

Список из одного или нескольких полностью определенных доменных имен или IP-адресов серверов SQL. Если серверу Postfix не удается подключиться к первому хосту в списке, он пробует установить соединение с другими хостами (в произвольном порядке). Если ни один сервер не доступен, Postfix откладывает работу до тех пор, пока какой-то сервер не станет доступен.

Если вы используете в качестве сервера `localhost`, Postfix автоматически использует сокет домена UNIX вместо TCP/UDP-соединения.

`dbname`

Имя базы данных, к которой подключается пользователь.

`table`

Имя таблицы, содержащей данные виртуального домена.

`select_field`

Поле, содержащее результат запроса (например, электронный адрес пользователя).

`where_field`

Поле, которому ищется соответствие при запросе базы данных (например, псевдоним электронного адреса).

`additional_conditions`

Дополнительные условия запроса. Например, вы можете запросить только активные в настоящий момент пользовательские учетные записи. Это необязательный параметр.

В табл. 14.1 показано соответствие полей индексированной карты параметрам базы данных. Вы можете использовать эту таблицу при создании команды `SELECT`, если не уверены в том, какие поля таблицы следует указать.

Таблица 14.1. Соответствие полей индексированных карт параметрам базы данных

Тип карты	Левая часть	Правая часть	Условия
Индексированная карта	Левый столбец	Правый столбец	–
Таблица базы данных SQL	`where_field`	`select_field`	`additional_conditions`

Защита SQL-конфигурации в Postfix

В настоящее время Postfix поддерживает два способа настройки команд `SELECT` для MySQL (и PostgreSQL).

Первый требует указания имени пользователя Postfix на сервере SQL и его пароля в файле `main.cf`. Этот способ не очень надежен, поэтому не будет рассмотрен в нашей книге (файл `main.cf` обычно является общедоступным, так что любой пользователь вашей системы сможет получить эти данные). Если вы настаиваете на использовании такого способа, обратитесь к файлу `MYSQL_README` в каталоге `readme` за дополнительной информацией, но имейте в виду, что новые версии Postfix не будут поддерживать эту возможность.

Второй способ предпочтительнее, т. к. он гораздо эффективнее обеспечивает безопасность. Команда `SELECT` (включая имя пользователя и пароль) хранится в отдельных файлах вне `main.cf`. Более того, вы помещаете их в каталог, доступный только пользователям `Postfix` и `root` (расположение файлов вы указываете в `main.cf`).

Для создания файловой структуры сначала создайте каталог, например `/etc/postfix/sql`, и укажите соответствующие права:

```
# mkdir /etc/postfix/sql
# chown postfix /etc/postfix/sql
# chgrp root /etc/postfix/sql
# chmod 500 /etc/postfix/sql
```

Теперь мы готовы добавлять туда файлы, которые будут скрыты от (большинства) любопытных глаз.

Создание запроса для доменов получателей

Первый файл, который вам следует добавить, будет определять параметры запроса, извлекающего домены, для которых сервер Postfix является местом конечного назначения. Добавляем следующую конфигурацию в файл, который назовем, например, /etc/postfix/sql/virtual_mailbox_domains.cf:

```
user = postfix
password = Yanggt!
dbname = mail
table = virtual_mailbox_domains
select_field = domain
where_field = domain
hosts = localhost
```

Как уже говорилось ранее, эти параметры соответствуют значениям в SQL-команде SELECT. Для предыдущего файла эта команда будет выглядеть следующим образом (domainpart — это доменная часть электронного адреса получателя входящего сообщения):

```
mysql> SELECT domain FROM virtual_mailbox_domains WHERE domain = 'domainpart'
```

Может показаться несколько странным, что мы читаем домен из таблицы, хотя уже знаем его, но цель этого запроса в том, чтобы узнать, присутствуют ли на самом деле в базе данных строки с соответствующей доменной частью. Если таких строк нет, то запрос ничего не возвращает, и сервер Postfix знает, что он не является местом конечного назначения для данного домена.

Примечание

Вам не нужно вводить ни одну из этих команд SELECT (Postfix создаст их автоматически при обращении к базе данных), но их полезно знать для отслеживания проблем базы данных при работе в командной строке MySQL.

Теперь нужно сообщить серверу Postfix (в файле main.cf), что он должен использовать MySQL, и указать ему, где найти параметры для virtual_mailbox_domains. Спецификация очень похожа на обычную индексированную карту, только ключевое слово hash заменено на mysql:

```
virtual_mailbox_domains = mysql:/etc/postfix/sql/virtual_mailbox_domains.cf
```

Создание запросов для идентификатора пользователя и группы

Теперь нам следует добавить параметры для запросов UID и GID (если помните, они определяют владельца файла виртуального почтового ящика). Добавим следующие строки в файл /etc/postfix/sql/virtual_uid_maps.cf:

```
user = postfix
password = Yanggt!
```

```
dbname = mail
table = virtual_users
select_field = virtual_uid
where_field = email
hosts = localhost
```

Соответствующая данному файлу команда SELECT выглядит так (recipient – это адрес получателя входящего сообщения):

```
mysql> SELECT virtual_uid FROM virtual_users WHERE email = 'recipient'
```

Используем параметр virtual_uid_maps в файле main.cf, чтобы указать серверу Postfix, где он может найти SQL-запрос для поиска UID:

```
virtual_uid_maps = mysql:/etc/postfix/sql/virtual_uid_maps.cf
```

Создание запроса SQL для GID аналогично только что описанной процедуре для UID. Используем имя файла /etc/postfix/sql/virtual_gid_maps.cf для создания SQL-команды SELECT:

```
user = postfix
password = Yanggt!
dbname = mail
table = virtual_users
select_field = virtual_gid
where_field = email
hosts = localhost
```

Соответствующая команда SELECT для данного файла будет такой:

```
mysql> SELECT virtual_gid FROM virtual_users WHERE email = 'recipient'
```

Затем сообщаем серверу Postfix, где искать GID. Задаем параметр virtual_gid_maps в файле main.cf как указывающий на файл SQL-запроса:

```
virtual_gid_maps = mysql:/etc/postfix/sql/virtual_gid_maps.cf
```

Создание запроса для получателей

Самым важным, наверное, является запрос, извлекающий имя почтового ящика получателя для заданного адреса получателя. Добавляем следующие параметры запроса в файл /etc/postfix/sql/virtual_mailbox_recipients.cf:

```
user = postfix
password = Yanggt!
dbname = mail
table = virtual_users
select_field = virtual_mailbox
where_field = email
additional_conditions = and active = '1'
hosts = localhost
```

Обратите внимание на наличие параметра additional_conditions. Такому файлу соответствует следующая команда SELECT:

```
mysql> SELECT virtual_mailbox FROM virtual_users WHERE email = 'recipient'
                                                  AND active = '1'
```

Параметр `virtual_mailbox_maps` сообщает серверу Postfix, где искать получателей виртуальных почтовых ящиков и их ящики. Добавляем следующую строку в файл `main.cf`:

```
virtual_mailbox_maps = mysql:/etc/postfix/sql/virtual_mailbox_recipients.cf
```

Создание запроса для псевдонимов

Наконец, пришло время указать параметры для запроса виртуальных псевдонимов. Помещаем следующие строки в файл `/etc/postfix/sql/virtual_alias_maps.cf`:

```
user = postfix
password = Yanggt!
dbname = mail
table = virtual_aliases
select_field = virtual_user_email
where_field = alias
hosts = localhost
```

Этому файлу соответствует такая команда `SELECT`:

```
mysql> SELECT virtual_user_email FROM virtual_aliases WHERE
       alias = 'recipient'
```

В завершение указываем серверу Postfix, где искать конфигурационный файл запроса для псевдонимов, посредством параметра `virtual_alias_maps` в файле `main.cf`:

```
virtual_alias_maps = mysql:/etc/postfix/sql/virtual_alias_maps.cf
```

Перезагружаем конфигурацию для вступления в силу всех изменений `main.cf` и приступаем к тестированию.

Тестирование управляемых базой данных доменов виртуальных почтовых ящиков

Если вы не знаете, что является источником вашей проблемы – Postfix или MySQL, то ее отслеживание может стать весьма утомительной задачей. Поэтому следует проверять MySQL и Postfix по отдельности. Если MySQL-проверки проходят успешно, это означает, что ваша проблема вызвана конфигурацией Postfix.

Тестирование MySQL

Первое, что необходимо проверить, – разрешено ли пользователю, имя и пароль которого вы указали в конфигурационных файлах запроса, обращаться к MySQL и делать запросы.

Затем мы пытаемся подключиться к базе данных, которая хранит данные нашего домена виртуальных почтовых ящиков. Обе проверки проводятся в следующей строке:

```
# mysql -u postfix -p -h localhost -A
```

При успешном входе в систему вы увидите такое сообщение:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 144 to server version: 3.23.58
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

Используйте команду `CONNECT` для доступа к базе данных `mail`. Вы должны получить подтверждение такого вида:

```
mysql> CONNECT mail;
Connection id:    145
Current database: mail
mysql>
```

Если не удается подключиться к серверу, проверьте имя и пароль пользователя, которые вы указали MySQL ранее в разделе «Создание пользователя MySQL для Postfix». Если же проблемы с подключением к базе данных `mail`, проверьте команды GRANT (см. тот же раздел).

Запрос доменов получателей

Пришло время выполнить команду `SELECT`, которая будет проверять, может ли ваш MySQL-пользователь Postfix просматривать домены виртуальных почтовых ящиков в таблице `virtual_mailbox_domains`:

```
mysql> SELECT domain FROM virtual_mailbox_domains WHERE
       domain = 'example.com';
```

В случае успешного выполнения вы должны увидеть одну строку вывода, содержащую входные данные запроса (из раздела «Создание запроса для доменов получателей» вы должны помнить, что это корректное поведение запроса):

```
+-------------+
| domain      |
+-------------+
| example.com |
+-------------+
1 row in set (0.00 sec)
```

Если совпадение не найдено, проверьте, присутствует ли домен в таблице. Это легко – для выбора всех строк таблицы опускаем инструкцию `WHERE`:

```
mysql> SELECT domain FROM virtual_mailbox_domains;
```

Запрос UID и GID виртуальных почтовых ящиков

Затем проверяем, может ли пользователь MySQL извлечь известный адрес получателя. Пытаемся извлечь поле `virtual_uid` из записи известно-

го адреса получателя в таблице virtual_users, как в следующем успешном примере, который показывает соответствие адреса bamm.bamm@example.com виртуальному почтовому ящику, имеющему UID, равный 1001:

```
mysql> SELECT virtual_uid FROM virtual_users WHERE email =
  'bamm.bamm@example.com';
+-------------+
| virtual_uid |
+-------------+
|        1001 |
+-------------+
1 row in set (0.00 sec)
```

Потом делаем то же самое для GID:

```
mysql> SELECT virtual_gid FROM virtual_users WHERE email =
  'bamm.bamm@example.com';
+-------------+
| virtual_gid |
+-------------+
|        1001 |
+-------------+
1 row in set (0.00 sec)
```

Запрос почтовых ящиков получателей

Проверяем, может ли MySQL-пользователь Postfix найти почтовый ящик для указанного получателя (вспоминаем, что речь идет о поле virtual_mailbox таблицы virtual_users). В следующем примере адрес bamm.bamm@example.com соответствует почтовому ящику example.com/bammbamm/ типа Maildir:

```
mysql> SELECT virtual_mailbox FROM virtual_users WHERE email =
  'bamm.bamm@example.com';
+-----------------------+
| virtual_mailbox       |
+-----------------------+
| example.com/bammbamm/ |
+-----------------------+
1 row in set (0.00 sec)
```

Запрос псевдонимов

Последнее, на что мы проверяем базу данных, – это псевдонимы. Для известного псевдонима пытаемся извлечь адрес получателя (поле virtual_user_email таблицы virtual_aliases), например:

```
mysql> SELECT virtual_user_email FROM virtual_aliases WHERE alias =
  'postmaster@example.com';
+-----------------------+
| virtual_user_email    |
+-----------------------+
| bamm.bamm@example.com |
+-----------------------+
1 row in set (0.00 sec)
```

Тестирование Postfix

Протестировать просмотр сервером Postfix базы данных MySQL можно без отправки каких-либо тестовых сообщений. Команда `postmap` может выполнять любые виды запросов, в том числе и в таблице MySQL. Приведем общий формат команды `postmap`, выполняющей MySQL-запрос:

```
# postmap -q "value" mysql:path-to-parameter-file
```

Например, такая команда ищет в базе данных MySQL известный домен виртуальных почтовых ящиков:

```
# postmap -q "example.com" mysql:/etc/postfix/sql/virtual_mailbox_domains.cf
```

В случае успеха в командной строке будут выведены соответствующие данные:

```
example.com
```

Если результат не выводится, проверьте, присутствует ли имя виртуального домена в вашей таблице (см. раздел «Запрос доменов получателей» ранее в этой главе). Если имя домена присутствует в таблице, проверьте сведения об имени пользователя и пароле в файле `virtual_mailbox_domains.cf`.

Запрос UID и GID

Продолжаем тестирование и запрашиваем в MySQL данные об UID и GID получателя известного почтового ящика:

```
# postmap -q "bamm.bamm@example.com" mysql:/etc/postfix/sql/
virtual_uid_maps.cf
1001
# postmap -q "bamm.bamm@example.com" mysql:/etc/postfix/sql/
virtual_gid_maps.cf
1001
```

Запрос получателей

Проверяем, может ли Postfix искать известных получателей. Команда `postmap`, соответствующая команде `SELECT` из раздела «Запрос почтовых ящиков получателей», выглядит так:

```
# postmap -q "bamm.bamm@example.com" mysql:/etc/postfix/sql/
virtual_mailbox_recipients.cf
example.com/bammbamm/
```

Запрос псевдонимов

Наконец, последняя проверка: просим сервер Postfix найти в базе данных MySQL адрес виртуального пользователя для известного псевдонима. Успешный запрос будет таким:

```
# postmap -q "postmaster@example.com" mysql:/etc/postfix/sql/
virtual_alias_maps.cf
bamm.bamm@example.com
```

15

Введение в SMTP-аутентификацию

SMTP-аутентификация – это способ идентификации клиентов независимо от их IP-адресов; он позволяет серверу пересылать сообщения от почтовых клиентов, чьи IP-адреса не входят в список доверенных. В этой главе изложены основы SMTP-аутентификации (SMTP AUTH). Вы не только узнаете об этом способе аутентификации и его преимуществах перед другими подходами, но и научитесь устанавливать и настраивать пакет Cyrus SASL, необходимый для поддержки SMTP-аутентификации в Postfix.

Архитектура и конфигурация Cyrus SASL

Первые серверы SMTP пересылали почту от любого клиента любому получателю. С возникновением проблемы спама агенты передачи сообщений обрели возможность принимать запросы на ретрансляцию только от определенных клиентов. Разработчики агентов передачи сообщений приняли решение идентифицировать такие клиенты по их IP-адресам, а администраторам оставалось только настроить свои системы так, чтобы они отвергали клиенты, не попавшие в список доверенных (рис. 15.1).

Сегодня попытки злоупотребления почтовыми рассылками остаются насущной проблемой, заставляющей администраторов тратить немало времени на усиление защиты своих серверов и введение дополнительных ограничений (см. главу 8). К тому же привязка разрешений на пересылку к IP-адресу становится все более трудоемкой по мере роста размеров и усложнения структуры современных сетей, кроме того, она совершенно не подходит для мобильных пользователей.

Мобильные пользователи (в соответствии с определением в RFC 2977) нуждаются в доступе к ресурсам своего домена из любой точки Интернета. К сожалению, такие пользователи почти никогда не используют

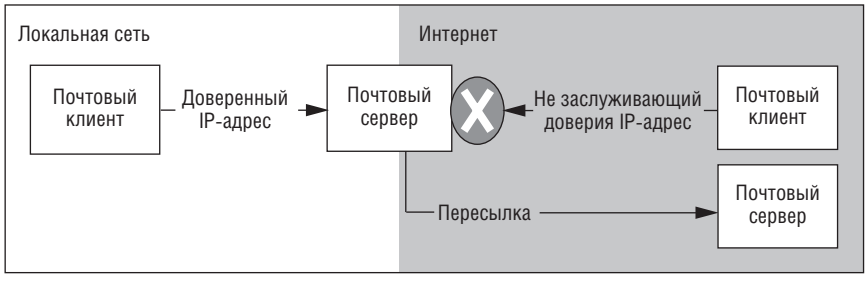

Рис. 15.1. *Современные почтовые серверы отказывают в пересылке IP-адресам, не вошедшим в список доверенных*

один и тот же IP-адрес, более того, ни они сами, ни администратор почтового сервера не знают заранее, какими будут их адреса, что делает бессмысленным применение правил, основанных на статических IP-адресах.

Есть несколько способов предоставления мобильным пользователям разрешений на пересылку:

- SMTP-after-POP и SMTP-after-IMAP.
- SMTP-аутентификация.
- Пересылка на основании сертификата.
- Использование виртуальных частных сетей (VPN).

Методы SMTP-after-POP и SMTP-after-IMAP (рис. 15.2) делегируют решение вопроса идентификации серверам POP и IMAP.

Вот основные этапы работы методов SMTP-after-POP и SMTP-after-IMAP:

1. Почтовый клиент проходит аутентификацию на POP- или IMAP-сервере.
2. После успешной аутентификации POP- или IMAP-сервер записывает IP-адрес клиента в базу данных, используемую совместно с поч-

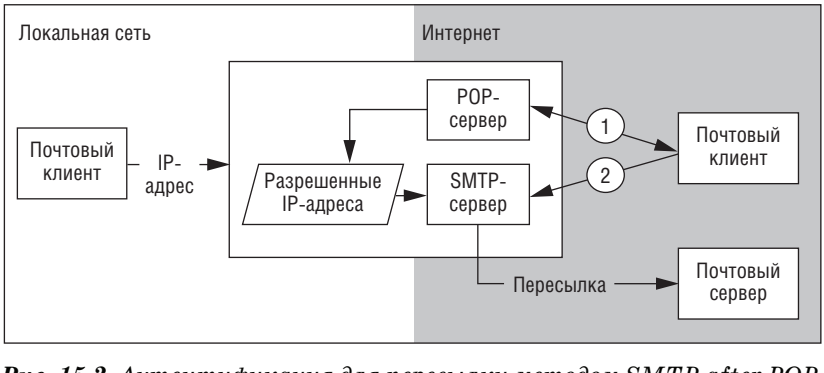

Рис. 15.2. *Аутентификация для пересылки методом SMTP-after-POP*

товым сервером. Этот адрес хранится в базе данных в течение ограниченного времени.

3. Почтовый клиент делает попытку переслать сообщение через SMTP-сервер.
4. SMTP-сервер ищет IP-адрес клиента в базе данных. Если адрес найден, сервер разрешает пересылку.

SMTP-аутентификация решает проблему «на корню» (рис. 15.3).

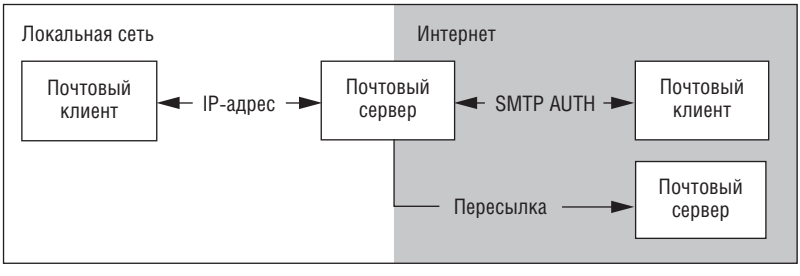

Рис. 15.3. *Аутентификация для пересылки методом SMTP AUTH*

Основные этапы работы метода SMTP AUTH:

1. SMTP-сервер предлагает почтовому клиенту пройти аутентификацию SMTP AUTH.
2. Клиент передает свои верительные данные серверу.
3. Сервер проверяет данные клиента и, если они верны, разрешает пересылку.

Пересылка на основании сертификата, рассматриваемая в главе 18, подразумевает передачу и проверку сертификата клиента TLS-соединения (рис. 15.4).

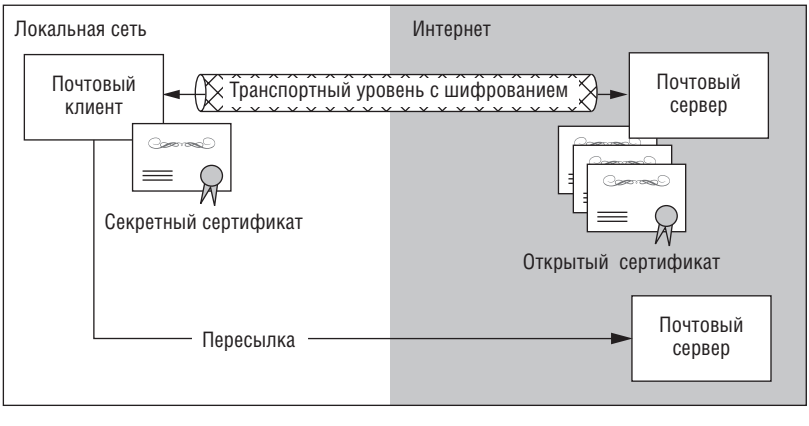

Рис. 15.4. *Аутентификация для пересылки с помощью клиентского сертификата*

Основные этапы пересылки на основании сертификата:

1. SMTP-сервер предлагает почтовому клиенту установить TLS-соединение.
2. Клиент посылает сертификат серверу.
3. Сервер проверяет сертификат и, если сертификат входит в список признаваемых, разрешает клиенту пересылку.

Виртуальные частные сети (VPN) предоставляют клиенту доступ к почтовому серверу путем создания защищенной виртуальной сети поверх обычной сети Интернет. В VPN администраторы контролируют адресацию, поэтому можно использовать пересылку на основании IP-адреса. Вследствие того, что настройка виртуальной сети никак не связана с настройкой SMTP-сервера, она не рассматривается в этой книге. Вот основные действия при использовании VPN:

1. Компьютер почтового клиента устанавливает соединение с частной сетью (VPN).
2. SMTP-сервер разрешает клиенту пересылку на основании его IP-адреса в виртуальной частной сети.

Какой подход лучше?

Пересылка на основании сертификата с использованием TLS – замечательный способ, обеспечивающий высокий уровень безопасности, но многие клиенты еще не поддерживают его. Кроме того, управление сертификатами означает немалые накладные расходы для клиентов и сервера. Так что для перехода на эту технологию компании или интернет-провайдеру требуются значительные усилия. Если вы пока не можете использовать для пересылки сертификаты, то ваш выбор сводится к SMTP-after-POP, SMTP AUTH и VPN.

С точки зрения системного архитектора метод SMTP-after-POP далек от идеала, т. к. решение проблемы реализуется не в рамках того сервера и протокола, где она возникает. Вместо этого другой сервер (POP или IMAP), который является как минимум настолько же сложным, как и SMTP-сервер, предлагает обходной путь. Это только усложняет дело, ведь два сервера должны взаимодействовать, а поскольку они почти наверняка разработаны разными людьми, весьма высок риск несовместимости, особенно при выпуске новых версий.

Но архитектура метода SMTP-after-POP не единственный его недостаток. На самом деле этот метод не обеспечивает должной безопасности, т. к. принимает решения на основе проверки IP-адреса. Но подделка IP-адреса не представляет особого труда[1] – атакующий может узнать

[1] Однако для приема пакетов, отправленных на «сымитированный» IP-адрес, злоумышленнику понадобится подменить таблицу маршрутизации или ARP-таблицу хотя бы на одном не подвластном ему хосте; опасные атаки с подменой IP-адреса практикуются в качестве DoS-атак, (*см. продолжение*)

IP-адрес почтового клиента, которому только что было выдано разрешение на пересылку на определенный период времени, и имитировать этот адрес до окончания данного периода. Такое мошенничество невозможно, если клиент должен проходить аутентификацию при отправке каждого нового сообщения.

Систему на основе виртуальной частной сети очень легко настроить, если у вас уже есть такая сеть, но ее создание только для почтового сервера требует слишком больших усилий. Кроме того, виртуальная частная сеть требует постоянной поддержки, т. к. каждому новому мобильному пользователю необходимо соответствующее программное обеспечение.[1]

Если вам нужно простое, независимое и надежное решение, то ваш выбор – SMTP AUTH.

SASL – простой протокол аутентификации и безопасности

Postfix реализует SMTP-аутентификацию при помощи протокола SASL (Simple Authentication and Security Layer – простой протокол аутентификации и безопасности). Описанный в RFC 2222 протокол SASL определяет структуру процесса аутентификации, и понимание его работы необходимо для понимания SMTP-аутентификации в целом. Существует несколько реализаций SASL; Postfix использует библиотеки Cyrus-SASL, которые происходят от первоначальной реализации SASL в проекте Cyrus.

Примечание

Проект Cyrus был осуществлен в университете Карнеги – Меллона с целью построения новой почтовой системы для университетского городка (дополнительную информацию можно найти по адресу *http://asg.web.cmu.edu/cyrus*).

Протокол SASL включает в себя три уровня, которые вам необходимо настроить. Эти три уровня изображены на рис. 15.5: интерфейс аутентификации, механизм и метод.

В приложении, использующем Cyrus SASL, процесс аутентификации состоит из следующих этапов:

(*продолжение*) а не для доступа к чужим данным; см. например RFC 2827 (на русском) по адресу *http://www.protocols.ru/files/RFC/rfc2827.pdf*. – *Примеч. науч. ред.*

[1] Справедливости ради стоит сказать, что все современные ОС в стандартной конфигурации имеют те или иные средства организации VPN. – *Примеч. науч. ред.*

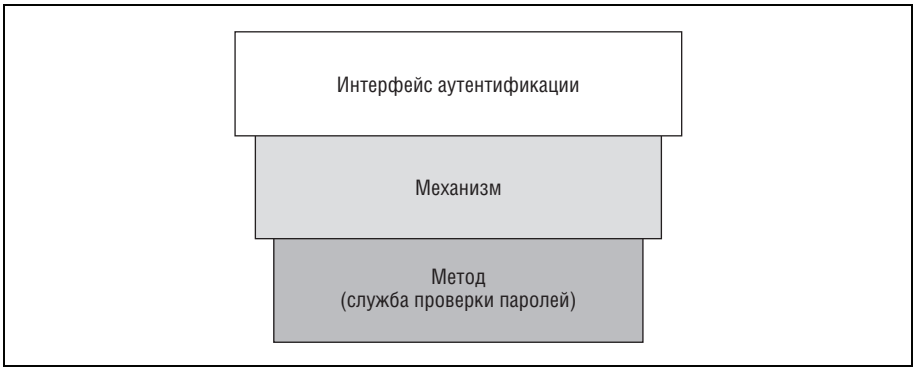

***Рис. 15.5.** Уровни SASL*

1. Поддерживающее SASL приложение (например, Postfix-демон `smtpd`) прослушивает сетевые соединения.
2. Клиент подключается и инициирует аутентификацию:
   a. Клиент выбирает механизм SMTP AUTH.
   b. Клиент готовится к передаче своих верительных данных в соответствии с требованиями выбранного механизма.
   c. Клиент сообщает серверу, какой механизм он выбрал.
   d. Клиент передает свои верительные данные.
3. Приложение сохраняет информацию о выбранном механизме и верительные данные.
4. Приложение передает информацию драйверу механизма, который передает ее службе проверки паролей.
5. Служба проверки паролей обращается к хранилищу аутентификационных данных, например `/etc/shadow`, которое пытается установить соответствие верительных данных клиента одной из своих записей.
6. Служба проверки паролей передает результат от хранилища аутентификационных данных демону `smtpd`.
7. На основе полученного результата демон `smtpd` предпринимает какое-то действие. Например, он может разрешить аутентифицированным пользователям пересылать почту.

В последующих разделах мы рассмотрим протокол Cyrus SASL более подробно. Вы получите всю необходимую информацию об интерфейсе аутентификации SASL, методах, механизмах и хранилищах данных аутентификации и о том, как подготовить Cyrus SASL и Postfix для использования SMTP AUTH на стороне сервера.

Интерфейс аутентификации

Интерфейс аутентификации предназначен для того, чтобы сообщить клиенту о доступности аутентификации и о том, какие механизмы могут быть использованы. Аутентификации могут требовать многие службы (например, LDAP и SMTP), при этом они используют различные протоколы «клиент-сервер». Поэтому SASL не имеет собственного интерфейса аутентификации, а предоставляет специальной службе и ее протоколу донесение информации об аутентификации до сведения клиента.

Что касается электронной почты, местом встречи клиента и сервера является SMTP-диалог. Протокол ESMTP интегрирует аутентификацию в диалог, добавляя ее в список функциональностей почтового сервера. Вы можете узнать, предоставляет ли почтовый сервер SMTP AUTH, подключившись к нему и выдав команду-приветствие EHLO, например:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 20480000
250-VRFY
250-ETRN
250-AUTH PLAIN LOGIN
250-AUTH=PLAIN LOGIN
250-XVERP
250 8BITMIME
QUIT
221 Bye
```

Выделенные жирным курсивом строки указывают на поддержку SMTP-аутентификации с механизмами аутентификации PLAIN и LOGIN. Вторая строка (со знаком равенства) предназначена для тех почтовых клиентов, которые не следуют итоговой спецификации SASL.

Предупреждение

Интерфейс приложения настраивается внутри приложения, обеспечивающего SMTP AUTH. В Postfix это демон smtpd; о его настройке вы узнаете в главе 16.

Механизмы SMTP AUTH

Механизмы SMTP AUTH, такие как PLAIN и LOGIN, определяют методику проверки в процессе аутентификации. О том, какие механизмы готов предоставить сервер, он сообщает через интерфейс приложения, в данном случае – посредством SMTP-диалога. При инициировании аутентификации клиент выбирает механизм, сообщает о своем выборе серверу и передает свои верительные данные.

Cyrus SASL предоставляет множество разнообразных механизмов SASL, которые отличаются способом передачи верительных данных и уровнем обеспечиваемой безопасности. Некоторые из них нестандартны и созданы специально для конкретных клиентов. Вам не нужно использовать все доступные вам механизмы – можно настроить сервер Postfix так, чтобы он предоставлял лишь ограниченный набор механизмов Cyrus SASL.

Совет

Практика показывает, что в среде, где необходимо поддерживать клиенты операционных систем Windows, Mac OS и UNIX, лучше всего использовать механизмы PLAIN, LOGIN и CRAM-MD5.

Рассмотрим эти механизмы:

ANONYMOUS

Механизм ANONYMOUS (определен в RFC 2245) был создан для того, чтобы разрешить анонимный доступ к почтовым услугам. Для использования этого механизма почтовому клиенту нужно лишь отправить серверу любую строку, и сервер разрешит клиенту пересылку.

Предупреждение

Не используйте механизм ANONYMOUS для Postfix, если не хотите, чтобы ваш почтовый сервер превратился в открытый ретранслятор. Спамеры умеют злоупотреблять этим механизмом.

CRAM-MD5, DIGEST-MD5

Cyrus SASL поддерживает два механизма с общим секретным ключом: CRAM-MD5 и его преемник, DIGEST-MD5. В основе этих методов лежит то, что клиент и сервер объединяет общий секретный ключ, обычно это пароль. Сервер формирует запрос на основе секретных данных, а клиент отвечает, доказывая, что секретный ключ ему известен. Это гораздо более безопасно, чем просто отправлять незашифрованный пароль по сети, но серверу все еще необходимо хранить этот секрет.

EXTERNAL

Механизм EXTERNAL позволяет привязать верительные данные TLS/SSL к SASL. В частности, механизм EXTERNAL позволяет извлечь некоторую разновидность имени пользователя из сертификата, используемого в процессе TLS/SSL-переговоров.

GSSAPI, KERBEROS_V4

Cyrus SASL включает в себя два механизма, использующих систему аутентификации Kerberos: KERBEROS_V4, который должен иметь возможность использовать любую реализацию Kerberos v4, и GSSAPI (проверенный для MIT Kerberos v5 и Heimdal Kerbe-

ros v5). Эти механизмы используют инфраструктуру Kerberos, поэтому у них нет базы данных паролей.

Примечание

К сожалению, у механизма KERBEROS_V4 есть некоторые проблемы с безопасностью. Используйте вместо него GSSAPI.

NTLM

Механизм NTLM – это нестандартный недокументированный механизм, разработанный Microsoft. Он включен в дистрибутив SASL для тех серверов, которые должны взаимодействовать с клиентами (например, Outlook) и серверами (такими, как Exchange) от Microsoft.

OTP

Механизм OTP (One-Time Password – одноразовый пароль) похож на CRAM-MD5 и DIGEST-MD5 тем, что он использует общий секретный ключ и обмен запросом/ответом. Однако OTP считается более безопасным, чем другие механизмы с общим секретным ключом, т. к. ключ используется для формирования и хранения последовательности одноразовых паролей, что предотвращает атаки воспроизведения, благодаря чему не нужно хранить секретные ключи на сервере.

PLAIN, LOGIN

Верительные данные, отправленные с использованием механизма PLAIN, передаются открытым текстом в кодировке base64. Это простой механизм, который реализует большинство почтовых клиентов, но строки в кодировке base64 легко поддаются декодированию. Механизм LOGIN – это тот же PLAIN, только он используется для почтовых клиентов, которые не соответствуют RFC (как Outlook Express).

Предупреждение

Передача верительных данных по сети открытым текстом чревата проблемами с безопасностью. Каждый, кто запустит анализатор пакетов, такой как snort или tcpdump, может прочитать секретные данные. Проблему можно обойти, используя TLS в сочетании с SMTP-аутентификацией без шифрования. В частности, вы можете указать серверу Postfix, что следует предоставлять механизмы открытого текста только после открытия TLS (см. главу 18).

SRP

Механизм SRP сочетает надежность аутентификации на основе использования общего секретного ключа с одноразовостью OTP. Он основывается на шифровании открытых ключей и использует в процессе аутентификации пароли, а не сертификаты.

Методы аутентификации (службы проверки паролей)

Методы аутентификации (называемые также службами проверки паролей) управляют механизмами, заботясь о взаимодействии между приложением, которое обеспечивает SMTP-аутентификацию, и хранилищем верительных данных.

Cyrus SASL предоставляет две службы проверки паролей: saslauthd и auxprop. Эти службы отличаются поддерживаемыми механизмами и хранилищами, к которым они могут подключаться. Приложение, предоставляющее аутентификацию через свой интерфейс, должно выбрать, какой службой проверки паролей воспользоваться (о настройке мы поговорим позже в разделе «Установка и настройка Cyrus SASL»). Приведем краткое описание служб:

saslauthd

saslauthd – это автономный демон, который может быть запущен с привилегиями суперпользователя. Он может подключаться к различным видам хранилищ, но главным образом к тем, которые требуют привилегий суперпользователя (например, `/etc/shadow`). Использование saslauthd ограничивается механизмами открытого текста (PLAIN и LOGIN).

auxprop

auxprop – это общая служба проверки паролей для ряда вспомогательных плагинов. Каждый плагин специализируется на определенном типе хранилища аутентификационных данных, такого как серверы SQL и sasldb2.

Предупреждение

Плагины auxprop подключаются к хранилищу с привилегиями приложения, обеспечивающего аутентификацию. В Postfix это демон `smtpd`, который работает с минимальным набором привилегий.

Версии Cyrus SASL до 2.x разрешали доступ к `/etc/shadow` посредством auxprop, что требовало расширения привилегий демона, обеспечивающего SMTP AUTH, и разрушало архитектуру безопасности Postfix. Настоятельно рекомендуем не использовать хранилище аутентификационных данных, которое заставляет расширять привилегии приложения.

Хранилища аутентификационных данных

Наконец, Cyrus SASL для проверки верительных данных, полученных от клиента, требует наличия одного или нескольких хранилищ аутентификационных данных. Служба проверки паролей проверяет, соответствуют ли верительные данные клиента тем, что находятся в хранилище. Перечислим хранилища аутентификационных данных, которые включены в официальный список поддерживаемых Cyrus SASL:

imap

Сервер IMAP может проверять верительные данные.

kerberos

Cyrus SASL может проверять мандаты Kerberos.

ldap

Cyrus SASL может читать верительные данные с сервера OpenLDAP.

pam

Cyrus SASL может читать данные из любых модулей, доступных ему посредством PAM (Pluggable Authentication Modules – подключаемые модули аутентификации).

passwd/shadow

Cyrus SASL может читать данные из баз данных системного пользователя (`/etc/passwd` и, возможно, `/etc/shadow`).

sasldb2

Cyrus SASL имеет собственную базу данных с именем `sasldb2`. Эта база данных необходима для сервера Cyrus IMAP, но он вам не нужен при использовании базы данных для SMTP-аутентификации.

sql

Cyrus SASL может обращаться к пользовательским данным на серверах SQL. В настоящее время поддерживаются серверы MySQL и PostgreSQL.

Планирование SMTP-аутентификации на стороне сервера

Если вы хотите, чтобы ваш почтовый сервер поддерживал SMTP-аутентификацию, вам нужно просто выполнить следующие действия:

1. Определить клиенты, которые будут использовать SMTP AUTH, и механизмы, поддерживаемые этими клиентами.
2. Определить хранилище аутентификационных данных, которое вы хотите использовать, и соответствующую службу проверки паролей.

Определение клиентов и поддерживаемых механизмов

Один из принципов обеспечения компьютерной безопасности говорит о том, что невозможно атаковать службу, которая не существует. Поэтому следует настроить в Postfix только те механизмы, которые необходимы вашим пользователям. В небольшой сети, где использование почтовых клиентов находится под вашим полным контролем, можно ограничить поддерживаемые механизмы. В табл. 15.1 представлена упрощенная версия справочника клиентов SASL (*http://www.melnikov.ca/mel/devel/SASL_ClientRef.html*) Алексея Мельникова, в которой перечислены механизмы, поддерживаемые различными почтовыми клиентами. Таблица предлагает обширные сведения о возможностях почтовых клиентов в отношении аутентификации POP, IMAP, ACAP и LDAP.

Таблица 15.1. Механизмы SMTP AUTH и поддерживающие их почтовые клиенты

Клиент	ANONYMOUS	CRAM-MD5	DIGEST-MD5	EXTERNAL	GSSAPI	Kerberos 4	LOGIN	NTLM	PLAIN	SCRAM-MD5	SKEY
AKmail	нет	да	нет	нет	нет	нет	нет	нет	нет	нет	нет
Bat! (SecureBat!)	нет	да	да	нет	нет	нет	да	нет	да	нет	нет
Control Data's IMAPSP	нет	да	нет	нет	нет	нет	нет	нет	да	нет	да
Eudora Pro 4.3 и выше	нет	да	нет	нет	нет	нет	да	нет	да	нет	нет
fetchmail 5.0.3	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет
Forte Agent	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет
Gnus	нет	да	нет	нет	нет	нет	да	нет	нет	нет	нет
JavaMail	нет	нет	нет	нет	нет	нет	да	нет	да	нет	нет
MH UCI 6.8.3	?	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет
Mozilla 1.0	нет	да	нет	да	нет	нет	нет	нет	да	нет	нет
Mulberry v3	нет	да	да	нет	да	нет	да	да	да	нет	нет
mutt 1.2.5i	n/a[a]	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Netscape Messenger 4.51+	нет	нет	нет	да	нет	нет	нет	нет	да	нет	нет
nmh 1.0.4	–[b]	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет
Novell Evolution	нет	да	да	нет	нет	да	нет	да	да	нет	нет
Orangsoft Winbiff > 2.30	нет	да	нет	нет	нет	нет	да	нет	да	нет	нет
Outlook Express > 4.0[c]	нет	нет	нет	нет	нет	нет	да	да	нет	нет	нет
Outlook > 98[c]	нет	нет	нет	нет	нет	нет	да	да	нет	нет	нет
Paladin	нет	да	нет	нет	нет	нет	нет	нет	нет	нет	нет
PalmOS Eudora	нет	да	нет	нет	нет	нет	нет	нет	нет	нет	нет
Pegasus Mail 3.12	нет	да	нет	нет	нет	нет	да	нет	нет	нет	нет
Pine 4.33 и выше	нет	да	нет	нет	да	да	да	нет	да	нет	нет
PMMail	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет
Sylpheed	нет	да	да	нет	нет	нет	да	нет	нет	нет	нет
Wanderlust	да	да	да	нет	нет	да	да	нет	нет	да	нет

[a] Не поддерживает SMTP напрямую, полагается на локальный агент передачи сообщений.

[b] См. библиотеку CMU SASL.

[c] Поддерживает черновой вариант №10 спецификации SMTP AUTH (например, «AUTH=», а не «AUTH »).

Однако если вам необходимо поддерживать как можно больше разных клиентов, то следует предоставлять хотя бы механизмы PLAIN, LOGIN и CRAM-MD5, которые поддерживает большинство из них.

Примечание

Вероятно, вы обратите внимание на то, что в табл. 15.1 приведено меньше механизмов, чем в справочнике клиентов SASL. Механизмов так много, что просто невозможно было процитировать весь справочник, но представленные в таблице механизмы принесут наибольшую пользу в гетерогенных сетях.

После определения механизмов, которые необходимы для поддержки почтовых клиентов, вам следует выбрать подходящее хранилище аутентификационных данных.

Определение хранилища аутентификационных данных и службы проверки паролей

Простым, но не слишком надежным местом, из которого можно было бы получить верительные данные пользователей, является локальная база данных пользователей вашей системы, `/etc/passwd`. Это хранилище аутентификационных данных уже существует, и если вы не являетесь интернет-провайдером, то, вероятно, уже создали учетные записи для всех пользователей почты.

К сожалению, очень многие администраторы[1] создают пользовательские учетные записи, разрешающие доступ к командному интерпретатору. Любой, кто получит доступ к именам пользователей и паролям, используемым в сеансе SMTP AUTH, легко может получить и доступ к командному интерпретатору на вашем сервере (что существенно облегчит получение прав суперпользователя). Уже одного этого факта достаточно для того, чтобы выбрать такое хранилище аутентификационных данных, которое никак не связано с пользователями вашей системы.

В теории идеальное хранилище аутентификационных данных не должно быть связано с пользователями системы, особенно если вы используете механизм открытого текста, т. к. последствия взлома или раскрытия верительных данных, используемых только для пересылки почты, не так серьезны, как взлом системы. Вы можете использовать независимую базу данных, такую как sasldb, SQL-сервер или даже LDAP-сервер. Механизмы, которые можно использовать для каждого из хранилищ, перечислены в табл. 15.2 и 15.3. Имейте в виду, что

[1] Честно говоря, я таких не встречал; надо быть совершенно беззаботным и наивным, чтобы десяткам или даже сотням людей, которые просто забирают почту с вашего сервера, давать интерактивный доступ к нему, этого даже не оправдать неумением верно задать умолчания для программы useradd. – *Примеч. науч. ред.*

выбор хранилища аутентификационных данных также определяет и службу проверки паролей.

Таблица 15.2. Хранилища аутентификационных данных и механизмы, совместимые со службой проверки паролей saslauthd

Хранилище	PLAIN	LOGIN	CRAM-MD5	DIGEST-MD5	OTP	NTLM
getpwent (обычные пользователи системы)	да	да	нет	нет	нет	нет
kerberos	да	да	нет	нет	нет	нет
pam	да	да	нет	нет	нет	нет
rimap (удаленный IMAP-сервер)	да	да	нет	нет	нет	нет
shadow	да	да	нет	нет	нет	нет
ldap	да	да	нет	нет	нет	нет

Таблица 15.3. Хранилища аутентификационных данных и механизмы, совместимые со службой проверки паролей auxprop

Хранилище	PLAIN	LOGIN	CRAM-MD5	DIGEST-MD5	OTP	NTLM
sasldb2	да	да	да	да	да	да
Sql	да	да	да	да	да	да

Когда хранилище аутентификационных данных выбрано, вы готовы к настройке Cyrus SASL для демона `smtpd`.

Установка и настройка Cyrus SASL

Библиотека Cyrus SASL необходима серверу Postfix, чтобы он мог аутентифицировать почтовые клиенты с помощью SMTP AUTH, а также для собственной аутентификации на удаленном почтовом сервере.

Если вам необходимо просто настроить Postfix-клиент `smtp` для аутентификации на удаленном сервере, то достаточно установить Cyrus SASL и перейти к главе 16.

Однако если вы хотите, чтобы Postfix, действуя как агент передачи сообщений, предоставлял SMTP AUTH удаленным почтовым клиентам, то вам следует установить и настроить Cyrus SASL. Кроме того, необходимо сообщить Postfix, как он должен взаимодействовать с Cyrus SASL для использования SMTP AUTH на стороне сервера. Итак, вам надо сделать следующее:

1. Установить Cyrus SASL.
2. Создать конфигурационный файл приложения Postfix.
3. Установить уровень журналирования.

4. Определить службу проверки паролей.
5. Выбрать механизмы SMTP AUTH.
6. Настроить saslauthd или auxprop.
7. Протестировать аутентификацию.

Установка Cyrus SASL

Если библиотека Cyrus SASL не предустановлена в вашей системе и не включена в нее в виде пакета, то вам надо скачать Cyrus SASL с веб-страницы *http://asg.web.cmu.edu/cyrus/download*.

Предупреждение

В следующем разделе подразумевается, что вы используете Cyrus SASL версии не ниже 2.1.17, при этом вполне естественно, что, когда вы будете читать эти строки, появятся более свежие версии. Если вы хотите использовать версию 2.1.17, то можете найти ее в CVS-архиве по адресу *http://asg.web.cmu.edu/cyrus/download/anoncvs.html*.

После распаковки библиотеки перейдите в каталог исходных текстов Cyrus SASL. Если вы используете версию SASL, полученную из CVS, то вам следует запустить `sh ./SMakefile` для создания сценария `configure`.

Запускаем следующую команду, чтобы настроить Cyrus SASL для всех хранилищ аутентификационных данных, описанных в оставшейся части главы:

```
# ./configure \
   --with-plugindir=/usr/lib/sasl2 \
   --disable-java \
   --disable-krb4 \
   --with-dblib=berkeley \
   --with-saslauthd=/var/state/saslauthd \
   --without-pwcheck \
   --with-devrandom=/dev/urandom \
   --enable-cram \
   --enable-digest \
   --enable-plain \
   --enable-login \
   --disable-otp \
   --enable-sql \
   --with-ldap=/usr \
   --with-mysql=/usr \
   --with-pgsql=/usr/lib/pgsql
```

Примечание

Cyrus SASL имеет множество других параметров конфигурации. Выполните в каталоге исходных текстов команду `./configure --help`, и вы узнаете, что поддерживает Cyrus SASL. Если вы считаете, что все хранилища вам не понадобятся, измените параметры.

После того как Makefile создаст сценарий конфигурирования, вы можете запустить команду `make` для сборки Cyrus SASL, а затем команду `make install` (от имени `root`) для установки библиотек.

Далее вам следует создать символическую ссылку. В процессе установки библиотеки SASL по умолчанию помещаются в каталог `/usr/local/lib/sasl2`, но параметры, заданные в сценарии конфигурирования, заставляют Cyrus SASL искать библиотеки в каталоге `/usr/lib/sasl2`. Создаем ссылку:

```
# ln -s /usr/local/lib/sasl2 /usr/lib/sasl2
```

Наконец, проверяем, настроен ли демон `syslogd` для записи в журнал сообщений Cyrus SASL. Сообщения Cyrus SASL используют для вывода в журнал группу `auth`, поэтому, если вы до сих пор не использовали эту группу, добавьте в файл `syslog.conf` приведенную ниже строку и перезапустите `syslogd`:

```
auth.*                          /var/log/auth
```

Создание файла конфигурации приложения Postfix

Каждому приложению, предоставляющему услуги SASL, необходимо сообщить, как использовать библиотеки SASL. Cyrus SASL вместо одного большого общего файла конфигурации имеет по одному файлу для каждого приложения. Это позволяет определить различные конфигурации для разных приложений. Файл конфигурации приложения для Postfix называется `smtpd.conf`, т. к. по умолчанию приложением Postfix, предоставляющим услуги SASL, является демон `smtpd`. Файл по умолчанию хранится в каталоге `/usr/lib/sasl2`.

Примечание

Пользователи Debian для использования SMTP AUTH должны поместить файл `smtpd.conf` в каталог `/etc/postfix/sasl`.

В некоторых операционных системах файл `smtpd.conf` содержит ряд настроек по умолчанию, так что заранее проверьте его наличие. Если файл не существует, создайте его от имени `root` следующими командами:

```
# touch /usr/lib/sasl2/smtpd.conf
# chmod 644 /usr/lib/sasl2/smtpd.conf
```

Эти команды не причинят вреда уже существующему файлу. После того как файл конфигурации создан, приступаем к настройке Postfix для использования библиотек SASL.

Предупреждение

Синтаксис файла конфигурации Cyrus отличается от синтаксиса, принятого в Postfix. Параметр и его значение должны быть указаны в одной строке. В Cyrus каждый параметр заканчивается двоеточием, а параметр и его значение разделяются пробелом.

Определение уровня журналирования

Первый параметр, который нужно указать в файле `/usr/lib/sasl2/smtpd.conf`, — это уровень журналирования (параметр `log_level`). Возможные значения параметра приведены в табл. 15.4.

Таблица 15.4. Уровни журналирования для Cyrus SASL

Значение log_level	Описание
0	Не вести журналирование
1	Записывать необычные ошибки, значение по умолчанию
2	Записывать все неудачные попытки аутентификации
3	Записывать предупреждения о нефатальных ошибках
4	Более подробно, чем 3
5	Более подробно, чем 4
6	Записывать трассировку внутренних протоколов
7	Записывать трассировку внутренних протоколов, включая пароли

Когда вы настроите и протестируете Cyrus SASL, нужно будет установить уровень журналирования в файле `smtpd.conf` не менее 3:

```
# Глобальные параметры
log_level: 3
```

Этот файл будет увеличиваться в размере по мере того, как вы будете выполнять действия, описанные в последующих разделах.

Определение службы проверки паролей

Теперь нужно сообщить серверу Postfix, какая служба проверки паролей будет использоваться для аутентификации пользователей. Вы должны точно определиться с тем, saslauthd или auxprop вы будете использовать, т. к. дальнейшие действия зависят от выбранной службы.

В Cyrus SASL служба проверки паролей определяется параметром `pwcheck_method`. Если вы планируете использовать saslauthd, добавьте в `smtpd.conf` такую строку:

```
# Глобальные параметры
log_level: 3
pwcheck_method: saslauthd
```

Если же вы хотите использовать вспомогательный плагин, ваш файл `smtpd.conf` будет таким:

```
# Глобальные параметры
log_level: 3
pwcheck_method: auxprop
```

Выбор механизмов SMTP AUTH

Cyrus SASL позволяет клиенту выбрать механизмы, которые будут использоваться для аутентификации. В определенных обстоятельствах это может привести к ошибкам аутентификации:

- Если вы предлагаете механизмы, требующие настройки, которая у вас не выполнена. Например, если вы не используете Kerberos, но ваш сервер предлагает этот механизм и клиент выбирает его, то аутентификация не будет успешной.
- Если вы выберете в качестве службы проверки паролей saslauthd, но не ограничите предлагаемые механизмы механизмами открытого текста. В данном случае аутентификация не будет успешной, если клиент выберет другой механизм, т. к. saslauthd может работать только с механизмами PLAIN и LOGIN.

Параметр `mech_list` позволяет вам гарантировать, что ваш сервер предлагает определенный список механизмов. Например, если вы используете saslauthd, то файл `smtpd.conf` *должен* быть таким:

```
# Глобальные параметры
log_level: 3
pwcheck_method: saslauthd
mech_list: PLAIN LOGIN
```

Для вспомогательных плагинов следует выбрать другой список, например так:

```
# Глобальные параметры
log_level: 3
pwcheck_method: auxprop
mech_list: PLAIN LOGIN CRAM-MD5 DIGEST-MD5
```

Когда механизмы выбраны, переходим к настройке saslauthd или вспомогательного плагина. Переходите к следующему разделу, если вы хотите настроить saslauthd, или же пропустите его и перейдите к разделу «Настройка вспомогательных плагинов (auxprop)».

Настройка saslauthd

saslauthd – это автономный демон, который взаимодействует с хранилищами аутентификационных данных. Настройка saslauthd осуществляется с помощью параметров командной строки. Прежде чем запускать демон, выполните следующие действия:

1. Проверьте, какие хранилища аутентификационных данных поддерживает ваша версия saslauthd.
2. Подготовьте окружение saslauthd.
3. Настройте хранилище аутентификационных данных для saslauthd.

Определение поддерживаемых хранилищ

Ничего не получится, если saslauthd не поддерживает выбранное вами хранилище аутентификационных данных. Команда `saslauthd -v` позволяет получить список хранилищ, поддерживаемых вашей версией saslauthd, например:

```
# saslauthd -v
saslauthd 2.1.17
authentication mechanisms: getpwent pam rimap shadow ldap
```

Обратите внимание на то, что saslauthd называет свои хранилища «механизмами аутентификации» (authentication mechanisms). Не путайте их с механизмами SMTP AUTH, такими как PLAIN и CRAM-MD5.

Если интересующего вас механизма нет в списке, придется пересобрать Cyrus SASL. Запустите `./configure --help` в каталоге исходных текстов Cyrus SASL и используйте соответствующие параметры для включения поддержки нужного хранилища.

Подготовка окружения saslauthd

Демону saslauthd необходим каталог состояний для хранения файла сокета и PID-файла. Сценарии установки Cyrus SASL не создают этот каталог, но если вы устанавливаете Cyrus SASL из бинарного пакета (такого, как RPM), инсталлятор пакета может создать каталог. Каталогом состояний по умолчанию является `/var/state/saslauthd`, также часто используется `/var/run/saslauthd`.

Вы можете определить местоположение каталога состояний в момент сборки, используя параметр `--with-saslauthd=DIR` в сценарии `configure`. Проверить наличие каталога состояний можно, запустив saslauthd в режиме отладки:

```
# saslauthd -a shadow -d
saslauthd[31076] :main     : num_procs  : 5
saslauthd[31076] :main     : mech_option: NULL
saslauthd[31076] :main     : run_path   : /var/run/saslauthd
saslauthd[31076] :main     : auth_mech  : shadow
saslauthd[31076] :main     : could not chdir to: /var/run/saslauthd
saslauthd[31076] :main     : chdir: No such file or directory
saslauthd[31076] :main      : Check to make sure the directory exists and is
saslauthd[31076] :main     : writable by the user this process runs as.
```

В этом примере `run_path` указывает каталог состояний, где saslauthd будет создавать сокет. Обратите внимание на то, что отладочный вывод saslauthd показывает на отсутствие такого каталога.

Если каталог не существует, создайте его и разрешите доступ только для `root` и членов группы `root`[1], например:

---

[1] В зависимости от настроек вашей системы (особенно если это FreeBSD), может понадобиться доступ группе `wheel`, а не группе `root`. – *Примеч. науч. ред.*

```
# mkdir /var/state/saslauthd
# chown root:root /var/state/saslauthd
# chmod 750 /var/state/saslauthd
```

Использование другого каталога состояний

Если вы хотите использовать каталог состояний, отличный от принятого по умолчанию (например, если в вашей системе уже есть каталог состояний для saslauthd), укажите в командной строке параметр `m dir`, чтобы изменить настройку по умолчанию. Например, если вы хотите использовать каталог `/var/run/saslauthd`, запустите демон следующим образом:

```
# saslauthd -m /var/run/saslauthd -a shadow
```

В данном случае путь – это имя каталога, не включающее в себя имя сокета (`mux`). Параметр `-a` относится к хранилищу аутентификационных данных, мы поговорим о нем в следующем разделе.

Необходимо также сообщить Postfix о новом каталоге состояний, задав параметр `saslauthd_path` в файле `smtpd.conf`. На этот раз имя файла сокета *обязательно* (см. строку, выделенную жирным курсовом):

```
# Глобальные параметры
log_level: 3
pwcheck_method: saslauthd
# saslauthd parameters
saslauthd_path: /var/run/saslauthd/mux
```

Каталог состояний подготовлен, теперь займемся подключением saslauthd к хранилищу аутентификационных данных.

Настройка хранилища аутентификационных данных для saslauthd

Демон saslauthd использует для выбора хранилища параметр `-a` *имя_хранилища*. Имя должно принадлежать хранилищу из списка, полученного командой `saslauthd -v` (см. табл. 15.2). В следующем примере saslauthd выбирает хранилище `shadow` для чтения данных из теневого файла паролей:

```
# saslauthd -a shadow
```

Следующие разделы будут посвящены наиболее часто используемым с saslauthd хранилищам аутентификационных данных. Полный перечень таких хранилищ можно найти на странице руководства saslauthd(8).

Аутентификация на основе локальных учетных записей пользователей

Демон saslauthd может обращаться к локальному файлу паролей (в большинстве UNIX-систем), а может обращаться к локальному те-

невому файлу паролей в системах, поддерживающих такие пароли. Для чтения данных из обычного файла паролей (/etc/passwd) используйте параметр -a getpwent.

