Почтовый сервер Postfix + Courier-Imap + SASL2 + TLS + PostgreSQL + Squirrelmail + ClamAV + Greylist

Вместо предисловия

Данная статья составлена из заметок, писавшихся для самого себя по материалам различных статей, с целью собрать все необходимое лично мне в одном месте. Так что на копирайты и оригинальность не претендую. Однако считаю, что сей манускрипт все же может оказаться полезным, особенно для начинающих. Прежде чем приступать к настройке по данной статье, рекомендую внимательно прочитать ее до конца. От чтения документации к устанавливаемым программам данная статья не освобождает, так как описание многих параметров конфигурационных файлов остались в тени. Также рекомендую ознакомиться с использованными в статье материалами, список которых дан в конце текста.

1. Предварительная настройка системы

Почтовый сервер, создание которого описывается ниже, был поднят на FreeBSD версии 6.0. Описывать установку и настройку самой ОС я не буду, скажу только о том, что нужно непосредственно для работы почтового сервера.

Первым делом необходимо полностью отключить Sendmail, который ставится по умолчанию вместе с системой. Для этого необходимо проделать следующее. Во-первых, в **/etc/rc.conf** допишем строки:

```
sendmail_enable="NONE"
mta_start_script=""
sendmail_outbound_enable="NO"
sendmail_submit_enable="NO"
sendmail_msp_queue_enable="NO"
```

Во-вторых, необходимо создать файл **/etc/periodic.conf** (если он не существует), и дописать туда следующие строчки:

```
daily_clean_hoststat_enable="NO"
daily_status_mail_reject_enable="NO"
daily_status_include_submit_mailq="NO"
daily_submit_queuerun="NO"
```

Понадобится еще доустановить из пакетов/портов **gmake** и **sudo**. Об их установке также не буду распространяться, из пакетов все ставится за 5 минут и без проблем. Но приведу здесь свой конфиг **sudoers** для дальнейшей работы. У меня он совсем небольшой:

/usr/local/etc/sudoers:

```
Cmnd_Alias POSTGRES = /usr/local/bin/createuser, /usr/local/bin/createdb, \
                      /usr/local/bin/createlang, /usr/local/bin/dropdb, \
                      /usr/local/bin/droplang, /usr/local/bin/dropuser, \
                      /usr/local/bin/initdb, /usr/local/bin/pg_config, \
                      /usr/local/bin/pg_controldata, /usr/local/bin/pg_ctl, \
                      /usr/local/bin/pg_dump, /usr/local/bin/pg_dumpall, \
                      /usr/local/bin/pg_resetxlog, /usr/local/bin/postgres, \
                      /usr/local/bin/postmaster, /usr/local/bin/psql

root        ALL = (ALL) ALL
pgsql       All = POSTGRES
```

Насколько я понимаю, все это обозначает следующее: root может творить все, что хочет, а пользователь pgsql может только запускать программы для работы с PostgreSQL из каталога **/usr/local/bin/.**

2. Установка программ

Все программное обеспечение ставится из портов по нижеприведенной схеме. Я рекомендую ставить приложения именно в перечисленном порядке. Ибо однажды я решил поставить Courier после Postfix, и он отказался собираться. Но если есть желание экспериментировать, то ставьте в том порядке, который вам нравится.

Версии устанавливаемых приложений даны просто для сведения.

2.1. OpenSSL 0.9.7i - из портов

Нужен для создания сертификатов безопасности сервера и клиентов.

cd /usr/ports/security/openssl
make OPENSSL_OVERWRITE_BASE=yes install

2.2. PostgreSQL 8.0.4 - из портов

В базе PostgreSQL будем хранить всю информацию о пользователях нашей почтовой системы, а также иную служебную информацию – транспортные таблицы, алиасы, и т.д.

cd /usr/ports/databases/postgresql80-server
make install

Опции установки – NLS и PAM (последняя нужна только если планируете использовать **saslauthd** и **PAM-PgSQL**)

2.3. CYRUS-SASL 2.1.21 - из портов

Этот порт будет обеспечивать нам SMTP-авторизацию пользователей почтового сервера.

cd /usr/ports/security/cyrus-sasl2
make WITHOUT_OTP=yes WITH_PGSQL=/usr/local/lib install

А вот дальше можно установить демона авторизации, а можно и не делать этого, зависит от того, каким способом будет выполняться обращение к БД пользователей для SMTP-авторизации. Если вы хотите использовать **saslauthd**, то необходимо будет поставить еще и PAM-PgSQL (см. пункт 2.6).
Я лично обошелся без **saslauthd**, но для полноты картины все же расскажу и о нем.

cd /usr/ports/security/cyrus-sasl2-saslauthd
make install

Если **saslauthd** использоваться не будет, то вышеуказанный порт ставить не надо.

2.4. Courier-Imap 4.0.6.1 - из портов

Courier-IMAP будет исполнять роль POP3 и IMAP сервера.

cd /usr/ports/mail/courier-imap
make install

Опции установки - OPENSSL и AUTH_PGSQL
Долго он ставится…

2.5. Postfix 2.2.5 - из портов

Postfix – основа всей нашей системы, собственно MTA.

cd /usr/ports/mail/postfix
make install

Опции установки - SASL2, TLS, PgSQL. Ближе к окончанию установки будет задано еще несколько вопросов, отвечаем на них, используя предлагаемые по умолчанию варианты:

You need user “postfix” added to group “mail”.
Would you like me to add it? – [y]? y
Would you like to activate Postfix in /etc/mail/mailer.conf [n]? n

2.6. PAM-PGSQL 0.6.1 - из портов

Нужен для организации доступа демона авторизации **saslauthd** к базе **PostgreSQL**. Естественно, ставить только в том случае, если вы будете использовать **saslauthd –** тогда установите его, как описано в пункте 2.3. Если **saslauthd** не используется, этот пункт просто пропускаете.

cd /usr/ports/security/pam-pgsql
make install

В ходе экспериментов по установке данной связки сей порт изматерился, что хочет библиотеку **libtool-1.3.5** – хотя уже была установлена **libtool-1.5.20**, и почему-то отказался ее скачивать. Пришлось найти в инете **libtool-1.3.5.tar.gz** и руками подсунуть в **/usr/ports/distfiles**. Возможно, вам это и не грозит.

2.7. PHP 4.4.1 - из портов

Нужен для организации WWW -доступа к нашему почтовому серверу. В процессе установки будет также собран и установлен WWW-сервер **Apache2**.

cd /usr/ports/lang/php4
make install

Опции установки – APACHE2 и OPENSSL

По завершению установки обращаем внимание на то, что инсталлятор предлагает нам добавить в конфиг Апача (**/usr/local/etc/apache2/httpd.conf**) следующие параметры:

AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps

Не лишним будет также добавить к списку файлов, определяющих индексный файл веб-директории по умолчанию еще и файл **index.php**

2.8. Squirrelmail 1.4.5 - из портов

Этот порт, собственно, и обеспечит доступ к нашему почтовому серверу через WWW-интерфейс.

cd /usr/ports/mail/squirrelmail
make install

В процессе установки вылезет окно «Options for php4-mbstring 4.4.1» – в нем оставим все как есть и нажмем «Ok».

По завершению установки опять-таки обратим внимание на рекомендации инсталлятора – нам предлагают изменить в файле **/usr/local/etc/php.ini** (этот файл не существует и нужно будет скопировать его из примера **/usr/local/etc/php.ini-recommended**) два параметра:

file_uploads = On
session.auto_start = 1

2.9. ClamAV 0.87 - из портов

Бесплатный и надежный антивирус, который будет проверять нашу почту на предмет наличия всякой гадости в письмах.

cd /usr/ports/security/clamav
make install

Опции установки – LIBUNRAR

2.10. ClamSMTP 1.5 - из портов

С помощью этого порта мы заставим работать в одной упряжке ClamAV и Postfix

cd /usr/ports/security/clamsmtp
make install

2.11. Postgrey 1.21 - из портов

Система защиты от спама на основе технологии «серых списков».

cd /usr/ports/mail/postgrey
make WITH_BDB_VER=43 install

Вместе с этим портом будет также установлена база данных **BerkeleyDB**, версию которой мы указали параметром **WITH_BDB_VER=43**. Для работы **postgrey** требуется **BerkeleyDB** версии не ниже 4.1.

По этой схеме все прошло без эксцессов. Можно приступать к конфигурированию.

3. Конфигурирование программ

3.1 Настройка PostgreSQL

Сначала с помощью **chsh** поменяем пользователю pgsql оболочку по умолчанию на **/sbin/nologin**. И слегка подточим запускной скрипт PostgreSQL **/usr/local/etc/rc.d/010.pgsql.sh**, который нам сделал порт. Найдем там строчки:

```
postgresql_command()
{
    su -l ${postgresql_user} -c "exec ${command} ${command_args}"
}

postgresql_initdb()
{
    su -l -c ${postgresql_class} ${postgresql_user} -c "exec
${prefix}/bin/initdb -D ${postgresql_data}"
}
```

И поменяем их так, чтобы работало через sudo:

```
postgresql_command()
{
    /usr/local/bin/sudo –u ${postgresql_user} ${command} ${command_args}
}

postgresql_initdb()
{
    /usr/local/bin/sudo -u ${postgresql_user} ${prefix}/bin/initdb -D ${post-
gresql_data}"
}
```

Теперь добавляем в **/etc/rc.conf** строку «**postgresql_enable=”YES**”» (чтобы PostgreSQL стартовал вместе с системой) и запускаем первичную инициализацию базы данных:

/usr/local/etc/rc.d/010.pgsql.sh initdb

И запустим PostgreSQL:

/usr/local/etc/rc.d/010.pgsql.sh start

Далее создадим пользователя БД, от имени которого будет работать Postfix:

sudo –u pgsql /usr/local/bin/createuser

Вводим имя пользователя postfix, на все последующие вопросы отвечаем «n», и вознаграждаемся сообщением CREATE USER. Но этот пользователь у нас имеет доступ к базе данных без пароля, надо это исправить:

sudo –u pgsql /usr/local/bin/psql template1
alter user postfix with password ‘postfix’;

На что нам говорят ALTER USER. Но нам нужен еще и пользователь для работы с самой базой данных, так как пользователю postfix прав на создание записей мы не предоставим. Первая мысль – выйти из psql и повторить вышеописанное createuser… но вот если сначала сделать

select * from pg_shadow;

то увидим, что пользователь pgsql уже существует. Так что на только остается сменить ему пароль:

alter user pgsql with password ‘postgres’;

Не путайте пользователей БД PostgreSQL – postfix и pgsql с одноименными пользователями системы! Это совершенно разные учетные записи, пользователи БД имеют отношение только к базе PostgreSQL.

Выйдем из psql (команда «\q») и настроим **/usr/local/pgsql/data/pg_hba.conf**. Тут надо будет привести строчки в самом конце файла вот к такому виду:

```
TYPE        DATABASE    USER        CIDR             METHOD

local       all         all                          md5
host        all         all         127.0.0.1/32     md5
#host       all         all         ::1/128          trust
```

Последняя строка – для IPv6. Надеюсь, она вам не нужна, поэтому просто закомментируйте ее. Перезапустите PostgreSQL. И наконец, создаем то, ради чего связались с PostgreSQL – нашу базу данных почтовых пользователей:

sudo –u pgsql /usr/local/bin/createdb mailbase

Теперь прежде чем что-то делать, у нас спрашивают пароль. Вводим пароль «postgres» и переходим к созданию таблиц:

sudo –u pgsql /usr/local/bin/psql mailbase

Опять вводим пароль и приступаем:

```
create table user_accounts (
login varchar(32) unique not null,
password varchar(32) unique not null,
smtplogin varchar(128) unique,
smtppassword varchar(128) unique,
email varchar(128) unique not null,
mailbox varchar(256) unique not null,
realname varchar(128));

create table user_aliases (
username varchar(32) not null,
alias varchar(128) not null);

create table transport (
location varchar(128) unique not null,
transport varchar(128) not null default 'virtual:');
```

В награду нам – сообщения CREATE TABLE после каждого ввода «;». Теперь дадим пользователю БД postfix право выполнять оператор select для этих таблиц:

grant select on user_accounts, user_aliases, transport to postfix;

И видим сообщение GRANT – права присвоены. После этого добавим в БД запись для тестового пользователя:

insert into user_accounts values ('testuser', 'password', 'smtptestuser', 'smtppassword', 'testuser@home.net', '/mail/virtual/testuser@home.net/Maildir/', 'Test user');

Если все прошло удачно, то увидим сообщение INSERT <циферки>. Можно полюбоваться на содержимое таблицы mailboxes, выполнив:

select * from user_accounts;

Не помешает также задать запись в таблице transport:

Insert into transport values ('home.net', 'virtual:');

Таблицу алиасов можно заполнить позднее.

