Настройка сервера Linux.

Часть 2.
Для самостоятельного изучения.

Артур Крюков

09.12.2008

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.

ЧТО МЫ БУДЕМ РАССМАТРИВАТЬ НА ЭТОМ КУРСЕ.

Задача курса — научить настраивать почтовый сервер.

Основные темы курса:

- Концепция системы электронной почты.
- Настройка почтового сервера.
- Настройка imap сервера.
- Подключение антивируса.
- Защита от спама.

В итоге вы получите работоспособный почтовый сервер, обслуживающий почту для вашего домена.

ПРЕДВАРИТЕЛЬНАЯ ПОДГОТОВКА.

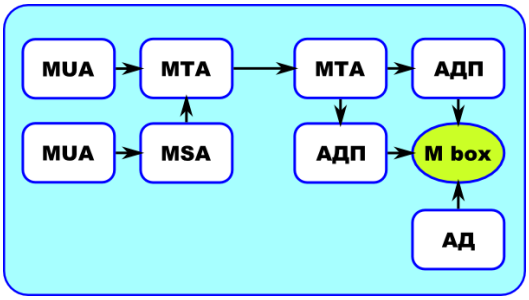
Для успешного прохождения данного курса вы должны:

- Знать основы работы в Linux.
- Знать основы работы сетей TCP/IP.
- Уметь устанавливать пакеты с программным обеспечением.
- Уметь настраивать DNS сервер.
- Уметь настраивать firewall в Linux.

Все указанные выше темы рассматривались в первой части курса.

КОМПОНЕНТЫ СИСТЕМЫ ЭЛЕКТРОННОЙ ПОЧТЫ.

Система электронной почты состоит из многих компонентов. Они могут быть выполнены в виде одной или нескольких программ.



На рисунке представлены основные компоненты:

- Пользовательский почтовый агент (MUA — Message User Agent).
- Транспортный агент (MTA — Message Transfer Agent).
- Агент подачи сообщений (MSA — Message Submission Agent).
- Агент доставки почты (АДП).
- Агент доступа (АД).

Если брать в качестве примера Microsoft Exchange, то большинство компонентов являются частью этой программы, т.е. находятся внутри программы. В большинстве реализаций системы электронной почты в UNIX принято каждый из компонентов делать в виде нескольких программ. В результате мы получаем большую свободу выбора. Вы можете комбинировать различные компоненты от разных производителей и получать почтовый сервер, настроенный под ваши конкретные задачи.

Например, в качестве MTA можно использовать sendmail или postfix. А для доставки сообщений в почтовые ящики mail.local или procmail.

ПОЛЬЗОВАТЕЛЬСКИЙ ПОЧТОВЫЙ АГЕНТ.

При помощи пользовательских агентов пользователи составляют и отправляют письма. Агент должен сформировать тело письма согласно стандарта и передать его на отправку транспортному агенту или агенту подачи.

В роли пользовательских агентов выступают такие программы как: The Bat, Outlook и Outlook Express. Если говорить про Linux: Evolution, KMail, pine и др.

При передаче письма *транспортному агенту* и *агенту подачи* используется протокол SMTP.

ТРАНСПОРТНЫЙ АГЕНТ.

Транспортный агент (*mail transport agent*) выполняет две основные задачи:

- прием почты от пользовательского агента и пересылка ее на другой транспортный агент.
- Прием почты от других транспортных агентов.

При приеме почты от пользователя он должен проверить правильность адреса назначения, возможность доставки почты и доставить почту по назначению.

На другой стороне транспортный агент проверяет:

- предназначено ли это письмо для данной машины (домена),
- есть ли почтовый ящик пользователя на машине.

После всех проверок он принимает письмо и передает его *Агенту доставки почты*.

В мире UNIX существует большое количество программ, реализующих функции транспортного агента. Среди наиболее популярных бесплатных реализаций транспортных агентов можно выделить *sendmail*, *postfix*, *exim* и *qmail*. Каждая из программ имеет свои достоинства и недостатки.

АГЕНТ ПОДАЧИ СООБЩЕНИЙ.

Агенты подачи сообщений — это одна из разновидностей режима работы транспортного агента. Агенты подачи применяются на почтовых узлах с напряженным трафиком. Его задача облегчить работу основного транспортного агента.

При помощи агента подачи сообщений пытаются разделить две составных части обработки почтовых сообщений: прием сообщения от клиента и его маршрутизацию (доставку). Это позволяет облегчить разработку программ для обработки электронной почты.

Агент подачи сообщений:

- Проверяет, являются ли имена узлов полностью определенными.
- Модифицирует заголовки сообщений, полученных от неправильно работающих пользовательских агентов.
- Проверяет все ошибки перед передачей письма транспортному агенту.
- Осуществляет (если это нужно) аутентификацию клиента.

Агент подачи слушает запросы на *587* порту, поэтому все пользовательские агенты необходимо сконфигурировать таким образом, чтобы они отправляли почту на этот порт.

Все особенности работы агента подачи описаны в RFC 2476.

АГЕНТ ДОСТАВКИ ПОЧТЫ.

Транспортный агент после получения почты сам не доставляет ее в почтовый ящик пользователя. Он передает ее агенту доставки почты, задача которого доставить почту в почтовый ящик пользователя.

В качестве агента доставки может выступать простейшая программа, которая просто складывает почту. Существуют и более сложные программы, которые при доставке почты могут осуществлять ее фильтрацию, например, procmail.

АГЕНТЫ ДОСТУПА.

Агенты доступа позволяют пользователю получить доступ к своему почтовому ящику. Они выполнены в виде программ, организующих доступ к почтовому ящику по протоколу *pop3* или *imap*.

Если пользователь работает локально на машине, на которой хранятся его почтовые ящики, и почтовые ящики выполнены в виде файлов, он может получить доступ без агента доступа, обращаясь к ним напрямую. По такому принципу работают программы mail и pine.

SMTP.

В качестве основного протокола взаимодействия в системе электронной почты используются протоколы SMTP (Simple Mail Transfer Protocol) и ESMTP (Extended SMTP). Они описаны в RFC2821, 1869,1870,1891 и 1985.

Протокол SMTP задумывался как простой протокол взаимодействия, при помощи которого пользователь мог напрямую общаться с почтовым транспортным агентом. Конечно же, сейчас пользователи сами не работают с транспортными агентами. Для облечения работы используются пользовательские агенты. Но они (пользовательские агенты) для взаимодействия с транспортными агентами используют протокол SMTP или ESMTP.

В качестве примера можно показать, как пользователь при помощи программы telnet может подключиться к транспортному агенту и отправить письмо.

ПОЛЬЗОВАТЕЛЬ, ПРИ ПОМОЩИ ТРАНСПОРТНОГО АГЕНТА МОЖЕТ ТОЛЬКО ОТПРАВЛЯТЬ ПИСЬМА. ПРИЕМ ПОЧТЫ ПРОИСХОДИТ ПРИ ПОМОЩИ АГЕНТОВ ДОСТУПА.

Для подключения к почтовому транспортному агенту использовалась программа telnet, с явным указанием порта — 25.

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
220 Artur ESMTP mail server
```

Теперь представимся.

```
ehlo artur
250-localhost Hello localhost [127.0.0.1], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH DIGEST-MD5 CRAM-MD5
250-STARTTLS
250-DELIVERBY
250 HELP
```

Команда ehlo или ее предшественница helo позволяют указать кто подключился к серверу. В ответ мы получаем подсказку об основных возможностях сервера, к которому мы подключились. Хоть ehlo и не обязательная команда, но все равно считается хорошим тоном сначала представиться.

Затем необходимо указать от кого отправляется письмо.

```
mail from: artur@kryukov.biz

250 2.1.0 artur@kryukov.biz... Sender ok
```

Если почтовый сервер согласен принять письмо от указанного пользователя, выдается сообщение с подтверждением.

Теперь укажем, кому предназначено письмо.

```
rcpt to: artur@kryukov.biz
250 2.1.5 artur@kryukov.biz... Recipient ok
```

Можно указать несколько адресов назначения.

```
rcpt to: root
250 2.1.5 root... Recipient ok
```

Ну и собственно само письмо. При помощи команды *data* начинаем ввод тела письма. Конец письма — это точка в начале строки.

```
data
354 Enter mail, end with "." on a line by itself
Hello user
I'm send a letter for you ^)
Regards, Artur
.
250 2.0.0 m7MEn569008029 Message accepted for delivery
```

После того, как мы ввели точку, сообщение было принято на доставку.

Остаётся только отключиться от сервера, при помощи команды *quit*.

```
quit
221 2.0.0 cosmos.kryukov.biz closing connection
Connection closed by foreign host.
```

Как видно из приведенного примера, в самом простом случае, транспортному агенту для доставки почты необходимо указать два параметра: адрес отправителя и адрес получателя, а также тело письма.

ПОЧТОВЫЙ СЕРВЕР SENDMAIL.

Sendmail — это наиболее распространенный транспортный агент в UNIX системах. Он поставляется почти со всеми дистрибутивами Linux, в том числе и с CentOS. Sendmail был написан Эриком Оллманом в 1983 году и в дальнейшем дорабатывался различными разработчиками.

Нет таких действий, которые бы Sendmail не мог делать с почтовыми сообщениями, за исключением проверки содержимого письма. Но для его проверки он может передать письмо сторонней программе. Обычно так включается проверка на вирусы в почтовых сообщениях.

КОНФИГУРАЦИОННЫЕ ФАЙЛЫ SENDMAIL.

Основной конфигурационный файл программы — sendmail.cf. Обычно он находится в директории /etc/mail. В файле определяются основные параметры и особенности работы sendmail, в том числе и дополнительные конфигурационные файлы, которые тоже обычно находятся в директории /etc/mail.

Кроме файла sendmail.cf в системе может присутствовать файл submit.cf, являющийся конфигурационным файлом агента подачи сообщений.

Какой из перечисленных файлов будет использоваться в качестве конфигурационного (то есть, в каком режиме будет работать программа) зависит от опций, с которыми sendmail запускается:

- -Ac — запуск в качестве агента подачи почты с использованием конфигурационного файла submit.cf.
- -Am — запускается в качестве основного почтового сервера с использованием файла sendmail.cf. (Параметр по умолчанию).

Файл submit.cf обычно никогда не редактируется.

Если посмотреть на содержимое файла sendmail.cf, в нем можно найти строки похожие на:

```
R$* < @ $=w > $* $: $1 < @ $2 . > $3
R$* < @ $=M > $* $: $1 < @ $2 . > $3
R$* < @ $={VirtHost} > $* $: $1 < @ $2 . > $3
```

Формат файла изначально создавался с учетом облегчения синтаксического анализа файла и поэтому он мало понятен обыкновенным пользователям. Администратору крайне редко приходится редактировать sendmail.cf. Для создания файла используются специальные макросы препроцессора m4, облегчающие задачу его создания и изменения.

КОНФИГУРАЦИЯ SENDMAIL ПРИ ПОМОЩИ ПРЕПРОЦЕССОРА M4.

Как было сказано в предыдущем разделе, препроцессор m4 облегчает администратору процесс создания и изменения конфигурационных файлов *sendmail.cf* и *submit.cf*.

После установки sendmail, конфигурационные макросы будут помещены в директорию */usr/share/sendmail-cf*[1]. В этой директории вы увидите файл *README*, где находится полноценная документация по всем макросам. Сами макросы располагаются в поддиректориях.

В директории *cf* находятся шаблоны макросов, которые можно использовать для создания файла *sendmail.cf* для разных типов UNIX.

[1] Директория, в которой находятся макросы, в разных дистрибутивах разная.

Для получения файла *sendmail.cf* необходимо создать файл, в котором будут описаны используемые макросы. Затем его пропускают через препроцессор и в результате получают конфигурационный файл.

```
m4 my.mc > /etc/mail/sendmail.cf
```

В дистрибутиве CentOS файл-шаблон находится в директории */etc/mail*.

Препроцессор m4 имеет очень строгий синтаксис. Любой лишний пробел, неправильное указание скобок ведет к появлению сообщения об ошибке или создание неправильного файла.

В качестве начала комментария используется инструкция *dnl*. Все, что будет написано после этой инструкции до конца строки, считается комментарием. Рекомендуется в конце каждой строки писать *dnl* для того, чтобы игнорировать случайные пробелы, которые были помещены в конце макроса.

Если необходимо указывать параметры или строки, их берут в кавычки. Следует обратить особое внимание на то, что открывающая кавычка — это обратная одинарная кавычка «`» (на клавиатуре расположена там же где и буква ё), а закрывающая — это одинарная прямая кавычка «'».

Так же, в файле на языке препроцессора можно использовать инструкцию *divert*, при помощи которой можно писать комментарии сразу на нескольких строках.

```
divert(-1)
#
# Copyright (c) 1998-2002 Sendmail, Inc.
# Copyright (c) 1983 Eric P. Allman.
# Copyright (c) 1988, 1993
# The Regents of the University of California.
divert(0)
```

divert(-1) — очищает буфер макросов от данных, оставшихся от предыдущих попыток. *divert(0)* — применяется для обозначения начала макроса.

Для того, чтобы можно было использовать макросы, их необходимо подключить при помощи директивы *include*.

```
include(`../m4/cf.m4')dnl
```

Сейчас мы рассмотрим основные макросы, которые используются при конфигурации sendmail. Остальные макросы будут рассмотрены по мере конфигурации почтового сервера.

МАКРОС VERSIONID.

Макрос VERSIONID применяется для идентификации версии создаваемого конфигурационного файла. Все, что указывается в качестве параметра, будет без изменений помещено в выходной файл, сразу после символа комментария.

```
VERSIONID(`My super mail server')dnl
```

МАКРОС OSTYPE.

В разных UNIX принято свое месторасположение и название дополнительных конфигурационных файлов sendmail. Каждый файл можно определить отдельно при помощи директивы define. Но для того, чтобы эти директивы не описывать каждый раз в файле mc, используют макрос OSTYPE.

```
OSTYPE(linux)dnl
```

В качестве параметра макроса указывают имя файла без расширения, находящегося в директории */usr/share/sendmail-cf/ostype*. В файле определяются параметры по умолчанию для каждого типа UNIX. Содержимое файла *linux.m4*:

```
VERSIONID(`$Id: linux.m4,v 8.13 2000/09/17 17:30:00 gshapiro Exp
$')
define(`confEBINDIR', `/usr/sbin')
ifdef(`PROCMAIL_MAILER_PATH',,
define(`PROCMAIL_MAILER_PATH', `/usr/bin/procmail'))
FEATURE(local_procmail)
```

МАКРОС DOMAIN.

Если у Вас есть много почтовых серверов с одинаковыми параметрами, можно создать файл в директории */usr/share/sendmail-cf/domain*, в котором эти параметры будут описаны.

Затем при помощи макроса DOMAIN подключить этот файл.

```
DOMAIN(my-domain)dnl
```

Например, файл *generic.m4* из директории domain:

```
VERSIONID(`$Id: generic.m4,v 8.15 1999/04/04 00:51:09 ca Exp $')
define(`confFORWARD_PATH',
`$z/.forward.$w+$h:$z/.forward+$h:$z/.forward.$w:$z/.forward')dnl
define(`confMAX_HEADERS_LENGTH', `32768')dnl
FEATURE(`redirect')dnl
FEATURE(`use_cw_file')dnl
```

МАКРОС MAILER.

Этот макрос применяется для определения того, каким образом sendmail может доставлять почту.

Точно так же, как и в предыдущих макросах, существует специальная директория mailer, в которой описаны агенты доставки почты, которыми может пользоваться sendmail.

Например, если предполагается доставка почты в локальные почтовые ящики и на удаленные транспортные агенты по протоколу smtp, необходимо описать эти возможности:

```
MAILER(local)dnl
MAILER(smtp)dnl
```

На самом деле mailer smtp включает сразу несколько транспортных агентов: smtp, esmtp, dsmtp, smtp8 и relay.

Если при доставке почты в локальные почтовые ящики требуется ее фильтрация, рекомендуется добавить поддержку программы procmail:

```
MAILER(procmail)dnl
```

МАКРОС FEATURE.

Макрос применяется для описания различных особенностей почтового сервера. Подавляющее большинство параметров будут определяться при помощи именно этого макроса. Более подробно макрос FEATURE мы будем рассматривать при конфигурации sendmail в других разделах.

ПОРЯДОК ОПИСАНИЯ МАКРОСОВ В ФАЙЛЕ ПРЕПРОЦЕССОРА M4.

Порядок описания файлов имеет значение. Особое внимание необходимо обратить на макросы MAILER. Они должны располагаться в конце файла, но перед описанием различных локальных конфигураций.

Ниже показан предпочтительный порядок описания макросов:

- VERSIONID
- OSTYPE
- DOMAIN
- local macro definitions
- FEATURE
- MAILER
- LOCAL_CONFIG
- LOCAL_RULE_*
- LOCAL_RULESETS

НАСТРОЙКА ПРОСТОГО ПОЧТОВОГО СЕРВЕРА.

Сейчас мы настроим простой почтовый сервер, принимающий почту для вашего домена[1].

Предположим, что имя домена st1.kryukov.biz.

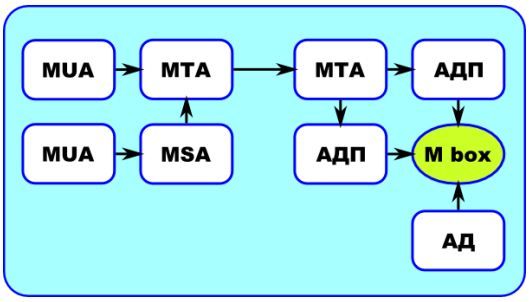
Итак, что мы должны получить:

- Почтовый сервер, обслуживающий домен st1.kryukov.biz.
- Для обслуживания почтовых ящиков будет использоваться программа cyrus-imap.

В результате, почтовый клиент на второй виртуальной машине сможет отправлять и принимать почту.

ВНИМАНИЕ! НА ДАННОМ ЭТАПЕ НЕ РАЗРЕШАЙТЕ ПРИНИМАТЬ НА ОТПРАВЛЕНИЕ ПОЧТУ С ВАШЕЙ ДОМАШНЕЙ ИЛИ РАБОЧЕЙ МАШИНЫ. У НАС ПОКА НЕ БУДЕТ СЕРЬЕЗНОЙ ЗАЩИТЫ ПОЧТОВОГО СЕРВЕРА. ПОЛЬЗУЙТЕСЬ ПОЧТОВОЙ ПРОГРАММОЙ, УСТАНОВЛЕННОЙ НА ВТОРОЙ ВИРТУАЛЬНОЙ МАШИНЕ.

Давайте посмотрим, что мы будем настраивать. Приведу еще раз схему, показанную в самом начале.



Почтовый сервер sendmail выполняет функции MTA. Его задача принять входящую почту, произвести необходимые проверки и отдать на доставку в почтовый ящик пользователя. Для доставки мы будем использовать протокол LMTP.

Cyrus IMAP будет выполнять функции АДП и АД. Он должен принять почту от MTA, поместить её в почтовый ящик пользователя и предоставить доступ пользователю к своему ящику.

НАСТРОЙКИ В DNS.

Для нормальной работы системы электронной почты, в DNS должна быть корректно настроена ваша зона.

- Во первых, запись MX должна ссылаться на машину, на которой будет работать ваш почтовый сервер.
- Во вторых, не мешало бы в зоне добавить запись TXT, помогающую бороться со спамом.

ЗАПИСЬ MX.

Для начала проверим, куда ссылается запись MX вашего домена.

