Безопасный Postfix

Anton Batenev

31.05.2013

Оглавление

Предисловие

Однажды, настраивая сервер, я совершенно случайно обнаружил, что отправляемая с сервера почта идет по открытому каналу связи несмотря на то, что настройки почтового клиента позволяли начинать защищенный сеанс в случае поддержки данного функционала принимающим почтовым сервером. В процессе решения данной проблемы выяснилась занимательная деталь — все исходящие соединения на 25-й порт проксировались через сторонний сервис, который в процессе работы изменял проходящие через него данные, нарушая тем самым предполагаемый ход сеанса связи. Выглядело это приблизительно так:

```
$ telnet gmail-smtp-in.l.google.com 25
Trying 173.194.70.27...
Connected to gmail-smtp-in.l.google.com.
Escape character is '^]'.
220 ************************************************
EHLO example.com
250-mx.google.com at your service, [xxx.xxx.xxx.xxx]
250-SIZE 35882577
250-8BITMIME
250-XXXXXXXA
250 ENHANCEDSTATUSCODES
```

Здесь особого внимания заслуживают «странные» строки:

```
220 ************************************************
...
250-XXXXXXXA
```

Для сравнения, посмотрим на начало «правильного» сеанса связи, полученного с другого сервера:

```
$ telnet gmail-smtp-in.l.google.com 25
Trying 173.194.71.27...
Connected to gmail-smtp-in.l.google.com.
Escape character is '^]'.
220 mx.google.com ESMTP xxxxxxxxxxxxxxxxxxxx - gsmtp
EHLO example.com
250-mx.google.com at your service, [xxx.xxx.xxx.xxx]
250-SIZE 35882577
250-8BITMIME
250-STARTTLS
250 ENHANCEDSTATUSCODES
```

Сравнивая оба сеанса мы видим различие двух строк, которые были «испорчены» сторонним сервисом:

```
220 mx.google.com ESMTP xxxxxxxxxxxxxxxxxxxx - gsmtp
...
250-STARTTLS
```

Порча первой строки не представляет никакой опасности — SMTP клиенту важен только код `220`, а приглашение (или т.н. «баннер») может быть любым.

Любопытна вторая измененная строка — `250-STARTTLS`, в которой анонсируется возможность удаленного SMTP сервера начать защищенный сеанс связи. Из за вмешательства стороннего сервиса, который скрыл информацию о наличии данного расширения на сервере, клиент SMTP продолжит сеанс связи без использования шифрования, что, в свою очередь, позволит стороннему сервису получить полный доступ к заголовкам и телу передаваемого письма.

Вне зависимости от того, с какой целью производится вмешательство в почтовый трафик (а это может быть и «благая» цель типа фильтрации спама или защиты от вирусов) подобное поведение называется MITM Downgrade Attack или атакой «человек посередине».

Пока вы пересылаете бородатые анекдоты, максимальная угроза от такой атаки состоит в том, что кто-то узнает о вашей неполиткорректности. Когда в отсылаемой корреспонденции начинают фигурировать конфиденциальные данные (например, данные о финансовых транзакциях), последствия от их утечки могут оказаться куда более серьезными.

Сервер находился на территории **Кипра**, была весна 2013 года и обострение паранойи...

Безопасная отправка почты

Начальная настройка

Большинство серверов в интернете не предполагают приема почты, а только отправку (уведомления о происходящих событиях, рассылки и т.д.). Тем не менее достаточно часто можно встретить «декларирование» возможности приема почты и (хотя существенно реже) сервера в состоянии open relay — почтовый сервер, открытый для пересылки почты от любых отправителей любым получателям.

Поскольку мы не предполагаем прием почты из внешнего мира, то первым шагом имеет смысл отключить декларирование данной возможности на уровне Postfix:

```
inet_interfaces = loopback-only
```

Параметр конфигурации inet_interfaces может принимать следующие значения:

- `all` — прослушивать все адреса на всех интерфейсах (это значение по умолчанию);
- `loopback-only` — прослушивать адреса только на loopback интерфейсах (обычно это `127.0.0.1` и `[::1]`);
- Список IP адресов для прослушивания через разделитель (пробел или запятая), где IPv6 адреса указываются в квадратных скобках.

Так же, имеет смысл явно указать список доверенных сетей:

```
mynetworks = 127.0.0.0, [::1]
```

Параметр конфигурации mynetworks задает список сетей или хостов через разделитель, которые могут отправлять почту **без авторизации**.

Отправка с использованием TLS

Для организации защищенного канала связи в Postfix используется TLS — безопасный протокол транспортного уровня. Для того, чтобы почтовый сеанс связи между двумя хостами производился по защищенному каналу связи должны быть выполнены следующие условия:

- Отправитель и получатель должны поддерживать расширения SMTP (ESMTP — SMTP Service Extensions, RFC 1869);
- Отправитель и получатель должны поддерживать расширение STARTTLS (SMTP Service Extension for Secure SMTP over Transport Layer Security, RFC 3207);
- В сеансе связи получатель должен декларировать поддержку расширения STARTTLS, а отправитель им воспользоваться.

На сегодняшний день поддержку ESMTP имеют подавляющее большинство почтовых серверов (стандарт датирован 1995-м годом), а на оставшиеся два пункта следует обратить более пристальное внимание.

Очевидно, что при регистрации или подписке на рассылку, пользователь будет использовать почтовый адрес, зарегистрированный на каком-либо популярном публичном почтовом сервисе. Используя базу данных из почти 60 тыс. адресов одного веб-проекта я получил следующую статистику по предпочтениям пользователей почты в Российском сегменте интернета:

домен	%
mail.ru	53.53
yandex.ru	22.06
gmail.com	9.56
rambler.ru	5.97
ukr.net	1.85
yahoo.com	1.53
прочие	5.50

Здесь в долю домена `@mail.ru` входят (помимо самого `@mail.ru`) `@inbox.ru`, `@list.ru`, `@bk.ru`. В долю домена `@yandex.ru` входят `@ya.ru`, все региональные поддомены `@yandex` и `@narod.ru`. К «прочим» отнесены все почтовые домены, набравшие менее 1% в общем распределении. На разных проектах цифры, естественно, могут отличаться, однако сохранится общая тенденция в списке лидеров.