```
# saslauthd -a getpwent
```

В системах, использующих теневые пароли, можно запустить демон saslauthd с параметром -a shadow, чтобы обеспечить ему доступ к файлу /etc/shadow; для доступа к этому файлу необходимо выполнять saslauthd от имени root:

```
# saslauthd -a shadow
```

PAM-аутентификация

Демон saslauthd поддерживает библиотеки PAM (Pluggable Authentication Modules – подключаемые модули аутентификации) для аутентификации SMTP-пользователей. Для получения доступа к хранилищам аутентификационных данных, поддерживаемых PAM, создайте файл /etc/pam.d/smtp и добавьте необходимые параметры конфигурации или добавьте соответствующие настройки в файл etc/pam.conf.

Приведем пример того, что можно было бы поместить в /etc/pam.d/smtp в системе Red Hat Fedora Core 1:

```
#%PAM-1.0
auth       required     /lib/security/pam_stack.so service=system-auth
account    required     /lib/security/pam_stack.so service=system-auth
```

Примечание

Файл конфигурации должен называться smtp, т. к. RFC 2554 говорит о том, что имя службы для SASL по SMTP – это smtp. Postfix передает значение smtp в качестве имени службы функции sasl_server_new(). Затем это имя службы передается saslauthd и, в конце концов, библиотеке PAM, которая ищет в smtp инструкции по аутентификации.

После настройки PAM запускаем saslauthd следующим образом:

```
# saslauthd -a pam
```

IMAP-аутентификация

Демон saslauthd поддерживает необычное хранилище аутентификационных данных, которое подключается к серверу IMAP. Особенность этого способа заключается в том, что имя пользователя и пароль проверяются сервером IMAP, а не библиотеками SASL.

При настройке IMAP в качестве хранилища аутентификационных данных для saslauthd используем два параметра: первый – это уже известный вам -a для выбора хранилища, второй параметр, -O, отвечает за выбор IMAP-сервера:

```
# saslauthd -a rimap -O imap.example.com
```

LDAP-аутентификация

Демон saslauthd может читать верительные данные с сервера OpenLDAP. Запросы LDAP и параметры подключения к серверу LDAP могут быть очень сложными, поэтому они не передаются в saslauthd в командной строке. Конфигурация считывается из отдельного файла, который по умолчанию имеет имя `/usr/local/etc/saslauthd.conf`, но вы можете указать и другое название в параметре `-O file`.

Рассмотрим пример файла `saslauthd.conf`:

```
ldap_servers: ldap://127.0.0.1/ ldap://172.16.10.7/
ldap_bind_dn: cn=saslauthd,dc=example,dc=com
ldap_bind_pw: Yanggt!
ldap_timeout: 10
ldap_time_limit: 10
ldap_scope: sub
ldap_search_base: dc=people,dc=example,dc=com
ldap_auth_method: bind
ldap_filter: (|(&(cn=%u)(&(uid=%u@%r)(smtpAuth=Y)))
ldap_debug: 0
ldap_verbose: off
ldap_ssl: no
ldap_start_tls: no
ldap_referrals: yes
```

Естественно, запрос (в примере он определен параметрами `ldap_search_base` и `ldap_filter`) зависит от вашей собственной конфигурации LDAP.

Примечание

На самом деле существует гораздо больше параметров LDAP, чем использовано в примере. Полный перечень вы найдете в модуле `auth_ldap` для saslauthd.

Давайте будем считать, что вы поместили конфигурацию в файл `/etc/saslauthd.conf`. Тогда будем запускать saslauthd так:

```
# saslauthd -a ldap -O /etc/saslauthd.conf
```

Настройка вспомогательных плагинов (auxprop)

В отличие от случая с хранилищами аутентификационных данных для saslauthd, приложения, использующие вспомогательные плагины auxprop, непосредственно запускают эти плагины, читая конфигурацию из собственного файла конфигурации SASL приложения. Как говорилось ранее, конфигурационный файл приложения для Postfix называется `smtpd.conf`. В последующих разделах мы покажем, как настроить вспомогательные плагины, которые поставляются с исходными текстами SASL.

Примечание

Здесь перечислены далеко не все плагины для SASL, например, не упомянут `ldapdb`, который присутствует в каталоге `/contrib` исходных файлов OpenLDAP.

Плагин ldapdb великолепен, но его сложно устанавливать, т. к. в него не включен ни Makefile, ни какой-либо другой инструмент для сборки. В этой главе мы сосредоточимся на конфигурациях для пользователей средней квалификации.

Использование плагина sasldb2

Библиотеки Cyrus SASL поставляются со стандартным плагином sasldb2, который применяется в основном в Cyrus IMAP, но может использоваться и отдельно. Плагин sasldb2 включает в себя две утилиты: saslpasswd2 для управления пользователями и sasldblistusers2 для получения списка всех пользователей в sasldb2.

Предупреждение

Имена базы данных и утилит заканчиваются цифрой 2, т. к. они принадлежат Cyrus SASL 2.x. Их пришлось переименовать во избежание конфликта с Cyrus SASL 1.x, т. к. интерфейс API изменился при переходе на новую версию. Если вам встретятся файлы, названия которых не содержат цифры, то, вероятнее всего, окажется, что они относятся к Cyrus SASL 1.x и не будут работать с версией 2.

Для настройки Cyrus SASL с использованием sasldb2 необходимо выполнить следующие действия:

1. Создать базу данных sasldb2.
2. Настроить SASL для чтения из базы данных.

Создание базы данных sasldb2

Вы можете создать базу данных sasldb2, выполнив от имени root команду saslpasswd2. Параметр -c создает базу данных sasldb2 в файле /etc/sasldb2. В приведенном ниже примере создается база данных, добавляется пользователь и область для хоста Postfix myhostname (невозможно создать базу данных, не добавив пользователя):

```
# saslpasswd2 -c -u `postconf -h myhostname` test
Password:
Again (for verification):
```

Предупреждение

В дополнение к верительным данным плагин sasldb2 запрашивает область. Postfix использует в качестве области значение параметра smtpd_sasl_local_domain (по умолчанию он пуст). Postfix может иметь только одну область для каждого экземпляра smtpd, ограничивая аутентификацию одной областью.

После создания базы данных ограничиваем доступ к ней пользователем root и группой postfix:

```
# chmod 640 /etc/sasldb2
# chgrp postfix /etc/sasldb2
```

```
# ls -l /etc/sasldb2
-rw-r-----    1 root    postfix    12288 Feb  4 16:23 /etc/sasldb2
```

Если вы предоставляете механизм OTP, то должны разрешить серверу Postfix запись в файл базы данных, чтобы он мог помечать просроченные пароли. Возможно, вам придется изменить принадлежность файла и права на него, если еще какой-то группе понадобится доступ к файлу базы данных.

Настройка SASL для чтения из базы данных sasldb2

Для того чтобы сообщить серверу Postfix о базе данных sasldb2, отредактируйте файл `smtpd.conf`, указав службу проверки паролей `auxprop` и плагин `sasldb`:

```
# Глобальные параметры
log_level: 3
pwcheck_method: auxprop
mech_list: PLAIN LOGIN CRAM-MD5 DIGEST-MD5
# параметры вспомогательных плагинов
auxprop_plugin: sasldb
```

Использование плагина sql

Cyrus SASL 2 предоставляет доступ к двум широко известным реляционным базам данных: MySQL и PostgreSQL. Доступ к обеим базам данных обеспечивает плагин `sql`, при этом используются одни и те же параметры конфигурации:

`sql_engine`

Параметр `sql_engine` указывает тип базы данных. В версии Cyrus SASL 2.1.17 вы можете выбрать `mysql` или `pgsql`.

`sql_hostnames`

Параметр `sql_hostnames` определяет имя сервера базы данных. Вы можете указать одно или несколько полностью определенных доменных имен или IP-адресов, разделенных запятыми. Если вы выбираете `localhost`, то обработчик SQL будет пытаться установить соединение через сокет.

`sql_database`

Параметр `sql_database` определяет имя базы данных, к которой вы хотите подключиться.

`sql_user`

Параметр `sql_user` определяет имя пользователя базы данных.

`sql_passwd`

Параметр `sql_passwd` определяет пароль (открытым текстом) для пользователя базы данных.

`sql_select`

Параметр `sql_select` определяет команду `SELECT`, используемую для поиска пароля по заданному имени пользователя и области.

`sql_insert`

Параметр `sql_insert` определяет команду `INSERT`, которая позволяет библиотеке SASL создавать пользователей в базе данных SQL (делая ее доступной таким программам, как saslpasswd2).

`sql_update`

Параметр `sql_update` определяет команду `UPDATE`, которая позволяет библиотеке SASL или плагину обновлять пользователя в базе данных SQL для такого механизма, как OTP. Параметр `sql_update` должен использоваться в сочетании с `sql_insert`.

`sql_usessl`

Параметр `sql_usessl` обеспечивает шифрование соединения с базой данных. По умолчанию шифрование отключено (`sql_usessl: no`); для включения SSL используйте значение `yes`, `1`, `on` или `true`.

При создании запросов вы можете использовать в параметрах следующие макросы Cyrus SASL:

`%u` Этот макрос заменяется именем пользователя, предоставленным в процессе аутентификации.

`%p` Это заполнитель для пароля; имя столбца по умолчанию для незашифрованных паролей.

`%r` Этот макрос заменяется областью (realm), указанной при аутентификации.

`%v` Указывает передаваемое значение, которое должно заменить существующее значение при SQL-операции `UPDATE` или `INSERT`.

Предупреждение

Если вы определяете запрос в файле `smtpd.conf` при помощи макросов, они должны по отдельности заключаться в одиночные кавычки (').

Настройка MySQL для SASL

Первое, что нужно сделать при настройке MySQL для SASL в Postfix, – это создать базу данных и таблицу. Приведенная ниже команда SQL создает таблицу с полями по умолчанию, наличия которых ожидает Cyrus SASL, а также с дополнительным полем, которое позволяет запретить пересылку для определенного пользователя:

```
mysql> CREATE DATABASE mail;
mysql> CONNECT mail;
mysql> CREATE TABLE `users` (
       `id` int(11) unsigned NOT NULL auto_increment,
       `username` varchar(255) NOT NULL default '0',
       `userrealm` varchar(255) NOT NULL default 'mail.example.com',
```

```
`userpassword` varchar(255) NOT NULL default '1stP@ss',
`auth` tinyint(1) default '1',
PRIMARY KEY  (`id`),
UNIQUE KEY `id` (`id`)
) TYPE=MyISAM COMMENT='SMTP AUTH relay users';
```

Как видите, все поля (id, username, userrealm, userpassword и auth) имеют значения по умолчанию. Особенно важно наличие пароля по умолчанию – атакующий не сможет попытать счастья с пустым паролем.

Затем создаем пользователя MySQL, который может читать и записывать данные в базу данных авторизации SASL. Например, такая последовательность команд создает пользователя с именем postfix:

```
mysql> CONNECT mysql;
mysql> INSERT INTO user VALUES
       ('localhost','postfix','','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y',
        'Y','Y','Y','Y','Y');
mysql> UPDATE mysql.user SET password=PASSWORD("Yanggt!") WHERE
       user='postfix'        AND host='localhost';
mysql> GRANT SELECT, UPDATE ON mail.users TO 'postfix'@'localhost';
mysql> FLUSH PRIVILEGES;
```

Добавляем в таблицу пользователей тестовую запись следующей командой:

```
mysql> INSERT INTO `users` VALUES
       (1,'test','mail.example.com','testpass',0);
```

Наконец, когда база данных MySQL готова, настраиваем хранилище аутентификационных данных для вспомогательного плагина в файле smtpd.conf:

```
# Глобальные параметры
log_level: 3
pwcheck_method: auxprop
mech_list: PLAIN LOGIN CRAM-MD5 DIGEST-MD5
# параметры вспомогательных плагинов
auxprop_plugin: sql
sql_engine: mysql
sql_hostnames: localhost
sql_database: mail
sql_user: postfix
sql_passwd: Yanggt!
sql_select: SELECT %p FROM users WHERE username = '%u' AND userrealm = '%r'
            AND auth = '1'
sql_usessl: no
```

Обратите внимание на то, что указана служба проверки паролей auxprop и плагин sql. Описания остальных параметров и макросов приведены в разделе «Использование плагина sql».

Примечание

Подробную информацию о параметрах и обозначениях вы найдете в файле `options.html` в каталоге `doc` Cyrus SASL.

Настройка PostgreSQL для SASL

Процесс настройки PostgreSQL для SASL очень похож на только что рассмотренный для MySQL. Создаем базу данных `mail` в PostgreSQL:

```
# createdb mail
CREATE DATABASE
```

Теперь подключаемся к базе данных и создаем таблицу для пользователей SASL следующим образом:

```
# psql -d mail
Welcome to psql 7.3.4, the PostgreSQL interactive terminal.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit
mail=# CREATE TABLE public.users
mail-# (
mail(#   id int4 NOT NULL,
mail(#   "username" varchar(255),
mail(#   "userrealm" varchar(255),
mail(#   "userpassword" varchar(255),
mail(#   auth int2 DEFAULT 0,
mail(#   CONSTRAINT id PRIMARY KEY (id)
mail(# ) WITHOUT OIDS;
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'id'
         for table 'users'
CREATE TABLE
mail=# COMMENT ON TABLE public.users IS 'mail users';
COMMENT
mail=#
```

Создаем пользователя, который может читать и писать данные в базу данных:

```
mail=# CREATE USER postfix PASSWORD 'Yanggt!' NOCREATEDB NOCREATEUSER;
CREATE USER
```

Предоставляем этому пользователю привилегии на операции `SELECT` и `UPDATE` для таблицы:

```
mail=# GRANT UPDATE,SELECT ON users TO postfix;
GRANT
```

Создаем тестовую учетную запись для таблицы `mail`:

```
mail=# INSERT INTO users VALUES
       ('1','test','mail.example.com','testpass','1');
```

Теперь, когда база данных PostgreSQL готова, настраиваем вспомогательный плагин в файле `smtpd.conf`:

```
# Глобальные параметры
log_level: 3
pwcheck_method: auxprop
mech_list: PLAIN LOGIN CRAM-MD5 DIGEST-MD5
# параметры вспомогательных плагинов
auxprop_plugin: sql
sql_engine: pgsql
sql_hostnames: localhost
sql_database: mail
sql_user: postfix
sql_passwd: Yanggt!
sql_select: SELECT %p FROM users WHERE username = '%u' AND realm = '%r' AND
auth = '1'
sql_usessl: no
```

Когда хранилище аутентификационных данных настроено, можно приступать к тестированию аутентификации.

Тестирование аутентификации

После того как вы настроили Cyrus SASL с saslauthd или вспомогательным плагином и одним хранилищем аутентификационных данных, необходимо протестировать то, что получилось, *прежде* чем настраивать SMTP AUTH в Postfix согласно описанию в главе 16. Опыт показывает, что большая часть проблем SMTP AUTH бывает вызвана некорректной настройкой Cyrus SASL, а не Postfix.

Начинаем с поиска инструментов тестирования. Если вы устанавливали Cyrus SASL не из исходных текстов, задача может оказаться не очень простой (по умолчанию инструменты хранятся в подкаталоге `sample/` дистрибутива в исходных текстах). Если вы используете Cyrus SASL из дистрибутива операционной системы, вам придется поискать программы с именами client и server.

Примечание

В разных дистрибутивах операционных систем имена исполняемых файлов совсем не обязательно будут одинаковыми, к тому же некоторые дистрибутивы устанавливают не все исполняемые файлы. Найдите пакеты `cyrus-*-devel` и посмотрите на названия программ в этих пакетах. Они не обязательно будут называться client и server.

Например, Red Hat Linux смешивает Cyrus SASL 1.x и 2.x, переименовывая исполняемые файлы в `sasl-sample-client` и `sasl-sample-server` для Cyrus SASL 1.x и `sasl2-sample-client` и `sasl2-sample-server` для 2.x.

Когда вы найдете клиентскую и серверную программы, выполните следующие действия для тестирования аутентификации:

1. Запустите saslauthd, если эта служба необходима используемому хранилищу аутентификационных данных.
2. Создайте файл конфигурации сервера.
3. Запустите серверную программу.
4. Протестируйте аутентификацию при помощи клиентской программы.

Запуск saslauthd

Если вы выбираете хранилище, которое использует службу проверки паролей saslauthd (т. е. не использует вспомогательный плагин, такой как sasldb, или базу данных SQL), то следует запустить saslauthd в режиме отладки из командной строки. Не используйте сценарий init; вам нужно будет использовать параметр `d`, который указывает основному экземпляру saslauthd на то, что следует не переходить в режим демона, а оставаться связанным с текущим терминалом и выводить отладочные данные.

Приведем пример запуска saslauthd для аутентификации на основе теневых паролей:

```
# saslauthd -m /var/state/saslauthd -a shadow -d
saslauthd[4401] :main            : num_procs  : 5
saslauthd[4401] :main            : mech_option: NULL
saslauthd[4401] :main            : run_path   : /var/run/saslauthd
saslauthd[4401] :main            : auth_mech  : shadow
saslauthd[4401] :ipc_init        : using accept lock file: /var/run/
                                   saslauthd/mux.accept
saslauthd[4401] :detach_tty      : master pid is: 0
saslauthd[4401] :ipc_init        : listening on socket: /var/run/saslauthd/mux
saslauthd[4401] :main            : using process model
saslauthd[4402] :get_accept_lock : acquired accept lock
saslauthd[4401] :have_baby       : forked child: 4402
saslauthd[4401] :have_baby       : forked child: 4403
saslauthd[4401] :have_baby       : forked child: 4404
saslauthd[4401] :have_baby       : forked child: 4405
```

Создание файла конфигурации сервера

Теперь нам нужно создать файл конфигурации для серверной программы. Если помните, каждому приложению SASL нужен собственный файл конфигурации, так что тестовой серверной программе потребуется файл `sample.conf`. Однако тестовая конфигурация должна совпадать с конфигурацией, используемой для Postfix. Проще всего добиться этого, создав символическую ссылку на файл `smtpd.conf`:

```
# cd /usr/lib/sasl2/
# ln -s smtpd.conf sample.conf
```

Запуск серверной программы

Запустите еще один командный процессор[1] и в нем запустите серверную программу с параметрами `-s` и `-p`, указывающими службу и порт для сервера:

```
# server -s rcmd -p 8000
trying 10, 1, 6
socket: Address family not supported by protocol
trying 2, 1, 6
```

Убедитесь в том, что данный порт еще не используется на вашей машине.

Примечание

Назначение `rcmd` недостаточно документировано.

Тестирование при помощи клиентской программы

Наконец, запускаем клиентскую программу и ждем, когда она подключится к серверу. Подключившись, клиентская программа просит вас ввести идентификатор аутентификации, идентификатор авторизации и пароль. Используйте параметр командной строки `-m` для выбора механизма SASL. Следующий пример использует `test` как идентификатор аутентификации и авторизации, а `testpass` – как пароль для `localhost` (`127.0.0.1`):

```
# client -s rcmd -p 8000 -m PLAIN 127.0.0.1
receiving capability list... recv: {11}
PLAIN LOGIN
PLAIN LOGIN
please enter an authentication id: test
please enter an authorization id: test
Password:
send: {5}
PLAIN
send: {1}
Y
send: {18}
test[0]test[0]testpass
successful authentication
closing connection
```

Ищем сообщение `successful authentication`. Вы также можете отслеживать взаимодействие со стороны серверной программы. В следующем примере отображено инициирование соединения, а также поступление и проверка верительных данных:

[1] В новом окне, на другом терминале – так, чтобы можно было видеть и тот, в котором saslauthd выводит отладочные данные. – *Примеч. науч. ред.*

```
# server -s rcmd -p 8000
trying 10, 1, 6
socket: Address family not supported by protocol
trying 2, 1, 6
accepted new connection
send: {11}
PLAIN LOGIN
recv: {5}
PLAIN
recv: {1}
Y
recv: {18}
test[0]test[0]testpass
successful authentication 'test'
closing connection
```

Если вы используете хранилище, которому необходима служба проверки паролей saslauthd, то вы можете посмотреть, что происходит, когда saslauthd проверяет верительные данные:

```
# saslauthd -m /var/run/saslauthd -a shadow -d
saslauthd[4547] :main            : num_procs  : 5
saslauthd[4547] :main            : mech_option: NULL
saslauthd[4547] :main            : run_path   : /var/run/saslauthd
saslauthd[4547] :main            : auth_mech  : shadow
saslauthd[4547] :ipc_init        : using accept lock file: /var/run/
                                   saslauthd/mux.accept
saslauthd[4547] :detach_tty      : master pid is: 0
saslauthd[4547] :ipc_init        : listening on socket: /var/run/
                                    saslauthd/mux
saslauthd[4547] :main            : using process model
saslauthd[4548] :get_accept_lock : acquired accept lock
saslauthd[4547] :have_baby       : forked child: 4548
saslauthd[4547] :have_baby       : forked child: 4549
saslauthd[4547] :have_baby       : forked child: 4550
saslauthd[4547] :have_baby       : forked child: 4551
saslauthd[4548] :rel_accept_lock : released accept lock
saslauthd[4548] :do_auth         : auth success: [user=test]
     [service=rcmd] [realm=] [mech=shadow]
saslauthd[4548] :do_request      : response: OK
saslauthd[4548] :get_accept_lock : acquired accept lock
```

Если что-то пойдет неправильно, вы сможете разобраться с проблемой, выполнив следующие действия:

1. Если вы пользуетесь службой проверки паролей saslauthd, то поищите строку `[reason=...]` в отладочных выходных данных.
2. Проверьте, корректно ли заданы пользователь и пароль в вашем хранилище аутентификационных данных.
3. Убедитесь, что у saslauthd есть права доступа к вашему хранилищу.
4. Убедитесь, что вы передали правильные строки для имени пользователя и пароля.

После успешной аутентификации переходите к главе 16 и настраивайте Postfix-демон `smtpd` для предоставления SMTP AUTH почтовым клиентам.

Будущее SMTP AUTH

Текущая реализация SMTP AUTH далека от окончательной. Ожидается, что в последующих версиях Postfix SASL претерпит значительные изменения. В настоящее время библиотеки SASL, которые обращаются к хранилищам аутентификации, встраиваются в демон `smtpd` и используются им же (это тот демон, который занимается взаимодействием с почтовыми клиентами). Текущая организация соединения изображена на рис. 15.6.

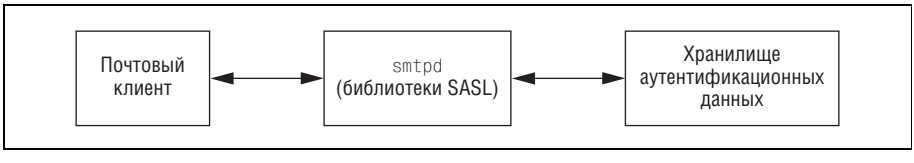

Рис. 15.6. *Текущая интеграция SASL в Postfix*

Доступ к хранилищам аутентификационных данных обычно осуществляется от имени привилегированного пользователя, так что каждый, кто сможет захватить управление демоном, ответственным за SMTP AUTH, будет чрезвычайно близок к базе данных пользователей системы.

Postfix всячески пытается избегать привилегированных процессов, особенно в случаях, когда демон аутентификации непосредственно открыт для (всегда враждебной) сети. Кроме того, сложность – это враг безопасности, а Cyrus SASL определенно является сложным.

В будущем в Postfix появится новый демон (возможно, он будет называться `sasld`), единственной задачей которого будет поддержание соединения между Postfix и библиотеками Cyrus SASL. Взаимодействие сервера Postfix с новым демоном будет максимально простым. Возможная новая модель соединения изображена на рис. 15.7.

Рис. 15.7. *Будущая интеграция SASL в Postfix*

В новой версии демон, который подвергается воздействию сети, будет менее уязвимым, что приблизит Postfix к достижению одной из его основных целей – обеспечению надежного и безопасного обслуживания.

16

SMTP-аутентификация

SMTP-аутентификация (SMTP AUTH) позволяет авторизованным почтовым клиентам с динамическими IP-адресами пересылать сообщения через ваш сервер, не превращая его в открытый ретранслятор. В этой главе будет показано, как активировать SMTP AUTH в серверной и клиентской частях Postfix.

Проверка поддержки SMTP AUTH в Postfix

В исходных текстах Postfix предусмотрена поддержка SMTP-аутентификации, но по умолчанию она не включена, т. к. Postfix поставляется без библиотеки SMTP AUTH, которая фактически выполняет эту работу. Вам нужно будет указать эту библиотеку при сборке Postfix.

Многие дистрибутивы предлагают пакеты Postfix с поддержкой SMTP AUTH. Вы без труда можете проверить, поддерживается ли SMTP AUTH в вашей версии Postfix, выполнив команду `ldd `postconf h daemon_directory`/smtpd` от имени `root` (ищем в выводе строку с `libsasl2.so`):

```
# ldd `postconf -h daemon_directory`/smtpd
        libldap.so.2 => /usr/lib/libldap.so.2 (0x00117000)
        liblber.so.2 => /usr/lib/liblber.so.2 (0x008a9000)
        libpcre.so.0 => /lib/libpcre.so.0 (0x00b86000)
        libsasl2.so.2 => /usr/lib/libsasl2.so.2 (0x00101000) ❶
        libssl.so.4 => /lib/libssl.so.4 (0x00b11000)
        libcrypto.so.4 => /lib/libcrypto.so.4 (0x00977000)
        libgssapi_krb5.so.2 => /usr/lib/libgssapi_krb5.so.2 (0x00afc000)
        libkrb5.so.3 => /usr/lib/libkrb5.so.3 (0x00a6f000)
        libcom_err.so.2 => /lib/libcom_err.so.2 (0x00a6a000)
        libk5crypto.so.3 => /usr/lib/libk5crypto.so.3 (0x00ad8000)
        libresolv.so.2 => /lib/libresolv.so.2 (0x00965000)
        libdl.so.2 => /lib/libdl.so.2 (0x008b8000)
        libz.so.1 => /usr/lib/libz.so.1 (0x008bd000)
```

```
libdb-4.1.so => /lib/libdb-4.1.so (0x00c18000)
libnsl.so.1 => /lib/libnsl.so.1 (0x008fe000)
libc.so.6 => /lib/libc.so.6 (0x0076e000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x008d0000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00759000)
libpthread.so.0 => /lib/libpthread.so.0 (0x00912000)
```

❶ Присутствие в выводе программы ldd строки `libsasl.so.2` (в настоящее время это основная версия Cyrus SASL) показывает, что пакет Postfix был собран с поддержкой SASL, а значит, вы уже можете приступать к настройке SMTP-аутентификации.

Примечание

Postfix поддерживает и более старую версию Cyrus SASL – 1.5. Если вы обнаружите `libsasl.so.7` в выводе команды `ldd`, это будет означать, что ваша версия Postfix была собрана с библиотеками Cyrus SASL 1.5. Если в выводе присутствует только `libsasl.so.7`, следует обновить программное обеспечение, перейдя на версию Cyrus SASL 2. Поддержка хранилищ аутентификационных данных в новой версии значительно усовершенствована по сравнению с версией 1.5.

Добавление поддержки SMTP AUTH в Postfix

Если ваша версия Postfix не поддерживает SASL, а вы хотите использовать SMTP AUTH, вам придется пересобрать Postfix. Начнем с поиска библиотек и заголовочных файлов Cyrus SASL в вашей системе. Библиотеки ищем такой командой:

```
# find /usr -name 'libsasl*.*'
/usr/lib/sasl2/libsasldb.la
/usr/lib/sasl2/libsasldb.a
/usr/lib/sasl2/libsasldb.so.2.0.15
/usr/lib/sasl2/libsasldb.so
/usr/lib/sasl2/libsasldb.so.2
/usr/lib/libsasl2.so.2.0.15
/usr/lib/libsasl2.so.2
/usr/lib/libsasl2.la
/usr/lib/libsasl2.a
/usr/lib/libsasl2.so
```

Как видите, библиотека SASL 2 хранится в каталоге `/usr/lib`. Запомните это и переходите к поиску соответствующих включаемых файлов:

```
# find /usr -name '*sasl*.h'
/usr/include/sasl/sasl.h
/usr/include/sasl/saslplug.h
/usr/include/sasl/saslutil.h
```

Примечание

В дистрибутивах Linux заголовочные файлы и библиотеки находятся в разных пакетах из-за ошибочного мнения, что тем самым экономится дисковое про-

странство. Если вам не удается найти в своей системе включаемые файлы для Cyrus SASL, но библиотеки вы нашли, то найдите и установите пакеты SASL, названия которых заканчиваются на `-dev` или `-devel`.

Если в вашей системе нет библиотеки Cyrus SASL, обратитесь к главе 15 за информацией о ее настройке и установке. После того как вы определите, где находятся заголовочные и включаемые файлы[1], выполните следующие действия для сборки Postfix с поддержкой SASL:

1. Распакуйте исходные тексты Postfix от имени обычного пользователя.
2. Перейдите в каталог исходных текстов Postfix.
3. Задайте параметры сборки и выполните команды `make makefiles` и `make`, например:

```
$ CCARGS="-DUSE_SASL_AUTH -I/usr/include/sasl" AUXLIBS="-L/usr/lib -lsasl2"
  make makefiles
$ make
```

   Помните о том, что здесь указаны параметры только для SASL; если вы захотите использовать какие-то дополнительные параметры, обратитесь к описаниям в файлах `*_README` подкаталога `README_FILES` каталога исходных текстов Postfix.
4. Переключитесь на суперпользователя (`root`).
5. Если это ваша первая установка Postfix, выполните команду `make install`. Если же вы заменяете или обновляете существующую версию, выполните команду `make upgrade`.
6. Проверьте, поддерживается ли SASL (см. раздел «Проверка поддержки SMTP AUTH в Postfix» в начале главы).

SMTP-аутентификация на стороне сервера

В этом разделе будет рассказано о том, как настроить Postfix-сервер `smtpd` для предоставления SMTP AUTH почтовым клиентам. После успешной аутентификации клиенты смогут пересылать сообщения через сервер Postfix, даже если их IP-адреса не относятся к диапазону, определенному в параметре конфигурации `mynetworks`.

Предупреждение

Корректная настройка SMTP AUTH на сервере подразумевает не только конфигурирование и связывание Postfix с библиотекой Cyrus SASL, но и конфигурирование Cyrus SASL для взаимодействия с хранилищем аутентификационных данных (см. главу 15).

[1] У автора так, но на самом деле имеются в виду заголовочные файлы и файлы библиотек. – *Примеч. науч. ред.*

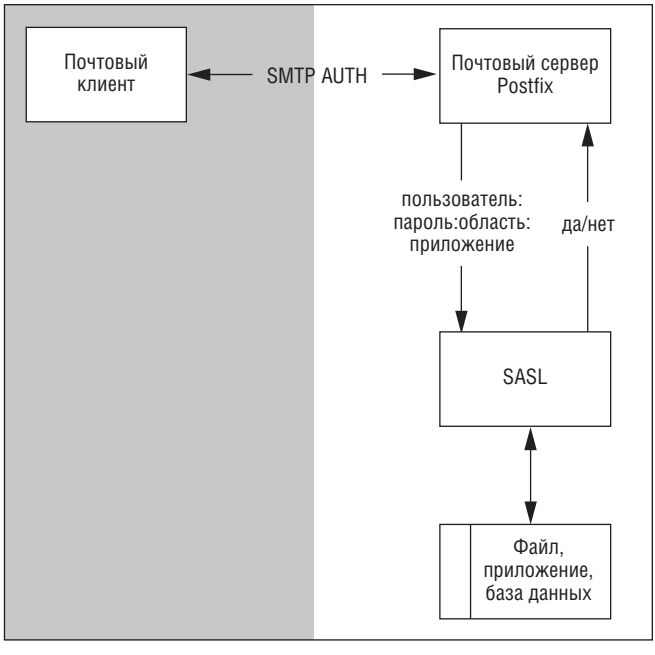

***Рис. 16.1.** Архитектура SMTP AUTH на стороне сервера*

На рис. 16.1 представлен почтовый клиент, который, прежде чем отправить сообщение для пересылки удаленному получателю, аутентифицируется на почтовом сервере. Сервер сравнивает верительные данные, полученные от почтового клиента, с данными, хранящимися в хранилище аутентификационных данных, и разрешает пересылку только в случае их совпадения.

Включение и настройка сервера

После настройки хранилища аутентификационных данных в SASL (согласно описанию в главе 15) вам следует настроить сервер следующим образом:

1. Включить серверную часть SMTP AUTH.
2. Настроить механизмы SASL, которые будут предоставляться клиентам.
3. Настроить поддержку SMTP AUTH для нестандартных почтовых клиентов.
4. Настроить область (realm), которую Postfix будет передавать библиотеке SASL.
5. Определить разрешения на пересылку в Postfix.

Примечание

Запустить Postfix с помощью chroot и обеспечить SMTP AUTH несложно. Следуйте приведенным в этой главе инструкциям по настройке серверной части аутентификации. Когда вы убедитесь в том, что SMTP-аутентификация без chroot работает, обратитесь к главе 20, где запуск SMTP AUTH в Postfix, запущенном через chroot, описан в качестве примера конфигурации с использованием chroot.

Включение серверной части SMTP AUTH

Первое, что необходимо сделать, – это включить SMTP-аутентификацию для Postfix-сервера `smtpd`, т. к. по умолчанию данная функциональность отключена. Используем параметр `smtpd_sasl_auth_enable` в файле `main.cf`:

```
smtpd_sasl_auth_enable = yes
```

Настройка механизмов SASL

Теперь следует определить механизмы аутентификации, которые сервер Postfix будет предлагать почтовым клиентам. Cyrus SASL предоставляет множество механизмов, начиная с анонимной «аутентификации» и заканчивая очень мощными системами, такими как Kerberos.

Управление предоставляемыми механизмами осуществляется параметром `smtpd_sasl_security_options`. Вы можете указать в нем разделенный запятыми список из одного или более значений, описанных далее:

`noanonymous`

Это значение гарантирует, что сервер действительно проверяет верительные данные клиента. Это настройка по умолчанию, которую необходимо сохранить, т. к. некоторым спамерам известно об анонимной SMTP-аутентификации. Список значений вашего параметра `smtpd_sasl_security_options` всегда должен включать в себя `noanonymous`; в противном случае ваш почтовый сервер почти наверняка превратится в открытый ретранслятор.

`noplaintext`

Добавлением в список значения `noplaintext` вы исключаете использование всех механизмов открытого текста, таких как PLAIN и LOGIN. Рекомендуется его использовать, т. к. отправляемые открытым текстом верительные данные могут быть легко перехвачены в сети.

`noactive`

Это значение исключает использование механизмов SASL, которые восприимчивы к активным (не по словарю) атакам. Например, взаимная аутентификация не чувствительна к активным атакам.

`nodictionary`

Исключаются все механизмы, не устойчивые к атакам по словарю. Атакующий по словарю использует грубую силу, пробуя пароль за паролем, пока один из них не подойдет.

`mutual_auth`

Использование `mutual_auth` означает поддержку только механизмов, обеспечивающих взаимную аутентификацию. В этом случае и сервер должен аутентифицировать себя для клиента.

При тестировании конфигурации изменять данный параметр не следует; значение по умолчанию `smtpd_sasl_security_options = noanonymous` охраняет вас от спамеров, но разрешает применение механизмов открытого текста, что несколько облегчает отладку. Затем, когда вы убедитесь, что SMTP AUTH работает, *следует* отключить механизмы открытого текста, расширив список значений параметра `smtpd_sasl_security_options` в файле `main.cf` как минимум до такого:

```
smtpd_sasl_security_options = noanonymous, noplaintext
```

Предупреждение

Почтовый клиент, использующий механизмы открытого текста, отправляет имя пользователя и пароль в виде строки в кодировке base64. Декодирование такой строки тривиально, так что кто угодно, прослушивающий SMTP-диалог, может использовать эти данные для злоупотребления вашим сервером. К сожалению, это единственный механизм, поддерживаемый Outlook Express. Если вы хотите предоставлять механизмы открытого текста, делайте это только с использованием зашифрованного соединения (см. главу 18).

Настройка SMTP AUTH для нестандартных почтовых клиентов

Следующее, что, вероятно, стоит сделать, – это настроить поддержку альтернативной нотации в SMTP-диалоге для «устаревших» почтовых клиентов, которые, однако, могут использовать SMTP AUTH.

«Устаревшие» почтовые клиенты не распознают SMTP AUTH, когда она предлагается так, как указано в RFC 2222, но они распознают более раннюю нотацию, использованную в черновом варианте этого стандарта, где между командой AUTH и названиями механизмов стоял не пробел, а знак равенства (=).

К «устаревшим» относятся старые версии Microsoft Outlook, Microsoft Outlook Express, и Netscape Mail.

Для поддержки таких почтовых клиентов установите параметр `broken_sasl_auth_clients` в файле `main.cf`:

```
broken_sasl_auth_clients = yes
```

После перезагрузки Postfix вы увидите в SMTP-диалоге еще одну строку AUTH, содержащую знак равенства, например:

```
# telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
```

```
250-mail.example.com
250-PIPELINING
250-SIZE 51200000
250-VRFY
250-ETRN
250-AUTH LOGIN PLAIN
250-AUTH=LOGIN PLAIN
250 8BITMIME
QUIT
```

Настройка области SASL

Возможно, вам нужно будет указать в файле `main.cf` область аутентификации (realm), которая будет отправляться используемой службе проверки паролей Cyrus SASL (такая необходимость определяется версией Cyrus SASL и выбором службы). Для аутентификации клиента сервер Postfix отправляет Cyrus SASL эту область вместе с верительными данными клиента. В Postfix вы можете задать область аутентификации в параметре `smtpd_sasl_local_domain` в файле `main.cf`; по умолчанию этот параметр пуст и должен оставаться пустым, если только вы не используете вспомогательный плагин, которому действительно требуется область аутентификации:

```
smtpd_sasl_local_domain =
```

Устанавливаем параметр в соответствии с используемой службой проверки паролей:

auxprop

Службы, использующие вспомогательные свойства, ожидают, что будет указана область аутентификации. Укажите в параметре `smtpd_sasl_local_domain` ту область аутентификации, которую имеют в хранилище аутентификационных данных ваши пользователи SMTP AUTH. Например, если ваши пользователи SMTP AUTH в `/etc/sasldb2` имеют область `example.com`, используйте такую настройку:

```
smtpd_sasl_local_domain = example.com
```

saslauthd для Cyrus SASL до версии 2.1.17

saslauthd до версии Cyrus SASL 2.1.17 не может иметь дело с областями, так что вам не следует их использовать. Удаляем значение `smtpd_sasl_local_domain` следующим образом:

```
smtpd_sasl_local_domain =
```

saslauthd для Cyrus SASL 2.1.17

saslauthd для версии Cyrus SASL 2.1.17 не знает, что делать с областью, так что просто игнорирует такую информацию. Не имеет значения, имя какой области будет указано, так что просто не трогайте параметр `smtpd_sasl_local_domain`.

saslauthd для Cyrus SASL 2.1.19

saslauthd для версий Cyrus SASL 2.1.19 и выше может поддерживать работу с областью. Используйте параметр `-r` при запуске saslauthd, если вы хотите передать имя области выбранной службе проверки паролей.

Настройка разрешений на пересылку

Выполняем последнюю операцию – разрешаем пересылку для клиентов, прошедших аутентификацию SASL. Для этого добавляем параметр `permit_sasl_authenticated` в список ограничений `smtpd_recipient_restrictions` своей конфигурации, например так:

```
smtpd_recipient_restrictions =
   [...]
   permit_sasl_authenticated,
   permit_mynetworks,
   reject_unauth_destination
   [...]
```

Необходимо поместить ключевое слово `permit_sasl_authenticated` достаточно близко к началу списка ограничений, чтобы аутентифицированный клиент не был случайно отвергнут из-за несоответствия какому-то другому правилу (главным образом речь идет о `reject_unauth_destination`).

Базовая настройка SMTP AUTH на сервере завершена. Перезагружаем конфигурацию и приступаем к тестированию.

Тестирование SMTP AUTH на стороне сервера

Тестирование SMTP-аутентификации на стороне сервера включает в себя следующие этапы:

1. Проверить почтовый журнал, для того чтобы найти ошибки, которые сервер Postfix мог выявить самостоятельно.
2. Проверить SMTP-диалог, чтобы убедиться в том, что `smtpd` предоставляет SMTP AUTH.
3. Аутентифицировать пользователя, чтобы убедиться в том, что сервер Postfix может взаимодействовать с Cyrus SASL.
4. Отправить тестовое сообщение удаленному пользователю, чтобы проверить, могут ли аутентифицированные пользователи пересылать сообщения нелокальным пользователям через наш сервер.

Проверка почтового журнала

Проверяем почтовый журнал, выводя все строки из `/var/log/maillog`, содержащие слова `reject`, `error`, `warning`, `fatal` или `panic`, за которыми следует двоеточие (`:`), при помощи такой команды:

```
# egrep '(reject|error|warning|fatal|panic):' /var/log/maillog
```

Вы не должны увидеть никаких ошибок, связанных с SMTP AUTH, если же вдруг они встретятся, проверьте свою конфигурацию на опечатки и на работоспособность Cyrus SASL.

Теперь включим для демона `smtpd` расширенное журналирование и оставим такой уровень журналирования на все время тестирования SMTP AUTH. Для этого добавим `-v` к команде `smtpd` в файле `master.cf`:

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
# ==========================================================================
smtp      inet  n       -       n       -       -       smtpd -v
#smtps    inet  n       -       n       -       -       smtpd
```

Изменения вступят в силу после перезагрузки Postfix.

Проверка SMTP-диалога

Убедимся в том, что сервер Postfix предоставляет SMTP AUTH почтовым клиентам, так что клиенты знают, когда они могут инициировать SMTP-аутентификацию. Подключитесь к своему серверу и отправьте ему приветствие `EHLO` (SMTP AUTH работает только в расширенном протоколе SMTP), например:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH NTLM LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-AUTH=NTLM LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
QUIT
221 Bye
```

Как видите, параметр `AUTH` не только сообщает вам о включении SMTP AUTH, но также предлагает перечень поддерживаемых механизмов аутентификации. Следом идет почти такая же строка для «устаревших» клиентов.

Если в SMTP-диалоге сервер не предлагает клиенту параметр `AUTH`, проверьте, выполнили ли вы следующие действия:

- Собрали Postfix с поддержкой Cyrus SASL.
- Корректно задали основные параметры, не допустили никаких опечаток в файле `main.cf` (используйте `postconf -n` для проверки параметров).
- Подключились к нужному серверу и использовали `EHLO`, а не `HELO`.

Аутентификация пользователя

Для аутентификации пользователя вам понадобится строка в кодировке base64, содержащая действительные имя и пароль из вашего хранилища аутентификационных данных. Например, для пользователя с именем `test` и паролем `testpass` используйте такую команду:

```
$ perl -MMIME::Base64 -e 'print encode_base64("test\0test\0testpass");'
```

Результатом ее выполнения будут такие данные:

```
dGVzdAB0ZXN0AHRlc3RwYXNz
```

Теперь подключаемся к серверу и с помощью команды `EHLO` устанавливаем расширенное SMTP-соединение, затем используем `AUTH PLAIN`, чтобы сообщить серверу Postfix о том, что мы хотим аутентифицироваться при помощи механизма открытого текста и передать строку в кодировке base64. Приведем пример успешного теста:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH NTLM LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-AUTH=NTLM LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
AUTH PLAIN dGVzdAB0ZXN0AHRlc3RwYXNz
235 Authentication successful
QUIT
221 Bye
```

Мы видим подтверждение успешной аутентификации – строку `235 Authentication successful`. Если возникают проблемы и сервер выводит строку `535 Error: authentication failed`, выполните следующие действия:

- Проверьте журнальный файл на наличие ошибок.
- Проверьте имя и пароль в хранилище аутентификационных данных.
- Проверьте конфигурацию Cyrus SASL в файле `/usr/lib/sasl2/smtpd.conf` (см. главу 15).
- Если будете вносить изменения в `/usr/lib/sasl2/smtpd.conf`, перезагрузите Postfix.
- Раскодируйте свою строку в кодировке base64 и сравните полученный результат с исходными данными (если там есть нулевые байты, перенаправьте вывод в файл и откройте его в текстовом редакторе), например:

```
$ perl -MMIME::Base64 -e 'print decode_base64("dGVzdAB0ZXN0AHRlc3RwYXNz");'
testtesttestpass
```

Предупреждение

Если вы отправляете свои журналы в списки рассылки, то, вероятно, следует изменять их, убирая имена пользователей и пароли, которые присутствуют в журнале в режиме расширенного журналирования. Кроме того, можно создать тестового пользователя, который будет удален по окончании тестирования.

Пересылка тестового сообщения

Наконец, нужно проверить, что сервер Postfix разрешает аутентифицированному пользователю пересылать сообщения. Однако для начала следует убедиться в том, что другие разрешения на пересылку не мешают вашим новым правилам, связанным с аутентификацией. Для этого подключайтесь к серверу с хоста или из сети, которым *не* разрешена пересылка *без* SMTP-аутентификации. Имеет смысл перепроверить все дважды, так что сначала попытайтесь отправить сообщение без использования SMTP AUTH.

Если у вас нет доступа к клиенту вне сетей, определенных в параметре `mynetworks`, то отключите параметр `mynetworks` и задайте на время тестирования `mynetworks_style = host`. В этом случае разрешения на пересылку будут ограничены лишь сервером, так что вы сможете использовать для тестирования сервера любой хост своей локальной сети.

Подключаемся к серверу, как и в предыдущем разделе, но *не* завершаем соединение после успешной аутентификации, а отправляем сообщение нелокальному пользователю. Рассмотрим пример отправки сообщения от `john.doe@example.com` адресату `echo@postfix-book.com`:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-AUTH DIGEST-MD5 CRAM-MD5 GSSAPI PLAIN LOGIN
250-AUTH=DIGEST-MD5 CRAM-MD5 GSSAPI PLAIN LOGIN
250-XVERP
250 8BITMIME
AUTH PLAIN dGVzdAB0ZXN0AHRlc3RwYXNz
235 Authentication successful
MAIL FROM: <john.doe@example.com>
250 Ok
RCPT TO: <echo@postfix-book.com>
250 Ok
DATA
```

```
354 End data with <CR><LF>.<CR><LF>
This is a server side SMTP AUTH test. If the mail is
accepted, relaying works.
.
250 Ok: queued as 3FF15E1C65
QUIT
221 Bye
Connection closed by foreign host.
```

Сервер отправляет в качестве ответа команде `RCPT TO:` сообщение `250 Ok`, что обычно является хорошим знаком, но вы все еще хотите подтверждения того, что письмо прошло «сквозь» сервер. Если этот тест не проходит, выполните следующие действия:

- Проверьте журнальный файл на предмет ошибок.
- Убедитесь в том, что вы корректно задали `permit_sasl_authenticated` (см. раздел «Настройка разрешений на пересылку» ранее в этой главе).
- Дважды перепроверьте свою строку в кодировке base64.
- Проверьте, нет ли опечаток в SMTP-диалоге.

Расширенная настройка сервера

С момента появления в Postfix поддержки SMTP AUTH появились новые параметры, которые обеспечивают более широкие возможности управления SMTP AUTH. В следующих подразделах вы узнаете о том, что Postfix умеет делать сейчас.

Выборочное предоставление SMTP AUTH

Вы можете сделать так, чтобы для определенных сетей Postfix не предлагал SMTP AUTH. Это чрезвычайно полезно при работе с почтовыми клиентами Netscape (Netscape 4.*x*), которые настаивают на использовании SMTP AUTH, как только она им предлагается, вне зависимости от того, были ли они настроены для использования аутентификации.

Установите параметр `smtpd_sasl_exceptions_networks` в файле `main.cf`, задавая его значение как список переменных, известных серверу Postfix из его собственной конфигурации (например, `mynetworks`), и (или) список подсетей в нотации CIDR (например, `172.16.0.117/32`):

```
smtpd_sasl_exceptions_networks = $mynetworks, 172.16.0.117/32
```

Соответствие имени пользователя SMTP AUTH имени отправителя конверта

Как только почтовый клиент аутентифицирован, он может отправлять сообщения, указывая любого отправителя конверта на свой выбор. Однако вы можете ввести ограничение, заставив почтовые клиенты использовать определенный адрес отправителя. Параметр `smtpd_sender_`

login_maps определяет путь к карте, которая сопоставляет адреса отправителей конвертов зарегистрированным именам SASL. Такая карта, например /etc/postfix/smtpd_sender_login_map, выглядит так:

```
flintstone@example.com     flintstone
rubble@example.com         rubble
sales@example.com          flintstone, rubble
```

В левой части карты приведены отправители конвертов, а в правой – одно или несколько (в виде списка, разделенного запятыми) зарегистрированных имен пользователей. Преобразуем карту командой postmap, например postmap hash:/etc/postfix/smtpd_sender_login_map, и в файле main.cf сообщаем серверу Postfix о том, что он должен использовать ее.

```
smtpd_sender_login_maps = hash:/etc/postfix/smtpd_sender_login_map
```

Вместо карты типа hash вы также могли бы использовать NIS-, LDAP- и SQL-запросы.

После того как вы сообщили серверу Postfix о наличии карты, следует выбрать одно из двух ограничений, чтобы определить, как поступать с клиентами, у которых имя отправителя конверта не совпадает с их регистрационным именем.

`reject_sender_login_mismatch`

Ограничение, заданное в карте, будет накладываться на все клиенты вне зависимости от того, используют ли они SMTP-аутентификацию.

`reject_unauthenticated_sender_login_mismatch`

Ограничение, заданное в карте, будет накладываться только на те клиенты, которые не прошли SMTP-аутентификацию.

Один из двух параметров указывается в списке ограничений smtpd_recipient_restrictions в файле main.cf:

```
smtpd_recipient_restrictions =
    ...
    reject_unauthenticated_sender_login_mismatch
    ...
```

SMTP–аутентификация на стороне клиента

SMTP-аутентификация на стороне клиента подразумевает использование Postfix-демонами smtp и lmtp библиотек Cyrus SASL для своей аутентификации на удаленном сервере. В клиентской конфигурации вам необходимо настроить Postfix, но *не* нужно беспокоиться о настройке Cyrus SASL. Оба демона, smtp и lmtp, могут использовать любой механизм, поддерживаемый библиотекой Cyrus SASL.

На рис. 16.2 изображено участие Postfix-демона smtp в сеансе SMTP AUTH с удаленным почтовым сервером. Клиент (smtp) отправляет верительные данные, хранящиеся в файле паролей SASL, для получения разрешения на пересылку от удаленного сервера.

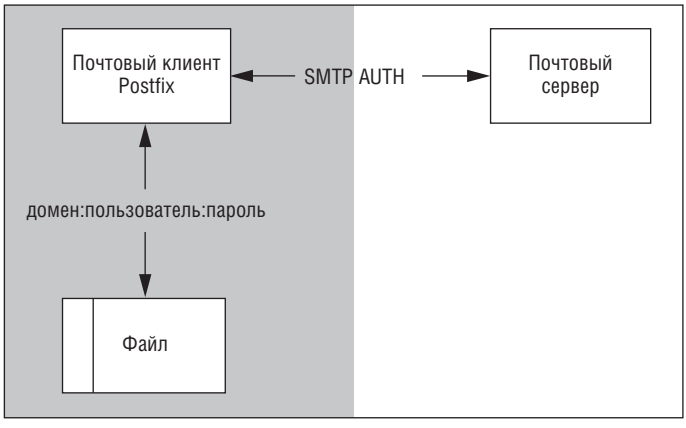

Рис. 16.2. *Архитектура SMTP AUTH на стороне клиента*

SMTP-аутентификация для SMTP-клиента Postfix

Настройка SMTP-аутентификации для клиента Postfix выполняется *гораздо* проще, чем для сервера. Хотя вам все еще будет требоваться библиотека Cyrus SASL, конфигурировать ее не придется.

Вот какие действия вам следует выполнить:

1. Запросить у удаленного сервера список поддерживаемых механизмов аутентификации.
2. Включить клиентскую часть SMTP AUTH.
3. Предоставить файл для хранения верительных данных SMTP AUTH.
4. Настроить Postfix на работу с файлом верительных данных.
5. Отключить ненадежные механизмы аутентификации.

Проверка на соответствие механизмов аутентификации

Сначала нужно определить, какие механизмы аутентификации предлагает удаленный сервер, и убедиться в том, что установленная у вас версия Cyrus SASL содержит библиотеки для поддержки этих механизмов. Подключитесь к своему почтовому серверу и отправьте приветствие `EHLO`, чтобы получить список механизмов, например:

```
$ telnet mail.remote-example.com 25
220 mail.remote-example.com ESMTP
EHLO mail.example.com
250-mail.remote-example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
```

```
250-AUTH LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-AUTH=LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
QUIT
221 Bye
```

Как видите, сервер поддерживает не только механизмы LOGIN, PLAIN, DIGEST-MD5 и CRAM-MD5, но и аутентификацию для «устаревших» клиентов, описанных ранее в разделе «Настройка SMTP AUTH для нестандартных почтовых клиентов».

Теперь выведем список библиотек в библиотечном каталоге Cyrus SASL. Например, если библиотеки Cyrus хранятся в каталоге `/usr/local`, выполните такую команду:

```
# ls -1 /usr/local/lib/sasl2/lib*.so
```

Выходные данные будут выглядеть примерно так:

```
/usr/local/lib/sasl2/libanonymous.so
/usr/local/lib/sasl2/libcrammd5.so
/usr/local/lib/sasl2/libdigestmd5.so
/usr/local/lib/sasl2/liblogin.so
/usr/local/lib/sasl2/libplain.so
/usr/local/lib/sasl2/libsasldb.so
```

Очевидно, что в данном случае поддерживаются механизмы ANONYMOUS, CRAMMD5, DIGEST-MD5, LOGIN и PLAIN (имейте в виду, что `libsasldb.so` – это не библиотека механизма аутентификации).

Собрав все эти сведения, сравните механизмы, предлагаемые удаленным сервером, и механизмы, поддерживаемые библиотеками Cyrus SASL вашего сервера, и вы узнаете, какие механизмы сможет применять ваш клиент Postfix для подключения к данному удаленному серверу.

Включение клиентской части SMTP AUTH

По умолчанию SMTP-аутентификация на стороне клиента выключена. Для ее включения необходимо установить параметр `smtp_sasl_auth_enable` в файле `main.cf` в значение `yes`:

```
smtp_sasl_auth_enable = yes
```

Аутентификация на стороне клиента включена, но вам еще нужно будет сообщить серверу Postfix, где следует искать секретные данные, необходимые для аутентификации, и какой из механизмов (из предлагаемых удаленным сервером) Postfix может использовать.

Хранение верительных данных SMTP AUTH

Теперь подготовим данные, которые клиент Postfix будет использовать для того, чтобы аутентифицировать себя на одном или несколь-

ких удаленных серверах. Создаем от имени root файл карты /etc/postfix/sasl_passwd (если он еще не существует):

```
# touch /etc/postfix/sasl_passwd
```

Совет

Postfix всегда открывает карты до выполнения chroot, так что это таблица может безопасно храниться за пределами области chroot jail.

Отредактируйте этот файл, поместив полностью определенное доменное имя почтового сервера, который требует аутентификации, с левой стороны, а разделенную двоеточием пару «имя пользователя – пароль» – с правой. Заведем, например, имена пользователей и пароли для mail.example.com и relay.another.example.com:

```
mail.example.com           test:testpass
relay.another.example.com  username:password
```

После редактирования файла sasl_passwd измените права на него так, чтобы читать его мог только пользователь root (в файле хранится конфиденциальная информация, которая не должна быть доступна локальным пользователям). Используйте для этого команды chown и chmod:

```
# chown root:root /etc/postfix/sasl_passwd && chmod 600 /etc/postfix/
sasl_passwd
```

Примечание

Не беспокойтесь о правах доступа для Postfix – он читает файл sasl_passwd *до* переключения на пользователя с меньшими правами и *до* входа в область chroot jail.

Когда права доступа будут заданы корректно, преобразуйте файл карты в индексированную карту для быстрого доступа Postfix (это надо делать при каждом изменении sasl_passwd):

```
# postmap hash:/etc/postfix/sasl_passwd
```

Настройка Postfix для использования верительных данных SMTP AUTH

Теперь надо сообщить клиенту Postfix, где находится только что созданная карта верительных данных аутентификации. Записываем в параметре smtp_sasl_password_maps в файле main.cf полный путь к файлу sasl_passwd, указывая при этом (с помощью спецификатора hash:), что значения карты хранятся в хеш-файле, например:

```
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
```

Отключение некоторых механизмов аутентификации

Выполняем последний шаг в нашей настройке клиента – отключаем использование ненадежных механизмов. Указываем в параметре smtp_

sasl_security_options разделенный запятыми список типов механизмов, которые клиент не может использовать (перечень существующих типов приведен выше в разделе «Настройка механизмов SASL»). По умолчанию параметр smtp_sasl_security_options установлен в значение noanonymous, но по возможности (если ваш сервер поддерживает механизм с шифрованием, такой как DIGEST-MD5 или CRAM-MD5) вам стоит также отключить использование механизмов открытого текста. Для этого добавьте в свой файл main.cf следующую строку:

```
smtp_sasl_security_options = noanonymous, noplaintext
```

Совет

Если удаленный сервер предлагает только механизмы открытого текста, а вы не хотите использовать их на незашифрованном соединении, то проверьте, поддерживает ли сервер механизм STARTTLS. Если да, то вы можете заставить Postfix использовать TLS (см. главу 18), так что клиент будет отправлять верительные данные открытым текстом только после открытия соединения с шифрованием.

Тестирование SMTP AUTH на стороне клиента

Проверка аутентификации вашего клиента требует как локального, так и удаленного тестирования:

1. Проверьте верительные данные на удаленном сервере, чтобы убедиться в том, что они действительны и известны удаленному серверу.
2. Проверьте файл журнала.
3. Используйте Postfix для отправки тестового сообщения удаленному пользователю – в случае успеха это будет означать, что вы можете пересылать сообщения через сервер.

Проверка верительных данных на удаленном сервере

Сначала убедимся в том, что ваши имя и пароль действительно работают. Подключитесь к удаленному серверу, как было описано ранее в разделе «Аутентификация пользователя», и пройдите аутентификацию, используя имеющееся имя и пароль.

Примечание

Если удаленный сервер не предлагает механизмы открытого текста по незашифрованному соединению, то вы можете попытаться использовать s_client OpenSSL для открытия сеанса TLS: посмотрите, поддерживает ли он механизмы открытого текста, затем попробуйте пройти аутентификацию (см. подробное описание в главе 18).

Вы также можете попытаться задать верительные данные в почтовом клиенте, работающем в графической оболочке, который поддерживает разнообразные механизмы аутентификации. Отправьте сообщение посредством такого клиента и посмотрите, примет ли сервер верительные данные.

Когда вы будете уверены в том, что сервер принимает ваши имя пользователя и пароль, вы можете приступить к настройке Postfix.

Проверка файла журнала

Затем ищем очевидные ошибки в журнале Postfix, пользуясь уже хорошо знакомой нам командой `egrep`:

```
# egrep '(reject|error|warning|fatal|panic):' /var/log/maillog
```

Использование клиента Postfix для отправки тестового сообщения удаленному пользователю

Последняя проверка состоит в отправке сообщения удаленному получателю с использованием демона почтового клиента Postfix (`smtp`). Выполните следующие действия:

1. Увеличьте уровень журналирования для демона `smtp`.
2. Отправьте сообщение удаленному пользователю.
3. Найдите в журнале подтверждение успешной аутентификации.

Увеличение подробности журналирования для демона smtp

Для увеличения объема выводимых в журнал сведений о демоне `smtp` отредактируйте файл `master.cf` и укажите для программы `smtp` аргумент `-v` (будьте внимательны и не перепутайте строку `smtpd` со строкой `smtp`):

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
# ==========================================================================
smtp      inet  n       -       n       -       -       smtpd
#smtps    inet  n       -       n       -       -       smtpd
...
smtp      unix  -       -       n       -       -       smtp -v
...
```

После перезагрузки конфигурации Postfix демон `smtp` будет записывать в журнал данные об SMTP AUTH *и* о действиях с верительными данными, например кодировании или декодировании строки в base64.

Предупреждение

Вероятно, вы захотите проявить осторожность при отправке своего журнала Postfix в список рассылки, т. к. `smtp` записывает верительные данные в журнал в виде открытого текста. Позаботьтесь о том, чтобы перед отправкой журнала заменить имя пользователя и пароль, например, на `XXX`.

Отправка тестового сообщения удаленному пользователю

Вы можете проверить, работает ли аутентификация на стороне клиента, используя почтовый клиент с графическим интерфейсом пользователя для передачи сообщения вашему демону Postfix. Можно также

использовать командную строку для передачи сообщения получателю echo@postfix-book.com, который отправляет обратно отправителю конверта сообщение, содержащее полный заголовок и тело исходного сообщения:

```
$ mail -s 'Testing client side authentication' echo@postfix-book.com
Testing...
.
Cc:
$
```

Проверка успешности аутентификации по файлу журнала

Наконец, проверяем наличие в журнале записи об успешной аутентификации, используя команду grep:

```
# grep '235 Authentication successful' /var/log/maillog
```

Если все прошло удачно, вы должны увидеть одну или несколько строк, подобных следующей, в которых говорится об успешной аутентификации на relay.example.com:

```
Jan 20 12:40:39 mail postfix/smtp[21740]: < relay.example.com[172.16.0.100]:
  235 Authentication successful
```

Примечание

Кроме того, новое сообщение должно появиться в почтовом ящике получателя; проверьте заголовки этого сообщения.

Lmtp-клиент

Настройка lmtp-клиента Postfix для использования SMTP AUTH очень похожа на настройку smtp-клиента (см. ранее в этой главе раздел «SMTP-аутентификация для SMTP-клиента Postfix»). Необходимо выполнить следующие операции:

1. Включить SMTP AUTH на стороне клиента, задав в файле main.cf параметр lmtp_sasl_auth_enable = yes.
2. Создать файл, хранящий верительные данные SMTP AUTH. Для этого обратитесь к разделу «Хранение верительных данных SMTP AUTH», только вместо имени файла smtp_passwd используйте lmtp_passwd (конечно, вы должны использовать файл верительных данных совместно с smtp-клиентом).
3. Настроить Postfix для использования этого файла с верительными данными SASL; установите lmtp_sasl_password_maps = hash:/etc/postfix/lmtp_passwd в файле main.cf.
4. Разрешить клиенту использовать только безопасные механизмы аутентификации (см. следующий раздел).

Отключение некоторых механизмов для lmtp-клиента

Процесс запрещения использования ненадежных механизмов для демона `lmtp` повторяет процесс, описанный для демона `smtp` в разделе «Отключение некоторых механизмов аутентификации». Например, если вы хотите гарантировать, что клиент не будет использовать механизмы открытого текста, то задайте параметр `lmtp_sasl_security_options` следующим образом:

```
lmtp_sasl_security_options = noplaintext, noanonymous
```

Однако существует одно небольшое отличие от настройки для демона `smtp`: если окажется, что ваш `lmtp`-клиент работает на той же машине, что и ваш сервер Postfix, и они взаимодействуют через сокеты, вы можете несколько смягчить требования и разрешить механизмы открытого текста:

```
lmtp_sasl_security_options = noanonymous
```

Тестирование SMTP AUTH для lmtp-клиента

Для тестирования `lmtp`-клиента выполните те же действия, которые были описаны ранее в разделе «Тестирование SMTP AUTH на стороне клиента». Кроме того, вы можете использовать утилиту `imtest` из пакета Cyrus IMAP. Это особенно полезно, если вам нужно, чтобы `lmtp`-клиент Postfix доставлял сообщения LMTP-серверу, который поставляется с сервером Cyrus IMAP.

17

Протокол TLS

К этому моменту мы убедились, что безопасность Postfix находится на должном уровне с системной точки зрения, т. е. Postfix прилагает значительные усилия для закрытия основных дыр в защите, через которые возможно вторжение в систему. Несмотря на всю важность этих усилий, кое-что остается за кадром. Проблема заключается в том, что протокол SMTP в силу своей архитектуры не способен защитить вас от злоумышленников, перехватывающих ваши сетевые пакеты. Ситуация выглядит безрадостной, и хотя вам может показаться, что опасность несколько преувеличена, в реальности встречаются такие случаи, когда защита от прослушивания трафика SMTP может оказаться весьма полезной.

Предположим, что некая компания, пересылающая информацию по Интернету с помощью почтовых серверов, решила применить шифрование данных – тогда каждый, чей сервер SMTP поддерживает SMTP AUTH с передачей данных открытым текстом, имеет причины для беспокойства. Критичные элементы сообщений и пароли будут передаваться открытым текстом в пакетах TCP, и каждый, кто знает маршрут прохождения данных, может сохранить сетевые пакеты на своем компьютере и извлечь из них передаваемые данные.

Справиться с этой проблемой поможет протокол безопасности транспортного уровня TLS (Transport Layer Security), в котором шифрование соединения между двумя хостами выполняется еще *до того*, как конфиденциальные данные попадают в сеть. Postfix позволяет использовать TLS даже для разрешения пересылки, основанной на сертификатах.

В этой главе изложена теория применения TLS в Postfix для тех случаев, когда последний выступает в качестве почтового клиента, простого почтового сервера или сервера, выдающего разрешение на пересылку на основании клиентских сертификатов. Из этой главы вы также узнаете о том, когда целесообразно применять TLS и что для этого нужно.

Основы TLS

По умолчанию SMTP-сеанс клиента с сервером не шифруется. Клиент просто устанавливает TCP-соединение и начинает передачу данных (рис. 17.1). Если содержимое не было зашифровано другими средствами, оно передается открытым текстом и может быть прочитано каждым, кто сможет перехватить поток данных. Такой нежелательный слушатель может легко увидеть данные, а если у него есть доступ к управлению маршрутизатором, то, возможно, и изменить их.

Подобные атаки становятся невозможны, когда клиент и сервер используют протокол TLS (рис. 17.2), поскольку в такой системе выполняются три условия:

Конфиденциальность

Взаимодействие клиента и сервера происходит в зашифрованном сеансе. Сторонний наблюдатель, не имеющий доступа к клиенту или серверу, не может дешифровать передаваемые данные.

Целостность

Даже если бы атака с внедрением посредника удалась, каждая из сторон немедленно обнаружила бы искажение данных.

Достоверность аутентификации

Клиент и сервер могут обменяться сертификатами, удостоверенными уполномоченным центром сертификации (CA – Certification

***Рис. 17.1.** Связь без шифрования – чтение доступно каждому*

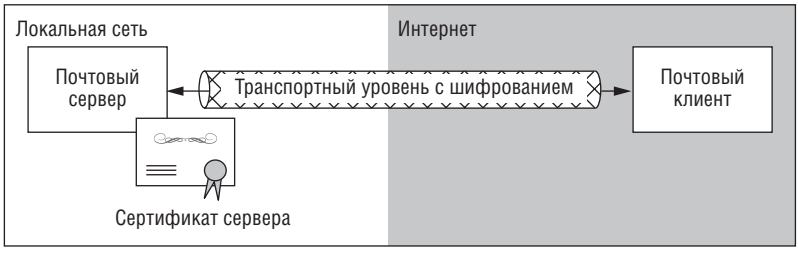

***Рис. 17.2.** Связь с шифрованием – чтение доступно только отправителю и получателю*

Authority), которые гарантируют аутентичность хостов-участников. Сертификат содержит и такую информацию, как полностью определенное доменное имя хоста. Например, одна из возможных атак – подмена DNS – будет обнаружена до отправки данных.

Вопреки распространенным заблуждениям в отношении TLS, имеют место следующие факты:

- TLS не защищает данные после того, как они получены сервером от клиента. Как только сервер принял и сохранил сообщение, оно снова представлено в обычном текстовом виде.
- TLS гарантирует шифрование только на пути от почтового клиента к почтовому серверу. Имейте в виду, что серверу может потребоваться передать сообщение следующему серверу. Часть серверов на пути сообщения к конечному пункту назначения может не поддерживать TLS. Так что даже если вы можете применить шифрование в пределах своего почтового узла, вы, скорее всего, не в силах контролировать процесс передачи после того, как сообщение покинуло пределы вашей организации. Как только на пути встретится сервер, не поддерживающий TLS, сообщение снова будет представлено открытым текстом.
- TLS не обязательно будет защищать сообщение, которое, будучи принятым, позднее возвращается как недоставимое. Нет никаких гарантий, что сообщение будет возвращаться тем же путем.

Если что-то из перечисленного для вас важно, то вам следует шифровать содержимое сообщения до его отправки. В числе популярных средств шифрования почты можно назвать S/MIME, PGP и GnuPG.

Как работает TLS

Протокол TLS выполняет шифрование соединения только между двумя хостами. Сеанс с использованием этого протокола проходит следующим образом:

1. Клиент устанавливает соединение с сервером.
2. Хосты начинают взаимодействие по протоколу SMTP.
3. Сервер с помощью ключевого слова `STARTTLS` предлагает использовать TLS в рамках SMTP-взаимодействия.
4. Если клиент может использовать TLS, он отвечает серверу ключевым словом `STARTTLS`.
5. Открытый сертификат сервера подписывается с помощью секретного ключа и отправляется клиенту.
6. Клиент проверяет подлинность сертификата сервера, сверяя подпись центра сертификации с имеющейся у него в корневом хранилище открытой подписью центра сертификации.
7. Клиент проверяет сертификат сервера на соответствие содержащейся в нем строки `Common Name` доменному имени сервера.

8. Клиент и сервер включают режим шифрования данных.
9. Хосты выполняют обмен данными.
10. Сеанс заканчивается.

Из этого описания видно, что сертификаты играют важную роль в TLS.

Понятие о сертификатах

Технология шифрования не зависит от сертификатов, но они необходимы для того, чтобы гарантировать, что общаться друг с другом будут только те хосты, которые действительно намеревались это сделать. Если каждый из хостов может проверить сертификат другого, то они договариваются о шифровании сеанса. В противном случае они полностью отказываются от шифрования и формируют предупреждение, т. к. отсутствует основа для взаимного доверия – аутентичность.

Как добиться доверия

Когда вы создаете сертификат, ваша система записывает на диск два файла: в одном находится открытый ключ, в другом – секретный ключ. Хост-отправитель шифрует с помощью секретного ключа некие данные, а хост-получатель использует открытый ключ для расшифровки и проверки аутентичности отправителя. Имеется возможность проверить открытый ключ, подписанный центром сертификации, который выступает в качестве гаранта достоверности хоста-отправителя.

Хост-получатель не обращается напрямую к центру сертификации при каждой проверке сертификата. Это не только сильно увеличило бы трафик, но и дало бы лишнюю возможность для манипуляций с верификационными данными, путешествующими по сети. Вместо этого получатель у себя сравнивает подпись центра сертификации в открытом ключе с контрольной суммой проверяемого сертификата. Центр сертификации вычисляет эту сумму и добавляет ее в сертификат отправителя в процессе его подписания. Любые изменения в подписанном сертификате приведут к изменению контрольной суммы и сделают его бесполезным, т. к. механизм TLS немедленно обнаружит подделку.

Чтобы установить доверительные отношения, клиент и сервер должны удовлетворять разным требованиям:

Клиент

Почтовый клиент, проверяющий аутентичность серверного сертификата, должен иметь доступ к открытому ключу центра сертификации. Кто-то должен поместить этот ключ в хранилище сертификатов операционной системы, откуда его смогут получить почтовый клиент и другие приложения.

Сервер

Почтовый сервер, отправляющий сертификат, должен располагать корректными секретными и открытыми ключами. Открытые ключи должны быть подписаны центром сертификации.

Какой центр сертификации вам нужен?