```
[root@server ~]# dig st1.kryukov.biz MX


; <<>> DiG 9.3.4-P1 <<>> st1.kryukov.biz MX
```

[1] Имя домена вы должны получить у преподавателя.

```
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1185
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;st1.kryukov.biz.               IN      MX

;; ANSWER SECTION:
st1.kryukov.biz.        86400   IN      MX      5 ns.st1.kryukov.biz.

;; AUTHORITY SECTION:
st1.kryukov.biz.        82119   IN      NS      cosmos.kryukov.biz.
st1.kryukov.biz.        82119   IN      NS      ns.st1.kryukov.biz.

;; ADDITIONAL SECTION:
ns.st1.kryukov.biz.     82119   IN      A       84.19.179.61
cosmos.kryukov.biz.     86400   IN      A       84.19.178.61

;; Query time: 19 msec
;; SERVER: 192.168.56.1#53(192.168.56.1)
;; WHEN: Sun Nov 23 17:56:00 2008
;; MSG SIZE  rcvd: 119

[root@server ~]#
```

Как видно из вывода программы dig, запись MX для домена *st1.kryukov.biz* определена, и ссылается на машину *ns.st1.kryukov.biz* с IP адресом *84.19.179.61*.

Таким образом, все почтовые сервера, будут отправлять почту, предназначенную для домена *st1.kryukov.biz* на машину с IP адресом *84.19.179.61*.

Если запись MX по каким либо причинам не определена, добавьте ее в файле описания зоны[1].

ТЕХНОЛОГИЯ SPF.

Применение spf (Sender Policy Framework) не спасет вас от спама, но поможет другим серверам не принимать спам. Давайте посмотрим, как она работает.

Предположим, что у вас есть почтовый сервер. К вам приходит письмо, где в поле *mail from* написан email: *student@st1.kryukov.biz*. С точки зрения принимающего почтового сервера в этом письме все правильно: домен существует, пользователь которому предназначено письмо существует и его надо принять. Проблема заключается в том, что письмо с таким полем *mail from* может прийти с любого почтового сервера. И оно не обязательно приходит с нашей машины.

[1] Подробно настройка DNS севера рассматривается на курсе «Настройка Linux сервера, часть 1».

Некоторые администраторы почтовых серверов включают проверку соответствия IP адреса имени машины отправителя. Они обращаются к системе DNS и получают IP адрес машины *st1.kryukov.biz*. А затем производят обратное преобразование, по IP адресу получают имя машины. Если в обратном преобразовании получается другое имя, почта не принимается.

Вроде бы хороший выход из ситуации, однако это не так. Представьте себе, что на машине с одним IP адресом хостятся много почтовых серверов, обслуживающих почту для разных доменов. Или провайдер, у которого вы арендуете IP адрес, из вредности не хочет прописывать в зоне обратного преобразования имя вашей машины. Получается, что почту посланную с вашего, вполне легитимного сервера, принимать не будут. Но ведь вы не виноваты.

Администраторы, которые делают проверку по обратному преобразованию, на самом деле не защитят себя от спама. Более того, к их клиентам не будет доходить нормальная почта. Эти админы откровенные лентяи, которым не важны потребности их пользователей. Им лениво настроить нормальную защиту от спама.

Поэтому, никогда не включайте такую проверку.

Еще один вариант, базироваться на записи MX. Если письмо пришло с машины, которая определена в этой записи — это правильное письмо. Тоже не лучший вариант. Он будет работать когда у вас только один почтовый сервер, который принимает и отправляет почту для вашего домена. Но он не будет работать, когда у вас будет более сложная структура почтовых серверов.

При помощи MX мы определяем имя принимающего почту почтового сервера, а не отправляющего. Т.е. сервер, который в вашей организации занимается оправкой почты, может быть не определен при помощи MX.

Но вернемся к spf, как она нам поможет. При помощи spf можно явно указать какие почтовые сервера занимаются отправкой почты вашего домена. И если в поле *mail from* написано имя вашего домена, принимающий почтовый сервер обратится к spf с вопросом, а действительно этот сервер может оправлять почту этого домена?

Если в spf сервер написан, значит письмо примем, если нет — значит не примем.

Spf описывается на DNS сервере соответствующего домена. И реализуется при помощи записи TXT. В нашем случае можно написать так:

```
@ IN TXT "v=spf1 ip4:84.19.178.61 ip4:84.19.179.61 -all"
```

v=spf1 – определяет версию используемого протокола.

Остальные части записи указывают, откуда можно или нельзя принимать письма от этого домена. Это так называемые механизмы. Существует 8 видов механизмов, но мы рассмотрим только некоторые из них.

Каждый механизм может начинаться с префикса:

- + — разрешить.
- - — запретить.
- ~ — мягкий запрет. В этом случае письмо только помечается.
- ? — нейтральный. Письмо принимается.

Если префикс явно не указан, используется префикс плюс. Т.е. записи mx и +mx эдентичны.

Механизмы рассматриваются в том порядке, в котором они описаны. Если механизм сработал, остальные механизмы не рассматриваются.

МЕХАНИЗМ ALL.

Этот механизм срабатывает всегда, на все сервера. Имеется в виду *любой сервер*. Обычно пишется в самом конце списка механизмов.

Например:

"v=spf1 -all" — у этого домена нет почтовых серверов, не принимать ни от кого.

"v=spf1 -all" — для этого домена принимать почту с любых серверов.

МЕХАНИЗМ IP4.

Механизм имеет дополнительный параметр — IP адрес машины или сети.

Например:

"v=spf1 ip4:192.168.12.0/24 -all"

Принимать почту от серверов, чей IP входит в указанную сеть. От остальных серверов, почту не принимать.

МЕХАНИЗМ A.

Механизм имеет дополнительный параметр — имя домена, откуда можно принимать почту.

Механизм имеет несколько видов записи:

- a — все машины текущего домена.
- a:any.com — все машины домена any.com.

МЕХАНИЗМ MX.

Механизм mx определяет машины, описанные при помощи записи mx текущего домена.

В качестве праметра можно указывать домен, чьи записи mx следует рассматривать.

Теперь посмотрим запись, приведенную в качестве примера в самом начале главы.

```
"v=spf1 mx a:ns.st1.kryukov.biz -all"
```

Тут говориться, что можно принимать письма от машин, описанных в записях MX текущего домена, машины с именем *ns.st1.kryukov.biz*. От остальных машин, принимать почту от имени нашего домена нельзя.

К сожалению spf — не обязательный механизм, и не все почтовые сервера его поддерживают. Но эта запись не сложная и добавить ее в свою зону не составляет особого труда.

НАСТРОЙКА СЕРВЕРА SENDMAIL.

Как уже говорилось выше, настройка сервера — это создание или редактирование файла с макросами.

По умолчанию, в системе уже присутствует такой файл: */etc/mail/sendmail.mc*. Основная настройка транспортного агента — это редактирование данного файла.

ИЗМЕНЕНИЕ НАСТРОЕК ПО УМОЛЧАНИЮ.

Давайте посмотрим на настройки по умолчанию. Я сразу буду описывать некоторые незакоментированные параметры.

```
include(`/usr/share/sendmail-cf/m4/cf.m4')dnl
```

Подключается основной файл с макросами.

```
VERSIONID(`setup for linux')dnl
```

Строка, которая будет помещена в конфигурационный файл. Её можно заменить на любую другую. На работу сервера она не влияет. Например, так:

```
VERSIONID(`my superserver')dnl
```

```
OSTYPE(`linux')dnl
```

Определяет расположение дополнительных конфигурационных файлов. С учетом особенностей операционной системы Linux.

```
define(`confDEF_USER_ID', ``8:12'')dnl
define(`confTO_CONNECT', `1m')dnl
define(`confTRY_NULL_MX_LIST', `True')dnl
define(`confDONT_PROBE_INTERFACES', `True')dnl
define(`PROCMAIL_MAILER_PATH', `/usr/bin/procmail')dnl
define(`ALIAS_FILE', `/etc/aliases')dnl
```

Файл, в котором располагаются почтовые псевдонимы. В современных дистрибутивах Linux, этот файл находится в директории */etc/mail*. Но в нашем дистрибутиве, он будет располагаться по старинке прямо в */etc*.

```
define(`STATUS_FILE', `/var/log/mail/statistics')dnl
define(`UUCP_MAILER_MAX', `2000000')dnl
define(`confUSERDB_SPEC', `/etc/mail/userdb.db')dnl
define(`confPRIVACY_FLAGS', `authwarnings,novrfy,noexpn,restrictqrun')dnl
define(`confAUTH_OPTIONS', `A')dnl
```

Этот параметр отвечает за аутентификацию пользователей. К аутентификации пользователей мы вернемся немного позднее.

```
TRUST_AUTH_MECH(`EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
define(`confAUTH_MECHANISMS', `EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN
PLAIN')dnl
define(`confTO_IDENT', `0')dnl
FEATURE(`no_default_msa', `dnl')dnl
FEATURE(`smrsh', `/usr/sbin/smrsh')dnl
FEATURE(`mailertable', `hash -o /etc/mail/mailertable.db')dnl
FEATURE(`virtusertable', `hash -o /etc/mail/virtusertable.db')dnl
FEATURE(redirect)dnl
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl
FEATURE(use_ct_file)dnl
FEATURE(local_procmail, `', `procmail -t -Y -a $h -d $u')dnl
FEATURE(`access_db', `hash -T<TMPF> -o /etc/mail/access.db')dnl
FEATURE(`blacklist_recipients')dnl
EXPOSED_USER(`root')dnl
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

А вот это очень коварная строка. Тут говориться, что sendmail будет слушать запросы только на *lo* интерфейсе. Поэтому мы эту строку закомментируем, чтобы он начал слушать запросы на всех сетевых интерфейсах. Строка должна выглядеть так:

```
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

```
FEATURE(`accept_unresolvable_domains')dnl
```

Это трока тоже будет нами закомментирована. Она заставляет сервер принимать почту от доменов, которые не преобразуются через DNS. Строка должна выглядеть так:

```
dnl FEATURE(`accept_unresolvable_domains')dnl
```

```
LOCAL_DOMAIN(`localhost.localdomain')dnl
MAILER(smtp)dnl
MAILER(procmail)dnl
```

Последние две строки, говорят, что sendmail будет использовать два мейлера: *smtp* — для доставки почты по сети, и *procmail* для доставки почты в локальные почтовые ящики.

ДОБАВЛЕНИЕ ПАРАМЕТРОВ.

СТРОКА ПРИГЛАШЕНИЯ.

Теперь посмотрим параметры, которые мы добавим, или раскоментируем.

```
define(`confSMTP_LOGIN_MSG', `wellcome to my server')dnl
```

Этот параметр определяет строку, которая будет выводится в качестве приглашения, при подключении к серверу.

Помните?

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
220 Artur ESMTP mail server
```

Обратите внимание на последнюю строку. Именно тут будет выводится сообщение, определенное в параметре. Зачем нашим *гостям* знать, какой сервер у нас используется?

ОЧЕРЕДЬ НА ОТПРАВЛЕНИЕ.

```
define(`confTO_QUEUEWARN', `4h')dnl
define(`confTO_QUEUERETURN', `5d')dnl
```

Эти параметры определяют, сколько времени письмо будет находиться в очереди на отправку.

QUEUEWARN — через сколько времени, пользователю, отправившему письмо, будет возвращено предупреждение о том, что письмо не было доставлено сразу, а было помещено в очередь на отправку. Мне кажется, что 4 часа — это много. Наверное, час лучший вариант.

QUEUERETURN — определяет через сколько времени письмо будет удалено из очереди на отправку. При удалении, пользователю посылается сообщение о невозможности доставки. 5 дней, при хорошей нагрузке на сервер, может привести к тому, что очередь на отправку будет переполнена и сервер начнет тормозить. По моему мнению, 2 дня — это оптимальный вариант.

В результате получится так:

```
define(`confTO_QUEUEWARN', `1h')dnl
```

```
define(`confTO_QUEUERETURN', `2d')dnl
```

ОГРАНИЧЕНИЯ НА РАЗМЕР ПИСЬМА И КОЛИЧЕСТВО ПОЛУЧАТЕЛЕЙ.

По умолчанию sendmail не ограничивает размер тела письма. Об этом ограничении вам придется позаботиться самим.

```
define(`confMAX_MESSAGE_SIZE',`10000000')dnl
```

Параметр — размер тела письма в байтах.

Так же можно ограничить количество получателей письма.

```
define(`confMAX_RCPTS_PERMESSAGE',`10')dnl
```

Значение параметра — количество email в полу *To*.

ЗАЩИТА ОТ DDOS.

```
define(`confMAX_DAEMON_CHILDREN', `20')dnl
```

Определяет, сколько одновременно процессов sendmail может находиться в оперативной памяти.

Для оправки и получения письма, запускается отдельный экземпляр sendmail. Если количество экземпляров не ограничивать, то мы получим неприятности на свою голову. Этот параметр нужно обязательно определить!

Максимальное количество экземпляров, которые должны находится в оперативной памяти, вы должны определить сами.

```
define(`confCONNECTION_RATE_THROTTLE', `3')dnl
```

Этот параметр определяет количество соединений в секунду. Параметр желательно определять, для снижения нагрузки на почтовый сервер.

ПОДКЛЮЧЕНИЕ CYRUS IMAP.

В качестве хранилища входящей почты мы будем использовать сервер Cyrus IMAP. Поэтому необходимо добавить несколько параметров, связанных с ним.

```
define(`confLOCAL_MAILER', `cyrusv2')dnl
define(`CYRUSV2_MAILER_ARGS', `FILE /var/lib/imap/socket/lmtp')dnl
```

Первая строка говорит, что для локальной доставки почты будет использоваться cyrus.

Вторая строка определяет доставку писем от sendmail к cyrus через локальный файл, по протоколу LMTP.

В самом конце файла, необходимо разобраться с мейлерами. Обязательно закомментируем майлер procmail и добавим майлер cyrusv2.

```
dnl MAILER(procmail)dnl
MAILER(cyrusv2)dnl
```

ИТОГОВЫЙ КОНФИГУРАЦИОННЫЙ ФАЙЛ.

После всех изменений, наш конфигурационный файл будет выглядеть так:

```
include(`/usr/share/sendmail-cf/m4/cf.m4')dnl
```

```
VERSIONID(`setup for linux')dnl
OSTYPE(`linux')dnl
define(`confSMTP_LOGIN_MSG', `wellcome to my server')dnl
define(`confDEF_USER_ID', ``8:12'')dnl
define(`confTO_CONNECT', `1m')dnl
define(`confTRY_NULL_MX_LIST', `True')dnl
define(`confDONT_PROBE_INTERFACES', `True')dnl
define(`PROCMAIL_MAILER_PATH', `/usr/bin/procmail')dnl
define(`ALIAS_FILE', `/etc/aliases')dnl
define(`STATUS_FILE', `/var/log/mail/statistics')dnl
define(`UUCP_MAILER_MAX', `2000000')dnl
define(`confUSERDB_SPEC', `/etc/mail/userdb.db')dnl
define(`confPRIVACY_FLAGS', `authwarnings,novrfy,noexpn,restrictqrun')dnl
define(`confAUTH_OPTIONS', `A')dnl
TRUST_AUTH_MECH(`EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
define(`confAUTH_MECHANISMS', `EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN
PLAIN')dnl
define(`confTO_QUEUEWARN', `1h')dnl
define(`confTO_QUEUERETURN', `2d')dnl
define(`confTO_IDENT', `0')dnl
define(`confMAX_MESSAGE_SIZE',`10000000')dnl
define(`confMAX_RCPTS_PERMESSAGE',`10')dnl
FEATURE(`no_default_msa', `dnl')dnl
FEATURE(`smrsh', `/usr/sbin/smrsh')dnl
FEATURE(`mailertable', `hash -o /etc/mail/mailertable.db')dnl
FEATURE(`virtusertable', `hash -o /etc/mail/virtusertable.db')dnl
FEATURE(redirect)dnl
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl
FEATURE(use_ct_file)dnl
define(`confMAX_DAEMON_CHILDREN', `20')dnl
define(`confCONNECTION_RATE_THROTTLE', `3')dnl
FEATURE(local_procmail, `', `procmail -t -Y -a $h -d $u')dnl
FEATURE(`access_db', `hash -T<TMPF> -o /etc/mail/access.db')dnl
FEATURE(`blacklist_recipients')dnl
EXPOSED_USER(`root')dnl
define(`confLOCAL_MAILER', `cyrusv2')dnl
define(`CYRUSV2_MAILER_ARGS', `FILE /var/lib/imap/socket/lmtp')dnl
LOCAL_DOMAIN(`localhost.localdomain')dnl
MAILER(smtp)dnl
MAILER(cyrusv2)dnl
```

СОЗДАНИЕ КОНФИГУРАЦИОННОГО ФАЙЛА.

Мы получили файл с макросами. Теперь его необходимо превратить в конфигурационный файл *sendmail.cf*.

Перейдем в директорию */etc/mail*.

```
cd /etc/mail
```

Запустим препроцессор. Результаты работы, препроцессор выдает на стандартный вывод.

```
m4 sendmail.mc > sendmail.cf
```

Поскольку стандартный вывод программы перенаправлен, на экран не должны выводиться сообщения. Если программа что то показывает на экране — это ошибка.