Попробуем посмотреть на уровень поддержки STARTTLS лидерами рынка:

домен	STARTTLS
mail.ru	нет
yandex.ru	нет
gmail.com	**да**
rambler.ru	нет
ukr.net	нет
yahoo.com	нет
прочие	28% да

Другими словами, перед нами возникает очень интересный факт — **поддержка TLS в реальной жизни может быть обеспечена всего для 11% получателей** и с очень большой вероятностью сервер получателя будет находиться за пределами Российской Федерации (например, сервис Google Apps — почта для домена). Тут есть над чем призадуматься...

В сложившейся ситуации отношения крупных игроков к безопасности своих пользователей все, что мы можем сделать — это вести разъяснительную работу среди посетителей своих сайтов и сервисов, предоставляя им актуальную информацию и рекомендации по обеспечению собственной безопасности. А пока, позаботимся о тех 11% счастливчиков.

Базовую конфигурацию для использования TLS при отправке сообщений можно представить в виде:

```
tls_high_cipherlist = HIGH:@STRENGTH

smtp_tls_loglevel               = 1
smtp_tls_security_level         = may
smtp_tls_session_cache_database = btree:/var/lib/postfix/smtp_tls_cache
smtp_tls_CAfile                 = /etc/ssl/certs/ca-certificates.crt
smtp_tls_protocols              = !SSLv2, !SSLv3
smtp_tls_ciphers                = high
smtp_tls_exclude_ciphers        = aNULL, MD5, CAMELLIA
```

Здесь:

- tls_high_cipherlist — список шифров с «высоким» уровнем защиты. Значение `HIGH:@STRENGTH` выбирает шифры с высоким уровнем защиты и упорядочивает их в порядке уменьшения стойкости шифра. Подробнее см. документацию по OpeenSSL;
- smtp_tls_loglevel — уровень детализации ведения лога соединений сеансов TLS (0-4, нет-отладка). Здесь включаем минимальный уровень логгирования в котором будет записываться информация о факте использования защищенного соединения;
- smtp_tls_security_level — уровень безопасности при установке защищенного соединения. Значение `may` предполагает использование TLS при поддержке второй стороной игнорируя дополнительные проверки;
- smtp_tls_session_cache_database — имя файла базы для кэширования информации о TLS сессиях (для FreeBSD может иметь значение, например, `btree:/var/db/postfix/smtp_tls_cache`);

- smtp_tls_CAfile — файл со списком сертификатов доверенных центров сертификации (для FreeBSD обычно `/usr/local/share/certs/ca-root-nss.crt`);
- smtp_tls_protocols — список поддерживаемых протоколов. Протоколы SSLv2 и SSLv3 в современном мире практически никто не использует, однако, если вы хотите дать шанс старому или неверно настроенному серверу, можно оставить этот параметр по умолчанию — `!SSLv2`;
- smtp_tls_ciphers — минимальный уровень шифрования. Значение `high` берется из списка шифров заданных в `tls_high_cipherlist`;
- smtp_tls_exclude_ciphers — список исключаемых шифров. Исключаются шифры без аутентификации (`aNULL`) и слабой (`MD5`) аутентификацией. Шифр CAMELLIA исключен ввиду его низкой распространенности (количества `HIGH` шифров более чем достаточно, чтобы сделать выбор).

В данной конфигурации ключевым параметром является `smtp_tls_security_level`. Поскольку использование сертификатов, подписанных доверенными центрами сертификации, может быть достаточно дорого (несмотря на то, что Go Daddy предлагает приобрести сертификат всего за 6$), многие почтовые сервера используют самоподписанные сертификаты. Использование самоподписанных сертификатов не влияет на стойкость шифрования, однако, такие сертификаты не проходят проверку при использовании более строгого уровня безопасности — в этой ситуации помогает значение параметра `may`, который позволяет устанавливать защищенное соединение даже если сертификат сервера получателя не пошел проверку.

Усиленный TLS

В базовой конфигурации TLS не производится никаких дополнительных проверок сертификатов, которые предоставляются сервером получателя, однако, существует как минимум один крупный почтовый сервис, предоставляющий возможность приема почты с использованием TLS и имеющий сертификат, подписанный доверенным центром сертификации — речь идет о сервисе Gmail.

Для усиления уровня безопасности в зависимости от домена получателя можно воспользоваться параметром smtp_tls_policy_maps в котором указать сопоставление домена и требуемого уровня безопасности:

```
smtp_tls_policy_maps = hash:/etc/postfix/tls_policy
```

Сам файл `tls_policy` с картой сопоставлений для gmail будет выглядеть следующим образом:

```
gmail.com     secure match=.google.com:google.com:.gmail.com:gmail.com
```

Здесь, для домена получателя gmail.com задается максимальный уровень проверки сертификата (`secure`) и сертификат должен быть выписан для поддомена или домена google.com или gmail.com. На данный момент, сертификат, предоставляемый MX серверами gmail.com, выписан на домен mx.google.com, однако, поскольку нельзя

гарантировать, что этот же сертификат будет использоваться и в дальнейшем, принятие любого сертификата для домена или поддомена компании является разумным компромиссом между надежностью доставки и безопасностью.

После изменения текстового варианта карты, необходимо создать саму базу данных выполнив команду:

```
# postmap /etc/postfix/tls_policy
```

Для получения имени домена, для которого выписан сертификат того или иного почтового сервера, необходимо получить список его MX серверов:

```
$ dig gmail.com mx
...
;; ANSWER SECTION:
gmail.com. 3600 IN MX  5 gmail-smtp-in.l.google.com.
gmail.com. 3600 IN MX 10 alt1.gmail-smtp-in.l.google.com.
gmail.com. 3600 IN MX 20 alt2.gmail-smtp-in.l.google.com.
gmail.com. 3600 IN MX 30 alt3.gmail-smtp-in.l.google.com.
gmail.com. 3600 IN MX 40 alt4.gmail-smtp-in.l.google.com.
...
```

После чего, выбрав требуемый сервер, получить информацию о сертификате:

```
$ openssl s_client -connect gmail-smtp-in.l.google.com:25 -starttls smtp
...
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=mx.google.com
   i:/C=US/O=Google Inc/CN=Google Internet Authority
 1 s:/C=US/O=Google Inc/CN=Google Internet Authority
   i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
...
```

