

# Настраиваем TLS/SASL-шифрование и аутентификацию в MTA Sendmail

*Андрей Маркелов*

При помощи механизма STARTTLS и сертификатов мы попробуем настроить взаимную аутентификацию и шифрование пересылаемой почты между двумя почтовыми серверами с MTA Sendmail. Также при помощи TLS защитим отправляемую клиентами на сервер почту и настроим почтовый релей для пересылки корреспонденции только тех пользователей, которые предъявят сертификат, выданный нашим центром сертификации (CA).

**S**TARTTLS (RFC 2487) является расширением протокола SMTP. STARTTLS в первую очередь предназначен для поддержки TLS-шифрования и аутентификации между двумя почтовыми серверами. Но как вы увидите далее, он с успехом может применяться и почтовыми клиентами.

MTA Sendmail в нашей тестовой конфигурации будет работать на двух серверах Red Hat Enterprise Linux 5.1. Выбор не является принципиальным – вы можете использовать ваш любимый или вовсе установить Sendmail в другой операционной системе, например Solaris. Единственное, с чем вы можете столкнуться, – отличие поставляемого по умолчанию в вашем дистрибутиве конфигурационного файла `sendmail.mc` от использующегося в Red Hat Enterprise Linux/Fedora.

Также предполагаем, что у нас есть настроенный центр сертификации. Для простоты будем использовать OpenSSL, но, естественно, это непринципиально. Один из наших тестовых серверов прекрасно справится с этой ролью. Конечно, в реальной работе вы не должны разворачивать CA и публичный почтовый сервер на одной машине.

В качестве «компенсации» засилья Open Source-инструментов в роли MUA используем штатную в Windows Vista программу Windows Mail.

## Базовые настройки и взаимодействие двух почтовых серверов

Начнем с сертификата для MTA. Необходимо отметить, что STARTTLS в данном случае защищает только трафик между двумя соответствующим образом настроенными почтовыми серверами. Нет никакой гарантии при отправке письма «во вне», что все релеи, через которые пройдет письмо, используют STARTTLS. Никогда не надо забывать, что отправка письма через Интернет аналогична отправке почтовой карточки по обыкновенной почте. Текст вашего письма может быть прочитан во всех точках, через которые пересылается письмо. Для гарантированной защиты электронной почты между отправителем и получателем необходимо использовать шифрование при помощи S/MIME или GnuPG/PGP.

Генерируем для нашего сервера приватный ключ и создаем запрос в центр сертификации:

```
# openssl genrsa 1024 > sendmail.key
# openssl req -new -key sendmail.key -out sendmail.csr
```

После чего необходимо переслать запрос на выдачу сертификата в центр сертификации. Администратор центра сертификации выпускает на основе запроса сертификат:

```
# openssl ca -in sendmail.csr -out sendmail.crt
```

Копируем полученный сертификат на почтовый сервер и кладем рядом с ключом в директорию `/etc/pki/tls/certs`. Права на оба файла должны быть выставлены 600 или 400.

Далее начинаем править конфигурационный файл `/etc/mail/sendmail.mc`. Редактируем пути и убираем комментарии со строк:

```
# Директория с сертификатами
define(`confCACERT_PATH', `/etc/pki/tls/certs')dnl

# Сертификат CA, выдавшего сертификат нашему серверу.
# Не забудьте его также скопировать в указанную директорию
define(`confCACERT', `/etc/pki/CA/cacert.pem')dnl

# Сертификат нашего сервера, используемый во время
# приема почты
define(`confSERVER_CERT', `
    `/etc/pki/tls/certs/sendmail.crt')dnl

# Приватный ключ нашего сервера
define(`confSERVER_KEY', `
    `/etc/pki/tls/certs/sendmail.key')dnl
```

и добавляем две новых строки:

```
# Сертификат и приватный ключ нашего сервера, используемый
# для отправления почты на другой сервер. Для простоты
# используем те же файлы, что и в первом случае
define(`confSERVER_CERT', `
    `/etc/pki/tls/certs/sendmail.crt')dnl
define(`confSERVER_KEY', `
    `/etc/pki/tls/certs/sendmail.key')dnl
```