Существуют несколько центров сертификации, каждый из которых с удовольствием подпишет ваш сертификат. Ответ на вопрос о том, какой из них лучше, зависит от предоставляемых ими услуг и цен, а также от причин, побудивших вас использовать сертификат:

Частное использование

Если вы планируете использовать сертификат только для личных целей, когда он нужен только вам или ограниченному числу пользователей (например, сотрудникам вашей компании или серверам контролируемой вами территориально распределенной сети), то вы можете сами для себя стать центром сертификации. Подпишите серверные сертификаты сами и раздайте корневой и серверный сертификаты своим клиентам и серверам. Так вы сэкономите свои усилия и деньги, но клиенты и серверы за пределами вашей организации не будут доверять вашему сертификату.

Официальное использование

Если вам надо поддерживать официальные контакты с внешними пользователями и почтовыми серверами, которых вы не можете контролировать, вам придется прибегнуть к услугам официального центра сертификации. Можете начать поиск со страницы PKI (*http://www.pki-page.org*), где вы найдете обширный список центров сертификации, расположенных по всему миру.

Создание сертификатов

Планируете ли вы подписывать сертификаты в официальном центре сертификации или собираетесь организовать собственный центр, в любом случае вам необходимо сначала создать заявку на сертификат.

Создать новый сертификат просто – достаточно запустить сценарий и выполнить несколько команд, которые проделают почти всю работу. Все, что вам нужно, – иметь под рукой некоторые данные, когда вы запускаете сценарий.

Необходимые данные

Большинство перечисленных ниже параметров говорят сами за себя, у вас не возникнет трудностей с их получением. Есть, однако, место, где требуется некоторая осторожность: `Common Name`. Значение, задаваемое в серверном сертификате, должно совпадать с именем вашего хос-

та в DNS. В противном случае протокол TLS заподозрит атаку типа «посредник» (man-in-the-middle) с похищением сертификата и немедленно прекратит проверку.

В сертификате клиента здесь обычно указывается его собственное имя.

Вот информация, которая вам понадобится:

- Страна (Country)
- Область (State or province)
- Город (City or other municipal area)
- Организация (Organization)
- Подразделение (Organization unit)
- Общепринятое имя (Common name)
- Электронный адрес (Email address)

Создание CA-сертификата

Если вы решили самостоятельно выпускать сертификаты, то первым делом вам надо создать сертификат вашего собственного центра сертификации. (Если вы пользуетесь услугами официального центра сертификации, можете пропустить весь текст до раздела «Распространение и установка CA-сертификата».) Запустите программу `misc/CA.pl -newca`, входящую в OpenSSL:

```
# ./CA.pl -newca
CA certificate filename (or enter to create)
Making CA certificate ...
Generating a 1024 bit RSA private key
......++++++
....++++++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase: ❶
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:EX
State or Province Name (full name) [Some-State]:Examplia
Locality Name (eg, city) []:Exampleton
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Example Inc.
Organizational Unit Name (eg, section) []:Certification Authority
Common Name (eg, YOUR name) []:mail.example.com
Email Address []:postmaster@example.com
```

❶ При генерации CA-сертификата вы должны ввести идентификационную фразу. Это необходимо будет делать каждый раз, когда вы в качестве центра сертификации будете подписывать или отзывать сертификаты.

Программа `misc/CA.pl` создает подкаталоги, в которые помещает файлы и каталоги, необходимые для работы центра сертификации. После выполнения программы у вас должен появиться новый подкаталог `misc/demoCA` такого вида:

```
# tree demoCA/
demoCA/
|-- cacert.pem ❶
|-- certs
|-- crl
|-- index.txt
|-- newcerts
|-- private
|   `-- cakey.pem ❷
`-- serial
```

❶ `cacert.pem` — это открытый ключ центра сертификации. Он должен находиться в корневых хранилищах сертификатов ваших хостов, чтобы они могли проверять подпись в открытом сертификате Postfix.

❷ `cakey.pem` — это секретный ключ центра сертификации. Он должен быть хорошо защищен, доступ к нему на чтение и запись должен иметь только администратор центра сертификации.

Распространение и установка CA-сертификата

На следующем шаге надо передать CA-сертификат всем клиентам, которые будут использовать TLS. Если вы используете собственный центр сертификации, то его сертификат находится в каталоге `misc/demoCA`. По умолчанию openssl сохранит его с именем `cacert.pem`. Если же вы пользуетесь услугами официального центра сертификации, найдите и скачайте его CA-сертификат.

Способ распространения CA-сертификата зависит главным образом от того, в каких приложениях он будет использоваться и каким будет их окружение. GUI-приложения обычно обращаются к корневому хранилищу сертификатов, предоставляемому операционной системой, которая обеспечивает централизованное управление всеми сертификатами.

На использующих OpenSSL серверах, не имеющих другого пользовательского интерфейса, кроме командной строки, нет *единого* централизованного корневого хранилища. Приложения командной строки могут использовать собственные хранилища, расположение которых должно быть указано в их индивидуальных настройках. Так как Postfix является именно таким приложением, вам надо настроить демон `smtp`, либо демон `smtpd`, либо их оба так, чтобы они получили доступ к хранилищу CA-сертификатов. Этот этап настройки будет рассмотрен в главе 18.

Примечание

Отдельные хранилища позволяют добиться большей гибкости при разработке специализированных решений, но они также усложняют поддержание их актуальности.

Установка в Windows

ОС Windows предпочитает другой формат сертификатов, и OpenSSL может выполнить это преобразование CA-сертификата. Вместо программы `CA.pl` надо использовать непосредственно `openssl`. В следующем примере SSL находится в каталоге `/usr/local/ssl`:

```
# cd /usr/local/ssl/misc/demoCA
# openssl x509 -in cacert.pem -out cacert.der -outform DER
```

После выполнения этих команд у вас в каталоге `/usr/local/ssl/misc/demoCA` появится новый файл `cacert.der`.

Установить CA-сертификат в Windows очень просто:

1. Скопируйте файл `cacert.der` на Windows-машину.
2. Двойным щелчком на `cacert.der` запустите процесс установки (рис. 17.3).

***Рис. 17.3.** Успешно установленный CA-сертификат в Windows*

3. Щелкните на кнопке Install Certificate и следуйте указаниям мастера Certificate Import Wizard.

4. Ответьте утвердительно на вопрос, следует ли добавить сертификат в корневое хранилище.
5. После окончания установки еще раз дважды щелкните на файле `cacert.der`, чтобы убедиться в том, что он уже установлен.

Установка в Linux (KDE–3.1.x)

Установить CA-сертификат для Linux/KDE так же просто, как и для Windows:

1. Скопируйте файл `cacert.der` на Linux-машину.
2. В броузере Konqueror двойным щелчком на `cacert.der` запустите процесс установки.
3. KDE запустит программу KDE Secure Certificate Import.
4. Выберите Import.
5. Появится диалоговое окно с подтверждением того, что импорт прошел успешно.
6. Запустите KDE Control Center и в левой панели выберите Security & Privacy.
7. Выберите Crypto в левой панели.
8. Щелкните на вкладке SSL Signers в правой панели.
9. Убедитесь в наличии нового сертификата (рис. 17.4).

Рис. 17.4. *Успешно установленный CA-сертификат в Linux*

Установка в Mac OS X

Процедура установки сертификата различается в разных версиях Mac OS X. Здесь приведена смесь операций, выполняемых в командной строке и в графической оболочке, которая должна работать во всех версиях (OS X до 10.3 и более поздних).

Для установки сертификата вручную выполните следующие действия:

1. Скопируйте файл `cacert.pem` в свой домашний каталог.
2. Откройте терминальное окно.
3. Импортируйте `cacert.pem` в хранилище `keychain`:

```
$ sudo certtool i cacert.pem k=/System/Library/Keychains/X509Anchors
...certificate successfully imported
```

Чтобы удостовериться в успешном импорте сертификата, проделайте следующее:

1. В графической оболочке откройте `keychain` в каталоге `/Applications/Utilities/`. Вы увидите только свое локальное хранилище.
2. В меню File выберите Add Keychain и добавьте в свое хранилище `/System/Library/Keychains/X509Anchors`.
3. Найдите ваш CA-сертификат в списке X509Anchors. В столбце Name вы должны увидеть указанное вами значение Common Name (рис. 17.5).

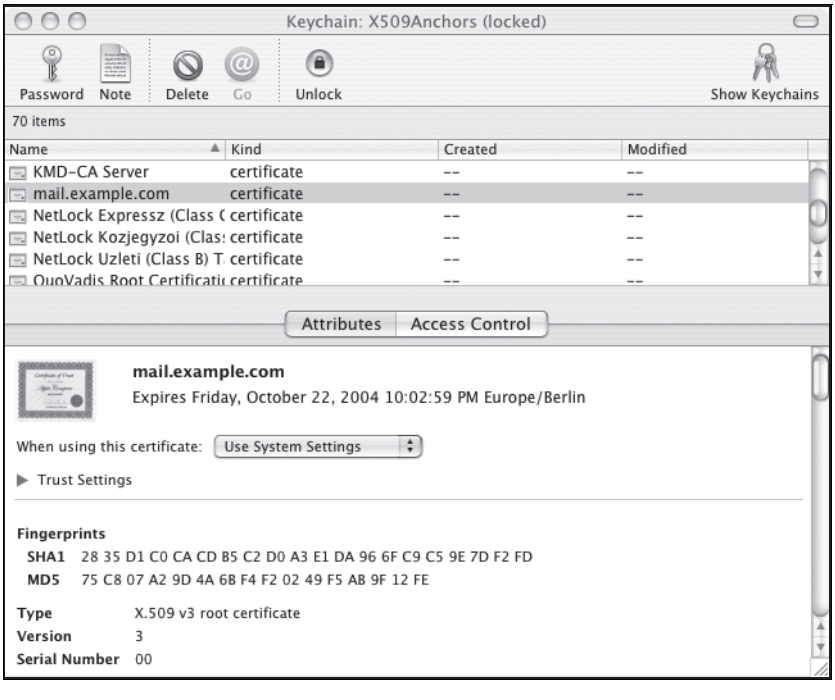

***Рис. 17.5.** Успешно установленный CA-сертификат в Mac OS X*

Создание сертификата сервера

После того как установлен CA-сертификат, пришло время создать сертификат сервера Postfix. Перейдите в каталог /usr/local/ssl/misc и выполните следующую команду. Она создаст заявку на сертификат, который в следующем разделе «Подписание сертификата сервера» будет подписан центром сертификации:

```
# openssl req -new -nodes -keyout postfix_private_key.pem -out
  postfix_private_key.pem -days 365[1]
Generating a 1024 bit RSA private key
.....................++++++
..........++++++
writing new private key to 'postfix_private_key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:EX
State or Province Name (full name) [Some-State]:Examplia
Locality Name (eg, city) []:Exampleton
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Example Inc.
Organizational Unit Name (eg, section) []:MX Services
Common Name (eg, YOUR name) []:mail.example.com
Email Address []:postmaster@example.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
#
```

Предупреждение

Не используйте программу misc/CA.pl для создания файла с секретным ключом, если только вы не изменили ее так, чтобы она не запрашивала идентификационную фразу, сохраняемую в ключе. Наличие такой фразы означает, что каждый раз, когда Postfix использует сертификат, кто-то должен эту фразу ввести! Postfix загружает сертификат при каждом запуске или перезапуске демонов smtp и smtpd, и если в нем предусмотрен пароль, пользователю придется вводить его. Излишне говорить, что невозможность автоматического рестарта для такого сервера, как Postfix, совершенно неприемлема.

[1] Надо помнить, что в примерах время жизни пароля часто задается равным 365 дням. Если менять ключи ежегодно вам не по душе, стоит увеличить этот срок до 3650 дней – за 10 лет точно что-то устареет: либо Postfix, либо алгоритм шифрования, либо еще что-то. – *Примеч. науч. ред.*

Подписание сертификата сервера

Последний этап создания серверного сертификата заключается в подписании его центром сертификации. Если вы пользуетесь услугами официального центра, следуйте его инструкциям. В противном случае запустите `openssl` из командной строки, чтобы создать файл `postfix_public_cert.pem` на основе `postfix_private_key.pem`.

```
# openssl ca -policy policy_anything -out postfix_public_cert.pem -infiles
  postfix_private_key.pem
Using configuration from /usr/local/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 1 (0x1)
        Validity
            Not Before: Nov  9 21:25:13 2003 GMT
            Not After : Nov  8 21:25:13 2004 GMT
        Subject:
            countryName               = EX
            stateOrProvinceName       = Examplia
            localityName              = Exampleton
            organizationName          = Example Inc.
            organizationalUnitName    = MX Services
            commonName                = mail.example.com
            emailAddress              = postmaster@example.com
        X509v3 extensions:
            X509v3 Basic Constraints:
            CA:FALSE
            Netscape Comment:
            OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
            9E:36:9D:9B:ED:4E:32:73:0E:86:55:2A:FF:1B:49:F9:1C:47:17:75
            X509v3 Authority Key Identifier:
           keyid:00:52:AD:B7:FA:C2:EF:01:1A:9E:7B:0F:57:DB:DC:E4:82:59:8D:0B
           DirName:/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority
Example Inc./CN=mail.example.com/emailAddress=postmaster@example.com
            serial:00
Certificate is to be certified until Nov  8 21:25:13 2004 GMT (365 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Файл `postfix_public_cert.pem` — это сертификат, который будет высылаться клиентам на начальной стадии установления TLS-соединения. Вместе с этим сертификатом Postfix будет также высылать подпись из файла `postfix_private_key.pem`. Хост-получатель при проверке сертификата `postfix_public_cert.pem` выполнит определенные вычисления на основании подписи, сформированной Postfix с использованием секрет-

ного ключа, и подписи в CA-сертификате. Результат должен соответствовать подписи в postfix_public_cert.pem. В случае несовпадения открытый ключ будет признан фальшивым и соединение немедленно завершится.

Подготовка сертификатов к использованию в Postfix

Независимо от того, где вы собираетесь использовать сертификаты – в smtp (почтовый клиент) или в smtpd (почтовый сервер), вы должны скопировать все сертификаты в каталог /etc/postfix/certs:

```
# mkdir /etc/postfix/certs
# cp cacert.pem /etc/postfix/certs
# cp ../*.pem /etc/postfix/certs
```

Затем вы должны запретить другим пользователям вашего хоста доступ к секретному ключу сервера postfix_private_key.pem:

```
# cd /etc/postfix/certs
# chmod 600 postfix_private_key.pem
```

В результате права доступа должны выглядеть так:

```
# ls -all certs/
total 20
drwxr-xr-x   2 root   root   4096 Nov  9 23:03 .
drwxr-xr-x   3 root   root   4096 Oct 28 00:13 ..
-rw-r--r--   1 root   root   1379 Nov  9 23:02 cacert.pem
-rw-------   1 root   root   1620 Nov  9 23:02 postfix_private_key.pem[1]
-rw-r--r--   1 root   root   3806 Nov  9 23:02 postfix_public_cert.pem
```

Теперь вы готовы к использованию в Postfix возможностей протокола TLS, описанных в следующей главе.

[1] Кстати, если никто не будет иметь права записи в этот файл, будет еще лучше: Postfix его прочтет все равно, а вот от случайной модификации администратором такие права доступа защитят. – *Примеч. науч. ред.*

18

Использование TLS

Протокол TLS (Transport Layer Security – безопасность транспортного уровня) присутствует в Postfix в двух формах: клиентской и серверной. Обе они, в дополнение к базовой функциональности TLS, имеют возможности настройки производительности, точной настройки уровня безопасности, а также предоставляют среду для безопасного использования открытого текста в SMTP AUTH и могут разрешать пересылку на основании клиентских сертификатов.

В этой главе рассказывается о том, как настроить клиентскую и серверную части протокола TLS. Вы познакомитесь с разными подходами к развертыванию TLS и с демонами, дополняющими основной набор демонов Postfix.

Проверка поддержки TLS в Postfix

Прежде чем приступать к работе с файлами конфигурации Postfix для TLS (описанными в RFC 2487), необходимо проверить, поддерживается ли протокол TLS в вашей версии Postfix. Это важно, т. к. базовая версия исходного кода Postfix вообще не поддерживает TLS: вам нужно будет установить специальный патч для обеспечения поддержки TLS и `STARTTLS`. Однако если вы используете бинарный дистрибутив[1], то может оказаться, что поддержка TLS у вас уже есть, т. к. многие ди-

[1] Хотя большинству специалистов ясно, что такое «двоичный дистрибутив», следует отметить, что такие вошедшие в русский язык фразы заставляют вначале сделать обратный перевод на английский («binary distribution»), а затем понять, что речь идет о Postfix, который был установлен из пакета с готовыми скомпилированными программами, а не из исходных текстов. – *Примеч. науч. ред.*

стрибутивы Linux включают ее в свои пакеты Postfix. В Postfix 2.2 TLS предлагается как дополнительная функциональность при сборке.

Примечание

Патч TLS для Postfix был разработан доктором Лутцем Джанике (Lutz Janicke), членом команды разработчиков OpenSSL, который профессионально занимается разработкой методик шифрования. На веб-сайте OpenSSL говорится: «Проект OpenSSL – это продукт совместных усилий по созданию надежного полнофункционального коммерческого инструментария с открытым исходным текстом, реализующего протоколы Secure Sockets Layer (SSL v2/v3) и Transport Layer Security (TLS v1), а также полную универсальную криптографическую библиотеку». Подробную информацию о проекте OpenSSL Project вы найдете на сайте *http://www.openssl.org*.

Хотя поддержка TLS реализована в виде патча, может оказаться, что ваша версия Postfix уже поддерживает TLS, т. к. многие дистрибутивы включают ее в свои пакеты Postfix.[1]

Для того чтобы определить, поддерживает ли TLS установленная у вас версия Postfix, проверьте наличие `tls` в выводе команды `postconf -d`. Команда `grep` должна вернуть параметры TLS и их значения по умолчанию, например:

```
# postconf -d | grep tls
smtp_enforce_tls = no
smtp_starttls_timeout = 300s
smtp_tls_CAfile =
smtp_tls_CApath =
smtp_tls_cert_file =
smtp_tls_cipherlist =
smtp_tls_dcert_file =
smtp_tls_dkey_file = $smtp_tls_dcert_file
smtp_tls_enforce_peername = yes
smtp_tls_key_file = $smtp_tls_cert_file
smtp_tls_loglevel = 0
smtp_tls_note_starttls_offer = no
smtp_tls_per_site =
smtp_tls_session_cache_database =
smtp_tls_session_cache_timeout = 3600s
smtp_use_tls = no
smtpd_enforce_tls = no
smtpd_tls_CAfile =
smtpd_tls_CApath =
smtpd_tls_ask_ccert = no
smtpd_tls_auth_only = no
smtpd_tls_ccert_verifydepth = 5
smtpd_tls_cert_file =
```

[1] Вы не ошиблись: автор действительно повторяет эту мысль дважды. В конце концов, это же важная мысль, не так ли? – *Примеч. науч. ред.*

```
smtpd_tls_cipherlist =
smtpd_tls_dcert_file =
smtpd_tls_dh1024_param_file =
smtpd_tls_dh512_param_file =
smtpd_tls_dkey_file = $smtpd_tls_dcert_file
smtpd_tls_key_file = $smtpd_tls_cert_file
smtpd_tls_loglevel = 0
smtpd_tls_received_header = no
smtpd_tls_req_ccert = no
smtpd_tls_session_cache_database =
smtpd_tls_session_cache_timeout = 3600s
smtpd_tls_wrappermode = no
smtpd_use_tls = no
tls_daemon_random_bytes = 32
tls_daemon_random_source =
tls_random_bytes = 32
tls_random_exchange_name = ${config_directory}/prng_exch
tls_random_prng_update_period = 60s
tls_random_reseed_period = 3600s
tls_random_source =
```

Наличие всех этих относящихся к TLS параметров указывает на то, что TLS поддерживается.

Сборка Postfix с поддержкой TLS

Если ваш бинарный пакет не включает в себя поддержку TLS, то придется собирать новую версию Postfix. Сначала определите, где в вашей системе находятся библиотеки и заголовочные файлы (включаемые файлы `.h`) OpenSSL. Для поиска используйте команду `find`:

```
# find /usr -name 'ssl.*'
```

Выполнение этой команды может занять некоторое время. При удачном завершении вы должны увидеть нечто подобное:

```
/usr/include/openssl/ssl.h
/usr/lib/libssl.so
/usr/lib/libssl.a
```

В данном примере `ssl.h` – это файл заголовков, `libssl.so` – динамическая версия библиотеки OpenSSL, а `libssl.a` – ее статическая версия.

Если вам не удается найти OpenSSL на своей машине, попробуйте поискать бинарный пакет в дистрибутиве вашей операционной системы. Необходимо установить инструментальные пакеты OpenSSL (обычно они называются `openssl-dev` или `openssl-devel`); в противном случае вы можете остаться без файлов заголовков.

Если вы хотите использовать более новую версию OpenSSL, чем та, что включена в ваш дистрибутив, или вам не удается найти бинарный пакет, соберите OpenSSL самостоятельно. Сейчас мы поговорим о том, как это сделать.

Сборка и установка OpenSSL из исходных текстов

OpenSSL, Postfix и патч TLS находятся в постоянной разработке. Поскольку патч TLS зависит как от Postfix, так и от OpenSSL, при скачивании исходных текстов и патча вам нужно убедиться в том, что все они подходят друг к другу.

Предупреждение

Некоторые дистрибутивы Linux поставляются с библиотеками OpenSSL, которые могут нарушить работу части вашей системы при перезаписи текущей установленной в ней версии OpenSSL. Если на вашем компьютере установлена версия OpenSSL 0.9.6 или выше, то продолжайте использовать ее до тех пор, пока не будете точно знать, как сконфигурировать новую версию, не вызвав конфликта с существующей.

OpenSSL версии 0.9.6 и выше отлично работает с Postfix TLS. В качестве альтернативы можно предложить установить новую библиотеку в другом каталоге, тем самым избежав проблем с перезаписью основных библиотек.

Проще всего определить, какой исходный текст вам подходит, обратившись к таблице совместимости TLS на веб-сайте Лутца Джанике *http://www.aet.tu-cottbus.de/personen/jaenicke/postfix_tls*. Вам нужно лишь выбрать версии исходных текстов OpenSSL, Postfix и патча TLS из одной строки таблицы – и вы готовы к сборке и установке исходных текстов.

Для установки OpenSSL выполните следующие действия:

1. Распакуйте архив исходных текстов OpenSSL от имени обычного пользователя командой `tar xzf openssl-`*`version`*`.tar.gz`, где *version* – это версия OpenSSL.
2. Перейдите в только что созданный каталог, содержащий исходные тексты OpenSSL.
3. Прочтите файл `INSTALL` и подумайте, нужны ли вам какие-то специальные параметры.
4. Если вы хотите собрать совместно используемые[1] библиотеки, выполните команду `configure` с параметром `--shared`; по умолчанию совместно используемые библиотеки не собираются. Если вы статически связываете Postfix с `libopenssl.a`, то вам придется перекомпилировать весь Postfix при обновлении OpenSSL, вызванном соображениями безопасности.
5. После запуска сценария `configure` для создания файлов Makefile запустите `make && make test`.
6. Переключитесь на суперпользователя (`root`) и запустите `make install`. Если вы не собираете совместно используемые библиотеки, то все готово.

[1] То есть динамические, или, что то же самое, разделяемые библиотеки. – *Примеч. науч. ред.*

7. Проверьте путь совместно используемых библиотек; при установке он выводится непосредственно перед завершением. Путь по умолчанию – `/usr/local/ssl/lib`.
8. Добавьте путь совместно используемых библиотек в путь поиска времени выполнения динамического компоновщика. Для Linux это означает, что вам следует добавить этот каталог в файл `/etc/ld.so.conf` и выполнить команду `ldconfig`. В Solaris нужно выполнить команду `crle`.

Сборка Postfix с поддержкой TLS

Теперь, когда у вас есть библиотеки и включаемые файлы OpenSSL, можно приступать к сборке новой, поддерживающей TLS, версии Postfix. Для этого потребуются исходные тексты Postfix и патч TLS.

Предупреждение

Прежде чем что-то делать, обязательно просмотрите файлы `README` и `INSTALL`; процедура установки поддержки TLS могла измениться с момента выхода книги.

Для сборки Postfix выполните следующие действия:

1. От имени обычного пользователя распакуйте исходные тексты Postfix и TLS в *отдельные* каталоги.
2. Перейдите в каталог исходных текстов Postfix.
3. Выполните команду `patch -p1 < ../tls_dir/pfixtls.diff` для применения патча, где *tls_dir* – это каталог, содержащий файл патча TLS с именем `pfixtls.diff`.
4. Укажите параметры сборки и запустите `make makefiles` и `make`, как указано ниже, здесь *ssl_prefix* – это ваш базовый каталог SSL, а *sasl2_prefix* – базовый каталог SASL2.

```
$ CFLAGS="-DUSE_SSL -DUSE_SASL_AUTH -Isasl2_prefix/include -Issl_prefix/
  includes" AUXLIBS="-Lssl_prefix/lib -lssl -lcrypto -Lsasl_prefix/lib -
  lsasl2" ❶
$ make makefiles
$ make
```

❶ Эти параметры означают сборку Postfix с поддержкой TLS и SASL2. Для получения информации об использовании дополнительных параметров обратитесь к соответствующему файлу каталога `readme` в дереве исходных текстов Postfix.

5. Переключитесь на суперпользователя (`root`).
6. Выполните `make install`, если это первая установка Postfix из исходных текстов, или `make upgrade`, если у вас уже есть установленная версия Postfix.
7. Если вы собрали OpenSSL как совместно используемую библиотеку, то с помощью команды `ldd `postconf -h daemon_directory`/smtpd`

убедитесь, что динамический компоновщик может найти все библиотеки, которые вы использовали для сборки Postfix.

После выполнения всех указанных операций вы должны располагать версией Postfix с поддержкой безопасности транспортного уровня.

Серверная часть TLS

Мы говорим о серверной части TLS, когда Postfix действует как почтовый сервер (MTA), предоставляющий TLS почтовым клиентам (рис. 18.1). Вы можете настроить Postfix для шифрования транспортного уровня, чтобы скрыть весь сеанс SMTP-взаимодействия для безопасного получения верительных данных SMTP AUTH открытым текстом или чтобы пересылать почту для клиентов на основе предоставленных ими сертификатов.

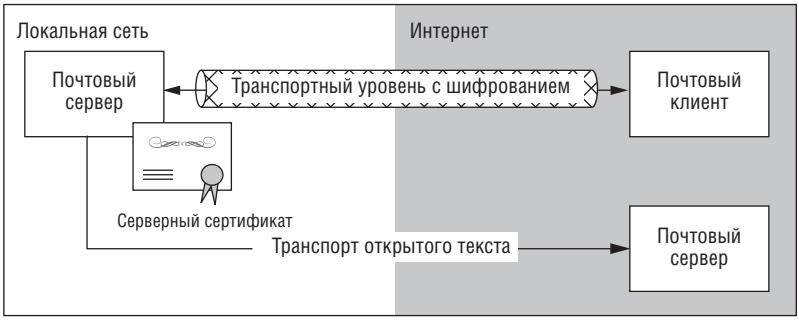

***Рис. 18.1.** TLS для почтового сервера Postfix*

Примечание

Postfix не передает ключевое слово `STARTTLS` утилите командной строки `sendmail`. Это мера безопасности; `sendmail` с поддержкой TLS потребовала бы доступа к секретному ключу сервера. Однако ключ принадлежит пользователю `root` и доступен для чтения только ему, а Postfix `sendmail` не имеет привилегий `root`.

Базовая конфигурация сервера

Для настройки серверной части TLS вам необходимо изменить пять параметров в конфигурационных файлах Postfix. Кроме того, следует задать два дополнительных параметра для облегчения отладки (это не только способствует выявлению проблем, но и помогает при настройке сеансов TLS, т. к. позволяет Postfix определить, какие клиенты могут использовать TLS, а какие – нет).

Настройка состоит из следующих этапов:

1. Включаем TLS в главном файле конфигурации Postfix.
2. Сообщаем Postfix, где искать сертификаты, необходимые для TLS.

3. Подключаем Postfix к генератору случайных чисел.
4. Увеличиваем уровень журналирования для получения полезной информации на время обучения использованию TLS.
5. Добавляем информацию в заголовки сообщений для последующего отслеживания TLS.

Включение серверной части TLS

По умолчанию серверная часть TLS не включена на серверах Postfix, способных ее поддерживать, так что Postfix не предлагает TLS клиентам, даже если они об этом просят. Для включения TLS на стороне сервера установите параметр `smtpd_use_tls` в файле `main.cf` в значение `yes`:

```
smtpd_use_tls = yes
```

После перезагрузки конфигурации Postfix будет предлагать почтовым клиентам ключевое слово `STARTTLS` в SMTP-диалоге, сообщая им тем самым о возможности открытия сеанса TLS. Однако включения серверной части TLS недостаточно – для того чтобы все заработало, необходимо указать Postfix, где искать сертификаты, необходимые для сеанса TLS. В журнале вы увидите сообщение, подобное такому:

```
Dec  1 03:07:13 mail postfix/smtpd[741]: TLS engine: do need at least RSA
_or_   DSA cert/key data
```

Определение путей к файлам сертификатов

Теперь будем указывать пути к файлам или каталогам, в которых хранятся серверные сертификаты. Как минимум необходимо предоставить серверный ключ и сертификат, подписанный центром сертификации, и соответствующий секретный ключ, который используется для создания запроса сертификата. Оба объявления должны содержаться в файле `main.cf`. В следующем примере сертификаты хранятся в `/etc/postfix/certs`:

```
smtpd_tls_key_file = /etc/postfix/certs/postfix_private_key.pem ❶
smtpd_tls_cert_file = /etc/postfix/certs/postfix_public_cert.pem ❷
```

❶ `smtpd_tls_key_file` – путь к секретному ключу.

❷ `smtpd_tls_cert_file` – путь к серверному сертификату.

Примечание

В приведенном примере предполагается, что серверный сертификат и секретный ключ находятся в разных файлах. Если вы решите поместить оба сертификата в один файл, то можете записать это при помощи следующей строки конфигурации: `smtpd_tls_cert_file = $smtpd_tls_key_file`.

После установки этих параметров вы можете запускать Postfix с TLS, работающим в серверном режиме, но в журнале все еще будут отображаться некоторые ошибки и предупреждения. Дело в том, что Postfix

не может передавать CA-сертификат и проверять сертификаты, отправленные почтовыми клиентами. Вам еще нужно настроить источник для CA-сертификатов.

Настройка корневого хранилища сертификатов Postfix

Как уже говорилось в главе 17, OpenSSL не имеет центрального корневого хранилища по умолчанию, так что вам нужно создать корневое хранилище, предназначенное специально для вашей почтовой системы, или использовать уже существующее в системе корневое хранилище. В рассматриваемой нами конфигурации используется файл `ca-bundle.crt`, поставляемый с модулем `mod_ssl` из состава Apache, который содержат несколько CA-сертификатов и серверов для Apache.

Примечание

Вы можете создать собственную коллекцию CA-сертификатов для уверенности в том, что они действительно получены от соответствующих центров сертификации и не были изменены какими-то третьими лицами при создании дистрибутива. Такой подход обеспечивает наибольшую безопасность, т. к. все усилия по повышению надежности посредством TLS пропадут напрасно, если окажется, что сертификаты, являющиеся основой всего механизма, поддельные.

Приступая к сбору CA-сертификатов для своего сервера, будьте готовы к тому, чтобы потратить на это значительное количество времени. Когда мы занимались этой главой, поиск сертификатов отнял у нас немало времени. Казалось, что почти все центры сертификации специально прячут необходимую информацию на своих веб-сайтах.

Для того чтобы найти в своей системе файл `ca-bundle.crt` модуля Apache `mod_ssl`, выполните в командной строке команду `locate ca-bundle.crt`:

```
$ locate ca-bundle.crt
/usr/share/ssl/certs/ca-bundle.crt
```

Если у вас нет команды `locate`, придется прибегнуть к более медленной команде `find`:

```
$ find / -name ca-bundle.crt
```

После того как вы нашли или создали корневое хранилище, следует настроить Postfix для его использования с помощью параметра `smtpd_tls_CAfile` в файле `main.cf`, например так:

```
smtpd_tls_CAfile = /usr/share/ssl/certs/ca-bundle.crt
```

Если вы действуете как собственный центр сертификации, то необходимо добавить в данное корневое хранилище свой CA-сертификат. Для того чтобы добавить свой сертификат в конец существующего хранилища, используйте команду `cat`, например:

```
$ cat /usr/local/ssl/misc/demoCA/cacert.pem >> /usr/share/ssl/certs/
          ca-bundle.crt
```

Примечание

Если ваш CA-сертификат является звеном в цепочке сертификатов, добавьте всю цепочку CA-сертификатов ниже корневого CA-сертификата.

Параметр `smtpd_tls_CAfile` используется в том случае, если все сертификаты находятся в одном файле. В качестве альтернативы Postfix предлагает параметр `smtpd_tls_CApath`, указывающий каталог, в котором сертификаты будут храниться в разных файлах.

Примечание

Способ хранения – это не единственное отличие между `smtpd_tls_CAfile` и `smtpd_tls_CApath`. Postfix обращается к файлам, указанным в `smtpd_tls_CApath`, только когда необходимо проверить сертификат. А файлы, перечисленные в параметре `smtpd_tls_CAfile`, Postfix читает при запуске, до входа в окружение chroot jail. Так что если вы решите использовать Postfix в chroot-окружении, то лучше задать параметр `smtpd_tls_CAfile`, тогда вы сможете поместить файлы сертификатов вне chroot jail.

Вероятно, стоит задать оба параметра, распределив некоторые CA-сертификаты из основного пакета, хранящегося в одном файле, по отдельным файлам. Postfix будет сначала читать файлы, указанные в параметре `smtpd_tls_CAfile`, затем, для надежности, просматривать `smtpd_tls_CApath`.

Подключение Postfix к генератору случайных чисел

Протокол TLS обеспечивает безопасный способ отправки почты не только потому, что шифрует соединение, но и потому, что никогда не использует повторно одну и ту же комбинацию цифр. Это реализуется за счет выбора (псевдо-) случайного числа для каждого нового TLS-сеанса.

OpenSSL не генерирует собственные случайные числа, т. к. большинство клонов Linux и BSD имеют встроенные источники случайных чисел в качестве системных устройств (в каталоге `/dev`).

Примечание

Если в вашей системе нет встроенного генератора случайных чисел, вы можете воспользоваться демоном генерирования псевдослучайных чисел (также созданным Лутцем Джанике). Для того чтобы настроить Postfix на использование этого демона, установите параметр `tls_random_exchange_name` в файле `main.cf`. Дополнительную информацию вы можете найти в файле `samples/sample-tls.cf` своего дистрибутива Postfix.

В системе обычно доступны два источника случайных чисел: `dev:/dev/random` и `dev:/dev/urandom`.

`/dev/random`

Генератор `/dev/random` предоставляет случайные данные высокого качества, но он не подходит для систем с интенсивным использованием TLS. Причина в том, что `/dev/random` может заблокироваться, если TLS запрашивает случайные данные слишком часто, истощая источник. Если такое случится, Postfix прекращает работу до тех пор, пока система не накопит достаточно энтропии для возобновления генерации чисел.

`/dev/urandom`

Генератор `/dev/urandom` никогда не блокируется, т. к. использует внутренний генератор псевдослучайных чисел для создания энтропийных данных. Используйте `/dev/urandom` в тех системах, которые автоматически запускают Postfix.

Для подключения Postfix к генератору случайных чисел задайте параметр `tls_random_source` в файле `main.cf` и перезагрузите конфигурацию:

```
tls_random_source = dev:/dev/urandom
```

Примечание

Версии OpenSSL выше 0.9.6 автоматически обнаруживают `/dev/urandom`. Если вы используете одну из таких версий, то нет необходимости в установке параметра `tls_random_source`. OpenSSL 0.9.7 делает еще один шаг вперед, распознавая и другие генераторы случайных чисел. Если вы пользуетесь OpenSSL 0.9.7, обратитесь к сопутствующей документации за подробностями.

Повышение уровня журналирования TLS

Подсистема TLS использует `smtpd_tls_loglevel` параметр для управления объемом связанной с TLS информации, записываемой в почтовый журнал. Существует пять уровней подробности вывода данных (табл. 18.1).

Таблица 18.1. Уровни журналирования для smtpd (значения параметра smtpd_tls_loglevel)

Уровень	Описание
0	Отсутствие журналирования TLS; значение по умолчанию
1	Информация о запуске и сертификате
2	Вся информация уровня 1 плюс сведения о различных этапах TLS-переговоров
3	Вся информация уровня 2 плюс шестнадцатеричный и ASCII-дампы переговорного процесса
4	Вся информация уровня 3 плюс шестнадцатеричный и ASCII-дампы всего процесса передачи сообщения после отправки почтовым клиентом ключевого слова `STARTTLS`

При первом включении серверной части TLS установите уровень журналирования равным 2 – у вас будет достаточно сведений для отладки, если что-то пойдет не так:

```
smtpd_tls_loglevel = 2
```

Добавление информации в заголовки сообщений

Возможно, вы захотите, чтобы ваш почтовый сервер добавлял TLS-информацию в заголовок Received каждого сообщения, отправленного при помощи TLS. Для этого надо установить параметр `smtpd_tls_received_header` в файле `main.cf`, например:

```
smtpd_tls_received_header = yes
```

После перезагрузки конфигурации вы должны увидеть в заголовках сообщений нечто подобное:

```
Received: from client.example.com (client.example.com [172.16.0.3])
      (using TLSv1 with cipher EDH-RSA-DES-CBC3-SHA (168/168 bits))
      (No client certificate requested)
      by mail.example.com (Postfix) with ESMTP id B637A7247
      for <tls-bounce@mail.examples.com>; Wed, 10 Dec 2003 23:37:02 +0100 (CET)
```

Тестирование серверной части TLS

При тестировании базовой конфигурации TLS вам следует выполнить три проверки:

1. Проверить в файле журнала, обнаружил ли сервер Postfix какие-либо ошибки.
2. Проверить наличие ключевого слова `STARTTLS` в SMTP-диалоге, чтобы определить, предлагает ли Postfix почтовым клиентам использовать TLS.
3. Проверить TLS при помощи программы openssl, чтобы показать, что Postfix может инициировать сеанс TLS, используя сертификаты, предоставленные в базовой конфигурации.

Проверка файла журнала

Начинаем тестирование с просмотра файла журнала, используя регулярное выражение:

```
$ egrep '(reject|error|warning|fatal|panic):' /var/log/maillog
```

Эта команда выводит все строки `/var/log/maillog`, содержащие слово `reject`, `error`, `warning`, `fatal` или `panic`, за которым следует двоеточие (`:`).

Если вы все сделали правильно, то в журнале не должно быть никаких ошибок, связанных с TLS. Если же они все-таки появились, проверьте, нет ли опечаток в файле конфигурации и корректно ли выданы разрешения на чтение сертификатов.

Проверка наличия ключевого слова STARTTLS в SMTP-диалоге

Теперь открываем сеанс telnet с сервером Postfix, чтобы проверить, предлагает ли он почтовым клиентам использовать TLS. Внимательно ищем в выходных данных ключевое слово `STARTTLS`:

```
$ telnet localhost 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH NTLM LOGIN PLAIN OTP DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
QUIT
221 Bye
```

Имитация TLS-сеанса «почтовый клиент – сервер» при помощи OpenSSL

Последняя проверка состоит в имитации сеанса почтового клиента с сервером с использованием параметра `openssl s_client`. Клиент OpenSSL может соединяться с удаленными хостами по TLS/SSL, выводя по ходу дела множество диагностических данных. Успех такой проверки означает, что серверная конфигурация TLS работает, и можно проводить тестирование для почтового клиента. В случае неудачи вы получите множество полезных отладочных данных для отслеживания ошибки.

Приведем пример успешного сеанса (в данном случае путь CA-сертификата – `/etc/postfix/certs`):

```
# openssl s_client -starttls smtp -CApath /etc/postfix/certs/ -connect
localhost:25
CONNECTED(00000003)
depth=1 /C=DE/ST=Bavaria/L=Munich/O=Postfix Book/OU=#Authoring/
CN=mail.example.com/emailAddress=postmaster@example.com
verify return:1
depth=0 /C=DE/ST=Bavaria/L=Munich/O=Postfix Book/OU=Mailserver/
CN=mail.example.com/emailAddress=postmaster@example.com
verify return:1
---
Certificate chain
 0 s:/C=DE/ST=Bavaria/L=Munich/O=Postfix Book/OU=Mailserver/
CN=mail.example.com/emailAddress=postmaster@example.com
   i:/C=DE/ST=Bavaria/L=Munich/O=Postfix Book/OU=#Authoring/
CN=mail.example.com/emailAddress=postmaster@example.com
 1 s:/C=DE/ST=Bavaria/L=Munich/O=Postfix Book/OU=#Authoring/
CN=mail.example.com/emailAddress=postmaster@example.com
   i:/C=DE/ST=Bavaria/L=Munich/O=Postfix Book/OU=#Authoring/
CN=mail.example.com/emailAddress=postmaster@example.com
```

```
---
Server certificate
-----BEGIN CERTIFICATE-----
MIID4DCCAOmgAwIBAgIBATANBgkqhkiG9w0BAQQFADCBnjELMAkGA1UEBhMCREUx
EDAOBgNVBAgTB0JhdmFyaWExDzANBgNVBAcTBk11bmljaDEVMBMGA1UEChMMUG9z
dGZpeCBCb29rMRMwEQYDVQQLFAojQXVOaG9yaW5nMRkwFwYDVQQDExBtYWlsLmV4
YW1wbGUuY29tMSUwIwYJKoZIhvcNAQkBFhZwb3N0bWFzdGVyQGV4YW1wbGUuY29t
MB4XDTAzMTAyMzIwMTkyOVoXDTAOMTAyMjIwMTkyOVowgZ4xCzAJBgNVBAYTAkRF
MRAwDgYDVQQIEwdCYXZhcmlhMQ8wDQYDVQQHEwZNdW5pY2gxFTATBgNVBAoTDFBv
c3RmaXggQm9vazETMBEGA1UECxMKTWFpbHNlcnZlcjEZMBcGA1UEAxMQbWFpbC5l
eGFtcGxlLmNvbTElMCMGCSqGSIb3DQEJARYWcG9zdG1hc3RlckBleGFtcGxlLmNv
bTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA9wBRlv3EsemFDqOX5L/4DUCt
8oIpdlOXOpNMKqh/LnWuFXivCy52dMMbWtQgWaR+xRKaacyLeIdeyDx5LwzOgOd6
3zT+M2TAwGi6eQp+u8NpIuDF3eKYRBPoLGMuQiWkOcwNjagXg+U1Q9oVBseMgg/a
OVj8aNasi4qJ2N59sbcCAwEAAaOCASowggEmMAkGA1UdEwQCMAAwLAYJYIZIAYb4
QgENBB8WHU9wZW5TUOwgR2VuZXJhdGVkIENlcnRpZmljYXRlMBOGA1UdDgQWBBQj
RXFGfepblNkc6G/57Et7xRI1eDCBywYDVR0jBIHDMIHAgBQJScWoXDhSbW76EWQI
GUMvoySuN6GBpKSBoTCBnjELMAkGA1UEBhMCREUxEDAOBgNVBAgTB0JhdmFyaWEx
DzANBgNVBAcTBk11bmljaDEVMBMGA1UEChMMUG9zdGZpeCBCb29rMRMwEQYDVQQL
FAojQXVOaG9yaW5nMRkwFwYDVQQDExBtYWlsLmV4YW1wbGUuY29tMSUwIwYJKoZI
hvcNAQkBFhZwb3N0bWFzdGVyQGV4YW1wbGUuY29tggEAMAOGCSqGSIb3DQEBBAUA
A4GBADUNOgZfc8ClIRir/9DboKup+MSijhlPi5bmMOj6OWNj6STiNrcjTaF8qH+6
LFxXbclJfWUHaEFvSLSeW79zh7KX67yOU46nVVYdF8+gHV/XnZK6f/6CpwcjOnQP
PI3GDtLoNXUlPqrngrJskWUuDcZwkQBlXinZlyMSs1gcSDSO
-----END CERTIFICATE-----
subject=/C=DE/ST=Bavaria/L=Munich/O=Postfix Book/OU=Mailserver/
CN=mail.example.com/emailAddress=postmaster@example.com
issuer=/C=DE/ST=Bavaria/L=Munich/O=Postfix Book/OU=#Authoring/
CN=mail.example.com/emailAddress=postmaster@example.com
---
No client certificate CA names sent
---
SSL handshake has read 2592 bytes and written 356 bytes
---
New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA
Server public key is 1024 bit
SSL-Session:
    Protocol  : TLSv1
    Cipher    : DHE-RSA-AES256-SHA
    Session-ID: D341BF543EB5690DA873EFD0B0B4CB2EF210930812C14F3DBB85
                BD1AE92C6CB3
    Session-ID-ctx:
    Master-Key: D4E3B4617214EDA8E1D2EAF54482FC65D1BD7BF5474F2FB2E2C
                0312BE098D8AF29ABC6603C4A89B7B413ED24D79375CD
    Key-Arg   : None
    Start Time: 1068108666
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
220 mail.example.com ESMTP Postfix (2.0.16-20030921)
QUIT
DONE
```

Настройка производительности сервера

Криптография – это серьезная нагрузка для вашего процессора. В начале каждого сеанса TLS клиент и сервер выполняют несколько операций с секретным ключом для подписания приветственных сообщений – это процесс, требующий множества вычислений. Большое количество одновременных сеансов TLS может серьезно замедлить работу почтового сервера.

По умолчанию Postfix-демон `smtpd` запоминает ключ сеанса для своих соединений. Однако Postfix позволяет процессам `smtpd` завершать свою работу после определенного периода бездействия, чтобы освободить ресурсы сервера и, возможно, загрузить новые данные о конфигурации. К сожалению, это означает, что при завершении работы `smtpd` сервер Postfix теряет информацию о ключе сеанса и вынужден повторно вычислять его, когда почтовый клиент возвращается для передачи нового сообщения.

Для того чтобы избежать потери информации о ключах сеансов при завершении работы экземпляра `smtpd`, Postfix может поддерживать кэш ключей сеансов (см. следующий раздел).

Настройка кэша для ключей сеансов TLS

Для того чтобы справиться с проблемой вычислительной нагрузки, которую может создать шифрование TLS, следует настроить процессы `smtpd` так, чтобы ключи сеансов хранились в базе данных. После того как один процесс `smtpd` сохраняет ключ, доступ к нему имеют все процессы `smtpd`, вне зависимости от того, работают ли они уже давно или же запущены только что. Наличие кэша ключей сеансов значительно снижает загрузку процессора.

Для включения кэширования ключей сеансов задайте параметр `smtpd_tls_session_cache_database` и запустите демон `tlsmgr`. Параметры `main.cf` имеют такой вид:

```
smtpd_tls_session_cache_database = sdbm:/etc/postfix/smtpd_scache
smtpd_tls_session_cache_timeout = 3600s
```

Примечание

Кэширование ключей сеансов требует параллельного доступа на запись к базе данных ключей. В Postfix только базы данных типа SDBM поддерживают такую возможность. Данный тип ключей распознают все поддерживающие TLS версии Postfix.

По умолчанию все ключи сеансов в базе данных теряют силу через один час (3600 секунд). RFC 2246 рекомендует в качестве максимального тайм-аута период в 24 часа. Вы можете изменить поведение по умолчанию, указав другое значение для параметра `smtpd_tls_session_cache_timeout` (в секундах).

Обслуживание кэша ключей сеансов TLS посредством tlsmgr

Postfix нуждается в активном сопровождении своей базы данных кэша ключей сеансов TLS. Из соображений безопасности вам следует удалять устаревшие ключи, а также избегать чрезмерного роста базы данных. Эти задачи решает демон `tlsmgr`, который присутствует только в поддерживающих TLS версиях Postfix. Вот что он делает:

- Помогает генерировать случайные числа в системах, не обладающих соответствующей встроенной поддержкой.
- Удаляет ключи с истекшим сроком действия из базы данных кэша сеансов в соответствии со значением параметра `smtpd_tls_session_cache_timeout`.
- При перезапуске Postfix пересоздает с нуля базу данных, указанную в параметре `smtpd_tls_session_cache_database`.

Для запуска `tlsmgr` следует проверить, включен ли он в файле `master.cf`. Если вы устанавливали версию из исходных текстов, то ничего менять не придется, а вот некоторые дистрибутивы отключают использование этого демона в своих пакетах Postfix, так что стоит проверить, не закомментирована ли соответствующая строка в вашем файле `master.cf`:

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
# ==========================================================================
...
tlsmgr    fifo  -       -       n       300     1       tlsmgr
...
```

Предупреждение

Никогда не помещайте базу данных кэша сеансов TLS в окружение chroot jail. Взломанная база данных кэша сеансов может быть использована для того, чтобы заставить почтовые клиенты поверить в то, что они общаются с надежным почтовым сервером и могут отправлять секретную информацию.

Вы можете запускать демон `tlsmgr` через chroot, т. к. он открывает базу данных ключей сеансов до смены корневого каталога, а следовательно, может читать и писать данные в базу данных, будучи запущенным посредством chroot.

После внесения изменений в файл `master.cf` следует перезагрузить конфигурацию Postfix для запуска `tlsmgr`.

Серверные меры безопасности при предоставлении SMTP AUTH

SMTP AUTH может предлагать механизмы открытого текста, такие как PLAIN и LOGIN. Некоторые почтовые клиенты, в частности Microsoft

Outlook и Outlook Express, в настоящее время могут использовать лишь эти механизмы. Пользователи обычно не обращают внимания на механизмы обеспечения безопасности, а вот у администраторов использование открытого текста для SMTP AUTH вызывает беспокойство, т. к. любой, кто может читать бинарные пакеты в сети, сможет без труда извлечь имена пользователей и пароли.

Вы можете защититься от передачи имени пользователя и пароля открытым текстом, предлагая SMTP AUTH только в сочетании с TLS.

Предоставление SMTP AUTH в сочетании с TLS

В Postfix существует параметр `smtpd_tls_auth_only` для предоставления SMTP AUTH только после открытия зашифрованного SMTP-соединения. По умолчанию этот параметр не включен, чтобы установить его, добавьте в файл `main.cf` следующую строку и перезагрузите конфигурацию:

```
smtpd_tls_auth_only = yes
```

Помните, что ограничение SMTP AUTH только сеансами TLS – это очень строгий способ, запрещающий использование механизмов открытого текста в незашифрованных сеансах SMTP, который попутно запрещает использование некоторых других (более надежных) механизмов в обычных SMTP-соединениях.

Для того чтобы проверить, вступило ли в силу принудительное применение TLS, убедимся в том, что сервер Postfix не предлагает SMTP AUTH незашифрованным сеансам. Подключитесь к своему серверу через порт 25 и отправьте приветствие `EHLO` *ваше.полностью.определенное.доменное.имя* следующим образом:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250 8BITMIME
QUIT
221 Bye
```

Обратите внимание на то, что в этом сеансе нет упоминания об аутентификации SMTP AUTH и ее механизмах, так что первый этап настройки можно считать выполненным.

Тестирование при использовании TLS

Теперь следует проверить, предлагается ли SMTP AUTH в сеансе TLS. Как и раньше при тестировании TLS, запускаем `openssl s_client` для

подключения к серверу и выдаем команду EHLO *ваше.полностью.определенное.доменное.имя*, как в сеансе telnet в предыдущем разделе. В этот раз вывод будет гораздо более объемным, но информацию SMTP AUTH все же можно отыскать – она в самом конце:

```
# openssl s_client -starttls smtp -CApath /etc/postfix/certs/ -connect
localhost:25
CONNECTED(00000003)
depth=1 /C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example
Inc./CN=mail.example.com/emailAddress=postmaster@example.com
verify return:1
depth=0 /C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=MX Services/
CN=mail.example.com/emailAddress=postmaster@example.com
verify return:1
---
Certificate chain
 0 s:/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=MX Services/
CN=mail.example.com/emailAddress=postmaster@example.com
   i:/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example Inc./
CN=mail.example.com/emailAddress=postmaster@example.com
 1 s:/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example Inc./
CN=mail.example.com/emailAddress=postmaster@example.com
   i:/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example Inc./
CN=mail.example.com/emailAddress=postmaster@example.com
---
Server certificate
-----BEGIN CERTIFICATE-----
MIID9jCCA1+gAwIBAgIBATANBgkqhkiG9w0BAQQFADCBpjELMAkGA1UEBhMCRVgx
ETAPBgNVBAgTCEV4YW1wbGlhMRMwEQYDVQQHEwpFeGFtcGxldG9uMSOwKwYDVQQK
EyRDZXJOaWZpY2FOaW9uIEF1dGhvcmlOeSBFeGFtcGxlIEluYy4xGTAXBgNVBAMT
EG1haWwuZXhhbXBsZS5jb2OxJTAjBgkqhkiG9w0BCQEWFnBvc3RtYXNOZXJAZXhh
bXBsZS5jb2OwHhcNMDMxMTA5MjEyNTEzWhcNMDQxMTA4MjEyNTEzWjCBpDELMAkG
A1UEBhMCRVgxETAPBgNVBAgTCEV4YW1wbGlhMRMwEQYDVQQHEwpFeGFtcGxldG9u
MRUwEwYDVQQKEwxFeGFtcGxlIEluYy4xFDASBgNVBAsTC01YIFNlcnZpY2VzMRkw
FwYDVQQDExBtYWlsLmV4YW1wbGUuY29tMSUwIwYJKoZIhvcNAQkBFhZwb3NObWFz
dGVyQGV4YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDL1OHc
H7lyo2bcDbafEeTvsSEGepsleBAmMsB1ohWLnjUcEmE5Rth9eF/TMYUABiWhnXOb
2HOKOalzuyjQqLtFHy4Bh6EcNeMdTtrEPZ2kYw+/ARkaGJrwzlNwfpzwuBhBr/qX
5FQstSG2cI4vMRkb2Vb9sq8aFneAMn+zH98v9QIDAQABo4IBMjCCAS4wCQYDVR0T
BAIwADAsBglghkgBhvhCAQOEHxYdT3BlblNTTCBHZW5lcmFOZWQgQ2VydGlmaWNh
dGUwHQYDVR0OBBYEFJ42nZvtTjJzDoZVKv8bSfkcRxd1MIHTBgNVHSMEgcswgciA
FABSrbf6wu8BGp57D1fb3OSCWYOLoYGspIGpMIGmMQswCQYDVQQGEwJFWDERMA8G
A1UECBMIRXhhbXBsaWExEzARBgNVBAcTCkV4YW1wbGVOb24xLTArBgNVBAoTJENl
cnRpZmljYXRpb24gQXVOaG9yaXR5IEV4YW1wbGUgSW5jLjEZMBcGA1UEAxMQbWFp
bC5leGFtcGxlLmNvbTElMCMGCSqGSIb3DQEJARYWcG9zdG1hc3RlckBleGFtcGxl
LmNvbYIBADANBgkqhkiG9w0BAQQFAAOBgQDOnDMeoWihd+TGQ+zJPF35RsZekYc2
OzayT4Ratkiv1GFKVRHVjr9iNgT3nywQonJzWVmqcm52LUBidtHhyY/VKLPhGCQM
VffjvUbVgBaygkVOXmVSrFq7w+A42ejqLCP/+Hi6o1RF9FfJoJPiyZ1LVStiIDYF
l2DRSfGKL4A+xw==
-----END CERTIFICATE-----
subject=/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=MX Services/
```

```
CN=mail.example.com/emailAddress=postmaster@example.com
issuer=/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example
Inc./CN=mail.example.com/emailAddress=postmaster@example.com
---
Acceptable client certificate CA names
/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example Inc./
CN=mail.example.com/emailAddress=postmaster@example.com
---
SSL handshake has read 2822 bytes and written 368 bytes
---
New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA
Server public key is 1024 bit
SSL-Session:
    Protocol  : TLSv1
    Cipher    : DHE-RSA-AES256-SHA
    Session-ID: 01DFC00E443BBA8E4E9FE65C7F398702D7BB95367E62D9
                CBD12F217A97A9B8FC
    Session-ID-ctx:
    Master-Key: E0F1C5F47787E3D9C9E236E38407555DE544C97BB9F81A
                CE3343C897DF8E50691AB432D03E2D79509F452DA7BB363CB8
    Key-Arg   : None
    Start Time: 1071223541
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-AUTH NTLM LOGIN PLAIN OTP DIGEST-MD5 CRAM-MD5
250-AUTH=NTLM LOGIN PLAIN OTP DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
QUIT
DONE
```

Выделенные жирным курсивом строки вывода AUTH показывают, что Postfix предлагает SMTP AUTH в соединениях с шифрованием. Теперь вы можете настраивать почтовые клиенты для использования механизмов открытого текста SMTP AUTH в сочетании с TLS.

Управление механизмами SASL в TLS

Существует более тонкий способ настройки запрета на применение механизмов открытого текста в обычных SMTP-соединениях – использование параметра `smtpd_sasl_tls_security_options`. Как и в предыдущем разделе, этот параметр указывает на то, что механизмы открытого текста должны быть защищены сеансом TLS, но дополнительно разре-

шает использование механизмов, обеспечивающих более высокий уровень безопасности, в сеансах без шифрования (для этого необходимо задать правильное сочетание параметров `smtpd_sasl_security_options` и `smtpd_sasl_tls_security_options`):

```
smtpd_sasl_security_options = noanonymous, noplaintext
smtpd_sasl_tls_security_options = noanonymous
```

Первая строка запрещает анонимную аутентификацию и аутентификацию механизмами открытого текста, но вторая строка подменяет ее, говоря, что в TLS-сеансе открытый текст допустим.

Тестирование SASL в сочетании с TLS

Как и при проверке запрета использования всех механизмов SMTP AUTH, начинаем тестирование конфигурации SASL с того, что посмотрим, не предлагает ли Postfix механизмы открытого текста в сеансе без шифрования. Подключитесь к своему серверу через порт 25, отправьте команду `EHLO` *ваше.полностью.определенное.доменное.имя* и посмотрите на результат:

```
$ telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH NTLM OTP DIGEST-MD5 CRAM-MD5
250-AUTH=NTLM OTP DIGEST-MD5 CRAM-MD5
250 8BITMIME
QUIT
221 Bye
```

Как видите, в строках SMTP AUTH не упоминаются механизмы открытого текста LOGIN и PLAIN, так что настройка `smtpd_sasl_security_options = noanonymous, noplaintext` работает.

Тестирование при использовании TLS

Убедившись в том, что параметр `smtpd_sasl_security_options` действует, проверим работу настройки `smtpd_sasl_tls_security_options = noanonymous`. Как и раньше, запускаем `openssl s_client` для подключения к серверу и выполняем команду `EHLO` *ваше.полностью.определенное.доменное.имя*. Результат должен выглядеть так:

```
# openssl s_client -starttls smtp -CApath /etc/postfix/certs/ -connect
localhost:25
CONNECTED(00000003)
depth=1 /C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example
Inc./CN=mail.example.com/emailAddress=postmaster@example.com
```

```
verify return:1
depth=0 /C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=MX Services/
CN=mail.example.com/emailAddress=postmaster@example.com
verify return:1
---
Certificate chain
 0 s:/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=MX Services/
CN=mail.example.com/emailAddress=postmaster@example.com
   i:/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example Inc./
CN=mail.example.com/emailAddress=postmaster@example.com
 1 s:/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example Inc./
CN=mail.example.com/emailAddress=postmaster@example.com
   i:/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example Inc./
CN=mail.example.com/emailAddress=postmaster@example.com
---
Server certificate
-----BEGIN CERTIFICATE-----
MIID9jCCA1+gAwIBAgIBATANBgkqhkiG9w0BAQQFADCBpjELMAkGA1UEBhMCRVgx
ETAPBgNVBAgTCEV4YW1wbGlhMRMwEQYDVQQHEwpFeGFtcGxldG9uMSOwKwYDVQQK
EyRDZXJOaWZpY2FOaW9uIEF1dGhvcmlOeSBFeGFtcGxlIEluYy4xGTAXBgNVBAMT
EG1haWwuZXhhbXBsZS5jb2OxJTAjBgkqhkiG9w0BCQEWFnBvc3RtYXNOZXJAZXhh
bXBsZS5jb2OwHhcNMDMxMTA5MjEyNTEzWhcNMDQxMTA4MjEyNTEzWjCBpDELMAkG
A1UEBhMCRVgxETAPBgNVBAgTCEV4YW1wbGlhMRMwEQYDVQQHEwpFeGFtcGxldG9u
MRUwEwYDVQQKEwxFeGFtcGxlIEluYy4xFDASBgNVBAsTC01YIFNlcnZpY2VzMRkw
FwYDVQQDExBtYWlsLmV4YW1wbGUuY29tMSUwIwYJKoZIhvcNAQkBFhZwb3NObWFz
dGVyQGV4YW1wbGUuY29tMIGfMAOGCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDL1OHc
H7lyo2bcDbafEeTvsSEGepsleBAmMsB1ohWLnjUcEmE5Rth9eF/TMYUABiWhnXOb
2HOKOalzuyjQqLtFHy4Bh6EcNeMdTtrEPZ2kYw+/ARkaGJrwzlNwfpzwuBhBr/qX
5FQstSG2cI4vMRkb2Vb9sq8aFneAMn+zH98v9QIDAQABo4IBMjCCAS4wCQYDVR0T
BAIwADAsBglghkgBhvhCAQOEHxYdT3BlblNTTCBHZW5lcmFOZWQgQ2VydGlmaWNh
dGUwHQYDVR0OBBYEFJ42nZvtTjJzDoZVKv8bSfkcRxd1MIHTBgNVHSMEgcswgciA
FABSrbf6wu8BGp57D1fb3OSCWYOLoYGspIGpMIGmMQswCQYDVQQGEwJFWDERMA8G
A1UECBMIRXhhbXBsaWExEzARBgNVBAcTCkV4YW1wbGVOb24xLTArBgNVBAoTJENl
cnRpZmljYXRpb24gQXVOaG9yaXR5IEV4YW1wbGUgSW5jLjEZMBcGA1UEAxMQbWFp
bC5leGFtcGxlLmNvbTElMCMGCSqGSIb3DQEJARYWcG9zdG1hc3RlckBleGFtcGxl
LmNvbYIBADANBgkqhkiG9w0BAQQFAAOBgQDOnDMeoWihd+TGQ+zJPF35RsZekYc2
OzayT4Ratkiv1GFKVRHVjr9iNgT3nywQonJzWVmqcm52LUBidtHhyY/VKLPhGCQM
VffjvUbVgBaygkVOXmVSrFq7w+A42ejqLCP/+Hi6o1RF9FfJoJPiyZ1LVStiIDYF
l2DRSfGKL4A+xw==
-----END CERTIFICATE-----
subject=/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=MX Services/
CN=mail.example.com/emailAddress=postmaster@example.com
issuer=/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example
Inc./CN=mail.example.com/emailAddress=postmaster@example.com
---
Acceptable client certificate CA names
/C=EX/ST=Examplia/L=Exampleton/O=Certification Authority Example Inc./
CN=mail.example.com/emailAddress=postmaster@example.com
---
SSL handshake has read 2822 bytes and written 368 bytes
---
```