Проверка валидности сертификата производится с указанием имени файла, содержащего список сертификатов доверенных центров сертификации:

```
$ openssl s_client -CAfile /etc/ssl/certs/ca-certificates.crt \
    -connect gmail-smtp-in.l.google.com:25 -starttls smtp
...
SSL-Session:
    ...
    Verify return code: 0 (ok)
...
```

Уведомления о невозможности доставки

В случае, когда сообщение не может быть доставлено получателю в течении определенного времени или возникает фатальная ошибка (например, сервер получателя

сообщил, что пользователя не существует), отправителю генерируется bounce сообщение, информирующее о невозможности доставки, **содержащее в себе исходное сообщение**. В случае, если bounce сообщение невозможно доставить отправителю, может генерироваться double-bounce (или 2bounce) сообщение, которое обычно отправляется администратору почтового сервера (по умолчанию на адрес postmaster), которое так же может содержать тело исходного сообщения или другую конфиденциальную информацию. Если double-bounce сообщение невозможно доставить, оно удаляется.

Время жизни сообщения в очереди по умолчанию составляет пять дней и контролируется параметром maximal_queue_lifetime. В современных реалиях никто не ждет важного сообщения столь длительный срок (а по моему опыту, если письмо не может быть отправлено в течении шести–восьми часов, то вероятность его успешной отправки в дальнейшем минимальна).

Время жизни bounce сообщения в очереди по умолчанию составляет так же пять дней и контролируется параметром bounce_queue_lifetime. Обычно, веб-приложения отправляют сообщения с адресов вида `noreply@example.com`, которые на практике никем не просматриваются (в лучшем случае, это реально существующий ящик, который постепенно забивается уведомлениями).

Типы сообщений, которые отправляются на адрес postmaster (если не указано иного в параметрах bounce_notice_recipient, 2bounce_notice_recipient, delay_notice_recipient и error_notice_recipient) контролируются параметром notify_classes и по умолчанию отправляются сообщения только о программных ошибках или нехватке ресурсов (на адрес указанный в параметре `error_notice_recipient`).

Если не уделить этой цепочке должного внимания, однажды может случиться так, что вы узнаете о проблеме слишком поздно или произойдет непреднамеренная утечка конфиденциальной информации. Поскольку здесь нет универсального решения и в каждом случае будут иметься свои ограничения и пожелания, попробую привести пару общих советов.

Ящик, который указан в качестве обратного адреса в сообщении, **обязан** существовать. Это требование связано с тем, что до сих пор существуют почтовые сервера, которые проверяют существование адреса отправителя через попытку обратного соединения. Однако, если данный ящик никем и никогда не просматривается, bounce сообщения на него уходить **не должны**. Этого поведения можно достичь через карту «виртуальных» адресов virtual_alias_maps:

```
virtual_alias_maps = hash:/etc/postfix/virtual
```

Где содержимое `/etc/postfix/virtual` представляет собой запись вида:

```
noreply@example.com devnull
```

А алиас `devnul` в файле локальных алиасов `/etc/aliases` представляет собой запись вида:

```
devnull: /dev/null
noreply: devnull
```

Сами же карты локальных алиасов задаются в конфигурации в виде:

```
alias_maps           = hash:/etc/aliases
alias_database       = $alias_maps
local_recipient_maps = $alias_maps
```

После изменения текстовых представлений не забудьте выполнить команды обновления баз данных для соответствующего представления:

```
# postalias /etc/aliases
# postmap /etc/postfix/virtual
```

Подобная конфигурация будет отправлять все сообщения, адресованные `noreply@example.com` (в т.ч. и bounce-сообщения), прямиком в «шредер» `/dev/null`.

Сообщения о наличии программных ошибок и нехватке ресурсов, конечно же, желательно получать раньше, чем начнут беспокоиться пользователи, однако лучше это делать через специализированные средства мониторинга — в этом случае рекомендуется подавить уведомления:

```
notify_classes =
```

Все прочие сообщения, адресованные локальному пользователю root (явно или через цепочку в `/etc/aliases`), имеет смысл так же или подавлять (рискуя не узнать вовремя что-то важное и надеясь на средства мониторинга) или дублировать на два и более адресов в **различных безопасных** доменах.

Отправка почты с использованием релея

На данном этапе конфигурации сторонние наблюдатели могут предположить куда и сколько писем мы отправляем, однако, они не могут узнать содержимое посланий. Но что будет, если они установят на пути прохождения соединения «блок-пост», подобный описанному в предисловии, или будут совершать «набеги» время от времени? Вариантов не много:

- Показать содержимое послания и пройти дальше — именно этот сценарий будет реализован на базе настроек из предыдущего раздела (параметр `smtp_tls_security_level` установлен в значение `may`);
- Вернуться назад с сообщением об ошибке — этот сценарий будет реализован для домена gmail.com на базе настроек из предыдущего раздела;
- Использовать обходную «секретную» дорогу.

Разберем последние два варианта. В общем случае, наиболее безопасным решением будет вернуться и попробовать повторить отправку через некоторые интервалы времени. После чего, по истечении времени ожидания прорыва блокады, удалить

сообщение из очереди исходящих сообщений. Этот сценарий подходит для обхода периодических набегов, но лишен смысла при наличии постоянного блок-поста — все письма будут возвращаться обратно и уничтожаться.

Использование обходных путей возможно только до тех пор, пока они не будут раскрыты и перекрыты противником (т.н. security through obscurity или «безопасность через сокрытие»), однако, в случае наличия постоянного блок-поста, эта мера является единственно возможной альтернативой полному раскрытию содержимого сообщения.

Начальная настройка

В простейшем случае, пересылка всей почты доверенному (на сколько в данном контексте вообще можно говорить о доверии) посреднику конфигурируется на клиенте следующим образом:

```
relayhost = [relay.example.com]:8025
```

Здесь relayhost задает имя (или IP адрес) сервера-релея и номер порта на который пересылается вся корреспонденция. Указание имени или адреса в квадратных скобках используется для того, чтобы пересылка велась напрямую на указанный хост. В противном случае, Postfix будет пересылать почту на MX запись указанного хоста. Нестандартный номер порта используется для упрощения конфигурации релея и для обхода фильтров, установленных на стандартных портах (хотя обычно альтернативным портом для SMTP является 587).