Шестое поле в таблице user_accounts описывает местоположение каталога почтового ящика пользователя. На моей системе под ящики пользователей отведен отдельный раздел, который

монтируется как **/mail**. Создадим каталог **/mail/virtual** для хранения почты, и присвоим права на доступ к нему для пользователя postfix и группы maildrop:

mkdir /mail/virtual
chown postfix:maildrop /mail/virtual

Осталось подточить PostgreSQL на предмет логирования обращений к БД, очень пригодится для отладки. Для этого создадим каталог **/var/pgsql**, сделаем пользователя **pgsql** его владельцем и в файле **/usr/local/pgsql/data/postgresql.conf** добавим, раскомментируем или изменим следующие параметры:

```
# все ошибки выдаем на STDERR...
log_destination = 'stderr'
# ... и перенаправляем их в файл
redirect_stderr = true
# каталог, где будет находиться лог-файл
log_directory = '/var/pgsql'
# имя лог-файла
log_filename = 'postgresql.log'
# далее задаем уровень детализации логов
client_min_messages = log
log_min_messages = info
log_min_error_statement = info
# сохраняем в логах информацию о подключениях, отключениях и времени
# выполнения операций
log_connections = true
log_disconnections = true
log_duration = true
# каждую строку лога предваряем текущим временем
log_line_prefix = '%s'
# и выводим в лог полную информацию
log_statement = 'all'
```

На этом с настройкой PostgreSQL можно закончить. Остается только придумать, как вносить записи в БД, не работая непосредственно с почтовым сервером. Вариантов много – доступ через www, или написание клиента БД PostgreSQL для удаленной работы с windows-машины секретарши. Компоненты для работы с PostgreSQL через ODBC или напрямую в природе имеются. А пока можно написать скрипты для добавления пользователей и алиасов непосредственно из shell:

Добавить пользователя – **ins_users.sh**

```
#!/bin/sh
PGPASSWORD=postgres
export PGPASSWORD
sudo -u pgsql /usr/local/pgsql/bin/psql -A -q -t -c \
 "insert into user_accounts values ('$1', '$2', '$3', '$4', '$5', '$6',
'$7')" mailbase
```

Добавим себя, любимого:

./ins_users.sh shrdlu shrdlupass shrdlu@home.net smtppass shrdlu@home.net /mail/virtual/shrdlu@home.net/Maildir/ Sergey_Svyatkin

Добавить алиас – **ins_aliases.sh**

```
#!/bin/sh
PGPASSWORD=postgres
export PGPASSWORD
sudo -u pgsql /usr/local/pgsql/bin/psql -A -q -t -c \
 "insert into user_aliases values ('$1', '$2')" mailbase
```

Добавим алиасы для root, postmaster и abuse:

./ins_aliases root shrdlu@home.net
./ins_aliases postmaster shrdlu@home.net
./ins_aliases abuse shrdlu@home.net

3.2. Настройка Postfix

Идем в **/usr/local/etc/postfix**, и правим файл **main.cf**. Поскольку я намерен выкинуть из него все, что сочту лишним – комментарии и параметры, значения которых я оставлю по умолчанию, то на всякий случай сделаю копию этого файла – **main.cf.old**. В итоге должно получиться следующее:

```
# каталог в котором будет находиться очередь писем, обрабатываемых postfix
queue_directory = /var/spool/postfix
# каталог выполняемых файлов административных команд postfix
command_directory = /usr/local/sbin
# каталог выполняемого файла демона postfix
daemon_directory = /usr/local/libexec/postfix
# Путь к команде, используемой взамен стандартного sendmail
sendmail_path = /usr/local/sbin/sendmail
# Путь к исполняемому файлу, используемому для перестройки псевдонимов
newaliases_path = /usr/local/bin/newaliases
# Путь к команде, отображающей состояние почтовой очереди
mailq_path = /usr/local/bin/mailq
# Документация в html-формате
html_directory = no
# База документации на все устанавливаемые программы
manpage_directory = /usr/local/man
# Примеры конфигурационных файлов postfix.
sample_directory = /usr/local/etc/postfix
# Файлы readme
readme_directory = no
# Имя пользователя, с правами которого работает почта
mail_owner = postfix
# Группа, от имени которой будут работать команды обработки почтовой очереди
setgid_group = maildrop
# Права по умолчанию, используемые агентом локальной доставки.
# Не указывайте здесь привилегированного пользователя или владельца Postfix!
default_privs = nobody
# Имя нашего хоста.
myhostname = mail.home.net
#Имя нашего домена
mydomain = home.net
# Адреса интерфейсов, на которых нужно ждать smtp-соединений
inet_interfaces = all
# Домены, для которых будет приниматься почта
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
# списки наших локальных пользователей:
local_recipient_maps =
# С каким кодом отклонять письма для несуществующих пользователей
unknown_local_recipient_reject_code = 550
# Подсети из которых почта принимается, не проходя многих проверок.
# Перекрывает параметр mynetworks_style. Можете добавить сюда ваши подсети,
# я же в дальнейшем буду использовать принцип «не доверяй никому», поэтому
# указал только локальную подсеть
mynetworks = 127.0.0.0/8
# Списки алиасов, используемых для локальной доставки
alias_maps = hash:/etc/mail/aliases
alias_database = hash:/etc/mail/aliases
# Используем Maildir-style для пользовательских ящиков
home_mailbox = Maildir/
# Строка представления вашего SMTP сервера (может быть любой)
smtpd_banner = $myhostname ESMTP
```

```
# запрещаем использование команды VRFY
disable_vrfy_command = yes
# Обязательно требуем использования команд HELO или EHLO
smtpd_helo_required = yes
```

Это базовая настройка, которую мы будем постепенно дополнять в ходе дальнейших усовершенствований. Теперь самое время выполнить команду **newaliases**, а потом сказать **postfix check**. Если postfix ни на что не отругается можно продолжать править **main.cf**. Добавляем туда:

```
# списки транспортов
transport_maps = pgsql:/usr/local/etc/postfix/transport.cf
# куда складывать почту транспорту virtual
virtual_mailbox_base = /
# списки почтовых ящиков
virtual_mailbox_maps = pgsql:/usr/local/etc/postfix/mailbox.cf
# списки алиасов
virtual_alias_maps = pgsql:/usr/local/etc/postfix/alias.cf
# uid и gid пользователей
virtual_uid_maps = static:125
virtual_gid_maps = static:126
```

Цифры 125 и 126 – это идентификаторы пользователя **postfix** и группы **maildrop** в файлах **/etc/passwd** и **/etc/group** соответственно. Перед тем, как вписывать эти числа, загляните в эти файлы – вдруг у вас они другие. То есть вся почта у нас будет принадлежать одному пользователю **postfix**.

И создаем необходимые вспомогательные файлы:

touch /usr/local/etc/postfix/transport.cf

Вписываем в него:

```
host = localhost
user = postfix
password = postfix
dbname = mailbase
table = transport
select_field = transport
where_field = location
```

touch /usr/local/etc/postfix/mailbox.cf

В него пишем:

```
host = localhost
user = postfix
password = postfix
dbname = mailbase
table = user_accounts
select_field = mailbox
where_field = email
```

touch /usr/local/etc/postfix/alias.cf

В него пишем:

```
host = localhost
user = postfix
password = postfix
dbname = mailbase
table = user_aliases
select_field = alias
```

```
where_field = username
```

Для верности еще раз сделаем **postfix check**. Если все нормально, то запустим Postfix командой **postfix start**, и запустив **telnet localhost 25**, попробуем послать письмо нашему пользователю testuser. Если все пройдет гладко, то мы увидим, что в каталоге **/mail/virtual** создался каталог **/testuser@home.net/Maildir**, а в нем – еще три каталога – **cur**, **new** и **tmp**. Все это говорит о том, что письмо нашло своего героя, и ожидает, пока он заберет его. Так что самое время перейти к настройке SMTP-авторизации. Но перед этим накропаем еще один скрипт для запуска Postfix во время старта системы (почему-то порт такого скрипта не предлагает):

touch /usr/local/etc/rc.d/360.postfix.sh

```
#!/bin/sh
case "$1" in
  start) echo "Starting POSTFIX..."
         /usr/local/sbin/postfix start
         ;;
  stop) echo "Stopping POSTFIX..."
         /usr/local/sbin/postfix stop
         ;;
  restart) echo "Restarting POSTFIX..."
         /usr/local/sbin/postfix reload
         ;;
esac
sleep 3
exit 0
```

3.3. Настройка SASL

3.3.1. Настройка без использования saslauthd.

Создаем файл **/usr/local/lib/sasl2/smtpd.conf** со следующим содержимым:

```
pwcheck_method: auxprop
log_level: 9
auxprop_plugin: sql
sql_engine: pgsql
sql_user: postfix
sql_passwd: postfix
sql_hostnames: 127.0.0.1
sql_database: mailbase
sql_select: select smtppassword from user_accounts where smtplogin='%u'
sql_verbose: yes
```

Строчки **log_level: 9** и **sql_verbose: yes** пригодятся только для отладки, впоследствии их можно удалить.

3.3.2. Настройка с использованием saslauthd и PAM-PGSQL.

Сначала заглянем в **/usr/local/share/doc/pam-pgsql** и почитаем **README**. После осмысления, чего от нас хотят, и чего хотим мы, проделываем следующее. Создаем файл **/usr/local/lib/sasl2/smtpd.conf** со следующим содержимым:

```
pwcheck_method: saslauthd
```

Теперь создаем файл **/etc/pam_pgsql.conf**:

```
database = mailbase
host = localhost
user = postfix
password = postfix
```

```
table = user_accounts
user_column = smtplogin
pwd_column = smtppassword
pw_type = clear
```

Затем создаем **/etc/pam.d/smtp** (именно smtp!):

```
auth        required    pam_pgsql.so
account     required    pam_pgsql.so
password    required    pam_pgsql.so
```

И в заключении, заглянем в **/usr/local/etc/rc.d**. Там нас будет интересовать файл **saslauthd.sh**, запускающий демона авторизации SASL. Заглянув внутрь этого файла, увидим, что нужно добавить в **/etc/rc.conf** строчку

```
saslauthd_enable="YES"
```

что мы и проделаем.

Поскольку все прекрасно работает и без связки **saslauthd/PAM**, то в дальнейшем я буду настраивать систему, опираясь только на способ, описанный в пункте 3.3.1.

3.3.3. Настройка Postfix и проверка работы SMTP-аутентификации.

Допишем в **/usr/local/etc/postfix/main.cf** строки:

```
# включаем плагин sasl2 для smtpd авторизации
smtpd_sasl_auth_enable = yes
# запрещаем анонимную аутентификацию
smtpd_sasl_security_options = noanonymous
# разрешаем авторизацию клиентов, использующих устаревшую версию команды AUTH
broken_sasl_auth_clients = yes
# имя области аутентификации SASL – лучше оставить пустым
smtpd_sasl_local_domain =
# ограничения, выполняющиеся при обработке команды RCPT TO
smtpd_recipient_restrictions = permit_sasl_authenticated,
                               reject_unauth_destination
```

Введем дополнительные ограничения на соединение с нашим сервером:

```
# Проверяем, что нам предъявляют в HELO/EHLO и:
# принимаем всю почту для postmaster и abuse, ибо postmaster обязан
# быть доступным при любом раскладе
smtpd_helo_restrictions = check_recipient_access
hash:/usr/local/etc/postfix/postmaster_access,
# существуют «особы, приближенные к особе» - клиенты, чей почтовый сервер
# настроен криво, но почту с которого надо все же принимать во избежание
# скандалов с руководством - заносим их в «белый» список
                          check_helo_access
hash:/usr/local/etc/postfix/helo_access,
# принимаем, если клиент прошел SASL-аутентификацию
                          permit_sasl_authenticated,
# принимаем, если клиент из доверенной подсети
                          permit_mynetworks,
# отвергаем хосты с неверными именами
                         reject_invalid_hostname,
# отвергаем хосты, которые не имеют A или MX записей в DNS
                          reject_unknown_hostname,
# отвергаем хосты с именами не в FQDN-форме
                          reject_non_fqdn_hostname
```

Не забудем создать файлы **postmaster_access** и **helo_access:**

touch /usr/local/etc/postfix/postmaster_access
touch /usr/local/etc/postfix/helo_access

Вписываем в **postmaster_access**:

```
postmaster@home.net OK
abuse@home.net OK
```

Вписываем в **helo_access**:

```
vasya.pupkin.local OK
```

И делаем

postmap postmaster_access
postmap helo_access

Теперь перезагрузим сервер, и приступим к проверке работоспособности SASL. Для начала, попробуем послать письмо куда-либо **за пределы home.net** (иначе письмо будет принято в любом случае и вы не увидите как работает SASL), на что получим в **/var/log/maillog**:

```
Nov 15 19:53:28 mail postfix/smtpd[668]: connect from unknown[192.168.211.1]
Nov 15 19:53:28 mail postfix/smtpd[668]: NOQUEUE: reject: RCPT from unknown[192.168.211.1]: 554 <shrd-
lu@svgc.ru>: Relay access denied; from=<testuser@home.net> to=<shrdlu@svgc.ru> proto=ESMTP
helo=<192.168.211.1>
Nov 15 19:53:28 mail postfix/smtpd[668]: disconnect from unknown[192.168.211.1]
```

Коротко и ясно. Теперь настроим почтовый клиент на использование SMTP-авторизации и повторим попытку. Видим следующее:

```
Nov 17 15:44:18 mail postfix/smtpd[528]: connect from unknown[192.168.211.1]
Nov 17 15:44:18 mail postfix/smtpd[528]: E08A160FD: client=unknown[192.168.211.1], sasl_method=CRAM-MD5,
sasl_username=smtptestuser
Nov 17 15:44:18 mail postfix/cleanup[536]: E08A160FD: message-id=<19327532578.20051117154102@home.net>
Nov 17 15:44:18 mail postfix/qmgr[445]: E08A160FD: from=<testuser@home.net>, size=679, nrcpt=1 (queue ac-
tive)
Nov 17 15:44:18 mail postfix/smtpd[528]: disconnect from unknown[192.168.211.1]
```

Главное – видим строчку

```
Nov 17 15:44:18 mail postfix/smtpd[528]: E08A160FD: client=unknown[192.168.211.1], sasl_method=CRAM-MD5,
sasl_username=smtptestuser
```

То есть, мы добились желаемого результата, и SASL таки принял нашу авторизацию.