Основная синтаксическая ошибка — неправильно указанные открывающая и закрывающая скобки.

Посмотрите первые 45 строк конфигурационного файла sendmail.cf.

```
head -45 sendmail.cf
##########################################################################
#####
#####   DO NOT EDIT THIS FILE!  Only edit the source .mc file.
#####
##########################################################################
##########################################################################
#####  $Id: cfhead.m4,v 8.116 2004/01/28 22:02:22 ca Exp $  #####
#####  $Id: cf.m4,v 8.32 1999/02/07 07:26:14 gshapiro Exp $  #####
#####  setup for linux  #####
#####  $Id: linux.m4,v 8.13 2000/09/17 17:30:00 gshapiro Exp $  #####
#####  $Id: local_procmail.m4,v 8.22 2002/11/17 04:24:19 ca Exp $  #####
#####  $Id: no_default_msa.m4,v 8.2 2001/02/14 05:03:22 gshapiro Exp $
####
#####  $Id: smrsh.m4,v 8.14 1999/11/18 05:06:23 ca Exp $  #####
#####  $Id: mailertable.m4,v 8.25 2002/06/27 23:23:57 gshapiro Exp $  #####
```

В комментариях есть наша метка.

```
#####  setup for linux  #####
```

Конфигурация sendmail завершена, теперь необходимо заняться конфигурацией IMAP сервера.

ОПРЕДЕЛЕНИЕ ДОМЕНА.

Осталось только определить имя домена, для которого sendmail будет принимать почту.

В файле с макросами мы использовали следующий макрос:

```
FEATURE(use_cw_file)dnl
```

Он означает, что в файле */etc/mail/local-host-names sendmail* будет искать имена доменов, для которых он должен принимать письма. Это текстовый файл, в котором домены пишуться по одному на строке.

Добавим туда имя своего домена.

```
# cat /etc/mail/local-host-names
# local-host-names - include all aliases for your machine here.
st1.kryukov.biz
#end
```

Очень важно запомнить, что последняя строка этого файла не читается. Поэтому она должны быть пустой или в ней должен быть комментарий.

ПСЕВДОНИМЫ.

Все бесплатные почтовые сервера в Linux позволяют использовать механизм почтовых псевдонимов. Sendmail не исключение.

Обычно файл, где вы будете определять псевдонимы, на ходится в директории /etc/mail и называется aliases. Но в CentOS он находится непосредственно в директории /etc.

```
/etc/aliases
```

Формат файла псевдонимов очень простой:

- одна запись — одна строка.
- В строке два поля, разделяемые двоеточием.
- Первое поле — имя псевдонима.
- Второе поле — куда отправить почту.

Например, необходимо всю почту, приходящую на почтовый ящик *root*, перенаправить в локальный почтовый ящик пользователя *student*. Запись будет выглядеть следующим образом:

```
root: student
```

Кстати, во время работы сервера на почтовый ящик пользователя root будут посылаться служебные письма. Например, отчеты генерируемые программой *logwatch*[1]. Раньше эти письма сохранялись в файле */var/spool/mail/root*. Но мы, для хранение писем будем использовать программу Cyrus IMAP, поэтому письма, предназначенные пользователю *root* к сожалению начнут теряться. Поэтому обязательно сделайте перенаправление на действующий почтовый ящик, в нашем случае — ящик пользователя *student*.

Имя почтового ящика, написанного в первой части может не существовать. Например, вся почта, приходящая на несуществующий ящик *supprot*, необходимо перенапрявлять на другой локальный почтовый ящик или email. В этом случае не надо заводить пользователя support и его почтовый ящик. Достаточно написать:

```
support: user_po_box
```

Где вместо *user_po_box* следует написать имя необходимого вам почтового ящика.

В правой части можно писать несколько записей, разделяя их через запятую.

```
support: student,director
```

В этом случае письмо, пришедшее по адресу *support*, будет перенаправлено в почтовые ящики пользователей **STUDENT** и **DIRECTOR**.

В правой части можно писать email, куда следует пересылать письмо.

```
support: artur@kryukov.biz,director,student
```

[1] Программа logwatch следит за файлами журнальной регистрации. Если в файлах обнаруживаются странные записи, она формирует отчет и отправляет его в виде письма на почтовый ящик пользователя root.

В этом примере показано, что можно одно письмо переслать сразу на несколько, как локальных, так и удаленных почтовых ящиков.

Если список почтовых ящиков очень большой, его можно поместить во внешний файл. Тогда запись будет такой:

```
support: :include:/путь/к/файлу/со/списком
```

Очень интересна возможность передавать письмо на обработку сторонней программе.

```
list: "|/путь/к/программе"
```

В этом примере, вся почта, приходящая на почтовый ящик list будет передана на стандартный вход программы.

Файл /etc/aliases — это обыкновенный текстовый файл и с ним почтовый сервер не работает. Поэтому, после редактирования файла, необходимо его преобразовать в файл базы данных, который использует sendmail. Это делается при помощи программы newaliases.

```
[root@server ~]# newaliases
/etc/aliases: 77 aliases, longest 10 bytes, 776 bytes total
[root@server ~]#
```

КОНФИГУРАЦИЯ CYRUS IMAP.

По умолчанию у нас установлен сервер dovecot, но я еще плотно с ним не работал, и не знаю, насколько он хорош. Поэтому мы установим классический, проверенный годами сервер Cyrus IMAP.

```
# yum install cyrus-imapd cyrus-imapd-utils
```

А dovecot удалим.

```
# yum remove dovecot
```

Обязательно убедитесь, что установлен пакет cyrus-sasl, он необходим для аутентификации пользователей.

```
# rpm -qa cyrus-sasl*
```

Если его нет, установите.

КОНФИГУРАЦИОННЫЕ ФАЙЛЫ.

У Cyrus IMAP два конфигурационных файла.

Сначала отредактируем общий конфигурационный файл */etc/cyrus.conf*.

Нас интересует раздел SERVICES, в котором описываются сервисы, обслуживаемые программой.

imap — IMAP сервер.

imaps — IMAP с SSL шифрованием.

pop3 — POP3 сервер.

pop3s — POP3 с SSL шифрованием.

В cyrus imap рекомендуется использовать именно IMAP протокол, так как в этом режиме сервер работает гораздо быстрее, да и у IMAP много полезных плюшек.

IMAPS, как и POP3S vs пока отключим. Для их работы необходимо создать набор ключей. Этим мы займемся позднее.

В результате, эта секция будет выглядеть следующим образом.

```
SERVICES {
  # add or remove based on preferences
  imap          cmd="imapd" listen="imap" prefork=5
  pop3          cmd="pop3d" listen="pop3" prefork=3
  lmtpunix      cmd="lmtpd" listen="/var/lib/imap/socket/lmtp" prefork=1
}
```

Второй конфигурационный файл /etc/imap.conf. Мы поправим некоторые параметры.

Во первых, добавим в администраторы сервера пользователя student. Во вторых, закомментируем строки связанные с SSL. В результате получается вот такой файл:

```
configdirectory: /var/lib/imap
partition-default: /var/spool/imap
admins: cyrus student
sievedir: /var/lib/imap/sieve
sendmail: /usr/sbin/sendmail
hashimapspool: true
sasl_pwcheck_method: saslauthd
sasl_mech_list: PLAIN
#tls_cert_file: /etc/pki/cyrus-imapd/cyrus-imapd.pem
#tls_key_file: /etc/pki/cyrus-imapd/cyrus-imapd.pem
#tls_ca_file: /etc/pki/tls/certs/ca-bundle.crt
```

ЗАПУСК ПОЧТОВОГО СЕРВЕРА.

Во первых, убедимся, что включен сервис аутентификации пользователей.

```
# /etc/init.d/saslauthd status
```

Если он не включен, запустим его и сделаем так, что бы он включался при каждом запуске системы.

```
# /etc/init.d/saslauthd start
# chkconfig saslauthd on
```

Запускаем IMAP сервер и делаем так, что бы он включался при каждом запуске системы.

```
# /etc/init.d/cyrus-imapd start
# chkconfig cyrus-imapd on
```

Перезапускаем sendmail.

```
# /etc/init.d/sendmail restart
```

Контролируем, открыты ли на прослушивание необходимые порты.

SMTP

```
# netstat -nlp | grep :25
tcp        0      0 0.0.0.0:25                  0.0.0.0:*
LISTEN      13945/sendmail: acc
```

POP3

```
# netstat -nlp | grep :110
```

```
tcp        0      0 0.0.0.0:110                 0.0.0.0:*
LISTEN      13896/cyrus-master

tcp        0      0 :::110                      :::*
LISTEN      13896/cyrus-master
```

IMAP

```
# netstat -nlp | grep :143

tcp        0      0 0.0.0.0:143                 0.0.0.0:*
LISTEN      13896/cyrus-master

tcp        0      0 :::143                      :::*
LISTEN      13896/cyrus-master
```

НАСТРОЙКА FIREWALL.

Открываем на редактирование файл с настройками firewall[1].

```
# vim ~/bin/rc.fw
```

В функции init добавим следующие строки.

```
### Mail server

$IPT -A INPUT -p tcp -m multiport --dports 25,110,143 -j ACCEPT
```

Сохраним файл, и применим изменения.

```
# rc.fw init
```

ПОЧТОВЫЕ ЯЩИКИ ПОЛЬЗОВАТЕЛЕЙ.

Для добавления почтового ящика нам придется делать два действия:

- Добавить пользователя в системе. Он необходим для аутентификации.
- Создать почтовый ящик в IMAP сервере.

ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯ В СИСТЕМЕ.

Конечно, Cyrus IMAP позволяет создавать своих пользователей, но это очень опасный механизм. В этом случае вся информация о пользователях, включая их пароли в открытом виде, будет хранится в файле */etc/sasldb2*. Поэтому мы будем пользоваться стандартным механизмом аутентификации, и нам придется создавать пользователей в системе.

Теперь посмотрим, какие атрибуты нужны нашему пользователю.

1. Пользователь не должен входить в систему по ssh. Поэтому ставим ему shell */sbin/nologin*.
2. Пользователю не нужна домашняя директория.

Добавим тестового пользователя и создадим ему пароль.

```
# useradd -s /sbin/nologin -d /home test

useradd: внимание: домашний каталог уже существует.

Никакие файлы из каталога skel копироваться не будут.

# passwd test
```

В дальнейшем, что бы поменять пароль на почтовый ящик, достаточно поменять пароль пользователя при помощи программы *passwd*.

[1] Подробно об этом файле, как и настройках firewall рассказывается на курсе «Настройка Linux сервера, часть 1».

ДОБАВЛЕНИЕ ПОЧТОВОГО ЯЩИКА.

Для того, что бы добавить почтовый ящик в IMAP сервере, необходимо пользоваться консолью управления *cyradm*.

Подключаемся к IMAP серверу как пользователь student. На приглашения ввода пароля, вводим пароль пользователя student.

```
# cyradm -u student localhost
IMAP Password:
               server.st1.kryukov.biz>
```

Список доступных команд можно получить при помощи команды help.:

```
server.st1.kryukov.biz> help
authenticate, login, auth          authenticate to server
chdir, cd                          change current directory
createmailbox, create, cm          create mailbox
deleteaclmailbox, deleteacl, dam   remove ACLs from mailbox
deletemailbox, delete, dm          delete mailbox
disconnect, disc                   disconnect from current server
exit, quit                         exit cyradm
help, ?                            show commands
info                               display mailbox/server metadata
listacl, lam, listaclmailbox       list ACLs on mailbox
listmailbox, lm                    list mailboxes
listquota, lq                      list quotas on specified root
listquotaroot, lqr, lqm            show quota roots and quotas for mailbox
mboxcfg, mboxconfig                configure mailbox
reconstruct                        reconstruct mailbox (if supported)
renamemailbox, rename, renm        rename (and optionally relocate) mailbox
server, servername, connect        show current server or connect to server
setaclmailbox, sam, setacl         set ACLs on mailbox
setinfo                            set server metadata
setquota, sq                       set quota on mailbox or resource
subscribe, sub                     subscribe to a mailbox
unsubscribe, unsub                 unsubscribe from a mailbox
version, ver                       display version info of current server
xfermailbox, xfer                  transfer (relocate) a mailbox to a dif-
ferent server
server.st1.kryukov.biz>
```

Для создания почтового ящика пользователя используется команда *cm* (create mailbox).

Почтовые ящики имеют иерархическую структуру. Корень начинается от user. Затем идет ящик, соответствующий логину пользователя (обычно он и является так называемым INBOX). В ящике пользователя можно создавать под папки.

Обычно администратору достаточно создать ящик пользователя, а подпапки пользователи создают сами.

Создадим ящик пользователя test.

```
server.st1.kryukov.biz> cm user.test
```

Посмотрим список почтовых ящиков, при помощи команды *lm* (list mailboxes).

```
server.st1.kryukov.biz> lm
user.test (\HasNoChildren)
server.st1.kryukov.biz>
```

Конечно, добавлять почтовые ящики вручную не очень удобно. В одной из следующих глав мы увидим скрипт, позволяющий автоматизировать эту рутинную процедуру.

ПРАВА ДОСТУПА.

Проверим права доступа на ящик. Это делается при помощи команды *lam* (list acl mailboxes).

```
server.st1.kryukov.biz> lam user.test
test lrswipkxtecda
server.st1.kryukov.biz>
```

Немного об ACL (access control list) или правах доступа. Ниже приведен список возможных ACL.

- l (lookup) — Пользователь может видеть, что ящик существует.
- r (read) — Пользователь может читать из ящика. Пользователь может выбрать ящик, получить данные, произвести поиск и скопировать данные из ящика.
- s (seen) — Удержание состояния пользователя. *Seen* и *Recent* флаги для пользователя устанавливаются.
- w (write) — Пользователь может менять флаги ки ключевые слова кроме *Seen* и *Deleted* (последние управляются др. правами доступа).
- i (insert) —Пользователь может помещать в ящик новое сообщение.
- p (post) — Пользователь может посылать почту на адрес данного ящика. Это право отличается от права *i* тем, что в данном случае система доставки добавляет к письму служебную информацию.
- c (create) — Пользователи могут создавать подъящики от этого ящика, а так же удалять или переименовывать этот ящик.
- d (delete) — Пользователь может хранить флаг *Deleted* и производить вычеркивание.
- a (administer) — Пользователь может менять ACL ящика.

Что бы изменить права доступа на ящик следует использовать команду *sam* (set ACL mailboxes).

Например, что бы дать все права на ящик пользователю *test*, команда будет выглядеть следующим образом:

```
server.st1.kryukov.biz> sam user.test test all
server.st1.kryukov.biz> lam user.test
test lrswipkxtecda
server.st1.kryukov.biz>
```

Вы можете дать доступ к ящику и другим пользователям. В следующем примере показано как установить права доступа к ящику *test* пользователю *cyrus*.

```
server.st1.kryukov.biz> sam user.test cyrus lri
server.st1.kryukov.biz> lam user.test
test lrswipkxtecda
cyrus lri
```

```
server.st1.kryukov.biz>
```

ОГРАНИЧЕНИЕ РАЗМЕРА ПОЧТОВОГО ЯЩИКА.

Квотрирование почтовых ящиков в Cyrus IMAP производится встроенными средствами.

Квота корня применяется к сумме объема использованного ящиком и всеми его подъящиками у которых нет своей квоты. Это означает, что у ящика может быть только одна корневая квота.

Для просмотра квот используется команда *lq* (list quota). Для установки — sq (set quota).

```
server.st1.kryukov.biz> sq user.test 100000
quota:100000
server.st1.kryukov.biz> lq user.test
 STORAGE 0/100000 (0%)
server.st1.kryukov.biz>
```

В этом примере на ящик user.test установлена квота размером 100Мб. Этак квота, будет распространяться и на вложенные ящики.

Доставка почты — это особый случай. При доставке сообщения в ящик, корневая квота этого ящика не должна быть превышена. Если квота не превышена, то только одно сообщение может быть доставлено независимо от его размера. Это вызывает превышение квоты данным ящиком, пользователь ставится в известность о превышении квоты. Если бы в таком случае доставка не разрешалась, то пользователь не узнал бы, что была почта которую нельзя доставить.

УДАЛЕНИЕ ЯЩИКА.

Для удаления папки или почтового ящика используется команда *dm* (delete mailbox).

Сначала создадим вложенный ящик.

```
server.st1.kryukov.biz> cm user.test.list
server.st1.kryukov.biz>
```

Посмотрим, какие ящики есть.

```
server.st1.kryukov.biz> lm user.test*
user.test (\HasChildren)          user.test.list (\HasNoChildren)
server.st1.kryukov.biz>
```

Удалим вложенную директорию list.

```
server.st1.kryukov.biz> dm user.test.list
deletemailbox: Permission denied
server.st1.kryukov.biz>
```

Ух ты! А удалить то нельзя. Посмотрим права на ящик.

```
server.st1.kryukov.biz> lam user.test
test lrswipkxtecda
cyrus lri
server.st1.kryukov.biz>
```

Пользователя student, с правами которого мы сейчас работаем, в списке нет. Значит, мы не сможем удалить ящик. Но не беда, мы же администраторы. Разрешим себе удалять папку и удалим ее.

```
server.st1.kryukov.biz> sam user.test.list student c
server.st1.kryukov.biz> lam user.test.list
test lrswipkxtecda
student kxc
cyrus lri
server.st1.kryukov.biz> dm user.test.list
server.st1.kryukov.biz> lm user.test*
user.test (\HasNoChildren)
server.st1.kryukov.biz>
```

ЛАБОРАТОРНАЯ РАБОТА. СОЗДАНИЕ ПОЧТОВОГО ЯЩИКА.

Цель этой лабораторной работы — создать почтовый ящик для пользователя student.

Мы будем пользоваться программой cyradm. И тут возникнет маленькое затруднение, student будет видеть почтовый ящик не как user.student, а как локальный бокс INBOX. С точки зрения администрирования это не удобно.

Для управления почтовыми ящиками мы будем пользоваться учетной записью пользователя cyrus. Сам пользователь уже существует, остается добавить ему пароль.

```
passwd cyrus
```

Запускаем программу управления.

```
# cyradm -u cyrus localhost
IMAP Password:
               server.st1.kryukov.biz>
```

Создаем почтовый ящик.

```
server.st1.kryukov.biz> cm user.student
```

Даем все права на ящик пользователю student.

```
server.st1.kryukov.biz> sam user.student student all
```

Для нормальной работы почтового клиента нам так же потребуются папки Trash и Sent. Создадим их.

```
server.st1.kryukov.biz> cm user.student.Sent
server.st1.kryukov.biz> cm user.student.Trash
server.st1.kryukov.biz> sam user.student.Sent student all
server.st1.kryukov.biz> sam user.student.Trash student all
```

Не забудьте подписаться на эти папки в почтовом клиенте.