Конфигурацию релея-посредника можно представить как:

```
mynetworks = 127.0.0.1, [::1], 1.2.4.5

smtpd_client_restrictions =
    permit_mynetworks,
    reject
```

Здесь:

- mynetworks — список доверенных сетей в который добавлен IP адрес пересылающего хоста;
- smtpd_client_restrictions — список проверок на этапе установки соединения. В данном случае разрешаются соединения из доверенных сетей, а остальные соединения отвергаются. Более подробно списки проверок описаны в документе SMTPD_ACCESS_README, часть из которых будет рассмотрена далее.

Дополнительно, для приема корреспонденции на нестандартном порту, необходимо добавить (в случае если планируется прием обычной почты) или изменить (в случае, если почта будет приниматься только от одного хоста) транспорт в `master.cf`:

```
8025      inet  n       -       -       -       -       smtpd
```

Пересылка с использованием TLS

Использование релея не отменяет необходимости передавать содержимое сообщений по защищенному каналу на случай, если нестандартный порт будет обнаружен. Для этого на пересылающем сервере необходимо настроить работу TLS. Настройка TLS сервера очень похожа на настройку TLS клиента:

```
tls_high_cipherlist = HIGH:@STRENGTH

smtpd_tls_loglevel               = 1
smtpd_tls_security_level         = may
smtpd_tls_session_cache_database = btree:/var/lib/postfix/smtpd_tls_cache
smtpd_tls_cert_file              = /etc/ssl/example.com.crt
smtpd_tls_key_file               = /etc/ssl/example.com.key
smtpd_tls_protocols              = !SSLv2, !SSLv3
smtpd_tls_ciphers                = high
smtpd_tls_exclude_ciphers        = aNULL, MD5, CAMELLIA
```

Здесь имеем два новых параметра — smtpd_tls_cert_file и smtpd_tls_key_file, которые, соответственно, являются именами файлов сертификата и приватного ключа. Предполагается, что сертификат был приобретен в доверенном центре сертификации, а о самоподписанных сертификатах будет рассказано далее. Следует обратить внимание, что обычно владельцем сертификата и приватного ключа является пользователь root, права на приватный ключ обычно выставляются в значение 0400 или 0600, на сертификат 0444 или 0644.

Поскольку клиент теперь всегда пересылает сообщения посреднику — он точно знает, что у посредника есть сертификат и точно знает какой. Для того, чтобы клиент мог убедиться в том, что посредник именно тот, за кого он себя выдает, или прервать отправку в случае ошибки проверки, необходимо изменить настройки TLS на клиентской стороне:

```
smtp_tls_security_level         = fingerprint
smtp_tls_fingerprint_digest     = sha1
smtp_tls_fingerprint_cert_match = XX:XX:XX:...
```

Здесь:

- `fingerprint` — повышение уровня безопасности при установке TLS соединения — явное указание, что требуется проверка отпечатка сертификата;
- smtp_tls_fingerprint_digest — проверка `SHA1` отпечатка вместо `MD5`, установленного по умолчанию;
- smtp_tls_fingerprint_cert_match — список отпечатков сертификатов доверенных серверов.

Для получения отпечатка сертификата сервера необходимо выполнить команду:

```
# openssl x509 -noout -fingerprint -in example.com.crt
SHA1 Fingerprint=XX:XX:XX:...
```

Полученная последовательность знаков после `SHA1 Fingerprint=` и будет являться необходимым значением для `smtp_tls_fingerprint_cert_match`.

Авторизация с использованием TLS

К сожалению, далеко не всегда имеется возможность указать на релее список доверенных сетей. Например, такая ситуация возможна, когда отправитель является виртуальным сервером и исходящий IP адрес может не соответствовать входящему, а использоваться адрес гипервизора, через который осуществляются все исходящие соединения со всех виртуальных серверов, которые он обслуживает — в этой ситуации вносить исходящий IP адрес клиента в список доверенных сетей является опрометчивым решением и необходима авторизация.

На клиентской стороне подключение сертификата для авторизации на релее осуществляется следующим образом:

```
smtp_tls_cert_file = /etc/ssl/example.com.crt
smtp_tls_key_file  = /etc/ssl/example.com.key
```

Здесь smtp_tls_cert_file и smtp_tls_key_file имя файла сертификата и приватного ключа клиента соответственно.

Т.к. пересылка почты осуществляется на нестандартном порту, включение проверки клиентского сертификата на релее рекомендуется осуществлять в конфигурации транспорта (`master.cf`), чтобы не влиять на обычную входящую почту (если таковая предполагается):

```
8025      inet  n       -       -       -       -       smtpd
  -o smtpd_tls_req_ccert=yes
  -o smtpd_tls_security_level=encrypt
  -o smtpd_tls_CAfile=/etc/ssl/certs/ca-certificates.crt
  -o smtpd_tls_fingerprint_digest=sha1
  -o relay_clientcerts=hash:/etc/postfix/relay_clientcerts
```

Здесь:

- smtpd_tls_req_ccert — обязательный запрос клиентского сертификата;
- smtpd_tls_security_level — максимальный уровень проверки сертификата;
- smtpd_tls_CAfile — файл со списком сертификатов доверенных центров сертификации (для FreeBSD обычно `/usr/local/share/certs/ca-root-nss.crt`);
- smtpd_tls_fingerprint_digest — проверка `SHA1` отпечатка вместо `MD5`, установленного по умолчанию;
- relay_clientcerts — карта со списком отпечатков разрешенных клиентских сертификатов (ключ/значение), где проверяется только ключ (отпечаток).

Таблица отпечатков клиентских сертификатов `/etc/postfix/relay_clientcerts` задается в виде:

```
XX:XX:XX:... example.com
```

Где `XX:XX:XX:...` — SHA1 отпечаток сертификата, получаемый выполнением команды:

```
# openssl x509 -noout -fingerprint -in example.com.crt
SHA1 Fingerprint=XX:XX:XX:...
```

Полученная последовательность знаков после `SHA1 Fingerprint=` и будет являться необходимым значением, а `example.com` — любое имя клиента (оно не проверяется сервером). После изменения текстового варианта карты, необходимо создать саму базу данных выполнив команду:

```
# postmap /etc/postfix/relay_clientcerts
```

Проверки валидности клиентского сертификата и его отпечатка недостаточно для того, чтобы Postfix разрешил пересылку почты (вспомним список проверок `smtpd_client_restrictions` в начале раздела). Для того, чтобы разрешить пересылку, необходимо разрешить клиентам, авторизованным по сертификату, пересылку в списке проверок smtpd_recipient_restrictions:

```
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_tls_clientcerts,
    reject
```

В данном случае разрешаются соединения из доверенных сетей и соединения от клиентов, авторизованных по сертификату (permit_tls_clientcerts), а остальные соединения отвергаются. Более подробно списки проверок описаны в документе SMTPD_ACCESS_README.