```
New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA
Server public key is 1024 bit
SSL-Session:
    Protocol  : TLSv1
    Cipher    : DHE-RSA-AES256-SHA
    Session-ID: 01DFC00E443BBA8E4E9FE65C7F398702D7BB95367E62D9
                CBD12F217A97A9B8FC
    Session-ID-ctx:
    Master-Key: E0F1C5F47787E3D9C9E236E38407555DE544C97BB9F81A
                CE3343C897DF8E50691AB432D03E2D79509F452DA7BB363CB8
    Key-Arg   : None
    Start Time: 1071223541
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-AUTH NTLM LOGIN PLAIN OTP DIGEST-MD5 CRAM-MD5
250-AUTH=NTLM LOGIN PLAIN OTP DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
QUIT
DONE
```

Как видите, в TLS-сеансе Postfix предлагает механизмы открытого текста LOGIN и PLAIN – это доказывает, что и настройка `smtpd_sasl_tls_security_options = noanonymous` работает. Теперь можно приступать к настройке почтовых клиентов для использования механизмов открытого текста SMTP AUTH в сочетании с TLS.

Пересылка на основании сертификатов: серверная часть

Postfix поддерживает пересылку почты для клиентов на основании предоставленных ими сертификатов (рис. 18.2), что является альтернативой пересылки с использованием SMTP AUTH. Это удобно в сетях, где вы не хотите (или не можете) использовать SMTP AUTH, или если вы хотите упростить процесс пересылки и шифрования транспортного уровня, объединив эти две операции.

Единственный недостаток данного метода состоит в том, что его поддерживает лишь один известный почтовый клиент с графическим пользовательским интерфейсом – Netscape/Mozilla. Однако, несмотря на это ограничение, такой подход оправдан в большой сети, где серверы Postfix установлены в нескольких разных местах, при этом их доступ в Интернет осуществляется только по коммутируемым каналам и они имеют динамические IP-адреса. В этой ситуации разумно сделать так, чтобы

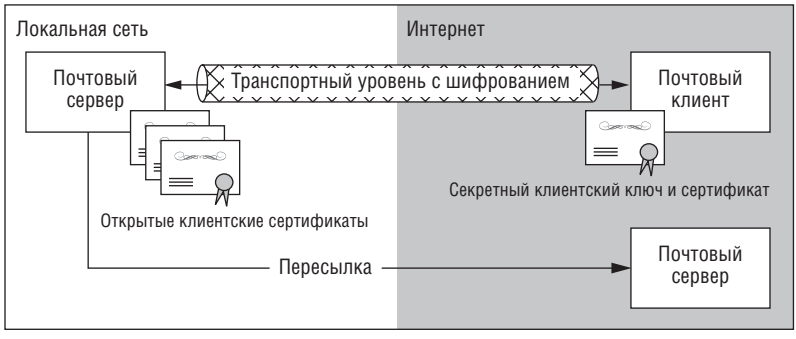

Рис. 18.2. *TLS для пересылки на основании сертификатов*

серверы Postfix с коммутируемым доступом пересылали исходящие сообщения серверу, имеющему статический IP-адрес, при этом пересылка производилась бы на основании сертификатов, чтобы гарантировать, что сообщения поступили от одного из коммутируемых серверов, а не от какой-то случайной третьей стороны. Кроме того, данный метод упрощает настройку сервера, делает работу Postfix более безопасной за счет исключения SMTP AUTH и защищает передачу внутренних сообщений.

Примечание

Время идет, и все меньше почтовых серверов принимают сообщения от коммутируемых и DSL-линий, т. к. именно с IP-адресов таких систем приходит огромное количество спама. Специальные черные списки на основе DNS (DNSBL), называемые списками пользователей коммутируемого доступа (Dial-Up User Lists, DUL), теперь запрещают целые подсети, о которых известно, что они используются машинами, подключенными через коммутируемые каналы.[1]

Для настройки сервера нужно выполнить следующие действия:

1. Настроить Postfix для запроса клиентских сертификатов.
2. Настроить Postfix для разрешения пересылки на основании предоставленных клиентами сертификатов.

Примечание

Пересылка на основании сертификатов требует настройки как клиентской, так и серверной частей TLS. В этом разделе будет рассказано о серверной части, а о настройке клиентской части мы поговорим далее в этой же главе.

[1] …и эта тенденция создает серьезные проблемы администраторам небольших организаций, чьи провайдеры выдали им адрес из такой попавшей в черный список подсети; к сожалению, это случается во всем мире. Пока мне представляется преждевременным включать на почтовых серверах запрет приема писем из таких сетей. – *Примеч. науч. ред.*

Настройка Postfix для запроса клиентских сертификатов

Первое, что следует сделать для включения пересылки на основании сертификатов, – это сообщить Postfix о необходимости явно запрашивать клиентские сертификаты. Это необходимо, т. к. обычно почтовые клиенты не предоставляют свои сертификаты автоматически. Устанавливаем значение параметра `smtpd_tls_ask_ccert` (по умолчанию он не включен):

```
smtpd_tls_ask_ccert = yes
```

Данный параметр также полезен для отладки, его можно оставить навсегда включенным, т. к. информация, добавленная в заголовок каждого сообщения, отправленного через TLS, не создает никакой угрозы для безопасности.

Примечание

Если сертификат недоступен, то почтовый клиент Netscape или жалуется на это, или предлагает несколько клиентских сертификатов на выбор. Это может раздражать, поэтому по умолчанию данная функциональность отключена. Однако если вы хотите использовать пересылку на основании сертификатов, то вашему SMTP-серверу сертификат необходим.

Настройка Postfix для разрешения пересылки на основании клиентских сертификатов

Патч TLS для Postfix включает в себя два дополнительных ограничения, которые могут управлять пересылкой посредством параметра `smtpd_recipient_restrictions`. Выбор ограничения зависит от вашего CA-сертификата:

Пересылка на основании клиентского сертификата

Вы можете создать карту сертификатов клиентов, которые могут пересылать почту через Postfix. Это безопасный способ для тех случаев, когда сертификаты ваших клиентов выданы несколькими разными (официальными) центрами сертификации.

Пересылка на основании CA-сертификата

Если вы поддерживаете свой собственный центр сертификации и обладаете полным контролем над сертификатами, то можете разрешить пересылку всем почтовым клиентам с сертификатами, подписанными вашим собственным центром сертификации.

Обе эти возможности описаны в следующих двух разделах.

Пересылка на основании клиентского сертификата

Если ваша настройка требует, чтобы почтовые клиенты для пересылки предоставляли сертификаты, подписанные одним или несколькими официальными центрами сертификации, следует выполнить следующие действия:

1. Создать список «отпечатков пальцев» клиентских сертификатов.
2. Преобразовать список в базу данных.
3. Разрешить таким клиентам пересылку.

Начинаем со сбора открытых сертификатов от почтовых клиентов, которым разрешена пересылка. Для каждого сертификата необходимо извлечь отпечаток MD5.

Примечание

Если вы не хотите делать это вручную, скачайте и запустите сценарий add_ccerts_to_relay_clientcerts.sh, который вычисляет MD5, копирует его в /etc/postfix/relay_clientcerts и создает соответствующую карту из содержимого данного файла.

Пусть у нас есть клиентский сертификат client_public_cert.pem. Извлекаем отпечаток MD5 такой командой:

```
# openssl x509 -noout -fingerprint -in client_public_cert.pem
```

Результат должен иметь такой вид:

```
MD5 Fingerprint=00:8B:02:30:9D:18:F4:81:5D:2F:48:E4:5B:17:82:A7
```

Отпечаток – это строка шестнадцатеричных чисел и двоеточий. Добавляем отпечаток вместе с именем хоста клиента в файл /etc/postfix/relay_clientcerts:

```
00:8B:02:30:9D:18:F4:81:5D:2F:48:E4:5B:17:82:A7 client_1.example.com
18:F4:81:5D:2F:82:A7:48:E4:5B:17:00:8B:02:30:9D client_2.example.org
...
```

TLS-реализации в Postfix требуется только отпечаток, но /etc/postfix/relay_clientcerts – это стандартная карта Postfix, так что в строке должно быть два элемента. Для правой части карты вы можете выбрать любую строку, в данном примере было использовано полностью определенное доменное имя клиента (это упрощает поиск и идентификацию отпечатка в карте).

Совет

В правой части также можно было бы использовать дату окончания действия клиентского сертификата для ускорения или автоматизации поиска просроченных сертификатов.

После добавления отпечатка преобразуем файл relay_clientcerts в карту Postfix командой postmap:

```
# postmap hash:/etc/postfix/relay_clientcerts
```

Эта команда создает /etc/postfix/relay_clientcerts.db, и создание списка завершено.

Теперь необходимо добавить в файл main.cf параметр, который бы сообщал Postfix о том, где искать созданную карту:

```
relay_clientcerts = hash:/etc/postfix/relay_clientcerts
```

Наконец, расширяем разрешения на пересылку, добавляя параметр `permit_tls_clientcerts` в список `smtpd_recipient_restrictions`:

```
smtpd_recipient_restrictions =
  ...
  permit_tls_clientcerts
  ...
```

Помните, что порядок элементов в списке `smtpd_recipient_restrictions` очень важен. Необходимо, чтобы параметр `permit_tls_clientcerts` находился вблизи начала списка.

Настройка завершена, и вам остается лишь перезагрузить конфигурацию Postfix для того, чтобы изменения вступили в силу.

Пересылка на основании CA-сертификата

Если вы хотите осуществлять пересылку только на основании действительного сертификата, то должны обладать полным контролем над клиентскими сертификатами. Вы должны создать собственный центр сертификации (CA) и самостоятельно подписывать клиентские сертификаты, более того, ваш CA должен быть единственным центром, известным вашему серверу Postfix. Это совершенно необходимо, т. к. в данном случае *единственным* критерием, используемым Postfix, является успешная проверка достоверности сертификата.

Предупреждение

Если вы используете официальный CA-сертификат или даже список официальных сертификатов, любой клиент в Интернете может получить сертификат, подписанный одним из этих центров сертификации, и в результате – право на пересылку, так что в итоге ваш сервер превратится в открытый ретранслятор.

Описанные ранее способы организации пересылки требуют создания и поддержки списка сертификатов клиентов, которым разрешена пересылка; преимущество данного способа в том, что для принятия решения вам нужен всего один CA-сертификат.

Для того чтобы Postfix разрешал пересылку клиентам с сертификатами, подписанными вашим личным центром сертификации, сначала сократите список центров сертификации до одного, вашего собственного открытого CA-сертификата. Ранее уже говорилось о том, что CA-файл управляется параметром `smtpd_tls_CAfile`, так что добавляем в файл `main.cf` строку такого вида:

```
smtpd_tls_CAfile = /usr/share/ssl/certs/cacert.pem
```

После того как мы убедились в том, что Postfix распознает только наш собственный сертификат, добавляем параметр `permit_tls_all_clientcerts` в список `smtpd_recipient_restrictions`:

```
smtpd_recipient_restrictions =
  ...
  permit_tls_all_clientcerts
  ...
```

Как всегда перезагружаем конфигурацию Postfix.

Дополнительные меры безопасности для TLS-сервера

До сих пор мы настраивали Postfix для работы с безопасным протоколом TLS. В этом разделе будет показано, как принудить клиенты к использованию протокола TLS и как отказаться от него в случае, если клиент не предоставляет сертификат.

Предупреждение

Будьте внимательны при использовании этих функций, т. к. при применении в несоответствующем окружении они могут разрушить вашу почтовую систему.

Принудительное использование TLS

Вы можете заставить все клиенты использовать TLS. Это удобно в частной сети, когда вам необходима уверенность в том, что весь трафик шифруется (например, в большой компании, имеющей несколько площадок). Для этого устанавливаем параметр `smtpd_enforce_tls` в файле `main.cf` в значение `yes` (по умолчанию – `no`) и перезагружаем конфигурацию Postfix:

```
smtpd_enforce_tls = yes
```

Предупреждение

Документ RFC 2487 говорит о том, что публичный почтовый сервер НЕ ДОЛЖЕН требовать использования ключевого слова `STARTTLS` для локальной доставки почты. Это правило вводится для того, чтобы не препятствовать взаимодействию в SMTP-инфраструктуре сети Интернет.

Вообще, требование TLS для каждого клиента на общедоступном почтовом сервере – это не очень удачная мысль, т. к. те клиенты, которые не могут использовать TLS или не настроены должным образом, оказываются заблокированы. Если вы требуете TLS на публичном почтовом сервере, то будьте готовы к тому, что значительная часть почты не сможет быть доставлена в вашу сеть.

Запрос клиентского сертификата

Вы можете сделать еще один шаг – потребовать от клиентов предоставления сертификатов. В этом случае Postfix не будет взаимодействовать по TLS с клиентом, не приславшим свой сертификат. Добавляем параметр `smtpd_tls_req_ccert` в файл `main.cf` и перезагружаем конфигурацию:

```
smtpd_tls_req_ccert = yes
```

Примечание

Эта настройка не запрещает клиенту использовать незашифрованное SMTP-соединение, если не задан описанный ранее параметр `smtpd_enforce_tls`. Используйте оба параметра вместе для очень строгой политики.

Клиентская часть TLS

Клиентская часть TLS используется, когда Postfix действует как почтовый клиент, который подключается к почтовым серверам, поддерживающим TLS (рис. 18.3). В зависимости от конфигурации для получения разрешения на пересылку Postfix может (выборочно) использовать TLS, отправляя верительные данные SMTP AUTH открытым текстом через TLS или предоставляя собственный клиентский сертификат.

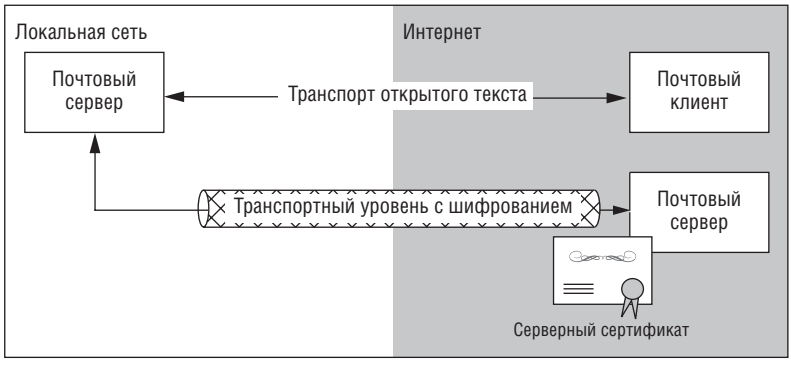

Рис. 18.3. *TLS для почтового клиента Postfix*

Базовая конфигурация клиента

Чтобы включить базовую поддержку TLS для почтового клиента Postfix, вам необходимо задать три параметра в файле `main.cf`. Можно также использовать еще один вспомогательный параметр для отладки сеансов TLS. Вам необходимо выполнить следующие действия:

1. Включить клиентскую часть TLS.
2. Настроить Postfix для проверки серверного сертификата.
3. Подключить Postfix к генератору случайных чисел.
4. Включить журналирование для клиентской части TLS.

Включение клиентской части TLS

Все поддерживающие TLS версии Postfix могут использовать клиентскую часть TLS, но по умолчанию она не включена. Postfix не использует ключевое слово `STARTTLS` при соединении с каким-либо сервером

(даже если сервер вынуждает использовать TLS) до тех пор, пока вы не добавите для клиента в файл `main.cf` такой параметр:

```
smtp_use_tls = yes
```

Когда этот параметр задан, Postfix-демон `smtp` отвечает на `STARTTLS` в SMTP-диалоге, если почтовый сервер на другом конце соединения предлагает `STARTTLS`. Однако для того чтобы все заработало, недостаточно включить клиентскую часть TLS. Клиент Postfix еще не знает, где хранятся CA-сертификаты, которые понадобятся ему для проверки сертификата сервера.

Проверка серверного сертификата

Когда SMTP-клиент Postfix открывает сеанс TLS с почтовым сервером, он пытается проверить подлинность сертификата, предоставленного сервером в этом сеансе. Postfix проверяет криптографическую подпись, которую центр сертификации добавляет в серверный сертификат, используя открытый ключ CA-сертификата. Поэтому у вас должно быть хранилище CA-сертификатов, где Postfix мог бы искать криптографические подписи и проводить их сравнение.

Примечание

В главе 17 говорилось о том, что OpenSSL не имеет центрального корневого хранилища для CA-сертификатов. Следовательно, вам нужно создать новое корневое хранилище. В рассматриваемой далее конфигурации используется `ca-bundle.crt` (см. раздел «Настройка корневого хранилища сертификатов Postfix» ранее в этой же главе). Кроме того, вы, возможно, захотите создать собственный файл корневого хранилища (см. тот же раздел).

Клиент в Postfix, как и сервер, может использовать два типа хранилищ CA-сертификатов: это может быть один файл, содержащий все CA-сертификаты, или каталог, включающий в себя несколько файлов, каждый из которых содержит один CA-сертификат.

Для почтового клиента, работающего в chroot-окружении, необходимо хранить все CA-сертификаты в одном файле, т. к. Postfix читает файлы при запуске перед выполнением chroot.

Если же вы не используете chroot, то можете хранить CA-сертификаты в каталоге, т. к. его проще поддерживать. Это особенно удобно в случае, если вы постоянно добавляете CA-сертификаты, т. к. тогда вам не придется перезапускать Postfix при добавлении каждого нового CA-сертификата. Но не используйте этот способ при работе в chroot-окружении, т. к. вам придется хранить там сертификаты, что сводит на нет основное назначение chroot (хранение секретной информацию за пределами «песочницы»).

Примечание

Как и при настройке серверной части TLS, вы можете применить оба подхода одновременно. Задайте оба параметра, описанных в последующих разделах,

и отделите несколько CA-сертификатов от остальных. Когда Postfix осуществляет поиск CA-сертификата, он сначала читает файл, а затем (если сертификат в файле не найден) обращается к каталогу.

Объединение всех CA-сертификатов в одном файле

Проще всего (и, возможно, лучше всего) хранить все CA-сертификаты в одном файле. Как уже говорилось, если вы работаете с Apache-модулем mod_ssl, то у вас уже есть такой файл – `ca-bundle.crt`. Найдите его, выполнив в командной строке `locate ca-bundle.crt`:

```
$ locate ca-bundle.crt
/usr/share/ssl/certs/ca-bundle.crt
```

Теперь сообщаем клиенту Postfix, что он должен использовать этот файл, указав в параметре `smtp_tls_CAfile` в файле `main.cf` путь к файлу `ca-bundle.crt` и перезагрузив конфигурацию:

```
smtp_tls_CAfile = /usr/share/ssl/certs/ca-bundle.crt
```

Примечание

Если вы поддерживаете собственный центр сертификации, добавьте свой CA-сертификат в корневое хранилище следующей командой:

```
# cat /usr/local/ssl/misc/demoCA/cacert.pem >> /usr/share/ssl/certs/ca-
  bundle.crt
```

Если ваш CA-сертификат является звеном в цепочке сертификатов, добавьте все вышележащие CA-сертификаты из вашей цепочки вплоть до корневого.

Хранение всех CA-сертификатов в каталоге

Для работы с CA-сертификатами, хранящимися в каталоге, укажите в параметре `smtp_tls_CApath` каталог, содержащий файлы сертификатов. Пожалуй, первое, что нужно сделать при установке Postfix, – создать каталог для файлов сертификатов:

```
# mkdir /etc/postfix/certs
```

Теперь помещаем в этот каталог все необходимые CA-сертификаты и создаем индексированную таблицу для их быстрого поиска. Индекс создаем при помощи программы `c_rehash`, поставляемой вместе с OpenSSL. Эта программа строит индекс и создает символические ссылки на CA-сертификаты:

```
# c_rehash /etc/postfix/certs/
Doing /etc/postfix/certs/
cacert.pem => e0dc2d06.0
WARNING: postfix_private_key.pem does not contain a certificate
         or CRL: skipping
postfix_public_cert.pem => 6df723a3.0
```

Примечание

Не забывайте запускать `c_rehash` при каждом добавлении нового CA-сертификата.

После того как вы проделали все эти действия, сообщите Postfix о том, что данный каталог должен использоваться как корневое хранилище CA-сертификатов, затем перезагрузите конфигурацию:

```
smtp_tls_CApath = /etc/postfix/certs
```

Подключение клиента Postfix к генератору случайных чисел

Для корректной инициализации процесса шифрования необходимо подключить Postfix к источнику случайных чисел. Настройка клиента выполняется аналогично настройке сервера (см. описание параметра `tls_random_source` в разделе «Подключение Postfix к генератору случайных чисел» выше в этой главе).

Журналирование действий клиентской части TLS

Прежде чем запускать Postfix для тестирования клиента TLS, увеличьте значение параметра `smtp_tls_loglevel` до 2, чтобы увидеть существенные TLS-события (значение по умолчанию – 0):

```
smtp_tls_loglevel = 2
```

Различные уровни журналирования рассмотрены при описании параметра `smtpd_tls_loglevel` в разделе «Повышение уровня журналирования TLS» ранее в этой главе.

Тестирование базовой клиентской функциональности

Для тестирования базовой клиентской функциональности TLS необходимо выполнить всего два действия:

1. Проверить в файле журнала, обнаружил ли какие-то ошибки клиент Postfix.
2. Отправить сообщение серверу, поддерживающему TLS.

Поиск ошибок в файле журнала

Для выявления проблем с TLS в файле журнала Postfix запускаем команду `egrep`:

```
$ egrep '(reject|error|warning|fatal|panic): ' /var/log/maillog
```

Если Postfix обнаружил какие-то неполадки, связанные с TLS, данная команда разыщет упоминания о них. Если все сделано правильно, никаких TLS-проблем быть не должно, но если они все же появились, проверьте, нет ли опечаток в файле конфигурации и корректно ли выданы разрешения на чтение сертификатов.

Отправка сообщения TLS-серверу

Теперь попробуем отправить сообщение TLS-серверу, чтобы посмотреть, использует ли TLS клиент Postfix.

Если вам не известны никакие TLS-серверы, имейте в виду, что создатель патча TLS Лутц Джанике поддерживает общедоступный почтовый сервер, который можно использовать для тестирования. Отправьте сообщение на адрес `postfix_tls-bounce@serv01.aet.tu-cottbus.de`, и сервер должен вернуть вам его обратно, добавив в исходное сообщение заголовки, сообщающие о том, было ли сообщение передано по TLS. Заголовок должен выглядеть примерно так:

```
Received: from mail.state-of-mind.de (mail.state-of-mind.de [212.14.92.89])
        (using TLSv1 with cipher EDH-RSA-DES-CBC3-SHA (168/168 bits))
        (Client did not present a certificate)
        by serv01.aet.tu-cottbus.de (Postfix) with ESMTP id 74C6B2330
        for <postfix_tls-bounce@serv01.aet.tu-cottbus.de>; Wed, 10 Dec 2003
        23:50:45 +0100 (MET)
```

Выборочное использование TLS

Выборочное использование TLS в клиентской части означает, что вы можете ввести политику безопасности для некоторых серверов, при этом не допуская исчезновения сообщений в случае неправильной настройки почтового сервера, предлагающего `STARTTLS`.

Примечание

Это достаточно часто случается с серверами Lotus Notes.

Необходимо выполнить три действия:

1. Включить выборочное использование TLS в своей конфигурации Postfix.
2. Создать карту политик, которая сообщает SMTP-клиенту, когда следует использовать TLS.
3. Настроить Postfix на распознавание того, предоставляют ли почтовые серверы TLS.

Включение выборочного использования TLS

Включите выборочное использование клиентской части TLS в файле `main.cf`, указав в параметре `smtp_tls_per_site` карту политик (в этом примере карта называется `/etc/postfix/tls_per_site`):

```
smtp_tls_per_site = hash:/etc/postfix/tls_per_site
```

Создание карты политик TLS

Карта политик TLS выглядит так же, как любая другая карта Postfix: каждая строка представляет собой запись, содержащую пару «ключ –

значение». С левой стороны помещаем имя хоста или домена (ключ), а с правой – политику TLS (значение). Для SMTP-клиента Postfix возможны следующие политики:

`NONE`

Отключает клиентскую часть TLS.

`MAY`

Позволяет клиенту использовать TLS, если удаленный сервер предлагает ключевое слово `STARTTLS`, но не заставляет его делать это в случае нежелания.

`MUST`

Заставляет клиент Postfix использовать TLS, если данный сервер предлагает TLS посредством ключевого слова `STARTTLS`. Кроме того, Postfix проверяет параметр `CommonName` серверного сертификата на соответствие полностью определенному доменному имени сервера.

`MUST_NOPEERMATCH`

Облегченная версия политики `MUST`. Клиент Postfix отвечает на `STARTTLS` и проверяет серверный сертификат, но игнорирует возможные различия параметра `CommonName` и полностью определенного доменного имени.

Примечание

Если вы настраиваете Postfix для использования карты политик TLS, то настройки карты всегда будут иметь приоритет перед значениями параметров в вашем файле `main.cf`. Если вы отключите протокол TLS, он все равно будет использоваться для хостов, присутствующих в карте. И наоборот, если вы включите TLS в файле `main.cf`, а хост не будет найден в карте политик, то TLS будет использоваться.

Начинаем создание карты с файла `/etc/postfix/tls_per_site`, который может выглядеть так:

```
dom.ain              NONE
host.dom.ain         MAY
important.host       MUST
some.host.dom.ain    MUST_NOPEERMATCH
```

Записав карту в формате ASCII, строим хеш-карту при помощи команды `postmap` – теперь карта будет доступна для Postfix:

```
# postmap hash:/etc/postfix/tls_per_site
```

Выявление TLS-серверов

Получение данных о серверах, предоставляющих TLS, полезно не только при отладке TLS-сеанса, но и при настройке выборочного использования TLS в Postfix. В файле `main.cf` установите в параметре `smtp_tls_note_starttls_offer` значение `yes`:

```
smtp_tls_note_starttls_offer = yes
```

Теперь при подключении вашего клиента Postfix к почтовому серверу, предлагающему STARTTLS, клиент записывает в почтовый журнал имя сервера следующим образом:

```
client postfix/smtp[1504]: Host offered STARTTLS: [mail.example.com]
```

Ваш TLS-клиент готов к работе.

Настройка производительности клиента

К производительности клиента относятся те же соображения, что изложены в разделе «Настройка производительности сервера» для TLS-сервера Postfix. Клиент использует для кэширования ключей сеанса тот же самый демон tlsmgr. Однако для клиента несколько изменяются названия параметров конфигурации: следует заменить smtpd на smtp.

Поэтому для включения кэширования проделайте все операции, описанные в разделе «Настройка производительности сервера», только используйте в файле main.cf параметры smtp_tls_session_cache_database и smtp_tls_session_cache_timeout:

```
smtp_tls_session_cache_database = sdbm:/etc/postfix/smtp_scache
smtp_tls_session_cache_timeout = 3600s
```

Безопасность клиентской части SMTP AUTH

В разделе «Серверные меры безопасности при предоставлении SMTP AUTH» вы узнали о том, как обезопасить процедуру SMTP AUTH на стороне сервера при помощи TLS. В этом разделе будет показано, как сделать это на стороне клиента. Вспомним, о чем шла речь: вам не хочется, чтобы ваш клиент отправлял имя пользователя и пароль посредством механизмов открытого текста SMTP AUTH по незашифрованному соединению. Если необходимо использовать именно механизмы открытого текста, клиент должен сначала открыть сеанс TLS.

Используя параметр smtp_sasl_security_options для незашифрованных соединений и параметр smtp_sasl_tls_security_options для сеансов TLS, вы можете ограничить SMTP AUTH:

```
smtp_sasl_security_options = noanonymous, noplaintext
smtp_sasl_tls_security_options = noanonymous
```

Первое правило запрещает использование анонимного механизма и механизмов открытого текста при отсутствии шифрования на транспортном уровне, а второе разрешает механизмы открытого текста при общении с сервером в рамках сеанса TLS.

Пересылка на основании сертификатов: клиентская часть

Пересылка на основании сертификатов – это надежный способ обеспечения пересылки сообщений для клиентов даже в том случае, если клиенты находятся в неизвестной серверу сети. О настройке сервера было рассказано в разделе «Пересылка на основании сертификатов: серверная часть». Однако настройка пересылки для клиентской части больше похожа не на настройку пересылки на сервере, а скорее на настройку самого сервера, т. к. вам нужно указать пути к сертификату, который клиент предоставляет ядру, и к ключу, который клиент будет использовать для инициирования соединения.

Указание путей к клиентскому сертификату и ключу

Для того чтобы клиент Postfix в начале TLS-сеанса предоставлял серверу сертификат, необходимо в файле `main.cf` указать в параметре `smtp_tls_cert_file` клиентский сертификат, а в параметре `smtp_tls_key_file` – ключ клиента, например:

```
smtp_tls_cert_file = /etc/postfix/certs/postfix_public_cert.pem
smtp_tls_key_file = /etc/postfix/certs/postfix_private_key.pem
```

Примечание

Если вы также хотите настроить серверную часть TLS для своего сервера Postfix, используйте повторно серверный сертификат и ключ, если только вам не нужно, чтобы сервер и клиент Postfix обладали разными «цифровыми личностями».

Теперь перезагружаем конфигурацию Postfix и начинаем тестирование.

Тестирование пересылки на основании сертификатов на стороне клиента

Тестирование состоит из трех этапов:

1. Проверка файла журнала на предмет очевидных ошибок.
2. Проверка того, что клиент отправляет свой сертификат.
3. Проверка того, что клиент может осуществлять пересылку на основании своего сертификата.

Проверка файла журнала

Как вы уже знаете, проверку файла журнала мы осуществляем при помощи команды `egrep`:

```
$ egrep '(reject|error|warning|fatal|panic):' /var/log/maillog
```

Если появляются какие-то ошибки, следуйте стандартному совету: проверьте, нет ли опечаток в файле конфигурации и убедитесь в том, что у клиента есть разрешение на чтение сертификатов.

Проверка отправки клиентского сертификата

Для того чтобы удостовериться в том, что клиент отправляет свой сертификат, отправляем сообщение TLS-серверу и смотрим, примет ли он клиентский сертификат. Если вы отправляете клиентский сертификат почтовому серверу Postfix, для которого значение параметра `smtpd_tls_received_header = yes`, в ваших заголовках появится нечто подобное:

```
Received: from client.example.com (client.example.com [172.16.0.3])
      (using TLSv1 with cipher EDH-RSA-DES-CBC3-SHA (168/168 bits))
      (Client CN "client.example.com", Issuer "mail.example.com" (verified OK))
      by mail.example.com (Postfix) with ESMTP id 63AC77247
      for <tls-bounce@mail.example.com>; Thu, 11 Dec 2003 19:48:38 +0100 (CET)
```

В третьей строке говорится о том, что почтовый клиент `Client` отправил почтовому серверу сертификат. Сертификат был подписан центром сертификации `mail.example.com`, и сервер смог проверить это.

Примечание

Если что-то идет не так и вы не знаете, сервер или клиент тому причиной, отправьте сообщение по адресу `postfix_tls-bounce@serv01.aet.tu-cottbus.de`. Как уже говорилось, этот сервер возвращает сообщение обратно, включив в него отладочную информацию TLS.

Проверка возможности пересылки на основании клиентского сертификата

Теперь проверяем способность клиента осуществить пересылку сообщения на основании предоставленного сертификата, отправив сообщение через TLS-сервер. Убеждаемся в том, что сервер пересылает сообщения на основании сертификатов, проверяя следующие условия:

1. Клиент не входит в сеть сервера и в любую сеть, которой сервер предоставляет доступ к пересылке на основании какого-то другого критерия, например значения параметра `mynetworks`.
2. Клиент не пользуется такими возможностями, как SMTP AUTH.
3. Получатель не включен в список `relayhosts` для мест конечного назначения.

Дополнительные меры безопасности для TLS-клиента

Вы можете заставить клиент использовать TLS или принять еще более строгие меры, разрешив взаимодействие с сервером лишь в том случае, если клиент может проверить полное имя сервера. Принуждение к использованию TLS со стороны клиента полезно только в том случае, когда вы можете управлять серверами, с которыми взаимодействует клиент.

Предупреждение

Некорректная настройка данной функциональности может нарушить работу вашей системы отправки почты.

Для принудительного использования протокола TLS должны быть выполнены следующие условия:

1. Сервер должен предоставлять TLS.
2. Значения `CommonName` в сертификате должны совпадать с полностью определенным доменным именем сервера.
3. Клиент должен иметь возможность проверить сертификат сервера, используя подпись центра сертификации.

Если не выполнено хотя бы одно из этих условий, то клиент не будет отправлять сообщение серверу. Вместо этого клиент оставит сообщение в своей очереди и запишет в почтовый журнал сообщение об ошибке `4xx`.

Примечание

Принуждение SMTP-клиента к использованию TLS полезно в частных сетях и в случае, если вы знаете, что ваш клиент пересылает все сообщения через один сервер.

Для принудительного использования клиентской части TLS задайте в файле `main.cf` параметр `smtp_enforce_tls` следующим образом:

```
smtp_enforce_tls = yes
```

Если такой режим слишком строг для каждодневного применения, вы можете разрешить передачу в случае несовпадения параметра `CommonName` в серверном сертификате с полностью определенным доменным именем сервера. Для этого установите в параметре `smtp_tls_enforce_peername` значение `no` (обычно этот параметр включен, если установлен параметр `smtp_enforce_tls`):

```
smtp_tls_enforce_peername = no
```

Предупреждение

Такая настройка создает риск проведения атаки с внедрением посредника.

19

Корпоративный почтовый сервер

В этой главе будет рассказано о том, как создать полную почтовую систему на основе Postfix, Cyrus SASL, Courier maildrop и Courier IMAP. Эти компоненты будут получать данные конфигурации и аутентификации от сервера OpenLDAP, поддерживающего службы каталогов.

Мы будем двигаться от базовой конфигурации к более сложной. На начальном этапе все приложения будут подключены к центральному серверу LDAP. Затем можно будет переходить к более сложным вопросам. В расширенной конфигурации мы добавим протокол безопасности транспортного уровня и покажем, как предоставить SMTP-аутентификацию на основе LDAP-запросов.

Прежде чем приступать к реализации рассматриваемого здесь корпоративного почтового сервера, необходимо хорошо разобраться со схемами LDAP и OpenLDAP. Если вы ранее не встречались с OpenLDAP, то можете начать с чтения руководства администратора OpenLDAP («OpenLDAP Administrator's Guide») по адресу *http://www.openldap.org/doc/admin22*.

Общая идея

На рис. 19.1 изображены приложения, с которыми нам придется иметь дело в этой главе, и связи между ними. Как видите, в самом центре всех служб находится OpenLDAP. Серверы приложений работают следующим образом:

- Postfix передает данные аутентификации LDAP-серверу, когда почтовые клиенты пытаются осуществить пересылку, используя SMTP AUTH. Кроме того, Postfix запрашивает у LDAP-сервера информацию о локальном пользователе и псевдониме при поступлении входящей почты. При приеме сообщения Postfix передает его агенту Courier maildrop.

- Агент Courier maildrop отвечает за локальную доставку. Он узнает у LDAP-сервера местоположение почтового ящика, а также просматривает правила фильтрации (например, чтобы поместить сообщения, помеченные как спам, в специальную папку с именем `.spam`).
- Пользователь подключается к серверу Courier IMAP для извлечения почты. Этот сервер запрашивает у LDAP-сервера верительные данные пользователя. LDAP также сообщает Courier, где следует искать почтовый ящик и какие UID и GID использовать для доступа к нему.

Структура каталога[1] LDAP

Первое, что нужно сделать для построения почтовой системы, – это создать дерево каталогов LDAP. Это может оказаться весьма сложной задачей. Одной из причин недостаточно широкого распространения LDAP является именно сложность создания каталога с нуля. При разработке структуры каталога, схем и атрибутов необходимо учитывать три ключевых фактора:

- Назначение каталога.
- Вашу организационную структуру.
- Требования серверов, которые используют каталог LDAP.

Основное назначение каталога в данной главе заключается в том, чтобы показать вам, как Postfix и другие серверы обращаются к серверу LDAP. Мы постарались сделать структуру как можно более простой с тем, чтобы вы могли сосредоточиться на конфигурировании приложений, не увязая в постижении структуры каталогов.

В этой главе мы будем создавать почтовую систему для компании Example Inc. Эта компания достигла первых успехов в продажах и расширилась настолько, что пришлось разделить ее на несколько отделов. Среди них отдел информационных технологий (которым руководит начинающий администратор по имени Bamm Bamm), отдел продаж и отдел закупок. Для простоты мы будем рассматривать лишь эти три подразделения.

Служба каталогов будет обеспечивать компоненты Postfix, Cyrus SASL, Courier maildrop и Courier IMAP данными о пользователях и конфигурации. В примере будет использоваться схема `authldap.schema`, которая поставляется вместе с Courier IMAP, т. к. она без проблем может использоваться другими серверами.

---

[1] Если вы еще не знакомы с LDAP, вам может показаться непонятным термин «каталог». По сути, сервер LDAP имеет дело с древовидной базой данных, в которой может храниться какая угодно информация. Эта база данных и называется «каталогом», а ее структура определяется так называемой схемой каталога, то есть определенным набором полей и взаимосвязей между ними. – *Примеч. науч. ред.*

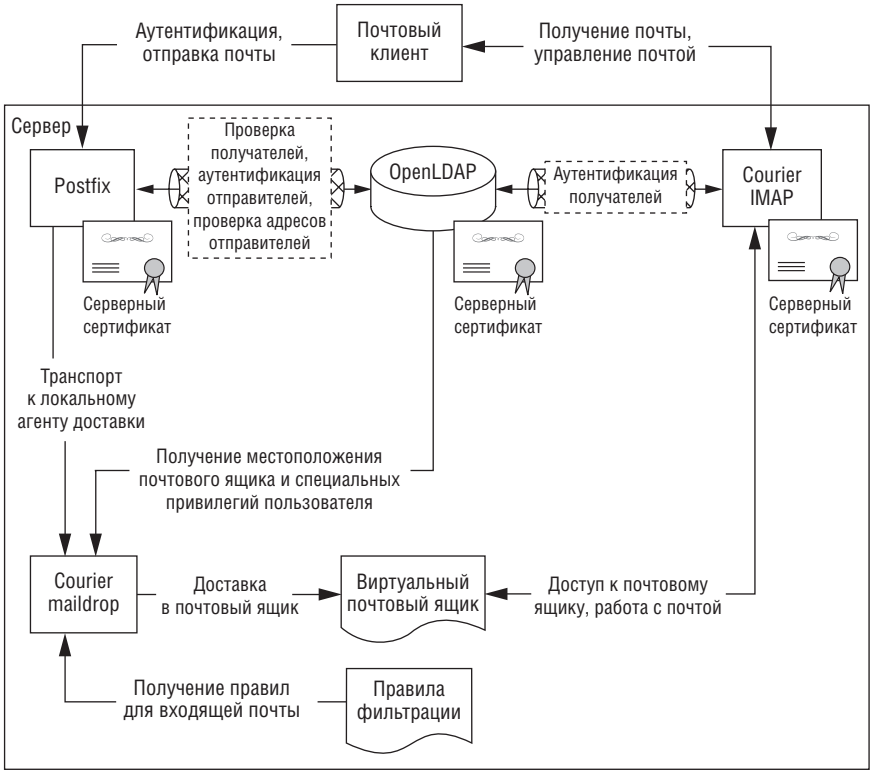

***Рис. 19.1.** Архитектура корпоративного почтового сервера*

Примечание

Если вы не хотите создавать собственный каталог, то можете скачать дамп LDIF (LDAP Data Interchange Format) для Example Inc. с веб-сайта *http://www.postfix-book.com*.

На рис. 19.2 изображено дерево каталогов, начинающееся с узла `dc=example,dc=com`, от которого отходят две большие ветки.

Левая ветвь содержит учетные записи аутентификации для серверов, к этой теме мы обратимся чуть позже в разделе «Тонкая настройка», когда будем говорить о защите данных LDAP и соединения.

Правая ветвь называется `ou=people,dc=example,dc=com`. Она содержит подузлы, например, представляющий организационную структуру подузел `ou=it,ou=people,dc=example,dc=com`. Двигаясь вниз, вы встретите другие подузлы, которые хранят пользовательские объекты с атрибутами и значениями, содержащими всю информацию, необходимую для предоставления всех пользовательских данных для всей почтовой системы.

Примечание

При помощи службы каталогов можно сделать еще очень много. Например, для Postfix вы можете добавить значения настроечных параметров `mydestination`, `relayhost`, `virtual_domain` и т. д.

Выбор атрибутов в схеме Postfix

Если, будучи новичком в мире LDAP, вы надеетесь на простое и быстрое решение, то у нас для вас плохие новости. Не существует схемы Postfix, которую можно было бы выбрать, заполнить данными и использовать. Возможно, вы будете еще более обескуражены, когда узна-

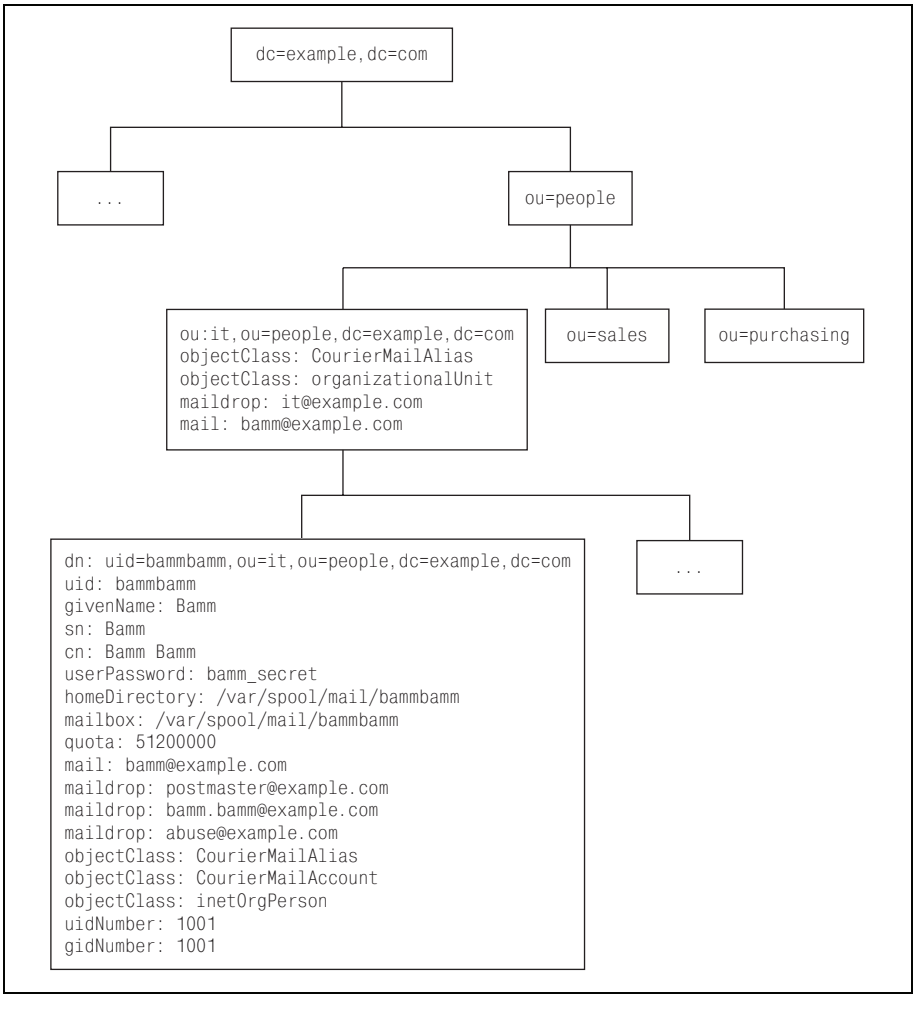

Рис. 19.2. Организационная ветвь для компании Example Inc.

ете, что это сделано нарочно, но на то есть серьезные причины. Практически все, что необходимо Postfix для данного решения, присутствует в других схемах, таких как `core.schema` в OpenLDAP, или в приложениях, подобных Courier IMAP, который поставляется с собственной схемой `authldap.schema`.

При выборе нужных схем и атрибутов для своих серверов необходимо изучить требования серверов. Например, если Postfix использует LDAP для карт, вы можете создать следующие сущности:

Места назначения, сети и хосты

Места назначения, сети и хосты представлены именами хостов и IP-адресами. Postfix просматривает их, чтобы определить, для каких доменов он должен принимать и пересылать почту, кроме того, он может использовать их при применении ограничений на хосты и сети.

Ваш каталог должен содержать атрибуты, которые описывают хосты и сети и, возможно, разрешают добавление нескольких значений в один LDAP-объект. Вы найдете соответствующие атрибуты в схемах для сетей, таких как примеры схем CORE и NIS.

Получатели и отправители

Просматривая получателей и отправителей, Postfix по умолчанию ищет *имя_пользователя@имя_хоста*. Существует множество схем, содержащих такие атрибуты, например `core.schema` и другие схемы для почты (для Sendmail или qmail). Однако не забывайте о той части почтовой системы, которая занимается доставкой. Многие серверы POP и IMAP имеют собственные схемы, и все они содержат атрибуты для определения адресов отправителей и получателей.

Псевдонимы

Пользователь может иметь несколько псевдонимов. Выбранный атрибут должен разрешать множественное добавление для объекта. В большинстве схем для почты есть атрибут псевдонима. Вполне вероятно, что подойдет схема, которую вы планируете использовать для адресов получателей и отправителей.

Списки

Списки состоят из одной записи для псевдонима и множества адресов получателей. Если у вас есть схема с обоими атрибутами, то у вас есть все, что нужно для списка.

Прежде чем браться на создание собственной схемы Postfix, попробуйте адаптировать к Postfix схемы для других серверов. Такой подход уменьшает нагрузку на вашу службу LDAP и ее сложность, а также обеспечивает большую гибкость при расширении каталога и увеличении количества использующих каталог серверов.

Как уже говорилось, в основе почтовой системы, создаваемой в этой главе, будет лежать схема `authldap.schema`, которая поставляется вме-

сте с Courier IMAP, т. к. в ней содержится практически все, что необходимо для полного каталога.

Проектирование ветвей

Мы разделим каталог на две основные ветви (см. рис. 19.2). Левая ветвь содержит учетные записи приложений, которые мы будем использовать позже в разделе «Тонкая настройка» при реализации контроля доступа для приложений, обращающихся к каталогу.

Правая ветвь будет содержать пользовательскую информацию. Она будет разбита на более мелкие элементы в соответствии со структурой отделов. Мы будем использовать объект `organizationalUnit` для создания таких элементов и в дальнейшем будем конфигурировать объекты `organizationalUnit` для хранения информации простых списков рассылки.

Ветви готовы, и мы приступаем к созданию пользовательского объекта.

Создание пользовательских объектов

Вы можете создавать пользовательские объекты из трех классов: `inetOrgPerson`, `CourierMailAccount` и `CourierMailAlias`, которые вы найдете в схемах `inetorgperson.schema`, `authldap.schema` (Courier IMAP) и `nis.schema`. Использование схемы `nis.schema` необходимо, т. к. `authlap.schema` зависит от некоторых ее атрибутов.

Используя атрибуты всех трех схем, будем описывать один пользовательский объект. Полный объект для пользователя Bamm Bamm из отдела информационных технологий, который мы будем использовать в этой главе, выглядит следующим образом:

```
dn: uid=bammbamm,ou=it,ou=people,dc=example,dc=com
uid: bammbamm
givenName: Bamm
sn: Bamm
cn: Bamm Bamm
userPassword: bamm_secret
homeDirectory: /var/spool/mail/bammbamm
mailbox: /var/spool/mail/bammbamm/Maildir
quota: 51200000S
mail: bamm@example.com
maildrop: postmaster@example.com
maildrop: bamm.bamm@example.com
maildrop: abuse@example.com
objectClass: CourierMailAlias
objectClass: CourierMailAccount
objectClass: inetOrgPerson
uidNumber: 1003
gidNumber: 1003
```

В последующих разделах вы узнаете о том, откуда берутся все эти атрибуты.

Создание отправителя и получателя

Необходимо создать объект, содержащий все пользовательские атрибуты и значения. Мы будем использовать схему `inetorgperson.schema`, т. к. она содержит дополнительные атрибуты, позволяющие вести адресную книгу в масштабе всей компании.

Вы можете создать уникальный пользовательский объект при помощи атрибута `uid`. Схема `inetorgperson.schema` также обеспечивает доступ к атрибуту `mail`, который можно использовать для адресов локальных получателей и адресов действительных отправителей. Эти два атрибута выглядят в схеме `inetorgperson.schema` следующим образом:

```
attributetype ( 0.9.2342.19200300.100.1.1
    NAME ( 'uid' 'userid' )
    DESC 'RFC1274: user identifier'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} )
attributetype ( 0.9.2342.19200300.100.1.3
    NAME ( 'mail' 'rfc822Mailbox' )
    DESC 'RFC1274: RFC822 Mailbox'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )
```

Примечание

Атрибут `mail` может быть добавлен в объект несколько раз. Далее вы увидите, что это удобно для создания списков рассылки.

Допустим, мы хотим создать объект. В этой главе будем использовать в качестве атрибута `uid` объединение имени и фамилии пользователя.

Определение псевдонимов

Подходящий атрибут для определения псевдонимов можно обнаружить в схеме `authldap.schema` из Courier IMAP. Вспомогательный (auxiliary) класс `CourierMailAlias` с атрибутом `maildrop` определяет в соответствии с RFC 822 почтовый ящик для почтового псевдонима:

```
attributetype ( 1.3.6.1.4.1.10018.1.1.4 NAME 'maildrop'
        DESC 'RFC822 Mailbox - mail alias'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreIA5SubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )
```

Примечание

Класс объекта `CourierMailAlias` является вспомогательным классом, т. е. не может быть добавлен в словарь сам по себе. Необходимо добавлять его в сочетании со структурным классом. Нам это подходит, т. к. уже используемый нами класс `inetOrgPerson` является структурным.

Создание объектов списков

Простейший вид списков, поддерживаемый Postfix без помощи диспетчеров списков (таких как Mailman), представляет собой псевдонимы, соответствующие списку получателей. К этому моменту у нас уже есть все необходимые атрибуты, так что осталось лишь придумать какой-то подходящий объект списка – это может быть все, что угодно, без атрибута `userPassword`.

На самом деле создавать дополнительный объект для списков не придется. Просто добавьте класс объекта `CourierMailAlias` в класс `organizationalUnit`, используемый для создания исходных ветвей.

Объект `CourierMailAlias` предоставляет нам доступ к атрибутам `maildrop` и `mail`. Теперь мы можем назначить имя псевдонима, например `all@example.com`, атрибуту `maildrop` узла `ou=people,dc=example,dc=com` и добавить почтовые записи для каждого члена организации. Полный объект списка мог бы выглядеть так:

```
dn: ou=people,dc=example,dc=com
ou: people
description: All employees
objectClass: CourierMailAlias
objectClass: organizationalUnit
maildrop: all@example.com
mail: bamm@example.com
mail: pebble@example.com
mail: mcbricker@example.com
mail: flintstone@example.com
mail: rubble@example.com
```

Теперь у LDAP-сервера есть получатели и псевдонимы, необходимые Postfix, так что мы можем заняться другими серверами.

Добавление атрибутов для остальных серверов

Когда сервер Postfix заканчивает обработку электронного сообщения, он отправляет его локальному агенту доставки (LDA), такому как Courier maildrop. LDA должен знать местоположение почтового ящика и пользователя и привилегии, которые он должен использовать. Агенты передачи сообщений[1], такие как Courier IMAP, должны также знать, где находится почтовый ящик.

Указываем местоположение почтового ящика при помощи атрибута `mailbox` из схемы Courier `authldap.schema`:

---

[1] Здесь автор немного заблуждается: Courier IMAP является агентом доступа к почтовому ящику (mail access agent – MAA), а не агентом передачи сообщений. Расположение почтового ящика действительно должно быть известно агенту передачи сообщений (MTA), но только если он выполняет функции и локального агента доставки, и MTA, как, например, конкурент Postfix – программа sendmail (*www.sendmail.org*). – *Примеч. науч. ред.*

```
attributetype ( 1.3.6.1.4.1.10018.1.1.1 NAME 'mailbox'
    DESC 'The absolute path to the mailbox for a mail account in
                                    a non-default location'
        EQUALITY caseExactIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

Эта схема также определяет атрибут `quota`, который может указывать максимальный размер почтового ящика:

```
attributetype ( 1.3.6.1.4.1.10018.1.1.2 NAME 'quota'
    DESC 'A string that represents the quota on a mailbox'
        EQUALITY caseExactIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

Однако при доступе к почтовым ящикам Courier должен использовать атрибуты, за которыми `authldap.schema` отсылает к схеме `nis.schema`:

```
attributetype ( 1.3.6.1.1.1.1.0 NAME 'uidNumber'
    DESC 'An integer uniquely identifying a user in an administrative domain'
    EQUALITY integerMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )
attributetype ( 1.3.6.1.1.1.1.1 NAME 'gidNumber'
    DESC 'An integer uniquely identifying a group in an administrative
                                                    domain'
    EQUALITY integerMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )
attributetype ( 1.3.6.1.1.1.1.3 NAME 'homeDirectory'
    DESC 'The absolute path to the home directory'
    EQUALITY caseExactIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

Атрибуты `uidNumber` и `gidNumber` хранят значения идентификатора пользователя почтового ящика и идентификатора группы. Эти номера нужны Courier maildrop для получения корректных прав на запись сообщений в почтовый ящик, они также необходимы Courier IMAP для записи и удаления сообщений. Кроме того, Courier maildrop требует использования атрибута `homeDirectory` для чтения правил фильтрации.

Базовая конфигурация

В этом разделе будет показано, как интегрировать поддержку LDAP в Postfix и другие серверы. Будет рассматриваться только базовая функциональность. О защите данных мы поговорим в разделе «Тонкая настройка» далее в этой главе.

Настройка Cyrus SASL

Одна из странностей в установке программного обеспечения для корпоративного почтового сервера заключается в том, что плагин `ldapdb` Cyrus SASL требует наличия библиотек OpenLDAP для разработки. Однако для того, чтобы OpenLDAP мог взаимодействовать с Cyrus SASL, ему необходимы библиотеки Cyrus SASL для разработки. И ес-

ли вы хотите установить оба продукта из исходных текстов, то такая взаимная зависимость может создать сложности.

Примечание

Плагин `ldapdb` требует OpenLDAP версии не ниже 2.1.27 или 2.2.6. Если у вас уже установлена подходящая версия OpenLDAP, поддерживающая SASL, то при сборке SASL с поддержкой `ldapdb` вам нужно будет лишь установить библиотеки для разработки OpenLDAP.

Для того чтобы обойти эту проблему, придется собрать и установить Cyrus SASL дважды. Первый раз делаем это без `ldapdb`, чтобы OpenLDAP мог собраться с библиотеками SASL. Далее в этой главе мы пересоберем Cyrus SASL с только что установленной библиотекой OpenLDAP, чтобы вы получили плагин `ldapdb`. Если Cyrus SASL не требуется другим приложениям на вашем сервере, вы можете использовать следующую команду конфигурации для получения минимального SASL, необходимого для сборки OpenLDAP:

```
# ./configure \
   --with-plugindir=/usr/lib/sasl2 \
   --disable-java \
   --disable-krb4 \
   --with-dblib=berkeley \
   --with-saslauthd=/var/state/saslauthd \
   --without-pwcheck \
   --with-devrandom=/dev/urandom \
   --enable-cram \
   --enable-digest \
   --enable-plain \
   --enable-login \
   --disable-otp
```

Теперь займемся сборкой OpenLDAP.

Настройка OpenLDAP

Если в вашей системе еще не установлен OpenLDAP, возьмите версию более новую, чем 2.1.27 или 2.2.6 (эта версия использует другую BerkeleyDB) из пакета или скачайте исходные тексты по адресу *http://www.openldap.org/software/download*. Если вы будете собирать версию на основе исходных текстов, обратитесь к разделу «Установка OpenLDAP из исходных текстов».

Установка OpenLDAP из исходных текстов

От имени обычного пользователя распакуйте архив и перейдите в созданный каталог. Запустите команду `configure`, указав (как минимум) перечисленные ниже параметры:

```
$ ./configure  --prefix=/usr --exec-prefix=/usr --bindir=/usr/bin \
  --sbindir=/usr/sbin sysconfdir=/etc --datadir=/usr/share \
  --includedir=/usr/include --libdir=/usr/lib --libexecdir=/usr/libexec \
  --localstatedir=/var sharedstatedir=/usr/com --mandir=/usr/share/man \
  --infodir=/usr/share/info --with-slapd --with-slurpd --without-ldapd \
  with-threads=posix --enable-static --enable-dynamic --enable-local \
  --enable-cldap --enable-rlookups --with-tls with-cyrus-sasl \
  --enable-wrappers --enable-passwd --enable-cleartext --enable-crypt \
  --enable-spasswd --enable-modules disable-sql --libexecdir=/usr/sbin \
  --localstatedir=/var/run --enable-ldbm --with-ldbm-api=berkeley \
  --enable-bdb --enable-ldap enable-meta --enable-monitor \
  --enable-null --enable-rewrite --disable-shared --with-kerberos=k5only
```

После завершения работы сценария конфигурации выполните `make depend`, `make` и `make test`, переключитесь на пользователя `root` и выполните `make install`. Вы готовы к настройке OpenLDAP.

Настройка сервера LDAP

Для настройки OpenLDAP-сервера `slapd` перейдите в каталог конфигурации (например, `/etc/openldap`) и отредактируйте файл `slapd.conf`. Следует добавить в него следующие строки:

```
# SCHEMATA
include     /etc/openldap/schema/core.schema
include     /etc/openldap/schema/cosine.schema
include     /etc/openldap/schema/inetorgperson.schema
include     /etc/openldap/schema/misc.schema
include     /etc/openldap/schema/nis.schema
include     /etc/openldap/schema/authldap.schema
# RUNTIME
pidfile     /usr/var/slapd.pid
argsfile    /usr/var/slapd.args
# DATABASE DEFINITIONS
database    ldbm
suffix      "dc=example,dc=com"
rootdn      "cn=Manager,dc=example,dc=com"
rootpw      {CRYPT}SHXa4LHVH8y3A
directory   /usr/var/openldap-data
# INDICES
index   objectClass                     eq
index   cn                              eq
index   mail,maildrop                   pres
index   mailbox,quota,uidNumber,gidNumber eq
```

В разделе `SCHEMATA` файла `slapd.conf` указываются схемы, которые должны загружаться при запуске.

В разделе `DATABASE DEFINITIONS` определено, что `"dc=example,dc=com"` будет суффиксом для верхней ветки дерева каталогов. Присваиваемый набор значений атрибутов `rootdn` (`"cn=Manager,dc=example,dc=com"`) и значение `rootpw` дает пользователю, определяемому `rootdn`, доступ в ката-

лог на чтение и запись. Используйте команду `slappasswd(8)` для создания зашифрованного пароля.

Раздел `INDICES` определяет, какие атрибуты должны быть индексированы. Наличие у атрибутов индекса значительно ускоряет их просмотр.

Управление SASL-аутентификацией в OpenLDAP

OpenLDAP с поддержкой SASL может использовать SASL в процессе аутентификации для связывания пользователей с сервером. Механизмы открытого текста предлагаться не будут, но GSSAPI может быть обеспечен (среди прочих).

Поскольку мы не будем настраивать полнофункциональный сервер Kerberos исключительно для использования данного механизма, лучше отключить его, чтобы клиенты не пытались его использовать. Для этого указываем в специальном конфигурационном файле SASL для OpenLDAP только те механизмы, которые хотим использовать. Создаем файл `/usr/lib/sasl2/slapd.conf` с такой настройкой:

```
mech_list: DIGEST-MD5
```

Подробнее об SMTP-аутентификации см. в главе 15.

Импорт каталога

Пришло время наполнять базу данных LDAP данными. Существует множество способов это сделать, но в нашем случае `slapd`, вероятно, еще не работает. Это означает, что мы можем воспользоваться для файла LDIF утилитой `slapadd`:

```
# slapadd -v -c -b "dc=example,dc=com" -l example.com.ldif
```

Предупреждение

Не используйте `slapadd`, когда slapd запущен. Эта утилита осуществляет запись напрямую в базу данных и может вызвать аварию в работе `slapd` и повреждение базы данных.

После успешного импорта файла LDIF запускаем `slapd`. В случае возникновения проблем с импортом обратитесь к slapadd(8), чтобы посмотреть, не нужны ли вам какие-то еще параметры (если же дела совсем плохи, придется заняться отладкой).

Настройка клиента LDAP

Для проверки конфигурации `slapd` и обеспечения взаимодействия с Courier IMAP необходимо настроить клиент OpenLDAP. Обычно требуется изменить несколько параметров в файле `/etc/openldap/ldap.conf`. Для начала достаточно включить следующие параметры:

```
URI ldap://mail.example.com
BASE dc=example,dc=com
```

Параметры URI и BASE указывают, к какому LDAP-серверу следует обращаться и в каком месте дерева начинать запросы. Корректно задав эти параметры, вы можете приступать к тестированию.

Тестирование LDAP

Проще всего протестировать сервер LDAP, используя клиентские средства, поставляемые в составе OpenLDAP. В командной строке запускаем ldapsearch для подключения клиента LDAP к серверу и выполняем запрос к каталогу. Рассмотрим успешный пример:

```
# ldapsearch -x -LLL -b "uid=bammbamm,ou=it,ou=people,dc=example,dc=com"
  "(objectclass=*)"
dn: uid=bammbamm,ou=it,ou=people,dc=example,dc=com
uid: bammbamm
givenName: Bamm
sn: Bamm
cn: Bamm Bamm
homeDirectory: /var/spool/mail/bammbamm
maildrop: postmaster@example.com
maildrop: bamm.bamm@example.com
maildrop: abuse@example.com
objectClass: CourierMailAlias
objectClass: CourierMailAccount
objectClass: inetOrgPerson
mailbox: /var/spool/mail/bammbamm/Maildir
quota: 5120000S
userPassword:: YmFtbV9zZWNyZXQ=
uidNumber: 1003
gidNumber: 1003
mail: bamm@example.com
```

Если вы получили такие выходные данные, это означает, что вы можете обращаться к каталогу и он находится именно там, где планировалось. Если выходные данные отсутствуют, задайте параметр loglevel (воспользовавшись описанием slapd.conf(5)) так, чтобы увеличить объем выводимой отладочной информации – она может оказаться полезной.

Настройка Postfix и LDAP

Для того чтобы проверить, включена ли уже в вашей версии Postfix поддержка LDAP и SASL, выполните такую команду:

```
$ ldd `/usr/sbin/postconf -h daemon_directory`/smtpd
        linux-gate.so.1 =>  (0x00bad000)
        libldap.so.2 => /usr/lib/libldap.so.2 (0x00882000)
        liblber.so.2 => /usr/lib/liblber.so.2 (0x00646000)
        libsasl2.so.2 => /usr/lib/libsasl2.so.2 (0x0098a000)
        libdb-4.2.so => /lib/tls/libdb-4.2.so (0x00a73000)
        libnsl.so.1 => /lib/libnsl.so.1 (0x00835000)
        libresolv.so.2 => /lib/libresolv.so.2 (0x00655000)
        libc.so.6 => /lib/tls/libc.so.6 (0x004e6000)
```

```
libdl.so.2 => /lib/libdl.so.2 (0x00603000)
libssl.so.4 => /lib/libssl.so.4 (0x0084c000)
libcrypto.so.4 => /lib/libcrypto.so.4 (0x04377000)
libpthread.so.0 => /lib/tls/libpthread.so.0 (0x0061c000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x004cd000)
libgssapi_krb5.so.2 => /usr/lib/libgssapi_krb5.so.2 (0x00d09000)
libkrb5.so.3 => /usr/lib/libkrb5.so.3 (0x006da000)
libcom_err.so.2 => /lib/libcom_err.so.2 (0x058fd000)
libk5crypto.so.3 => /usr/lib/libk5crypto.so.3 (0x05902000)
libz.so.1 => /usr/lib/libz.so.1 (0x00609000)
```

Строки, содержащие имена библиотек `ldap` и `sasl2`, указывают на то, что поддержка LDAP и SASL уже встроена в вашу версию Postfix.

В главе 15 было рассказано, как настраивать поддержку SASL. Теперь же в дополнение к SASL нам нужна еще поддержка LDAP. Вы можете обеспечить ее за счет переменных окружения `CCARGS` и `AUXLIBS`, которые используются в процессе сборки. Если помните, для сборки с поддержкой SASL мы делали такие настройки:

```
$ CCARGS="-DUSE_SASL_AUTH -I/usr/local/include AUXLIBS="-L/usr/local/lib
  -lsasl2" make makefiles
```

Для того чтобы собрать библиотеки Cyrus SASL с поддержкой LDAP, необходимо найти библиотеки и заголовочные файлы LDAP в своей системе. Если вы не знаете, где они находятся, то библиотеки можно поискать следующим образом:

```
# find /usr -name 'libldap*.*'
/usr/local/lib/libldap.so.2
/usr/local/lib/libldap.so.2.0.122
/usr/local/lib/libldap_r.so.2
/usr/local/lib/libldap_r.so.2.0.122
/usr/local/lib/libldap.so
/usr/local/lib/libldap.a
/usr/local/lib/libldap_r.so
/usr/local/lib/libldap_r.a
# find /usr -name 'liblber*.*'
/usr/local/lib/liblber.so.2.0.122
/usr/local/lib/liblber.so.2
/usr/local/lib/liblber.so
/usr/local/lib/liblber.a
```

Выходные данные показывают, что библиотеки LDAP находятся в каталоге `/usr/local/lib`; каталоги `/usr/lib` и `/usr/include` просматриваются компилятором, препроцессором и компоновщиком автоматически. Запомните этот адрес и займитесь поиском соответствующих заголовочных файлов:

```
# find /usr -name '*ldap*.h'
/usr/local/include/ldap.h
/usr/local/include/ldap_cdefs.h
/usr/local/include/ldap_schema.h
```

```
/usr/local/include/ldap_utf8.h
/usr/local/include/ldap_features.h
```

Примечание

Если вам удалось обнаружить в своей системе библиотеки, а заголовочные файлы для LDAP – нет, то, вероятно, нужно будет установить пакеты для разработки LDAP из операционной системы. Вам понадобятся пакеты, названия которых заканчиваются на `-dev` или `-devel`.