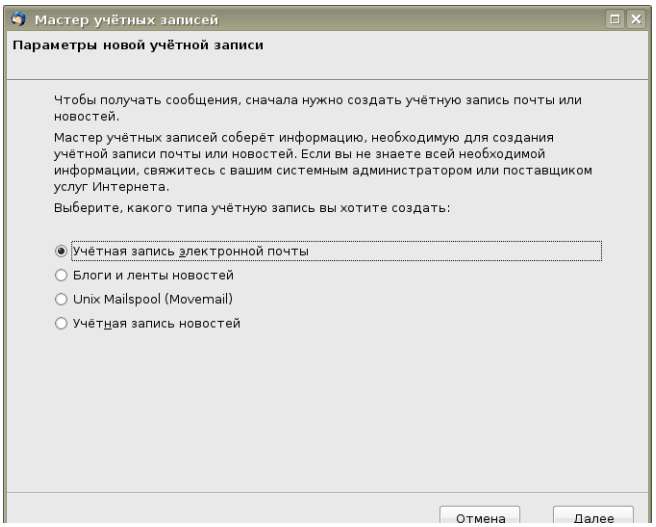
ПОДКЛЮЧЕНИЕ ПОЧТОВОГО КЛИЕНТА.

Запустите вторую виртуальную машину, на которой установлен Linux с графической оболочкой.

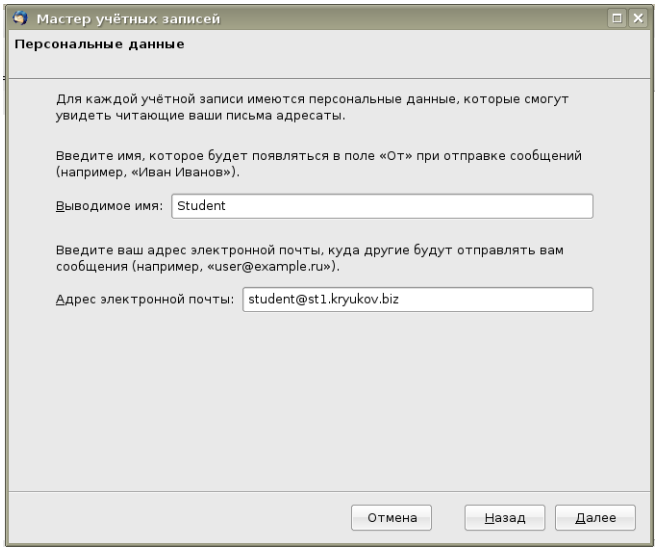
Проверьте настройку сети. Работает ли DNS.

Теперь настроим почтовый клиент Mozilla Thunderbird.

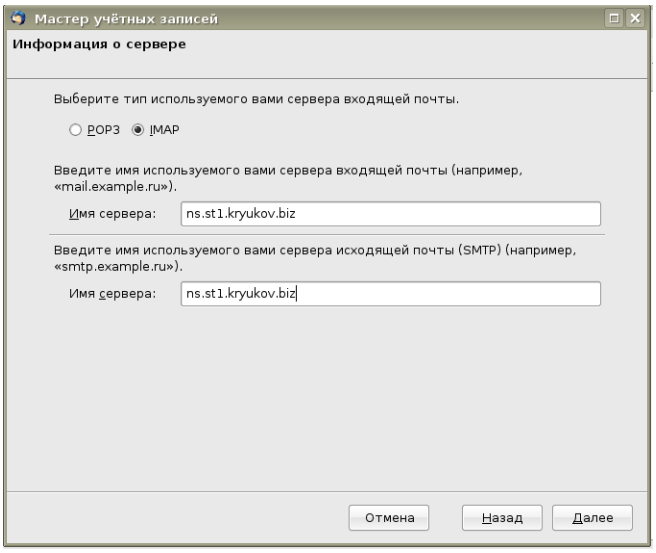
При первом запуске вам предложат импортировать настройки из других почтовых клиентов. Поскольку у на их нет, просто нажимайте кнопку *Next*.



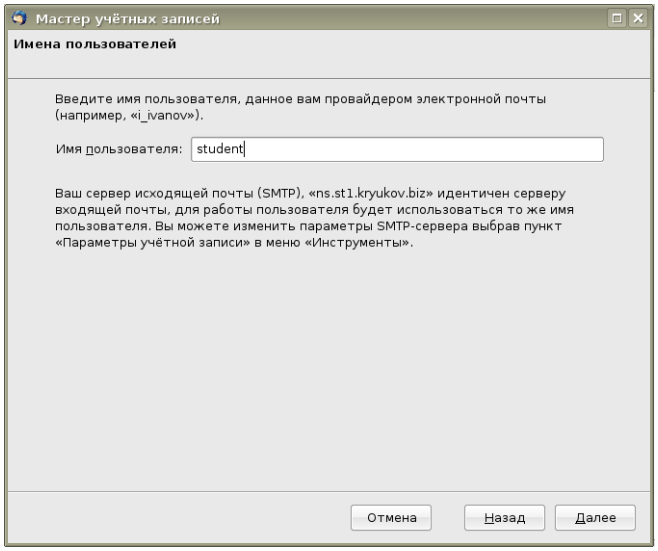
Выбираем *Учетная запись электронной почты*. Нажимаем кнопку *Next*.



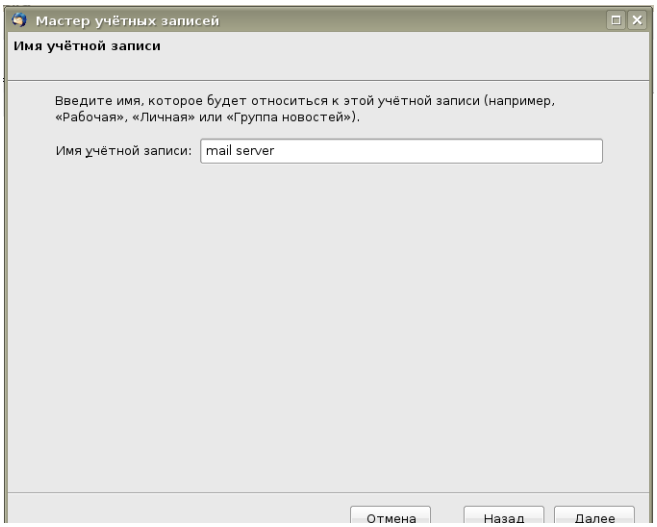
Вводим имя пользователя и его email. Нажимаем кнопку *Далее*.



Вводим имя или IP адрес вашего почтового сервера, как сервера для приема и отправки почты. Предварительно убедившись, что он вам доступен по имени. И нажимаем кнопку *Далее*.

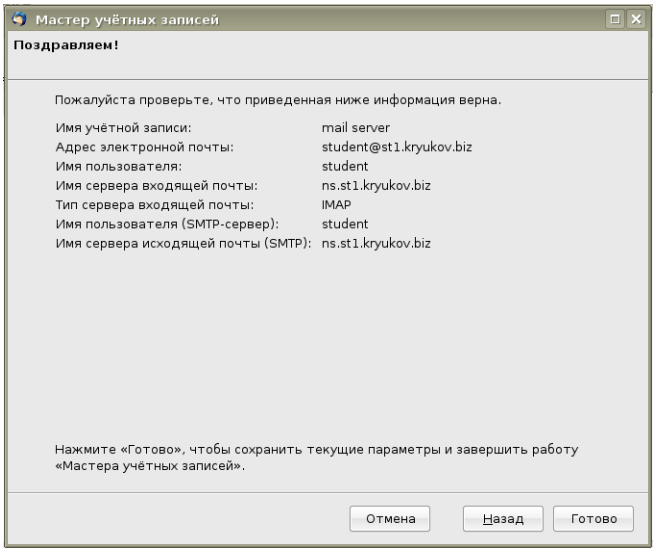


Вводим имя пользователя *student* и нажимаем кнопку *Далее*.



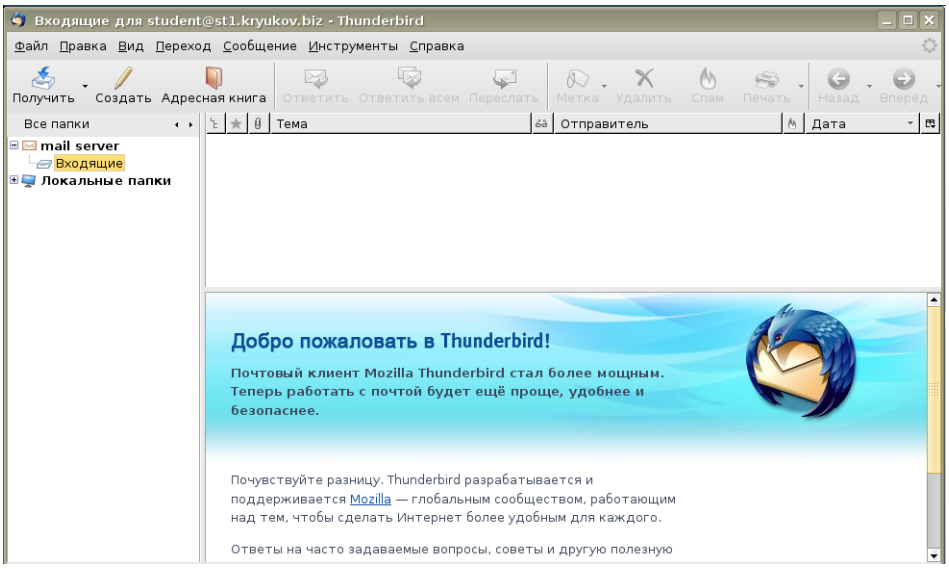
Вводим имя учетной записи. Тут можно написать все что угодно.

Нажимаем кнопку *Далее*.

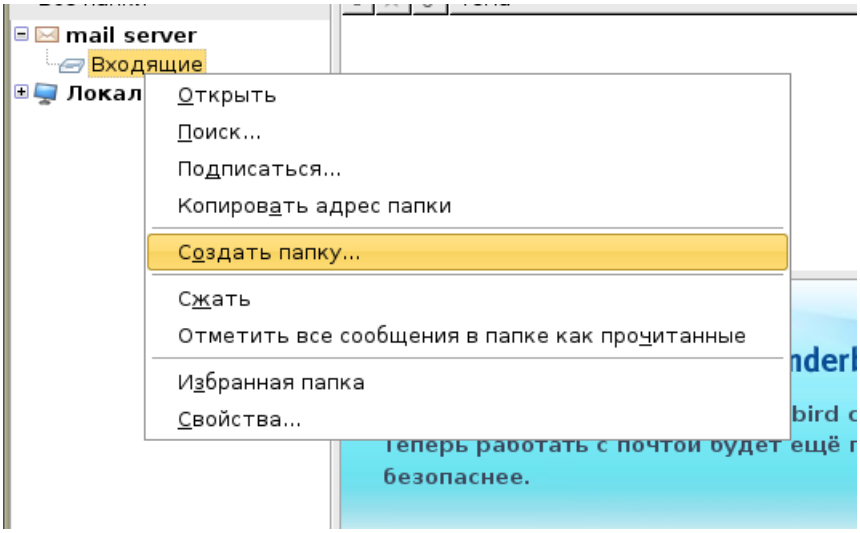


Подтвердите правильность настроек.

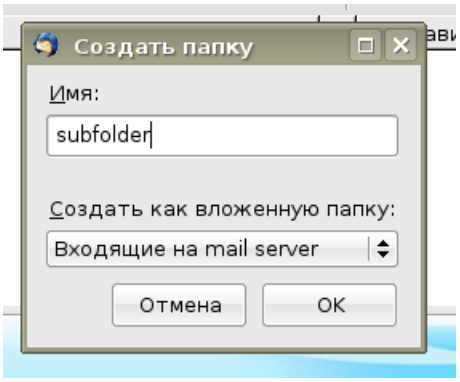
Программа подключиться к IMAP серверу и попросит ввести пароль пользователя. Вы введете пароль пользователя student.



Нажмите правую кнопку мыши на папке *Входящие*. И выберите *Создать папку*.



В появившемся окне введите имя папки.

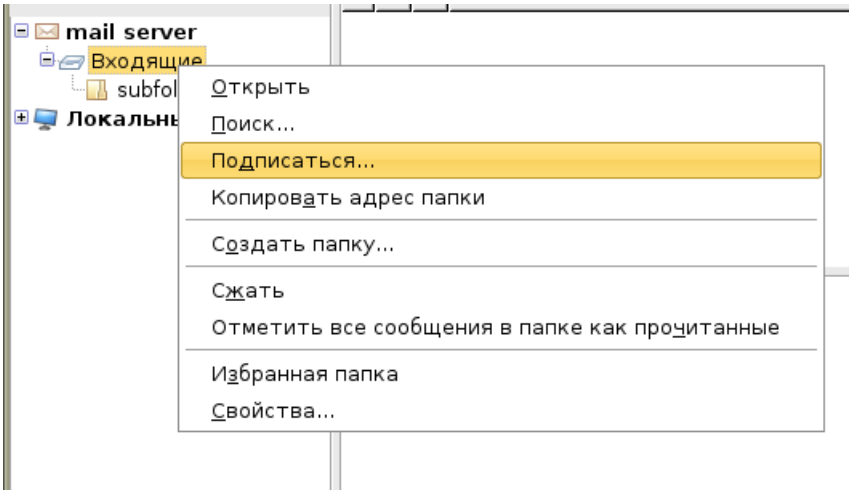


И нажмите кнопку OK.

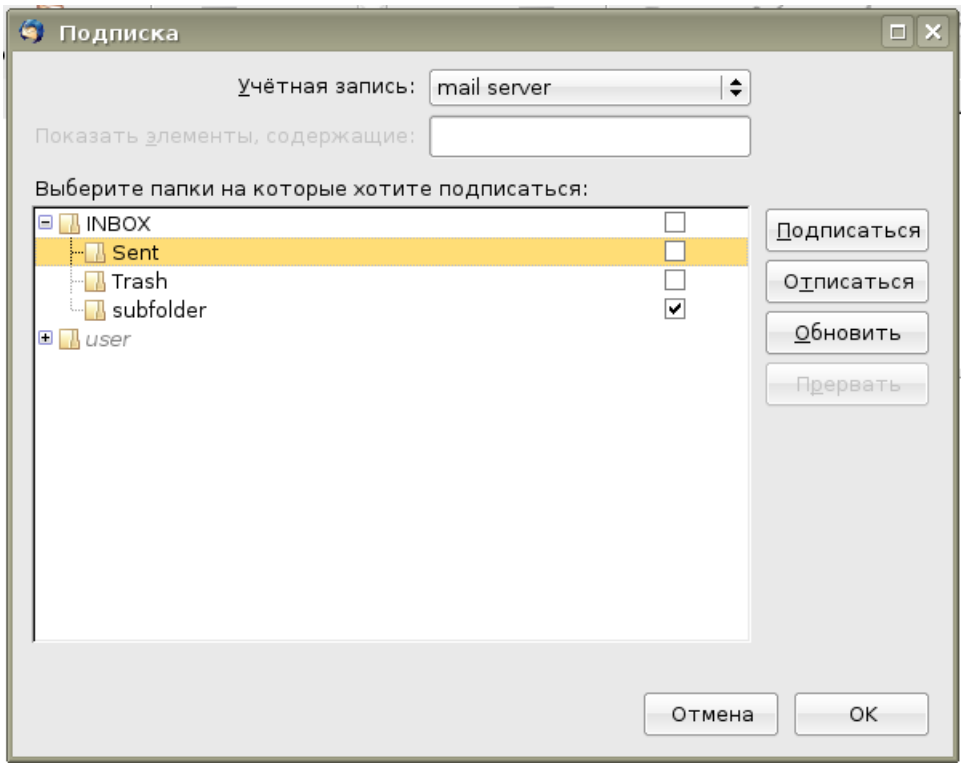
ПОДПИСКА НА ПАПКИ.

Когда вы на сервер создаете дополнительные папки для клиента, клиент по умолчанию их не отображает и ими не пользуется.

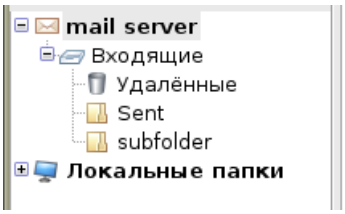
Что бы воспользоваться этими папками, на них нужно *подписаться*. Нажмите правую кнопку мыши на папке *Входящие* и выберите *Подписаться*.



В появившемся окне раскройте список INBOX.

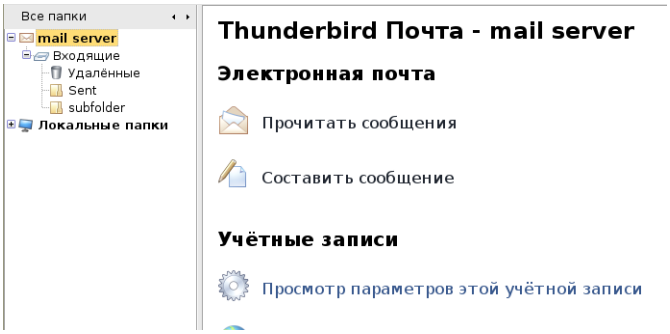


Поставьте галочки напротив папок, на которые вы хотите подписаться и нажмите *OK*.