Авторизация с использованием логина и пароля

Использование авторизации на релее по логину и паролю может потребоваться в случае если у вас нет контроля над релеем для конфигурации авторизации по сертификатам, или когда получатели распределены по небольшому числу почтовых серверов для которых у отправителя имеются аккаунты.

В обоих случаях для авторизации необходимо сконфигурировать клиент в виде:

```
smtp_sasl_security_options =
smtp_sasl_auth_enable      = yes
smtp_sasl_password_maps    = hash:/etc/postfix/sasl_passwd
```

Где:

- smtp_sasl_security_options — опции авторизации. Пустое значение предполагает использование любого возможного способа авторизации, т.к. опции по умолчанию могут быть несовместимы с настройками релея (например, gmail.com);

- smtp_sasl_auth_enable — включение авторизации по логину и паролю;
- smtp_sasl_password_maps — карта авторизации.

Карта авторизации `/etc/postfix/sasl_passwd` представляет собой список вида:

```
[smtp.gmail.com]:587    gmail_user:gmail_password
[smtp.yandex.ru]:587    yandex_user:yandex_password
...
```

Ключом для карты авторизации является имя релея (в том виде, в котором оно будет использоваться) и порт релея (порт 587 — альтернативный порт для SMTP). Имя пользователя и пароль указываются через двоеточие для соответствующего релея. После изменения текстового варианта карты, необходимо создать саму базу данных и установить права доступа выполнив команды:

```
# postmap /etc/postfix/sasl_passwd
# chmod 0600 /etc/postfix/sasl_passwd
# chmod 0600 /etc/postfix/sasl_passwd.db
```

В случае использования одного релея пересылку можно задать в виде:

```
relayhost = [smtp.gmail.com]:587
```

В случае, если необходимо использовать разные релееи в зависимости от домена получателя, необходимо задать карту транспортов transport_maps:

```
transport_maps = hash:/etc/postfix/transport
```

Где задать соответствие домена получателя транспорту и релею получателя (подробнее с форматом карты можно ознакомиться по ссылке Postfix transport table format):

```
gmail.com     smtp:[smtp.gmail.com]:587
yandex.ru     smtp:[smtp.yandex.ru]:587
...
*             error:Transport not found
```

После изменения текстового варианта карты, необходимо создать саму базу данных:

```
# postmap /etc/postfix/transport
```

При отправке почты, где в качестве релея выступает публичный почтовый сервис, можно столкнуться с различными ограничениями. Так, например, некоторые почтовые службы требуют, чтобы имя отправителя письма соответствовало имени авторизованного на релее пользователя.

Для перезаписи адреса отправителя используется параметр smtp_generic_maps:

```
smtp_generic_maps = hash:/etc/postfix/generic
```

Где карта перезаписи адресов `/etc/postfix/generic` выглядит как:

```
@example.com    yandex_user@yandex.ru
```

Здесь все адреса из домена `@example.com` (например, `www-data@example.com`) будут перезаписаны адресом `yandex_user@yandex.ru`. После изменения текстового варианта карты, необходимо создать саму базу данных:

```
# postmap /etc/postfix/generic
```

Для работы с несколькими релеями, когда требуется изменять адрес отправителя в зависимости от домена получателя, требуется «расщепить» транспорт smtp на несколько виртуальных транспортов для каждого из которых задать свой параметр `smtp_generic_maps`. В этом случае карту транспортов `/etc/postfix/transport` можно представить как:

```
gmail.com    gmail:
yandex.ru    yandex:
...
*            error:Transport not found
```

Здесь различным доменам получателя соответствуют различные виртуальные транспорты. Сами же виртуальные транспорты конфигурируются в `master.cf`:

```
gmail     unix  -       -       n       -       -       smtp
  -o transport_maps=
  -o relayhost=[smtp.gmail.com]:587
  -o smtp_generic_maps=hash:/etc/postfix/generic_gmail

yandex    unix  -       -       n       -       -       smtp
  -o transport_maps=
  -o relayhost=[smtp.yandex.ru]:587
  -o smtp_generic_maps=hash:/etc/postfix/generic_yandex
```

В приведенном примере для каждого виртуального транспорта отключается карта транспортов (чтобы избежать бесконечной рекурсии), устанавливается хост-порт соответствующего релея и уникальная виртуальная карта в каждой их которых можно (нужно) перезаписать все адреса из домена `@example.com` на адрес пользователя в доменах `@gmail.com` и `@yandex.ru` соответственно.

Проксирование через TOR

Одним из способов обхода постоянного блок-поста является использование TOR — «сети луковой маршрутизации». TOR может использоваться только в паре с доверенным релеем — собственным или публичным почтовым сервером, т.к. попытка

прямой отправки почты через TOR, скорее всего, окончится неудачей — подавляющее большинство выходов SMTP из сети TOR находится под санкциями крупных почтовых систем. Так же, использование сети TOR не дает никаких гарантий относительно наличия или отсутствия дополнительных блок-постов в точке выхода, но периодическая смена маршрутов дает шанс на нахождение неконтролируемой точки выхода.

Для запуска SOCKS-прокси достаточно в файле конфигурации TOR (`/etc/tor/torrc` или `/usr/local/etc/tor/torrc` для FreeBSD) установить адрес и порт для принятия входящих соединений (документация по опциям):

```
SocksPort 9050
SocksListenAddress 127.0.0.1
```

Для проксирования трафика Postfix в SOCKS-прокси потребуется ввести дополнительный транспорт, который бы поддерживал данный тип проксирования. Наиболее простым способом является загрузка приложения с библиотекой, которая прозрачно перенаправляет все TCP соединения в прокси. Такой библиотекой, например, является tsocks.