3.4. Настройка Courier-Imap

Для настройки Courier-IMAP нам следует для начала отредактировать **/usr/local/etc/authlib/authdaemonrc**. Поправьте в нем следующие параметры:

```
# список модулей, используемых демоном авторизации
authmodulelist="authpgsql"
# включение вывода отладочной информации:
# DEBUG_LOGIN=2   - включение отладочной информации и логирование паролей.
# нужно только на этапе отладки!
DEBUG_LOGIN=2
```

После этого отредактируем файл **/usr/local/etc/authlib/authpgsqlrc**, с тем, чтобы в нем содержалось следующее:

```
# порт для соединения с PostgreSQL
PGSQL_PORT 5432
```

```
# имя пользователя, который будет работать с БД
PGSQL_USERNAME postfix
# пароль этого пользователя
PGSQL_PASSWORD postfix
# К какой БД подключаемся
PGSQL_DATABASE mailbase
# Имя таблицы в БД
PGSQL_USER_TABLE user_accounts
# Поле таблицы БД, содержащее пароль пользователя
PGSQL_CLEAR_PWFIELD password
# id владельца почтового ящика (у нас postfix)
PGSQL_UID_FIELD 125
# id группы, владеющей ящиком (у нас maildrop)
PGSQL_GID_FIELD 126
# Поле таблицы БД, содержащее логин пользователя
PGSQL_LOGIN_FIELD login
# ? домашний каталог, что ли?
PGSQL_HOME_FIELD mailbox
# Поле таблицы БД, содержащее имя пользователя
PGSQL_NAME_FIELD realname
# Поле таблицы БД, содержащее путь к ящику
PGSQL_MAILDIR_FIELD mailbox
```

Теперь можно запускать Courier. При установке из порта заботливый инсталлятор уже создал для нас все необходимые скрипты запуска всех сервисов Courier-Imap в каталоге **/usr/local/etc/rc.d/**. Нас сейчас интересуют скрипты **courier-authdaemon.sh, courier-imap-pop3d.sh** и **courier-imap-imapd.sh**. Не поленимся заглянуть в них, и увидим, что для автоматического запуска при загрузке сервера нам необходимо всего лишь поместить в **/etc/rc.conf** нижеприведенные параметры:

```
courier_authdaemond_enable="YES"
courier_imap_pop3d_enable="YES"
```

Если вместо сервера POP вы планируете использовать сервер IMAP, то добавьте в **/etc/rc.conf**

```
courier_imap_imapd_enable="YES"
```

вместо

```
courier_imap_pop3d_enable="YES"
```

А в общем-то, ничто не мешает использовать POP и IMAP одновременно. Проделав это, перезагружаем сервер (или стартуем вышеуказанные сервисы вручную), и пробуем проверить почту с помощью **telnet localhost 110** (143 для IMAP) или через почтовый клиент**.**

3.5. Настройка TLS-SSL

Сразу скажу, что реализация данной части – дело тяжелое. Потому что будет очень много рутинной возни, если делать все по правилам.

Первым делом нам понадобится доверенный сертификат – **Certificate Authority** (далее **CA**), чтобы иметь возможность подписывать и проверять клиентские сертификаты. Если все делать по правилам, то такой сертификат нужно создать, а затем подписать его у одного из корневых доверенных центров сертификации. Но это стоит денег. Если ваша организация заинтересована, скажем, в развитии своего представительства в Internet, то именно это и стоит сделать. Но если вы всего лишь небольшая фирма, которая нуждается всего-навсего в почтовом сервере, то можно обойтись самоподписанным доверенным сертификатом.

Итак, приступаем. Для начала определимся с местом, где у нас будут лежать наш сертификат и сертификаты пользователей. Поразмыслив, я решил держать их в каталоге **/usr/local/ssl**, но вы,

возможно, решите, что есть и более подходящее для них место – оставляю это на ваше усмотрение. Внутри этого каталога проделаем следующее:

mkdir db
mkdir ca
mkdir clients
touch /usr/local/ssl/db/index.txt
echo "01" > /usr/local/ssl/db/serial

Скрипт для создания нашего самоподписанного доверенного сертификата (CA):

touch /usr/local/ssl/make_ca.sh

```
#!/bin/sh
openssl req -new -newkey rsa:4096 -nodes -keyout ./ca/ca.key -x509 -days 3650
\
        -subj /C=RU/ST=Russia/L=Samara/O=HOMENET\
Inc/OU=IT/CN=mail.home.net/emailAddress=postmaster@home.net \
        -out ./ca/ca.crt
```

Запускаем скрипт **make_ca.sh**. Какое-то время он будет возиться с генерацией секретного ключа длиной в 4096 бит, на слабой машине это займет время… можете уменьшить длину ключа до 1024 или 2048 бит, если очень не терпится все поскорее попробовать. В итоге мы получим два файла в каталоге **/usr/local/ssl/ca**: **ca.crt** – это наш самоподписанным доверенный сертификат, и **ca.key** – его секретный ключ. Теперь нам необходим скрипт для создания клиентских сертификатов:

touch /usr/local/ssl/make_cleint_cert.sh

```
#!/bin/sh
openssl req -new -newkey rsa:4096 -nodes -keyout ./clients/client_$1.key \
      -subj /C=RU/ST=Russia/L=Samara/O=HOMENET/OU=IT/CN=$1/emailAddress=$2 \
        -out ./clients/client_$1.csr;
openssl ca -config ca.config -in ./clients/client_$1.csr -out
./clients/client_$1.crt -batch;
openssl pkcs12 -export -in ./clients/client_$1.crt -inkey
./clients/client_$1.key \
        -certfile ./ca/ca.crt -out ./clients/client_$1.p12 -passout pass:$3
```

Прежде чем запускать этот скрипт, создадим файл конфигурации для подписывания запросов на сертификацию:

touch /usr/local/ssl/ca.config

```
[ ca ]
default_ca              = CA_CLIENT
[ CA_CLIENT ]
dir                     = /usr/local/ssl
certs                   = $dir/clients
new_certs_dir           = $dir/db
database                = $dir/db/index.txt
serial                  = $dir/db/serial
certificate             = $dir/ca/ca.crt
private_key             = $dir/ca/ca.key
default_days            = 3650
default_crl_days        = 3
default_md              = md5
policy                  = policy_anything
[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
```

```
localityName            = optional
organizationName        = optional
organizationalUnitName = optional
commonName              = supplied
emailAddress            = optional
```

Скрипт **make_ca.sh** будет использоваться крайне редко – для первого создания нашего CA, и для генерации нового по истечению срока действия текущего CA. Однако, опцией «**-days**» мы устанавливаем срок действия сертификата аж на 10 лет, так что часто это делать не придется, разве что в случае компрометации сертификата (тьфу-тьфу-тьфу!).

Скрипт **make_cleint_cert.sh** будет использоваться нами гораздо чаще, чем хотелось бы – если все делать по правилам. То есть, мы должны выдать уникальный сертификат каждому сотруднику, имеющему почтовый ящик на нашем сервере. Причем выдать его на не слишком длительный срок, в идеале – месяца на 3, но во всяком случае не больше, чем на год. А по прошествии этого срока выдать всем новые сертификаты. Прибавьте к этому увольняемых и нанимаемых сотрудников… в общем, если у вас много народу – этим лучше заниматься специальной группе security officer's. Зато при этом у вас будет полноценный шифрованный канал с авторизацией и аутентификацией пользователей.

Но можно поступить и проще. Создать всего лишь один клиентский сертификат, не на персону, а на всю контору, и использовать его. И можно еще дополнительно облегчить себе жизнь, задав ему срок действия, равный сроку действия CA. Лет эдак на 20. Все это, естественно, будет делаться в ущерб безопасности, так как ни о какой аутентификации пользователей речи не идет, да и в случае кражи сертификата вы можете очень нескоро узнать об этом. Но шифрование у вас все равно останется. Какой вариант выбрать – решайте сами.

Теперь создадим клиентский сертификат. Запустим скрипт **make_cleint_cert.sh** с параметрами «имя_клиента» «e-mail клиента» «пароль на файл клиентского сертификата»:

make_cleint_cert.sh Test_User testuser@home.net q1w2e3

Сначала опять будет генерироваться ключ, затем будет создан и подписан сертификат пользователя. На экран вываливается информация о сертификате, и вот мы получили все необходимое:

client_Test_User.crt – файл клиентского сертификата
client_Test_User.key – файл закрытого ключа
client_Test_User.csr – запрос на подписание сертификата
client_Test_User.p12 – клиентский сертификат для передачи клиенту

Теперь научим почтовые службы работать с этими сертификатами. Начнем с Postfix. Допишем в **/usr/local/etc/postfix/main.cf** следующие строки:

```
# использовать tls для приёма почты
smtpd_use_tls = yes
# включать возможность авторизации только в режиме tls
smtpd_tls_auth_only = yes
# уровень детализации логов в режиме tls
smtpd_tls_loglevel = 3
# Запрашивать заголовки сообщений с информацией о сертификатах и шифровании
smtpd_tls_received_header = yes
# Периодичность очистки кэша TLS-сессии
smtpd_tls_session_cache_timeout = 3600s
# генератор случайных чисел для TLS
tls_random_source = dev:/dev/urandom
# наши ssl ключи и сертификаты
smtpd_tls_key_file = /usr/local/ssl/ca/ca.key
smtpd_tls_cert_file = /usr/local/ssl/ca/ca.crt
smtpd_tls_CAfile = /usr/local/ssl/ca/ca.crt
```

И перезапустим Postfix – **postfix reload**. Теперь попробуем зайти через **telnet localhost 25** и дать команду **ehlo**. Нам вывалят список поддерживаемых команд, в числе которых мы должны

увидеть **starttls**. Напечатаем **starttls**, и сервер ответит нам – **Ready to start TLS**. После этого два раза наберем **quit**, чтобы попрощаться с сервером.

Не забывайте – теперь благодаря включению smtpd_tls_auth_only = yes мы можем использовать SMTP-авторизацию только в режиме TLS. Если вам необходимо авторизовать клиентов, которые не используют TLS – отключите эту опцию.

Остается только передать клиенту файл с расширением p12 – в этот файл записаны и защищены паролем записываются сертификат клиента, CA сертификат и секретный ключ клиента. После того, как сертификат передан клиенту, следует удалить на сервере файлы клиентского закрытого ключа, запроса на подписание и конечно самого сертификата в формате PKS#12. Ну и, разумеется, необходимо передать ему также и пароль «q1w2e3» – пароль на файл клиентского сертификата. Предположим, что мы это уже проделали, попробуем его установить.