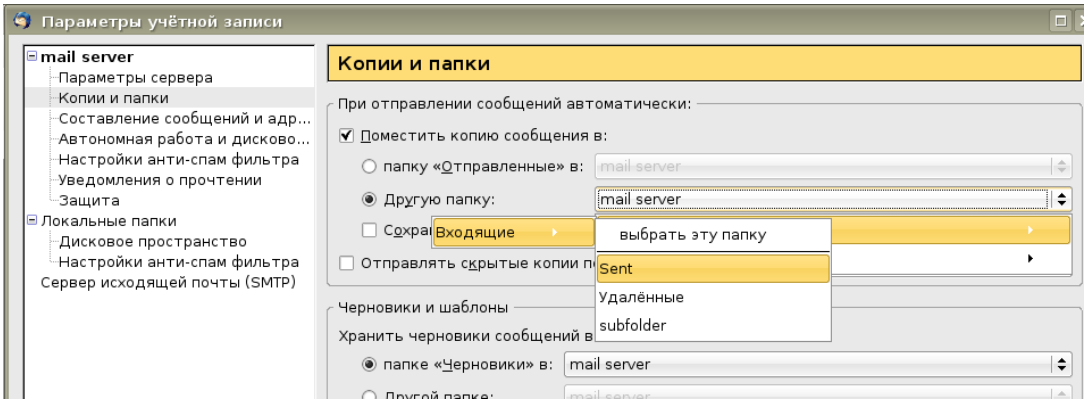


В результате, клиент будет отображать папку Trash как Удаленные. А вот для того, что бы вся исходящая почта попадала в папку Sent, клиент нужно настроить.

Выберите *mail server*.



После этого в правой части выберите *Просмотр параметров этой учетной записи*.



В появившемся окне выбираете *Копии и папки*.

В разделе *При отправлении сообщение автоматически*, выбираете *Другую папку*.

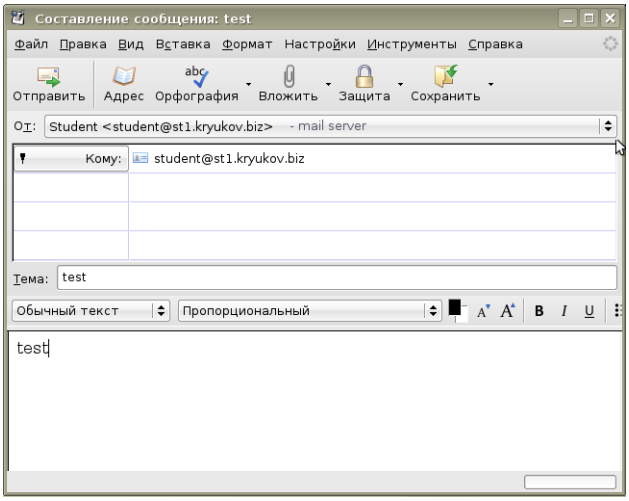
В списке выбираете меню *mail server -> Входящие -> Sent*.

Нажимаете OK.

В принципе, не плохо было бы аналогичным образом создать папку для черновиков и шаблонов.

ОТПРАВКА ПОЧТЫ.

Теперь попытаемся отправить письмо самому себе. Нажимаем кнопку *Создать*, и водим текст письма.



Нажимаем кнопку *Отправить*.

Вас спросят пароль пользователя student. Введите пароль, если хотите можно поставить галочку, что бы клиент его запомнил и не задавал лишних вопросов.

Если все правильно, во *Входящих* должно появиться письмо. А в *Sent*, копия отправленного письма.

Если же у вас что то не работает, тут два варианта:

1. посмотреть логи, а именно /var/log/maillog и попытаться разобраться в ситуации самому.
2. Обратиться к преподавателю и с его помощью попытаться разобраться с ошибками.

ЗАКЛЮЧЕНИЕ.

В результате мы получили простейший почтовый сервер, не лишенный недостатков.

Давайте посмотрим, чего не умеет наш сервер:

- Логины и пароли пользователей при отправке и приеме почты передаются в открытом виде. Надо добавить шифрование TLS или SSL.
- Нет защиты от вирусов. Надо добавить антивирус.
- Нет защиты от спама. Надо добавить антисам фильтр.
- Нет поддержки SPF.

В следующих разделах мы будем добавлять недостающий функционал к нашему почтовому серверу. Но для работы, например spam фильтра, нам потребуется настроить MySQL. А для включения шифрования, создать сертификаты и ключи SSL.

Поэтому сейчас мы немного отвлечемся от почтового сервера и посмотрим на то как быстро настроить MySQL и создать сертификаты и ключи SSL. Разумеется — это будет не полное описания возможностей работы соответствующих систем, но его хватит для решения наших задач.

НАСТРОЙКА ШИФРОВАНИЯ ПРИ ПОДКЛЮЧЕНИИ К ПОЧТОВОМУ СЕРВЕРУ.

На предыдущем курсе, для создания ключей и сертификатов мы пользовались утилитой командной строки *openssl*. Сейчас же я хочу показать вам надстройку, которая облегчает работу с сертификатами — программу *TinyCA2*.

Для запуска программы необходим графический интерфейс. Если у вас под рукой есть машина с установленным Linux — установите программу на нее. Если такой машины нет, можно настроить графику на нашем сервере. Правда придется устанавливать много лишних пакетов и работать при помощи консоли виртуальной машины.

Итак, у кого есть своя машина с Linux, следующий раздел можете смело пропустить.

ВКЛЮЧЕНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА НА СЕРВЕРЕ.

Подключитесь к вашей машине при помощи консоли виртуальной машины.

Поскольку наш сервер работает в виртуальной машине vmware, мы должны установить соответствующий драйвер видеокарточки. Драйвер по зависимостям потянет за собой установка X сервера.

```
yum install xorg-x11-drv-vmware
```

Соглашайтесь на установку пакетов по зависимостям.

Теперь, для удобства работы в графическом интерфейсе в виртуальной машине установим vmwaretools. Для этого в WEB интерфейсе виртуальной машины в закладке *Summury*, разделе *Status*, кликнитесь на ссылке *install vmware tools*.

В результате, виртуальная машина вместо CD-ROM подключит виртуальный диск, на котором будут находиться программы, необходимые для установки. Подключим этот виртуальный CD-ROM.

```
mount /dev/hdc /mnt
```

Установим пакет.

```
rpm –ihv /mnt/VMwareTools-2.0.0-*.rpm
```

После установки, запустим программу конфигурации.

```
vmware-config-tools.pl
```

Дальше просто ответьте на вопросы программы и выберите желаемое разрешение экрана.

Проверьте, запускается ли X сервер.

```
X
```

В результате у вас должен быть черный экран, с передвигающейся по нему мышкой.

Обратите внимание на то, что теперь вам не надо нажимать клавиши Alt+Ctrl для того, что бы переключиться из виртуальной машины на вашу машину.

Что бы закрыть X сервер, нажмите комбинацию клавиш

```
Alt+Ctrl+BackSpace.
```

Теперь установим оконный менеджер. Я предпочитаю KDE, поэтому буду ставить его.

```
yum install kdepim kde-i18n-Russian bitsream-vera-fonts \
liberation-fonts
```

Установка указанных пакетов потянет за собой установку по зависимостям основных пакетов KDE.

После установки, можно смело запускать X-ы:

```
startx
```

УСТАНОВКА ПРОГРАММЫ TINYCA2.

Программу можно найти на сайте http://tinyca.sm-zone.net/.

В дистрибутиве Open SuSE, она находится в основном репозитории. Для RedHat и CantOS, необходимо подключить репрозитарий DAG[1].

При включенном репрозитарии DAG, установка программы происходит стандартный образом.

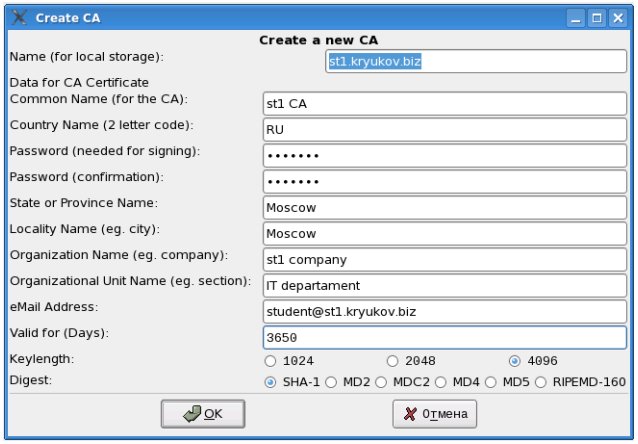
```
yum install tinyca2
```

ИСПОЛЬЗОВАНИЕ ПРОГРАММЫ.

В графической оболочке запустите программу.

```
tinyca2
```

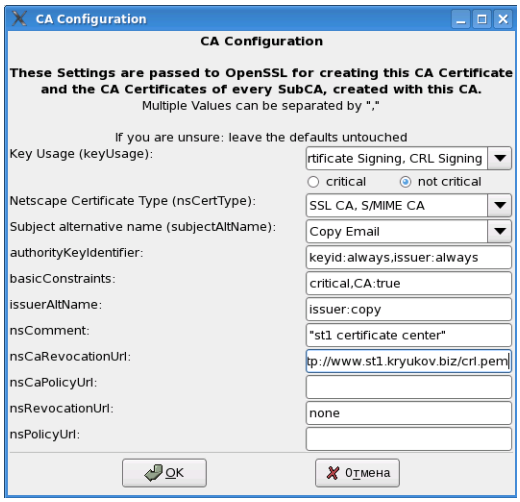
При первом запуске программа поспросит вас создать центр сертификации.



Поле *Name* предназначено для имени вашего центра сертификации. Дело в том, что программа может создать несколько центров сертификации.

[1] Как подключать этот репозиторий, было рассказано на курсе «Настройка Linux сервера, чать 1».

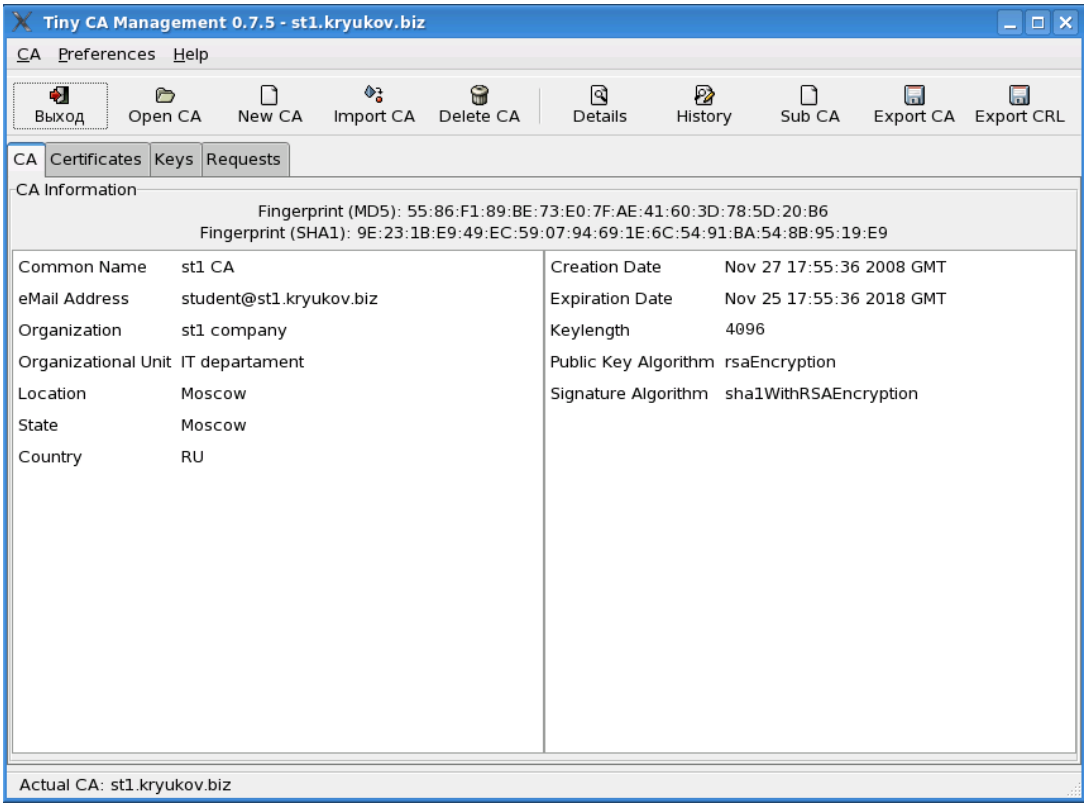
Заполните все поля формы и нажмите *OK*.



В следующем окне вас попросят ввести параметры центра сертификации. Я оставил все поля по умолчанию. Только изменил значение поля *nsComment* и добавил URL в поле, *nsCaRevocationUrl*. В одном из следующих курсов мы настроим WEB сервер, и через него будем распространять список отозванных сертификатов.

Нажмите кнопку *OK*.

После первоначальных настроек, окно программы выглядит так.

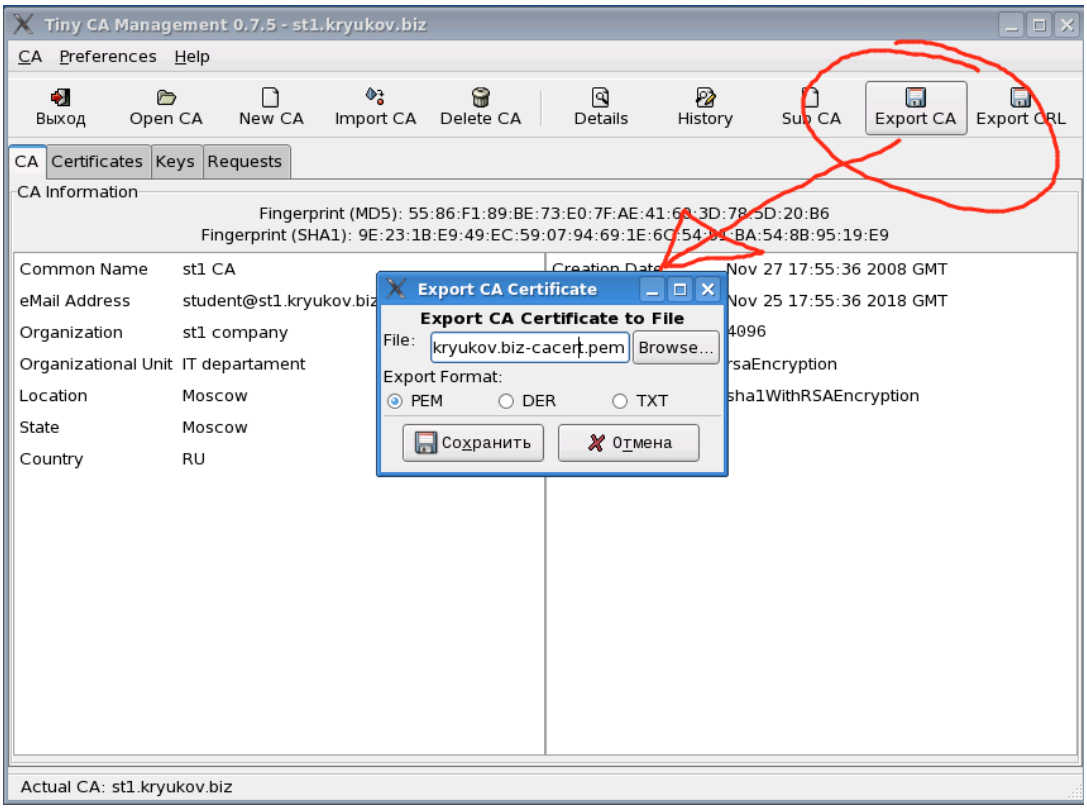


Все файлы, которые необходимы для работы программы будут находиться в вашей домашней директории, в директории *~/.TinyCA*.

При помощи программы вы будете генерировать сертификаты, отзывать сертификаты, формировать списки отзыва сертификатов и т.п. Но полученные файлы, вам придется самостоятельно переносить в нужные директории.

Повторюсь, программу можно запустить и использовать на любой машине в сети. Но полученные файлы вам придется самостоятельно переносить на соответствующие машины в сети.

В первую очередь экспортируем сертификат, нашего центра сертификации.



Нажимаем кнопку *Export CA*.

Выбираем имя и месторасположение сертификата.

Выбираем формат *PEM* или *DER*.

Нажимаем кнопку *Сохранить*.

Этот сертификат мы должны скопировать на все машины и во все программы, которые будут пользоваться сертификатами, выписанными нашим центром сертификации. Для удобства распространения, его можно поместить на WEB сервер нашей компании.