Теперь распаковываем исходные тексты Postfix от имени обычного пользователя и переходим в каталог исходных текстов Postfix. Настраиваем и собираем Postfix с поддержкой как SASL (`DUSE_SASL_AUTH`), так и LDAP (`-DHAS_LDAP`):

```
$ CCARGS="-DHAS_LDAP -DUSE_SASL_AUTH -I/usr/local/include" AUXLIBS="-lldap
  -llber -L/usr/local/lib -lsasl2"
$ make makefiles
$ make
```

После завершения сборки переключаемся на суперпользователя (`root`) и запускаем `make install` или `make upgrade` соответственно случаю. Проверяем, действительно ли включена поддержка SASL и LDAP (см. описание в начале главы).

Просмотры LDAP

В списках рассылки Postfix обычно мало вопросов, связанных с LDAP, особенно по сравнению с количеством вопросов, относящихся к базам данных. Многие считают, что использование Postfix (или чего-то другого с теми же целями) в сочетании с LDAP сродни шаманству, и пытаются всеми силами этого избежать. Однако все совсем не так. Для настройки LDAP-запросов следует выполнить следующие действия:

1. Создать каталог для конфигурационных файлов LDAP.
2. Создать файл конфигурации Postfix для LDAP.
3. Проверить LDAP-запрос.
4. Настроить Postfix для использования LDAP-запросов в данной конфигурации.

Все эти действия будут дважды проделаны и описаны в последующих разделах. Сначала мы будем настраивать запрос для локальных получателей, а затем – запрос для почтовых псевдонимов.

Создание каталога конфигурации LDAP

Правильно настроенный LDAP-сервер отвергает запросы данных каталога, связанных с безопасностью. Каталог требует, чтобы пользователи, желающие получить такие данные, сначала аутентифицировались («связались» – «bind») на LDAP-сервере. Вы узнаете, как создать связанного пользователя для Postfix, в разделе «Тонкая настройка» далее

в этой главе, но первый шаг по созданию дополнительного каталога конфигурации необходимо сделать сейчас. Дело в том, что верительные данные связанного пользователя Postfix должны храниться в файлах конфигурации Postfix, а помещать их в файл `main.cf` мы не хотим, чтобы они не стали общедоступными.

Предупреждение

Для создания конфигураций LDAP во внешних файлах требуется версия Postfix 2.x. Вы можете задать все параметры запросов LDAP в файле `main.cf`, но тогда пароли LDAP-пользователей будут находиться в `main.cf`, что небезопасно – любой пользователь системы UNIX, в которой работает Postfix, сможет их прочитать.

Если вы все же захотите использовать для хранения верительных данных файл `main.cf`, обратитесь к разделу «Backwards Compatibility» на странице руководства ldap_table(5).

Для создания каталога конфигурации создаем каталог `/etc/postfix/ldap`, доступный только для пользователей `root` и `postfix`. Будем хранить там все файлы конфигурации с картами LDAP и ссылаться на них из `main.cf`.

```
# mkdir /etc/postfix/ldap
# chgrp postfix /etc/postfix/ldap
# chmod 750 /etc/postfix/ldap
```

Добавление LDAP-запросов для локальных получателей

Давайте теперь займемся основными параметрами Postfix, которые необходимо задать для обращения к серверу LDAP. Начнем с отмены операций LDAP-связывания. Затем создадим набор параметров, хранящих информацию, используемую для проверки локальных получателей.

Отмена LDAP-связывания

По умолчанию, прежде чем осуществлять запрос, Postfix пытается аутентифицироваться на LDAP-сервере посредством LDAP-связывания. Пока мы только приступаем к настройке, лучше отключить эту аутентификацию, чтобы не усложнять дело. Создаем файл конфигурации запросов LDAP с именем `/etc/postfix/ldap/local_recipients.cf` и отключаем связывание следующим параметром:

```
bind = no
```

Настройка хоста LDAP

Вы можете сообщить серверу Postfix о том, где следует искать службу каталога, используя параметры конфигурации `server_host` и `server_port`. Параметр `server_host` определяет тип соединения (`ldap://`, `ldaps://` или `ldapi://`) как часть URL-адреса одного или нескольких LDAP-сер-

веров и может включать порт сервера. По умолчанию этот параметр задан следующим образом: `server_host = ldap://localhost:389`.

Дополнительно вы можете задать параметр `server_port` (по умолчанию установлен порт 389) для указания порта сервера LDAP, но это имеет смысл, только если все ваши LDAP-серверы прослушивают один и тот же порт. В противном случае можно просто добавить номер порта в конец URL, например:

```
server_host = ldap://mail.example.com:389, ldaps://auth.example.com:636
```

В этой главе LDAP-сервер работает на том же хосте, что и другие серверы, и прослушивает порт LDAP по умолчанию (389). В данном случае можно задать параметр `server_host` так:

```
server_host = ldap://mail.example.com
```

Указание ветви

Задаем параметр `search_base`, чтобы указать серверу Postfix, с какой ветви следует начинать поиск. У параметра нет значения по умолчанию, так что его всегда необходимо устанавливать. Добавляем `dn`-часть ветви для пользовательских объектов:

```
search_base = ou=people,dc=example,dc=com
```

Определение атрибутов результатов LDAP

В завершение настройки карты LDAP необходимо определить атрибут для хранения ключа, к которому Postfix обращается при просмотре. Используется точно такая же логика, что и при работе с любыми другими уже известными вам типами карт. Соответствие имен параметров для ключей и значений полям индексированной карты приведено в табл. 19.1.

Таблица 19.1. Соответствие полей индексированной карты параметрам LDAP-запроса

Тип карты	Левая сторона	Правая сторона	Условия
Индексированная карта	Ключ	Значение	–
LDAP-запрос	`query_filter`	`result_attribute`, `result_filter`	`special_result_attribute`

Как видите, атрибут ключа запроса определяется параметром `query_filter`. Согласно примеру из каталога фирмы Example Inc. указываем атрибут для адресов получателей локальной почты (атрибут `mail`) и определяем часть полного почтового адреса, которую сервер Postfix должен предоставить серверу LDAP. Для подстановки используются следующие групповые символы:

%s	Полный почтовый адрес (например, bamm@example.com).
%u	Локальная часть без символа @ и указания домена (например, bamm).
%d	Доменная часть без локальной части и символа @ (например, example.com).

Записи каталога в данной главе содержат только полные почтовые адреса, такие как bamm@example.com, поэтому мы будем использовать %s. Настраиваем Postfix для выполнения запросов на основе полностью определенного доменного имени:

```
query_filter = (mail=%s)
```

Примечание

Стандартный синтаксис запросов и результатов LDAP определен в RFC 2254 (*http://www.rfc-editor.org/rfc/rfc2254.txt*). Вы можете задавать свой запрос так, как вам нравится. Например, если в вашей схеме есть атрибут mailboxActive, который указывает на активный (неотключенный) почтовый ящик, вы можете задать параметры запроса следующим образом:

```
query_filter = (&(mail=%s)(mailboxActive=1))
```

Затем определяем атрибут, который Postfix будет использовать для запроса результата. В нашем распоряжении два параметра: result_attribute для настройки собственно атрибута и result_filter для отфильтровывания ненужных вам частей результата LDAP-запроса.

В этой главе нужно будет лишь проверить наличие локального почтового адреса, т. к. агентом доставки здесь является Courier maildrop, а не Postfix. Postfix необходимо только знать, действителен ли адрес локального получателя, так что вы можете использовать любой возвращаемый атрибут.

Примечание

Это означает, что Postfix принимает любое значение, которое возвращает сервер LDAP, в качестве доказательства существования локального получателя для входящего сообщения. Если же сервер LDAP не возвращает значение, то Postfix отвергает сообщение.

Выберите простой атрибут, который легко идентифицировать при тестировании запроса. В нашем случае отлично подходит uid, так что именно его мы и укажем как атрибут результата в файле local_recipients.cf конфигурации Postfix:

```
result_attribute = uid
```

Активация карты запросов

Файл /etc/postfix/ldap/local_recipients.cf готов, теперь необходимо активировать эту карту в основной конфигурации Postfix. Указываем

в файле main.cf в параметре local_recipient_maps список карт, которые Postfix будет просматривать при поиске локальных получателей. Для повышения производительности запросов LDAP используйте демон proxymap (см. описание в главе 5):

```
local_recipient_maps = proxy:ldap:/etc/postfix/ldap/local_recipients.cf
```

Каждый процесс proxymap делает запросы от имени нескольких клиентов и может (но не обязан) кэшировать результаты поиска.

Новая карта готова к использованию, перезагружаем конфигурацию Postfix и переходим к тестированию.

Тестирование получателей LDAP

Пока единственное, что мы можем проверить, – это может ли Postfix найти действительного локального получателя. Отправить тестовое сообщение невозможно, т. к. настройкой локальной доставки мы еще не занимались.

Используем команду postmap для запроса к серверу LDAP об известном нам локальном получателе bamm@example.com. Однако, прежде чем это делать, следует переключиться на пользователя postfix для уверенности в том, что пользователь имеет права на чтение файла конфигурации LDAP и выполнение запроса. Успешный тест должен выглядеть так:

```
# su - postfix
$ /usr/sbin/postmap -q "bamm@example.com" ldap:/etc/postfix/ldap/
  local_recipients.cf
bammbamm
```

Запрос возвращает значение uid для bamm@example.com; в данном случае значением данного атрибута является bammbamm, так что данная конфигурация работает. Если же ничего не получается, добавьте в команду postmap параметр -v для вывода более подробной информации. Кроме того, вы можете добавить в файл конфигурации запросов LDAP параметр debuglevel:

```
debuglevel = 1
```

Примечание

Уровень подробности отладочной информации можно увеличить до 3, тогда вы получите все необходимые данные для решения проблемы.

Обращение к LDAP за почтовыми псевдонимами

Настройка Postfix для обращения к серверу LDAP за почтовыми псевдонимами состоит из тех же операций, которые были приведены в разделе «Добавление LDAP-запросов для локальных получателей», нужно будет лишь указать другой параметр result_attribute для результата запроса и использовать параметр query_filter для извлечения из результатов определенного атрибута.

Имена псевдонимов на сервере каталогов компании Example Inc. присваиваются атрибуту maildrop. Файл конфигурации для псевдонимов (например, /etc/postfix/ldap/virtual_aliases.cf) будет иметь такой вид:

```
bind = no
server_host = ldap://mail.example.com
search_base = ou=people,dc=example,dc=com
query_filter = (maildrop=%s)
result_attribute = mail
```

Настройка Postfix для использования карт запросов псевдонимов LDAP

Когда файл конфигурации запросов псевдонимов LDAP готов, следует подключить его к вашей конфигурации Postfix, задав параметр virtual_alias_maps в файле main.cf. Используется тот же синтаксис, что и для карт получателей, описанный ранее в разделе «Активация карты запросов»:

```
virtual_alias_maps = proxy:ldap:/etc/postfix/ldap/virtual_aliases.cf
```

Перезагружаем конфигурацию и приступаем к тестированию.

Тестирование карт запросов псевдонимов LDAP

Как и раньше, мы пока не можем отправить сообщение для тестирования конфигурации запросов LDAP, так что используем команду postmap. Вспоминаем о том, что postmaster@example.com является псевдонимом для bamm@example.com в каталоге. Переключаемся на пользователя postfix и смотрим, работает ли карта псевдонимов:

```
# su - postfix
$ /usr/sbin/postmap -q "postmaster@example.com" ldap:/etc/postfix/ldap/
virtual_aliases.cf
bamm@example.com
```

Если вы не получите никаких выходных данных, добавьте в команду postmap параметр -v для вывода более подробной информации. Кроме того, вы можете добавить в файл конфигурации запросов LDAP параметр debuglevel (и повысить уровень подробности вывода отладочной информации до 3, в зависимости от необходимого вам объема данных):

```
debuglevel = 1
```

Проверка списков

Может быть, вы помните, что структура простых списков, описанная ранее в разделе «Создание объектов списков», использует псевдонимы. Поэтому, выполнив запрос postmap с именем списка, мы должны быть готовы извлечь нескольких получателей для псевдонима:

```
# su - postfix
$ /usr/sbin/postmap -q "all@example.com" ldap:/etc/postfix/ldap/
virtual_aliases.cf
bamm@example.com,pebble@example.com,mcbricker@example.com,
```

```
flintstone@example.com, rubble@example.com
```

Когда сервер LDAP возвращает несколько результатов, Postfix собирает их вместе и преобразует в список, разделенный запятыми (как в предыдущем примере).

Передача функции доставки агенту Courier maildrop

Рассматриваемая в этой главе конфигурация не использует ни один из агентов доставки Postfix (демоны `local`, `maildrop` и `virtual`). Причина в том, что сторонние агенты доставки поддерживают правила фильтрации и квоты. Например, пользователь `bammbamm` может помещать все сообщения для `postmaster@example.com` в отдельную подпапку. В этом разделе будет показано, как передать функцию локальной доставки сообщений агенту Courier maildrop.

Создание локального транспорта

Начинаем с определения новой транспортной службы в файле `master.cf`. Не бойтесь испортить свой текущий локальный агент доставки (если у вас уже работает какой-либо), ведь сервер Postfix начнет использовать новую службу лишь после внесения соответствующих изменений в файл `main.cf`. Новый транспорт будет `pipe`-транспортом и будет называться `maildrop`. Добавляем в файл `master.cf` следующие строки:

```
maildrop  unix  -       n       n       -       -       pipe
  flags=Rhu user=vmail argv=/usr/local/bin/maildrop -d ${recipient} -w 75
```

Флаги данного `pipe(8)`-транспорта работают следующим образом:

`D`	Присоединяет спереди заголовок получателя сообщения Delivered-To с адресом получателя конверта. Таким образом имитируется локальный демон Postfix и предотвращается зацикливание почты.
`R`	Присоединяет спереди заголовок сообщения `Return-Path` с адресом отправителя конверта. RFC 2821 требует этого от локального демона Postfix.
`h`	Преобразует к нижнему регистру в командной строке доменную часть адреса `$recipient` и имя хоста `$nexthop`.
`u`	Преобразует к нижнему регистру в командной строке локальную часть адреса `$recipient`.
`user`	`user=vmail` означает, что `/usr/local/bin/maildrop -d ${recipient}` должен запускаться от имени пользователя `vmail` (вы узнаете об этом пользователе далее при настройке Courier maildrop).
`-w`	Задает уровень предупреждений для deliverquota(8) равным 75% от квоты почтового каталога. Пропустите этот флаг, если не хотите вводить квоты.

После добавления службы отредактируйте файл `main.cf`, чтобы сообщить Postfix о том, что данная служба должна использоваться в качестве локального транспорта:

```
local_transport = maildrop
```

Предупреждение

Существуют два побочных эффекта использования maildrop вместо локального агента доставки. Во-первых, локальный агент доставки просматривает карты псевдонимов, а Courier maildrop этого делать не умеет. Однако мы уже справились с этой проблемой в разделе «Настройка Postfix для использования карт запросов псевдонимов LDAP» путем установки параметра `virtual_alias_maps`. Второе ограничение заключается в том, что maildrop не следит за петлями Delivered-To, если не задать правило фильтрации. Этим мы займемся в разделе «Создание почтового фильтра».

Ограничение одновременно доставляемых сообщений

Перед проверкой нового транспорта необходимо убедиться в том, что он настроен для доставки сообщений только одному пользователю одновременно. Не беспокойтесь о возможных потерях производительности, т. к. это нормально для большинства почтовых серверов. Даже локальный транспорт Postfix имеет такое ограничение (параметр `local_destination_recipient_limit`).

Синтаксис параметра создания ограничения для других LDA имеет вид *имя_службы*`_destination_recipient_limit`, где *имя_службы* – это первое поле файла `master.cf`. Добавьте в файл `main.cf` такую строку для только что определенной службы Courier maildrop:

```
maildrop_destination_recipient_limit = 1
```

Перезагружаем конфигурацию Postfix и начинаем тестирование.

Тестирование LDA

Для тестирования LDA просто отправьте сообщение по одному из адресов вашей карты получателей и посмотрите в почтовый журнал. Вы должны увидеть, что Postfix использует новый транспорт maildrop:

```
# echo foo | /usr/sbin/sendmail -f "" postmaster@example.com
# tail -f /var/log/maillog
Jun 29 14:39:13 mail postfix/pickup[5122]: AC7B94400C: uid=0 from=<>
Jun 29 14:39:13 mail postfix/cleanup[5127]: AC7B94400C:
    message-id=<20040629123913.AC7B94400C@mail.example.com>
Jun 29 14:39:13 mail postfix/qmgr[5123]: AC7B94400C:
    from=<>, size=285, nrcpt=1 (queue active)
Jun 29 14:39:13 mail postfix/pipe[5130]: AC7B94400C:
    to=<bamm@example.com>, orig_to=<postmaster@example.com>,
    relay=maildrop, delay=0, status=sent (example.com)
Jun 29 14:39:13 mail postfix/qmgr[5123]: AC7B94400C: removed
```

Если в журнале отсутствует maildrop, то включите детальное журналирование для `smtpd` в файле `master.cf`, перезагрузите конфигурацию и попробуйте отправить еще одно сообщение.

Настройка Courier maildrop

Courier maildrop – это локальный агент доставки, который забирает сообщения от агента передачи сообщений, такого как Postfix, и хранит их в почтовом ящике получателя в формате Maildir. Maildrop также может применять к сообщениям фильтры. Еще одной интересной возможностью является введение квот для каталогов (сообщения в формате Maildir хранятся в отдельных файлах в каталоге).

Подготовка системы

Courier maildrop запрещает неавторизованным пользователям осуществлять запись в почтовые ящики. Прежде чем собирать исполняемые файлы, вы должны выбрать доверенных пользователей и группы. Создайте хотя бы одного нового пользователя с номерами пользователя и группы, совпадающими со значениями, которые вы присвоили атрибутам `uidNumber` и `gidNumber` в вашем каталоге LDAP. При обращении к почтовому ящику Courier maildrop и IMAP извлекают эти атрибуты из каталога.

Конфигурация, рассматриваемая в данной главе, использует следующие идентификаторы:

```
uidNumber: 1003
gidNumber: 1003
```

Если вы этого еще не сделали, создайте пользователя и группу с соответствующими значениями. Например, следующие команды добавляют пользователя и группу `vmail` в системе Linux:

```
# useradd -u 1003 vmail
# groupadd -g 1003 vmail
```

Установка Courier maildrop

В момент написания этой книги поддержка LDAP в Courier maildrop находилась на этапе бета-тестирования. При корректной настройке все работает хорошо, но при возникновении ошибки вы не получите сведений о том, что произошло. Учитывая темпы развития Courier maildrop, скорее всего, эта недоработка будет устранена, когда вы будете читать нашу книгу. Для получения поддержки LDAP скачайте текущую версию maildrop с сайта *http://www.courier-mta.org/download.php#maildrop*.

Распакуйте архив от имени обычного пользователя и перейдите в созданный каталог. Courier maildrop использует GNU Autoconf, поэтому параметры сборки задаются следующим образом:

```
$ ./configure --enable-restrict-trusted=1 --enable-trusted-users='root vmail'
  --enable-trusted-groups='root vmail' --enable-maildirquota --with-trashquota
  -enable-maildropldap
```

Убедитесь в том, что вы указали параметр `--enable-maildropldap` для поддержки LDAP. Если вам необходима поддержка квот, используйте параметры `--enable-maildirquota` и `--with-trashquota`, но сначала прочтите раздел «Подготовка квот Maildir». После завершения работы сценария конфигурации запустите `make`. Если все идет хорошо, переключитесь на пользователя `root` и выполните `make install-strip install-man` для установки бинарных файлов без отладочной информации и страниц руководства.

Теперь, когда `maildrop` установлен, необходимо как-то разобраться с Postfix-демоном `pipe`, который отказывается запускать какие бы то ни было процессы от имени `root`. Для того чтобы решить проблему, сделайте владельцем файла `maildrop` пользователя `root`.

```
# chmod 750  /usr/local/bin/maildrop
# chmod u+s /usr/local/bin/maildrop
# chown root:vmail /usr/local/bin/maildrop
# ls -l /usr/local/bin/maildrop
-rwsr-x---   1 root   vmail   165552 Jun 25 12:48 /usr/local/bin/maildrop
```

Не беспокойтесь о вмешательстве в обычные политики безопасности Postfix. Демон `pipe` запускает `maildrop`, который изначально запускается от имени `root`, но, получив корректные идентификаторы пользователя и группы от LDAP, переключается на данного пользователя и группу.

Настройка Courier maildrop

Проще всего создать конфигурацию `maildrop` для LDAP, скопировав файл `maildropldap.config` из каталога исходных текстов `maildrop` в `/etc/maildropldap.config` (именно там Courier maildrop по умолчанию ищет файл конфигурации LDAP). Отредактируем этот файл, чтобы он соответствовал нашей конфигурации.

Для рассматриваемого нами в этой главе примера файл конфигурации должен быть таким:

```
hostname              mail.example.com
basedn                dc=example,dc=com
filter                &(objectclass=inetorgperson)
timeout               5
search_method         mail
mail_attr             mail
uid_attr              uid
uidnumber_attr        uidNumber
gidnumber_attr        gidNumber
maildir_attr          mailbox
homedirectory_attr    homeDirectory
quota_attr            quota
```

Создание почтовых ящиков Maildir

Все пользователи данного корпоративного почтового сервера являются виртуальными пользователями. Они не связаны с локальными пользовательскими учетными записями, и когда вы создаете пользователей в своем каталоге LDAP, никакой сценарий автоматически не создает домашний каталог или почтовый ящик.

Прежде чем тестировать программу `maildrop`, необходимо создать пользовательские почтовые ящики. Пока что делаем это вручную (позже вы, конечно, сможете автоматизировать эту процедуру при помощи сценариев). В состав Courier maildrop входит утилита `maildirmake`, которая создает почтовый ящик Maildir и несколько подпапок по умолчанию.

Для того чтобы узнать, где находится почтовый ящик, maildrop анализирует атрибут `homeDirectories` на сервере LDAP. Однако, т. к. мы работаем с виртуальными пользователями, можно создать технологический каталог `/home/mailskel` и просто скопировать его в пользовательские домашние каталоги, не заботясь о назначении прав доступа.

Вот как можно создать такой дополнительный почтовый каталог:

```
# mkdir /home/mailskel
# chgrp vmail /home/mailskel
# chmod 770 /home/mailskel
# ls -dall /home/mailskel
drwxrwx---    6 root     vmail        4096 Jun 28 17:52 /home/mailskel
```

Теперь можно создать другой каталог, `/home/mailskel/.templateDir`, и сделать пользователя `vmail` его владельцем:

```
# mkdir /home/mailskel/.templateDir
# chown vmail /home/mailskel/.templateDir/
# chgrp vmail /home/mailskel/.templateDir/
# chmod 700 /home/mailskel/.templateDir/
# ls -dall /home/mailskel/.templateDir
drwx------    2 vmail    vmail        4096 Jun 29 22:27 /home/mailskel
                                                       /.templateDir
```

Мы готовы к созданию настоящего почтового каталога. Однако прежде чем запускать `maildirmake` для создания базовой структуры каталога Maildir, необходимо переключиться на пользователя `vmail`, чтобы `maildrop` позже имел возможность обращаться к его точной копии:

```
# su - vmail
$ maildirmake /home/mailskel/.templateDir/Maildir
```

Проверяем, создала ли утилита `maildirmake` каталог:

```
$ ls -la /home/mailskel/.templateDir/Maildir
total 20
drwx------    5 vmail    vmail        4096 Jun 29 22:31 .
drwx------    3 vmail    vmail        4096 Jun 29 22:31 ..
drwx------    2 vmail    vmail        4096 Jun 29 22:31 cur
```

```
drwx------    2 vmail    vmail        4096 Jun 29 22:31 new
drwx------    2 vmail    vmail        4096 Jun 29 22:31 tmp
```

Как видите, у вас есть каталог `Maildir`, включающий в себя три подкаталога: `cur`, `new` и `tmp`, в которых хранятся входящие сообщения (в зависимости от их состояния).

Запустим `maildirmake` еще несколько раз для создания подпапок `Drafts`, `Trash` и `Spam`:

```
$ maildirmake -f Drafts /home/mailskel/.templateDir/Maildir
$ maildirmake -f Trash /home/mailskel/.templateDir/Maildir
$ maildirmake -f Spam /home/mailskel/.templateDir/Maildir
$ ls -la /home/mailskel/.templateDir/Maildir
total 32
drwx------    8 vmail    vmail        4096 Jun 29 22:39 .
drwx------    3 vmail    vmail        4096 Jun 29 22:31 ..
drwx------    2 vmail    vmail        4096 Jun 29 22:31 cur
drwx------    5 vmail    vmail        4096 Jun 29 22:39 .Drafts
drwx------    2 vmail    vmail        4096 Jun 29 22:31 new
drwx------    5 vmail    vmail        4096 Jun 29 22:39 .Spam
drwx------    2 vmail    vmail        4096 Jun 29 22:31 tmp
drwx------    5 vmail    vmail        4096 Jun 29 22:39 .Trash
```

Обратите внимание на то, что внутри каждого из этих подкаталогов есть подкаталоги `cur`, `new` и `tmp`.

Шаблон почтового каталога теперь можно использовать как основу для создания виртуальных почтовых ящиков Maildir для других пользователей. Выполните `cp -pR` от имени суперпользователя для сохранения права собственности и привилегий при копировании шаблона. Например, вы можете создать почтовый ящик для пользователя `bammbamm` следующим образом:

```
# cp -pR /home/mailskel/.templateDir/ /var/spool/mail/bammbamm
# ls -all /var/spool/mail/bammbamm
total 12
drwx------    3 vmail    vmail        4096 Jun 29 22:31 .
drwxrwx---    9 root     vmail        4096 Jun 29 22:55 ..
drwx------    8 vmail    vmail        4096 Jun 29 22:39 Maildir
# ls -all /var/spool/mail/bammbamm/Maildir/
total 32
drwx------    8 vmail    vmail        4096 Jun 29 22:39 .
drwx------    3 vmail    vmail        4096 Jun 29 22:31 ..
drwx------    2 vmail    vmail        4096 Jun 29 22:31 cur
drwx------    5 vmail    vmail        4096 Jun 29 22:39 .Drafts
drwx------    2 vmail    vmail        4096 Jun 29 22:31 new
drwx------    5 vmail    vmail        4096 Jun 29 22:39 .Spam
drwx------    2 vmail    vmail        4096 Jun 29 22:31 tmp
drwx------    5 vmail    vmail        4096 Jun 29 22:39 .Trash
```

Пока не создавайте каталоги для остальных пользователей, прочитайте сначала следующий раздел.

Создание почтового фильтра

После создания шаблона каталога Maildir вы можете создать набор фильтров доставки по умолчанию для Courier maildrop. Для начала создадим глобальный набор правил, которые будут применяться к каждому получателю почты на вашем сервере, в каталоге /etc/maildropr. Для отладки будет удобно поместить в домашний каталог каждого пользователя файл журнала таким правилом:

```
logfile "$HOME/maildrop.log"
```

Теперь подумаем о правилах фильтрации. Первое правило будет предотвращать появление петель Delivered-To, о которых мы упоминали в разделе «Создание локального транспорта». Оно всегда должно находиться в начале файла maildroprc, чтобы исполняться прежде, чем будут предприняты какие-то другие действия. Второе правило говорит агенту maildrop, что он должен помещать любое сообщение со статусом X-Spam-Status: Yes в заголовке в подпапку .spam почтового ящика получателей ($DEFAULT):

```
logfile "$HOME/maildrop.log"
if ( /^Delivered-To: $LOGNAME@mail.example.com/:h )
{
  echo "This message is looping, it already has my Delivered-To: Header!"
  EXITCODE = 1
  exit
}
# Add Delivered-To: header
xfilter "reformail -A'Delivered-To: $LOGNAME@mail.example.com'"
if (/^X-Spam-Status: Yes/)
{
   to $DEFAULT/.spam/
}
```

Вы можете выполнять фильтрацию для каждого пользователя в отдельности, добавив файл .mailfilter с дополнительными правилами в домашний каталог получателя. Необходимо использовать точный набор привилегий, в противном случае maildrop откажется следовать инструкциям из .mailfilter. Покажем, как создать такой файл для пользователя bammbamm:

```
# su - vmail
$ cd /var/spool/mail/bammbamm
$ touch .mailfilter
$ chmod 600 .mailfilter
```

Вспомним, что предыдущий пользователь также является postmaster. Поэтому следующее правило фильтрации .mailfilter помещает сообщения, в заголовке которых в качестве получателя указан postmaster@example.com, в каталог .postmaster:

```
if (/^To.*postmaster@example\.com/)
{
```

```
    to "$DEFAULT/.postmaster/"
}
```

Примечание

Глобальные правила фильтрации имеют более высокий приоритет, чем правила для отдельных пользователей.

Вы еще не создали папку `.postmaster`, поэтому необходимо создать ее, чтобы пользоваться данным правилом. Переключаемся на пользователя `vmail` и запускаем `maildirmake`:

```
# su - vmail
$ cd /var/spool/mail/bammbamm
$ maildirmake -f postmaster Maildir
```

Правила фильтрации maildrop могут значительно упростить жизнь пользователей, т. к. им не придется надеяться на свой почтовый клиент. Дополнительную информацию о возможностях фильтрации maildrop вы найдете на странице руководства maildropfilter(5). Обратите особое внимание на примеры из раздела «Others» (Прочее) для сайта *http://www.dotfiles.com*.

Подготовка квот Maildir

Кажется, что квоты – это замечательная функциональность, и здорово было бы ее использовать, но, прежде чем вы решите сделать это, имейте в виду, что система квот `maildrop` является предметом споров, поскольку квоты Maildir не всегда надежны. Точка зрения разработчика Postfix представлена в высказывании Виктора Духовны (Victor Duchovni):

> Камнем преткновения является то, что пользователи «maildir++» против надежного кода квот, который всегда гарантированно работает! Они хотят найти компромисс между прочностью и надежностью, с одной стороны (файлы состояния никогда не нужно пересоздавать, пользователь всегда находится в рамках квот, ...), и простотой использования – с другой (не нужно думать об интерфейсе квот файловой системы, квоты могут быть «мягкими», т. е. при достижении заданных пределов доступна некоторая ограниченная конфигурируемая функциональность).

Естественно, этот аспект квот общеизвестен (см. *http://www.inter7.com/courierimap/README.maildirquota.html*), но многие администраторы все же предпочитают воспользоваться преимуществами Courier maildrop, испытывая при этом небольшие неудобства. Все случаи индивидуальны, и только вы сами можете решить, использовать ли вам квоты.

Если вы решите использовать квоты, необходимо добавить `-w 75` в строку конфигурации для службы (см. раздел «Создание локального транспорта»), чтобы при заполнении почтового каталога на 75% («мягкое»

ограничение) формировалось предупреждение. Шаблон этого предупреждения вы должны будете придумать сами. Создаем шаблон в виде файла с простым текстом под именем /usr/local/etc/quotawarnmsg, содержащего такое сообщение:

```
From: MAILER-DAEMON <>
To: Valued Customer:;
Subject: Mail quota warning
Mime-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 7bit

Your mailbox on the server is now more than 75% full.
So that you can continue to receive mail you need to remove
some messages from your mailbox.
```

При сохранении предупреждения о квотах в почтовом ящике получателя maildrop добавляет заголовки Message-ID и Date.

Тестирование Courier maildrop

Пришло время протестировать нашу версию Courier maildrop. Выполним следующие операции:

1. Проверим, работает ли Courier maildrop без Postfix.
2. Используем программу sendmail, поставляемую с Postfix, для доставки сообщения агенту Courier maildrop.
3. Проверим работу фильтров.
4. Проверим работу квот (если вы их используете).

Тестирование автономного Courier maildrop

Переключаемся на пользователя vmail и выполняем команду maildrop так, как это сделал бы сервер Postfix:

```
# su - vmail
$ /usr/local/bin/maildrop -d bamm@example.com
this is a test message
```

Для отправки сообщения нажмите в строке Ctrl-D. Проверьте правильность работы maildrop, анализируя код завершения:

```
$ echo $?
0
```

Нулевой код завершения означает успешное исполнение и доставку. Конечно, следует еще проверить, создал ли maildrop новый файл в почтовом каталоге пользователя bamm:

```
$ ls -l /var/spool/mail/bammbamm/Maildir/new
total 4
-rw-------    1 vmail    vmail          23 Jun 30 12:12
  1088590342.M975018P6589V000000000000302I0001840E_0.mail.example.com,S=4
```

Если у вас включено журналирование Courier maildrop, то в журнале доставка будет отражена следующим образом:

```
Date: Mon Aug  9 09:05:56 2004
From:
Subj:
File: /var/spool/mail/bammbamm/Maildir                              (12)
```

Тестирование совместной работы Courier maildrop и Postfix

Для проверки совместной работы Postfix и maildrop используем исполняемый файл Postfix `sendmail` и смотрим в почтовый журнал:

```
# echo foo | /usr/sbin/sendmail -f rubble@example.com bamm@example.com
# tail -f /var/log/maillog
Jul 26 23:20:58 mail postfix/pickup[27883]: 608DD229EF5: uid=0
  from=<rubble@example.com>
Jul 26 23:20:58 mail postfix/cleanup[28429]: 608DD229EF5:
  message-id=<20040726212058.608DD229EF5@mail.example.com>
Jul 26 23:20:58 mail postfix/qmgr[27882]: 608DD229EF5:
  from=<rubble@example.com>, size=288, nrcpt=1 (queue active)
Jul 26 23:20:58 mail postfix/pipe[28432]: 608DD229EF5:
  to=<bamm@example.com>, relay=maildrop, delay=0, status=sent (example.com)
Jul 26 23:20:58 mail postfix/qmgr[27882]: 608DD229EF5: removed
```

В журнале видно, что Postfix переслал сообщение агенту `maildrop`. Если вы включили журналирование в `/etc/maildroprc`, то в журнальном файле `maildrop.log` пользователя Bamm Bamm вы должны найти такую запись:

```
Date: Mon Jul 26 23:24:42 2004
From: rubble@example.com (root)
Subj:
File: /var/spool/mail/bammbamm/Maildir                (345)
```

В записи говорится о том, что сообщение от `rubble@example.com` (идентифицированного сервером Postfix как `root`) было доставлено в `/var/spool/mail/bammbamm/Maildir`, т. е. в почтовый ящик для входящей почты пользователя Bamm Bamm.

Тестирование фильтров Courier maildrop

Для проверки фильтрации создаем и отправляем файл `testmessage`, который содержит что-то, что могло бы заставить заработать правила фильтрации из `/etc/maildroprc`. Вот сообщение, к которому должно быть применено правило фильтрации спама, созданное нами в разделе «Создание почтового фильтра»:

```
From: Barney <rubble@example.com>
To: Bamm Bamm <bamm@example.com>
Subject: Test message tagged as SPAM
X-Spam-Status: Yes
foo bar
```

Отправляем сообщение при помощи sendmail:

```
# /usr/sbin/sendmail -f rubble@example.com bamm@example.com < testmessage
```

В дополнение к проверке подпапки спама проверяем файл maildrop.log, который должен выглядеть примерно так:

```
Date: Mon Jul 26 22:29:24 2004
From: Barney <rubble@example.com>
Subj: Test message tagged as SPAM
File: /var/spool/mail/bammbamm/Maildir/.spam/                    (412)
```

Сообщение доставлено в подпапку .spam, так что глобальный фильтр работает.

Теперь изменим тестовое сообщение, чтобы посмотреть, работают ли локальные фильтры:

```
From: Barney <rubble@example.com>
To: Postmaster <postmaster@example.com>
Subject: Test message for Postmaster
```

Отправим это сообщение по адресу postmaster@example.com:

```
# /usr/sbin/sendmail -f rubble@example.com postmaster@example.com <
testmessage
```

Запись в файле журнала maildrop.log, подтверждающая попадание сообщения в папку для Postmaster, должна выглядеть так:

```
Date: Mon Jul 26 23:36:48 2004
From: Barney <rubble@example.com>
Subj: Test message for Postmaster
File: /var/spool/mail/bammbamm/Maildir/.postmaster               (391)
```

Проверка квот Courier maildrop

Наконец, если вы настроили maildrop для использования квот Maildir, необходимо проверить, работают ли мягкое и жесткое ограничения. Создаем тестовое сообщение размером в 5 Мбайт:

```
#  dd if=/dev/zero of=/root/testmessage bs=5M count=1
1+0 records in
1+0 records out
# ls -all testmessage
-rw-r--r--  1 root root 5242880 Jul 27 09:25 testmessage
```

Используем утилиту ldapmodify, чтобы уменьшить квоту для пользователя Bamm Bamm. Сделаем так, чтобы после первого сообщения заполнение ящика достигло мягкого предела, а после второго тестового сообщения – жесткого. Создаем файл modify_bammbamm_quota.ldif с нашими изменениями:

```
dn: uid=bammbamm,ou=it,ou=people,dc=example,dc=com
changetype: modify
replace: quota
quota: 6990507S
```

Теперь запускаем ldapmodify и импортируем изменения из файла modify_bammbamm_quota.ldif:

```
# ldapmodify -x -D "cn=Manager,dc=example,dc=com" -w secret
-f modify_bammbamm_quota.ldif
```

Отправляем первое из двух тестовых сообщений по адресу bamm@example.com:

```
# /usr/sbin/sendmail -f rubble@example.com bamm@example.com <
/root/testmessage
```

Проверяем, доставлено ли оно в почтовый ящик пользователя Bamm Bamm:

```
# ls -la
-rw-------  1 vmail vmail 5243221 Jul 27 09:38 1090913892.\
    M24019P29629V0000000000000302I00229EF6_0.mail.example.com,S=5243221
-rw-r-----  1 vmail vmail     447 Jul 27 09:38 1090913893.\
    M932062P29629V000000000000000302I00229F00_warn.mail.example.com,S=447
```

Видим дополнительное сообщение, в имени файла которого присутствует строка warn. Используем cat, чтобы просмотреть его:

```
# cat 1090913893.M932062P29629V000000000000000302I00229F00_warn.mail.examp-
le.com\,S\=447
From: MAILER-DAEMON <>
To: Valued Customer:;
Subject: Mail quota warning
Mime-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 7bit

Your mailbox on the server is now more than 75% full. So that you can
continue to receive mail you need to remove some messages from your mailbox.
```

maildrop доставил сообщение и создал предупреждение о достижении квоты в почтовом ящике получателя. Почтовый ящик заполнен на 75%, т. е. прием следующего сообщения привел бы к превышению жесткого предела, поэтому maildrop вынужден вернуть сообщение обратно на rubble@example.com:

```
# /usr/sbin/sendmail -f rubble@example.com bamm@example.com <
/root/testmessage
```

В журнале вы должны увидеть, что сообщение было возвращено отправителю:

```
# tail -f /var/log/maillog
Jul 27 09:59:18 mail postfix/pickup[29788]: 4C083229F09: uid=0
  from=<rubble@example.com>
Jul 27 09:59:18 mail postfix/cleanup[29793]: 4C083229F09:
  message-id=<20040727075918.4C083229F09@mail.example.com>
Jul 27 09:59:18 mail postfix/qmgr[29789]: 4C083229F09:
  from=<rubble@example.com>, size=5250843, nrcpt=1 (queue active)
```

```
Jul 27 09:59:19 mail postfix/pipe[29795]: 4C083229F09:
  to=<bamm@example.com>, relay=maildrop, delay=1, status=bounced (permission
  denied. Command output:   maildrop: maildir over quota. )
```

При возврате отправитель `rubble@example.com` будет оповещен о том, что доставка не удалась по следующей причине:

```
<bamm@example.com>: permission denied. Command output: maildrop:
maildir over quota.
```

Настройка Courier IMAP

Последний сервер, который необходимо настроить, – это Courier IMAP. Для тех, кто еще не знаком с ним, скажем, что Courier поддерживает формат Maildir и предоставляет клиентам службы POP, POP-SSL, IMAP и IMAP-SSL.

Установка Courier IMAP

Для установки сервера Courier IMAP скачайте исходные тексты с сайта *http://www.courier-mta.org/download.php#imap*. Распакуйте архив от имени обычного пользователя, перейдите в только что созданный каталог и выполните команду `configure`:

```
$ ./configure --enable-workarounds-for-imap-client-bugs --enable-unicode
  --without-authpgsql --without-socks
$ make
```

В процессе настройки автоматически выполняется поиск библиотек LDAP на вашем компьютере.

Примечание

Если вы используете Red Hat, добавьте в параметры конфигурации `--with-redhat`, чтобы активировать поддержку этой ОС.

После завершения настройки переключитесь на пользователя `root` и выполните `make install install-configure` для установки программного обеспечения и документации.

Настройка Courier IMAP для использования его демона аутентификации LDAP

Courier использует модульное хранилище данных аутентификации (если Courier IMAP был собран с параметрами по умолчанию, то модули находятся в каталоге `/usr/lib/courier-imap/libexec/authlib`). Чтобы настроить демон аутентификации Courier (`authdaemon`) для эксклюзивной аутентификации LDAP, измените значение параметра `authmodulelist` в файле `/usr/lib/courier-imap/etc/authdaemonrc` на следующее:

```
authmodulelist="authldap"
```

Для согласованности вам следует удалить все остальные имена модулей из `authmodulelist`.

Во время редактирования файла `authdaemonrc` перейдите в конец файла и измените или добавьте параметр `version`, так чтобы он включал в себя только `authdaemond.ldap` (в противном случае `authdaemon` выбирает встреченный первым модуль `authdaemond.*`):

```
version="authdaemond.ldap"
```

Настройка хранилища аутентификационных данных

Вы должны сообщить хранилищу аутентификационных данных о своем сервере LDAP и каталоге. Для настройки модуля `authdaemond.ldap` измените записи по умолчанию в файле `/usr/lib/courier-imap/etc/authldaprc` так, чтобы они соответствовали вашему серверу и каталогу. Для работы примера, рассматриваемого в данной главе, указываем такие параметры:

```
LDAP_SERVER             mail.example.com
LDAP_BASEDN             dc=example,dc=com
LDAP_MAIL               mail
LDAP_FILTER             (objectClass=inetorgperson)
LDAP_HOMEDIR            homeDirectory
LDAP_MAILDIR            mailbox
LDAP_MAILDIRQUOTA       quota
LDAP_CLEARPW            userPassword
LDAP_UID                uidNumber
LDAP_GID                gidNumber
```

Примечание

Комментарии в файле `authldaprc` весьма полезны для пояснения значения параметров.

Создание IMAP-сертификата

Последнее, что осталось сделать, – создать сертификат безопасности для Courier IMAP. Courier создает такой сертификат автоматически при запуске `imapd-ssl` (и он поставляется вместе с утилитой `mkimapdcert`), но мы не можем его использовать, т. к. данный сертификат не подписан нашем центром сертификации.

Для создания сертификата следует выполнить следующие операции:

1. Считаем, что вы создали свой центр сертификации (как было описано в главе 17), тогда вы можете создать сертификат `imapd` следующим образом :

```
# openssl req -new -nodes -keyout imapd_private_key.pem -out
  imapd_private_key.pem -days 365
```

2. Подписываем ключ в своем центре сертификации, создавая открытый сертификат:

```
# openssl ca -policy policy_anything -out imapd_public_cert.pem -infiles
  imapd_private_key.pem
```

3. Создание файла сертификата для Courier IMAP несколько отличается от такой же операции для Postfix TLS и OpenLDAP. Соединяем два созданных файла для создания файла `imapd.pem`:

```
# cat imapd_private_key.pem imapd_public_cert.pem > imapd.pem
```

4. Копируем сертификат в тот каталог, где его может обнаружить Courier, и соответствующим образом указываем привилегии, чтобы защитить секретный ключ внутри файла:

```
# cp imapd.pem /usr/lib/courier-imap/share/imapd.pem
# chmod 600 /usr/lib/courier-imap/share/imapd.pem
# chown root /usr/lib/courier-imap/share/imapd.pem
```

5. Запускаем SSL-экземпляр `imapd`:

```
# /usr/lib/courier-imap/libexec/imapd-ssl.rc start
```

6. Используем команду `ps`, чтобы убедиться в запуске `imapd`:

```
# ps auxwww | grep imapd-ssl
root  1676  0.0  0.3  1940  500 ?  S  16:08  0:00 /usr/lib/ \
  courier-imap/libexec/couriertcpd -address=0 \
  -stderrlogger=/usr/lib/courier-imap/sbin/courierlogger \
  -stderrloggername=imapd-ssl -maxprocs=40 -maxperip=4 \
  -pid=/var/run/imapd-ssl.pid -nodnslookup -noidentlookup 993 \
  /usr/lib/courier-imap/bin/couriertls \
root  1680  0.0  0.2  1952  340 ?  S  16:08  0:00 /usr/lib/ \
  courier-imap/sbin/courierlogger imapd-ssl
root  1810  0.0  0.4  4600  564 pts/0  S  17:24  0:00 grep imapd-ssl
```

Примечание

Вы можете использовать сценарии в `/usr/lib/courier-imap/libexec`, скопировав их в свой каталог `init.d`, чтобы Courier запускался и останавливался автоматически при изменении уровня запуска.

Тестирование IMAP-сервера

Для того чтобы проверить, доступен ли ваш IMAP-сервер и могут ли пользователи зарегистрироваться на нем, подключитесь к порту 143 вашего сервера и откройте сеанс:

```
# telnet mail.example.com 143
* OK [CAPABILITY IMAP4rev1 UIDPLUS CHILDREN NAMESPACE \
    THREAD=ORDEREDSUBJECT THREAD=REFERENCES SORT QUOTA \
    IDLE ACL ACL2=UNION STARTTLS] Courier-IMAP ready. \
    Copyright 1998-2004 Double Precision, Inc.  \
    See COPYING for distribution information.
```

Увидев такое приветственное сообщение, зарегистрируйтесь:

```
. login bammbamm bamm_password
. OK LOGIN Ok.
```

Теперь выберите папку входящих сообщений и посмотрите, правильно ли все работает:

```
. select INBOX
* FLAGS (\Draft \Answered \Flagged \Deleted \Seen \Recent)
* OK [PERMANENTFLAGS (\* \Draft \Answered \Flagged \Deleted \Seen)] Limited
* 5 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 1089237749] Ok
* OK [MYRIGHTS "acdilrsw"] ACL
. OK [READ-WRITE] Ok
```

Наконец, отключитесь от сервера:

```
. logout
* BYE Courier-IMAP server shutting down
. OK LOGOUT completed
```

Тестирование IMAP по TLS

Для тестирования IMAP по протоколу TLS используем вместо `telnet` утилиту `s_client` из OpenSSL и подключаемся к порту 993 (порт IMAPS[1]). Этот клиент выводит достаточно много данных о проверке сертификата, но после того, как соединение установлено, вы можете использовать его так же, как делали это для предыдущего незашифрованного сеанса.

```
# openssl s_client -CAfile /usr/share/ssl/certs/cacert.pem -connect
localhost:993
CONNECTED(00000003)
depth=1 /C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=Certification
Authority/\
    CN=mail.example.com/emailAddress=certmaster@example.com
verify return:1
depth=0 /C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=IMAP \
    services/CN=mail.example.com/emailAddress=postmaster@example.com
verify return:1
---
Certificate chain
 0 s:/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=IMAP services/\
    CN=mail.example.com/emailAddress=postmaster@example.com
   i:/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=Certification
    Authority/ CN=mail.example.com/emailAddress=certmaster@example.com
---
Server certificate
-----BEGIN CERTIFICATE-----
```

[1] То есть IMAP поверх SSL. – *Примеч. науч. ред.*

```
MIIEDDCCA3WgAwIBAgIBAzANBgkqhkiG9w0BAQQFADCBsDELMAkGA1UEBhMCRVgx
ETAPBgNVBAgTCEV4YW1wbGlhMRMwEQYDVQQHEwpFeGFtcGxldG9uMRUwEwYDVQQK
EwxFeGFtcGxlIEluYy4xIDAeBgNVBAsTF0NlcnRpZmljYXRpb24gQXV0aG9yaXR5
MRkwFwYDVQQDExBtYWlsLmV4YW1wbGUuY29tMSUwIwYJKoZIhvcNAQkBFhZjZXJ0
bWFzdGVyQGV4YW1wbGUuY29tMB4XDTA0MDcxMzEzNTUzMloXDTA1MDcxMzEzNTUz
MlowgaYxCzAJBgNVBAYTAkVYMREwDwYDVQQIEwhFeGFtcGxpYTETMBEGA1UEBxMK
RXhhbXBsZXRvbjEVMBMGA1UEChMMRXhhbXBsZSBJbmMuMRYwFAYDVQQLEw1JTUFQ
IHNlcnZpY2VzMRkwFwYDVQQDExBtYWlsLmV4YW1wbGUuY29tMSUwIwYJKoZIhvcN
AQkBFhZwb3NObWFzdGVyQGV4YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GN
ADCBiQKBgQC95UUtw3dVVGghNLPEN3YBw/iKMkXtNhXllLAUEshZEIDGGjB1q9W8
QC4mLBOsWTYLTXWUbvoJHmBCmf6tzVvOi932r4KTDzanLP7EDc4tvg8ouhFxUEka
lVA+1g3l5oY8v1LIOYWxS8fpmRQENYHWncoShmXRPjg4wO6/2pZaawIDAQABo4IB
PDCCATgwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmFO
ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFFK61+FMcqcC3M/Em3X2I8JCn8JuMIHd
BgNVHSMEgdUwgdKAFMNGZ7/NorS6WpJQJZ2IhDno97iXoYG2pIGzMIGwMQswCQYD
VQQGEwJFWDERMA8GA1UECBMIRXhhbXBsaWExEzARBgNVBAcTCkV4YW1wbGVOb24x
FTATBgNVBAoTDEV4YW1wbGUgSW5jLjEgMB4GA1UECxMXQ2VydGlmaWNhdGlvbiBB
dXRob3JpdHkxGTAXBgNVBAMTEG1haWwuZXhhbXBsZS5jb2OxJTAjBgkqhkiG9w0B
CQEWFmNlcnRtYXNOZXJAZXhhbXBsZS5jb22CAQAwDQYJKoZIhvcNAQEEBQADgYEA
iqd/nOvihp1EWF+K7hgbptl9v13tzyuE3TMSI3oXtGnQtYNLTvx3eaYDBecUQaI1
q1ocQBsvz17+noz9jwD69UlBWUANwxDuObPHmnr7CeePVnv6fAyZ4Jg9x8vAPzDD
Nu/Tu88MOkEVQ2XT35oPM+gDy3Mw44NrYB2xhky8Ptg=
-----END CERTIFICATE-----
subject=/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=IMAP services/\
    CN=mail.example.com/emailAddress=postmaster@example.com
issuer=/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=Certification
Authority/\
    CN=mail.example.com/emailAddress=certmaster@example.com
---
No client certificate CA names sent
---
SSL handshake has read 1202 bytes and written 340 bytes
---
New, TLSv1/SSLv3, Cipher is AES256-SHA
Server public key is 1024 bit
SSL-Session:
    Protocol  : TLSv1
    Cipher    : AES256-SHA
    Session-ID:
7AA6E031976D8B3846F1B2C8FCBEBEB777C89BAD16E548C0D8FE0B170BF1D49B
    Session-ID-ctx:
    Master-Key: E41CE39B98EFF3395936404F7142D2804FA7BBE63ADB6A57F3FB51\
    3A756E6D55F548A5765AC27F99F862F46664131C72
    Key-Arg   : None
    Krb5 Principal: None
    Start Time: 1089732738
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
* OK [CAPABILITY IMAP4rev1 UIDPLUS CHILDREN NAMESPACE THREAD=ORDEREDSUBJECT \
```

```
    THREAD=REFERENCES SORT QUOTA IDLE AUTH=PLAIN ACL ACL2=UNION] Courier-IMAP
     ready. \
     Copyright 1998-2004 Double Precision, Inc.  See COPYING for distribution
     information.
. login bamm@example.com bamm_secret
. OK LOGIN Ok.
. logout
* BYE Courier-IMAP server shutting down
. OK LOGOUT completed
closed
```

Поздравляем! Вы стали обладателем работающего почтового сервера на основе LDAP. Но работа еще не закончена.

Тонкая настройка

Ваш почтовый сервер функционирует, но есть еще чем заняться. В настоящий момент каждый, кто может подключиться к LDAP-серверу, может извлечь секретные данные, например атрибут `userPassword`. Кроме того, все данные, извлекаемые различными серверами (включая Postfix), используют в своих LDAP-сеансах обычный текст.

Первое, что вам стоит сделать, – это разобраться с обозначенными проблемами безопасности. В этом разделе будет показано, как использовать связывание пользователей на LDAP-сервере, чтобы ограничить данные, отправляемые сервером LDAP клиенту. Кроме того, вы узнаете, как зашифровать сам сеанс LDAP. В качестве приятного дополнения вы сможете настроить аутентификацию SMTP на основе модуля `ldapdb` и использовать ее для введения политики компании, направленной на предотвращение возможных злоупотреблений со стороны адресов отправителей.

Расширение каталога

Управление доступом к каталогу LDAP и введение SMTP-аутентификации требует создания учетных записей для всех серверов, зависящих от каталога. Необходимо расширить дерево каталога, добавив еще одну ветвь, которая будет хранить серверы отдельно от почтовых пользователей. Новая ветвь `ou=auth,dc=example,dc=com`, хранящая объекты класса `account`, изображена на рис. 19.3.

В этом каталоге в текстовом виде будем использовать такие атрибуты и схемы:

```
dn: ou=auth,dc=example,dc=com
ou: auth
objectClass: organizationalUnit
dn: uid=postfix,ou=auth,dc=example,dc=com
uid: postfix
objectClass: account
```

```
objectClass: simpleSecurityObject
description: Postfix Bind user
userPassword: {CRYPT}9lGRsJNHN5DrI
dn: uid=couriermaildrop,ou=auth,dc=example,dc=com
uid: couriermaildrop
objectClass: account
objectClass: simpleSecurityObject
description: Courier Maildrop Bind user
userPassword: {CRYPT}lA8iQdmwZRC86
dn: uid=courierimap,ou=auth,dc=example,dc=com
uid: courierimap
objectClass: account
objectClass: simpleSecurityObject
description: Courier IMAP Bind user
userPassword: {CRYPT}S1t1/3ENmjk1s
dn: uid=ldapdb,ou=auth,dc=example,dc=com
uid: ldapdb
objectClass: inetOrgPerson
givenName: ldapdb
sn: ldapdb
cn: ldapdb
userPassword: AvaAgO7i
mail: ldapdb
saslAuthzTo: ldap:///
ou=people,dc=example,dc=com??sub?(objectclass=inetOrgPerson)
```

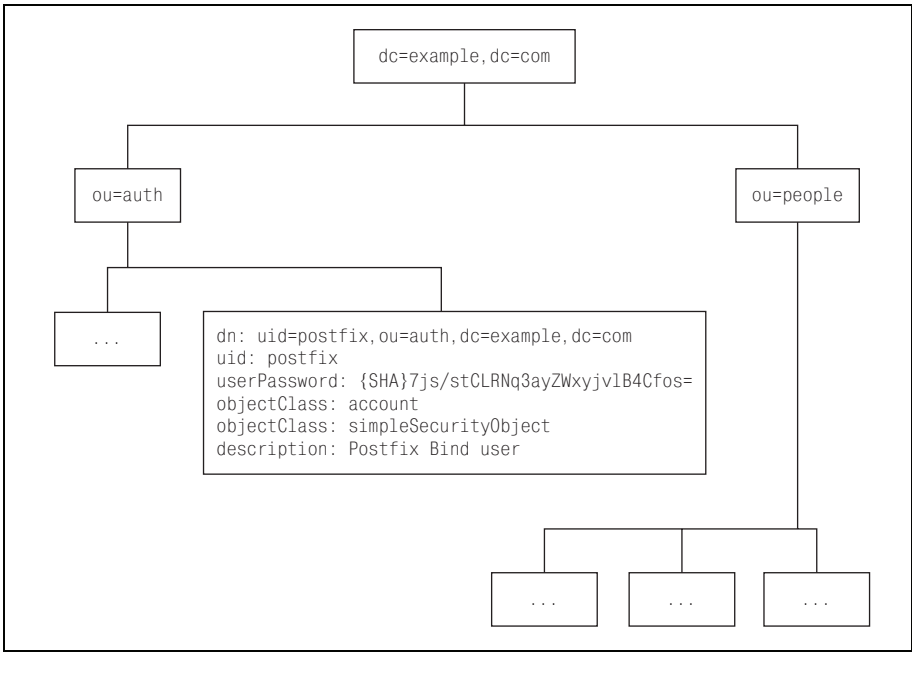

Рис. 19.3. *Ветвь аутентификации для компании Example Inc.*

Каждый сервер, обращающийся к каталогу LDAP через модуль `ldapdb` (Postfix, Courier maildrop, Courier IMAP и Cyrus SASL), будет иметь собственную учетную запись. Атрибуты объекта `uid=ldapdb, ou=auth, dc=example, dc=com` отличаются от остальных атрибутов:

- Пароль хранится в виде открытого текста для поддержки работы механизма аутентификации общего секрета DIGEST-MD5 (этот механизм не может обращаться к зашифрованным паролям).
- Новый атрибут `mail` используется для аутентификации пользователей вместо `uid`.
- Атрибут `saslAuthzTo` определяет, где модуль `ldapdb` может взять идентификационные данные другого пользователя.

Добавление аутентификации для серверов

Первое, что мы реализуем, – это пересылка почты по результатам SMTP-аутентификации, чтобы мобильные и удаленные пользователи с IP-адресами вне доверенной локальной сети могли отправлять почту. Мы уже занимались этим без использования LDAP в главе «Введение в SMTP-аутентификацию», теперь же будем собирать и настраивать Cyrus SASL с плагином `ldapdb`. Можно было бы использовать автономный демон saslauthd, но, к сожалению, он работает только с механизмами открытого текста. Когда вы прочтете этот раздел, ваш Postfix-сервер сможет предложить почтовым клиентам механизмы аутентификации открытого текста (`PLAIN`, `LOGIN`) и общего секрета (`CRAM-MD5`, `DIGEST-MD5`).

Процесс аутентификации слегка отличается от описанного в главе 16 при обсуждении взаимодействия Postfix и Cyrus SASL, т. к. он использует для проверки аутентификации идентификатор `authorization ID`. Этот идентификатор появился в проекте Cyrus IMAP, который позволял группе пользователей действовать от имени одного пользователя. Например, вы можете так настроить Cyrus IMAP, чтобы позволить другим пользователям читать почту пользователя, находящегося в отпуске, не меняя никакие пароли.

Процесс аутентификации `ldapdb` работает следующим образом:

1. Почтовый клиент, использующий SMTP-аутентификацию, подключается к серверу Postfix и передает `username` (и идентификационную фразу, если применяется механизм открытого текста). В контексте SASL это имя пользователя является *идентификатором аутентификации*.
2. Postfix-демон `smtpd` обращается к библиотеке SASL для проверки верительных данных. Библиотека SASL поручает выполнение этой задачи плагину `ldapdb`, который вступает в контакт с сервером LDAP.
3. Конфигурация `ldapdb` для Postfix-демона `smtpd` содержит еще одно имя пользователя `username` и идентификационную фразу `passphrase` (`ldapdb` передает идентификационную фразу, только если клиент использует механизм открытого текста). В контексте SASL это имя

пользователя называется *идентификатором авторизации*, т. к. оно имеет право на извлечение passphrase из имени пользователя (идентификатора аутентификации).

4. Если ldapdb удается извлечь passphrase из сервера LDAP, он сравнивает эту строку со значением passphrase, предоставленным почтовым клиентом. Способ проверки зависит от используемого почтовым клиентом механизма аутентификации. Для механизмов открытого текста применяется простое сравнение строк, а для механизмов общего секрета ldapdb вычисляет строки, используя passphrase, а затем уже сравнивает их.

   Если установлено соответствие, то аутентификация считается успешной, ldapdb передает информацию обратно в SASL, который сообщает серверу Postfix о том, что можно разрешить пересылку.

Установка патча ldapdb

Плагин ldapdb – это вклад в Cyrus SASL архитектора OpenLDAP Говарда Чу (Howard Chu), который сделал возможным использование механизмов общего секрета для аутентификации SASL. Планировалось, что ldapdb войдет в будущие версии Cyrus SASL, но в текущей (Cyrus SASL 2.1.19) его пока нет. Чтобы упростить процедуру получения поддержки ldapdb для Cyrus SASL, мы создали патч, который вы можете загрузить с веб-сайта *http://www.postfix-book.com*.

Для того чтобы применить патч к файлам исходных текстов Cyrus SASL, распакуйте свежий дистрибутив SASL и перейдите в только что созданный каталог. Установите патч, как показано в следующем примере:

```
# patch -p1 < ../cyrus-sasl-2.1.19-ldapdb.patch
patching file config/openldap.m4
patching file configure.in
patching file doc/install.html
patching file doc/options.html
patching file doc/readme.html
patching file doc/sysadmin.html
patching file lib/staticopen.h
patching file plugins/ldapdb.c
patching file plugins/Makefile.am
patching file plugins/makeinit.sh
```

В случае успешного выполнения соберите Cyrus SASL следующим образом (обратите внимание на параметры --with-ldap*):

```
# ./configure \
   --with-plugindir=/usr/lib/sasl2 \
   --disable-java \
   --disable-krb4 \
   --with-dblib=berkeley \
   --with-saslauthd=/var/state/saslauthd \
```

```
--without-pwcheck \
--with-devrandom=/dev/urandom \
--enable-cram \
--enable-digest \
--enable-plain \
--enable-login \
--disable-otp \
--with-ldap=/usr \
--with-ldapdb
```

После завершения настройки вы можете установить или обновить текущую версию SASL, запустив `make install`. Если вы этого еще не сделали, создайте символическую ссылку из `/usr/local/lib/sasl2` на `/usr/lib/sasl2`, т. к. SASL будет считать, что библиотеки находятся в данном каталоге (для параметров конфигурации, использованных в данном примере).

В итоге вы должны увидеть новые библиотеки `ldapdb` в каталоге `/usr/lib/sasl2`:

```
# ls -la libldapdb.*
-rwxr-xr-x  1 root root  702 Jul 16 20:43 libldapdb.la
lrwxrwxrwx  1 root root   19 Jul 16 20:43 libldapdb.so -> libldapdb.so.2.0.19
lrwxrwxrwx  1 root root   19 Jul 16 20:43 libldapdb.so.2 -> libldapdb.so.2.0.19
-rwxr-xr-x  1 root root  94948 Jul 16 20:43 libldapdb.so.2.0.19
```

Настройка ldapdb

Как уже отмечалось ранее в разделе «Добавление аутентификации для серверов», чтобы сделать SASL доступным для Postfix, необходимо создать конфигурацию SASL для Postfix-демона `smtpd` в файле `/usr/lib/sasl2/smtpd.conf`, например:

```
pwcheck_method: auxprop
auxprop_plugin: ldapdb
mech_list: PLAIN LOGIN DIGEST-MD5 CRAM-MD5
ldapdb_uri: ldap://mail.example.com
ldapdb_id: proxyuser
ldapdb_pw: proxy_secret
ldapdb_mech: DIGEST-MD5
log_level: 7
```

С LDAP связаны два дополнительных параметра конфигурации: `ldapdb_id` и `ldapdb_pw`. Имя пользователя в данном случае является идентификатором авторизации. (Помните, что плагину `ldapdb` необходим `smtpd` для использования идентификатора авторизации, прежде чем он сможет представлять почтового пользователя и повторно аутентифицироваться.)