Запускаем The Bat!. Выбираем наш почтовый ящик, идем в Account – Properties – General. Видим там кнопочку «Edit personal Certificates», и нажимаем ее. В открывшемся окне нажимаем кнопку «Import…» и открываем файл **client_Test_User.p12**. В окне «Input PFX password – Input password to decrypt data stored in client_Test_User.p12» вводим пароль на файл клиентского сертификата – «q1w2e3». Вылезает окно «Input PFX password – Input password to decrypt a private key stored in client_Test_User.p12» – еще раз вводим «q1w2e3». И видим, что в окне теперь отображаются два сертификата – наш персональный и доверенный сертификат сервера. Выбрав любой из них, и нажав кнопку «View», увидим, что «This CA root certificate is not trusted because it is not in the Trusted Root CA». То есть The Bat! не желает просто так доверять нашему самоподписанному сертификату. Исправляем ситуацию, переходим на закладку «Certification Path», выбираем наш CA, и нажимаем кнопку «Add to Trusted». На запрос подтверждения отвечаем «Yes». Теперь видим, что наши сертификаты The Bat! считает «правильными». Теперь настроим The Bat! на отправку почты через TLS: идем Account – Properties – Transport и в списке Connections выбираем «Secure to regular port (STARTTLS)». Напишем письмо нашему test user'у, отправим его и посмотри в логи:

```
Nov 18 01:56:40 mail postfix/smtpd[1490]: initializing the server-side TLS engine
Nov 18 01:56:40 mail postfix/smtpd[1490]: connect from unknown[192.168.211.3]
Nov 18 01:56:41 mail postfix/smtpd[1490]: setting up TLS connection from unknown[192.168.211.3]
Nov 18 01:56:41 mail postfix/smtpd[1490]: SSL_accept:before/accept initialization
Nov 18 01:56:41 mail postfix/smtpd[1490]: read from 08096980 [080C0000] (11 bytes => -1 (0xFFFFFFFF))
Nov 18 01:56:41 mail postfix/smtpd[1490]: SSL_accept:error in SSLv2/v3 read client hello A
Nov 18 01:56:41 mail postfix/smtpd[1490]: read from 08096980 [080C0000] (11 bytes => 11 (0xB))
Nov 18 01:56:41 mail postfix/smtpd[1490]: 0000 16 03 01 00 2f 01 00 00|2b 03 01                  ..../...
+..
```

[пропустим этот кошмар...]

```
Nov 18 01:56:41 mail postfix/smtpd[1490]: 0020 fc 2e c5 3a dc d2 b3 23|9b 9c 51 bb 49 2b f9     ...:...#
..Q.I+.
Nov 18 01:56:41 mail postfix/smtpd[1490]: SSL_accept:SSLv3 flush data
Nov 18 01:56:41 mail postfix/smtpd[1490]: TLS connection established from unknown[192.168.211.3]: TLSv1
with cipher RC4-SHA (128/128 bits)
Nov 18 01:56:41 mail postfix/smtpd[1490]: 6EF9860DE: client=unknown[192.168.211.3]
Nov 18 01:56:41 mail postfix/cleanup[1496]: 6EF9860DE: message-id=<1852745703.20051119205209@home.net>
Nov 18 01:56:41 mail postfix/qmgr[1365]: 6EF9860DE: from=<testuser@home.net>, size=752, nrcpt=1 (queue ac-
tive)
Nov 18 01:56:41 mail postfix/smtpd[1490]: disconnect from unknown[192.168.211.3]
Nov 18 01:56:41 mail postfix/virtual[1498]: 6EF9860DE: to=<testuser@home.net>, relay=virtual, delay=0,
status=sent (delivered to maildir)
Nov 18 01:56:41 mail postfix/qmgr[1365]: 6EF9860DE: removed
```

Боже, какой ужас… надо срочно изменить детализацию логов… вместо **smtpd_tls_loglevel = 3** поставим **smtpd_tls_loglevel = 1**. Но главное, мы увидели, что установилось защищенное соединение – «TLS connection established from unknown[192.168.211.3]: TLSv1 with cipher RC4-SHA (128/128 bits)», и наше письмо ушло под его защитой. Но это еще не все. Если мы посмотрим в исходный текст письма на стороне получателя, то увидим там:

```
Received: from stationxp01.home.net (unknown [192.168.211.3])
        (using TLSv1 with cipher RC4-SHA (128/128 bits))
        (No client certificate requested)
        by mail.home.net (Postfix) with ESMTP id 6EF9860DE
        for <testuser@home.net>; Fri, 18 Nov 2005 01:56:41 +0000 (UTC)
```

То есть, на данный момент имеем защищенное соединение, но не имеем аутентификации. Исправляемся. Добавим в **/usr/local/etc/postfix/main.cf** строчки:

```
# требовать сертификаты от клиентов
smtpd_tls_ask_ccert = yes
# место расположения отпечатков клиентских сертификатов
relay_clientcerts = hash:/usr/local/etc/postfix/fingerprints
```

Снимем отпечаток с пользовательского сертификата:

openssl x509 –fingerprint –in client_Test_User.crt

и скопируем последовательность разделенных двоеточиями цифр из строки «MD5 Fingerprint» в файл **usr/local/etc/postfix/fingerprints**. После этой числовой последовательности можно (и нужно) для собственной информации и для того, чтобы отработала команда **postmap** дописать туда имя клиента. Делаем:

postmap fingerprints
postfix reload

И снова отправим письмо многострадальному ТестЮзеру. Оп-ля. Опять не то, потому что получаем:

```
Received: from stationxp01.home.net (unknown [192.168.211.3])
        (using TLSv1 with cipher RC4-SHA (128/128 bits))
        (Client did not present a certificate)
        by mail.home.net (Postfix) with ESMTP id CE91E60DA
        for <testuser@home.net>; Sat, 19 Nov 2005 21:25:48 +0000 (UTC)
```

То есть я при отправке письма для testuser сертификат серверу почему-то не представил. Защищенное соединение установилось, но аутентификация не прошла. Очень долго я бился с этим вопросом, но ничего не выходило. Но вот я наткнулся на это: http://www.opennet.ru/openforum/vsluhforumID14/414.html. Процитирую оттуда:

>как заставить the bat (3.5 у меня) предоставлять клиентские сертификаты при отправке >почты с использованием технологии TLS? Т.е. в настройках аккаунта я сертификат >импортировал, однако в заголовке письма (Client did not present a certificate), >через почтового клиента мозиллы все работает.

the_bat редкостная дрянь в свете работы с ssl. могу вам сообщить, что в него вкомпилирован openssl_0.95 от 99года... то есть, по идее нормально с tls он будет работать только с продуктами собранными с поддержкой этого анахронизма. (в 0.96 изменилась структура, вплоть до названий переменных). баг на работу с ssl открыт нами ещё в 2000 году, однако воз и ныне там.

Вот тебе и раз. Вот тебе и мой любимый The Bat. Поэкспериментировав с Outlook Express обнаружил, что сей продукт ведет себя точно также. И чего от этой дряни еще ожидать… Ставить Microsoft Outlook мне не захотелось. Но в том же самом треде упоминалась Mozilla, о которой я только слышал, но не видел. Скачал Mozilla Thunderbird (далее, говоря «Mozilla», я всегда буду иметь в виду именно почтовый клиент Mozilla Thunderbird, а не одноименный браузер), и начал настраивать. Пропущу общую настройку, перейду сразу к сертификатам. Меню Tools – Account Settings – ветка Security – кнопка «Manage Certificates». В открывшемся окне жмем кнопку «Import» и выбираем наш сертификат **client_Test_User.p12**. Вводим наш хитрый пароль «q1w2e3» для дешифрации сертификата и видим в окошке наш CA и свой персональный сертификат. Если выделим персональный сертификат, и нажмем «View», то увидим, что Mozilla не признает его – «Could not verify this certificate because the issuer is not trusted». Нехорошо это. Переходим на вкладку «Authorities» и ищем в списке наш CA «HOMENET Inc». Выделяем «mail.home.net» и жмем кнопку «Edit». В открывшемся окошечке «Edit CA certificate trust settings». Возможностей немного, нас интересует флажок «This certificate can identify mail users». Этот флажок и включаем, хотя никто не мешает выделить и все остальные. Жмем «Ок», возвращаемся на вкладку «Your certificates», и

снова посмотрим на наш сертификат – «This certificate has been verified for the following uses:» – и ниже перечислено, для чего именно данный сертификат разрешен к использованию. Пробуем отправить почту, и видим, как Mozilla ругается под заголовком «Web site Certified by an Unknown Authority». Хм. Ну ладно. Снова вернемся в меню Tools – Account Settings – ветка Security – кнопка «Manage Certificates» –вкладка «Authorities» – ищем наш CA «HOMENET Inc – кнопка «Edit». Включаем все оставшиеся флажки и выходим. Пробуем отправить почту – тишь, благодать, и ушедшее по назначению письмо. И вот теперь testuser, получив письмо, может в нем увидеть то, к чему мы стремились:

```
Received: from [192.168.211.3] (unknown [192.168.211.3])
        (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
        (Client CN "Test_User", Issuer "mail.home.net" (verified OK))
        by mail.home.net (Postfix) with ESMTP id 569B160DA
        for <testuser@home.net>; Mon, 21 Nov 2005 20:02:12 +0000 (UTC)
```

Значит, можно быть уверенным, что письмо пришло по шифрованному каналу, и именно от меня.

Итого имеем: безопасную передачу почты по шифрованному каналу мы имеем с любым почтовым клиентом, а вот аутентификацию – на данный момент только с Мозиллой. Если аутентификация не менее важна, чем безопасность, то придется срочно переходить на нее…

Теперь займемся настойками Courier-Imap. Отредактируем файл **/usr/local/etc/courier-imap/pop3d**, поменяем в нем всего два параметра (для наших целей этого вполне достаточно):

```
# используемые алгоритмы авторизации в обычном режиме и в режиме TLS
POP3AUTH="LOGIN CRAM-MD5 CRAM-SHA1"
POP3AUTH_TLS="LOGIN CRAM-MD5 CRAM-SHA1"
```

Далее отредактируем файл **/usr/local/etc/courier-imap/pop3d-ssl**, в нем поменяем следующие параметры:

```
# запускаем работу механизма ssl для pop3-сервера
POP3DSSLSTART=YES
# Список алгоритмов шифрации
TLS_CIPHER_LIST=«HIGH:MEDIUM:!aNULL:+SHA1:+MD5:+HIGH:+MEDIUM»
# Сертификат сервера (pem-формат). Внутри данного файла должны содержаться
# сертификат и НЕЗАШИФРОВАННЫЙ секретный ключ, поэтому доступ к данному файлу
# на чтение должен быть предоставлен только пользователю courier.
TLS_CERTFILE=/usr/local/ssl/ca/ca.crt
# путь к каталогу (или файлу),содержащему "доверенные" сертификаты. Сервер
# использует данные сертификаты, если опция TLS_VERIFYPEER установлена в PEER
# или REQUIREPEER.
TLS_TRUSTCERTS=/usr/local/ssl/clients/
```

В заключении добавим в /etc/rc.conf строку

```
courier_imap_pop3d_ssl_enable="YES"
```

А строку

```
courier_imap_pop3d_enable="YES"
```

наоборот закомментируем, запретив использование POP3 без шифрования. И перезапустим сервер. После этого попробуем получить почту по защищенному каналу. Для этого зайдем в свойства нашего аккаунта в Mozilla, и в ветке Server settings включим флажки «Use secure connection (SSL)» (обратим внимание, что в окошечке Port при этом номер порта 110 автоматически сменится на 995) и «Use secure authentication». Пробуем получить почту. И ничего не происходит, Mozilla надолго задумалась. Смотрим в maillog:

```
Feb 24 13:34:11 mail pop3d-ssl: couriertls: /usr/local/ssl/ca/ca.crt: error:0906D06C:PEM
routines:PEM_read_bio:no start line
```

Правильно – Courier хочет, чтобы параметр **TLS_CERTFILE** указывал на файл, содержащий и секретный ключ, и сертификат сервера. Исправляемся:

cd /usr/local/ssl/ca
cp ca.key server.pem
cat ca.crt >> server.pem

В итоге имеем ключ сервера и его сертификат в одном файле. Теперь в файле **/usr/local/etc/courier-imap/pop3d-ssl** поправим параметр **TLS_CERTFILE**:

```
TLS_CERTFILE=/usr/local/ssl/ca/server.pem
```

Перезапустим Courier-Imap (**/usr/local/etc/rc.d/courier-imap-pop3d-ssl.sh restart**) и снова пытаемся проверить почту. О чудо, письмо пришло. Что же в логах? Смотрим:

```
Feb 24 13:39:51 mail authdaemond: LOG:  statement: SELECT login, '', password, 125, 126, mailbox, mailbox,
'', realname, '' FROM user_accounts WHERE login = 'testuser'
Feb 24 13:39:52 mail authdaemond: LOG:  duration: 26.169 ms
Feb 24 13:39:52 mail authdaemond: Authenticated: sysusername=<null>, sysuserid=125, sysgroupid=126, home-
dir=/mail/virtual/testuser@home.net/Maildir/, address=testuser, fullname=Test User,
maildir=/mail/virtual/testuser@home.net/Maildir/, quota=<null>, options=<null>
Feb 24 13:39:52 mail authdaemond: Authenticated: clearpasswd=password, passwd=<null>
Feb 24 13:39:52 mail pop3d-ssl: LOGIN, user=testuser, ip=[192.168.211.3]
Feb 24 13:39:52 mail pop3d-ssl: LOGOUT, user=testuser, ip=[192.168.211.3], top=0, retr=776, time=0
```

Если заглянуть в debug.log, то увидим, что авторизация у нас прошла с использованием CRAM-MD5:

```
Mar  2 19:14:08 mail pop3d-ssl: Connection, ip=[192.168.211.3]
Mar  2 19:14:09 mail authdaemond: received auth request, service=pop3, authtype=cram-md5
Mar  2 19:14:09 mail authdaemond: authpgsql: trying this module
Mar  2 19:14:09 mail authdaemond: cram: challenge=PDE2NTlCMUIwMkM4Q0M0OTJCRkI3ODcwRUQxMDFDQkJGQG1haWw+,
response=dGVzdHVzZXIgOGRmN2Y0ZDJhYTIxZGMwYjdjNzk0ZTUzMjUxYWY4ZGQ=
Mar  2 19:14:09 mail authdaemond: cram: decoded challenge/response, username 'testuser'
Mar  2 19:14:09 mail authdaemond: SQL query: SELECT login, '', password, 125, 126, mailbox, mailbox, '',
realname, '' FROM user_accounts WHERE login = 'testuser'
Mar  2 19:14:09 mail authdaemond: cram validation succeeded
```

Как видим, все прошло успешно. Единственное замечание – в логе открытым текстом виден наш пароль. Происходит это оттого, что на период отладки я установил в файле **/usr/local/etc/authlib/authdaemonrc** параметр **DEBUG_LOGIN=2**. По завершению отладки надо будет вернуть ему значение по умолчанию 0.