Он потребуется нам и на нашем сервере, поэтому мы его скопируем в директорию */etc/pki/CA*.

```
cp st1.kryukov.biz-cacert.pem /etc/pki/CA
```

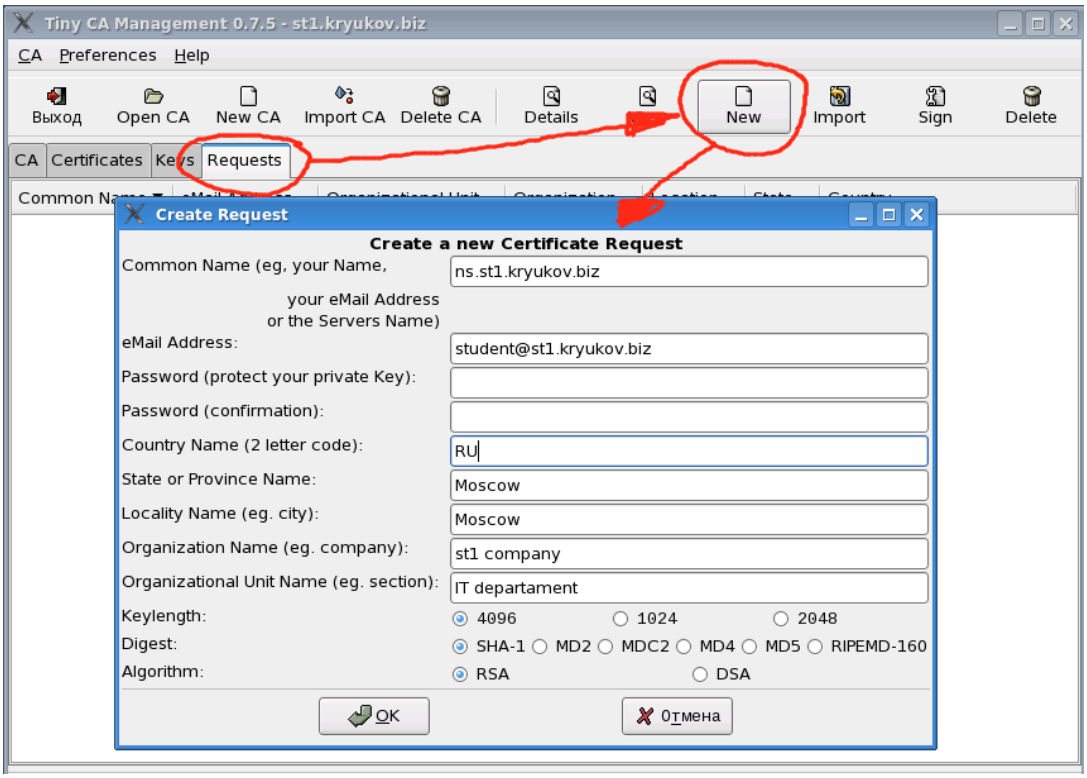
И сделаем так, что бы файл сертификата был доступен на чтение всем пользователям системы.

```
chmod +x /etc/pki/CA
chmod a+r /etc/pki/CA/st1.kryukov.biz-cacert.pem
```

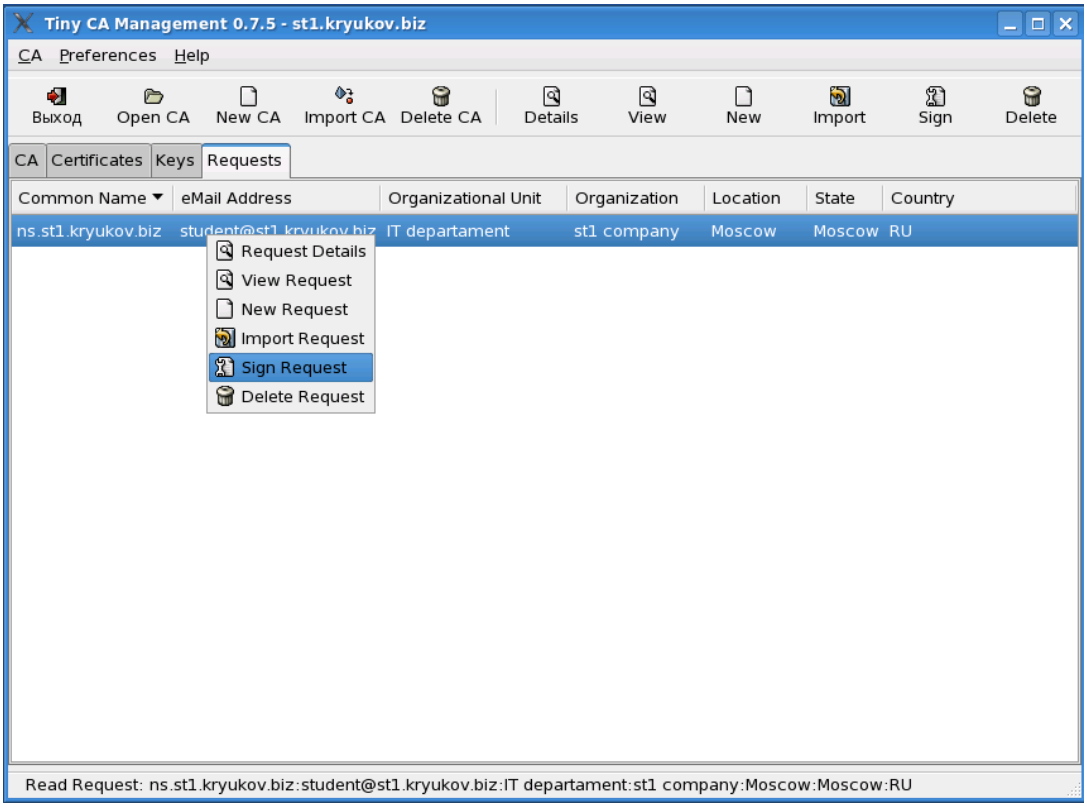
СОЗДАНИЕ КЛЮЧА И СЕРТИФИКАТА ДЛЯ ПОЧТОВОГО СЕРВЕРА.

Теперь создадим ключ и сертификат для нашего почтового сервера. Для этого перейдем в раздел *Requests*.

Нажимаем кнопку *New*, для того, что бы создать новый запрос. И заполняем появившуюся форму.

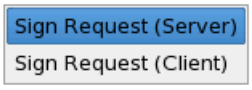


В поле *Common Name* обязательно пишите имя вашего почтового сервера. Если там будет что либо другое, почтовые клиенты будут ругаться на несоответствие имен, и предупреждать вас о попытке обмана.



В списке выберите созданный запрос. Нажмите на нем правую кнопку мыши и выберите — Sign Request (подписать запрос).

Затем вас попросят выбрать, для кого мы создаем сертификат, для клиента или сервера.



Выбирайте сервер.

Теперь вас попросят ввести пароль на ключ центра сертификации и указать срок жизни генерируемого ключа и сертификата.

365 дней — это один год. Вполне нормальный срок для сертификата.

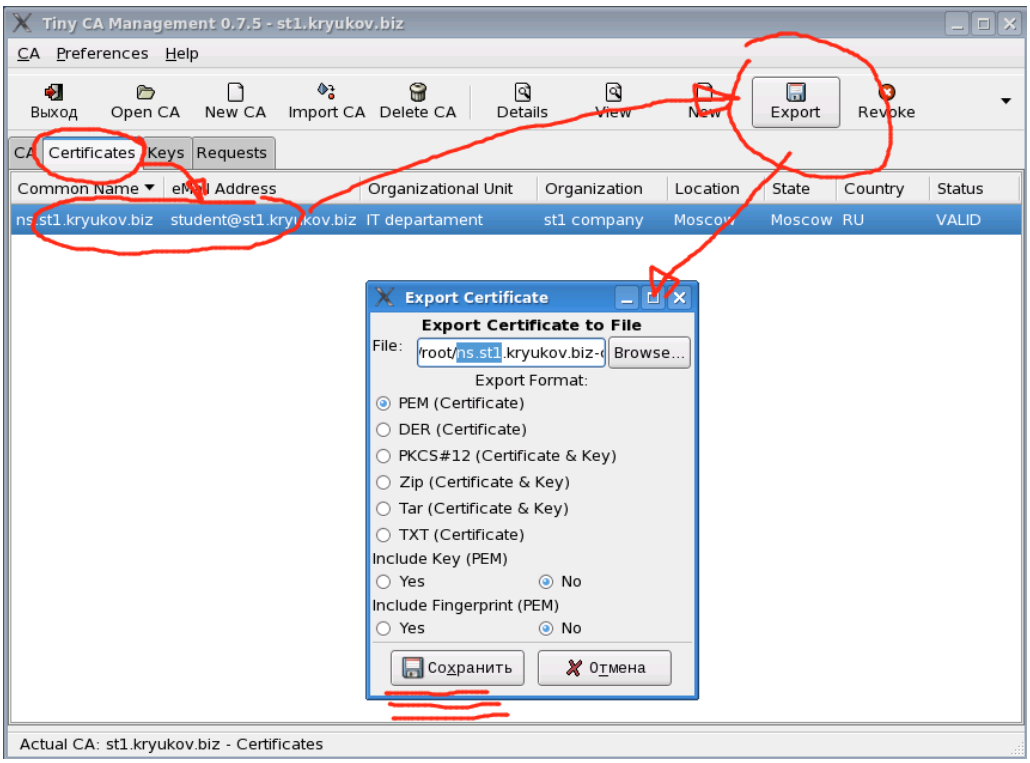
Нажимайте кнопку *OK*.

Таким образом, мы подпишем запрос ключом нашего центра сертификации и сформируем приватный ключ и сертификат для почтового сервера. В закладках *Sertificates* и *Keys* появятся, соответственно, сертификат и ключ.

Нам осталось экспортировать их в файлы и подставить в почтовый сервер.

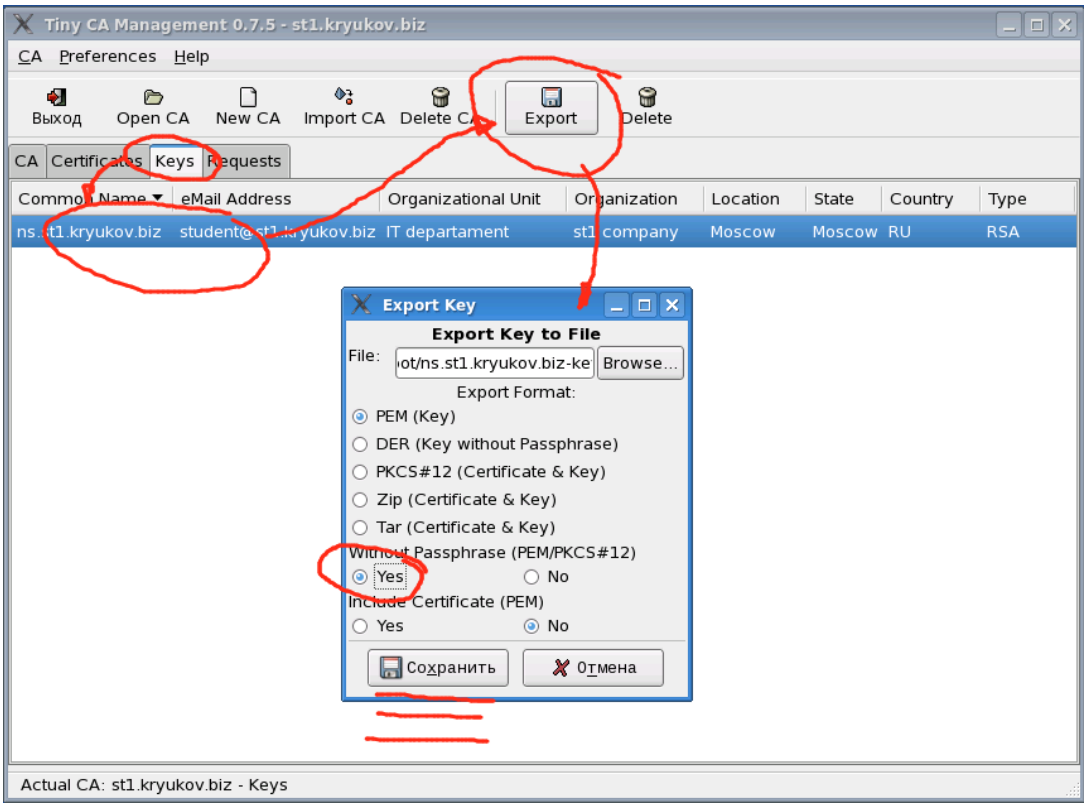
ЭКСПОРТИРОВАНИЕ СЕРТИФИКАТА И КЛЮЧА.

Выбираем закладку *Certificates*. В списке выбираем сертификат, который мы хотим экспортировать в файл. Нажимаем кнопку *Export*.



В появившемся окне изменяем имя файла на *ns.st1.kryukov.biz-cert.pem* и нажимаем кнопку *Сохранить*.

Аналогичные действия производим и с приватным ключом.



Обратите внимание на то, что мы не сохраняем пароль в файле ключа. Это сделано специально для того, что бы не вводить пароль при каждом запуске почтового сервера.

Полученные файлы скопируем в нужные нам директории и установим соответствующие права доступа.

```
# cp ns.st1.kryukov.biz-key.pem /etc/pki/tls/private
# cp ns.st1.kryukov.biz-cert.pem /etc/pki/tls/certs
# chgrp mail /etc/pki/tls/certs/ns.st1.kryukov.biz-cert.pem
# chmod g+r /etc/pki/tls/certs/ns.st1.kryukov.biz-cert.pem
# chgrp mail /etc/pki/tls/certs/ns.st1.kryukov.biz-cert.pem
# chmod g+r /etc/pki/tls/certs/ns.st1.kryukov.biz-cert.pem
```

Мы передали файлы ключа и сертификата группе *mail* потому, что IMAP сервер работает с правами этой группы, и он должен иметь доступ на чтение к этим файлам.

НАСТРОЙКА SENDMAIL.

Теперь нам необходимо настроить sendmail, что бы он начал использовать шифрование при подключении клиентов. Для этого программе необходимо указать файлы с ключами и сертификатами.

Переходим в директорию */etc/mail* и открываем на редактирование файл *sendmail.mc*.

Добавляем следующие строки:

```
define(`confCACERT_PATH', `/etc/pki/tls/certs')dnl
define(`confCACERT', `/etc/pki/CA/st1.kryukov.biz-cacert.pem')dnl
```

```
define(`confSERVER_CERT', `/etc/pki/tls/certs/ns.st1.kryukov.biz-cert.pem')dnl
define(`confSERVER_KEY', `/etc/pki/tls/private/ns.st1.kryukov.biz-key.pem')dnl
```

После перезагрузки сервера, при подключении на 25 порт, у нас появится возможность шифрования трафика.

Сейчас рекомендуется еще открывать специальный порт *smtps* (465/tcp) для подключения с шифрованием по ssl. Для этого потребуется добавить следующие параметры:

```
DAEMON_OPTIONS(`Port=smtp, Name=MTA')dnl
DAEMON_OPTIONS(`Port=smtps, Name=TLSMTA, M=s')dnl
```

Сохраняем файл и пропускаем его через препроцессор m4.

```
m4 sendmail.mc > sendmail.cf
```

Рестартуем почтовый сервер.

```
# service sendmail restart
Останавливается sm-client:                                 [  OK  ]
Останавливается sendmail:                                  [  OK  ]
Запускается sendmail:                                      [  OK  ]
Запускается sm-client:                                     [  OK  ]
[root@server mail]#
```

Смотрим конец файла журнальной регистрации на наличие/отсутствие ошибок при запуске сервера.

```
tail /var/log/maillog
```

Контролируем, были ли открыты соответствующие порты.

```
# netstat -nlp | grep sendmail
tcp        0      0 0.0.0.0:465                 0.0.0.0:*
LISTEN      19628/sendmail: acc
tcp        0      0 0.0.0.0:25                  0.0.0.0:*
LISTEN      19628/sendmail: acc
#
```

Теперь откроем в firewall доступ к серверу по 465 порту. Открываем на редактирование файл *~/bin/rc.fw*. И изменяем правило:

```
$IPT -A INPUT -p tcp -m multiport --dports 25,110,143 -j ACCEPT
```

На

```
$IPT -A INPUT -p tcp -m multiport --dports 25,110,143,465 -j ACCEPT
```

Сохраняем файл. И инициализируем firewall.

```
# rc.fw init
```

НАСТРОЙКА IMAP СЕРВЕРА.

В файле */etc/imapd.conf* добавляем следующие строки.

```
tls_cert_file: /etc/pki/tls/certs/ns.st1.kryukov.biz-cert.pem
tls_key_file: /etc/pki/tls/private/ns.st1.kryukov.biz-key.pem
tls_ca_file: /etc/pki/CA/st1.kryukov.biz-cacert.pem
```

Сохраняем файл.

Открываем на редактирование файл */etc/cyrus.conf*. Убираем комментарии со следующих строк.

```
imaps           cmd="imapd -s" listen="imaps" prefork=1
pop3s           cmd="pop3d -s" listen="pop3s" prefork=1
```

Закроем POP3.

```
#pop3           cmd="pop3d" listen="pop3" prefork=3
```

Сохраняем файл. Перезапускаем сервер.

```
# /etc/init.d/cyrus-imapd restart
```

Контролируем файл журнальной регистрации.

```
# tail /var/log/maillog
```

Смотрим, какие порты открыты на прослушивание.

```
# netstat -nlp | grep cyrus
tcp        0      0 0.0.0.0:993                 0.0.0.0:*
LISTEN      20984/cyrus-master
tcp        0      0 0.0.0.0:995                 0.0.0.0:*
LISTEN      20984/cyrus-master
tcp        0      0 :::993                      :::*
LISTEN      20984/cyrus-master
tcp        0      0 :::995                      :::*
LISTEN      20984/cyrus-master
unix  2      [ ACC ]     STREAM     LISTENING     77312  20984/cyrus-master
/var/lib/imap/socket/lmtp
#
```

Меняем правила в firewall. Открываем на редактирование файл *~/bin/rc.fw*. И изменяем правило:

```
$IPT -A INPUT -p tcp -m multiport --dports 25,110,143,465 -j ACCEPT
```

На

```
$IPT -A INPUT -p tcp -m multiport --dports 25,465,993,995 -j ACCEPT
```

Сохраняем файл. И инициализируем firewall.

```
# rc.fw init
```

НАСТРОЙКА ПОЧТОВОГО КЛИЕНТА.

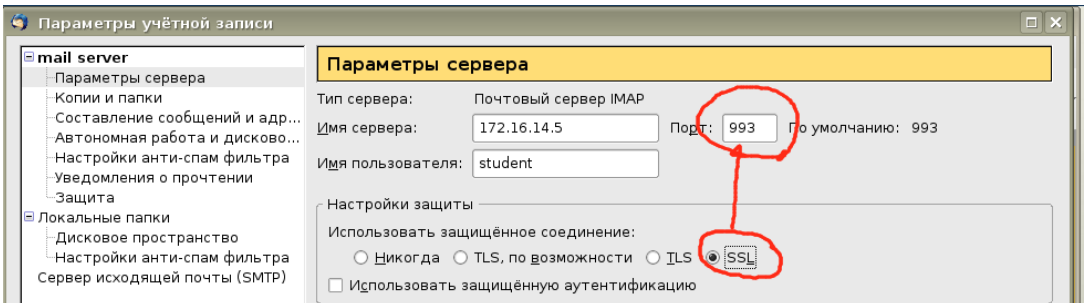
Запустите виртуальную машину, на которой работает почтовый клиент.

Запустите программу thunderbird.

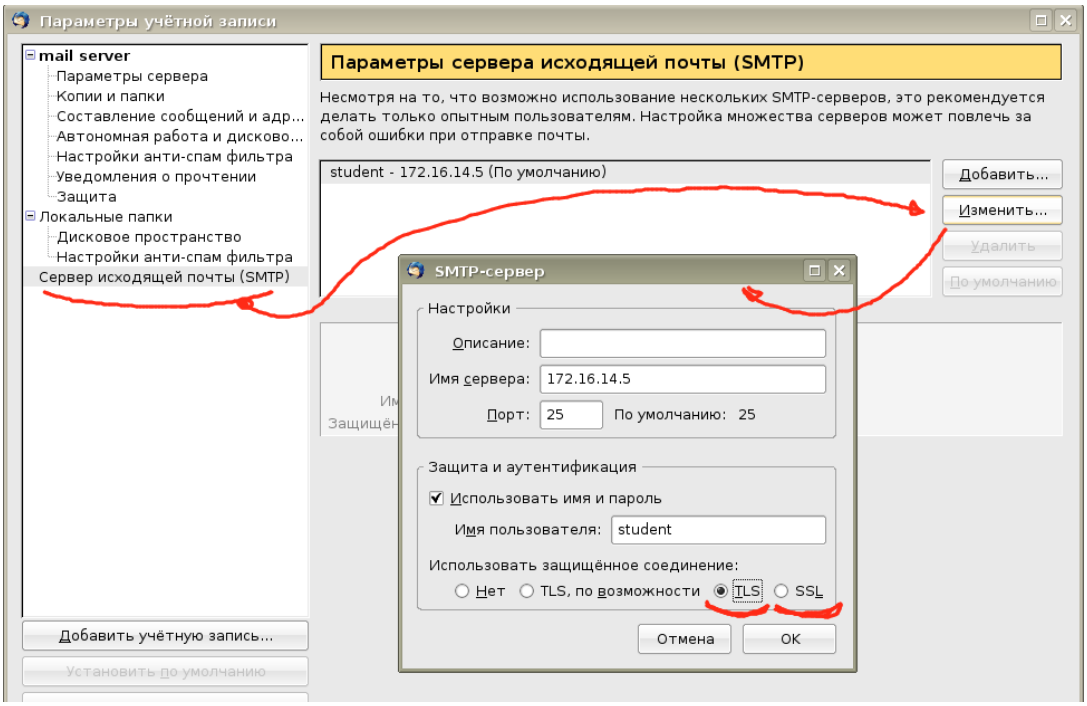
Перейдите в настройки учетной записи.