Конфигурация параметров работы библиотеки tsocks задается в файле `/etc/tsocks.conf` (или `/usr/local/etc/tsocks.conf` для FreeBSD):

```
local = 127.0.0.1/255.255.255.255
server = 127.0.0.1
server_type = 5
server_port = 9050
```

Здесь параметр `local` является списком хостов, которые достижимы без использования прокси, остальные параметры указывают на адрес и порт SOCKS-5 прокси TOR.

Для создания транспорта Postfix необходимо создать скрипт с именем `/usr/lib/postfix/smtp_socks` вида:

```
#!/bin/sh

export LD_PRELOAD=/usr/lib/libtsocks.so
exec /usr/lib/postfix/smtp $@
```

и дать ему права на выполнение (`chmod 0755 smtp_socks`). Во FreeBSD исполняемые файлы транспортов находятся в директории `/usr/local/libexec/postfix`, а библиотека в `/usr/local/lib`.

Далее необходимо включить новый транспорт непосредственно в самой конфигурации Postfix в файле `master.cf` исправив соответствующую строку:

```
smtp      unix  -       -       n       -       -       smtp_socks
```

Здесь следует обратить внимание на значение `n` в поле `chroot` — в Debian оно по умолчанию установлено в значение `-` (или `yes`), что будет препятствовать загрузке библиотеки `libtsocks.so` без дополнительных настроек chroot-окружения.

Пассивная безопасность

Пассивная безопасность заключается в предоставлении принимающему серверу дополнительной информации об отправителе, благодаря которой он может выбрать оптимальный способ обработки сообщения на основе внутренних политик или предпочтений пользователя. Это может быть указание контактных данных в полях `*-c-email` записей WHOIS домена, дополнительные заголовки в письме или использование SPF / DKIM / DMARC (их, в отличии от TLS, поддерживают все крупные почтовые сервисы).

Использование SPF

Использование технологии SPF позволяет указать принимающему серверу список адресов хостов, которые имеют право отправлять почту от имени домена. Для использования SPF достаточно на DNS, обслуживающем домен отправителя, завести TXT запись вида:

```
example.com. IN TXT "v=spf1 a mx -all"
```

Более подробно с полями записи SPF можно ознакомиться на сайте openspf.org. В данном примере мы разрешили принимать почту со всех серверов, на которые указывают `A` и `MX` записи домена и запретили для всех остальных (`-all`).

При использовании SPF следует обратить внимание на следующие моменты:

- Использование SPF является лишь **рекомендацией** для принимающей стороны — далеко не все почтовые сервисы отвергнут письмо, пришедшее с «чужого» IP адреса только на основании того, что в записи установлено значение `-all`.
- Если MX записи домена указывают на IPv6 адреса, то при проверке они **не** разворачиваются механизмом `mx` и эти адреса требуется перечислить в явном виде через механизм `ip6` (точно так же, механизм `a` задает только IPv4 адреса домена, но не IPv6).
- Если длина SPF записи превышает 1000 байт (приблизительно), рекомендуется разбить запись на части с использованием механизма `include`.

Использование DKIM

Технология DKIM позволяет подтвердить отправителя сообщения и неизменность содержимого сообщения. Для автоматической подписи всех исходящих сообщений на сервере должен быть установлен `dkim-filter`. Конфигурация фильтра задается в файле `/etc/dkim-filter.conf` в виде:

```
Syslog          yes
Selector        default
Mode            s
InternalHosts   /etc/mail/dkim-hosts.conf
KeyList         /etc/mail/dkim-keys.conf
Socket          inet:10034@127.0.0.1
```

Здесь:

- `Syslog` — вести лог работы фильтра;
- `Selector` — селектор по умолчанию при использовании `KeyList`;
- `Mode` — режим работы только на подпись сообщения (без проверки подписи);
- `InternalHosts` — список хостов, для которых должны подписываться сообщения (вместо проверки);
- `KeyList` — имя файла с конфигурацией ключей для подписи сообщений (см. ниже);
- `Socket` — адрес и порт, на котором фильтр будет ожидать входящие соединения (существует возможность задать имя файл-сокета, но при работе Postfix в chroot это потребует дополнительных настроек).

Файл `dkim-hosts.conf` представляет собой список имен хостов или IP адресов (по одному на каждую строку) исходящая почта с которых будет подписываться фильтром. В простейшем случае:

```
localhost
example.com
```

Файл `dkim-keys.conf` представляет собой набор строк с указанием (через двоеточие) маски отправителя, имени подписывающего домена и имени файла ключа для подписи. Например:

```
*@example.com:example.com:/etc/mail/key/default
```

Здесь письма с отправителем из домена `@example.com` будут подписываться подписью домена example.com с селектором `default` (был задан ранее в конфигурации), ключ которого задан в файле `default`.

Для создания приватного и публичного ключей DKIM необходимо, находясь в директории `/etc/mail/key`, воспользоваться утилитой `dkim-genkey`. Запуск утилиты без параметров создаст пару файлов: `default.private` и `default.txt`, которые являются приватным ключом и текстом TXT записи, которую требуется добавить в DNS. Файл `default.private` необходимо переименовать в `default`.

В файле `default.txt` приведено содержимое TXT записи для DNS. Дополнительно, рекомендуется создать следующие записи:

```
default._domainkey.example.com. IN TXT "v=DKIM1; g=*; k=rsa; ..."
_adsp._domainkey.example.com.   IN TXT "dkim=discardable"
_domainkey.example.com.         IN TXT "o=-; r=admin@example.com"
```

Здесь:

- `default._domainkey` — публичный DKIM ключ для селектора default домена example.com (подробности в RFC 4870);

- `_adsp._domainkey` — подпись должна быть сделана тем же доменом, который указан в адресе отправителя, в противном случае получатель может отбросить письмо (второе допустимое значение `all`, которое является более мягким, подробности в RFC 5617);
- `_domainkey` — политика подписи (`o=-` — подписываются все сообщения, `r=` — адрес для отчетов о невалидной подписи).

Для включения подписи исходящих сообщений в Postfix необходимо добавить конфигурацию:

```
smtpd_milters     = inet:127.0.0.1:10034
non_smtpd_milters = inet:127.0.0.1:10034
```

Здесь smtpd_milters и non_smtpd_milters задают адрес и порт фильтра `dkim-filter` для сообщений проходящих через сервер smtpd и локальных сообщений, отправленных через утилиту `sendmail`.