Продолжаем настройку. В файле **/usr/local/etc/courier-imap/pop3d-ssl** поправим следующие параметры:

```
# включать механизм TLS в любых случаях
POP3_TLS_REQUIRED=1
# аутентификация клиентов через сертификаты – проверяем, если клиент передал
# сертификат
TLS_VERIFYPEER=PEER
```

Перезапускаем **courier-imap-pop3d-ssl**, и пробуем снова получить письмо. Mozilla сообщает, что «Could not establish an encrypt connection because your certificate was rejected by mail.home.net. Error code: -12271.». Смотрим в лог почтового сервера, и видим там сообщение:

```
Feb 24 13:45:58 mail pop3d-ssl: couriertls: accept: error:140890B2:SSL routines:SSL3_GET_CLIENT_CERTIFI-
CATE:no certificate returned
```

Тут я надолго застрял, пока не дал себе труд внимательно перечитать статью В. Калошина «ISPMail-HOWTO. v.2.0», где и нашел выход из этой ситуации. Идем в каталог **/usr/local/ssl/clients**, и делаем следующее:

cp client_Test_User.crt client_Test_User.pem

c_rehash /usr/local/ssl/clients

Видим сообщение:

Doing /usr/local/ssl/clients
client_Test_User.pem => 88f5eb9d.0

Перезапускаем **courier-imap-pop3d-ssl**, и проверяем почту. Теперь все в порядке.
По идее, дальше следовало бы в **/usr/local/etc/courier-imap/pop3d-ssl** изменить параметр TLS_VERIFYPEER:

```
TLS_VERIFYPEER=REQUIREPEER
```

Нужно это для того, чтобы почта отдавалась только если клиент предоставляет ssl-сертификат, которому мы доверяем. Однако, ни Mozilla, ни другие клиенты с этим параметром почему-то не работают как должно. Если его включить, то станет невозможным забирать с сервера почту, а в логах увидим сообщение:

```
error 140890C7:SSL routines:SSL3_GET_CLIENT_CERTIFICATE:peer did not return a certificate
```

Обидно. К сожалению, с аутентификацией по сертификатам через почтовый клиент здесь ничего не выходит. Но то, что у нас получилось – авторизация CRAM-MD5 по шифрованному каналу тоже дает неплохую защиту, так что можно остановиться и на этом.

Остается только слегка изменить скрипт, создающий клиентские сертификаты, чтобы каждый раз не копировать сертификаты, не выполнять **c_rehash** вручную и автоматом добавлять отпечатки пользовательских ключей в **/usr/local/etc/postfix/fingerprints**. Помучавшись немного со сценариями оболочки, я решил переписать скрипт на Perl, получилось вот что (на изящество не претендую):

touch /usr/local/ssl/make_cleint_cert.pl

```
#!/usr/bin/perl
`openssl req -new -newkey rsa:4096 -nodes -keyout
./clients/client_$ARGV[0].key -subj
/C=RU/ST=Russia/L=Samara/O=HOMENET/OU=IT/CN=$ARGV[0]/emailAddress=$ARGV[1]
-out ./clients/client_$ARGV[0].csr`;
`openssl ca -config ca.config -in ./clients/client_$ARGV[0].csr -out
./clients/client_$ARGV[0].crt -batch`;
`openssl pkcs12 -export -in ./clients/client_$ARGV[0].crt -inkey
./clients/client_$ARGV[0].key -certfile ./ca/ca.crt -out
./clients/client_$ARGV[0].p12 -passout pass:$ARGV[2]`;
`cp ./clients/client_$ARGV[0].crt ./clients/client_$ARGV[0].pem`;
`c_rehash /usr/local/ssl/clients`;
`openssl x509 -fingerprint -in ./clients/client_$ARGV[0].crt >
/usr/local/ssl/clients/fingerprint.txt`;
open (FP, "</usr/local/ssl/clients/fingerprint.txt");
flock (FP,2);
$line = <FP>;
chomp $line;
close (FP);
($head, $fingerprint) = split('=', $line);
$result = $fingerprint." ".$ARGV[0];
open (FP, ">>/usr/local/etc/postfix/fingerprints");
flock (FP,2);
print FP "$result\n";
close (FP);
`cd /usr/local/etc/postfix`;
`postmap /usr/local/etc/postfix/fingerprints`;
`postfix reload`;
exit 0;
```

На этом с настройкой TLS-SSL слава богу, закончено. Теперь можно аналогичным образом настроить IMAP-SSL. Тем более, что IMAP нам так или иначе понадобится для работы Squirrel-Mail. Отредактируем в файле **/usr/local/etc/courier-imap/imapd** следующие параметры:

```
# адреса, на которых imapd-сервер ожидает соединения, 0 – все адреса.
# поскольку imapd нам нужен только для работы Squirrelmail, то разрешим
# соединения только с localhost
ADDRESS=127.0.0.1
# определяет возможности imapd-сервера в ответ на команду CAPABILITY
IMAP_CAPABILITY="IMAP4rev1 UIDPLUS CHILDREN NAMESPACE THREAD=ORDEREDSUBJECT
THREAD=REFERENCES SORT QUOTA AUTH=CRAM-MD5 AUTH=CRAM-SHA1 AUTH=CRAM-SHA256
IDLE"
```

Далее отредактируем файл **/usr/local/etc/courier-imap/imapd-ssl**, в нем поменяем следующие параметры:

```
# запускаем работу механизма ssl для imap-сервера.
IMAPDSSLSTART=YES
# включать механизм TLS в любых случаях
IMAP_TLS_REQUIRED=1
# Список доступных алгоритмов шифрования.
TLS_CIPHER_LIST=«HIGH:MEDIUM:!aNULL:+SHA1:+MD5:+HIGH:+MEDIUM»
# Сертификат сервера (pem-формат). Внутри данного файла должны содержаться
# сертификат и НЕЗАШИФРОВАННЫЙ секретный ключ, поэтому доступ к данному файлу
# на чтение должен быть предоставлен только пользователю courier.
TLS_CERTFILE=/usr/local/ssl/ca/server.pem
# путь к каталогу (или файлу),содержащему "доверенные" сертификаты. Сервер
# использует данные сертификаты, если опция TLS_VERIFYPEER установлена в PEER
# или REQUIREPEER.
TLS_TRUSTCERTS=/usr/local/ssl/clients/
# аутентификация клиентов через сертификаты – проверяем, если клиент
# передал сертификат
TLS_VERIFYPEER=PEER
```

В заключении добавим в **/etc/rc.conf** строки

```
courier_imap_imapd_enable="YES"
courier_imap_imapd_ssl_enable="YES"
```

И перезагрузим сервер или запустим службы IMAP вручную.

Для более тонких настроек серверов pop3 и imap рекомендую ознакомится с содержимым файлов **imapd-ssl.dist**, **imapd.dist**, **pop3d-ssl.dist**, и **pop3d.dist** из каталога **/usr/local/etc/courier-imap.**

3.6. Настройка ClamAV.

Начнем настраивать наш антивирус. Для начала ознакомимся со стартовыми скриптами, которые порты ClamAV и ClamSMTP положили при установке в **/usr/local/etc/rc.d** – **clamav-clamd.sh**, **clamav-freshclam.sh** и **clamsmtpd.sh**. Первый скрипт запускает демона антивируса, второй позволяет в автоматическом режиме обновлять антивирусные базы, а третий запускает демона, с помощью которого антивирус будет вызываться Postfix'ом при получении каждого письма. Ознакомившись с их содержимым, видим, что для автозапуска всего этого при старте сервера необходимо добавить в **/etc/rc.conf** следующие строчки:

```
clamav_clamd_enable=”YES”
clamav_freshclam_enable=”YES”
clamsmtpd_enable=”YES”
```

Далее разбираемся с конфигурационными файлами. Начнем с **/usr/local/etc/clamd.conf**. Приводим его вот такому виду:

```
LogFile /var/log/clamav/clamd.log
LogTime
LogSyslog
LogVerbose
PidFile /var/run/clamav/clamd.pid
TemporaryDirectory /tmp
DatabaseDirectory /var/db/clamav
LocalSocket /var/run/clamav/clamd
FixStaleSocket
User clamav
ScanRAR
```

Затем поправим **/usr/local/etc/freshclam.conf**:

```
DatabaseDirectory /var/db/clamav
UpdateLogFile /var/log/clamav/freshclam.log
LogVerbose
LogSyslog
PidFile /var/run/clamav/freshclam.pid
DatabaseOwner clamav
AllowSupplementaryGroups
DNSDatabaseInfo current.cvd.clamav.net
DatabaseMirror database.clamav.net
```

И в заключении отредактируем **/usr/local/etc/clamsmtpd.conf**:

```
OutAddress: 10026
ClamAddress: /var/run/clamav/clamd
Header: X-Virus-Scanned: ClamAV using ClamSMTP
TempDirectory: /tmp
User: clamav
VirusAction: /usr/local/sbin/virus_action.sh
```

Все остальные параметры в этих файлах можно оставить по умолчанию, но тем не менее, будет не вредно все же ознакомиться с ними, изучив примеры конфигурационных файлов, которые создают соответствующие порты.