Выберите *Параметры сервера*.

Установите галочку *SSL*.



Выберите раздел *Сервер исходящей почты*. Нажмите кнопку *Изменить*. И выберите *TLS*, если собираетесь использовать 25-й порт. Или *SSL*, если собираетесь использовать 465-й порт.



Нажмите кнопку *OK*. И еще раз *OK*.

Попробуйте принять почту с сервера.

Если у вас возникли проблемы с приемом почты, смотрите, что IMAP сервер пишет в логах (*/var/log/maillog*). Скорее всего, он не может открыть файлы с ключами и сертификатами. Выставьте правильные права доступа. Или вы ошиблись в имени(ах) файла(ов).

ПОДКЛЮЧЕНИЕ АНТИВИРУСА.

Существует большое количество платных антивирусов для Linux. Но я предпочитаю пользоваться бесплатным антивирусом СdamAV. Очень удачный проект.

В этой главе мы рассмотрим, как подключить ClamAV к sendmail, что бы он проверял всю проходящую почту.

УСТАНОВКА АНТИВИРУСА.

ClamAV не входит в стандартную поставку дистрибутива CentOS, но его можно найти в репрозитории DAG.

Поскольку этот репрозиторий у нас уже установлен, достаточно установить готовые пакеты при помощи yum.

```
# yum install clamav
```

Кроме самого антивируса, потребуется установить так называемый мильтер, при помощи которого sendmail будет обращаться к антивирусу.

```
# yum install clamav-milter
```

Все пакеты были установлены, можно переходить к настройке антивруса.

Делаем так, что бы антивирус запускался при старте системы.

```
# chkconfig clamd on
```

Проверяем.

```
# chkconfig --list clamd
clamd           0:выкл  1:выкл  2:вкл   3:вкл   4:вкл   5:вкл   6:выкл
#
```

НАСТРОЙКА CLAMAV.

Конфигурационный файл ClamAV — /etc/clamd.conf

Все параметры в конфигурационном файле хорошо описаны, поэтому приведу готовый файл, из которого удалены комментарии.

```
LogFile /var/log/clamav/clamd.log
LogFileMaxSize 0
LogTime yes
LogSyslog yes
PidFile /var/run/clamav/clamd.pid
TemporaryDirectory /var/tmp
DatabaseDirectory /var/clamav
LocalSocket /tmp/clamd.socket
FixStaleSocket yes
TCPSocket 3310
TCPAddr 127.0.0.1
MaxConnectionQueueLength 30
MaxThreads 50
ReadTimeout 300
```

```
User clamav
AllowSupplementaryGroups yes
ScanPE yes
ScanELF yes
DetectBrokenExecutables yes
ScanOLE2 yes
ScanPDF yes
ScanMail yes
ScanArchive yes
ArchiveBlockEncrypted no
```

Обратите внимание на параметр

```
LocalSocket /tmp/clamd.socket
```

Он определяет файл типа сокет, при помощи которого мы будем посылать запросы антивирусу.

Запускаем антивирус.

```
# service clamd start
```

Проверяем файлы журнальной регистрации.

```
# tail /var/log/clamav/clamd.log
# tail /var/log/clamav/freshclam.log
```

Последний файл относиться к программе, обновляющей базу данных антивируса — freshclam. Программа обновления автоматически запускается раз в день при помощи системы CRON. *Смотрите содержимое директории /etc/cron.daily*. Если вам потребуется более частое обновление антивирусной базы данных, вам придется самостоятельно добавить задание в CRON.

ПОДКЛЮЧЕНИЕ АНТИВИРУСА К SENDMAIL.

В sendmail есть замечательный механизм, позволяющий передавать проходящую через него почту на обработку сторонним программам — входной (input) и выходной (output) фильтры.

Мы будем использовать макрос INPUT_MAIL_FILTER.

В файл с макросами /etc/mail/sendmail.mc добавим следующую строку:

```
INPUT_MAIL_FILTER
(`clmilter',`S=local:/var/clamav/clmilter.socket,F=,T=S:4m;R:4m')dnl
```

ЭТОТ ПАРАМЕТР ДОЛЖЕН БЫТЬ ЗАПИСАН ОДНОЙ СТРОКОЙ!

И создадим конфигурационный файл sendmail.

```
# cd /etc/mail
# m4 sendmail.mc > sendmail.cf
```

Сам sendmail пока не перезапускайте. Нам еще надо запустить программу мильтер. Задача этой программы получать информацию от входного фильтра sendmail и передавать ее на проверку антивирусу.

Саму программу мы уже установили, пакет называется clamav-milter. Прежде чем запускать sendmail необходимо запустить мильтер.

Сначала убедимся, что в конфигурации sendmail мы указали правильный файл типа socket, через который будут передавать информацию sendmail и мильтер.

```
]# cat /etc/sysconfig/clamav-milter
### Simple config file for clamav-milter, you should
### read the documentation and tweak it as you wish.
CLAMAV_FLAGS="
    --config-file=/etc/clamd.conf
    --force-scan
    --local
    --max-children=10
    --noreject
    --outgoing
    --quiet
"
SOCKET_ADDRESS="local:/var/clamav/clmilter.socket"
```

Теперь запустим мильтер и сделаем так, что бы программа стартовала при каждом запуске компьютера.

```
# service  clamav-milter start
# chkconfig clamav-milter on
```

Если при запуске программы выдается предупреждение о возможных проблемах, связанных с установленной локалью, не обращайте на это внимание.

Теперь перезапустим sendmail.

```
# service sendmail restart
```

А сейчас мы на некоторое время отвлечемся от настройки почтового сервера и займемся вопросами установки на настройки базы данных MySQL.

MYSQL.

Почему настройка MySQL попала на курс посвященный почтовому серверу, а не была вынесена в отдельный курс?

- Во-первых, этот первый случай, когда нам потребуется наличие базы данных. Она необходима для работы антиспам фильтра.
- Во-вторых, я не планировал делать всеобъемлющий курс по MySQL. Я не настолько хорошо знаю его с точки зрения оптимизации таблиц, запросов, индексов. Максиму про что я могу рассказать — это ежедневные административные задачи: запуск, добавление баз данных и пользователей, вопросы бекапа.

Вообщем, сейчас мы рассмотрим вопрос связанные с базовой администрацией MySQL.

ПЕРВЫЙ ЗАПУСК.

Устанавливать базу данных мы будем традиционным способом, при помощи yum.

Сначала посмотрим, что было поставлено во время установки дистрибутива.

```
# rpm -qa '*mysql*'
perl-DBD-mysql-4.008-1.el5.rf
mysql-connector-odbc-3.51.12-2.2
mysql-5.0.45-7.el5
mysql-server-5.0.45-7.el5
libdbi-dbd-mysql-0.8.1a-1.2.2
#
```

Итак, нам не хватает двух пакетов: mysql-devel и php-mysql. Установим их.

```
# yum install mysql-devel php-mysql
```

Зачем нам потребуется php-mysql? Скоро увидите ☺.

Теперь необходимо запустить программу. Это тоже делается стандартно, при помощи стартовых скриптов.

```
# /etc/rc.d/init.d/mysqld start
```

Делаем так, чтобы сервер запускался при старте системы.

```
# chkconfig mysqld on
```

УСТАНОВКА ПАРОЛЯ ПОЛЬЗОВАТЕЛЯ ROOT.

При первом запуске выдается предупреждение — необходимо создать пароль для пользователя root. И предлагается вариант как это можно сделать:

```
/usr/bin/mysqladmin -u root password 'new-password'
/usr/bin/mysqladmin -u root -h server.st1.kryukov.biz password 'new-password'
```

Прежде, чем мы создадим пароль, поговорим о пользователе root в MySQL.

Во-первых, надо запомнить, что у MySQL свой набор пользователей. Список пользователей MySQL храниться во внутренней базе данных, и он никак не связан с пользователями Linux.

После первого запуска в базе пользователей находится только пользователь root, у которого установлены все права на управление базой, но не установлен пароль. Поменять пароль поль-

зователя при первом запуске программы возможно несколькими способами, один из них — использовать программу mysqladmin.

```
# mysqladmin -u root password 'newpassword'
```

Вместо newpassord вы напишите свой собственный пароль.

Проверим, можете ли вы подключаться к базе данных. Для этого используем программу-клиент mysql.

```
# mysql -u root –p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

После параметра –u пишем пользователя, с правами которого мы хотим подключиться к базе данных. Параметр –p заставит программу спросить вас пароль пользователя.

НИКОГДА НЕ ПЕРЕДАВАЙТЕ ПАРОЛЬ В КАЧЕСТВЕ АРГУМЕНТА КОМАНДНОЙ СТРОКИ!

В клиенте можно вводить различные SQL запросы. Посмотрим содержимое таблицы user.

Сначала выберем базу данных, с которой мы будем работать.

```
mysql> use mysql;
```

Теперь сформируем запрос.

```
mysql> select host,user,password from user;
+-----------------------+------+------------------+
| host                  | user | password         |
+-----------------------+------+------------------+
| localhost             | root | 4123060a330e24ae |
| server.st1.kryukov.biz | root |                  |
| 127.0.0.1             | root |                  |
+-----------------------+------+------------------+
3 rows in set (0.00 sec)
```

По результатам запроса видно, что пароль установлен только для пользователя root, подключающегося с машины localhost.

Установим пароль для остальных подключений. Заодно научимся менять пароль другим способом.

```
mysql> SET PASSWORD for 'root'@'127.0.0.1' = PASSWORD('newpassword');
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Проверим.

```
mysql> select host,user,password from user;
+-----------------------+------+------------------+
| host                  | user | password         |
+-----------------------+------+------------------+
| localhost             | root | 4123060a330e24ae |
| server.st1.kryukov.biz | root |                  |
| 127.0.0.1             | root | 4123060a330e24ae |
+-----------------------+------+------------------+
3 rows in set (0.00 sec)


mysql>
```

Аналогичным образом добавим пароль для входа пользователя с третьего хоста.

```
mysql> SET PASSWORD for 'root'@'server.st1.kryukov.biz' = PASS-
WORD('newpassword');
Query OK, 0 rows affected (0.00 sec)


mysql> select host,user,password from user;
+-----------------------+------+------------------+
| host                  | user | password         |
+-----------------------+------+------------------+
| localhost             | root | 4123060a330e24ae |
| server.st1.kryukov.biz | root | 4123060a330e24ae |
| 127.0.0.1             | root | 4123060a330e24ae |
+-----------------------+------+------------------+
3 rows in set (0.00 sec)


mysql>
```

Выйдем из программы.

```
mysql> quit
Bye
[root@server ~]#
```

ВОССТАНОВЛЕНИЕ ПАРОЛЯ ПОЛЬЗОВАТЕЛЯ ROOT.

Что делать если вы забыли пароль пользователя root в MySQL? Ведь вы не сможете подключаться к серверу и управлять им.

Вы должны выключить сервер MySQL при помощи стартового скрипта.

```
# service mysqld stop
```

Затем запустить сервер по новой, но при запуске следует передать параметр отменяющий проверку разрешений: *--skip-grant-tables*.

Для этого вам придётся внести некоторые изменения в стартовый скрипт.

```
# vim /etc/rc.d/init.d/mysqld
```

Ищем в функции start следующие строки:

```
/usr/bin/mysqld_safe   --datadir="$datadir" --socket="$socketfile" \
                --log-error="$errlogfile" --pid-file="$mypidfile" \
                >/dev/null 2>&1 &
```

Добавляем параметр. В результате строки будут выглядеть так:

```
/usr/bin/mysqld_safe   --datadir="$datadir" --socket="$socketfile" \
                --log-error="$errlogfile" --pid-file="$mypidfile" \
                --skip-grant-tables >/dev/null 2>&1 &
```

Сохраняем файл и запускаем сервер.

```
# service mysqld start
```

Теперь мы можем подключиться пользователем root без указания пароля.

```
# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Дальше меняем пароль пользователю root. Но на этот раз нам придётся вносить изменения непосредственно в таблицу user. Соответственно сейчас мы рассмотрим еще один способ смены пароля пользователя.

Переключаемся на базу данных mysql.

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> UPDATE user SET Password=PASSWORD('newpassword') WHERE User='root';
Query OK, 0 rows affected (0.04 sec)
Rows matched: 3  Changed: 0  Warnings: 0

mysql>
```

Таким образом, мы поменяли пароль пользователю root, откуда бы он не подключался. Поскольку сейчас сервер не использует таблицу ограничения доступа, мы должны ее включить явным образом.

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Теперь сервер может работать дальше в обычном режиме.

ТОЛЬКО НЕ ЗАБУДЬТЕ УДАЛИТЬ ПАРАМЕТР ИЗ СТАРТОВОГО СКРИПТА! ИНАЧЕ СЕРВЕР ВСЕГДА БУДЕТ ЗАПУСКАТЬСЯ БЕЗ РАЗРЕШЕНИЯ ПОЛНОМОЧИЙ.

СОЗДАНИЕ И УДАЛЕНИЕ БАЗ ДАННЫХ И ПОЛЬЗОВАТЕЛЕЙ.

В дальнейшем вам достаточно часто придется самостоятельно создавать базы данных, которые будут использоваться различными приложениями. Давайте рассмотрим, как это можно делать.

СОЗДАНИЕ БАЗЫ ДАННЫХ.

Для работы мы будет использовать программу mysql. Подключитесь к серверу с правами пользователя root. Ведь именно он имеет все привилегии для создания баз данных и пользователей.

```
# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Для создания базы данным мы будем пользоваться оператором CREATE DATABASE.

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
```

В простейшем варианте достаточно указать только имя создаваемой базы данных.

```
mysql> CREATE DATABASE sample;
Query OK, 1 row affected (0.00 sec)

mysql>
```

В нашем примере была создана база sample.

Если посмотреть содержимое директории, где хранятся базы данных MySQL, вы увидите, что была создана директория *sample*.

```
# ls /var/lib/mysql/
ibdata1  ib_logfile0  ib_logfile1  mysql  mysql.sock  sample  test
# ls /var/lib/mysql/sample/
db.opt
#
```

СОЗДАНИЕ ПОЛЬЗОВАТЕЛЯ.

Следующий шаг — создание пользователя или пользователей, которые будут работать с этой базой данных, и выдать им соответствующие полномочия.

Для создания учетной записи в MySQL можно использовать операторы *CREATE USER* или *GRANT*. Мы будем пользоваться первым оператором, поскольку он не только создаёт пользователя, но и одновременно назначает ему права доступа к базе данных.

Например, чтобы создать пользователя user и дать ему все права на бузу данных sample, следует выполнить такой оператор:

```
mysql> GRANT ALL PRIVILEGES ON sample.* TO 'user'@'localhost' IDENTIFIED BY
'newpassword';
Query OK, 0 rows affected (0.03 sec)

mysql>
```

Теперь подключимся к базе *sample* и например проверим, сможет ли пользователь *user* создать там таблицу.

```
# mysql -u user -p sample
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE TABLE t (  id INT(11) default NULL auto_increment,\
  s char(60) default NULL, \
  PRIMARY KEY (id) );
Query OK, 0 rows affected (0.07 sec)

mysql> DROP TABLE t;
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
#
```

Мы попытались создать таблицу *t*, это нам удалось. Затем мы таблицу удалили и вышли из программы. Значит, пользователь получил все привилегии на базу данных.

По поводу ограничений, которые можно устанавливать при помощи оператора GRANT можно почитать в документации в Интернет: http://dev.mysql.com/doc/refman/5.1/en/grant.html

Удалять привилегии можно при помощи оператора *REVOKE*.

Если вы хотите создать пользователя с правами администратора, то в конце оператора *GRANT* необходимо добавить *WITH GRANT OPTION*. Но лучше не рисковать и оставить право на изменение привилегий (вызов оператора *GRANT*), только у одного пользователя.

УДАЛЕНИЕ ПОЛЬЗОВАТЕЛЯ.

Удаление пользователя возможно при помощи оператора *DROP USER*.

```
mysql> use mysql
```

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select host,user,password from user;
+-----------------------+------+------------------+
| host                  | user | password         |
+-----------------------+------+------------------+
| localhost             | root | 4123060a330e24ae |
| server.st1.kryukov.biz | root | 4123060a330e24ae |
| 127.0.0.1             | root | 4123060a330e24ae |
| localhost             | user | 3a9eb1070a0130ca |
+-----------------------+------+------------------+
4 rows in set (0.02 sec)

mysql> DROP USER 'user'@'localhost';
Query OK, 0 rows affected (0.02 sec)

mysql> select host,user,password from user;
+-----------------------+------+------------------+
| host                  | user | password         |
+-----------------------+------+------------------+
| localhost             | root | 4123060a330e24ae |
| server.st1.kryukov.biz | root | 4123060a330e24ae |
| 127.0.0.1             | root | 4123060a330e24ae |
+-----------------------+------+------------------+
3 rows in set (0.00 sec)

mysql>
```

УДАЛЕНИЕ БАЗЫ ДАННЫХ.

Удалять базу данных мы будем при помощи оператора *DROP DATABASE*.