При использовании фильтра DKIM следует обратить внимание на формат перевода строк в теле письма — фильтр требует перевода строк в стиле unix (`\n`) в то время как SMTP требует окончание строк CRLF — необходимые преобразования производятся автоматически.

Использование DMARC

Использование технологии DMARC позволяет явно указать получателю письма рекомендуемые действия в случае, если письмо не проходит проверку SPF и/или DKIM и получать отчеты о применении технологии почтовым сервисом. Так же, как и предыдущие техники, политика конфигурируется через TXT запись на DNS домена или поддомена в виде:

```
example.com. IN TXT "v=DMARC1; aspf=s; adkim=s;
    p=reject; sp=reject; rua=mailto:admin@example.com"
```

В данном примере мы включаем строгую проверку SPF и DKIM записи для домена (подпись должна быть сделана тем же доменом, с которого отправлено сообщение) и рекомендуем отклонять все сообщения для домена и всех поддоменов в случае ошибки проверки с отсылкой отчетов на адрес `admin@example.com` (к сожалению, отсылка отчетов не работает ни у кого — возможно в будущем).

С примерами записей и назначением полей можно ознакомиться в справочных системах mail.ru, google.com или на сайте инициаторов инициативы dmarc.org.

Использование дополнительных заголовков

Сообщения, генерируемые автоматически, так или иначе попадают под определение «массовые рассылки». Для массовых рассылок у крупных почтовых сервисов устанавливаются свои «правила хорошего тона» соблюдая которые можно улучшить процент доставки и **прочтения** сообщений получателями. С общими требованиями можно ознакомиться в документе Senders Best Communications Practices (PDF). С требованиями основных почтовых сервисов можно ознакомиться по ссылкам:

- mail.ru;
- yandex.ru;
- gmail.com;

Из технических требований к массовым рассылкам можно выделить следующие пункты:

- Ящик отправителя должен существовать (об этом говорилось ранее).
- Должны использоваться SPF / DKIM / DMARC (о которых говорилось ранее).
- Рассылку рекомендуется вести с выделенного домена (например, robot.example.com вместо example.com) — это позволит разделить личную и автоматическую корреспонденцию как для удобства получателей так и для собственного (личная корреспонденция на основном домене может обслуживаться сторонним сервисом, например Google Apps).
- Сервер должен иметь валидную PTR запись (запись, которая позволяет получить имя хоста по IP адресу). Обычно обслуживанием зоны `in-addr.arpa` занимается техподдержка хостера, однако, современные хостеры (типа Hetzner) позволяют управлять PTR записями из панели управления сервером. Для проверки PTR записи сервера можно воспользоваться утилитой `nslookup` указав в качестве аргумента требуемый IP адрес.
- Каждое сообщение должно снабжаться заголовками `Precedence: bulk` и `List-Unsubscribe`. Подробнее с форматом заголовка `List-Unsubscribe` можно ознакомиться на сайте list-unsubscribe.com — в общем случае его значением является URL, который позволяет отписать пользователя без совершения дополнительных действий (типа авторизации на сайте) и/или почтовый адрес для отсылки уведомления о желании пользователя отписаться от рассылки. Параметрами URL могут являться идентификатор пользователя и некоторый хэш, проверка которого позволяет совершить требуемое действие/
- Принадлежность к массовым автоматическим рассылкам указывается заголовками `Precedence: bulk` и `Auto-Submitted: auto-generated`.

Дополнительно, имеет смысл зарегистрировать почтовый домен в сервисе postmaster.mail.ru для отслеживания ошибок конфигурации и FBL. При использовании данного сервиса может оказаться полезным использование заголовка X-Mailru-Msgtype.

Проверка валидности e-mail

При регистрации на сайте и осуществлении подписки бывает важно проверить валидность указанного e-mail адреса как на соответствие стандартам, так и на возможные опечатки (например, mail.ry вместо mail.ru) или наличие домена в списке блокируемых (например, mailinator.com).

В случае отсутствия проверки на этапе регистрации, отсылку почты на домены с опечатками и домены без авторизации можно блокировать через карту виртуальных адресов `virtual_alias_maps` как это было описано ранее для bounce сообщений.

Почтовый адрес

RFC 2822 гласит:

- Почтовый адрес состоит из имени и домена, разделенных символом «@» (раздел 3.4.1);
- Имя может состоять из букв, цифр и символов: «!», «#», «$», «%», «&», «'», «*», «+», «-», «/», «=», «?», «^», «_», «`», «{», «|», «}», «~»: (раздел 2.3.4), а так же символа «.» (точка) за исключением того, что имя не может начинаться с точки, заканчиваться на точу или содержать две точки подряд;
- Имя может быть заключено в двойные кавычки «"» и содержать уже любые символы (раздел 3.2.5);
- Максимальная длина имени составляет 64 символа;
- Имя регистро-зависимо за исключением служебного имени postmaster (RFC 5321, раздел 4.1.1.3);
- Для совместимости с устаревшим RFC-822, адрес может быть заключен в треугольные скобки и содержать символ «@» (раздел 4.4).

Имя домена

RFC 1035 гласит:

- Общая длина имени домена ограничена 255 символами, длина имени узла ограничена 63 символами (раздел 2.3.4);
- Имя узла может состоять из букв, цифр и символа «-» за исключением начала и окончания;
- Узлы разделяются символом «.» (точка);
- Имя домена регистро-независимо;
- Имя домена может заканчиваться на символ «.» (точка), подразумевая корневой узел, хотя обычно последняя точка не пишется (подробнее см. FQDN).

У кого длиннее?

- В качестве примера использования потенциала длины имени можно привести проект, где предлагается бесплатно получить самый длинный email адрес и почувствовать на себе все прелести обработки такого адреса веб-формами, почтовым программным обеспечением, системами антиспама и т.д.;
- Максимальная длина почтового адреса, согласно RFC 5321 (раздел 4.5.3.1.3) составляет 256 символов (при этом остается ограничение на длину имени в 64 символа и длину домена в 255 символов).