Настройка политики авторизации OpenLDAP

Postfix настроен для связывания (bind) с сервером OpenLDAP; теперь необходимо настроить OpenLDAP `slapd` для аутентификации пользова-

теля и его авторизации с тем, чтобы он стал другим пользователем. Сначала выбираем политику, которая определяет, каким образом slapd будет обрабатывать аутентифицированных пользователей. Устанавливаем параметр sasl-authz-policy в файле slapd.conf в одно из перечисленных ниже значений:

- `none` — Отменяет авторизацию. Значение по умолчанию.
- `from` — Требует, чтобы каждый пользователь LDAP явно разрешил одному или нескольким пользователям действовать как идентификатор авторизации. Для того чтобы разрешить авторизацию, они должны добавить атрибут saslAuthzFrom в их собственный пользовательский объект. Этот атрибут содержит dn (отличительное имя) пользователя, который может присваивать себе их личность.
- `to` — Разрешает всем пользователям действовать как идентификатор авторизации по умолчанию. При использовании данной политики пользователь добавляет в собственный пользовательский объект атрибут saslAuthzTo, который определяет, где разрешено присваивать личность другого пользователя. Администратор каталога LDAP должен создать правило, которое позволяет лишь ограниченному числу пользователей присваивать себе другие личности (для предотвращения злоупотреблений).
- `both` — Активирует обе политики: from и to.

Примечание

Более подробная информация приведена в главе 10 руководства администратора OpenLDAP («OpenLDAP Administrator's Guide») по адресу *http://www.openldap.org/doc/admin22/sasl.html#SASL%20Proxy%20Authorization*.

В нашей книге мы будем использовать политику to, т. к. нам кажется, что администратору проще ограничить круг пользователей, которые могут принимать личности других пользователей, чем добавлять новый атрибут в каждый пользовательский объект. Для того чтобы настроить политику для рассматриваемого в книге примера, добавьте в slapd.conf следующую строку:

```
sasl-authz-policy       to
```

Настройка OpenLDAP-связываний на основе SASL

Последний этап организации связывания с сервером OpenLDAP на основе SASL состоит в настройке slapd с той ветвью дерева каталога, где хранятся верительные данные. Вам также нужно указать атрибут запроса для заданного идентификатора аутентификации. Используем фильтр поиска sasl-regexp, определяющий, где демон smtpd должен искать пользователей (идентификатор авторизации), которых нужно аутентифицировать.

Помещаем фильтр в файл конфигурации slapd.conf:

```
sasl-regexp
    uid=(.*),cn=.*,cn=auth
    ldap:///dc=example,dc=com??sub?(&(objectclass=inetOrgPerson)(mail=$1))
```

Примечание

Предыдущее регулярное выражение создано для поиска не `uid` (регистрационного имени), а почтового адреса. Это сделано специально, т. к. многие интернет-провайдеры и провайдеры почтовых услуг используют почтовый адрес в качестве регистрационного имени пользователя. Существует множество причин этого явления, приведем две основные: пользователю приходится запоминать на одно значение меньше и проще обмениваться аутентификационными данными с другими системами, такими как серверы Radius.

Фильтр на месте, мы перезапускаем сервер OpenLDAP, проверяем, нет ли ошибок в журнале, и приступаем к тестированию.

Тестирование плагина ldapdb

В конфигурации `ldapdb` следует проверить три компонента:

- Механизм ограничений
- Прямое связывание на основе SASL
- Аутентификация на основе `ldapdb`

Тестирование механизма ограничений

Убедитесь в том, что работает параметр `mech_list`, заданный в файле конфигурации `/usr/lib/sasl2/slapd.conf`. Выполните команду `ldapsearch` для вывода механизмов, которые `slapd` предлагает клиентам:

```
# ldapsearch -LLL -x -s base -b "" "(objectClass=*)" supportedSASLMechanisms
dn:
supportedSASLMechanisms: DIGEST-MD5
```

Если ваши выходные данные выглядят так же, значит, все хорошо. В противном случае проверьте, нет ли опечаток в файле `slapd.conf`.

Тестирование прямого связывания на основе SASL

Используйте команду `ldapwhoami` для проверки того, может ли пользователь в вашем каталоге напрямую связываться с `slapd` при помощи SASL (без плагина `ldapdb`):

```
# ldapwhoami -U bamm@example.com
SASL/DIGEST-MD5 authentication started
Please enter your password:
SASL username: bamm@example.com
SASL SSF: 128
SASL installing layers
dn:uid=bammbamm,ou=it,ou=people,dc=example,dc=com
```

Тем самым проверяется, можем ли мы на самом деле аутентифицироваться как пользователь. В случае неудачи дальше ничего уже не по-

лучится, и вам необходимо вернуться к разделу «Настройка OpenLDAP-связываний на основе SASL» и посмотреть, не сделали ли вы ошибок при настройке.

Тестирование ldapdb-аутентификации

Cyrus SASL включает в себя утилиты `server` и `client`, которые позволяют протестировать аутентификацию независимо от других серверов. Их использование избавляет от возможных побочных эффектов таких пакетов, как Postfix, и позволяет свести на нет проблемы, связанные с SASL, в том числе и вызванные применением `ldapdb`.

Для проверки аутентификации сначала создаем символическую ссылку из `smtpd.conf` на `sample.conf` для использования утилитой `server`:

```
# cd /usr/lib/sasl2/
# ln -s smtpd.conf sample.conf
```

Теперь открываем терминальное окно и запускаем `server`:

```
# ./server -s rcmd -p 23
trying 10, 1, 6
socket: Address family not supported by protocol
trying 2, 1, 6
```

Открываем второе терминальное окно и запускаем утилиту `client` с LDAP-идентификатором `uid` пользователя на вашем LDAP-сервере. Программа запрашивает идентификаторы аутентификации и авторизации. Вот как следует поступить для пользователя `bammbamm` (`dn: uid=bammbamm,ou=it,ou=people,dc=example,dc=com`):

```
# ./client -s rcmd -p 23 -m PLAIN mail.example.com
receiving capability list... recv: {31}
LOGIN PLAIN DIGEST-MD5 CRAM-MD5
LOGIN PLAIN DIGEST-MD5 CRAM-MD5
please enter an authentication id: bammbamm
please enter an authorization id: bammbamm
Password:
send: {5}
PLAIN
send: {1}
Y
send: {29}
bammbamm[0]bammbamm[0]bamm_secret
successful authentication
closing connection
```

Примечание

Идентификатор авторизации будет заменен значением, которое указано для параметра `ldapdb_id` в файле `sample.conf`, но вам все же нужно ввести что-то в ответ на приглашение на ввод.

Если вы видите в выводе слова `successful authentication` (как в предыдущем примере), то аутентификация на основе `ldapdb` работает. В противном случае обратитесь к своему журналу аутентификации и журналам Cyrus SASL, чтобы попытаться понять, что пошло не так.

Защита данных каталога

Из соображений безопасности вы наверняка не захотите, чтобы все данные на вашем LDAP-сервере могли быть прочитаны каждым, кто обращается к этому серверу. Вы можете ограничить доступ к серверу на чтение, используя списки контроля доступа для пользователей, связывающихся с LDAP-сервером. Для этого необходимо выполнить две операции:

1. Настроить сервер LDAP для ограничения доступа на чтение.
2. Настроить зависящие от LDAP пакеты для связывания с сервером LDAP.

Ограничение доступа к чтению каталога

Ранее в разделе «Расширение каталога» мы создали связанных пользователей для нашего каталога. Теперь нам необходимо сообщить `slapd`, к каким частям каталога могут обращаться связанные пользователи, а также что могут делать другие (например, анонимные) пользователи.

Добавляем в файл `/etc/openldap/slapd.conf` следующие правила:

```
# Правила доступа для saslAuthzTo
access to dn.subtree="dc=example,dc=com" attr=saslAuthzTo
    by dn.base="cn=Manager,dc=example,dc=com"                    write
    by *                                                         read
# Правила доступа для userPassword
# Аутентифицированные пользователи (сами) и Администратор каталога могут
изменять собственные пароли.
# Остальные могут обращаться к паролям в процессе аутентификации.
access to dn.subtree="dc=example,dc=com"
    attr=userPassword
    by self                                                      write
    by dn.base="cn=Manager,dc=example,dc=com"                    write
    by dn.base="uid=courierimap,ou=auth,dc=example,dc=com"       read
    by *                                                         auth
# Правила доступа для uidNumber, gidNumber, mailbox, homeDirectory, quota
# Изменять эти значения может только Администратор.
# Приложения, которым нужны эти значения, могут читать их.
# Аутентифицированные пользователи могут читать собственные данные.
# Все остальные не имеют доступа к этим данным.
access to dn.subtree="dc=example,dc=com"
    attr=uidNumber attr=gidNumber attr=mailbox attr=homeDirectory attr=quota
    by dn.base="cn=Manager,dc=example,dc=com"                    write
    by dn.base="uid=courierimap,ou=auth,dc=example,dc=com"       read
```

```
    by dn.base="uid=couriermaildrop,ou=auth,dc=example,dc=com"  read
    by dn.base="uid=postfix,ou=auth,dc=example,dc=com"          read
    by self                                                     read
    by *                                                        none
# Правила доступа для атрибутов mail, uid и maildrop
# Приложения могут обращаться к этим атрибутам.
# Аутентифицированные пользователи также могут обращаться к этим атрибутам.
# Все остальные также могут читать эти значения.
access to dn.subtree="dc=example,dc=com"
    attrs=mail attr=uid attr=maildrop
    by dn.base="uid=courierimap,ou=auth,dc=example,dc=com"      read
    by dn.base="uid=postfix,ou=auth,dc=example,dc=com"          read
    by dn.base="uid=ldapdb,ou=auth,dc=example,dc=com"           read
    by self                                                     read
    by *                                                        read
# Общее правило
# Все атрибуты, которые не были упомянуты ранее, доступны
# на чтение кому угодно.
access to * by * read
```

После добавления правил перезапустите свой LDAP-сервер.

Это достаточно простые правила ограничения доступа, которые хорошо подходят для схемы и приложений данной главы. Вы можете создать более точные правила (за информацией обратитесь к странице руководства slapd.access(5) или к одной из свежих книг по LDAP).

Настройка клиентов LDAP как связанных пользователей

После того как определены правила контроля доступа, необходимо настроить каждый пакет, обращающийся к серверу LDAP, так, чтобы он подключался к серверу как связанный пользователь.

Настройка Postfix как связанного пользователя

Для того чтобы Postfix подключался к серверу LDAP как связанный пользователь, необходимо изменить три параметра в файлах конфигурации в каталоге `/etc/postfix/ldap`: `bind`, `bind_dn` и `bind_pw`.

- Параметр `bind` необходимо установить в значение `yes` для использования связанного пользователя.
- Параметр `bind_dn` задает отличительное имя связанного пользователя. В примере данной главы мы указываем `bind_dn = uid=postfix, ou=auth, dc=example,dc=com`.
- Параметр `bind_pw` определяет пароль (открытым текстом).

Примечание

Эти данные необходимо обезопасить, т. к. они включают в себя пароль. Именно поэтому в начале главы мы создали отдельный каталог для файлов конфигурации запросов LDAP.

В результате этих действий все ваши файлы конфигурации в каталоге /etc/postfix/ldap должны содержать такие строки:

```
bind = yes
bind_dn = uid=postfix,ou=auth,dc=example,dc=com
bind_pw = Yanggt!
```

Настройка Courier maildrop как связанного пользователя

Параметры Courier maildrop практически совпадают с параметрами для Postfix, с тем лишь отличием, что среди них нет параметра для включения системы связывания (если вы не укажете отличительное имя для связывания, то Courier не будет пытаться осуществить связывание). Поэтому мы помещаем в свой файл /etc/maildropldap параметры binddn и bindpw, и конфигурация должна выглядеть примерно так:

```
binddn      uid=couriermaildrop,ou=auth,dc=example,dc=com
bindpw      Yrj6Hl6r
```

Предупреждение

Теперь, когда пароль задан открытым текстом, убедитесь в том, что /etc/maildropldap доступен только пользователю root.

Настройка Courier IMAP как связанного пользователя

Последним настраиваем для работы в качестве связанного пользователя клиент Courier IMAP. Как и в случае с Courier maildrop, вы задаете всего два параметра в файле /usr/lib/courier-imap/etc/authldaprc: LDAP_BINDDN и LDAP_BINDPW.

```
LDAP_BINDDN      uid=courierimap,ou=auth,dc=example,dc=com
LDAP_BINDPW      X5mYpl6p
```

Предупреждение

Как и предыдущем случае, убедитесь в том, что файл конфигурации IMAP LDAP (/usr/lib/courier-imap/etc/authldaprc) доступен только пользователю root.

Проверка серверных ограничений

Для тестирования серверов вспомним, о чем мы говорили ранее в разделе «Тестирование получателей LDAP»: нам необходимо лишь убедиться в том, что каждая составляющая может извлекать данные с сервера LDAP. Если возникнут проблемы, обратитесь к файлам журналов. В частности, посмотрите внимательно на файл журнала slapd. Если вам нужно будет изменить уровень журналирования slapd, измените значение параметра loglevel (см. страницу руководства slapd.conf(5)).

Шифрование LDAP-запросов

Мы уже защищаем данные на LDAP-сервере от неавторизованных пользователей, но еще не сделали ничего для того, чтобы обезопасить передачу данных по небезопасной сети. В этом разделе будет показано, как использовать TLS для защиты уровня передачи данных.

Настройка TLS для OpenLDAP

Прежде чем приступать к настройке TLS для вашей версии LDAP, проверьте, имеется ли поддержка TLS в `slapd`. При наличии таковой запуск `ldd` для `slapd` обычно выводит зависимости от библиотеки SSL:

```
# ldd /usr/sbin/slapd
        libslapd_db-4.1.so => /usr/lib/libslapd_db-4.1.so (0x00116000)
        libsasl2.so.2 => /usr/lib/libsasl2.so.2 (0x00101000)
        libkrb5.so.3 => /usr/lib/libkrb5.so.3 (0x00a6f000)
        libk5crypto.so.3 => /usr/lib/libk5crypto.so.3 (0x00ad8000)
        libcom_err.so.2 => /lib/libcom_err.so.2 (0x00a6a000)
        libssl.so.4 => /lib/libssl.so.4 (0x00b11000)
        libcrypto.so.4 => /lib/libcrypto.so.4 (0x00977000)
        libcrypt.so.1 => /lib/libcrypt.so.1 (0x008d0000)
        libresolv.so.2 => /lib/libresolv.so.2 (0x00965000)
        libdl.so.2 => /lib/libdl.so.2 (0x008b8000)
        libwrap.so.0 => /usr/lib/libwrap.so.0 (0x00bfe000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x00912000)
        libc.so.6 => /lib/libc.so.6 (0x0076e000)
        libgssapi_krb5.so.2 => /usr/lib/libgssapi_krb5.so.2 (0x00afc000)
        libz.so.1 => /usr/lib/libz.so.1 (0x008bd000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00759000)
        libnsl.so.1 => /lib/libnsl.so.1 (0x008fe000)
```

Для настройки `slapd` с TLS необходимо выполнить следующие действия:

1. Создать X509-сертификат для `slapd`.
2. Настроить `slapd` так, чтобы он предлагал TLS.
3. Настроить клиенты LDAP для использования TLS.

Создание X509-сертификатов для slapd

Как и для любого другого сервера, предлагающего шифрование на основе SSL, вам необходимо создать для `slapd` сертификаты, содержащие открытый и секретный ключи. Создание сертификатов описано в главе 17.

Если у вас работает собственный центр сертификации, то можно создать секретный ключ `slapd_private_key.pem` для `slapd` так:

```
# openssl req -new -nodes -keyout slapd_private_key.pem -out \
  slapd_private_key.pem -days 365
```

Соответствующая команда для создания открытого ключа `slapd_public_cert.pem` будет такой:

```
# openssl ca -policy policy_anything -out slapd_public_cert.pem -infiles \
```

```
slapd_private_key.pem
```

Теперь создаем в каталоге `/etc/openldap` подкаталог `certs` для ключей. После того как вы скопируете файлы `.pem` в подкаталог, он должен выглядеть так:

```
# ls -la /etc/openldap/certs/
total 24
drwx------    2 ldap     ldap         4096 Jun 21 22:31 .
drwxr-xr-x    4 root     root         4096 Jun 21 23:12 ..
-rw-------    1 ldap     ldap         1624 Jun 21 22:31 slapd_private_key.pem
-rw-------    1 ldap     ldap         3807 Jun 21 22:31 slapd_public_cert.pem
```

Предупреждение

Убедитесь в том, что вы изменили права собственности и привилегии на файлы так, чтобы они были доступны только пользователю, который запускает `slapd`.

Настройка slapd для предоставления TLS

Для того чтобы рассказать `slapd` о новых файлах сертификатов, необходимо добавить четыре параметра в файл `slapd.conf`:

```
TLSCACertificateFile /usr/share/ssl/certs/cacert.pem
TLSCertificateFile /etc/openldap/certs/slapd_public_cert.pem
TLSCertificateKeyFile /etc/openldap/certs/slapd_private_key.pem
TLSVerifyClient demand
```

Параметр `TLSCACertificateFile` указывает местоположение CA-сертификата. В данном примере `slapd` ищет данный сертификат в единственном файле, но вам может потребоваться и нечто большее. В этом случае не указывайте имя файла, тогда `slapd` будет осуществлять просмотр всего каталога. Вам также нужно будет использовать утилиту `c_rehash` из набора утилит OpenSSL для индексации вашего каталога (см. главу 18).

Примечание

CA-сертификаты должны находиться в таком месте, которое доступно для остальных клиентов LDAP.

Параметр `TLSVerifyClient` вы можете установить в значение `never`, `allow`, `try` или `demand` для ограничения доступа к некоторым видам аутентификации. Вам нужно быть уверенными в том, что настройки TLS работают на стороне клиента, поэтому устанавливаем параметр в значение `demand`, чтобы заставить клиенты использовать TLS. Так можно будет без труда определить, поддерживает ли клиент TLS.

Настройка TLS для клиентов LDAP

В этом разделе будет показано, как включить клиентскую поддержку LDAP в Postfix и Courier IMAP. В настоящее время Courier maildrop

не поддерживает TLS, но на самом деле это не представляет проблемы, т. к. локальному агенту доставки не нужно знать никакие пользовательские пароли.

Включение поддержки LDAP в Postfix

Для обращения к серверу LDAP по TLS для Postfix требуется собственный открытый сертификат и секретный ключ. Вы уже создавали файлы ключей в главе 17, и теперь можете использовать их вновь.

Включение TLS для LDAP требует добавления пяти параметров во все файлы конфигурации запросов LDAP в каталоге `/etc/postfix/ldap`:

```
version = 3
tls_ca_cert_file = /usr/share/ssl/certs/cacert.pem
tls_cert = /etc/postfix/certs/postfix_public_cert.pem
tls_key = /etc/postfix/certs/postfix_private_key.pem
start_tls = yes
```

Параметры работают следующим образом:

`version`

Указывает версию протокола LDAP. По умолчанию Postfix использует версию 2, но для TLS необходима версия 3.

`tls_ca_cert_file`

Задает CA-сертификат.

`tls_cert`

Задает открытый сертификат для Postfix.

`tls_key`

Задает секретный ключ для Postfix.

`start_tls`

Включает TLS.

Разрешение запросов LDAP для Courier IMAP

Настройка Courier IMAP для обращения к серверу LDAP по TLS будет отличаться от настройки Postfix, т. к. Courier не имеет собственного клиента LDAP, а использует клиент OpenLDAP. Необходимо настроить клиент OpenLDAP для использования TLS, а затем указать Courier IMAP, что следует запрашивать LDAP по TLS из клиента OpenLDAP.

Обычно файл конфигурации клиента OpenLDAP – это `/etc/openldap/ldap.conf`. Для включения поддержки TLS следует добавить в него следующие параметры `TLS_*`:

```
URI ldap://mail.example.com
BASE dc=example,dc=com
TLS_CACERT /usr/share/ssl/certs/cacert.pem
TLS_CERT /etc/openldap/certs/slapd_public_cert.pem
TLS_KEY /etc/openldap/certs/slapd_private_key.pem
TLS_REQCERT demand
```

Эти параметры похожи на параметры настройки для Postfix:

`TLS_CACERT`

Задает CA-сертификат, который используют остальные пакеты данной главы.

`TLS_CERT`

Задает открытый сертификат клиента.

`TLS_KEY`

Задает секретный ключ клиента. Если хотите, можете использовать сертификат и ключ `slapd`.

`TLS_REQCERT`

Определяет политику для запроса серверного сертификата. Установите его в значение `demand`, чтобы потребовать использования TLS или закрытия соединения.

Вы уже сообщили клиенту OpenLDAP о сертификате и ключах, так что осталось только в файле `/usr/lib/courier-imap/etc/authldaprc` указать Courier IMAP, что следует запрашивать TLS для LDAP:

```
LDAP_TLS        1
```

Перезапускаем Courier IMAP. Переходим к тестированию.

Тестирование TLS

Параметры TLS, которые мы только что добавили в конфигурацию, значительно усложняют нашу систему, так что важно их тщательно проверить.

1. Проверяем приложение, к которому обращаются все остальные, – сервер LDAP. Проверяем собственный клиент сервера LDAP.
2. Если они работают, смотрим, может ли Postfix подключиться к серверу LDAP. Проверяем, может ли Courier IMAP использовать клиент LDAP для получения SSL.

Тестирование сервера OpenLDAP

Начинаем с проверки того, предлагает ли сервер OpenLDAP сертификаты TLS. Не существует простой команды для осуществления такой проверки на порте 389 (где обычно находится TLS1 для LDAP). Посмотрим, можно ли использовать порт 636 (где OpenLDAP предоставляет SSL), чтобы проверить, подойдут ли сертификаты для `slapd`.

Используем утилиту `s_client` для подключения к серверу:

```
# openssl s_client -CAfile /usr/share/ssl/certs/cacert.pem -connect
  localhost:636
CONNECTED(00000003)
depth=1 /C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=Certification
Authority/\
    CN=mail.example.com/emailAddress=certmaster@example.com
```

```
verify return:1
depth=0 /C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=LDAP services/\
    CN=mail.example.com/emailAddress=ldapmaster@example.com
verify return:1
---
Certificate chain
 0 s:/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=LDAP services/\
    CN=mail.example.com/emailAddress=ldapmaster@example.com
   i:/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=Certification
   Authority/\
    CN=mail.example.com/emailAddress=certmaster@example.com
 1 s:/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=Certification
    Authority/\
    CN=mail.example.com/emailAddress=certmaster@example.com
   i:/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=Certification
    Authority/\
    CN=mail.example.com/emailAddress=certmaster@example.com
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIEDDCCA3WgAwIBAgIBAjANBgkqhkiG9w0BAQQFADCBsDELMAkGA1UEBhMCRVgx
ETAPBgNVBAgTCEV4YW1wbGlhMRMwEQYDVQQHEwpFeGFtcGxldG9uMRUwEwYDVQQK
EwxFeGFtcGxlIEluYy4xIDAeBgNVBAsTF0NlcnRpZmljYXRpb24gQXV0aG9yaXR5
MRkwFwYDVQQDExBtYWlsLmV4YW1wbGUuY29tMSUwIwYJKoZIhvcNAQkBFhZjZXJO
bWFzdGVyQGV4YW1wbGUuY29tMB4XDTA0MDcxMzEzNTQwNloXDTA1MDcxMzEzNTQw
NlowgaYxCzAJBgNVBAYTAkVYMREwDwYDVQQIEwhFeGFtcGxpYTETMBEGA1UEBxMK
RXhhbXBsZXRvbjEVMBMGA1UEChMMRXhhbXBsZSBJbmMuMRYwFAYDVQQLEw1MREFQ
IHNlcnZpY2VzMRkwFwYDVQQDExBtYWlsLmV4YW1wbGUuY29tMSUwIwYJKoZIhvcN
AQkBFhZsZGFwbWFzdGVyQGV4YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GN
ADCBiQKBgQDcqVcyPn4qhI65sAdPgu+Et2vzWsyHT/IT39mZ6Gqrh150a/eQA7Lz
GmUKR/t/W4ol28ygN/udpkHZiTUDjUC5ENF7kqk4vnx/4DpwmDmOjNgO7JJEr0FL
cOJl/KqZzAItBh32KtIhV8BQcdlfzdoxEqO7MkxRw1pu7LyLo5qOkwIDAQABo4IB
PDCCATgwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmFO
ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFCKdRZglsm4/1io2sltD1riyCE+KMIHd
BgNVHSMEgdUwgdKAFMNGZ7/NorS6WpJQJZ2IhDno97iXoYG2pIGzMIGwMQswCQYD
VQQGEwJFWDERMA8GA1UECBMIRXhhbXBsaWExEzARBgNVBAcTCkV4YW1wbGVOb24x
FTATBgNVBAoTDEV4YW1wbGUgSW5jLjEgMB4GA1UECxMXQ2VydGlmaWNhdGlvbiBB
dXRob3JpdHkxGTAXBgNVBAMTEG1haWwuZXhhbXBsZS5jb20OxJTAjBgkqhkiG9w0B
CQEWFmNlcnRtYXNOZXJAZXhhbXBsZS5jb22CAQAwDQYJKoZIhvcNAQEEBQADgYEA
AZCH5A23WVdIdO9NkD23Bz3HF+MyOf8fUx1CaQbLwo572mjgB/O3H7K969bU/te2
BeLOjifMo/vexXPMeajwzDnIKm/yJO7eNt85eeKciI6MZJVhvuPvtp/Rc5vArcas
HNqpmm7oDAEFIRclHsfhsyAHwsTTr18UGndfL3Hetkw=
-----END CERTIFICATE-----
subject=/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=LDAP services/\
    CN=mail.example.com/emailAddress=ldapmaster@example.com
issuer=/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=Certification
Authority/CN=mail.example.com/emailAddress=certmaster@example.com
---
Acceptable client certificate CA names
/C=EX/ST=Examplia/L=Exampleton/O=Example Inc./OU=Certification Authority/\
    CN=mail.example.com/emailAddress=certmaster@example.com
---
```

```
SSL handshake has read 2402 bytes and written 352 bytes
---
New, TLSv1/SSLv3, Cipher is AES256-SHA
Server public key is 1024 bit
SSL-Session:
    Protocol  : TLSv1
    Cipher    : AES256-SHA
    Session-ID: 21430E35213A797176B28B16BF24D20EC9019902B5B09
                FCEDDA0333682FD6F7D
    Session-ID-ctx:
    Master-Key: 45636217FD3136A536CE62618DBC1CA92E6E0B1E773F75120632
                F761C289943BB85F78369C622A0D78DB60726147465F
    Key-Arg   : None
    Krb5 Principal: None
    Start Time: 1089733870
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
QUIT
DONE
```

Если получен код возврата `0 (ok)`, сертификаты проверены и подтверждены. Далее проверяем, можно ли использовать клиент LDAP для обращения к серверу по TLS.

Тестирование клиента OpenLDAP

Для того чтобы проверить, работает ли клиент OpenLDAP, попробуем извлечь данные при помощи команды `ldapsearch`:

```
# ldapsearch -ZZ -x -LLL "(mail=bamm@example.com)" userPassword
dn: ou=people,dc=example,dc=com
dn: ou=it,ou=people,dc=example,dc=com
dn: uid=bammbamm,ou=it,ou=people,dc=example,dc=com
userPassword:: YmFtbV9zZWNyZXQ=
```

Такая команда `ldapsearch` особенно полезна благодаря тому, что она читает настройки по умолчанию в файле `ldap.conf` (если вы помните, TLS/LDAP-конфигурация Courier IMAP от них зависит). Кроме того, этот запрос имитирует запрос Courier IMAP. Параметр `-ZZ` вынуждает `ldapsearch` использовать TLS.

Если эта команда не работает, исследуйте свой файл `ldap.conf` и файлы журналов.

Тестирование Postfix

Чтобы проверить, поддерживает ли Postfix LDAP в сочетании с TLS, выполним команду `postmap`, запрашивающую в каталоге данные об известном получателе, например:

```
# postmap -q "bamm@example.com" ldap:/etc/postfix/ldap/local_recipients.cf
bammbamm
```

Если ничего не получается, посмотрите, что будет без поддержки TLS. Если без TLS все работает, заново включите настройки TLS и установите параметр `debuglevel` в файле конфигурации LDAP для Postfix (например, `/etc/postfix/ldap/local_recipients.cf`) как минимум в 1, чтобы получить более подробные записи в журнале.

Тестирование Courier IMAP

Мы только что тестировали базовую конфигурацию Courier IMAP в разделе «Тестирование клиента OpenLDAP». Чтобы убедиться, что все работает, подключаемся к порту IMAP и регистрируемся, как мы это делали в разделе «Тестирование IMAP-сервера». Успешная регистрация должна выглядеть следующим образом:

```
# telnet mail.example.com 143
* OK [CAPABILITY IMAP4rev1 UIDPLUS CHILDREN NAMESPACE \
    THREAD=ORDEREDSUBJECT THREAD=REFERENCES SORT QUOTA \
    IDLE ACL ACL2=UNION STARTTLS]
    Courier-IMAP ready. Copyright 1998-2004 Double Precision, Inc.\
    See COPYING for distribution information.
. login bamm@example.com bamm_secret
. OK LOGIN Ok.
. logout
* BYE Courier-IMAP server shutting down
. OK LOGOUT completed
```

Ограничение для адресов отправителей

После того как пользователи успешно подключатся к серверу Postfix, пройдя SMTP-аутентификацию, они могут пересылать сообщения через ваш сервер, используя любой выбранный ими адрес отправителя. Если вы не настолько доверяете своим пользователям, то можете использовать ограничение `reject_authenticated_sender_login_mismatch`, появившееся в версии Postfix 2.1, которое указывает Postfix на то, что следует требовать использования действительных адресов отправителей. Ограничение работает следующим образом:

1. Пользователь осуществляет подключение, предоставляя SMTP-аутентификации имя пользователя.
2. Когда пользователь пытается отправить сообщение, Postfix извлекает имя отправителя конверта.
3. Postfix ищет отправителя конверта в картах, указанных в параметре `smtpd_sender_login_maps`. В идеале это должна быть карта на основе LDAP.
4. Если поиск возвращает то же имя пользователя, которое было использовано при SMTP-аутентификации, то Postfix принимает сообщение. В противном случае Postfix отклоняет сообщение.

Создание карты LDAP

В нашем каталоге уже есть все данные для карты `smtpd_sender_login_maps`, поэтому разумно использовать их повторно не только для того, чтобы не создавать дополнительных данных, но и чтобы не заниматься дополнительной работой по их сопровождению. Для создания карты создаем в каталоге `/etc/postfix/ldap` новый файл с необходимыми для запроса настройками.

Указывая атрибуты, которые будут извлекаться с сервера LDAP как имена пользователей, вы можете разрешить использовать только почтовые адреса, хранящиеся в атрибуте `mail`, или разрешить отправителям использовать и их псевдонимы (которые хранятся в атрибуте `maildrop`).

Пусть, вы хотите создать конфигурацию, разрешающую использование обоих атрибутов в файле запроса `/etc/postfix/ldap/mail_from_login.cf`. Если включена поддержка TLS, то весь файл должен выглядеть так:

```
version = 3
debuglevel = 0
server_host = ldap://mail.example.com
tls_ca_cert_file = /usr/share/ssl/certs/cacert.pem
tls_cert = /etc/postfix/certs/postfix_public_cert.pem
tls_key = /etc/postfix/certs/postfix_private_key.pem
tls_random_file = /dev/urandom
start_tls = yes
bind = yes
bind_dn = uid=postfix,ou=auth,dc=example,dc=com
bind_pw = Yanggt!
search_base = ou=people,dc=example,dc=com
query_filter = (|(mail=%s)(maildrop=%s))
result_attribute = mail
```

Только два параметра в этом файле должны отличаться от других файлов в каталоге `/etc/postfix/ldap`: `query_filter` и `result_attribute`. Когда файл карты запроса будет готов, устанавливаем параметр `smtpd_sender_login_maps` в файле `main.cf`:

```
smtpd_sender_login_maps = ldap:/etc/postfix/ldap/mail_from_login.cf
```

Настройка smtpd-ограничения

Наконец, разрешаем отправителям использовать только их собственные почтовые адреса, добавляя параметр `reject_authenticated_sender_login_mismatch` в список ограничений в файле `main.cf`:

```
smtpd_recipient_restrictions =
   permit_mynetworks
   reject_authenticated_sender_login_mismatch
   permit_sasl_authenticated
   reject_unauth_destination
```

Примечание

Это ограничение применяется только к аутентифицированным пользователям и ни к кому другому. Поэтому пользователи из ваших доверенных сетей могут

отправлять почту от имени любого отправителя. Кроме того, входящая почта работает для всего остального Интернета.

Тестирование smtpd-ограничения

Начнем тестирование новой конфигурации с проверки того, работает ли карта. Используем команду `postmap` для поиска обычного получателя:

```
# postmap -q "bamm@example.com" ldap:/etc/postfix/ldap/mail_from_login.cf
bammbamm
```

Если разрешено использование псевдонимов в качестве адресов отправителей, то используем `postmap` для запроса псевдонима:

```
# postmap -q "postmaster@example.com" ldap:/etc/postfix/ldap/mail_from_login.cf
bammbamm
```

Если это работает, осталось только опробовать настоящий сеанс SMTP. Подготавливаем строку аутентификации в кодировке base64:

```
# perl -MMIME::Base64 -e \
'print encode_base64("bamm\@example.com\0bamm\  @example.com\0bamm_secret");'
YmFtbUBleGFtcGxlLmNvbQBiYW1tQGV4YW1wbGUuY29tAGJhbW1fc2VjcmV0
```

Затем подключаемся к серверу Postfix с удаленного хоста и аутентифицируемся при помощи этой строки:

```
# telnet mail.example.com 25
220 mail.example.com ESMTP Postfix
EHLO client.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-AUTH PLAIN LOGIN CRAM-MD5 DIGEST-MD5
250-AUTH=PLAIN LOGIN CRAM-MD5 DIGEST-MD5
250 8BITMIME
AUTH PLAIN YmFtbUBleGFtcGxlLmNvbQBiYW1tQGV4YW1wbGUuY29tAGJhbW1fc2VjcmV0
235 Authentication successful
```

Теперь пытаемся отправить сообщение от имени другого пользователя (аутентифицировались мы как Bamm Bamm):

```
MAIL FROM: <rubble@example.com>
250 Ok
RCPT TO: <wietse@porcupine.org>
553 <rubble@example.com>: Sender address rejected: not owned by user
bamm@example.com
```

Postfix выявляет несоответствие на этапе `RCPT TO`, т. к. именно в этот момент начинает действовать параметр `smtpd_recipient_restrictions`.

Вы также можете попробовать отправить сообщение от имени реального пользователя. Если получится, то проверять больше нечего. Мы закончили. Радуйтесь!

20

Работа Postfix в окружении chroot

Функция chroot создает дополнительный барьер против вторжения; этот барьер имеет смысл только тогда, когда хост уже работает в защищенной конфигурации.

– Витсе Венема

Запуск Postfix в окружении chroot изолирует сервер от основного дерева каталогов операционной системы. Это делается с целью защитить систему от злоумышленника, сумевшего взломать Postfix. Использование этой возможности предполагает наличие в каталоге chroot лишь минимального набора файлов, приложений и других ресурсов.

Для создания окружения chroot (так называемой «песочницы») не нужны дополнительные программы. В дистрибутиве имеются вспомогательные сценарии, выполняющие эту работу. В данной главе наряду с изложением теории вопроса приведен практический пример совместной работы Postfix и chroot с использованием SASL/TLS.

Поставляемые с Postfix chroot-сценарии находятся в каталоге `examples/chroot-setup` в разделе исходных текстов. Если вы используете двоичные пакеты Postfix, то, вероятно, их создатели поместили туда ряд сценариев, выполняющих синхронизацию содержимого каталога chroot с внешней файловой системой.

Надежный способ создания chroot-окружения состоит из следующих этапов:

1. Создание работоспособной конфигурации без использования chroot.
2. Поочередный перенос демонов Postfix в chroot-окружение.

Следуя этим правилам, вы легко сможете идентифицировать демон, испортившийся при переносе в chroot, и соответствующим образом откорректировать содержимое каталога.

Как работает окружение chroot?

Представьте себе грабителя, влезающего через разбитое окно к вам в дом и попадающего в хорошо защищенную комнату. В ней не только нет ничего ценного, но и имеющиеся в ней предметы никак не помогут проникнуть дальше в дом. Злоумышленнику не остается ничего другого, кроме как ретироваться.

Этот способ сработает, если архитектор предусмотрит, чтобы в защищенной комнате находился только самый необходимый минимум вещей. Более того, все предметы должны быть хорошо закреплены, чтобы их нельзя было использовать для взлома этой комнаты.

Окружение chroot в среде UNIX сильно напоминает такую защищенную комнату.

Основные принципы настройки chroot

Перечислим основные принципы настройки chroot:

Запускайте приложения с наименьшими возможными привилегиями

Чем больше у пользователя прав, тем больший вред он может нанести системе. В частности, пользователь `root` и выполняющиеся от его имени программы способны с легкостью преодолеть барьер chroot-окружения. Поэтому выполняющиеся в таком окружении программы не должны иметь привилегий суперпользователя; их привилегии должны быть строго ограничены требованиями выполняемой ими работы.

Вовремя отменяйте привилегии

Приложения, обслуживающие порты с низкими номерами, например порт 25, могут потребовать запуска с привилегиями `root` для получения доступа к портам. Однако после того как приложение получило то, что ему требуется, оно должно корректно аннулировать избыточные права доступа.

«Песочница» должна быть маленькой и пустой

Держите в chroot-окружении только необходимый минимум файлов. Чем меньше их там, тем меньше у взломщика шансов навредить вам.

Сделайте владельцем файлов пользователя root и только ему выдайте права на запись

Приложения, выполняющиеся в chroot-окружении, не должны иметь возможности изменять находящиеся там файлы. Сделайте владельцем файлов суперпользователя и только ему дайте разрешение на запись.

Используйте внешние ссылки на файлы конфигурации

Символические ссылки, ведущие за пределы chroot-окружения, не будут работать для системы, выполняющейся внутри «песочницы». В не-

которых системах файлы конфигурации используются совместно демонами из chroot-окружения и работающими в пользовательском режиме программами. При этом все демоны должны иметь доступ к файлам конфигурации независимо от того, находятся ли они внутри или снаружи chroot-окружения. Вместо того чтобы пересобирать эти программы с указанием нового пути (например, `/chroot/named/etc/named.conf`), лучше создать символическую ссылку, ведущую *извне внутрь* окружения, как в этом примере:

```
# ln -s /chroot/named/etc/named.conf /etc/named.conf
```

При таком способе большинство программ работает нормально, но вам следует проявлять осторожность, редактируя такие файлы, как `/etc/named.conf`, – изменения в них влияют на систему внутри окружения.

Техническая реализация

Демоны выполняют вызов chroot и отказ от привилегий самостоятельно. Это позволяет им обратиться к файлам `/etc/postfix` (и к картам) до того, как они будут изолированы в chroot-окружении.

Системный вызов chroot() влияет на то, как процесс – после входа в chroot-окружение – воспринимает файловую систему. Вот как это работает:

1. Демон `master` осуществляет вызов `chdir(queue_directory)`.
2. Демон `master` запускает остальные демоны Postfix, сообщая каждому из них, должен ли он сменить файловую систему и отказаться от привилегий.

За исключением очень редких случаев, выйти из такого окружения невозможно.

Как chroot влияет на Postfix?

Запуск сервера Postfix в chroot-окружении влияет на то, как он воспринимает файловую систему. Предположим, что Postfix запущен в `/var/spool/postfix`. Несмотря на то что другие приложения могут видеть и `/var/spool/postfix` (каталог очередей), и все остальное в системе, демоны Postfix считают точку `/var/spool/postfix` корнем `/`, т. к. были запущены через chroot. У них нет доступа за пределы `/var/spool/postfix`.

В связи с этим вам может понадобиться скопировать ряд файлов, находящихся за пределами досягаемости демонов, в «песочницу» Postfix.

Исполняемые файлы (демоны)

Вам не надо копировать демоны в chroot-окружение, так как они запускаются демоном `master`, который запущен без chroot.

Библиотеки, необходимые исполняемым файлам

Программы загружают библиотеки до того, как перейдут в chroot-окружение, поэтому их копировать не требуется.

Карты

Программы открывают статические библиотеки до перехода в chroot. Однако вам, возможно, придется создать в chroot-окружении сокет для карт, управляемых базой данных (см. замечание о сокетах в этом списке).

Файлы конфигурации

Демоны читают файлы конфигурации до перехода в chroot. Копировать их не требуется.

Сокеты

Файлы сокетов, таких как mysql или SASL, должны быть доступны демонам внутри chroot-окружения. Имейте в виду, что клиентская библиотека MySQL может искать сокеты в таком месте, как `/var/run/mysql.socket`, поэтому вам надо будет поместить ее, к примеру, в *queue_directory*`/var/run/mysql.socket`.

Файлы, необходимые библиотекам

Библиотекам C для корректной работы нужен доступ к таким файлам, как `/etc/resolv.conf` и `/etc/localtime`. Эти файлы надо скопировать в `/var/spool/postfix`.

Вспомогательные сценарии для chroot

С Postfix поставляются несколько дополнительных сценариев, расположенных в `examples/chroot-setup` в каталоге исходных текстов. Эти сценарии помогут настроить chroot-окружение в вашей конкретной системе.

Когда вы запускаете Postfix через chroot, он проверяет наличие в этом окружении ряда необходимых файлов, в частности `/var/spool/postfix/etc/resolv.conf`, и их актуальность. Если файлы внутри и снаружи chroot-окружения отличаются, Postfix запишет в журнал предупреждение. Очень важно отслеживать эти сообщения и устранять несоответствия.

Демоны в chroot-окружении

Поскольку демон `master` запускает остальные демоны Postfix, именно он сообщает этим демонам, должны ли они стартовать через chroot или нет. Использование chroot регулируется файлом конфигурации `/etc/postfix/master.cf`.

Активация chroot

Чтобы демон был запущен через chroot, вам надо указать в файле `master.cf` вызывающую его службу. Обратитесь к столбцу chroot в файле `master.cf`, чтобы узнать текущее состояние демона. В исходных текстах Postfix для каждого демона соответствующий параметр установлен в значение `n`, хотя в некоторых дистрибутивах Postfix что-то мо-

жет быть изменено. То есть сразу после стандартной установки Postfix ни один из демонов не запускается через chroot. Чтобы изменить это положение, поменяйте значение параметра chroot с n на - или y.

Например, если вы захотите, чтобы демон smtpd запускался через chroot, то ваш конфигурационный файл мог бы быть таким:

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
# ==========================================================================
smtp      inet  n       -       y       -       -       smtpd
#smtps    inet  n       -       n       -       -       smtpd
#  -o smtpd_tls_wrappermode=yes -o smtpd_sasl_auth_enable=yes
...
pickup    fifo  n       -       n       60      1       pickup
cleanup   unix  n       -       n       -       0       cleanup
...
```

Обратите внимание, что для службы smtp значение в столбце chroot изменено на y. После перезагрузки конфигурации Postfix демон master будет запускать эту службу через chroot.

Ограничения chroot и передача функций

В документации master.cf говорится, что из chroot могут быть запущены почти все демоны, *кроме* pipe, virtual и local:

pipe

Демон pipe запускает внешние программы, обычно расположенные вне каталога очередей Postfix, поэтому ему, как правило, необходимы файлы, находящиеся вне каталога очередей.

local

Демон local занимается локальной доставкой и должен иметь доступ в домашние каталоги пользователей. Запуск local из chroot не имел бы смысла, так как это означало бы, что пользовательские каталоги находятся в /var/spool/postfix, внутри окружения, куда могут получить доступ злоумышленники.

virtual

Демон virtual, как и local, обеспечивает локальную доставку, так что аргументы против его запуска из chroot-окружения повторяют аргументы, приведенные для демона local.

Если другие демоны работают в chroot-окружении, то для получения доступа к данным конфигурации им может потребоваться использование демона proxymap. Например, если работающему в chroot-окружении SMTP-серверу для отклонения сообщений несуществующим локальным адресатам нужен доступ к системному файлу passwd, то на такой случай копия файла passwd в chroot-окружении не поддерживается, т. к. это подрывало бы саму идею запуска Postfix в «песочнице».

Для того чтобы не помещать в chroot-окружение карты, содержащие секретные данные, можно поручить просмотр и поиск демону `proxymap` (который работает не в chroot-окружении), используя следующий параметр конфигурации:

```
local_recipient_maps = proxy:unix:passwd.byname $alias_maps
```

Имейте в виду, что поскольку демон `proxymap` выходит за пределы chroot-окружения, он не может обеспечивать доступ к картам, участвующим в обеспечении безопасности.

Библиотеки, конфигурационные файлы и другие файлы chroot

Для корректной работы многим программам необходимы внешние файлы, а именно:

- Библиотеки и другие совместно используемые объекты
- Файлы конфигурации
- Другие файлы, такие как сокеты или устройства

Существует несколько способов определения того, какие файлы необходимы программе. Для выявления зависимости от совместно используемых библиотек в большинстве UNIX-систем можно использовать команду `ldd` или `chatr`. Однако при стандартной установке Postfix никаких проблем возникнуть не должно, так как все демоны запускаются и загружают свои библиотеки прежде, чем перейти в chroot-окружение.

Для того чтобы узнать, какие файлы конфигурации вам необходимы, используйте программу `strace` в Linux, `truss` – в Solaris или `ktrace` – в других вариациях UNIX. Покажем, как можно использовать `strace` для запуска программы *program*:

```
# strace -o outputfile program
```

Ищем в выходном файле вызовы `open()`:

```
# grep open outputfile  | grep ENOENT
open("/etc/ld.so.preload", O_RDONLY)    = -1 ENOENT (No such file
                                                     or directory)
open("/usr/share/locale/C/libdst.cat", O_RDONLY) = -1 ENOENT (No such file
                                                     or directory)
open("/usr/share/locale/C/LC_MESSAGES/libdst.cat", O_RDONLY) = -1 ENOENT
                                          (No such file or directory)
open("/usr/share/locale/C/libdst.cat", O_RDONLY) = -1 ENOENT (No such file
                                                     or directory)
open("/usr/share/locale/C/LC_MESSAGES/libdst.cat", O_RDONLY) = -1 ENOENT
                                          (No such file or directory)
open("/usr/share/locale/C/libisc.cat", O_RDONLY) = -1 ENOENT (No such file
                                                     or directory)
open("/usr/share/locale/C/LC_MESSAGES/libisc.cat", O_RDONLY) = -1 ENOENT
                                          (No such file or directory)
```

```
open("/usr/share/locale/C/libisc.cat", O_RDONLY) = -1 ENOENT (No such file
                                                    or directory)
open("/usr/share/locale/C/LC_MESSAGES/libisc.cat", O_RDONLY) = -1 ENOENT
                                                (No such file or directory)
open("/usr/share/locale/C/libdns.cat", O_RDONLY) = -1 ENOENT (No such file
                                                    or directory)
open("/usr/share/locale/C/LC_MESSAGES/libdns.cat", O_RDONLY) = -1 ENOENT
                                                (No such file or directory)
open("/usr/share/locale/C/libdns.cat", O_RDONLY) = -1 ENOENT (No such file
                                                    or directory)
open("/usr/share/locale/C/LC_MESSAGES/libdns.cat", O_RDONLY) = -1 ENOENT
                                                (No such file or directory)
```

Примечание

Помните, что программа может пытаться открывать несколько версий файла конфигурации, прежде чем найдет подходящую.

Есть и другой способ: можно подключить `strace` к работающему процессу, чтобы посмотреть, что он пытается делать. Например, если вы хотите посмотреть, что делает демон `master`, поступите следующим образом:

```
# ps auxwww|grep master
root  9004  0.0  0.3  3452  940 ?  S    07:49   0:00 /usr/lib/postfix/master
# strace -p 9004
Process 9004 attached - interrupt to quit
select(76, [10 11 12 15 17 18 21 23 24 26 27 29 30 32 33 35 36 38 39 41 42 44
45 47 48 50 51 53 54 56 57 59 60 62 63 65 66 68 69 71 72 74 75], [], [10 11
12 15 17 18 21 23 24 26 27 29 30 32 33 35 36 38 39 41 42 44 45 47 48 50 51 53
54 56 57 59 60 62 63 65 66 68 69 71 72 74 75], {22, 790000} <unfinished ...>
Process 9004 detached
```

Преодоление ограничений chroot

Сценарии из дистрибутива в исходных текстах, помогающие создать chroot-окружение Postfix (в `examples/chroot-scripts`), могут рассказать о том, какие системные файлы следует скопировать в это chroot-окружение (например, файлы часовых поясов, необходимые для библиотеки C). Перечислим файлы, которые вам понадобятся.

DNS

Для корректного разрешения имен библиотеке C необходимы некоторые файлы; например, в Linux-системе вам потребуются `/etc/resolv.conf`, `/etc/nsswitch.conf` и `/etc/hosts`. Они должны находиться внутри «песочницы», и вспомогательные сценарии могут скопировать их туда.

Настройки времени

Если вы обнаружите, что выводимая демоном Postfix журнальная информация сдвинута на несколько часов, необходимо скопировать в chroot-окружение информацию о часовом поясе (`/etc/localtime`).

Сокеты

Вы без труда сможете настроить Postfix и saslauthd для работы внутри chroot-окружения Postfix. Нужно лишь задать для Postfix и saslauthd разные пути к сокету saslauthd.

Сначала создаем в chroot-окружении Postfix (обычно это `/var/spool/postfix`) все необходимые подкаталоги `run_path`:

```
# mkdir /var/spool/postfix/var
# mkdir /var/spool/postfix/var/run
# mkdir /var/spool/postfix/var/run/saslauthd
# chmod 770 /var/spool/postfix/var/run/saslauthd
# chgrp postfix /var/spool/postfix/var/run/saslauthd
```

Теперь добавляем в файл `/usr/lib/sasl/smtpd.conf` параметр `saslauthd_path`, который укажет запускаемому из chroot демону `smtpd`, где следует искать сокет. Отсекаем ту часть пути, которая ведет в «песочницу» (то есть `/var/spool/postfix`), и указываем путь `run_path`, включая в него значение – имя сокета (`mux`):

```
saslauthd_path: /var/run/saslauthd/mux
```

Наконец, запускаем saslauthd с параметром `-m`, который определяет место создания сокета. Указываем полный путь, каким он будет при взгляде извне «песочницы»:

```
# /usr/sbin/saslauthd -m /var/spool/postfix/var/run/saslauthd -a shadow
```

Таким образом, оба приложения используют один и тот же сокет для взаимодействия даже в том случае, если Postfix работает в chroot-окружении.

IV

Настройка Postfix

В этой части книги приведены советы по улучшению производительности вашего сервера. Начав с обсуждения банальных вопросов, таких как кэширование DNS и превращение в открытый ретранслятор, мы перейдем к более сложным задачам и рассмотрим предотвращение возврата недоставимой почты и создание выделенного транспорта. Далее мы обсудим использование черных списков для ограничения потока входящей почты, а также весьма интересные экспериментальные возможности версии Postfix 2.1.

Параллелизм удаленных клиентов и ограничение частоты запросов

В главе 21 рассматривается новая возможность ограничения количества клиентских соединений. Это контрмера, вводимая для защиты Postfix от SMTP-клиентов, которые засыпают демон `smtpd` слишком большим числом одновременных соединений.

Настройка производительности

Postfix работает быстро, но в некоторых случаях его можно сделать еще быстрее. Если ваш сервер Postfix работает не так быстро, как хотелось бы, то обязательно прочтите главу 22.

21

Параллелизм удаленных клиентов и ограничение частоты запросов

Версии Postfix 2.1 и 2.2 реализуют параллелизм удаленных клиентов и ограничение количества одновременных запросов. Ограничение количества соединений используется для защиты Postfix от SMTP-клиентов, которые заваливают демон `smtpd` слишком большим количеством одновременных соединений. В этой главе будет рассмотрено несколько ситуаций, в которых полезно ограничение скорости, и показано, как его настроить.

Причины ограничения количества соединений

Даже хорошо настроенная версия Postfix может обработать лишь конечный объем почтового трафика в единицу времени. Пропускная способность сервера зависит от производительности дискового ввода-вывода, производительности процессора и скорости работы антивирусных программ, подключенных к Postfix. В версиях Postfix, предшествующих 2.1, один клиент мог использовать все доступные серверные процессы `smtpd`, блокируя работу всех прочих клиентов, пытающихся доставить почту.

Ограниченные возможности оборудования, захватывающие соединения клиенты и сложные конфигурации – вот что вызывает необходимость ограничения объема входящей почты. Кроме того, в некоторых случаях ограничение количества соединений предотвращает задержку доставки сообщений или каким-то другим способом уменьшает вредное воздействие на Postfix:

Нашествия вирусов и червей

Новые вирусы и черви, распространяющиеся по сети, обычно стремятся к скорейшему размножению. Ограничение количества соединений замедлит распространение вредоносного программного обеспечения.

Почтовые бомбы

Почтовые бомбы – это большой непрерывный поток сообщений, обычно отправляемых какой-то одной системой на ваш сервер. Обычно это происходит по злому умыслу, но может получиться и случайно (например, мы имели дело с антивирусным продуктом, который отправлял одно сообщение для каждого зараженного файла, и когда речь шла о полностью зараженной системе, почтовый сервер мог получать более 100 сообщений в минуту).

Неуправляемые клиенты

Термин «неуправляемые клиенты» (runaway clients) является обобщением для вирусов, червей и почтовых бомб и обозначает любой неконтролируемый клиент, отправляющий в вашу систему непрекращающийся поток сообщений. Такой клиент не обязательно является злоумышленником, проблема может быть вызвана ошибкой программирования или настройки.

Спам от открытых прокси-серверов

Открытые прокси-серверы часто используются спамерами для того, чтобы скрыть истинного отправителя сообщения. При вводе ограничения на сообщения от открытых прокси-серверов Postfix отклоняет входящие сообщения от них с кодом временной ошибки. Такие серверы-посредники не имеют собственного механизма очередей, поэтому не пытаются повторно доставить сообщение, так что большие наплавы спама от открытых прокси-серверов не будут угрожать вашей системе.

Для введения ограничений необходимо собрать статистические данные и изменить значения некоторых параметров, определяющих, сколько последовательных и параллельных соединений клиенты могут осуществить с демоном `smtpd`. В последующих разделах будет показано, как это сделать.

Сбор статистики соединений

Прежде чем приступать к ограничению клиентских соединений, вам необходимо знать, какие клиенты подключаются к серверу и сколько успешных и одновременных соединений они инициируют в ходе обычной работы. В этом нам поможет демон `anvil`, который собирает статистику соединений и записывает максимальное количество соединений и их частоту.

Примечание

Сбор статистики клиентских соединений полезен не только для реализации ограничения их количества. Например, программа просмотра журнала может считывать эту информацию и изменять правила межсетевого экрана, блокируя выявленные неуправляемые клиенты. Ограничение частоты запросов появилось в Postfix сравнительно недавно, поэтому на момент написания этой книги никаких общеизвестных программ для его реализации не было.

Запуск демона anvil

Как и другие демоны Postfix, `anvil` управляется демоном `master`. По умолчанию демон `anvil` включен, но все же стоит проверить, не закомментирована ли строка конфигурации этого демона в файле `master.cf`:

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
# ==========================================================================
anvil     unix  -       -       n       -       1       anvil
```

Демон `anvil` запускается по требованию и записывает все собранные им данные в почтовый журнал через определенные промежутки времени. Его сообщения выглядят следующим образом:

```
Dec 20 01:19:16 mail postfix/anvil[8991]: statistics: max connection count 4
  for (10.0.0.1:smtp:216.129.165.190) at Dec 20 01:18:35
Dec 20 01:29:16 mail postfix/anvil[8991]: statistics: max connection rate 9/
  60s for (10.0.0.1:smtp:62.243.72.19) at Dec 20 01:22:11
Dec 20 01:29:16 mail postfix/anvil[8991]: statistics: max connection count 2
  for (10.0.0.1:smtp:62.243.72.19) at Dec 20 01:22:09
Dec 20 01:39:16 mail postfix/anvil[8911]: statistics: max connection rate 3/
  60s for (10.0.0.1:smtp:146.82.138.6) at Dec 20 01:37:04
Dec 20 01:49:16 mail postfix/anvil[8991]: statistics: max connection rate 2/
  60s for (10.0.0.1:smtp:218.18.32.248) at Dec 20 01:46:35
Dec 20 01:49:16 mail postfix/anvil[8991]: statistics: max connection count 2
  for (10.0.0.1:smtp:218.18.32.248) at Dec 20 01:46:35
Dec 20 01:59:16 mail postfix/anvil[8991]: statistics: max connection rate 3/
  60s for (10.0.0.1:smtp:146.82.138.6) at Dec 20 01:50:58
Dec 20 01:59:16 mail postfix/anvil[8991]: statistics: max connection count 2
  for (10.0.0.1:smtp:171.67.16.117) at Dec 20 01:55:33
Dec 20 02:09:16 mail postfix/anvil[8991]: statistics: max connection rate 2/
  60s for (10.0.0.1:smtp:216.136.204.119) at Dec 20 02:03:38
Dec 20 02:19:16 mail postfix/anvil[8991]: statistics: max connection rate 2/
  60s for (10.0.0.1:smtp:63.161.42.51) at Dec 20 02:09:29
Dec 20 02:19:16 mail postfix/anvil[8991]: statistics: max connection count 2
  for (10.0.0.1:smtp:130.149.17.13) at Dec 20 02:11:52
```

Изменение интервала журналирования anvil

По умолчанию демон `anvil` записывает в почтовый журнал статистические данные каждые десять минут или при завершении работы (напри-

мер, если вы перезагружаете конфигурацию Postfix или демон сам завершает работу по окончании периода `max_idle_seconds`). Если вы хотите увеличить или уменьшить этот интервал, установите соответствующее значение параметра `client_connection_status_update_time` в файле `main.cf`:

```
client_connection_status_update_time = 10m
```

Изменения вступят в силу сразу же после перезагрузки конфигурации Postfix.

Примечание

Интервал журналирования назначается исходя из соображений удобства диагностики и не зависит от внутренней работы демона `anvil`. Уменьшение интервала не приводит к более частому сбору статистик демоном `anvil`. Программы Postfix обращаются ко всем текущим данным `anvil` через взаимодействие процессов, а не просматривают журнальные файлы.

Ограничение частоты клиентских соединений

По умолчанию Postfix не накладывает ограничений на количество последовательных соединений одного клиента. Так что, если ничего не изменить, клиент может подключаться и отключаться так часто, как ему заблагорассудится, при этом `smtpd` будет зря растрачивать ценные ресурсы на просмотр DNS, приветствия TLS и т. д.

Для введения ограничения на частоту клиентских соединений Postfix должен подсчитать количество соединений за определенный промежуток времени. Этот промежуток называется единицей времени учета (rate time unit), и вы можете задать его при помощи параметра `client_connection_rate_time_unit`. Значение по умолчанию равно одной минуте:

```
client_connection_rate_time_unit = 60s
```

Теперь можно ввести ограничение на число разрешенных соединений от одного клиента в единицу времени учета, установив параметр `smtpd_client_connection_rate_limit`. Например, следующая настройка в сочетании с используемым значением по умолчанию для единицы времени учета позволяет одному клиенту осуществлять не более 30 подключений в течение 60 секунд:

```
smtpd_client_connection_rate_limit = 30
```

Тестирование ограничений на количество клиентских соединений

Проще всего проверить ограничения, выполнив все настройки и несколько дней наблюдая за записями в файлах журнала. Некорректная настройка не может привести к потере почты или испортить вашу систему, т. к. Postfix отвергает соединения от клиентов, на которых нало-

жены ограничения, с кодами временных ошибок. Правильно настроенный клиент повторит попытку доставки, например, Postfix по умолчанию пытается доставить сообщение в течение пяти дней (см. параметр `maximal_queue_lifetime`).

Если вы хотите провести тестирование незамедлительно, выполните следующие действия:

1. Сгенерируйте большой объем почтового трафика, используя такую программу, как `smtp-source`.
2. Отправьте почту с IP-адреса, который не освобожден от ограничений на количество соединений.

Примечание

На время тестирования вы можете уменьшить значение параметра `client_connection_status_update_time` до одной минуты, для того чтобы чаще собирать статистику.

Для осуществления проверки сделайте следующие настройки в файле `main.cf` и перезагрузите конфигурацию сервера:

```
smtpd_client_event_limit_exceptions= ❶
client_connection_rate_time_unit = 60s
smtpd_client_connection_rate_limit = 1 ❷
client_connection_status_update_time = 1m ❸
```

❶ Установка параметра в пустое значение означает, что все клиенты подлежат применению ограничений на количество соединений. Используйте такое значение только для тестирования.

❷ Это значение также сделано таким маленьким исключительно в целях тестирования. На самом деле вы ведь не хотите, чтобы ваш почтовый сервер разрешал лишь одно соединение от клиента в минуту.

❸ Статистика будет записываться в журнал каждую минуту, так что вам не придется ждать отчета целых десять минут. Но и этот параметр не следует оставлять таким после завершения тестирования.

Теперь нам нужно сгенерировать объем трафика, достаточный для того, чтобы активировать ограничения. Используем на сервере Postfix команду `smtp-source`:

```
$ smtp-source -m 10 -f sender@example.com -t recipient@example.com
127.0.0.1:25
```

Примечание

Если программа `smtp-source` не включена в ваш дистрибутив, то возьмите ее из файлов исходных текстов Postfix.

Такая команда отправляет десять тестовых сообщений от `sender@example.com` получателю `recipient@example.com` через SMTP-сервер `127.0.0.1` (`localhost`). Как вы помните, ограничение для клиента было установле-

но равным одному соединению в минуту, так что генерируемый командой трафик выходит за установленные границы.

Результат будет виден в выходных данных `smtp-source`:

```
smtp-source: fatal: bad startup: 450 Too many connections from 127.0.0.1
```

Кроме того, в журнале вы увидите следующее:

```
Jan  9 09:04:16 mail postfix/smtpd[26530]: connect from localhost[127.0.0.1]
Jan  9 09:04:16 mail postfix/smtpd[26530]: 12AA515C06F:
  client=localhost[127.0.0.1]
Jan  9 09:04:16 mail postfix/smtpd[26530]: disconnect from
  localhost[127.0.0.1]
Jan  9 09:04:16 mail postfix/smtpd[26530]: connect from localhost[127.0.0.1]
Jan  9 09:04:17 mail postfix/smtpd[26530]: warning: Too frequent connections:
  2 from 127.0.0.1 for service localhost:smtp
Jan  9 09:04:17 mail postfix/smtpd[26530]: disconnect from
  localhost[127.0.0.1]
```

Рассмотрим еще один пример, в котором параметру `smtpd_client_connection_rate_limit` присвоено значение 30. Postfix отклоняет соединения всех клиентов, превысивших данную максимально разрешенную частоту отправки сообщений, используя код ошибки 450, и формирует предупреждение, содержащее имя и адрес клиента и имя демона:

```
Dec 20 02:39:03 mail postfix/smtpd[18431]: warning: Too frequent connections:
  31 from 81.199.6.44 for service 10.0.0.1:smtp ❶
Dec 20 02:39:04 mail postfix/smtpd[17840]: warning: Too frequent connections:
  32 from 81.199.6.44 for service 10.0.0.1:smtp
Dec 20 02:39:04 mail postfix/smtpd[17878]: warning: Too frequent connections:
  33 from 81.199.6.44 for service 10.0.0.1:smtp
...
Dec 20 02:39:15 mail postfix/smtpd[18440]: warning: Too frequent connections:
  65 from 81.199.6.44 for service 10.0.0.1:smtp
Dec 20 02:39:15 mail postfix/smtpd[18432]: warning: Too frequent connections:
  66 from 81.199.6.44 for service 10.0.0.1:smtp
Dec 20 02:39:16 mail postfix/anvil[8991]: statistics: max connection rate 72/
  60s for (10.0.0.1:smtp:81.199.6.44) at Dec 20 02:39:15 ❷
```

❶ Клиент, работающий с адреса `81.199.6.44`, превысил допустимый предел в 30 соединений в минуту, что привело к появлению в журнале предупреждения `Too frequent connections`.

❷ Данный клиент (`81.199.6.44`) установил рекорд по количеству соединений (72 за 60 секунд), о чем нам сообщает демон `anvil`.

Ограничение параллельных клиентских соединений

По умолчанию количество параллельных соединений для клиента может быть не больше половины разрешенного по умолчанию количест-

ва процессов. Так что два клиента могут занять все smtpd-процессы, которые может запустить Postfix. Количество разрешенных параллельных соединений для одного клиента регулируется параметром smtpd_client_connection_count_limit. Например, такая настройка в файле main.cf разрешает 25 параллельных соединений для одного клиента:

```
smtpd_client_connection_count_limit = 25
```

Предупреждение

Разрешенное количество процессов smtpd (или значение параметра default_process_limit) должно быть значительно больше, чем значение smtpd_client_connection_count_limit; в противном случае один клиент мог бы занять все доступные процессы smtpd.