Для проверки наших настроек зайдём на 25-й порт сервера при помощи команды `telnet`:

```
# openssl ca -in sendmail.csr -out sendmail.crt
```

```
Trying 192.168.0.17...
Connected to station17.example.com (192.168.0.17).
Escape character is '^]'.
220 station17.example.com ESMTP Sendmail 8.13.8/8.13.8; Thu, 31
Jul 2008 14:21:33 +0400
```

```
EHLO station18.example.com
```

```
250-station17.example.com Hello station18.example.com
[192.168.0.18], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-STARTTLS
250-DELIVERBY
250 HELP
```

```
STARTTLS
```

```
220 2.0.0 Ready to start TLS
```

```
QUIT
```

Повторяем те же действия по настройке (включая выпуск второго сертификата) для второго сервера. В нашей ситуации оба сервера используют один и тот же удостоверяющий центр. Если используются разные CA, то MTA будут необходимы корневые сертификаты обоих.

Теперь если мы попробуем на одном из серверов локально запустить MUA, например `mutt`, и отправить сообщение пользователю второго сервера, то на первом сервере в `/var/log/maillog` увидим примерно такие сообщения:

```
Jul 31 01:07:21 station18 sendmail[9558]: STARTTLS=client,
relay=station17.example.com., version=TLSv1/SSLv3, verify=OK,
cipher=DHE-RSA-AES256-SHA, bits=256/256
Jul 31 01:07:21 station18 sendmail[9558]: m6UL7Lpc009556:
to=<andrey@station17.example.com>, ctladdr=root@station18.example.com> (0/0), delay=00:00:00, xdelay=00:00:00, mailer=esmtpp,
pri=120467, relay=station17.example.com. [192.168.0.17],
dsn=2.0.0, stat=Sent (m6UL7LpK009193 Message accepted for
delivery)
```

А на принимающем соответственно:

```
Jul 31 01:07:21 station17 sendmail[9193]: STARTTLS=server,
relay=station18.example.com [192.168.0.18], version=TLSv1/SSLv3,
verify=OK, cipher=DHE-RSA-AES256-SHA, bits=256/256
Jul 31 01:07:21 station17 sendmail[9193]: m6UL7LpK009193:
from=<root@station18.example.com>, size=748, class=0, nrcpts=1,
msgid=<20080730210721.GA9548@station18.example.com>, proto=ESMTP,
daemon=MTA, relay=station18.example.com [192.168.0.18]
Jul 31 01:07:21 station17 sendmail[9194]: m6UL7LpK009193:
to=<andrey@station17.example.com>, delay=00:00:00, xde-
lay=00:00:00, mailer=local, pri=31037, dsn=2.0.0, stat=Sent
```

Как видно из журнала, серверы аутентифицировали друг друга при помощи сертификатов, выданных одним и тем же CA, и при пересылке письма информация была зашифрована. Данный факт можно проверить, запустив на одном из тестовых серверов утилиту Wireshark или tcpdump.

## Настройка TLS для клиента

В текущем состоянии Sendmail готов работать с клиентом, принимая от него почту в зашифрованном виде, однако никакой проверки клиента при помощи механизма TLS не используется. Зато клиенту не нужны сертификаты, и всего лишь достаточно поставить соответствующую галку – «использовать защищенное соединение, TLS».

В журнале почтового сервера это выглядит примерно так:

```
Jul 30 23:29:09 station17 sendmail[8228]: STARTTLS=server,
relay=station51.example.com [192.168.0.51], version=TLSv1/SSLv3,
verify=NO, cipher=DHE-RSA-AES256-SHA, bits=256/256
Jul 30 23:29:09 station17 sendmail[8228]: m6UJT8GH008228:
from=<andrey@station17.example.com>, size=341, class=0, nrcpts=1,
msgid=<4890C39D.4020404@station17.example.com>, proto=ESMTP,
daemon=MTA, relay=station51.example.com [192.168.0.51]
Jul 30 23:29:09 station17 sendmail[8229]: m6UJT8GH008228:
to=<root@station17.example.com>, ctladdr=<andrey@station17.example.com> (507/508), delay=00:00:00, xdelay=00:00:00, mailer=local,
pri=30623, dsn=2.0.0, stat=Sent
```

Как видно, напротив verify мы видим значение «NO». Попробуем «закрутить гайки».