Скрипт, упомянутый в параметре VirusAction, предназначен для выполнения неких действий при обнаружении вируса. Например, можно отправлять postmaster'у письмо с уведомлением об успешном уничтожении вируса:

touch /usr/local/sbin/virus_action.sh

```
#!/bin/sh
virmsg="/tmp/msg.body"
touch $virmsg
echo "Hello, Postmaster!" >> $virmsg
echo  >> $virmsg
echo "-----------------------------------------------------" >> $virmsg
echo "ClamAv has been detect a virus" $VIRUS  >> $virmsg
echo "in mail from: " $SENDER >> $virmsg
echo "to " $RECIPIENTS >> $virmsg
echo >> $virmsg
echo "Message has been killed" >> $virmsg
date "+%d-%m-%Y %H:%M:%S" >> $virmsg
echo "---" >> $virmsg
echo "Best regards - your ClamAv"  >> $virmsg
cat $virmsg | mail -s "Virus detected!" postmaster@home.net
rm -f $virmsg
```

И в заключении, научим Postfix использовать все вышеперечисленное. Для начала добавим в **/usr/local/etc/postfix/main.cf** следующие строчки:

```
content_filter = scan:127.0.0.1:10025
```

```
receive_override_options = no_address_mappings
```

А в **/usr/local/etc/postfix/master.cf** допишем вот такой фрагмент:

```
scan      unix  -       -       n       -       16      smtp
    -o smtp_send_xforward_command=yes
127.0.0.1:10026 inet n  -       n       -       16      smtpd
    -o content_filter=
    -o
receive_override_options=no_unknown_recipient_checks,no_header_body_checks
    -o smtpd_helo_restrictions=
    -o smtpd_client_restrictions=
    -o smtpd_sender_restrictions=
    -o smtpd_recipient_restrictions=permit_mynetworks,reject
    -o mynetworks_style=host
    -o smtpd_authorized_xforward_hosts=127.0.0.0/8
```

Теперь попробуем проверить, как это все у нас работает. Перезагружаем сервер, или вручную перезапускаем Postfix, ClamAV, FreshClam и ClamSMTP. Пока сервер перезагружается, подготовим тестовый вирус – создадим в любом текстовом редакторе файл eicar.com, содержащий сигнатуру тестового вируса:

X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*

И попробуем отправить этот файл нашему Test User'у. Как и ожидалось, письма мы не получим, а в **/var/log/clamd.log** видим сообщение:

```
Fri Feb 24 19:54:48 2006 -> /tmp/clamsmtpd.NIuyXl: Eicar-Test-Signature FOUND
```

Также можно заглянуть и в **/var/log/maillog**, там мы тоже видим, что антивирус расправился с непрошенным гостем:

```
Feb 24 19:57:35 mail postfix/smtpd[651]: connect from unknown[192.168.211.3]
Feb 24 19:57:35 mail postfix/smtpd[651]: setting up TLS connection from unknown[192.168.211.3]
Feb 24 19:57:36 mail postfix/smtpd[651]: fingerprint=F6:1C:F1:CC:9F:1E:A7:96:D2:CC:6A:D7:EE:4F:4B:88
Feb 24 19:57:36 mail postfix/smtpd[651]: Verified: subject_CN=Test_User, issuer=mail.home.net
Feb 24 19:57:36 mail postfix/smtpd[651]: TLS connection established from unknown[192.168.211.3]: TLSv1
with cipher DHE-RSA-AES256-SHA (256/256 bits)
Feb 24 19:57:36 mail postfix/smtpd[651]: 7AD581704C: client=unknown[192.168.211.3], sasl_method=CRAM-MD5,
sasl_username=smtptestuser
Feb 24 19:57:36 mail postfix/cleanup[658]: 7AD581704C: message-id=<43FF2E50.1040008@home.net>
Feb 24 19:57:36 mail postfix/qmgr[467]: 7AD581704C: from=<testuser@home.net>, size=1276, nrcpt=1 (queue
active)
Feb 24 19:57:36 mail postfix/smtpd[651]: disconnect from unknown[192.168.211.3]
Feb 24 19:57:36 mail clamsmtpd: 100001: accepted connection from: 127.0.0.1
Feb 24 19:57:36 mail postfix/smtpd[660]: connect from localhost[127.0.0.1]
Feb 24 19:57:36 mail postfix/smtpd[660]: CC5931704F: client=unknown[192.168.211.3]
Feb 24 19:57:36 mail clamsmtpd: 100001: from=testuser@home.net, to=testuser@home.net, status=VIRUS:Eicar-
Test-Signature
Feb 24 19:57:36 mail postfix/smtp[659]: 7AD581704C: to=<testuser@home.net>, relay=127.0.0.1[127.0.0.1],
delay=0, status=sent (250 Virus Detected; Discarded Email)
Feb 24 19:57:36 mail postfix/smtpd[660]: disconnect from localhost[127.0.0.1]
Feb 24 19:57:36 mail postfix/qmgr[467]: 7AD581704C: removed
```

Если теперь мы проверим почту postmaster'а, то увидим, что ему пришло письмо с уведомлением об обнаружении и убиении зараженного письма.

Можете «усложнить» задачу, и заархивировать eicar.com с помощью WinRAR версии 3 – результат будет тем же. Только не пробуйте закрывать архив паролем – ClamAV не взломщик, а антивирус, запароленные архивы он по умолчанию не трогает (и не должен). Хотя есть возможность заставить его просто прибивать запароленные или зашифрованные архивы, но это уж чересчур.

3.7. Настройка Postgrey.

Postgrey – это очень простая и при этом очень эффективная система защиты от спама. Принцип ее работы состоит в том, что любое приходящее на почтовый сервер письмо отбрасывается с

сообщением «Временно не может быть принято, повторите попытку позже», одновременно запоминая информацию об этом письме в своей базе данных, помещая его в так называемый «серый» список. Получив такой ответ, «добропорядочный» сервер-отправитель добросовестно повторит попытку снова передать нам это письмо спустя какой-то промежуток времени. На сей раз, сравнив пришедшее письмо с информацией из своего «серого» списка, Postgrey пропустит это письмо, а информация о нем будет перемещена в «белый» список. При следующей попытке обмена письмами между нами и этим отправителем письма будут приниматься уже без задержек.

Но если это письмо пришло нам от спамера? В этом случае ставка делается на то, что спамер будет либо настойчиво долбиться раз за разом, практически не делая пауз между попытками доставить сообщение, либо же наоборот, получив ответ о невозможности доставить письмо, прекратит попытки переслать его нам. При таком раскладе по прошествии некоторого времени Postgrey переместит информацию о письме из «серого» списка в «черный», и при следующей попытке спамера всучить нам свой мусор его письма будут отвергнуты без лишних проволочек.

Через определенные промежутки времени происходит очистка «черных» и «белых» списков, так что вышеописанные ситуации могут снова повторяться для систем, уже бывших ранее известными нашему серверу.

У этой системы есть и достоинства и недостатки. К достоинствам относится ее простота как в работе, так и в конфигурировании (по сути, почти никакого конфигурирования и не требуется, кроме как настройка MTA для использования Postgrey) и высокая надежность. К недостаткам можно отнести возникновение задержек при доставке почты, что в некоторых случаях может быть критично. Этот недостаток достаточно легко обходится занесением известных отправителей в постоянный «белый» список, однако это требует дополнительных телодвижений, за что и критикуется. Ну и конечно, не исключена вероятность ложных срабатываний – но от этого не застрахован ни один антиспам-фильтр. Лично у меня Postgrey работает уже больше года, и никаких нареканий на него за это время не было, только пользователи одно время приставали с вопросами: «Что случилось, спама нет, а нормальная почта ходит исправно?» ☺

Итак, приступим. Начинаем, как всегда, с изучения каталога **/usr/local/etc/rc.d**, где видим стартовый скрипт **postgrey.sh**. Из его содержимого видно, что для автоматического запуска Postgrey во время старта системы нужно поместить в **/etc/rc.conf** строку:

```
postgrey_enable=”YES”
```

Что мы и проделаем. Далее пытаемся найти описание работы сего чуда. После долгих и безуспешных поисков обнаруживаем в **/usr/ports/mail/postgrey/work/postgrey-1.21** файл README, из которого явствует, что нам следует выполнить команду «**perldoc postgrey**». Хм, и впрямь документацию показывает… читаем, и видим, что нам необходимо в **/usr/local/etc/postfix/main.cf** добавить к smtpd_recipient_restrictions параметр «check_policy_service inet:127.0.0.1:10023». Причем добавлять этот параметр следует ОБЯЗАТЕЛЬНО после «reject_unauth_destination». Ну что ж, проделаем это:

```
smtpd_recipient_restrictions = permit_sasl_authenticated,
                               reject_unauth_destination,
                               check_policy_service inet:127.0.0.1:10023
```

А теперь попробуем проверить, как оно работает. Перезагрузим сервер. И попробуем отправить письмо нашему ТестЮзеру, скажем, от админа сервера mail.work.net. Пробуем получить нашу почту – ничего нет. Смотрим в maillog. Во-первых, видим, что запустился Postgrey:

```
Feb 25 15:16:15 mail postgrey[533]: Process Backgrounded
Feb 25 15:16:15 mail postgrey[533]: 2006/02/25-15:16:15 postgrey (type Net::Server::Multiplex) starting!
pid(533)
Feb 25 15:16:15 mail postgrey[533]: Binding to TCP port 10023 on host localhost
Feb 25 15:16:15 mail postgrey[533]: Setting gid to "225 225"
Feb 25 15:16:15 mail postgrey[533]: Setting uid to "225"
```

А далее видно, что произошло с письмом к ТестЮзеру:

```
Feb 25 15:17:05 mail postfix/smtpd[579]: connect from unknown[192.168.132.200]
Feb 25 15:17:05 mail postfix/smtpd[579]: NOQUEUE: reject: RCPT from unknown[192.168.132.200]: 450 <tes-
tuser@home.net>: Recipient address rejected: Greylisted for 300 seconds (see
http://isg.ee.ethz.ch/tools/postgrey/help/home.net.html); from=<root@work.net> to=<testuser@home.net> pro-
to=ESMTP helo=<mail.work.net>
Feb 25 15:17:05 mail postfix/smtpd[579]: disconnect from unknown[192.168.132.200]
```

Как видим, в ответ на попытку доставить почту для testuser@home.net сервер ответил, что письмо не принято, но занесено в «серый» список на 300 секунд. Теперь надо подождать некоторое время, пока удаленный сервер снова попытается доставить нам почту. И вот наконец:

```
Feb 25 15:28:40 mail postfix/smtpd[635]: connect from unknown[192.168.132.200]
Feb 25 15:28:40 mail postgrey[533]: delayed 695 seconds: client=192.168.132.200, from=root@work.net,
to=testuser@home.net
Feb 25 15:28:40 mail postfix/smtpd[635]: E2C0717050: client=unknown[192.168.132.200]
Feb 25 15:28:40 mail postfix/cleanup[642]: E2C0717050: message-id=<20060225150827.0DF5AB842@mail.work.net>
Feb 25 15:28:40 mail postfix/qmgr[467]: E2C0717050: from=<root@work.net>, size=600, nrcpt=1 (queue active)
Feb 25 15:28:40 mail postfix/smtpd[635]: disconnect from unknown[192.168.132.200]
Feb 25 15:28:41 mail clamsmtpd: 100000: accepted connection from: 127.0.0.1
Feb 25 15:28:41 mail postfix/smtpd[644]: connect from localhost[127.0.0.1]
Feb 25 15:28:41 mail postfix/smtpd[644]: 416F417053: client=unknown[192.168.132.200]
Feb 25 15:28:41 mail postfix/cleanup[642]: 416F417053: message-id=<20060225150827.0DF5AB842@mail.work.net>
Feb 25 15:28:41 mail postfix/qmgr[467]: 416F417053: from=<root@work.net>, size=813, nrcpt=1 (queue active)
Feb 25 15:28:41 mail postfix/smtp[643]: E2C0717050: to=<testuser@home.net>, relay=127.0.0.1[127.0.0.1],
delay=1, status=sent (250 Ok: queued as 416F417053)
Feb 25 15:28:41 mail clamsmtpd: 100000: from=root@work.net, to=testuser@home.net, status=CLEAN
Feb 25 15:28:41 mail postfix/smtpd[644]: disconnect from localhost[127.0.0.1]
Feb 25 15:28:41 mail postfix/qmgr[467]: E2C0717050: removed
Feb 25 15:28:41 mail postfix/virtual[646]: 416F417053: to=<testuser@home.net>, relay=virtual, delay=0,
status=sent (delivered to maildir)
Feb 25 15:28:41 mail postfix/qmgr[467]: 416F417053: removed
```

Итак, теперь Postgrey «знаком» с сервером mail.work.net. и в течении какого-то промежутка времени будет принимать от него почту без лишних проволочек. В этом легко убедиться, снова отправив письмо от root@work.net к testuser@home.net – теперь оно придет мгновенно.

В заключении, несколько слов о постоянных белых списках. Обратите внимание – в каталоге **/usr/local/etc/postfix** есть два файла: **postgrey_whitelist_clients** и **postgrey_whitelist_recipients**. В первом из них можно указывать доверенные домены, почта с которых не будет заноситься в серые списки, а будет доставляться безо всяких проверок. Во втором файле указываются ваши адреса, почта на которые также не должна подвергаться проверке Postgrey – по умолчанию туда занесены адреса postmaster и abuse. Это служебные адреса, и они ДОЛЖНЫ быть доступны всем и всегда.
Рекомендую сразу же очистить содержимое файла **postgrey_whitelist_clients** (что за данные туда внесены неизвестно) и вписать туда те домены, которым вы доверяете. И если вы не используете в smtpd_recipient_restrictions опцию permit_mynetworks, а в параметре mynetworks у вас указано только 127.0.0.0./8, то впишите ваш домен в **postgrey_whitelist_clients**.

3.8. Настройка доступа к почтовому серверу через WWW.