Тестирование ограничений на параллельные клиентские соединения

Как и в случае с ограничением частоты соединений, проще всего проверить ограничения для параллельных соединений, применив все настройки и понаблюдав несколько дней за журнальными записями. Однако если вы предпочитаете незамедлительную проверку, сгенерируйте при помощи команды smtp-source множество параллельных соединений с IP-адреса, который не освобожден от ограничений на количество соединений.

На время тестирования, вероятно, стоит уменьшить значение параметра client_connection_status_update_time до одной минуты.

Для проведения проверки сделайте следующие установки в файле master.cf и перезагрузите конфигурацию:

```
smtpd_client_connection_limit_exceptions = ❶
client_connection_rate_time_unit = 60s
smtpd_client_connection_count_limit = 1 ❷
client_connection_status_update_time = 1m ❸
```

❶ Все клиенты подлежат применению ограничений на количество соединений. Используйте такое значение только для тестирования.

❷ Это значение также сделано таким маленьким исключительно в целях тестирования. На самом деле мы не хотим, чтобы почтовый сервер разрешал лишь одно соединение от клиента в минуту.

❸ Статистика будет записываться в журнал каждую минуту, так что вам не придется ждать отчета целых десять минут. Используйте это значение только для тестирования.

Вы можете открыть несколько параллельных соединений, запустив на своем Postfix-сервере команду smtp-source:

```
$ smtp-source -s 10 -m 10 -f sender@example.com -t recipient@example.com
127.0.0.1:25
```

Параметр `-m 10` говорит о том, что следует отправить десять тестовых сообщений, а параметр `-s 10` устанавливает десять параллельных сеансов SMTP. Учитывая то, что у нас установлено ограничение в одно соединение для каждого клиента, `smtp-source` выдаст нам такое сообщение об ошибке:

```
smtp-source: fatal: bad startup: 450 Too many connections from 127.0.0.1
```

Журнал должен выглядеть так:

```
Jan  9 09:14:15 mail postfix/smtpd[28438]: warning: Too many connections:
  2 from 127.0.0.1 for service localhost:smtp
Jan  9 09:14:15 mail postfix/smtpd[28438]: disconnect from
localhost[127.0.0.1]
Jan  9 09:14:15 mail postfix/smtpd[28437]: warning: Too many connections:
  2 from 127.0.0.1 for service localhost:smtp
Jan  9 09:14:15 mail postfix/smtpd[28437]: disconnect from
localhost[127.0.0.1]
Jan  9 09:14:15 mail postfix/smtpd[28439]: warning: Too many connections:
  3 from 127.0.0.1 for service localhost:smtp
Jan  9 09:14:15 mail postfix/smtpd[28439]: disconnect from
localhost[127.0.0.1]
Jan  9 09:14:15 mail postfix/smtpd[28440]: warning: Too many connections:
  4 from 127.0.0.1 for service localhost:smtp
Jan  9 09:14:15 mail postfix/smtpd[28440]: disconnect from
localhost[127.0.0.1]
```

Далее приведены записи журнала, сделанные при установке параметра `smtpd_client_connection_count_limit` в значение 25. Как и в случае проверки ограничений для частоты соединений, Postfix отправляет клиенту, открывающему слишком много параллельных соединений, код ошибки 450, отключается и записывает в журнал сообщение, содержащее имя и адрес клиента:

```
Dec  3 09:12:53 mail postfix/smtpd[19883]: warning: Too many connections:
  26 from 213.165.64.165 for service 10.0.0.1:smtp ❶
Dec  3 09:12:53 mail postfix/smtpd[19884]: warning: Too many connections:
  27 from 213.165.64.165 for service 10.0.0.1:smtp
...
Dec  3 09:13:15 mail postfix/smtpd[19894]: warning: Too many connections:
  35 from 213.165.64.165 for service 10.0.0.1:smtp
...
Dec  3 09:16:47 mail postfix/anvil[7958]: statistics: max connection count
  37 for (10.0.0.1:smtp:213.165.64.165) at Dec  3 09:12:3 ❷
```

❶ Клиент `213.165.64.165` превысил установленный предел в 25 соединений, поэтому в журнале появилось предупреждение `Too many connections`.

❷ Клиент `213.165.64.165` установил рекорд, открыв 37 соединений с `smtpd`.

Освобождение клиентов от ограничений

Вы можете использовать параметр smtpd_client_connection_limit_exceptions, для того чтобы освободить авторизованные хосты и сети от клиентских ограничений, рассмотренных в данной главе. Его нотация позволяет указывать выражения для сетей и масок, имена хостов и имена доменов.

По умолчанию Postfix предоставляет освобождение от клиентских ограничений всем хостам, указанным в параметре mynetworks. Если вы хотите использовать более строгие ограничения, обратитесь к sample-smtpd.cf, smtpd(8) и anvil(8).

Рассмотрим пример, в котором всем хостам из $mynetworks, подсети 10.45.207.0/24 и домену example.com разрешено осуществлять столько соединений, сколько им угодно:

```
smtpd_client_connection_limit_exceptions =
  $mynetworks,
  10.45.207.0/24,
  .example.com
```

22

Настройка производительности

Postfix быстро работает и в «коробочной» конфигурации, но, как и другие пакеты, его обычно удается настроить так, чтобы он работал еще быстрее. Кроме того, возможны ситуации, в которых сервер Postfix будет работать не так хорошо, как вам хотелось бы, что может быть вызвано ограничениями аппаратных и программных средств сервера или же другими неблагоприятными условиями: большим наплывом почты или недоставимыми сообщениями.

В этой главе будет показано, как обнаружить и проанализировать наиболее часто встречающиеся проблемы производительности.

Простые усовершенствования

Начнем с рассмотрения нескольких элементарных, но не совсем очевидных приемов, которые помогут решить простые проблемы или вообще избежать их. Всегда помните о том, что многие проблемы производительности вызваны некорректной настройкой, например неправильно составленным файлом `/etc/resolv.conf`. Последующие разделы расположены не в порядке убывания значимости, все рассматриваемые в них вопросы одинаково важны.

Ускорение DNS-поиска

Postfix выполняет множество запросов DNS: для SMTP необходимы MX- и A-записи. Кроме того, многие ограничения Postfix используют DNS для проверки имени хоста клиента или просмотра черного списка. Поэтому чрезвычайно важно, чтобы ваш сервер мог быстро просматривать записи DNS, особенно если вы обрабатываете большой объем трафика.

Тестирование DNS-запросов

Основной проблемой разрешения имен DNS является слишком долгое исполнение запросов. Вы можете использовать команду `dig` для просмотра DNS и вывода подробной информации о выполнении запроса:

```
$ dig www.example.com
; <<>> DiG 9.2.3rc4 <<>> www.example.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48136
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.example.com.               IN      A
;; ANSWER SECTION:
www.example.com.        172800  IN      A       192.0.34.166
;; Query time: 174 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Oct  6 09:40:52 2003
;; MSG SIZE  rcvd: 49
```

В данном примере запрос выполняется 174 миллисекунды. Теперь запустим этот запрос снова:

```
$ dig www.example.com
; <<>> DiG 9.2.3rc4 <<>> www.example.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6398
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.example.com.               IN      A
;; ANSWER SECTION:
www.example.com.        172765  IN      A       192.0.34.166
;; Query time: 18 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Oct  6 09:41:27 2003
;; MSG SIZE  rcvd: 49
```

Повторный запрос для того же хоста занял всего 18 миллисекунд, т. е. приблизительно в десять раз меньше. Причина в том, что эта машина обращалась к кэширующему DNS-серверу.

Если поиск занимает очень много времени (или вообще происходит тайм-аут), это означает, что у вас есть проблемы с DNS, которые могут быть вызваны несколькими причинами:

Настройки `resolv.conf`

Если Postfix работает в chroot-окружении, возможно, что вы изменили файл `/etc/resolv.conf`, но забыли скопировать обновленный файл в chroot-окружение (обычно `/var/spool/postfix/etc/resolv.conf`).

Серверы доменных имен, перечисленные в файле /etc/resolv.conf, могут оказаться медленными или вообще не обслуживать запросы. Используйте команду dig, чтобы проверить для каждой строки /etc/resolv.conf, своевременно ли указанные серверы отвечают на ваши DNS-запросы.

Сетевые проблемы

Ваш канал связи с Интернетом может работать не так, как должен, или может быть переполнен. В таком случае следует подумать об увеличении пропускной способности канала или об управлении трафиком с целью повышения приоритета запросов к серверам имен.

Настройки межсетевого экрана

Межсетевой экран может блокировать пакеты сервера имен, пересылаемые на ваш почтовый сервер и от него.

Плохо работающий кэширующий сервер имен

Если у вас есть локальный кэширующий сервер имен, убедитесь в том, что он действительно работает.

Повышение производительности DNS-поиска

Если с вашими настройками /etc/resolv.conf, сетью и межсетевым экраном все хорошо, а вам все еще хочется ускорить свои DNS-запросы, то подумайте о запуске локального кэширующего сервера, такого как dnscache из пакета djbdns, или экземпляра BIND на вашем сервере или в вашей сети. Кэширование значительно ускоряет процесс поиска и в то же время уменьшает загруженность сети, т. к. повторный поиск не приводит к отправке исходящего пакета.

Проверка на отсутствие вашего сервера в списке открытых ретрансляторов

Если ваш сервер работает как открытый ретранслятор, то готовьтесь к тому, что многие почтовые серверы будут отклонять любые сообщения от ваших серверов. Более того, спамеры будут использовать вашу систему для отправки своей почты, увеличивая ее загруженность, т. к. системе придется обрабатывать не только ваших пользователей, но и злоумышленников.

Обычно сервер попадает в черный список, как только будет подтверждено, что он является открытым. Исключение из черного списка потребует множества усилий и может занять несколько дней или даже недель. Поэтому необходимо убедиться в том, что ваша система не является открытым ретранслятором. Попробуйте найти свой IP-адрес на сайте *http://openrbl.org*. Если ваш адрес присутствует в списке, немедленно исправьте положение. Пересылка должна быть разрешена только в следующих случаях:

- Клиент пользователя указан в параметре mynetworks.

- Клиент пользователя успешно прошел процедуру SMTP-аутентификации.
- Клиент пользователя успешно аутентифицировался при помощи клиентского сертификата TLS.

Отклонение сообщений несуществующим пользователям

Представляется разумным отклонять сообщения для пользователей, не существующих в вашей системе. Если бы сервер Postfix принимал такие сообщения, ему пришлось бы посылать отправителю уведомление о том, что сообщение не было доставлено. В случае спама и вирусов адрес отправителя практически наверняка не является истинным источником сообщения. В итоге уведомления MAILER-DAEMON будут несколько дней засорять очередь.

Сама по себе эта проблема была бы не такой серьезной, но дело в том, что если вы принимаете сообщения для пользователей, которые не существуют в системе, то система может хранить их в каком-то месте, которое рано или поздно заполнится; если же у вас работает ретранслятор (см. главу 13), то конечному получателю сообщения в конце концов придется отправить возврат отправителю конверта сообщения (значение `Return-Path` в заголовке сообщения). Кроме того, этот возврат сам может оказаться недоставимым, т. к. домен, указанный как домен отправителя, может не принимать никакой почты.

В любом случае эти возвраты заполонят вашу очередь или отправятся в почтовый ящик, указанный в параметре `2bounce_recipient` (он может принадлежать учетной записи `postmaster`). Возможно, вы столкнулись с подобной проблемой, если в вашей очереди присутствует что-то похожее:

```
$ mailq
-Queue ID- --Size-- ----Arrival Time---- -Sender/Recipient-------
63BE9CF331    10658 Mon Jan 12 14:38:30  MAILER-DAEMON
       (connect to mail3.quickspress.com[63.89.113.198]: Connection timed out)
                                          platinum@quickspress.com
1C932CF30E     3753 Sat Jan 10 16:16:38  MAILER-DAEMON
             (connect to mx.unrealdeals.biz[69.5.69.110]: Connection refused)
                                          EntrepreneurCareers@unrealdeals.biz
98EC3CF3F9     5505 Sat Jan 10 20:25:06  MAILER-DAEMON
         (connect to fhweb8.ifollowup.com[216.171.193.38]: Connection refused)
                                          root@fhweb8.ifollowup.com
50B14CF31E     5196 Mon Jan 12 11:35:11  MAILER-DAEMON
      (connect to mail.refilladvice.net[218.15.192.166]: Connection timed out)
                                          clintoncopeland@refilladvice.net
F4009CF39D     5452 Sun Jan 11 01:58:27  MAILER-DAEMON
         (connect to fhweb9.ifollowup.com[216.171.193.39]: Connection refused)
                                          root@fhweb9.ifollowup.com
-- 30 Kbytes in 5 Requests.
```

Здесь мы видим пять сообщений, которые были возвращены исходным отправителям (обратите внимание на то, что отправителем является MAILER_DAEMON), но во всех случаях почтовый сервер получателя оказался недостижимым.

Для того чтобы отклонять сообщения для несуществующих в системе получателей, следует указать в параметрах `local_recipient_maps` и `relay_recipient_maps` (последний используется, если у вас работает шлюз, который просто пересылает почту внутренним почтовым серверам) карты, содержащие действительных получателей.

Если возвраты выходят из-под контроля, то можно использовать черные списки RHSBL (см. главу 8) для отклонения сообщений от тех серверов, которые совсем не принимают возвраты (т. к. все возвраты, которые должны быть отправлены этим серверам, остаются в вашей почтовой очереди несколько дней). Например, вы можете использовать черный список RHSBL на веб-сайте RFC-Ignorant.Org (*http://rfc-ignorant.org*):

```
check_rhsbl_sender dsn.rfc-ignorant.org
```

Блокирование сообщений от сетей из черных списков

Существует множество видов списков заблокированных адресов (blocklists) и черных списков DNS, в которых по разным причинам содержатся индивидуальные IP-адреса, целые диапазоны IP-адресов и даже домены отправителей (по крайней мере по одному списку для каждого типа ненадлежащего поведения).

Наиболее полезны списки, в которых приведены открытые ретрансляторы и открытые прокси-серверы, т. к. их можно автоматически и объективно проверять. Перечислим лишь несколько таких списков:

- `relays.ordb.org`
- `list.dsbl.org`
- `cbl.abuseat.org`
- `dul.dnsbl.sorbs.net`

Примечание

Черные списки меняются очень часто. Уже завтра может оказаться, что актуальный сегодня черный список больше не обслуживается.

Случайное попадание сервера в такие черные списки маловероятно, т. к. существуют четкие критерии занесения в них адресов. Работа в качестве открытого прокси-сервера или открытого ретранслятора считается неправильной, так что использование таких списков вызывает общественное давление на администраторов неправильно настроенных систем (конечно, они могут не знать или просто не думать об этом).

Отклонение сообщений из неизвестных доменов отправителей

Если это возможно, не принимайте сообщения, отправитель конверта которых относится к недействительному домену. В случае проблем с доставкой уведомление об ошибке всегда отправляется отправителю конверта, и если этот адрес относится к несуществующему домену, то уведомление будет некуда отправить.

Postfix пытается отправить сообщение об ошибке, обнаруживает, что доставить его невозможно (т. к. домен отправителя конверта не существует), и отправляет его в почтовый ящик `2bounce_notice_recipient`.

Вы можете обойти эту проблему, добавив ограничение `reject_unknown_sender_domain` в список `smtpd_sender_restrictions` или `smtpd_recipient_restrictions`, о чем мы говорили в главе 8.

Уменьшение частоты попыток повторной передачи

Если у вас есть большой объем почты, который вашему серверу не удается доставить с нескольких первых попыток, подумайте об использовании резервного ретранслятора (задайте параметр `fallback_relay`) или об увеличении времени отсрочки передачи (параметр `maximal_backoff_time`) для уменьшения частоты, с которой отложенные сообщения вновь попадают в очередь `active`.

При отсутствии резервного ретранслятора Postfix тратит драгоценное время на попытки доставить сообщения выключенным или недостижимым машинам. Каждой из таких попыток доставки соответствует один процесс `smtp`, который находится в состоянии ожидания до тех пор, пока не будет достигнут тайм-аут. Резервный ретранслятор может заняться черной работой – повторной доставкой сообщений, которые не удалось доставить с первого раза. Это означает, что ваш обычный почтовый сервер сможет работать с тайм-аутами по умолчанию или даже уменьшить их значения, ускорив тем самым доставку.

Увеличение же значения параметра `maximal_backoff_time` увеличивает максимальный промежуток времени, в течение которого сервер игнорирует некоторое место назначения после неудачной доставки. Следовательно, сервер Postfix осуществляет меньше попыток общения с проблемными серверами.

Поиск узких мест

В этом разделе будет рассказано о том, как выявить узкие места вашей системы. Возможно, прежде чем разбираться с этими вопросами, вам стоит вернуться к главе 5, чтобы вспомнить, какой демон чем занимается.

Для выполнения своей работы все демоны Postfix обращаются к одной или нескольким очередям, так что необходимо знать статус очередей. Вот несколько очередей, которые будут нас интересовать:

- `incoming`
- `deferred`
- `active`
- `maildrop`

Когда сообщение попадает в систему, оно становится файлом, который сервер Postfix перемещает из очереди в очередь. Если в одной из очередей накапливается очень много сообщений, это может быть вызвано проблемами производительности. Для того чтобы разобраться с разными очередями, Виктор Духовны (Victor Duchovni) создал замечательную утилиту `qshape`, показывающую распределение сообщений между очередями. Эта программа читает файлы очередей напрямую, минуя команду `mailq`, следовательно, ее могут запускать только пользователи `root` и `postfix`. Вы можете загрузить ее с веб-сайта по адресу *http://sb-serv.stahl.bau.tu-bs.de/~hildeb/postfix/scripts*. В последних версиях Postfix этот сценарий включен в архив исходных текстов.

Утилита `qshape` выводит табличное представление содержимого очередей Postfix. Строки содержат количество сообщений для конкретного места назначения, а также их общее количество. Столбцы отображают возраст сообщений. Посмотрим, например, на выходные данные для очереди `hold` (параметр `hold`), которые содержат первые десять строк распределения доменов отправителей (параметр `-s`) для полученного спама (это будут самые злостные спамеры):

```
# qshape -s hold
                                T   5  10  20  40  80 160 320 320+
                            TOTAL  12   0   0   0   0   2   2   0   6
                        hotmail.de   2   0   0   0   0   0   0   0   2
alb-24-194-161-132.nycap.rr.com      1   0   0   0   0   1   0   0   0
                       freeenet.de   1   0   0   0   0   0   1   0   0
                     x4u2.desy.de    1   0   0   0   0   0   0   0   1
                          csi.com    1   0   0   0   0   0   1   0   0
                            da.ru    1   0   0   0   0   0   0   0   1
                        freeuk.com   1   0   0   0   0   1   0   0   0
mx5.outrageouscourtiers.com          1   0   0   0   0   0   0   0   1
                       online.de     1   0   0   0   0   0   0   0   0
                   molgen.mpg.de     1   0   0   0   0   0   0   0   1
                      charite.de     1   0   0   0   0   0   0   0   0
```

Столбец `T` содержит общее количество сообщений для каждого домена. Другие столбцы приводят количество сообщений, возраст которых превышает некоторое значение (измеряемое в минутах), но при этом не превышает возраст сообщений из столбца, расположенного правее. В нашем случае есть два сообщения, которые якобы отправлены с `hotmail.de`. Возраст обоих превышает 320 минут.

По умолчанию утилита qshape выводит статистику для двух очередей: incoming и active, т. к. они непосредственно связаны с общей производительностью. Вы можете указать в командной строке другой набор очередей:

```
$ qshape deferred
$ qshape incoming active deferred
```

Итак, мы умеем отслеживать занятые очереди. В последующих разделах будет рассказано о том, как избавиться от нехватки ресурсов для каждого типа очередей. Мы также приведем формулы, которые позволят определить, может ли система обработать заданный объем почты и когда следует использовать резервные ретрансляторы.

Очередь Incoming

Как уже говорилось в предыдущем разделе «Поиск узких мест», Postfix-служба cleanup сохраняет все новые сообщения в очереди incoming. До тех пор пока новые файлы не будут завершены и готовы для дальнейшей обработки, они получают режим доступа 0600, затем – 0700. В обычных условиях очередь incoming практически пуста и содержит только файлы с доступом 0600, т. к. диспетчер очередей должен иметь возможность импортировать новые сообщения в очередь active, как только служба cleanup закончит с ними работать.

Однако очередь incoming начинает расти, если скорость поступления сообщений превышает скорость, с которой диспетчер очередей может перемещать сообщения в очередь active. На этом этапе единственное, что замедляет работу диспетчера очередей, – это служба trivial-rewrite. Если диспетчер очередей работает недостаточно быстро, то возможно, что вы используете медленные средства поиска (такие как MySQL и LDAP) или вам следует ускорить работу серверов, предоставляющих сервисы поиска.

Примечание

Если вы используете IPC-карты (interprocess communication – взаимодействие процессов) с высокой или переменной задержкой, такие как LDAP и SQL, серверу Postfix требуется больше времени для получения почты. Соответственно Postfix будет запускать больше процессов smtpd (и cleanup) и скоро достигнет разрешенного предела для количества процессов smtpd. Из-за этих медленных табличных просмотров агент доставки (local, pipe, virtual, lmtp), вероятно, завершит работу за меньшее время, чем необходимо smtpd для получения почты, так что Postfix будет запускать меньше агентов доставки, чем ожидается.

Для улучшения ситуации при работе с LDAP можно попытаться избежать LDAP-связывания. Установите в файле конфигурации запросов LDAP bind = no. Тогда Postfix будет связываться с сервером LDAP анонимно, за счет чего будут исключены накладные расходы на аутентификацию и проверку пароля.

Надо сказать, что поиск в IPC-картах сам по себе занимает больше времени, чем поиск в картах на основе файлов (таких как hash, btree, dbm и cdb), и демоны

Postfix не могут ничего сделать, пока ожидают завершения поиска. Если бы поиск был более быстрым, то быстрее был бы и Postfix.

Мы говорили о том, что у таких карт есть и свои достоинства, которые могут перевешивать недостатки. Одним из основных плюсов является то, что серверу Postfix не нужно убивать и заново запускать процесс для повторного открытия карты при изменении ее содержимого. Так что вы можете решить все-таки использовать именно такие карты и выполнить работы по настройке базы данных.

Если нехватка ресурсов имеет место именно в очереди `incoming`, то прием сообщений имеет больший приоритет, чем их отправка. Для того чтобы предотвратить истощение исходящего потока, можно поэкспериментировать с параметром `in_flow_delay`, чтобы ограничить интенсивность поступления сообщений, когда диспетчер очередей начнет отставать. Служба `cleanup` останавливается на количество секунд, указанное в параметре `in_flow_delay`, прежде чем создавать новый файл очереди, если она не может получить маркер от диспетчера очередей.

Дело в том, что число процессов `cleanup` обычно ограничено параллелизмом сервера SMTP (`smtpd`). Количество входящих сообщений в секунду может превышать количество исходящих сообщений не более чем на частное от деления числа SMTP-соединений на значение параметра `in_flow_delay`. Для того чтобы определить текущее количество входящих SMTP-соединений, используйте команды `ps` и `grep`:

```
# ps auxww| grep smtpd | grep -v grep |  wc --lines
     22
```

В этой системе работают 22 процесса `smtpd`. Эта команда учитывает все процессы `smtpd`, так что если у вас несколько конфигураций `smtpd` (например, если вы используете фильтр содержимого, который возвращает почту обратно в очередь по SMTP), то вам следует использовать специальный шаблон `grep` для определения количества демонов `smtpd`, принимающих сообщения от внешней сети:

```
# ps auxww| grep smtpd | grep -v grep | grep -v localhost | wc --lines
      9
```

Если вы используете для максимального количества процессов значение по умолчанию, равное 100, а значение `in_flow_delay` равно одной секунде, то этого достаточно для ограничения потока от одного неуправляемого источника большого количества сообщений одним сообщением в секунду. Однако это ограничение недостаточно строгое для того, чтобы изменить слишком высокую интенсивность поступления сообщений от множества источников одновременно.

Если ваш сервер атакуют с разных сторон, то лучше всего сделать SMTP-сеансы максимально короткими (нулевое значение `smtpd_error_sleep_time` и низкое значение `smtpd_hard_error_limit`, сервер Postfix будет отключать соединения, для которых данный предел будет превышен). Применяйте этот способ, только если очередь `incoming` растет да-

же в том случае, когда очередь active не полна, а служба trivial-rewrite использует быстрый механизм поиска транспорта.

Если вы примените эти лекарства, но очередь incoming все равно останется переполненной, притом что очередь active не переполнена, то, вероятнее всего, проблема в вашей системе ввода-вывода. Сообщения прибывают и записываются на диск, но процессам smtpd и qmgr необходим доступ к этому же ресурсу (очередь на диске), и вы ограничены скоростью подсистемы ввода-вывода.

В этом случае необходимо добавить *очень много* памяти почтовому серверу (чтобы увеличить пул обмена с дисковой памятью через кэш для операционной системы) или переместить каталог очереди на одно из следующих устройств:

- Массив RAID в режиме Striping.
- RAM-диск с резервным питанием (для безрассудных смельчаков – при отказе системы почта будет потеряна).

Очередь maildrop

Сообщения, переданные Postfix-командой sendmail, но еще не отправленные в основные очереди Postfix службой pickup, находятся в очереди maildrop (вы можете отправить сообщение при помощи команды sendmail, и оно будет добавлено в очередь maildrop даже в том случае, если система Postfix не работает). Однопоточная служба pickup просматривает каталог очереди maildrop периодически или при получении уведомления о прибытии нового сообщения от программы postdrop.

Скорость, с которой служба pickup может вставлять сообщения в основные очереди, главным образом определяется временем доступа к диску, т. к. она должна передать сообщение на постоянное хранение, прежде чем завершить работу. Это же относится и к программе postdrop, которая записывает сообщения в каталог maildrop.

Служба pickup является однопоточной, поэтому она может доставлять только одно сообщение одновременно, при этом общая скорость ограничена задержкой дискового ввода-вывода (и загрузкой процессора, если она имеет место), вызываемой службой cleanup, т. к. каждое сообщение, обрабатываемое pickup, должно пройти через cleanup. Как вы помните, cleanup выполняет проверки header_checks, body_checks и т. д., которые могут потреблять значительную процессорную мощность. Затем cleanup записывает сообщение в файл очереди – и этот процесс ограничен задержкой дискового ввода-вывода. Если у вас наблюдается переполнение очереди maildrop, то, возможно, вы столкнулись с одной из следующих проблем:

- Избыточная интенсивность поступления локальных сообщений.
- Избыточное потребление вычислительной мощности службой cleanup, вызванное избыточными проверками тела сообщений.

Однако помните, что, когда полна очередь active, служба cleanup пытается замедлить вброс сообщений, делая для каждого сообщения паузу в соответствии с параметром in_flow_delay. В этом случае переполнение очереди maildrop может быть результатом переполнения на следующих этапах обработки.

Не пытайтесь доставлять большой объем сообщений через службу pickup. Если вы обрабатываете большие объемы сообщений, то вам следует избегать использования фильтров содержимого, которые возвращают проанализированную почту обратно посредством sendmail и postdrop. Вместо этого используйте для ввода почты SMTP-соединение. Существует множество программ, которые могут сделать это, в том числе mini_sendmail (*http://www.acme.com/software/mini_sendmail*).

Если вы получили множество локально переданных сообщений, то, возможно, у вас произошло зацикливание переадресации или вышла из-под контроля программа нотификации. Кроме того, команда postsuper -r может помещать выбранные сообщения в очередь maildrop для повторной обработки. Это полезно для переустановки старых настроек content_filter, но повторная постановка в очередь большого количества сообщений (посредством команды postsuper -r) может привести к резкому увеличению размера очереди maildrop.

Очередь deferred

Если Postfix в силу каких-то временных причин не может доставить сообщение некоторым из своих получателей, он помещает сообщение в очередь deferred в надежде на последующую удачную доставку. Диспетчер очередей просматривает очередь deferred через определенные промежутки времени, задаваемые параметром queue_run_delay. Как уже говорилось в главе 5, диспетчер очередей выбирает сообщения по кругу из двух очередей – incoming и deferred, с тем чтобы предотвратить преобладание отложенных сообщений в очереди active.

При каждом просмотре очереди deferred какая-то ее часть отправляется обратно в очередь active для повторения попытки доставки, т. к. при отправке каждого сообщения в очередь deferred делается отметка о времени его «замораживания». В Postfix это достигается путем сдвига времени модификации файла очереди в будущее. Файл очереди получит право на новую попытку отправки только тогда, когда наступит время его изменения.

Время замораживания может быть не меньше значения параметра minimal_backoff_time и не больше значения maximal_backoff_time. Postfix устанавливает время следующей попытки, удваивая время пребывания сообщения в очереди и корректируя полученное значение так, чтобы оно попадало в указанные рамки. В результате получается, что Postfix чаще пытается отправить более свежие сообщения.

Если на вашем интенсивно работающем сайте большая очередь отложенных сообщений, вы можете так настроить параметры queue_run_delay, minimal_backoff_time и maximal_backoff_time, чтобы после первой неудачной доставки задержка была небольшой, а после нескольких неудачных попыток – более долгой. Тем самым вы уменьшите частоту повторной передачи старых сообщений, а значит, и количество ранее отложенной почты в очереди active.

Предупреждение

Частой причиной больших очередей отложенных сообщений является отсутствие проверки действительности получателей на этапе открытия SMTP-соединения (в разделе «Сообщения неизвестным получателям» главы 8 объяснялось, почему необходимо проверять получателей).

Если сервер, на котором накоплен большой объем отложенной почты, временно отключается, то возможна такая ситуация, когда для всех сообщений очереди deferred время повторной попытки наступит одновременно при последующем включении сервера. Это может привести к значительному увеличению загруженности очереди active. Кроме того, это же явление будет повторяться с периодичностью, примерно равной значению maximal_backoff_time (в секундах), если большая часть сообщений опять окажется отложенной.

Идеальным решением проблемы будет сдвиг стандартного времени между попытками отправки на некоторую случайную величину, чтобы снизить вероятность одновременной отправки всей очереди deferred на повторную обработку.

Очередь active

Как уже говорилось в главе 5, диспетчер очередей является планировщиком агентов доставки, который пытается обеспечить быструю и корректную доставку сообщений всем получателям в рамках выделенных ресурсов. Переполнение очереди active происходит тогда, когда одно или несколько мест назначения принимают сообщения с меньшей скоростью, чем они поступают.

Если сервер назначения в течение некоторого времени не функционирует, то диспетчер очередей помечает его как «dead» и сразу же задерживает всю почту для этого направления, вообще не передавая ее агенту доставки. Поэтому такие сообщения быстро покидают очередь active и попадают в очередь deferred. Если же сервер назначения просто очень медленный или какая-то проблема приводит к чрезмерно интенсивному поступлению сообщений, то очередь active растет и переполняется сообщениями, предназначенными для медленного сервера. Бороться с переполнением можно всего двумя способами:

- Уменьшением интенсивности поступления сообщений.
- Увеличением пропускной способности.

Увеличение пропускной способности требует или увеличения параллелизма (количества одновременно работающих Postfix-процессов `smtp`), или уменьшения задержки доставки (за счет использования более быстрой сети, изменения типа карты, ускорения DNS-поиска и т. д.). Для увеличения параллелизма следует увеличить значение параметра `default_process_limit` в файле `main.cf`. Однако если вы хотите сделать его зависящим от места назначения, то необходимо найти тот медленный транспорт (например, транспорт для `content_filter`) или сервер (например, некоторые крупные серверы бесплатной почты), который преобладает в очереди `active` (для этого удобно использовать `qshape`). Выявив нарушителя (пусть это будет транспорт с именем `name`), устанавливаем параметр `name_destination_concurrency_limit`. Более подробную информацию о такой настройке вы найдете в разделе «Настройка альтернативного транспорта» в конце главы.

Не забывайте о том, что максимальное количество процессов, используемых для любой службы, ограничено параметрами в файлах `master.cf` и `main.cf`.

Примечание

Имейте в виду, что ваша операционная система должна суметь обработать увеличившееся количество процессов и открытых файлов. Прочтите раздел «Running hundreds of processes» («Запуск сотен процессов») на сайте Postfix по адресу *http://www.postfix.org/faq.html*.

Задержку в некоторых случаях можно уменьшить за счет ускорения DNS-поиска (см. раздел «Ускорение DNS-поиска» в начале главы) и просмотра карт (см. главу 5, раздел «Базы данных (MySQL, PostgreSQL, LDAP)»). Также может помочь уменьшение тайм-аута для сильно загруженных систем с множеством MX-хостов. Однако ничто из этого не поможет, если принимающая сторона не справляется с получением (например, если вы отправляете сообщения медленным серверам, таким как некоторые бесплатные почтовые серверы).

Еще одной причиной переполнения очереди `active` может явиться необоснованное вливание в нее всей очереди `deferred`. Очередь `deferred` хранит сообщения, которые, вероятно, не будут доставлены, по крайней мере, с первой произвольной попытки. Более того, вероятно, что неудачная попытка отправки, приводящая к отсрочке, займет много времени, ведь серверу Postfix придется ждать истечения тайм-аута.

Предупреждение

Некоторым администраторам хочется поскорее попытаться снова отправить все сообщения большой очереди `deferred`, но это может оказаться неэффективным решением и даже усугубить положение. Выпускайте все сообщения из очереди `deferred` только в том случае, если считаете, что большинство сообщений действительно достигнут своих адресатов при следующей попытке отправки! Так что сначала подумайте, разберитесь, а потом уже выгружайте эту очередь!

Наконец, по возможности избегайте перезагрузки конфигурации и перезапуска Postfix. При перезапуске диспетчера очередей может оказаться, что в каталоге очереди `active` есть сообщения, а реальная очередь `active` (в памяти) пуста. Для того чтобы восстановить состояние очереди в памяти сервера, диспетчер очередей перемещает все сообщения `active` обратно в очередь `incoming` и полагается на обычный просмотр `incoming` для заполнения `active`. Процесс перемещения сообщения туда и обратно с повторением поиска в транспортных таблицах и повторной загрузкой сообщений в память требует больших затрат.

Предупреждение

Команда `postfix reload` перезапускает диспетчер очередей, так что следует избегать изменений в файлах конфигурации, для вступления которых в силу необходимо выполнение команды `postfix reload` на загруженных рабочих серверах.

Неравенство переполнения очереди возвратами

Если очередь `deferred` полна недоставимых возвратов, то в переполнении очереди виноват не сервер Postfix. Переполнение является следствием высокой средней задержки при наличии большого запаса недоставимой почты, т. к. демоны `smtp` пытаются отправить почту по истечении заданных промежутков времени. Виктор Духовны (Victor Duchovni) вывел неравенство переполнения, которое поможет вам определить, есть ли у вас подобная проблема.

В очереди с большим числом возвратов, которые никогда не будут доставлены, количество ненужных сообщений, попадающих в очередь `active` за один проход очереди `deferred`, определяется такой формулой:

$$\frac{\texttt{size\_of\_the\_queue} \times \texttt{queue\_run\_delay}}{\texttt{maximal\_backoff\_time}}$$

Количество сообщений, обработанных за проход очереди, не превышает следующей величины:

$$\frac{\texttt{queue\_run\_delay} \times \texttt{default\_process\_limit}}{\texttt{smtp\_connect\_timeout} \times \texttt{M}}$$

При исчерпании доступного количества процессов можно считать, что количество возвратов намного больше количества процессов. (Предполагается, что `default_process_limit` определяется количеством демонов `smtp`. Если вы увеличили значение в столбце `maxproc` в файле `master.cf`, то подставьте в выражение это значение.)

Объединяем полученные ранее выражения и получаем:

$$\frac{\texttt{size\_of\_the\_queue} \times \texttt{queue\_run\_delay}}{\texttt{maximal\_backoff\_time}} \geq \frac{\texttt{queue\_run\_delay} \times \texttt{default\_process\_limit}}{\texttt{smtp\_connect\_timeout} \times \texttt{M}}$$

Умножив обе части уравнения на отношение maximal_backoff_time / queue_run_delay, получим неравенство переполнения:

$$P \times B \geq Q \times T \times M$$

где параметры имеют следующие значения:

P – максимальное разрешенное количество процессов транспорта smtp, полученное при выполнении такой команды:

```
# postconf default_process_limit
default_process_limit = 100
```

Примечание

Проверьте значение maxproc в строке для smtp файла master.cf, чтобы посмотреть, существует ли явное ограничение для процессов транспорта smtp.

B – максимальное время отсрочки, полученное такой командой:

```
# postconf maximal_backoff_time
maximal_backoff_time = 4000s
```

Q – количество возвратов в очереди (предполагается, что количество разных мест назначения, в которые адресованы эти возвраты, составляет не менее P/destination_concurrency_limit).

T – тайм-аут smtp-соединения, полученный такой командой:

```
# postconf smtp_connect_timeout
smtp_connect_timeout = 30s
```

M – среднее количество IP-адресов типа MX. Если в домене несколько MX-записей, Postfix должен проверить каждую. Вам не нужно точное количество; просто оцените его и используйте qshape для оценки количества MX-записей в преобладающих направлениях назначения.

Примечание

По умолчанию каждому серверу назначения отводится не более 20 агентов доставки (default_destination_concurrency_limit = 20), так что продолжайте добавлять серверы назначения до тех пор, пока не будет достигнуто максимальное разрешенное количество процессов.

На деле же следует просто определить диапазон для M и, возможно, ограничить его, задав параметр smtp_mx_address_limit (верхний предел для количества MX-адресов, которые будет проверять сервер Postfix).

Если ваша система не удовлетворяет этому неравенству, это означает наличие серьезных проблем. Сделайте все возможное для уменьшения значения правой части и увеличения его левой части.

Когда вы будете пытаться уменьшить значение в правой части, помните о следующих фактах:

- Вы можете уменьшить Q, только удалив возвраты.
- Вы можете уменьшить T, создав выделенный smtp-транспорт для доменов получателей возвратов и уменьшив для него тайм-аут SMTP-соединения.
- Вы не можете уменьшить M, т. к. не вы запускаете серверы получателей и службы DNS.

Для увеличения значения слева можно предложить следующие действия:

- Увеличить P легко. Измените разрешенное по умолчанию количество процессов или отредактируйте файл master.cf, разрешив большее количество smtp-процессов.
- Чтобы увеличить B, увеличьте максимальное время отсрочки.

Давайте рассмотрим несколько примеров. Первый выполнен на версии Postfix 1.*x* для операционной системы Solaris с 2000 отложенных сообщений в очереди.

В версии Postfix 1.*x* параметр smtp_connection_timeout – это тайм-аут TCP-соединения операционной системы. В Solaris его значение по умолчанию равно приблизительно 180 секундам, в Linux оно гораздо больше. Если вы используете значение по умолчанию для default_process_limit (оно равно 50), то результат будет таким:

```
P * B >= Q * T
50 * 4000 >= 2000 * 180
200000 >= 360000
```

Неравенство не выполнено, т. к. максимальное время отсрочки maximal_backoff_time, равное 4000 секунд, слишком мало, особенно если серверы назначения отложенных сообщений имеют несколько MX-записей.

Теперь рассмотрим случай использования версии Postfix 1.1.11 (или выше) с 2000 отложенных сообщений в очереди.

Для более поздних версий Postfix (как минимум 1.1.11-20020717) тайм-аут соединения равен 30 секундам на всех платформах. В версии Postfix 1.1.12-20021212 ограничение на число процессов по умолчанию увеличено до 100. Теперь картина будет следующей:

```
P * B >= Q * T
100 * 4000 >= 2000 * 30
400000 >= 60000
```

Неравенство выполнено. И более того, т. к. время отсрочки по умолчанию равно 4000 секунд, размер очереди значительно ниже критического уровня даже в том случае, если количество MX-хостов для стандартного сервера назначения отложенной почты равно 4. Более новые версии Postfix могут обрабатывать гораздо большие очереди, полные отложенных сообщений.

При установке значения default_process_limit равным 100 критический размер очереди приблизительно таков:

```
100 * 4000 / 30 = 13000
```

Если количество MX-записей больше 1, то он может быть меньше. Если количество процессов ограничено 500, тайм-аут равен 10 секундам, а maximal_backoff_time – 4 часам, то критический размер очереди выражается таким огромным числом:

```
500 * 14400 / 10 = 720000
```

Однако этот абсурдный предел соответствовал бы 500 процессам, занятым попытками отсылки новых сообщений каждые 10 секунд (другими словами, каждую секунду очередь deferred покидали бы и вновь попадали в нее 50 сообщений).

Использование резервных ретрансляторов

Если основная очередь страдает от перегрузки, то стоит создать второй сервер (или экземпляр Postfix, поддержка многоэкземплярности планируется в версии 2.2) для работы с повторными попытками доставки. Тогда вы сможете нормально настроить основную очередь, возможно, с очень небольшими тайм-аутами, а все не доставленные с первой попытки сообщения попадут (для осуществления повторных попыток доставки) в резервную очередь или на резервный ретранслятор, который будет настроен специальным образом, как было описано ранее в этой главе.

Для создания такой конфигурации установите в файле main.cf для вашего основного сервера Postfix следующие параметры:

```
smtp_connect_timeout = 5s
smtp_mx_address_limit = 3
#fallback_relay = [127.0.0.1]:20025
# для нескольких экземпляров Postfix на одной машине
fallback_relay = fallback.example.com
# для другого экземпляра на другой машине
```

Затем задайте максимально допустимый размер очереди для сервера, указанного в параметре fallback_relay. Например, используйте такие параметры, чтобы задать размер, равный 720 000:

```
smtp_connect_timeout = 10s
smtp_mx_address_limit = 5
default_process_limit = 500
bounce_queue_lifetime = 2d
maximal_backoff_time = 4h
```

Примечание

Возможно, вы захотите немного увеличить значение smtp_connect_timeout. Некоторые хосты и сети действительно настолько медленно работают...

Повышение пропускной способности

Если ваша машина пересылает большой объем входящей почты, то можно сделать так, чтобы пересылкой сообщений в домены-получатели занимался отдельный транспорт. В этом разделе будем называть такой транспорт «ретранслятором». В версии Postfix 2.*x* он настраивается так:

1. Присваиваем параметру `relay_destination_concurrency_limit` большое значение (например, 50).
2. Делаем так, чтобы запись `master.cf` для ретранслятора содержала строку `-o smtp_ connect_timeout=$relay_connect_timeout` (без пробелов вокруг знака равенства).
3. Устанавливаем в `main.cf` значение `relay_connect_timeout` равным 5 или 1.

Если вы используете фильтрацию содержимого (проверку на вирусы) при помощи `content_filter` на основе SMTP, то убедитесь в том, что в настройке отправляющего транспорта присутствует `-o disable_dns_lookups=yes`. Это также полезно при отправке всей почты фиксированному серверу, когда не приходится просматривать MX-записи (например, при использовании функциональности `relayhost`).

Настройка альтернативного транспорта

Если вы постоянно отправляете большие объемы почты в сети со множеством почтовых обменников (характерным примером является Hotmail), то не имеет смысла использовать значения тайм-аутов по умолчанию. Вероятно, сервер Postfix мог бы доставлять почту, предназначенную для таких доменов, быстрее, если бы не тратил столько времени на каждый неработоспособный почтовый обменник. Этому вопросу и будет посвящен данный раздел.

Начинаем с определения нового `smtp`-транспорта `deadbeats` в нашем файле `master.cf`. Для этого копируем строку для транспорта `smtp`, переименовываем его в `deadbeats` и уменьшаем значение `smtp_connect_timeout`:

```
deadbeats unix  -       -       -       -       -       smtp
     -o smtp_connect_timeout=$deadbeats_connect_timeout
```

Тайм-аут по умолчанию равен 30 секундам, мы же зададим для `deadbeats_connect_timeout` в файле `main.cf` значение 5 секунд:

```
deadbeats_connect_timeout = 5
```

Теперь все в том же файле `main.cf` сообщим серверу Postfix о том, что следует использовать данный специальный транспорт для отправки сообщений определенным доменам назначения, – установим параметр `transport_maps`:

```
transport_maps = hash:/etc/postfix/transport
```

Создаем карту `/etc/postfix/transport` следующим образом:

```
yahoo.com               deadbeats:
# yahoo.com has 3 MX host, with 9 A records in total
compuserve.com          deadbeats:
# compuserve.com has 3 MX hosts with 4 A records each
aol.com                 deadbeats:
# aol.com has 4 MX hosts with 18 A records in total
hotmail.com             deadbeats:
# hotmail.com has 4 MX hosts with 10 A records in total
hotmail.de              deadbeats:
# hotmail.de has no MX hosts, but 6 A records
```

Для того чтобы карта заработала, запускаем команду `postmap hash:/etc/postfix/transport` и перезагружаем конфигурацию.

Приложения

Последнюю часть этой книги составляют приложения. Они помогут вам начать работу с Postfix, в случае необходимости – устранить неисправности, а кроме того, в них будут предложены некоторые справочные материалы, которые могут понадобится в процессе настройки сервера:

Установка Postfix

Приложение A содержит инструкции по установке Postfix из исходных текстов, а также из дистрибутивов Debian и Red Hat Linux.

Устранение неисправностей Postfix

Если у вас возникли проблемы при попытке изменения конфигурации, обратитесь к приложению B: в нем рассмотрены наиболее часто встречающиеся проблемы и даны общие советы по выяснению их причин.

Справочник подсетей в нотации CIDR и кодов ответов SMTP

Не каждый может запомнить, как записываются подсети в нотации CIDR и как расшифровываются коды ответов SMTP-сервера. Мы собрали эти справочные сведения в приложении C.

A

Установка Postfix

В этом приложении будет рассказано о том, как собрать Postfix из исходных текстов, а также о том, как установить, подготовить и собрать пакеты для Debian Linux и Red Hat Linux.

Исходные тексты Postfix

Ссылки на исходные тексты Postfix вы найдете по адресу *http://www.postfix.org/download.html*. Существует множество сайтов-зеркал, среди которых вы можете выбрать ближайший к вам для ускорения процесса.

Прежде чем загружать исходные тексты, необходимо понять разницу между экспериментальными (промежуточными) и официальными версиями. Официальная версия остается неизменной (за исключением исправления ошибок и патчей для обеспечения переносимости). Зато промежуточные версии содержат новейшие непроверенные функции. Та часть кода промежуточных версий, которая работает (и перестает изменяться), со временем попадает в официальную редакцию.

Официальные редакции Postfix называются следующим образом: `postfix-`*`a`*`.`*`b`*`.`*`c`*`.tgz`, где *a*, *b* и *c* имеют такие значения:

`a` Старший номер редакции (значительные изменения структуры пакета).

`b` Младший номер редакции (новые возможности).

`c` Номер патча (исправления ошибок).

Имена промежуточных версий имеют вид `postfix-`*`a`*`.`*`b`*`-`*`yyyymmdd`*`.tgz`, где *yyyymmdd* – это дата выпуска. Параметр конфигурации `mail_release_date` хранит дату выпуска как для официальной, так и для промежуточной версий.

Когда вы устанавливаете официальный патч, номер патча и дата выпуска изменяются. Однако новая промежуточная версия будет иметь только другую дату выпуска, даже если в ней исправлены те же самые ошибки, что и в патче официальной версии.

Установка патчей

Устанавливая патчи сторонних компаний, вы можете получить доступ к ряду специальных функций. Перечень патчей Postfix опубликован по адресу *http://www.postfix.org/addon.html* (если вы не умеете устанавливать патчи, то, вероятно, лучше этого не делать).

Патчи имеют свою собственную документацию, и, учитывая то, что они значительно изменяют поведение и возможности Postfix, следует внимательно отнестись к предложенным инструкциям.

Сборка и установка на основе исходных текстов

После распаковки пакета исходных текстов командой `tar xfz postfix-a.b.c.tgz`, вы, вероятно, захотите подогнать сборку под свои потребности, указав интересующие вас возможности. Документация по таким возможностям, как поддержка BerkeleyDB, PCRE, MySQL и SASL (SMTP-AUTH), находится в каталоге `README_FILES`. Каждый файл этого каталога содержит инструкции по настройке переменных окружения, влияющих на процесс сборки.

Если вас интересует функциональность, которая описана в книге и не входит в конфигурацию по умолчанию, то, вероятнее всего, вы найдете инструкции по сборке Postfix с данной функциональностью в начале соответствующей главы. Полный перечень доступных возможностей приведен в файле `INSTALL`, поставляемом вместе с Postfix. Процедура сборки всегда одинакова:

1. Устанавливаем в переменной окружения `AUXLIBS` параметры компоновщика.
2. Устанавливаем в переменной окружения `CCARGS` параметры компилятора и препроцессора.
3. Запускаем `make makefiles` для создания Makefile.

Например, на машине с древней HP-UX 10.20 команда для включения в сборку поддержки BerkeleyDB выглядела следующим образом (путь к библиотекам – `/users2/local/BerkeleyDB-4.0.14/lib`, путь к включаемым файлам – `/users2/local/BerkeleyDB-4.0.14/include`):

```
$ AUXLIBS='-L/users2/local/BerkeleyDB-4.0.14/lib -ldb' \
  CCARGS='-DHAS_DB -I/users2/local/BerkeleyDB-4.0.14/include' \
  make makefiles
```

Если вместо этого вы хотите установить CDB, то установите патч, затем библиотеки CDB и выполните такую команду:

```
$ AUXLIBS='-L/usr/local/lib -lcdb' \
  CCARGS='-DHAS_CDB -I/usr/local/include' \
  make makefiles
```

Затем от имени обычного пользователя запустите `make`:

```
$ make
```

После сборки пакета необходимо определить, впервые ли устанавливается Postfix, или речь идет о переходе на более новую версию. Для первой установки выполните от имени `root` такую команду:

```
# make install
```

Она задаст вам ряд вопросов о путях установки.

Если же вы обновляете существующую версию, то выполните от имени `root` такие команды:

```
# postfix stop
# make upgrade
# postfix start
```

При обновлении версии инсталлятор извлекает пути из файла `main.cf` уже существующей версии и использует пути и файлы конфигурации повторно.

Запуск и остановка Postfix

Как вы уже знаете, сервером Postfix можно управлять при помощи программы `postfix`, которая принимает следующие параметры:

`start` Запуск почтовой системы Postfix, а также запуск проверки конфигурации, которая будет описана далее.

`stop` Корректное отключение почтовой системы, работающие процессы завершают работу при первой возможности.

`check` Проверка корректности конфигурации Postfix. Эта команда предупреждает о неправильных правах собственности и доступа для каталогов или файлов и создает недостающие каталоги.

Установка Postfix для Debian Linux

Установка Postfix для Debian Linux без труда осуществляется посредством команды `apt-get`. Несмотря на то что версии Postfix со временем меняются, действия, необходимые для установки и интеграции в Debian Linux, остаются неизменными.

Установка Postfix

При первой установке системы в Debian вы можете установить Postfix из пакета. Однако если вручную не изменить выбор почтовой системы по умолчанию, то после первой загрузки системы агентом передачи со-

общений по умолчанию будет назначен exim. Для того чтобы определить, установлен ли Postfix в вашей системе, выполним команду dpkg, которая выведет номер установленной версии (при ее наличии):

```
$ dpkg -l 'postfix*'
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err:
uppercase=bad)
||/ Name                 Version           Description
+++-====================-==================-
=============================================================
ii  postfix              1.1.11.0-3      A high-performance mail transport agent
un  postfix-dev          <none>            (no description available)
un  postfix-doc          <none>            (no description available)
ii  postfix-ldap         1.1.11.0-3      LDAP map support for Postfix
ii  postfix-mysql        1.1.11.0-3      MYSQL map support for Postfix
ii  postfix-pcre         1.1.11.0-3      PCRE map support for Postfix
un  postfix-snap         <none>            (no description available)
un  postfix-snap-dev     <none>            (no description available)
un  postfix-snap-doc     <none>            (no description available)
un  postfix-snap-ldap    <none>            (no description available)
un  postfix-snap-mysql   <none>            (no description available)
un  postfix-snap-pcre    <none>            (no description available)
un  postfix-snap-tls     <none>            (no description available)
un  postfix-tls          <none>            (no description available)
```

Как видите, установлена версия Postfix 1.1.11.0-3. Эта система также поддерживает карты ldap, mysql и pcre. Отсутствует пакет postfix-tls (пакет, поддерживающий TLS и SASL) либо одна или несколько экспериментальных промежуточных версий (пакет postfix-snap).

Если вы хотите попробовать промежуточную версию Postfix, выполните такую команду:

```
# apt-get install postfix-snap
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  postfix postfix-ldap postfix-mysql postfix-pcre
The following NEW packages will be installed:
  postfix-snap
0 packages upgraded, 1 newly installed, 4 to remove and 0  not upgraded.
Need to get 567kB of archives. After unpacking 47.1kB will be freed.
Do you want to continue? [Y/n]
```

Перед установкой пакета postfix-snap система управления пакетами Debian удаляет несовместимые пакеты, такие как exim и Sendmail.

Запуск и остановка Postfix

Политика Debian предписывает поставку системных демонов со сценариями запуска и останова (они хранятся в каталоге /etc/init.d). Вы можете запустить Postfix вручную при помощи сценария /etc/init.d/postfix start и остановить Postfix при помощи /etc/init.d/postfix stop.

Установка обновления

Установка обновлений в Debian также осуществляется посредством команды apt-get, например:

```
# apt-get update
Hit http://marillat.free.fr unstable/main Packages
Hit http://marillat.free.fr unstable/main Release
Hit http://smarden.org woody/unofficial Packages
Ign http://smarden.org woody/unofficial Release
Hit http://smarden.org woody/pape Packages
...
Reading Package Lists... Done
Building Dependency Tree... Done
# apt-get upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
0 packages upgraded, 0 newly installed, 0 to remove and 0  not upgraded.
```

Обновления оставляют текущую конфигурацию незатронутой, если только не должны быть выполнены абсолютно необходимые изменения, такие как добавления или изменения в файле master.cf и изменения каталога очередей.

Сборка из пакета исходных текстов Debian

Если вы хотите собрать версию Postfix из пакета исходных текстов для Debian, то следует начать с получения пакета исходных текстов:

```
# apt-get source postfix
Reading Package Lists... Done
Building Dependency Tree... Done
Need to get 2382kB of source archives.
Get:1 http://http.us.debian.org unstable/main postfix 2.1.3-1 (dsc) [832B]
Get:2 http://http.us.debian.org unstable/main postfix 2.1.3-1 (tar) [1972kB]
Get:3 http://http.us.debian.org unstable/main postfix 2.1.3-1 (diff) [409kB]
Fetched 1977kB in 9s (216kB/s)
dpkg-source: extracting postfix in postfix-2.1.3
```

Имея в системе исходный код, можно изменить параметры сборки, изменив debian/rules или другие файлы каталога debian. Когда ваша конфигурация будет вас устраивать, попробуйте собрать Postfix, используя такие команды:

```
# cd postfix-2.1.3
# dpkg-buildpackage
```

Предположим, вы попытались, но получили сообщения об ошибках, например:

```
dpkg-buildpackage: source package is postfix
dpkg-buildpackage: source version is 2.1.3-1
dpkg-buildpackage: source maintainer is LaMont Jones <lamont@debian.org>
dpkg-buildpackage: host architecture is i386
dpkg-checkbuilddeps: Unmet build dependencies: libdb4.2-dev libgdbm-dev
libldap2-dev (>= 2.1) libmysqlclient10-dev libsasl2-dev postgresql-dev
dpkg-buildpackage: Build dependencies/conflicts unsatisfied; aborting.
dpkg-buildpackage: (Use -d flag to override.)
```

Это означает, что не хватает некоторых пакетов, необходимых для сборки данного конкретного пакета исходных текстов Postfix. Устанавливаем пакеты следующим образом:

```
# apt-get install libdb4.2-dev libgdbm-dev libldap2-dev libmysqlclient10-dev
libsasl2-dev postgresql-dev
...
```

Теперь снова пытаемся собрать Postfix. Процесс сборки должен выглядеть так:

```
# dpkg-buildpackage
dpkg-buildpackage: source package is postfix
dpkg-buildpackage: source version is 2.1.3-1
dpkg-buildpackage: source maintainer is LaMont Jones <lamont@debian.org>
dpkg-buildpackage: host architecture is i386
 debian/rules clean
test -f debian/rules
dh_clean build
...
dpkg-deb: building package `postfix-doc' in `../postfix-doc_2.1.3-
1_all.deb'.
 dpkg-genchanges
dpkg-genchanges: including full source code in upload
dpkg-buildpackage: full upload (original source is included)
```

Пока все хорошо, но вы, возможно, захотите убедиться в том, что пакеты на месте:

```
# cd ..
# ls -l *.deb
-rw-r--r--  1 root  src    97592 Jul  5 13:11 postfix-dev_2.1.3-1_all.deb
-rw-r--r--  1 root  src    662758 Jul  5 13:11 postfix-doc_2.1.3-1_all.deb
-rw-r--r--  1 root  src    33644 Jul  5 13:11 postfix-ldap_2.1.3-1_i386.deb
-rw-r--r--  1 root  src    29646 Jul  5 13:11 postfix-mysql_2.1.3-1_i386.deb
-rw-r--r--  1 root  src    29430 Jul  5 13:11 postfix-pcre_2.1.3-1_i386.deb
-rw-r--r--  1 root  src    29920 Jul  5 13:11 postfix-pgsql_2.1.3-1_i386.deb
-rw-r--r--  1 root  src    140110 Jul  5 13:11 postfix-tls_2.1.3-1_i386.deb
-rw-r--r--  1 root  src    763570 Jul  5 13:11 postfix_2.1.3-1_i386.deb
```

Если все в порядке, устанавливаем пакеты, используя `dpkg -i`:

```
# dpkg -i postfix_2.1.3-1_i386.deb
```

Установка Postfix для Red Hat Linux

Вы можете установить Postfix в Red Hat Linux при помощи системы RPM (Red Hat Package Manager – менеджер пакетов Red Hat). Начиная с версии Red Hat Linux 7.3 вы можете установить Postfix даже одновременно с Sendmail. Выбор запускаемого агента передачи сообщений будет осуществляться за счет переключения между ними посредством системы `alternatives`.

Как и в случае с пакетами Debian, несмотря на то, что версии Postfix могут меняться с течением времени, действия по установке Postfix в системе Red Hat Linux останутся неизменными на долгие годы.

Получение Postfix для Red Hat Linux

По умолчанию инсталлятор Red Hat не включает в себя Postfix, но вы можете добавить его во время установки. Для того чтобы проверить, присутствует ли Postfix в вашей системе, обратитесь к менеджеру пакетов:

```
# rpm -q postfix
postfix-2.1.1-3.fc1
```

В данном случае менеджер пакетов вывел текущую установленную версию. Если же Postfix в вашей системе отсутствует, то вы получите примерно такой результат:

```
# rpm -q postfix
package postfix is not installed
```

Примечание

RPM выводит информацию только о программном обеспечении, установленном при помощи RPM. Сведения о приложениях, которые были собраны и установлены из исходных текстов, не выводятся.

Получение Postfix на CD

Самый удобный способ получения Postfix на своем компьютере состоит в копировании его с компакт-диска Red Hat на винчестер и установке RPM-пакета. Вставьте диск в дисковод и смонтируйте его такой командой:

```
# mount /dev/cdrom /mnt/cdrom/
```

Скопируйте Postfix RPM на жесткий диск:

```
# rpm -ivh /mnt/cdrom/RedHat/RPMS/postfix-XX-xx.rpm
```

Получение Postfix с сайта Red Hat

Вы также можете скачать пакет Postfix с FTP-сайта Red Hat или одного из его зеркал (перечень зеркал приведен по адресу *http://www.red-*

hat.com/download/mirror.html). После загрузки пакета на вашу машину сообщите менеджеру пакетов о том, что необходимо извлечь и установить файл:

```
# rpm -ivh ftp://USER:PASSWORD@HOST:PORT/path/to/postfix-XX-xx.rpm
```

Помните, что Red Hat обновляет пакеты не так часто, как выходят новые разработки Postfix. Если вы хотите использовать пакет Postfix с новейшими возможностями, но не хотите собирать его из исходных текстов, вам стоит обратить внимание на RPM, которые поддерживает Саймон Дж. Мадд (Simon J. Mudd).

Получение RPM Postfix Саймона Дж. Мадда

RPM Саймона обычно более актуальны, чем поставляемые с дистрибутивами Red Hat. Вы можете скачать готовые бинарные пакеты для различных платформ, включая Linux для Alpha, Sparc и IBM S390 (мейнфрейм), или получить SRPM (RPM Source Package – пакет исходных текстов RPM). SRPM обеспечивают поддержку сборки бинарных пакетов с разнообразными параметрами, приведенными на веб-сайте.

Перечень сайтов-зеркал для RPM и SRPM Саймона указан по адресу *http://postfix.wl0.org/en/mirrors/*.

Получение Postfix с веб–сайта rpmfind.net

Наконец, RPM для многих дистрибутивов можно найти на веб-сайте rpmfind.net. Укажите в своем броузере адрес *http://www.rpmfind.net*, выполните поиск по слову «postfix» и загрузите подходящий RPM.

Сборка RPM на основе SRPM

Из соображений безопасности не стоит собирать RPM из SRPM от имени пользователя `root`. Однако сборка RPM не от имени суперпользователя требует некоторых подготовительных работ. В частности, необходима некоторая структура каталогов для сборки RPM из исходных текстов или из SRPM. По умолчанию эти каталоги находятся внутри `/usr/src/redhat`.

Создание структуры каталогов и переменных окружения

При сборке RPM от имени обычного пользователя вы не можете использовать каталог по умолчанию, т. к. право на запись в каталоги по умолчанию имеет только пользователь `root`. Используйте предложенный сценарий для создания в домашнем каталоге необходимой структуры каталогов и создайте соответствующие переменные окружения для RPM:

```
#!/bin/sh
# rpmuser Build user rpmbuild environment
# Author: Tuomo Soini <http://tis.foobar.fi>
#
```

```
# create directories
for i in SOURCES SPECS BUILD SRPMS RPMS/i386 RPMS/i486 RPMS/i586 RPMS/i686 \
RPMS/athlon RPMS/noarch
do
  mkdir -p $HOME/rpm/$i
done
unset i
# set environment variables
echo "%_topdir $HOME/rpm" >> $HOME/.rpmmacros
# EOF
```

Пусть для сборки RPM вы создали пользователя `rpmuser`. После запуска `rpm_prepare.sh` в домашнем каталоге пользователя должны появиться такие каталоги и подкаталоги:

```
$ tree
.
|-- rpm
|   |-- BUILD
|   |-- RPMS
|   |   |-- athlon
|   |   |-- i386
|   |   |-- i486
|   |   |-- i586
|   |   |-- i686
|   |   `-- noarch
|   |-- SOURCES
|   |-- SPECS
|   `-- SRPMS
`-- rpm_prepare.sh
12 directories, 1 file
```

Сценарий также создает необходимые переменные окружения в файле `.rpmmacros`. Вам нужно будет настраивать окружение при каждом входе и выходе из системы пользователя, собирающего RPM (используйте команду `echo "%_topdir $HOME/rpm" >> $HOME/.rpmmacros`).

Примечание

Более подробную информацию о сборке RPM можно получить в разделе «RPM HOWTO» на веб-сайте *http://rpm.org*.