Сгенерируем сертификат для пользователя. Раз мы начали работать с OpenSSL, для упрощения задачи предположим, что наш пользователь также воспользуется этой утилитой. Естественно, какими средствами будет сгенерирован запрос в центр сертификации, нам не важно. В «боевой» среде скорее всего пользователь воспользуется каким-либо веб-интерфейсом.

Итак, от лица пользователя:

```
# openssl genrsa 1024 > andrey.key
# openssl req -new -key andrey.key -out andrey.csr
```

И от лица администратора CA:

```
# openssl ca -in andrey.csr -out andrey.crt
```

Далее необходимо «упаковать» ключ и сертификат в формат PKCS#12:

```
# cat andrey.key andrey.crt > andrey.pem
# openssl pkcs12 -export -in andrey.pem -out testusercert.p12 -name "Andrey's Personal cert"
```

«Скармливаем» персональный сертификат и сертификат CA выбранному MUA, в нашем случае программе Windows Mail.

А на сервере остается прописать, кому разрешена пересылка почты через наш сервер:

```
# cat /etc/mail/access
```

```
Connect:localhost.localdomain RELAY
Connect:localhost RELAY
Connect:127.0.0.1 RELAY
CERTISSUER:/C=GB/ST=Berkshire/L=Newbury/O=My+20Company+20Ltd/CN=Station+2018+20CA RELAY
CERTSUBJECT:/C=GB/ST=Berkshire/L=Newbury/O=My+20Company+20Ltd/CN=Station+2018+20CA RELAY
```

В данном случае мы разрешаем пересылку только обладателям сертификатов, выданных нашим CA.

Параметры CERTISSUER и CERTSUBJECT мы заполняем значениями, взятыми из полей Issuer и Subject сертификата CA:

```
# openssl x509 -in cacert.pem -noout -subject -issuer
```

```
subject= /C=GB/ST=Berkshire/L=Newbury/O=My Company Ltd/CN=Station 18 CA
issuer= /C=GB/ST=Berkshire/L=Newbury/O=My Company Ltd/CN=Station 18 CA
```

При этом ряд символов, в том числе «{», «<», «>», «()», «()», «()», «+», «}», и пробел заменяются на шестнадцатеричные ASCII-коды, предваряемые символом «плюс». Соответствующие коды можно посмотреть в странице руководства:

```
# man ascii
```

Успешная попытка обладателя сертификата использовать наш почтовый сервер в качестве реля в журнале /var/log/messages будет выглядеть примерно следующим образом:

```
Jul 31 02:10:31 station17 sendmail[9700]: STARTTLS=server,
relay=station51.example.com [192.168.0.51], version=TLSv1/SSLv3,
verify=OK, cipher=AES128-SHA, bits=128/128
Jul 31 02:10:31 station17 sendmail[9700]: m6UMAVTH009700:
from=<andrey@station17.example.com>, size=1128, class=0, nrcpts=1,
msgid=<57A621A4BFCB411F9ACDE6A1D9CE3E62@AndreyPC>, proto=ESMTP,
daemon=MTA, relay=station51.example.com [192.168.0.51]
Jul 31 02:10:31 station17 sendmail[9702]: STARTTLS=client,
relay=station18.example.com, version=TLSv1/SSLv3, verify=OK,
cipher=DHE-RSA-AES256-SHA, bits=256/256
Jul 31 02:10:31 station17 sendmail[9702]: m6UMAVTH009700:
to=<root@station18.example.com>, ctladdr=<andrey@station17.example.com> (507/508), delay=00:00:00, xdelay=00:00:00, mailer=esmtpl,
pri=121128, relay=station18.example.com [192.168.0.18],
dsn=2.0.0, stat=Sent (m6UMAVR4009739 Message accepted for delivery)
```

Если вы увидели подобные сообщения в журнале, значит наша цель достигнута. Убедиться в том, что соответствующий сеанс защищен, вы можете при помощи утилиты Wireshark.

Таким образом, как видите, при всей относительной сложности настройки MTA Sendmail, механизм STARTTLS включается достаточно просто.

Удачи. 