Для начала сконфигурируем Web-сервер Apache, и попробуем к нему подключиться. Первым делом (как и всегда) изучаем скрипты запуска, которые порт положил в **/usr/local/etc/rc.d**. Видим, что к Апачу относятся два файла – **000.apache2libs.sh** и **apache2.sh**. Изучение первого ни на какие мысли не наводит, а из второго видно, что необходимо поместить в **/etc/rc.conf** строчки **apache2_enable="YES"** и **apache2ssl_enable="YES"**. Начнем с обычного Апача без SSL. Перезагружаем сервер, запускаем Opera и пытаемся зайти на http://mail.home.net. Пускает и показывает страничку, сообщающую, что «если вы это видите, значит все в порядке». Значит, так тому и быть. Теперь попробуем завести Апач с SSL. Добавим в **apache2ssl_enable="YES"** в **/etc/rc.conf** и снова перезагрузимся. Пробуем снова зайти на http://mail.home.net – по идее, сейчас нас должны послать куда подальше… так оно и есть. Пробуем https://mail.home.net – и опять отлуп. Ну идем разбираться в чем дело. Хм, а ведь Апач и вовсе не запустился… жалуется на отсутствие сертификатов – что есть правильно, мы ж ему не сказали где лежат наши сертификаты. Исправляемся, правим **/usr/local/etc/apache2/ssl.conf**:

```
SSLCertificateFile /usr/local/ssl/ca/ca.crt
SSLCertificateKeyFile /usr/local/ssl/ca/ca.key
```

И перезагружаемся. На сей раз Апач ни на что не отругался. Снова пробуем постучаться на http://mail.home.net – пускает. На https://mail.home.net – ага, вылезает окошечко «Certificate signer not found» – Opera не знает о существование HOMENET Inc. Если мы сейчас нажмем кнопку «Accept», то увидим стандартную стартовую страничку Апача, а в адресной строчке появится значок замочка и надпись «HOMENET Inc (RU)». Щелкнув мышкой по этой надписи, мы увидим информацию о сертификате, а также строчку «TLS v1.0 256 bit AES (1024 bit DHE_RSA/SHA)», что говорит нам об установлении шифрованного канала между нами и сервером mail.home.net с ключом шифрования длиной в 256 бит – уже неплохо. Закроем эту страничку и попробуем научить Opera работать с сертификатом, который мы уже выдали нашему Тест Юзеру.

Идем в меню Tools – Preferences – вкладка Advanced – Security. Жмем кнопку «Manage Certificates…». Откроется Certificate Manager. На вкладке Personal жмем кнопку «Import…» и ищем по дискам наш сертификат client_Test_User.p12. Нас просят ввести пароль сертификата (q1w2e3, если еще не забыли), вводим его и попадаем в окно Import key and certificate. Так… в поле ниже видим две надписи – Test_User и mail.home.net… выделяем mail.home.net и жмем «Ok». Снова просят ввести пароль. Еще раз вводим наш хитрый пароль. Видим, что в персональные сертификаты добавился сертификат ТестЮзера. Но в поле Issuer пусто. Ладно… нажмем кнопку «Ok» и снова зайдем в Manage Certificates… О, в поле Issuer появилась надпись mail.home.net! Заглянем теперь на вкладку Authorities и теперь тоже увидим mail.home.net в списке сертификационных центров. По идее, Opera теперь должна без вопросов заходить на https://mail.home.net – ну-ка, проверим… да, так и есть. Что уже есть хорошо. Но еще далеко не все.

Теперь начинаем плотно изучать **/usr/local/etc/apache2/httpd.conf** и **/usr/local/etc/apache2/ssl.conf**. Наша задача – разрешать www-доступ к почте только через безопасное соединение и только людям, имеющим наши сертификаты, плюс помимо этого еще и иметь область, не имеющую отношения к веб-почте, доступ в которую возможен и без SSL-шифрования.

Начнем с нуля. Предварительно подготовим файловую структуру www-сервера. По умолчанию Апач размещает ее в **/usr/local/www**, туда же установился и SquirrelMail, ну пусть там оно все и остается (хотя, конечно, можно переместить контент www-сервера и в другое более подходящее вам место). Удалим из **/usr/local/www** все, кроме каталога **squirrelmail**, и создадим там каталоги **/cgi-bin** (для ваших CGI-программ) и **/data** (здесь будем держать данные, доступ к которым можно осуществлять без SSL-шифрования). В каталоге **/usr/local/www/data** создадим простенький index.html, содержащий что-нибудь вроде «Welcome to HOMENET!».

Скопируем **usr/local/etc/apache2/httpd.conf** в **usr/local/etc/apache2/httpd.conf.default**, а **/usr/local/etc/apache2/ssl.conf** в **/usr/local/etc/apache2/ssl.conf.default** и напишем конфиги Апача заново. У меня получилось вот что:

/usr/local/etc/apache2/httpd.conf

```
#
### Section 1: Global Environment
#
# корневой каталог http-сервера
ServerRoot "/usr/local"
ScoreBoardFile /var/run/apache_runtime_status
# месторасположение pid-файла
PidFile /var/run/httpd.pid
# время в секундах перед получением или отсылкой таймаута
Timeout 300
# разрешаем постоянные соединения (более одного запроса с соединения)
KeepAlive On
# максимальное количество запросов в постоянном соединении
MaxKeepAliveRequests 100
# время в течении которого ожидается новый запрос клиента при постоянном
# соединении
KeepAliveTimeout 15
<IfModule prefork.c>
# число одновременно запускаемых процессов сервера
StartServers         5
# минимальное число зарезервированных процессов сервера
MinSpareServers      5
```

```
# максимальное число зарезервированных процессов сервера
MaxSpareServers     10
# максимальное число порождаемых серверных процессов
MaxClients         150
# максимальное количество запросов, обслуживаемых одним процессом сервера
# 0 - неограничено
MaxRequestsPerChild  0
</IfModule>
# слушаем на порту 80 в ожидании соединений
Listen 80
#
# Dynamic Shared Object (DSO) Support
#
# загружаем необходимые нам модули
LoadModule access_module libexec/apache2/mod_access.so
LoadModule alias_module libexec/apache2/mod_alias.so
LoadModule auth_module libexec/apache2/mod_auth.so
LoadModule cgi_module libexec/apache2/mod_cgi.so
LoadModule dir_module libexec/apache2/mod_dir.so
LoadModule log_config_module libexec/apache2/mod_log_config.so
LoadModule logio_module libexec/apache2/mod_logio.so
LoadModule mime_module libexec/apache2/mod_mime.so
LoadModule mime_magic_module libexec/apache2/mod_mime_magic.so
LoadModule setenvif_module libexec/apache2/mod_setenvif.so
<IfDefine SSL>
LoadModule ssl_module libexec/apache2/mod_ssl.so
</IfDefine>
# модуль PHP рекомендуется загружать после модуля SSL
LoadModule php4_module libexec/apache2/libphp4.so
#
### Section 2: 'Main' server configuration
#
# пользователь и группа, от имени которых запускается Apache
User www
Group www
# почтовый адрес администратора www-сервера
ServerAdmin postmaster@home.net
# имя вашего сервера
ServerName mail.home.net:80
# используем в ссылках на себя самого имя хоста и порт, предоставленные
# клиентом
UseCanonicalName Off
# корневой каталог документов www-сервера
DocumentRoot "/usr/local/www/data"
<Directory />
# запрещаем в этом каталоге запуск CGI, переходы по символическим ссылкам,
# SSI и выдачу индекса каталога
    Options None
# запрещаем перекрытие директив из файла .htaccess
    AllowOverride None
# и запрещаем доступ к этому каталогу
    Order Deny,Allow
    Deny from all
</Directory>
# похожим образом настраиваем свойства для каталога документов
<Directory "/usr/local/www/data">
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
# список файлов, которые будут переданы клиенту при обращении к каталогу
DirectoryIndex index.html index.php
# устанавливаем имя файла, содержащего дополнительные правила конфигурации
# для каждого каталога
AccessFileName .htaccess
# запрещаем просмотр этих файлов через браузер
<FilesMatch "^\.ht">
```

```
    Order allow,deny
    Deny from all
</FilesMatch>
# конфигурация MIME-типов
TypesConfig etc/apache2/mime.types
DefaultType text/plain
<IfModule mod_mime_magic.c>
    MIMEMagicFile etc/apache2/magic
</IfModule>
# сохраняем в логах IP-адреса клиентов вместо имен их хостов
HostnameLookups Off
# путь к лог-файлу ошибок
ErrorLog /var/log/httpd-error.log
# Уровень детализации логов
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn
# форматы логов
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" com-
bined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
CustomLog /var/log/httpd-access.log combined
# изменяем заголовок HTTP-ответа сервера на «Server: Apache»
ServerTokens Prod
# отключаем подпись Apache внизу сгенерированных им страниц
ServerSignature Off
# алиас и каталог для CGI-программ
ScriptAlias /cgi-bin/ "/usr/local/www/cgi-bin/"
<Directory "/usr/local/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
# используем русский и английский языки
AddLanguage en .en
AddLanguage ru .ru
# перечисляем используемые кодировки
AddCharset ISO-8859-1  .iso8859-1  .latin1
AddCharset ISO-8859-2  .iso8859-2  .latin2 .cen
AddCharset ISO-8859-3  .iso8859-3  .latin3
AddCharset ISO-8859-4  .iso8859-4  .latin4
AddCharset ISO-8859-5  .iso8859-5  .latin5 .cyr .iso-ru
# For russian, more than one charset is used (depends on client, mostly):
AddCharset WINDOWS-1251 .cp-1251   .win-1251
AddCharset CP866       .cp866
AddCharset KOI8-r      .koi8-r .koi8-ru
AddCharset KOI8-ru     .koi8-uk .ua
AddCharset ISO-10646-UCS-2 .ucs2
AddCharset ISO-10646-UCS-4 .ucs4
AddCharset UTF-8       .utf8
# определяем дополнительные типы данных
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
# определяем обработчик CGI
AddHandler cgi-script .cgi
AddHandler type-map var
# следующие директивы модифицируют HTTP-ответы с тем, чтобы избежать
# некоторых известных проблем с браузерами
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0
```

```
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
BrowserMatch "Microsoft Data Access Internet Publishing Provider" redirect-
carefully
BrowserMatch "MS FrontPage" redirect-carefully
BrowserMatch "^WebDrive" redirect-carefully
BrowserMatch "^WebDAVFS/1.[0123]" redirect-carefully
BrowserMatch "^gnome-vfs" redirect-carefully
BrowserMatch "^XML Spy" redirect-carefully
BrowserMatch "^Dreamweaver-WebDAV-SCM1" redirect-carefully
# используем внешний файл конфигурации ssl
<IfModule mod_ssl.c>
    Include etc/apache2/ssl.conf
</IfModule>
```

/usr/local/etc/apache2/ssl.conf

```
# Настраиваем генератор случайных чисел, указываем метод
# для рандомизации. Указывается два источника рандомизации:
# начальный (startup) и инициализирующийся при присоединении
# клиента (connect). Возможные значения для этого параметра:
# builtin - встроенный источник рандомизации,
# file:имя_файла - читаются данные из файла,
# file:имя_файла число_байт -  читается только определённое
# число байт из файла
SSLRandomSeed startup file:/dev/urandom 1024
SSLRandomSeed connect file:/dev/urandom 1024
<IfDefine SSL>
# Слушаем на порту 443(https).
Listen 443
##
##  SSL Global Context
##
# определяем дополнительные типы данных
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl    .crl
# Определяем способ запроса пароля к секретному ключу сервера как built-in,
# или можно определить путь к внешнему скрипту, но в большинстве случаев
# достаточно built-in-метода.
SSLPassPhraseDialog  builtin
# Определяем кеширование данных ssl в файле данных (none для отключения
# кеширования) и таймаут обновления данных в кеше. Можно использовать
# при указании файла кеша специальный формат shm (shm:/path/file [size]),
# который обеспечивает высокую производительность и работает через механизм
# shared memory.
SSLSessionCache         dbm:/var/run/ssl_scache
SSLSessionCacheTimeout  300
SSLMutex  file:/var/run/ssl_mutex
# Определяем виртуальный сервер, висящий на том же
# IP-адресе, что и основной, но слушающий на 443 порту.
<VirtualHost _default_:443>
# Основные настройки виртуального сервера.
DocumentRoot "/usr/local/www/squirrelmail"
ServerName mail.home.net:443
ServerAdmin postmaster@home.net
ErrorLog /var/log/httpd-error.log
TransferLog /var/log/httpd-access.log
<Directory />
    AllowOverride None
    Order Allow,Deny
    Allow from all
</Directory>
# включаем ssl для данного виртуального сервера.
SSLEngine on
SSLProtocol -all +TLSv1 +SSLv3
# Список доступных алгоритмов шифрования. Формат списка такой же, как и у
```

```
# команды openssl ciphers (MEDIUM, HIGH, NULL, элементы разделяются :,
# исключение элемента осуществляется знаком !).
SSLCipherSuite HIGH:MEDIUM:!aNULL:+SHA1:+MD5:+HIGH:+MEDIUM
# сертификат сервера. данный сертификат должен быть в формате pem.
SSLCertificateFile /usr/local/ssl/ca/ca.crt
# Секретный ключ сервера, которым он будет расшифровывать данные клиента.
# Если ключ зашифрован, то при запуске апача будет спрашиваться пароль, как
# определено в опции SSLPassPhraseDialog.
SSLCertificateKeyFile /usr/local/ssl/ca/ca.key
# Следующие директивы описывают CA-сертификаты. Если указан параметр
# SSLCACertificatePath, то CA-сертификаты загружаются из указанного каталога
# (в этом каталоге должны также размещаться символические ссылки с именем
# hash-value.N, при добавлении нового файла сертификата в данный каталог
# необходимо воспользоваться специальным makefile, идущим вместе с модулем).
# Если указан параметр SSLCACertificateFile, то все CA-сертификаты хранятся
# в одном pem-файле (сертификаты просто записываются один за другим в данный
# файл без всяких разделитетлей, т.к. любой сертификат имеет маркер начала и
# конца)
#SSLCACertificateFile /usr/local/ssl/ca/ca.crt
# Список сертификатов, которые считаются недействительными (certificate
# revocation list - CRL) по неким причинам: по происшествии срока действия,
# по потере секретного ключа и.т.д.
#SSLCARevocationFile /usr/local/ssl/db/crl.pem
# Требуем от клиента посылки своего сертификата для выполнения
# аутентификации.
#SSLVerifyClient require
# Число шагов от сертификата клиента до CA-сертификата в иерархии
# сертификатов.
#SSLVerifyDepth  1
<FilesMatch "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory "/usr/local/www/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>
# Переменные, устанавливаемые для некоторых браузеров. Вообще-то все браузеры
# должны получить перед завершением ssl-сеанса сообщение от сервера, что
# соответствует стандарту, но некоторые очень некорректно (мягко говоря)
# завершают работу с ssl, поэтому приходится устанавливать переменную
# ssl-unclean-shutdown, которая говорит о том, что при  завершении соединения
# сервер не должен ни посылать сообщение, ни принимать подтверждение от
# браузера (что тоже может делаться некоторыми браузерами).
SetEnvIf User-Agent ".*MSIE.*" \
         nokeepalive ssl-unclean-shutdown \
         downgrade-1.0 force-response-1.0
CustomLog /var/log/httpd-ssl_request.log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
</IfDefine>
```