Сборка и установка RPM

Вы не можете определить по исходным текстам параметры, с которыми могут быть собраны бинарные дистрибутивы. Можно провести обходной маневр, установив пакет исходных текстов в новый каталог сборки RPM при помощи `rpm -ivh postfix-XX-xx.src.rpm`, а затем посмотреть на сценарий, используемый для сборки файлов спецификации:

```
$ less rpm/SOURCES/make-postfix.spec
...
# The following external variables if set to 1 affect the behaviour
#
# POSTFIX_MYSQL         include support for MySQL's MySQL packages
# POSTFIX_MYSQL_REDHAT  include support for RedHat's mysql packages
# POSTFIX_MYSQL_PATHS   include support for locally installed mysql binary,
#                       providing the colon seperated include and
#                       library paths ( /usr/include/mysql:/usr/lib/mysql )
# POSTFIX_MYSQL_QUERY   include support for writing full select statements
#                       in mysql maps
# POSTFIX_LDAP          include support for openldap packages
# POSTFIX_PCRE          include support for pcre maps
# POSTFIX_PGSQL         include support for PostGres database
# POSTFIX_SASL          include support for SASL (1, 2 or 0 to disable)
# POSTFIX_TLS           include support for TLS
# POSTFIX_IPV6          include support for IPv6
# POSTFIX_VDA           include support for Virtual Delivery Agent
...
# To rebuild the spec file, set the appropriate environment
# variables and do the following:
#
# cd `rpm --eval '%{_sourcedir}'`
# export POSTFIX_MYSQL=1        # for example
# sh make-postfix.spec
# cd `rpm --eval '%{_specdir}'`
# rpmbuild -ba postfix.spec
```

Следуйте инструкциям данного сценария для задания необходимых переменных окружения и создания файла спецификации. Когда файл спецификации будет готов, собираем RPM такой командой:

```
$ rpmbuild -ba rpm/SPECS/postfix.spec
```

После ее успешного выполнения переключаемся на пользователя `root` и устанавливаем Postfix:

```
# rpm -ivh /path/to/postfix-XX-xx.rpm
```

Осталось лишь сообщить серверу Red Hat о том, что следует использовать Postfix в качестве MTA.

Переключение на Postfix

По умолчанию агентом передачи сообщений для серверов Red Hat является Sendmail. Изменить эту настройку можно, используя порт `alternatives` для переключения на Postfix.

Примечание

Начиная с версии Red Hat 7.3 дистрибутив поставляется с портом Debian, который называется `alternatives`. Эта команда делает возможной одновременную

установку в одной системе нескольких программ, выполняющих одинаковые или похожие функции.

Переключитесь на пользователя `root`, вызовите `alternatives --config mta` и укажите Postfix в качестве MTA по умолчанию:

```
# alternatives --config mta
There are 2 programs which provide 'mta'.
  Selection    Command
-----------------------------------------------
*+ 1           /usr/sbin/sendmail.sendmail
   2           /usr/sbin/sendmail.postfix
Enter to keep the default[*], or type selection number: 2
```

Будучи агентом передачи сообщений по умолчанию, Postfix запустится системой Red Hat в момент загрузки. Чтобы проверить уровни запуска, выполните команду `chkconfig`:

```
# chkconfig --list postfix
postfix         0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

Удаление агента передачи сообщений Sendmail

После установки Postfix нет смысла в сохранении программы Sendmail, так что удаляем ее:

```
# rpm -e sendmail
```

Запуск и остановка Postfix в Red Hat Linux

RPM Red Hat Linux обычно поставляются со сценариями запуска и останова, которые хранятся в каталоге `/etc/init.d`. Вы можете запустить Postfix при помощи `/etc/init.d/postfix start` и остановить его, используя `/etc/init.d/postfix stop`.

B

Устранение неисправностей Postfix

В этом приложении собраны советы по устранению разнообразных проблем с Postfix, включая настройку системного регистратора, проблемы с конфигурированием и сетевыми настройками, а также общие системные вопросы. Как обычно, когда мы имеем дело с неисправностью, необходимо сначала понять, где именно возникла проблема, а потом уже пытаться с ней разобраться. Для Postfix, включающего в себя несколько отдельных подсистем, это особенно важно.

Проблемы при запуске Postfix и просмотре журнала

Если Postfix не занимается обработкой ваших сообщений, наиболее «очевидной» причиной является то, что Postfix просто не работает. Postfix должен работать даже в том случае, если вы только отправляете почту посредством команды `sendmail`. Проще всего определить, работает ли Postfix, выполнив команду `postfix start`:

```
# postfix start
postfix/postfix-script: starting the Postfix mail system
```

Если вы видите такое сообщение, это означает, что Postfix не работал, т. к. команда пытается его запустить. Если же Postfix уже работает в вашей системе, то сообщение будет таким:

```
# postfix start
postfix/postfix-script: fatal: the Postfix mail system is already running
```

В журнале при этом должны появиться подобные сообщения:

```
Jul  5 22:49:29 mail postfix/postfix-script: starting the Postfix mail system
Jul  5 22:49:29 mail postfix/master[14835]: daemon started -- version 2.1.3
```

Если вы не видите таких сообщений, срочно проверьте конфигурацию системы syslog. Необходимо убедиться в том, она записывает файлы mail.*; полные журналы необходимы для выявления любых неполадок.

Мы рекомендуем объединить все журнальные записи почтовой службы (или той службы, для которой настроен Postfix) в один журнальный файл. Некоторые версии (например, для Debian GNU/Linux) делят журнал на несколько файлов, но это делает чтение журнала весьма утомительным занятием. Для обеспечения простоты просмотра убедитесь в том, что в файле /etc/syslog.conf присутствует такая запись:

```
# (- Log all the mail messages to one place.)
mail.*                                                  -/var/log/maillog
```

Предположим, вы видите сообщения в командной строке и в журнале, но все еще сомневаетесь в том, что Postfix действительно работает. Иногда не вредно побыть параноиком, – дело в том, что Postfix может запуститься и сразу же аварийно завершиться в случае серьезной проблемы. Используйте команды ps и grep для того, чтобы посмотреть, действительно ли работают демоны Postfix. Если они работают, то команда выполнится следующим образом:

```
# ps aux|grep postfix
root      5035  0.0  0.4  2476 1100 ?      S    09:29   0:00 /usr/lib
                                                          /postfix/master
postfix   5036  0.0  0.3  2404  936 ?      S    09:29   0:00 pickup
                                                          -l -t fifo -u -c
postfix   5037  0.0  0.3  2440  964 ?      S    09:29   0:00 qmgr -l -n
                                                       qmgr -t fifo -u -c
```

Мы видим, что Postfix-демон master работает как root, а диспетчер очередей qmgr и служба pickup – как пользователь postfix, так что система запущена и работает.

Запуск Postfix может оказаться неудачным по нескольким причинам, но чаще всего выясняется, что демон Postfix не может найти совместно используемую библиотеку. Чтобы решить этот вопрос, сначала найдем используемые Postfix каталоги при помощи такой команды:

```
# postconf | grep directory
command_directory = /usr/sbin
config_directory = /etc/postfix
daemon_directory = /usr/lib/postfix
mail_spool_directory = /var/mail
manpage_directory = /usr/local/man
process_id_directory = pid
program_directory = /usr/sbin
queue_directory = /var/spool/postfix
readme_directory = no
require_home_directory = no
sample_directory = /etc/postfix
tls_random_exchange_name = ${config_directory}/prng_exch
```

Ищем путь к каталогу daemon_directory, находим его и переходим в этот каталог. Смотрим на его содержимое:

```
# cd /usr/lib/postfix
# ls -l
total 384
-rwxr-xr-x    1 root     root        16588 Sep 12 18:50 bounce
-rwxr-xr-x    1 root     root        22684 Sep 12 18:50 cleanup
-rwxr-xr-x    1 root     root         4248 Sep 12 18:50 error
-rwxr-xr-x    1 root     root        10344 Sep 12 18:50 flush
-rwxr-xr-x    1 root     root        20508 Sep 12 18:50 lmtp
-rwxr-xr-x    1 root     root        31956 Sep 12 18:50 local
-rwxr-xr-x    1 root     root        22388 Sep 12 18:50 master
-rwxr-xr-x    1 root     root        33084 Sep 12 18:50 nqmgr
-rwxr-xr-x    1 root     root         7248 Sep 12 18:50 pickup
-rwxr-xr-x    1 root     root        10496 Sep 12 18:50 pipe
-rwxr-xr-x    1 root     root        27424 Sep 12 18:50 qmgr
-rwxr-xr-x    1 root     root        12160 Sep 12 18:50 qmqpd
-rwxr-xr-x    1 root     root         7456 Sep 12 18:50 showq
-rwxr-xr-x    1 root     root        25000 Sep 12 18:50 smtp
-rwxr-xr-x    1 root     root        44712 Sep 12 18:50 smtpd
-rwxr-xr-x    1 root     root         5612 Sep 12 18:50 spawn
-rwxr-xr-x    1 root     root        10284 Sep 12 18:50 trivial-rewrite
-rwxr-xr-x    1 root     root        10400 Sep 12 18:50 virtual
```

Демоны Postfix из столбца command файла /etc/postfix/master.cf должны присутствовать в этом каталоге. Вы можете исследовать зависимости совместно используемой библиотеки одной программы при помощи команды ldd (работает в Linux, Solaris и других распространенных вариантах UNIX; в других системах может использоваться другая команда):

```
# ldd `postconf -h daemon_directory`/smtpd
        libpostfix-master.so.1 => /usr/lib/libpostfix-master.so.1 (0x4001d000)
        libpostfix-global.so.1 => /usr/lib/libpostfix-global.so.1 (0x40023000)
        libpostfix-dns.so.1 => /usr/lib/libpostfix-dns.so.1 (0x4003c000)
        libpostfix-util.so.1 => /usr/lib/libpostfix-util.so.1 (0x40040000)
        libdb3.so.3 => /usr/lib/libdb3.so.3 (0x4005d000)
        libnsl.so.1 => /lib/libnsl.so.1 (0x40105000)
        libresolv.so.2 => /lib/libresolv.so.2 (0x40119000)
        libgdbm.so.1 => /usr/lib/libgdbm.so.1 (0x4012a000)
        libc.so.6 => /lib/libc.so.6 (0x40130000)
        libdl.so.2 => /lib/libdl.so.2 (0x4024b000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Приведенный выше пример показывает, что с демоном smtpd все в порядке, т. к. для каждой зависимой библиотеки имеется реальный файл. Однако вы могли бы получить и менее утешительные данные:

```
# ldd `postconf -h daemon_directory`/smtpd
        libpostfix-master.so.1 => /usr/lib/libpostfix-master.so.1 (0x4001d000)
        libpostfix-global.so.1 => /usr/lib/libpostfix-global.so.1 (0x40023000)
```

```
libpostfix-dns.so.1 => /usr/lib/libpostfix-dns.so.1 (0x4003c000)
libpostfix-util.so.1 => /usr/lib/libpostfix-util.so.1 (0x40040000)
libdb3.so.3 => not found
libnsl.so.1 => /lib/libnsl.so.1 (0x4005d000)
libresolv.so.2 => /lib/libresolv.so.2 (0x40071000)
libgdbm.so.1 => /usr/lib/libgdbm.so.1 (0x40082000)
libc.so.6 => /lib/libc.so.6 (0x40088000)
libdl.so.2 => /lib/libdl.so.2 (0x401a3000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

В этом случае `libdb3.so.3` отсутствует. Программа, которая не может найти все свои совместно используемые библиотеки, не будет работать. Если вы работаете с Linux и устанавливаете пакет Postfix, предназначенный для другого дистрибутива (или другой версии вашего дистрибутива), то можете столкнуться с этой проблемой только в процессе работы, и вам придется принять решение.

Наилучшим решением является поиск пакета Postfix, подходящего для вашего дистрибутива, или сборка Postfix из исходных текстов (см. приложение А). Однако если вы обязательно хотите попытаться работать с тем, что у вас уже есть, попробуйте разыскать `libdb3.so.3` следующим образом:

```
# find / -name libdb3.so.3
/usr/lib/libdb3.so.3
```

Эта команда, вероятно, будет выполняться очень долго (она просматривает всю вашу файловую систему), но если вам удастся найти библиотеку, то вы сможете добавить путь к ее каталогу в файл `/etc/ld.so.conf` и запустить `ldconfig`. Конечно, это может привести к конфликтам библиотек и символов. С совместно используемыми библиотеками следует обращаться очень аккуратно, и лучше вообще этого не делать, если вы не до конца понимаете, что делаете.

Команда `find` не обязательно вам поможет, ведь библиотеки может и не быть в вашей системе. В конце концов, если вы не сможете решить эту проблему, вам нужно будет сделать выбор. Если поиск пакета Postfix не рассматривается как возможное решение, а сборка из исходных текстов представляется устрашающей, то следует подумать о смене операционной системы или дистрибутива.

Подключение к Postfix

Если после успешного запуска Postfix ведет себя не так, как хотелось бы, проверьте, принимает ли ваш сервер соединения с портом 25. Подключаемся к SMTP-порту. Вот как выглядит успешное соединение:

```
# telnet localhost 25
220 mail.example.com ESMTP Postfix
QUIT
221 Bye
```

Вы можете подключиться к интерфейсу обратной связи, но это не означает, что такую же возможность имеют все пользователи Интернета. Пусть адрес вашего компьютера 10.1.2.233; попробуем подключиться к этому адресу:

```
# telnet 10.1.2.233 25
220 mail.example.com ESMTP Postfix
QUIT
221 Bye
```

Если не получится, то первое, что следует проверить, – установлен ли параметр inet_interfaces в файле main.cf и не исключен ли из него IP-адрес, к которому мы пытаемся подключиться. По умолчанию должны прослушиваться все интерфейсы.

Проверка сети

Если с конфигурацией Postfix все хорошо и Postfix перезапущен, но установить соединение так и не удается, проверьте межсетевой экран или настройку фильтрации IP в своей сети. Возможно, ваша система блокирует порт по умолчанию. Придется просмотреть множество мест, т. к. фильтрация IP может производиться через сценарий межсетевого экрана операционной системы (например, вызывающего iptables или ipf), а может выполняться вне данного компьютера выделенными межсетевыми экранами и маршрутизаторами.

Следует проверить все и везде.

Если конфигурация все еще представляется корректной, выходим за рамки локальной сети. Возможно, ваш интернет-провайдер блокирует входящий трафик для вашего порта 25 (и попутно исходящий трафик для порта 25 другой машины). Если окажется, что интернет-провайдер отклоняет входящий трафик и отказывается открыть порт, то останется лишь сменить провайдера.

Чтобы проверить, можно ли связаться с вашим компьютером извне, выполняем такую команду:

```
# telnet relay-test.mail-abuse.org
```

Когда вы выполняете такое соединение, relay-test.mail-abuse.org проверяет пересылку для машины, осуществляющей соединение. Если ваш интернет-провайдер (или ваш собственный межсетевой экран) не блокирует входящие соединения с портом 25, то в файле журнала должно быть совсем немного сообщений.

Если вам не удалось подключиться к предыдущему хосту, возможно, у вас проблема с разрешением имен. Проверим это предположение следующей командой:

```
# host relay-test.mail-abuse.org
relay-test.mail-abuse.org is an alias for cygnus.mail-abuse.org.
cygnus.mail-abuse.org has address 168.61.4.13
```

Вы должны увидеть IP-адрес, как в только что приведенном выводе. Если же он отсутствует, вы не можете разрешать имена хостов. Вероятно, присутствует ошибка в файле /etc/resolv.conf или /etc/nsswitch.conf (или в них обоих). Возможна и более ужасная ситуация: компьютер даже не может подключиться к Интернету. Успешная проверка должна выглядеть так (используйте Ctrl-C для остановки теста):

```
# ping 134.169.9.107
PING 134.169.9.107 (134.169.9.107): 56 data bytes
64 bytes from 134.169.9.107: icmp_seq=0 ttl=54 time=12.1 ms
64 bytes from 134.169.9.107: icmp_seq=1 ttl=54 time=12.1 ms
64 bytes from 134.169.9.107: icmp_seq=2 ttl=54 time=12.1 ms
--- 134.169.9.107 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 12.1/12.1/12.1 ms
```

Проверка прослушивающего процесса

Если Postfix работает и сеть проверена, но тестовые соединения все еще не работают, то используйте команду netstat для проверки того, действительно ли Postfix прослушивает порт 25:

```
# netstat -t -a | grep LISTEN
tcp        0      0 *:printer               *:*                     LISTEN
tcp        0      0 localhost:domain        *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 *:smtp                  *:*                     LISTEN
```

Вывод показывает, что есть серверы, прослушивающие порты принтера, домена, SSH и SMTP (числовые эквиваленты имен можно найти в /etc/services). Какой-то сервер прослушивает порт 25 (smtp), но Postfix ли это?

Ответ может дать команда lsof. Если в выводе будет присутствовать прослушивающая порт 25 программа sendmail, значит, ваша старушка sendmail еще активна:

```
# lsof -i tcp:25
COMMAND    PID USER   FD   TYPE DEVICE SIZE NODE NAME
sendmail 25976 root    4u  IPv4 228618       TCP mail.example.com:smtp
(LISTEN)
```

Уничтожьте этот процесс и отредактируйте файлы запуска системы так, чтобы он не появился после перезагрузки системы (если возможно, полностью удалите Sendmail из системы, ведь, если помните, вы собирались пользоваться Postfix, не так ли?).

Примечание

Команда lsof – это чрезвычайно мощное средство, которое может показать все открытые файлы (и процессы, использующие файлы), но она очень зависит от вашего ядра. Убедитесь в том, что ваша версия lsof актуальна и соответствует вашему текущему ядру. Устаревшая lsof вообще не возвращает данных для

интернет-соединений. Если вы не уверены в том, что она работает, запустите `lsof -i`.

Если вы используете Postfix, то вывод `lsof` должен содержать сведения о Postfix-демоне `master`, прослушивающем порт 25:

```
# lsof -i tcp:25
COMMAND   PID USER   FD   TYPE DEVICE SIZE NODE NAME
master  26079 root   11u  IPv4 228828       TCP *:smtp (LISTEN)
```

Использование сервером Postfix ваших параметров конфигурации

Файл `main.cf` длинен и труден для чтения. Какой-то параметр конфигурации может встретиться дважды, а где-то среди комментариев может затеряться опечатка. Выполните команду `postconf` для вывода используемой Postfix конфигурации. Разницу можно увидеть просто по количеству строк:

```
# cd /etc/postfix
# wc -l main.cf
# postconf -n | wc -l
```

Вывод `postconf -n` перечисляет все настройки из файла `main.cf`, включая и те параметры, значения которых совпадают с заданными по умолчанию. При изменении `main.cf` необходимо проверить, видны ли эти изменения Postfix, выполнив команду `postconf`.

Вы можете предпочесть использование команды `postconf -e` *`параметр=значение`* для программной установки параметра файла `main.cf` в *`значение`*. Эта небольшая хитрость позволяет выполнять изменения конфигурации Postfix при помощи сценариев командного интерпретатора или заданий `cron`.

Если вы столкнулись с проблемой конфигурирования, вам стоит обратиться к странице руководства postconf(5).

Оповещение о проблемах Postfix

Когда вы впервые начинаете работать с Postfix, может быть нелегко оценить, какую информацию следует сообщить `postfix-users@postfix.org`. Программа `postfinger` (созданная Саймоном Дж. Маддом) извлекает большую часть необходимой информации. Для того чтобы посмотреть, что она делает, отправьте самому себе по почте выходные данные `postfinger` вместе со своими собственными вопросами:

```
# postfinger | /usr/sbin/sendmail youraddress@your.domain
```

Естественно, при этом предполагается, что исходящая почта в вашей системе работает. Если ничего другого не получится, вы можете передать вывод в другую систему.

Если ваша проблема связана с SMTP-AUTH и соответственно с SASL, используйте сценарий Патрика `saslfinger` (`saslfinger` – это чрезвычайно полезный сценарий, который пытается помочь вам отладить настройки SMTP AUTH). Программа `saslfinger` собирает разнообразную информацию о Cyrus SASL и Postfix из вашей системы и отправляет ее на стандартный выход `stdout`. Для того чтобы посмотреть, что она делает, отправьте самому себе по почте выходные данные `saslfinger` вместе со своими собственными вопросами:

```
# saslfinger -s | /usr/sbin/sendmail youraddress@your.domain
```

Примечание

Программа `postfinger` входит в дистрибутив исходных текстов Postfix начиная с версии 2.1. Вы также можете получить ее здесь: *ftp://ftp.wl0.org/sources/postfinger*.

Программа `saslfinger` не входит в дистрибутив исходных текстов Postfix. Вы найдете ее по адресу *http://postfix.state-of-mind.de/patrick.koetter/saslfinger*.

Увеличение объема журнальной информации

Если вы испытываете трудности с определенными этапами вашей установки Postfix, то вы можете увеличить объем записываемой в журнал информации для отдельных демонов. Для этого добавьте `-v` в конец конфигурационной записи демона в файле `/etc/postfix/master.cf`, как это сделано в нашем примере для демона `smtpd`:

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (50)
# ==========================================================================
smtp      inet  n       -       -       -       -       smtpd -v
```

Для того чтобы изменения вступили в силу, перезагрузите конфигурацию Postfix. Теперь демон должен стать более многословным. Если информации все еще недостаточно, вы можете снова добавить в запись `-v`, тогда выходных данных будет еще больше. Не забудьте уменьшить их объем до обычного после завершения отладки, т. к. подробное журналирование генерирует массу строк в файле журнала, снижая общую производительность системы.

Журналирование в зависимости от клиента

Если ваш почтовый сервер сильно загружен, вследствие чего увеличение подробности журналирования для всех клиентов приведет к тому, что вы просто утонете в море выводимой информации, вы можете увеличить уровень журналирования избирательно (для некоторых клиентов) при помощи параметра `debug_peer_list`. Следующий пример показывает, как сделать журналирование `smtpd` более подробным только для клиентов с адресами `10.0.0.1` и `10.0.0.4`:

```
debug_peer_list = 10.0.0.1, 10.0.0.4
```

В качестве значения этого параметра вы можете указать один или несколько хостов, доменов, адресов или сетей. Для того чтобы изменение вступило в силу незамедлительно, выполните команду `postfix reload`.

Журналирование и демон qmgr

Часто возникает проблема отсутствия вывода в журнал данных от процесса `qmgr`. Диспетчер очередей должен передавать в журнал такие записи:

```
Aug  5 17:05:26 hostname postfix/qmgr[308]: A44F828C71:
  from=<bamm@example.com>, size=153136, nrcpt=1
(queue active)
```

Если подобная информация в журнале отсутствует, это может быть вызвано двумя причинами:

Проблемы libc

Реализация libc повреждена (клиент syslog не осуществляет повторное подключение при перезапуске сервера syslogd). В этом случае следует обновить libc.

Демон qmgr запущен из chroot-окружения

Postfix-процесс `qmgr` запущен из chroot (см. `master.cf`), но внутри chroot-окружения нет сокета syslog. Обратитесь к странице руководства syslog(8), чтобы узнать, как задавать дополнительные сокеты и определить сокет для chroot-окружения Postfix.

Другие ошибки конфигурирования

Существуют три постоянно встречающиеся ошибки:

Проблемы при открытии файлов

Если у вас возникает проблема с открытием файла, про который известно, что он существует, проверьте, не определен ли он как карта в файле конфигурации (например, он может начинаться с `hash:` префикс). Если это так, выполните команду `postmap` для файла открытого теста, содержащего данные карты, чтобы создать индексированную версию.

Также проверьте корректность привилегий и права собственности. Не забудьте о правах на исполнение для каталога и всех вышестоящих каталогов.

Проблемы с правами доступа

Если у вас возникают проблемы с правами доступа, то посмотрите, может ли Postfix исправить их автоматически, используя команду `post-install`:

```
# /etc/postfix/post-install set-permissions upgrade-configuration
```

Эта команда (в дополнение к решению вопросов с правами доступа) вносит соответствующие изменения в файлы `main.cf` и `master.cf`, так что перед ее выполнением у вас может возникнуть желание сохранить резервную копию своей конфигурации.

Комментарии

Любая строка, первый непустой символ которой является знаком «решетки» (#), – это комментарий. Postfix не принимает какой-либо другой синтаксис для комментариев. Если команда `postconf` выводит какой-то незнакомый параметр, дело может быть в том, что вы где-то не там поставили знак # в файле конфигурации.

Лабиринты chroot-окружения

Слишком часто возможность работать в chroot-окружении вызывает удивительные проблемы. По умолчанию версия Postfix *никогда* не работает из chroot-окружения. Слишком много всего может пойти не так, поэтому Витсе благоразумно решил *не* использовать chroot по умолчанию. К сожалению, сотрудники, занимающиеся поддержкой других пакетов, иногда просто помешаны на обеспечении безопасности.

Демоны Postfix открывают все свои карты до входа в «песочницу». Однако системные файлы, которые необходимы для поиска DNS, других просмотров, связанных с хостами, просмотров сетевых служб, поиска часовых поясов, и прочие компоненты, встречающиеся в библиотеках, использованных при сборке Postfix, должны находиться внутри chroot-окружения. Персонал, отвечающий за поддержку пакета, должен предоставить сценарии, которые копируют необходимые файлы из их исходных каталогов файловой системы в «песочницу». Обычно необходимы `/etc/resolv.conf` и `/etc/nsswitch.conf`. Дистрибутив исходных текстов Postfix включает в себя подкаталог `examples/chroot-setup` со сценариями для создания chroot-окружения в разных операционных системах. Матиас Эндри (Matthias Andree) написал сценарий `LINUX2` для Linux.

Теоретически в пакет должен быть включен механизм корректного создания «песочницы» для конкретной операционной системы или дистрибутива. Однако для начинающих это может быть слишком сложно. Вместо того чтобы исправлять ваше chroot-окружение, о существовании которого вы даже и не подозревали, вероятно, следовало бы вывести из chroot все демоны Postfix на тот период, пока Postfix не работает в полную силу. Отредактируйте файл `/etc/postfix/master.cf` и посмотрите на каждую запись:

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (50)
# ==========================================================================
smtp      inet  n       -       -       -       -       smtpd
```

Дефис в столбце chroot указывает на то, что `smtpd` *работает* из chroot-окружения. Замените его на значение `n` и перезагрузите конфигурацию Postfix. Не забывайте и о том, что не каждый демон Postfix может работать из chroot-окружения. С большинством демонов проблем не возникнет, но вы вполне можете столкнуться с разными странностями, если попытаетесь запустить в «песочнице» демон `pipe`, `local` или `virtual`.

Подробную информацию о chroot-окружении вы найдете в главе 20.

Решение проблем файловой системы

Большинство современных UNIX-систем имеют файловые системы с журналированием, что, однако, не может защитить от случайного повреждения файловой системы, особенно если у вас не очень хорошее оборудование или вы используете новомодную, но не полностью отлаженную файловую систему. Если происходят очень странные вещи (например, каталоги превращаются в файлы), необходимо срочно перезагрузить систему с полным принудительным выполнением `fsck` для всех дисков, используя такую последовательность команд:

```
# touch /forcefsck
# sync
# reboot
```

Да, время работы вашей машины без перезапуска не будет таким симпатично большим, как хотелось бы, и пользователи будут недовольны, но невозможно решить проблемы файловой системы без принудительного `fsck`. В подобных случаях мы успешно раздражали многих пользователей Red Hat и Debian.

Проблемы с библиотеками

Postfix широко применяет совместно используемые библиотеки, такие как BerkeleyDB. Данная конкретная библиотека вызывает множество проблем, т. к. у нее существует огромное количество разных версий с разными форматами хранимых на диске данных. Все составляющие почтовой службы, такие как Postfix, POP-before-SMTP, dracd, postgrey и другие средства, которые обращаются к картам типа `hash:` или `btree:` и изменяют их, должны использовать совместимые библиотеки BerkeleyDB.

Более того, форматы файлов различных версий BerkeleyDB несовместимы, т. е. может оказаться, что приложение не сможет прочитать

карту, записанную другим приложением, использующим другую версию BerkeleyDB.

Для проверки библиотек, которые использует Postfix, выполняем команду `ldd`, как было описано в разделе «Проблемы при запуске Postfix и просмотре журнала»:

```
# ldd `postconf -h daemon_directory`/smtpd
        libpcre.so.0 => /usr/local/lib/libpcre.so.0 (0x4001d000)
        libdb-3.1.so => /lib/libdb-3.1.so (0x40028000)
        libnsl.so.1 => /lib/libnsl.so.1 (0x400a1000)
        libresolv.so.2 => /lib/libresolv.so.2 (0x400b8000)
        libc.so.6 => /lib/libc.so.6 (0x400ca000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

В предыдущем выводе видно, что `smtpd` связан с версией BerkeleyDB-3.1.*x*, так что все остальные программы, которым необходимы карты Postfix `hash:` и `btree:`, должны использовать ту же самую версию BerkeleyDB (или по крайней мере версию с таким же форматом хранения на диске).

Несогласованность демонов

Если попытка обновления версии Postfix заканчивается неудачей или вы проводите ее нестандартным образом (например, устанавливая из исходных текстов поверх RPM, вместо того чтобы сначала удалить старую версию), то могут произойти странные явления. У вас могут оказаться демоны из разных версий Postfix.

Для определения того, к какой версии Postfix принадлежит демон, используйте `strings` для всех двоичных файлов демонов:

```
# strings /usr/libexec/postfix/smtpd | grep 2003
2.0.13-20030706
20030706
# strings /usr/libexec/postfix/cleanup | grep 2003
2.0.13-20030706
20030706
```

В данном случае версии (представленные датами) действительно совпадают. В нашем примере использован 2003 год, если же вы пользуетесь версиями другого года выпуска, то укажите его в команде `grep`.

Версия tlsmgr

Одна из часто встречающихся проблем, вызванных смешением демонов из несовместимых версий Postfix, связана с демоном `tlsmgr`. Нагрузка кажется чрезвычайно большой, идентификаторы процессов постоянно увеличиваются, но ничего не работает, почтовый трафик невелик, как и очереди.

Возможно, вы обновляли версию Postfix и оставили старый демон `tlsmgr` и старый файл `master.cf`, управляющий `tlsmgr`.

Проблема в том, что новый Postfix запускает старый `tlsmgr`, но этот демон сразу же завершает свою работу со статусом 0, т. к. он не может работать с новой версией Postfix. Postfix ничего не пишет в журнал, т. к. код завершения 0 не является аномалией. Однако Postfix незамедлительно снова порождает `tlsmgr`, и процесс повторяется.

Если это случится у вас, закомментируйте строку `tlsmgr` в файле `master.cf`, а затем перезагрузите конфигурацию Postfix для восстановления нормальной работы. Затем вы можете установить работающее обновление с совместимой версией `tlsmgr`.

Нагрузочное тестирование Postfix

Для того чтобы узнать, сколько почтового трафика может обработать ваша версия Postfix, необходимо нагрузочное тестирование. Для формирования достаточной нагрузки вам необходим быстрый генератор почты. В комплект поставки Postfix именно для этих целей включены две тестирующие программы: `smtp-source` и `smtp-sink`.

`smtp-source`

Эта программа подключается к хосту через порт TCP (по умолчанию это порт 25) и отправляет одно или несколько сообщений последовательно или параллельно. Программа может общаться как по SMTP (по умолчанию), так и по LMTP. Она предназначена для измерения производительности сервера.

`smtp-sink`

Этот тестовый сервер прослушивает указанный хост (или адрес) и порт. Программа получает сообщения от сети и удаляет их. Эту программу можно использовать для измерения производительности клиента и сервера.

Начнем нагрузочное тестирование нашей версии Postfix с программы `smtp-source`. В следующем примере будет отправлено 100 сообщений по 5 Кбайт каждое в 20 параллельных сеансах серверу Postfix, работающему на хосте `localhost` (порт 25). Нам интересно, сколько времени все это займет, так что используем еще команду `time`:

```
$ time ./smtp-source -s 20 ❶ -l 5120 ❷ -m 100 ❸ -c ❹ \
  -f sender@example.com ❺ -t recipient@example.com ❻ localhost:25 ❼
100
real    0m4.294s
user    0m0.060s
sys     0m0.030s
```

❶ 20 параллельных сеансов.

❷ 5 Кбайт – размер сообщений.

❸ Всего 100 сообщений.

❹ Выводить счетчик.

❺ Отправитель конверта.

❻ Получатель конверта.

❼ SMTP-сервер назначения.

В нашем примере «вливание» почты заняло 4,294 секунды. Вам также необходимы сведения о том, сколько заняла реальная доставка. Эти данные вы найдете в журнале, там же вы сможете убедиться, что каждое последнее сообщение, поступившее для `recipient@example.com`, было получено.

Теперь обратимся к программе `smtp-sink`, чтобы определить, сколько сообщений в секунду от вашего ужасного программного обеспечения, отправляющего массу почты, может обработать ваш сервер. Postfix должен обрабатывать каждое исходящее сообщение, даже если сервер на другом конце соединения уничтожит его (так что протестировать чистую производительность вашей массовой рассылки можно только в случае, если ваш сервер подключен непосредственно к `smtp-sink`).

В следующем примере создается прослушиватель SMTP для порта 25 хоста `localhost`:

```
$ ./smtp-sink -c localhost:25 1000
```

Теперь можно проводить наши клиентские тесты.

Если вы хотите знать, какие накладные расходы возникают из-за использования сети, и провести управляемый эксперимент, чтобы определить, какова теоретическая максимальная пропускная способность вашего почтового сервера, то можно организовать взаимодействие `smtp-source` и `smtp-sink`. Откройте два окна и в первом запустите модель сервера:

```
# ./smtp-sink -c localhost:25 1000
100
```

Затем начинайте отправлять сообщения на этот сервер посредством `smtp-source`, открытой во втором окне:

```
$ time ./smtp-source -s 20 -l 5120 -m 100 -c \
  -f sender@example.com -t recipient@example.com localhost:25
100
real    0m0.239s
user    0m0.000s
sys     0m0.040s
```

Выходные данные показывают, что `smtp-sink` принимает сообщения гораздо быстрее, чем Postfix. Прием сообщений занял всего 0,239 секунды, что в 18 раз быстрее, чем у Postfix. Разве не замечательно было бы, если бы вы могли таким же образом выбрасывать всю входящую почту?

Дисковый ввод-вывод

При проведении нагрузочного тестирования может обнаружиться неуместно большая средняя нагрузка. Если вы не используете фильтрацию содержимого, то из-за ограниченности возможностей ввода-вывода Postfix подсистема ввода-вывода может быть переполнена.

Если команда `top` показывает высокую загрузку, такую как `10.7`, но ни один из ваших процессов на самом деле не использует процессор, то, возможно, такая нагрузка вызвана тем, что ядро использует большую часть мощности процессора для ввода-вывода, не позволяя работать процессам. Причина того, что ядро выполняет так много операций ввода-вывода, заключается в том, что их запросили слишком много процессов (которые теперь ожидают их выполнения).

Ядра Linux 2.6 поддерживают вывод статуса iowait в команде `top`. Для того чтобы проверить состояние дел в ядрах версии 2.4.*x* (в которых нет специального средства вывода статуса iowait), можно добавить модуль ядра. Оливер Велниц (Oliver Wellnitz) написал именно такой модуль, и вы можете найти его по адресу *ftp://ftp.ibr.cs.tu-bs.de/os/linux/people/wellnitz/programming*. Этот модуль вычисляет загрузку другим способом и выводит результат в файловую систему `/proc`, например, так:

```
# cat /proc/loadavg-io
rq 0.30 0.23 0.14
io 0.08 0.31 0.27
```

В данном примере `rq` – это количество процессов, которые находятся в состоянии `TASK_RUNNING`, а `io` – количество процессов, которые находятся в состоянии `TASK_UNINTERRUPTIBLE` (ожидание ввода-вывода). Их сумму ядро обычно и называет нагрузкой (`load`).

Если у вас возникли подобные проблемы, то вам нужны более быстрые диски или более серьезное решение, такое как SSD (solid state disk – твердотельный диск, RAM-диск с автономным питанием) или же RAID с зеркалированием и расслоением для каталога очередей. Более подробную информацию вы найдете в подразделе «Очередь Incoming» раздела «Поиск узких мест» главы 22.

Можно предложить еще одно решение, которое может помочь, но не всегда, – избавиться от синхронных обновлений каталога очередей. Если вы используете файловую систему ext2 или ext3, то попробуйте такую команду:

```
# chattr -R -S /var/spool/postfix/
```

В текущих версиях Postfix такая настройка используется по умолчанию.

Слишком много соединений

При настройке почтового сервера у вас может возникнуть желание решить все проблемы сразу. Если вы хотите, чтобы ваша система Postfix была устойчивой, то изменяйте каждый раз что-то одно, особенно если планируете использовать LDAP или SQL. Попробуйте такую последовательность действий:

1. Постройте систему *без* карт LDAP (т. е. используя карты `hash`, `btree` или `dbm`).
2. Используйте соответствующие команды `ldapsearch` для извлечения всех необходимых данных с вашего сервера LDAP. С помощью языка сценариев, например Perl или Python, переформатируйте данные так, чтобы ими можно было заполнить файл карты Postfix.
3. Когда Postfix будет корректно работать без LDAP, заменяйте карты соответствующими LDAP-картами по одной. Проверяйте каждую карту LDAP от имени пользователя `postfix`:

   ```
   $ postmap -q - ldap:mapname < keyfile
   ```

   Файл `keyfile` содержит список адресов (ключей), к которым следует обращаться. Если карта возвращает разумные данные, то изменяем соответствующий параметр конфигурации `_maps`, с тем чтобы Postfix использовал карту LDAP.
4. Для уменьшения количества открытых таблиц поиска сделайте так, чтобы одну открытую таблицу совместно использовали несколько процессов Postfix (при помощи демона `proxymap`, см. раздел «Демоны Postfix» главы 5).

C

Справочник подсетей в нотации CIDR и кодов отклика SMTP

В первом разделе приложения рассмотрена нотация Classless Inter Domain Routing (CIDR – бесклассовая междоменная маршрутизация), которую Postfix может использовать в картах типа `cidr:` и при задании параметра `mynetworks`. Во втором разделе приведены возможные коды отклика SMTP-сервера из RFC 2821.

Подсети CIDR Notation

В нотации CIDR IP-адрес представлен в виде A.B.C.D/*n*, где A.B.C.D – префикс[1], а *n* – длина IP-префикса. Длина IP-префикса определяет количество значащих бит, используемых для идентификации сети. Например, 192.9.205.22/18 означает, что первые 18 бит используются для представления сети, а оставшиеся 14 – для идентификации хостов. Наиболее распространенными являются префиксы 8, 16, 24 и 32.

Даже если вы утверждаете, что уже с 10 лет на короткой ноге с компьютерами и были одним из первых, кто вышел в Интернет (еще тогда, когда слово ARPANET что-то значило), у вас все же могут быть проблемы с запоминанием масок подсетей в нотации CIDR. В табл. C.1 перечислены маски подсетей и их эквиваленты.

---

[1] На самом деле A.B.C.D – это адрес, а *n* – длина префикса; сам префикс нам не видно, т. к. префикс – это просто часть 32-битного адреса, представленного в виде A.B.C.D. В приведенном здесь примере префикс – это старшие 18 бит адреса 195.9.205.22. – *Примеч. науч. ред.*

Таблица С.1. Подсети в нотации CIDR

Префикс CIDR	Маска сети	Двоичное значение	Количество сетей
/1	128.0.0.0	10000000000000000000000000000000	128 доменов класса A
/2	192.0.0.0	11000000000000000000000000000000	64 домена класса A
/3	224.0.0.0	11100000000000000000000000000000	32 домена класса A
/4	240.0.0.0	11110000000000000000000000000000	16 доменов класса A
/5	248.0.0.0	11111000000000000000000000000000	8 доменов класса A
/6	252.0.0.0	11111100000000000000000000000000	4 домена класса A
/7	254.0.0.0	11111110000000000000000000000000	2 домена класса A
/8	255.0.0.0	11111111000000000000000000000000	1 домен класса A
/9	255.128.0.0	11111111100000000000000000000000	128 доменов класса B
/10	255.192.0.0	11111111110000000000000000000000	64 домена класса B
/11	255.224.0.0	11111111111000000000000000000000	32 домена класса B
/12	255.240.0.0	11111111111100000000000000000000	16 доменов класса B
/13	255.248.0.0	11111111111110000000000000000000	8 доменов класса B
/14	255.252.0.0	11111111111111000000000000000000	4 домена класса B
/15	255.254.0.0	11111111111111100000000000000000	2 домена класса B
/16	255.255.0.0	11111111111111110000000000000000	1 домен класса B
/17	255.255.128.0	11111111111111111000000000000000	128 доменов класса C
/18	255.255.192.0	11111111111111111100000000000000	64 домена класса C
/19	255.255.224.0	11111111111111111110000000000000	32 домена класса C
/20	255.255.240.0	11111111111111111111000000000000	16 доменов класса C
/21	255.255.248.0	11111111111111111111100000000000	8 доменов класса C
/22	255.255.252.0	11111111111111111111110000000000	4 домена класса C
/23	255.255.254.0	11111111111111111111111000000000	2 домена класса C
/24	255.255.255.0	11111111111111111111111100000000	1 домена класса C
/25	255.255.255.128	11111111111111111111111110000000	128 хостов
/26	255.255.255.192	11111111111111111111111111000000	64 хоста
/27	255.255.255.224	11111111111111111111111111100000	32 хоста
/28	255.255.255.240	11111111111111111111111111110000	16 хостов
/29	255.25 5.255.248	11111111111111111111111111111000	8 хостов
/30	255.255.255.252	11111111111111111111111111111100	4 хоста
/31	255.255.255.254	11111111111111111111111111111110	2 хоста
/32	255.255.255.255	11111111111111111111111111111111	1 хост

Коды отклика сервера

Приведенные коды отклика сервера могут помочь вам понять сообщения в журнале или задать коды отклика, отличные от настроек Postfix по умолчанию. Далее процитирован отрывок из RFC 2821, раздел 4.2.

4.2.1. Важность кодов отклика и теоретические вопросы

Каждая из трех цифр кода отклика имеет свой уровень значимости. Первая цифра определяет успех, неудачу или незавершенность команды. Для простых клиентов SMTP или при получении неизвестного кода можно определить дальнейшие действия (продолжение, повтор, отказ и т. п.), ограничившись первой цифрой кода. Клиенты SMTP, желающие получить более точную информацию о происходящем (ошибка почтовой системы, некорректный синтаксис и т. п.), могут использовать вторую цифру кода. Третья цифра и дополнительная информация в отклике служат для предоставления наиболее подробных сведений.

Первая цифра кода может принимать пять значений:

1yz - позитивный предварительный отклик
: Команда была воспринята, но запрошенные действия пока не выполнены, поэтому в данном отклике не передается окончательное подтверждение. Клиенту SMTP следует передать другую команду, указывающую серверу на необходимость продолжения или прекращения запрошенной ранее операции.

Примечание

Обычный (не расширенный) протокол SMTP не содержит команд, способных вернуть отклик данного типа, поскольку в нем не предусмотрено продолжение или прерывание команд.

2yz - позитивный окончательный отклик
: Запрошенная операция успешно завершена и могут вводиться новые команды.

3yz - позитивный промежуточный отклик
: Команда была воспринята, но запрошенные действия пока не выполнены, и сервер ждет дополнительной информации. Клиенту SMTP следует передать другую команду, содержащую требуемые данные. Отклики этой группы используются в командах с последовательным выполнением (например, DATA).

4yz - негативный отклик о временных проблемах
: Команда не принята, и запрошенная операция не выполнена. Однако условия, не позволяющие выполнить команду, носят временный характер, и операция может быть запрошена вновь. Отправителю следует вернуться к началу последовательности команд (если таковая была). Понятие «временный» (transient) является недостаточно строгим; взаимодействующие стороны (клиент и сервер SMTP) должны одинаково интерпретировать его. Для каждого отклика этой группы время может различаться, но клиенту SMTP ничто не запрещает продолжать попытки. Различия между временными и постоянными проблемами (коды 5yz) достаточно условны, и отклики 4yz обычно возвращаются в тех случаях, когда возможен позитивный результат при повторе без изменения формы команды и свойств отправителя или получателя (то есть команду можно просто повторить без изменений).

5yz - негативный отклик о постоянных проблемах

Команда не принята, и запрошенная операция не выполнена. Клиент SMTP не должен просто повторять команду, поскольку она заведомо не будет выполнена. Некоторые «постоянные» проблемы могут быть решены корректировкой команд, поэтому пользователь (человек) может запросить у клиента SMTP повтора операции после корректировки команд или их порядка.

[. . .]

4.2.3. Коды откликов в порядке номеров

211 System status, or system help reply Отклик с системной справкой или состоянием системы.

214 Help message Информация о работе с сервером или отдельных командах.

220 <domain> Service ready Служба для указанного домена готова.

221 <domain> Service closing transmission channel Закрывается канал передачи для указанного домена.

250 Requested mail action okay, completed Операция благополучно завершена.

251 User not local; will forward to <forward-path> Нелокальный пользователь – почта будет перенаправляться по пути <forward-path> (см. раздел 3.4).

252 Cannot VRFY user, but will accept message and attempt delivery Не удается проверить почтовый ящик, но сообщение принято и сервер попытается его доставить (см. раздел 3.5.3).

354 Start mail input; end with <CRLF>.<CRLF> Начало ввода данных. Завершение по <CRLF>.<CRLF>.

421 <domain> Service not available, closing transmission channel Для указанного домена обслуживание невозможно и канал связи закрывается. Это может быть откликом на любую команду, если известно, что сервис отключен.

450 Requested mail action not taken: mailbox unavailable Запрошенная операция невозможна – почтовый ящик недоступен (например, занят).

451 Requested action aborted: error in processing Запрошенная операция прервана в результате ошибки.

452 Requested action not taken: insufficient system storage Запрошенная операция не выполнена по причине нехватки пространства (на диске).

500 Syntax error, command unrecognized Синтаксическая ошибка, команда не распознана (это может говорить о слишком длинной команде).

501 Syntax error in parameters or arguments Синтаксическая ошибка в параметрах или аргументах.

502 Command not implemented Команда не реализована (см. раздел 4.2.4).

503 Bad sequence of commands Некорректный порядок команд.

504 Command parameter not implemented Параметры команды не реализованы.

550 Requested mail action not taken: mailbox unavailable Запрошенная операция невозможна – почтовый ящик недоступен (например, почтовый ящик не найден, к нему нет доступа или команда отвергнута по соображениям используемой политики).

551 User not local; please try <forward-path> Нелокальный пользователь – попытайтесь использовать путь <forward-path> (см. раздел 3.4).

552 Requested mail action aborted: exceeded storage allocation Запрошенная операция прервана по причине превышения выделенного (дискового) пространства.

553 Requested action not taken: mailbox name not allowed Запрошенная операция не выполнена – недопустимый почтовый ящик (например, синтаксическая ошибка в имени ящика).

554 Transaction failed или No SMTP service here Отказ транзакции или отсутствие поддержки сервиса SMTP (при попытке соединения).

Глоссарий

A

Active Directory. Служба каталогов Microsoft. Централизованная система для распределения данных о пользователях, сетях и параметрах безопасности. Можно воспринимать ее как LDAP с Kerberos только без соответствия стандартам.

ADSI (Active Directory Service Interface). Интерфейс API, позволяющий разработчикам сценариев или программ на C/C++ свободно запрашивать объекты в Active Directory и манипулировать ими.

A-запись. Запись в DNS, устанавливающая соответствие между именем хоста и IP-адресом.

B

base64, алгоритм кодирования. Система кодирования данных, которая преобразует двоичные данные в печатаемые символы ASCII. Одна из кодировок при передаче MIME-содержимого, используемая для электронной почты в Интернете.

Boolean. Тип переменной, имеющей логическое значение: истина или ложь.

C

CA (Certification Authority – центр сертификации). Орган, выдающий цифровые сертификаты и управляющий ими.

chroot. Системный вызов `chroot()` указывает новый корневой каталог для процесса.

CNAME-запись. Запись в DNS, определяющая синоним (мнемоническое имя, alias) для канонического имени (canonical name). В зависимости от используемой ветки DNS (прямой или обратной) запись CNAME может ссылаться на запись A или PTR. См. также *A-запись* и *PTR-запись*.

D

DCF-77. Радиосигнал точного времени, передаваемый в ультрадлинноволновом диапазоне на частоте 77,5 кГц. Официальным источником этого сигнала является метрологический институт Physikalisch-Technische Bundesanstalt (*http://www.ptb.de*), который распространяет точное время для Германии. Каждый может использовать этот сигнал для синхронизации устройства с официальным временем.

DMZ (Demilitarized Zone – демилитаризованная зона). Нейтральная зона между частной и общедоступной сетями, которая обеспечивает пользователям из общедоступной сети контролируемый доступ к данным, предоставленным частной сетью. См. также *межсетевой экран*.

DUL (dial-up user list – список пользователей коммутируемого доступа). Черный список в формате RBL, содержащий IP-адреса известных пулов коммутируемого доступа.

DNS (domain name service – служба доменных имен). Универсальная служба обработки запросов с распределенной структурой и тиражированием, главным образом используемая в Интернете для преобразования имен хостов в сетевые адреса.[1]

E

ESMTP (Extended Simple Mail Transfer Protocol – расширенный простой протокол электронной почты). Набор расширений исходного протокола SMTP, который позволяет почтовому клиенту запрашивать почтовый сервер о предоставляемых им услугах.

F

FIFO (first-in, first-out – первым вошел – первым вышел). Дисциплина обслуживания заявок или данных. Очередь относится к системе FIFO; первый прибывший первым обрабатывается. Суть в том, что сначала всегда обрабатывается самая старая заявка.

FQDN (fully qualified domain name – полностью определенное доменное имя). Полное имя системы, состоящее из локального имени хоста и имени домена, включая домен верхнего уровня. Например, `mail` – это имя хоста, а `mail.example.com` – соответствующее FQDN. Полностью определенного доменного имени должно быть достаточно для определения уникального адреса в сети Интернет для любого хоста. Такой процесс (называемый разрешением имен) использует службу доменных имен (DNS).

---

[1] Довольно часто используется и для маршрутизации почты. – *Примеч. науч. ред.*

I

IANA (Internet Assigned Number Authority – Агентство по распределению нумерации в Интернете). Центральный орган регистрации различных номеров, присваиваемых в рамках IP-протокола, таких как номера портов, протоколов, предприятий, параметров, кодов и типов.

IMAP (Internet Message Access Protocol – протокол доступа к сообщениям в сети Интернет). Протокол, который позволяет клиенту обращаться к сообщениям электронной почты на сервере и манипулировать ими. Он позволяет манипулировать удаленными папками сообщений (почтовыми ящиками), обеспечивая функциональность, аналогичную локальным почтовым ящикам.

Протокол IMAP включает в себя операции по созданию, удалению и переименованию почтовых ящиков; проверку поступления новых сообщений; окончательное удаление сообщений; поиск и выборочное извлечение атрибутов сообщений, текстов и их частей. Он не определяет средство для отправки почты – эту функцию выполняет протокол передачи почты, такой как SMTP.

IPC (interprocess communication – взаимодействие процессов). Интерфейс API и сопутствующая поддержка, позволяющие нескольким процессам общаться друг с другом.

L

LHS (left-hand side – левая сторона). В карте с двумя столбцами левая сторона – это левый столбец. В Postfix левая сторона записи называется ключом.

LDAP (Lightweight Directory Access Protocol – облегченный протокол службы каталогов). Протокол доступа к онлайновым службам каталогов. Определяет достаточно простой протокол для обновления и поиска каталогов, работающий поверх TCP/IP.

LMTP (Local Mail Transfer Protocol – локальный протокол электронной почты). Производный протокол от SMTP. См. также *SMTP*.

M

mbox, формат. Формат файла, используемый для хранения электронных сообщений. Все сообщения сцепляются в один файл, при этом их разделяют строка `From` в начале каждого сообщения и пустая строка в конце сообщения.

MIME (Mutipurpose Internet Mail Extensions – многоцелевые расширения электронной почты). Набор стандартов для формата электронной почты в сети Интернет.

MTA (message transport agent – агент передачи сообщений). Программа, занимающаяся получением входящих электронных сообщений и (или) доставкой их индивидуальным пользователям. Может также передавать нелокальные сообщения их удаленным адресатам. См. также *IMAP*, *POP* и *SMTP*.

MUA (mail user agent – пользовательский почтовый агент, почтовый клиент). Программа, которая позволяет пользователю создавать и читать электронные письма. Почтовый клиент обеспечивает интерфейс между пользователем и агентом передачи сообщений (MTA – message transfer agent). Исходящая корреспонденция со временем передается для доставки агенту MTA, а входящие сообщения забираются оттуда, где их оставит агент доставки сообщений (MDA – mail delivery agent).

Mumble. Это слово часто используется в списках рассылки Postfix для обозначения набора параметров, которые имеют похожие имена, отличающиеся лишь какой-то частью. Например, название `smtpd_mumble_restrictions` подразумевает под собой все ограничения `smtpd`, такие как `smtpd_client_restrictions`, `smtpd_sender_restrictions`, `smtpd_recipient_restrictions`, `smtpd_ data_restrictions` и т. д.

MX-запись (mail exchange record – почтовая запись). Запись в DNS, которая указывает хост, обрабатывающий корреспонденцию, отправленную на его имя.[1]

N

NAT (network address translation – трансляция сетевых адресов). Трансляция сетевых адресов (иногда также называемая маскировкой сети или IP-маскировкой) – это методика, используемая при организации компьютерной сети для того, чтобы обеспечить частной сети выход в сеть общего пользования через единую точку. В ее основе лежит подмена IP-адресов сетевых пакетов, проходящих через маршрутизатор или межсетевой экран.

NTP (Network Time Protocol – протокол сетевого времени). Интернет-протокол, который применяется для синхронизации компьютерных часов с некоторым эталонным временем. NTP – это стандартный протокол сети Интернет, который был первоначально разработан профессором Дэвидом Л. Миллсом из Делаверского университета.

[1] Не совсем так: запись в описании домена `domain IN MX relay` означает, что почту на адрес `user@domain` следует отправлять компьютеру `relay`; иногда используется в виде `host IN MX relay` и в таком случае определяет, куда следует отправлять почту, адресованную на `user@host`. – *Примеч. науч. ред.*

P

PGP (Pretty Good Privacy). Криптографическая программа, которая используется для обеспечения конфиденциальности передачи данных и подтверждения аутентичности отправителя информации.

POP (Post Office Protocol – почтовый протокол). Протокол для получения электронной почты с сервера. Обеспечивает незамедлительную загрузку сообщений с сервера. См. также *IMAP*, *SMTP*.

PTR-запись. Запись в DNS, сопоставляющая IP-адресу имя хоста.

Q

QMQP (Quick Mail Queuing Protocol – быстрый протокол почтовых очередей). Протокол QMQP обеспечивает централизованную очередь сообщений внутри кластера хостов. На одном центральном сервере работает агент передачи сообщений. Остальные хосты не имеют собственных очередей сообщений, они передают каждое новое сообщение центральному серверу по протоколу QMQP. Этот протокол был изобретен Д. Дж. Бернштайном (D.J. Bernstein), который также создал qmail.

R

RAID (Redundant Array of Independent Disks – избыточный массив независимых дисков). Способ хранения данных на нескольких дисках. Все диски RAID-массива воспринимаются операционной системой как один диск. В RAID-массиве возможна балансировка операций ввода-вывода, что повышает производительность.

RFC (Request for Comments – запрос на комментарии). Официальный документ группы IETF (Internet Engineering Task Force – рабочая группа по проектированию Интернета). Эти документы являются информационными или же предназначены для того, чтобы стать стандартами Интернета и обеспечивать возможность взаимодействия сетей и приложений. Никто не может изменить RFC, но можно написать новый RFC, который заменит собой существующий.

RHS (right-hand side – правая сторона). В карте с двумя столбцами правая сторона – это последний столбец. В Postfix правая сторона записи называется значением.

root. См. *суперпользователь*.

S

S/MIME (Secure Multipurpose Internet Mail Extensions – безопасные многоцелевые расширения электронной почты). Стандарт, описывающий безопасный метод пересылки почты. Он определяет способ включения в тело сообщения цифрового сертификата и информации о кодировании.

SSL (Secure Socket Layer – протокол безопасных соединений). См. *TLS*.

Sendmail. Один из самых старых и наиболее распространенных MTA в Интернете.

SASL (Simple Authentication and Security Layer – простой протокол аутентификации и безопасности). Базовый механизм аутентификации, определенный в RFC 2222 (*ftp://ftp.rfc-editor.org/in-notes/rfc2222.txt*) для приложений, использующих соединения на основе протоколов IMAP, LDAP или SMTP. Предоставляет сервисы аутентификации для таких приложений, также может находить пользовательские данные в различных хранилищах.

SMTP (Simple Mail Transfer Protocol – простой протокол передачи сообщений). Протокол, определенный в стандарте STD 10, RFC 821, используемый для передачи электронной почты между компьютерами (*http://www.faqs.org/rfcs/std/std- index.html*).

SQL (Structured Query Language – язык структурированных запросов). Язык программирования, который используется для взаимодействия с базами данных.

T

tarpit. Сервис в компьютерной системе (обычно на сервере), который создает задержку входящих соединений на максимально возможное время. Такой сервис снижает эффективность действий злоумышленника, таких как рассылка спама или атака по словарю, замедляя процесс атаки. Название возникло из аналогии с болотом (tar pit – смоляная яма), попав в которое, животные медленно тонут в трясине. Используется также немецкое название teergrube.

telnet. Сетевой протокол, использующийся в Интернете. Это также название программы, которая применяется для создания сеанса telnet на удаленном компьютере.

TLS (Transport Layer Security – безопасность транспортного уровня). Протокол TLS (ранее называвшийся SSL) – это протокол, обеспечивающий шифрование соединения между клиентом и сервером. Не следует путать его с технологиями шифрования сообщений, такими как S/MIME и PGP, которые шифруют содержимое, но не сам коммуникационный канал.

U

UCE (unsolicited commercial email – незапрашиваемая коммерческая электронная почта). Более точное определение спама. Не следует путать с коммерческой электронной почтой, на которую получатели подписываются по собственной инициативе.

UNIX. Операционная система, которая была создана в исследовательском центре Bell Labs в 1969 году.

UUCP (UNIX-to-UNIX Copy Protocol – протокол взаимодействия UNIX-машин). Протокол и программа для UNIX, позволяющие одной UNIX-системе отправлять файлы другой системе по последовательному каналу, при этом кабель может соединять непосредственно последовательные порты двух машин или на каждом конце телефонной линии может стоять модем.

Существует программное обеспечение, позволяющее UUCP работать по Ethernet, хотя здесь возможен более удачный выбор, например scp для передачи файлов, SMTP для электронной почты и NNTP для новостей.

А

Атака по словарю. Нахождение адреса получателя методом просмотра списка возможных вариантов, часто – списка слов из словаря.

Атака с внедрением посредника (man-in-the-middle attack). Атака, при которой атакующий внедряется в канал связи. Атакующий имеет возможность читать и изменять сообщения, отправляемые одной стороной другой стороне, не позволяя сторонам узнать, что канал связи между ними скомпрометирован.

Б

Библиотека. Набор скомпилированных программных модулей, которые могут быть связаны с компилируемой программой. Библиотеки играют роль вспомогательного кода для других программ. См. также *включаемый файл*.

В

Включаемый файл. Включаемые (заголовочные) файлы содержат прототипы функций, определения констант и макросы, необходимые для компиляции программы. Большинство включаемых файлов соответствуют библиотекам. См. также *библиотека*.

Вложение. Файл внутри сообщения электронной почты.

Внешнее приложение. Приложение вне Postfix, например сценарий или антивирусная программа.

Вредоносная программа (malware – от malicious software). Программа или файл, причиняющие вред компьютеру.

Г

Групповое программное обеспечение (groupware). Совокупность выполняющих различные функции тесно интегрированных приложений, таких как электронная почта, программы планирования, адресная база данных и новостные службы.

Д

Демон. Процесс, который работает и выполняет свои функции в фоновом режиме.

Динамический IP-адрес. IP-адрес, который назначается сетевому интерфейсу компьютера из пула IP-адресов при подключении к сети.

Динамическое связывание. Способ исполнения программы, при котором операционная система загружает и связывает библиотечный код для исполняемого файла в момент исполнения. См. также *библиотека*.

Дистрибутив. Комплект программ операционной системы, включающий в себя ядро, различное бесплатное программное обеспечение и, в некоторых случаях, коммерческое программное обеспечение. Термин наиболее часто используется в отношении Linux.

Домен. Группа компьютеров, имена которых имеют общий суффикс – имя домена. См. также *домен верхнего уровня*, *DNS*.

Домен верхнего уровня. Последняя и наиболее значимая часть полностью определенного доменного имени в Интернете, расположенная после последней точки. Например, хост `mail.example.com` находится в домене верхнего уровня `com` (предназначенном для коммерческих организаций).

К

Карта. В Postfix карта – это таблица из двух столбцов, каждая строка которой представляет собой запись, сопоставляющую ключу значение. Ключ и значение иногда называют левой и правой стороной соответственно. См. также *LHS* и *RHS*.

Комментарий. Примечание внутри файлов конфигурации или исходного текста, которое содержит вспомогательную информацию об элементах конфигурации или кода.

Копия. Элемент заголовка электронного сообщения, помеченный как cc: (от carbon copy) и содержащий перечень дополнительных адресов, по которым должно быть отправлено сообщение; такой заголовок виден всем получателям.

Л

Ложноположительный. Ошибочно определенный как «истина» результат.

М

Макрос. Короткая инструкция, преобразуемая в более длинный набор инструкций.

Маршрутизатор. Компьютерное сетевое устройство, определяющее следующую точку сети, в которую следует переслать пакет с данными, чтобы он достиг своего получателя.

Межсетевой экран (firewall). Шлюз, который управляет входящими и исходящими соединениями между частной сетью и сетью общего пользования. Также может обеспечивать управляемый доступ извне в демилитаризованную зону. См. также *DMZ*.

О

Открытый прокси-сервер (open proxy). Неправильно настроенный прокси-сервер, который обрабатывает заявки от третьих сторон. Открытые прокси-серверы могут использоваться для передачи корреспонденции серверам.

Открытый ретранслятор (open relay). SMTP-сервер, который пересылает почту между третьими сторонами. Ретрансляция сообщения третьей стороны имеет место, когда почтовый сервер обрабатывает сообщение электронной почты, в котором ни отправитель, ни получатель не являются локальными пользователями.

П

Патч (patch). Дополнение (временное) к некоторому коду, обычно служащее для исправления существующей ошибки или для поддержки новой функциональности.

Порт. Порт (сетевой) – это интерфейс взаимодействия с компьютерной программой по сети. Сетевые порты обычно пронумерованы, а реализация протокола (например, TCP или UDP) связывает номер порта с отправляемыми им данными. Принимающая сторона использует номер порта для того, чтобы узнать, какой компьютерной программе следует отправить данные.

Пробельный символ (whitespace). Пробельный символ – это любой символ, который занимает место, но не отображается на дисплее.[1]

Прокси-сервер (proxy) . Сервер, который действует от имени другого сервера[2], обычно незаметно для пользователей.

Р

Разрешение имен. Процесс сопоставления имени хоста IP-адресу.

Регулярное выражение. Расширенная система (встречаются сокращения regexp, regex, regxp – от regular expression) сличения с образцом,

---

[1] Такие символы также часто называют «пустыми». – *Примеч. науч. ред.*

[2] ...или от имени любого другого компьютера в широком смысле этого слова, не обязательно сервера. – *Примеч. науч. ред.*

которая в действительности является реализацией недетерминированного конечного автомата, принимающего определенный язык.

С

Сертификат. Файл, который хранит информацию, удостоверяющую личность человека или компьютера.

Слепая[1] копия. Элемент заголовка электронного сообщения, помеченный как bcc: (от blind carbon copy) и содержащий перечень адресов, по которым должно быть отправлено сообщение; такой заголовок не виден получателям. См. также *Копия*.

Сокет домена UNIX. Сокеты домена UNIX (корректный термин в стандарте POSIX – это POSIX Local IPC Sockets) используются в основном для организации взаимодействия процессов и поэтому называются также сокетами IPC. Такие соединения существуют внутри локального компьютера и не используют передачу по физической сети.[2]

Суперпользователь. Привилегированный пользователь (также называемый пользователем `root`), имеющий все права и разрешения на доступ во всех режимах (однопользовательском и многопользовательском) UNIX-подобной операционной системы.

Ш

Шестнадцатеричная система счисления. Система счисления с основанием 16, в которой числа записываются при помощи символов 0–9 и A–F или a–f.

Я

Ядро. Основа операционной системы. Ядро отвечает за предоставление разнообразным компьютерным программам надежного доступа к аппаратной части компьютера.

---

[1] Существенно чаще этот термин переводится как «скрытая копия», так как «слепая копия» напоминает о временах пишущих машинок, в которые при печати под копирку самая нижняя копия оказывалась «слепой» (плохого качества). – *Примеч. науч. ред.*

[2] Эти сокеты представляют собой элемент механизма взаимодействия процессов в системах UNIX и к термину «домен», упомянутому выше, никакого отношения не имеют. – *Примеч. науч. ред.*

Алфавитный указатель

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

А

Б

В

Г

Д

З

И

К

Л

М

Н

O

П

P

C

Т

У

Ф

Х

Ц

Ч