Примеры

RFC 3696 дает нам несколько занимательных примеров **валидных** почтовых адресов:

```
<@hosta.int,@jkl.org:userc@d.bar.org>
jsmith@[192.168.2.1]
Abc\@def@example.com
Abc\\@example.com
Abc\\\@def@example.com
Fred\ Bloggs@example.com
Joe.\\Blow@example.com
"Abc@def"@example.com
"Fred Bloggs"@example.com
user+mailbox@example.com
customer/department=shipping@example.com
$A12345@example.com
!def!xyz%abc@example.com
_somename@example.com
```

Несмотря на валидность адресов в примере выше, большинство почтовых систем имеют свои ограничения на имя пользователя и пропуск «экзотических» форм адресов в качестве валидных может приносить больше вреда, чем пользы, усложняя логику проверок и внося дополнительные ошибки.

Что делать?

- Регулярное выражение из Perl модуля Mail::RFC822::Address для проверки адреса по RFC 822;
- Email::Valid — модуль Perl, поддерживающий проверку по RFC 822 и дополнительные проверки (существование MX и т.д.);
- Email::Address — еще один модуль Perl, но уже для проверки по RFC 2822;
- Zend_Validate_EmailAddress — класс из ZendFramework;
- Облегченный парсер для PHP (без проверки MX и т.д.);
- Его же версия для Ruby и Python.

Анализ bounce сообщений

Вместо удаления bounce сообщений в `/dev/null`, как это было предложено ранее, может быть полезным их автоматический анализ с целью блокировки несуществующих адресов или предотвращения повторной отправки на несуществующий адрес для улучшения репутации в глазах крупных почтовых систем.

По историческим причинам bounce сообщения формируются в формате, удобном для анализа человеком, но не машиной. Несмотря на существование стандарта RFC 3464,

который призван решить эту проблему, современное состояние дел далеко от идеала и разные почтовые сервера продолжают использовать собственные форматы.

Для получения возможности анализировать тело сообщения, вернувшегося на адрес отправки (например, `noreply@example.com`), можно воспользоваться все той же таблицей `virtual_alias_maps` указав в качестве приемника не адрес получателя, а скрипт обработки сообщения:

```
noreply:    "|/etc/postfix/scripts/noreply.sh"
```

Скрипт обработки (в примере `noreply.sh`) должен принадлежать пользователю `mail` (`chown mail:mail noreply.sh`) и иметь бит исполнения (`chmod 0555 noreply.sh`). Содержимое скрипта можно представить в виде:

```
#!/bin/bash

PID_TEMP_FILE="/tmp/noreply@example.com.$$.tmp"

cat > "${PID_TEMP_FILE}"

TEMP_FILE=$(egrep -i '^Message-Id:' "${PID_TEMP_FILE}" | \
    /usr/bin/head -n 1 | /usr/bin/awk '{print $2;}' |    \
    /usr/bin/cut -d '<' -f 2 | /usr/bin/cut -d '>' -f 1)

TEMP_FILE="/tmp/bounce/${TEMP_FILE}"

if [ ! -d "/tmp/bounce" ]; then
    mkdir "/tmp/bounce"
    if [ $? -ne 0 ]; then
        rm -f "${PID_TEMP_FILE}"
        exit 0
    fi
fi

mv -f "${PID_TEMP_FILE}" "${TEMP_FILE}"

if [ $? -ne 0 ]; then
    rm -f "${PID_TEMP_FILE}"
    exit 0
fi

if [ -f "/etc/postfix/scripts/analyze.sh" ]; then
    /etc/postfix/scripts/analyze.sh "${TEMP_FILE}" 2>&1 > /dev/null
fi
```

Приведенный в примере скрипт совершает следующую последовательность действий:

- Сохраняет тело сообщения во временный файл (для получения дополнительной информации);

- Получает из временного файла значение поля `Message-Id` (см. далее);
- Проверяет наличие директории для хранения файлов `/tmp/bounce` и создает ее в случае необходимости;
- Перемещает временный файл в файл с именем значения поля `Message-Id` (для простоты дальнейшей диагностики и сохранения уникальности имени файла);
- Запускает скрипт анализа содержимого `analyze.sh`, передавая в качестве параметра имя файла с телом сообщения и подавляя любой вывод ошибок;
- При отсутствии скрипта `analyze.sh` содержимое сообщения просто будет сохраняться в директории `/tmp/bounce` в файле с уникальным именем и пригодно для дальнейшего анализа.

Сам скрипт анализа содержимого сообщения может быть достаточно нетривиальным и качество его работы зависит исключительно от мастерства автора скрипта (дополнительно можно попробовать использовать готовые анализаторы типа Mail::DeliveryStatus::BounceParser). В качестве ориентиров можно посоветовать использовать поиск значений заголовков:

- `X-AutoReply` — автоответчик mail.ru (не является bounce сообщением и должно игнорироваться);
- `X-Autogenerated` — автоответчик rambler.ru (так же не является bounce сообщением);
- `Diagnostic-Code` — собственно, описание причины возврата.

В результате работы анализатора должно приниматься решение о (не)возможности или (не)целесообразности дальнейшей отправки сообщений данному адресату. При этом, следует проявлять осторожность, т.к. формально злоумышленнику ничто не мешает сгенерировать нужное количество fake-bounce сообщений и заблокировать всю базу пользователей.

Наиболее удобным способом хранения подобной информации является ее хранение в базе данных общего назначения (MySQL, PostgreSQL). Для обеспечения возможности Postfix делать запросы в базу данных требуется установка дополнительного пакета (например, `postfix-pgsql` для Debian) или сборка Postfix с поддержкой соответствующих возможностей (для FreeBSD).

Для блокирования отправки сообщений получателям из базы данных требуется подключить дополнительную карту виртуальных адресов:

```
virtual_alias_maps =
    hash:/etc/postfix/virtual,
    proxy:pgsql:/etc/postfix/virtual.cf
```

Сам файл `virtual.cf` представляет собой файл конфигурации с параметрами соединения с базой данных и целевым SQL запросом (список поддерживаемых баз и параметры конфигурации описаны в DATABASE_README):

```
hosts = psql.example.com
user = bouncemail
password = ******
dbname = bouncemail
query = SELECT 'devnull' FROM bouncemail WHERE email = '%s'
```

В приведенном примере SQL запроса все сообщения на почтовые адреса, которые присутствуют в таблице `bouncemail` базы, будут отправлены в алиас `devnull` (который, в свою очередь, развернется в `/dev/null` — «шредер»). Управление списком адресов в базе или критериями выборки остается на усмотрение владельца почтового сервера.