Проверим, как у нас все это работает. Перезапустим Апач (**/usr/local/etc/rc.d/apache2 restart**), и приведем **index.html** в каталоге **/usr/local/www/data** к вот такому виду:

```
<html><head><title>Welcome to HOMENET!</title></head>
<body><center><h2>Welcome to HOMENET!<hr>
<a href="https://mail.home.net">Enter to web mail</a></h2>
</center></body></html>
```

Попробуем влезть на http://mail.home.net. Только на сей раз для чистоты эксперимента вместо Opera воспользуемся Internet Explorer, которому мы пока не устанавливали SSL-сертификаты. Открываем нашу страничку, никаких вопросов нам не задают – отлично, значит, имеем незащищенную область, куда можно складывать общедоступные документы. Теперь нажмем на ссылку «Enter to web mail». Первым делом видим окошечко, где нам сообщают, что начинается просмотр через безопасное соединение – пока все прекрасно, жмем «Ок». Через некоторое время появится окошко, уведомляющее нас, что IE не смог проверить сертификат нашего сервера – на данный мо-

мент это нормально, мы еще не устанавливали в IE наши сертификаты, Нажимаем «Yes», чтобы продолжить просмотр. И наконец нам выдают страничку, с таким текстом: «**ERROR: Config file "config/config.php" not found. You need to configure SquirrelMail before you can use it**». Все правильно, мы еще не сконфигурировали SquirrelMail, поэтому получаем отлуп. Однако в статусной строке IE появился значок замочка, уведомляющий нас о том, что мы находимся в защищенной зоне www-сервера под защитой ключа шифрования длиной в 128 бит. На данном этапе это как раз то, чего мы добивались. Теперь можно перейти к конфигурированию SquirrelMail.

Напоминаю, что для работы SquirrelMail нам необходим работающий IMAP4 сервер, поэтому если вы до сей поры использовали только POP3/POP3-SSL, вам необходимо вернуться к разделам 3.4 – 3.5 и настроить службы imap Courier-IMAP. Если imap у вас уже работает, можно продолжать.

Перейдем в каталог **/usr/local/www/squirrelmail** и запустим скрипт **configure**. Это программа конфигурирования SquirrelMail. Не вредно будет в свободное время поблуждать по всем менюшкам для ознакомления с ее возможностями. А пока приступим к конфигурированию. Выбираем пункт 2 – Server settings. Там первым делом изменяем Domain на наш home.net. Возвращаемся в главное меню и выбираем пункт 10 – Languages. Указываем Default Language – ru_RU, Default Charset – koi8-r. И на этом можно и закончить (для начала). Печатаем «Q» для выхода и соглашаемся с предложением сохранить изменения. Видим сообщение: «You might want to test your configuration by browsing to http://your-squirrelmail-location/src/configtest.php». Значит, запускаем Internet Explorer, и идем по адресу https://mail.home.net/src/configtest.php. И вот результат:

```
SquirrelMail configtest
This script will try to check some aspects of your SquirrelMail configuration and point you to errors
whereever it can find them. You need to go run conf.pl in the config/ directory first before you run this
script.

SquirrelMail version: 1.4.5
Config file version: 1.4.0
Config file last modified: 27 February 2006 14:58:35

Checking PHP configuration...
    PHP version 4.4.1 OK.
    PHP extensions OK.
Checking paths...
    Data dir OK.
    Attachment dir OK.
    Plugins are not enabled in config.
    Themes OK.
    Default language OK.
    Base URL detected as: https://mail.home.net/src
Checking outgoing mail service....
    SMTP server OK (220 mail.home.net ESMTP)
Checking IMAP service....
    IMAP server ready (* OK [CAPABILITY IMAP4rev1 UIDPLUS CHILDREN NAMESPACE THREAD=ORDEREDSUBJECT
THREAD=REFERENCES SORT QUOTA IDLE ACL ACL2=UNION STARTTLS] Courier-IMAP ready. Copyright 1998-2005 Double
Precision, Inc. See COPYING for distribution information.)
    Capabilities: * CAPABILITY IMAP4rev1 UIDPLUS CHILDREN NAMESPACE THREAD=ORDEREDSUBJECT THREAD=REFER-
ENCES SORT QUOTA IDLE ACL ACL2=UNION STARTTLS
Checking internationalization (i18n) settings...
     gettext - Gettext functions are available. You must have appropriate system locales compiled.
     mbstring - Mbstring functions are available.
     recode - Recode functions are unavailable.
     iconv - Iconv functions are unavailable.
     timezone - Webmail users can change their time zone settings.
Checking database functions...
    not using database functionality.

Congratulations, your SquirrelMail setup looks fine to me!

Login now
```

Вроде как все в порядке. Пытаемся залогиниться, нажав на ссылку «Login now». Попадаем на страничку ввода имени и пароля, где группа разработчиков SquirrelMail приветствует нас на чистейшем русском языке. Введем логин и пароль нашего ТестЮзера – и ура, попадаем в почтовый ящик со стандартным www-интерфейсом. Пошлем ТестЮзеру письмо, нажмем ссылку «Обновить» – и видим наше только что отправленное письмо.

Теперь добьемся того, чтобы в защищенную зону www-сервера мог получить доступ только человек, имеющий наш сертификат – на текущий момент это может сделать любой. Для этого необходимо в файле **/usr/local/etc/apache2/ssl.conf** раскомментировать строки

```
#SSLCACertificateFile /usr/local/ssl/ca/ca.crt
#SSLVerifyClient require
#SSLVerifyDepth  1
```

и перезапустить Апач. Попытаемся теперь войти на https://mail.home.net и увидим, что нам предлагают указать сертификат для доступа к защищенной области. Поскольку у нас не установлен в IE ни один сертификат, то попасть туда мы не сможем. Если же сейчас мы попытаемся использовать для доступа Opera, то мы сможем указать наш сертификат и зайти в почту, так как в Оперу мы уже установили наш сертификат. Но давайте научим этому и Internet Explorer. Найдем на диске наш сертификат client_Test_User.p12 и дважды щелкнем по нему мышкой. Запустится мастер импорта сертификатов, следуя указаниям которого мы проинсталлируем наш сертификат. Замечу, что чтобы защитить наш сертификат от неавторизованного доступа, следует задействовать функцию "Enable strong private key protection" (Разрешить защиту приватного ключа). Кроме того, чтобы защититься от кражи сертификата стоит отменить возможность экспортировать сертификат: флажок "Mark this key as exportable" (Пометит этот ключ как экспортируемый) следует отключить. Также следует поставить уровень безопасности браузера в состояние "High" (Высокий). Мы увидим это в следующем шаге Import Wizard (Мастера Импорта). При таких настройках пароль будет запрашиваться всегда, когда браузеру потребуется использовать этот клиентский сертификат.

Проделав все это, снова запустим IE и попытаемся войти на https://mail.home.net. И теперь мы можем указать, что для входа на этот сервер следует предъявить сертификат на имя ТестЮзера. Выбрав его из списка, нажав «Ок» и введя пароль, который мы установили для сертификата при его импорте в браузер, мы успешно оказываемся на страничке входа SquirrelMail.

Осталось решить последний вопрос – что делать, если мы узнали, что выданный нами сертификат пользователя скомпрометирован или же просто понадобилось запретить кому-либо из клиентов доступ к нашему серверу. Для этого предусмотрена процедура отзыва сертификатов. Чтобы отозвать сертификат, необходимо обновить информацию о нем в нашей базе сертификатов – не просто так мы городили несколько служебных каталогов и файлов в **/usr/local/ssl**. Вначале создадим список отозванных сертификатов. Перейдем в каталог **/usr/local/ssl** и создадим скрипт **update_revocation_list.sh**:

```
#!/bin/sh
openssl ca -config ca.config -gencrl -md sha1 -out ./db/crl.pem
```

Этим скриптом мы создаем или обновляем список отозванных сертификатов. Обратите внимание: в файле **/usr/local/ssl/ca.config** есть параметр

```
default_crl_days       = 3
```

Это означает, что вы должны обновлять список отозванных сертификатов не реже чем раз в 3 дня, поэтому необходимо предусмотреть запуск скрипта **update_revocation_list.sh** по cron раз в трое суток.

Собственно отзыв сертификата будем осуществлять скриптом **revoke_cert.sh** (создадим его в каталоге **/usr/local/etc/postfix/certs**):

```
#!/bin/sh
openssl ca -config ca.config -revoke ./clients/$1
openssl ca -config ca.config -gencrl -md sha1 -out ./db/crl.pem
```

В заключении, необходимо сказать Апачу, где смотреть список отозванных сертификатов. Раскомментируем в **/usr/local/etc/apache2/ssl.conf** строку

```
#SSLCARevocationFile /usr/local/ssl/db/crl.pem
```

Перезапустите Апач. А теперь отзовем сертификат ТестЮзера командой

revoke_cert.sh client_Test_User.crt

И попытаемся теперь зайти на https://mail.home.net через Internet Explorer. Выполняем все формальности с выбором сертификата и вводом пароля, но вместо ожидаемой странички Squirrel-Mail видим сообщение Security Alert «The security certificate for this site has been revoked. This site should not be trusted.» Попасть в защищенную зону веб-сервера мы более не можем.

Заключение

Если у вас получилось сделать все вышеописанное, то в ваших руках достаточно гибкая и защищенная почтовая система, пригодная для использования в организации среднего масштаба. Но конечно, нет предела совершенству, многое осталось за рамками статьи, так что все в ваших руках, совершенствуйте эту систему. Удачи!

В заключении, нельзя не упомянуть источники информации и людей, без которых эта статья не была бы написана.

1. «ISPMail-HOWTO». v.2.0 © Viacheslav "multik" Kaloshin
2. «Основы работы с OpenSSL» © Vsevolod A. Stakhov
3. «Использование OpenSSL» © Vsevolod A. Stakhov
4. «Работа с Web-сервером Russian Apache» © Артем Подстрешный
5. «Apache 2 с SSL/TLS: Шаг за Шагом» © Artur Maj
6. «Почтовая система на базе Postfix, PostgreSQL, с фильтрацией вирусов и спама» © Владимир В. Агапов AKA -=Onix=-
7. Postfix Documentation - http://www.postfix.org/documentation.html
8. Courier-Imap Documentation http://www.courier-mta.org/imap/
9. ClamAv Documentetion http://www.clamav.net/doc/
10. ClamSMTP Documentation http://memberwebs.com/nielsen/software/clamsmtp/
11. Cyrus SASL library, version 2 http://www.csu.ac.ru/~srg/local_doc/libsasl2/
12. и многие многие другие, писавшие в форумах opennet.ru

За сим прощаюсь

Сергей Святкин AKA SHRDLU
2005-2006