```
mysql> DROP DATABASE sample;
Query OK, 0 rows affected (0.08 sec)

mysql>
```

БЕКАП БАЗЫ ДАННЫХ.

Сразу хочу предупредить — мы не будет рассматривать все возможности бекапа, доступные в MySQL.

В принципе, для бекапа базы данных можно скопировать директорию с самой базой. Напомню, что все базы данных представляют из себя файлы, которые находятся в директории */var/lib/mysql*. Но это не самый удачный способ.

Для создания резервной копии базы данных мы будем использовать утилиту *mysqldump*.

Давайте попробуем как это делается. Сначала создадим пользователя и базу данных.

```
# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE DATABASE sample;
Query OK, 1 row affected (0.04 sec)

mysql> GRANT ALL PRIVILEGES ON sample.* TO 'user'@'localhost' IDENTIFIED BY
'newpassword';
Query OK, 0 rows affected (0.35 sec)

mysql> quit
```

В базе создадим одну таблицу и внесем некоторые данные.

```
# mysql -u user -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use sample
Database changed
mysql> CREATE TABLE t (  id INT(11) default NULL auto_increment,  s
char(60) default NULL,  PRIMARY KEY (id) );
Query OK, 0 rows affected (0.05 sec)

mysql> mysql> INSERT INTO t (s) VALUE ('test string');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO t (s) VALUE ('test string 2');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO t (s) VALUE ('test string 3');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM t;
```

```
+----+---------------+
| id | s             |
+----+---------------+
|  1 | test string   |
|  2 | test string 2 |
|  3 | test string 3 |
+----+---------------+
3 rows in set (0.00 sec)


mysql> quit
```

Теперь сделаем бекап базы данных sample.

```
# mysqldump -u user -p --opt sample > ~/sample-2008-12-03.dump.sql
Enter password:
#
```

Итак, по порядку:

- -u user — указываем пользователя, с правами которого подключаемся к базе данных. Этот пользователь должен иметь соответствующие права доступа к базе, для которой мы делаем бекап.
- -p — заставляет программу спросить пароль пользователя.
- --opt — включает несколько параметров:
  - --add-drop-table — Добавить команду *DROP TABLE* перед каждой командой *CREATE TABLE*. Таким образом, при восстановлении из этого бекапа, старые данные удаляются и каждая таблица создается снова.
  - --add-locks — Добавить команды *LOCK TABLES* перед выполнением и *UNLOCK TABLE* после выполнения каждого дампа таблицы (для ускорения доступа к MySQL).
  - --create-options — Включает все, специфичные для MySQL параметры в оператор *CREATE TABLE*.
  - --disable-keys — Добавляет выражение */*!40000 ALTER TABLE tb_name DISABLE KEYS */;* и */*!40000 ALTER TABLE tb_name ENABLE KEYS */;* в выводе результата. Это ускорит загрузку данных на сервер MySQL, так как индексы создаются после внесения всех данных.
  - --extended-insert — Использовать команду *INSERT* с новым многострочным синтаксисом (повышает компактность и быстродействие операторов ввода).
  - --lock-tables — Этот параметр блокирует все таблицы, на время выполнения бекапа.
  - --quick — Выводить дамп непосредственно на стандартный вывод (stdout) без буферизации запроса.
  - --set-charset — Добавляет SET NAMES.
- sample — имя базы данных, для которой мы делаем бекап.

Результат работы программы — текстовый файл с SQL инструкциями. Используя этот файл, можно восстановить базу данных. Ниже покажу весь получившийся файл, благо он не большой.

```
-- MySQL dump 10.11
--
-- Host: localhost    Database: sample
-- ------------------------------------------------------
-- Server version       5.0.45
```

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOR-
EIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;


--
-- Table structure for table `t`
--

DROP TABLE IF EXISTS `t`;
CREATE TABLE `t` (
  `id` int(11) NOT NULL auto_increment,
  `s` char(60) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;


--
-- Dumping data for table `t`
--

LOCK TABLES `t` WRITE;
/*!40000 ALTER TABLE `t` DISABLE KEYS */;
INSERT INTO `t` VALUES (1,'test string'),(2,'test string 2'),(3,'test
string 3')
;
/*!40000 ALTER TABLE `t` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;


/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

```
-- Dump completed on 2008-12-03 12:44:39
```

Теперь удалим базу данных и посмотрим как ее можно восстановить.

```
# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> DROP DATABASE sample;
Query OK, 1 row affected (0.85 sec)

mysql> quit
Bye
```

Если в результате аварии на сервере был удален пользователь, перед восстановлением базы данных его необходимо создать.

```
# mysql -u user -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE DATABASE sample;
Query OK, 1 row affected (0.00 sec)

mysql> use sample
Database changed
mysql> source ~/sample-2008-12-03.dump.sql
```

Тут будет показан процесс заполнения базы данных.

```
mysql> SELECT * FROM t;
+----+---------------+
| id | s             |
+----+---------------+
|  1 | test string   |
|  2 | test string 2 |
|  3 | test string 3 |
+----+---------------+
3 rows in set (0.00 sec)
```

```
mysql> quit
Bye
```

Как видите, все данные были восстановлены.

Хочу еще раз повториться, что существует много разных способов бекапа MySQL. Представленный вам способ наиболее простой и его вполне хватает для небольших баз данных, например, обслуживающих WEB сайты.

ЗАЩИТА ОТ СПАМА.

Для защиты от спама мы будем использовать несколько технологий.

- Dnsbl сервера. Сервера, содержащие черные списки.
- Антиспам фильтр DSPAM.
- Контроль spf записей.

DNSBL.

Подключение dnsbl в sendmail происходит при помощи специального макроса FEATURE.

```
FEATURE(`dnsbl',`Имя сервера',`Сообщение')dnl
```

В Интернет существует большое количество бесплатных DNSBL серверов. К каким из них подключаться, решать только вам. Вы должны понимать, что не все сервера адекватны.

Ниже я приведу список серверов, услугами которых пользуюсь я.

```
FEATURE(`dnsbl',`dul.dnsbl.sorbs.net',`"554 Rejected " $&{client_addr} "
found in dul.dnsbl.sorbs.net"')dnl


FEATURE(`dnsbl',`smtp.dnsbl.sorbs.net',`"554 Rejected " $&{client_addr} "
found in smtp.dnsbl.sorbs.net"')dnl


FEATURE(`dnsbl',`combined.njabl.org',`Message from $&{client_addr} rejected
- Server Blacklisted by combined.njabl.org')dnl


FEATURE(`dnsbl',`sbl-xbl.spamhaus.org',`Message from $&{client_addr} re-
jected - Server Blacklisted by sbl-xbl.spamhaus.org')dnl
```

В конфигурационном файле */etc/mail/sendmail.mc* можно указать несколько серверов. Sendmail будет обращаться к ним по порядку. Если IP адрес передающего вам почту сервера будет обнаружен хотя бы в одном из них, письмо принято не будет.

Хочу обратить ваше внимание, что ПРОВЕРКА В DNSBL ПРОИСХОДИТ ДО ТОГО, КАК БУДЕТ ПРОВЕРЯТЬСЯ АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЯ.

Предположим, что вы находитесь дома. Интернет вам предоставляет какой либо провайдер, типа домашних сетей. При каждом подключении вам выдается IP адрес, предназначенный для внутренних сетей. Т.е. когда вы выходите в Интернет, у провайдера происходит NAT преобразование и подставляется IP source вашего провайдера.

Скорее всего, его IP адрес уже находится в каком нибудь DNSBL сервере и вы не сможете отправить письмо через свой сервер. Как быть? В этом случае необходимо перенести аутентификацию пользователя до проверки в DNSBL. Если аутентификация пройдет успешно, тогда проверка в DNSBL производиться не будет. Макрос delay_checks предназначен именно для этих целей:

```
FEATURE(`delay_checks')dnl
```

После добавления макросов, создайте конфигурационный файл почтового сервера и перезапустите его.

```
# m4 sendmail.mc > sendmail.cf

# service sendmail restart
```

АНТИСПАМ ФИЛЬТР.

К сожалению DNSBL отсекает не весь спам, который приходит на ваш почтовый сервер. Поэтому придется ставить антиспам фильтр. Существует большое количество таких фильтров, как платных, так и бесплатных. Я предпочитаю использовать DSPAM.

В отличии от широко известного spamassasin, написанного на интерпретируем языка perl, DSPAM написан на C, что значительно увеличивает скорость работы программы и не сильно загружает сервер при большом потоке писем.

Еще одно преимущество программы — каждый пользователь может самостоятельно обучать программу. Т.е. для каждого пользователя будут свои предпочтения, что он считает или не считает спамом. Большинство других фильтров не имеют возможности индивидуальной тренировки.

УСТАНОВКА ПОГРАММЫ.

К сожалению, DSPAM не входит в стандартную поставку дистрибутива, поэтому нам придется его собирать из исходных кодов.

Исходные коды программы можно скачать тут http://dspam.nuclearelephant.com/. Берите самую последнюю версию программы. На момент написания — это была версия 3.8.0.

Скопируйте архив с исходными кодами в домашнюю директорию и распакуйте его.

```
# tar -xzf dspam-3.8.0.tar.gz
```

Перейдите в директорию dspam-3.8.0.

```
# cd dspam-3.8.0
```

Программа собирается традиционным способом, при помощи configure.

```
./configure --sysconfdir=/etc \
  --prefix=/usr/local \
  --with-dspam-home=/var/dspam \
  --with-storage-driver=mysql_drv \
  --with-mysql-includes=/usr/include/mysql \
  --with-mysql-libraries=/usr/lib/mysql \
  --enable-preferences-extension \
  --enable-virtual-users \
  --enable-daemon \
  --disable-trusted-user-security \
  --enable-debug
# make
# make install
```

При запуске *configure* обязательно указывайте базу данных, с которой будет работать DSPAM. Это делается при помощи параметра *--with-storage-driver*. Мы будем использовать MySQL, поэтому значение параметра равно *mysql_drv*.

Параметр *--enable-virtual-users* позволит работать с почтовыми ящиками пользователей нескольких доменов.

Так же нам потребуется http://www.viegasdelima.com/dspam/sendmail+DSPAM+Cyrus.tar.gz.

```
# cd
```

```
# wget http://www.viegasdelima.com/dspam/sendmail+DSPAM+Cyrus.tar.gz
```

Это набор дополнительных файлов, при помощи которым мы будем связывать спам фильтр и IMAP сервер.

Распакуйте архив в домашней директории пользователя.

```
# cd
# tar -xzf sendmail+DSPAM+Cyrus.tar.gz
```

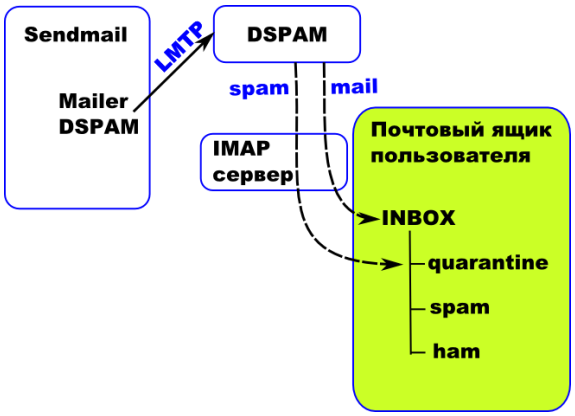
У вас появится еще один архив: *locals.tar.gz*. Распакуйте его.

```
# tar -xzf locals.tar.gz
```

В результате у вас появится директория *~/locals*. Чуть позже, мы воспользуемся файлами из этой директории.

КОНФИГУРАЦИЯ ПРОГРАММЫ.

Сначала давайте разберемся, как будет подключаться спам фильтр к sendmail.



Всю входящую почту, sendmail при помощи специального мейлера dspam, будет передавать программе DSPAM. Поскольку почтовый сервер и антиспам находятся на одной машине, мы будем использовать файл типа socket. Общение будет происходить по протоколу LMTP (Local mail transfer protocol).

DSPAM принимает решение, является ли письмо спамом. Если это нормальное письмо, оно доставляется в папку *INBOX*. Если письмо является спамом, оно доставляется в папку quarantine.

В дальнейшем пользователь сможет обучать спам фильтр. Если пользователь считает, что письмо является спамом, он должен переместить письмо в папку *spam*. Если письмо, находящее в папке *quarantine*, не является спамом, пользователь должен перетащить его в папку *ham*. Как уже говорилось выше, каждый пользователь индивидуально настраивает антиспам фильтр.

Обратите внимание на то, что при такой конфигурации антиспам обрабатывает только входящую почту. Исходящую и транзитную почту он обрабатывать не будет.

ИЗМЕНЕНИЯ В КОНФИГУРАЦИОННОМ ФАЙЛЕ.

Конфигурационный файл программы — */etc/dspam.conf*.

В первую очередь мы должны определить, как будет принимать почту антиспам. В файле найдите блок закомментированных параметров:

```
#DeliveryHost          127.0.0.1
```

```
#DeliveryPort          24
#DeliveryIdent         localhost
#DeliveryProto         LMTP
```

Поскольку почтовый сервер и антиспам работают на одной машине, мы будем передавать данные через файл типа socket по протоколу LMTP. В результат нам придется определить два параметра:

```
DeliveryHost /var/lib/imap/socket/lmtp
DeliveryProto       LMTP
```

Обязательно включите параметр:

```
EnablePlusedDetail      on
```

Он заставляет антиспам указывать папки пользователя через +, что очень удобно, если вы используете логины пользователей с точкой в имени. Например, *a.kryukov*. При включении этого параметра DSPAM будет передавать данные серверу IMAP в следующем виде: *a.kryukov+папка*. По умолчанию, было бы так: *a.krukov.папка*.

Следующий параметр определяет папку, в которую будет помещаться спам.

```
QuarantineMailbox       +quarantine
```

Параметр TrainingMode определяет режим обучения антиспам фильтра. В случае небольших почтовых систем (около 500 почтовых ящиков) можно включить постоянный режим обучения.

```
TrainingMode teft
```

Параметр tb определяет размер буфера на обучение. Например, если его значение равно 5-ти, то письмо будет считаться спамом или не спамом только после 5-ти случаев. Обычно размер буфера ставят равным нулю.

```
Feature tb=0
```

Параметр

```
Preference "spamAction=deliver"
```

Заставляет антиспам фильтр доставлять письмо, помеченное как спам, в почтовый ящик пользователя. Но оно будет доставляться не в INBOX. А в папку, определенную при помощи параметра *QuarantineMailbox*.

```
Preference "signatureLocation=headers"
```

Параметр указывает, куда помещать сигнатуру, специальную метку, по которой антиспам запоминает письмо. Возможны два варианта: непосредственно в тело письма: *message*. Но в этом случае странные цифры увидит пользователь. И *headers* — в заголовке письма. Мы выбираем последний вариант — хранить сигнатуры в заголовке.

```
Preference "showFactors=off"
```

Параметр запрещает писать в заголовке письма информацию о баллах, которое письмо набрало при проверке на спам.

При помощи параметра

```
Preference "spamSubject=SPAM"
```

Мы можем добавлять в *Subject* письма любую последовательность символов. Но мы не будем использовать этот параметр.

Теперь посмотрим раздел, посвященный работе программы с базой данных. Поскольку мы будем использовать MySQL, будем править соответствующий раздел.

Сначала укажем файл, через который мы будем подключаться к базе данных.

```
MySQLServer     /var/lib/mysql/mysql.sock
```

Затем определим пользователя в MySQL, его пароль и имя базы данных.

```
MySQLUser               dspam
MySQLPass               passwordspam
MySQLDb                 dspam
MySQLConnectionCache    20
```

Следующий параметр позволит сохранять в базе UID пользователя, которому принадлежит почтовый ящик. Это очень удобно, если у ящика есть несколько почтовых псевдонимов. Не потребуется обучать антиспам для каждого псевдонима.

```
MySQLUIDInSignature    on
```

Поскольку мы будем запускать DSPAM в режиме демона, придется определить параметры, при помощи которых клиентские процессы сапм фильтра, будут обращаться к серверной части.

Для начала определим параметры сервера.

```
ServerMode auto
ServerPass.Relay1       "passwordforclient"
ServerParameters        "--deliver=innocent,spam"
ServerDomainSocketPath  "/var/dspam/dspam.sock"
```

И клиентской части.

```
ClientHost      /var/dspam/dspam.sock
ClientIdent     " passwordforclient @Relay1"
```

В результате, если убрать все комментарии и пустые строки, конфигурационный файл будет выглядеть следующим образом:

```
Home /var/dspam
StorageDriver /usr/local/lib/libmysql_drv.so
TrustedDeliveryAgent "/usr/bin/procmail"
DeliveryHost /var/lib/imap/socket/lmtp
DeliveryProto       LMTP
EnablePlusedDetail      on
QuarantineMailbox       +quarantine
OnFail error
Trust root
Trust mail
Trust mailnull
Trust smmsp
Trust daemon
TrainingMode teft
TestConditionalTraining on
Feature whitelist
Feature tb=0
Algorithm graham burton
```

```
Tokenizer chain
PValue bcr
WebStats on
Preference "spamAction=deliver"
Preference "signatureLocation=headers"
Preference "showFactors=off"
AllowOverride trainingMode
AllowOverride spamAction spamSubject
AllowOverride statisticalSedation
AllowOverride enableBNR
AllowOverride enableWhitelist
AllowOverride signatureLocation
AllowOverride showFactors
AllowOverride optIn optOut
AllowOverride whitelistThreshold
MySQLServer     /var/lib/mysql/mysql.sock
MySQLUser               dspam
MySQLPass               masterspamnow
MySQLDb                 dspam
MySQLConnectionCache    20
MySQLUIDInSignature    on
HashRecMax              98317
HashAutoExtend          on
HashMaxExtents          0
HashExtentSize          49157
HashPctIncrease 10
HashMaxSeek             10
HashConnectionCache     10
Notifications   off
PurgeSignatures 14          # Stale signatures
PurgeNeutral    90          # Tokens with neutralish probabilities
PurgeUnused     90          # Unused tokens
PurgeHapaxes    30          # Tokens with less than 5 hits (hapaxes)
PurgeHits1S     15          # Tokens with only 1 spam hit
PurgeHits1I     15          # Tokens with only 1 innocent hit
LocalMX 127.0.0.1
SystemLog on
UserLog   on
Opt out
ServerMode auto
ServerPass.Relay1       "passwordforclient"
ServerParameters        "--deliver=innocent,spam"
ServerDomainSocketPath  "/var/dspam/dspam.sock"
```

```
ClientHost      /var/dspam/dspam.sock
ClientIdent     "passwordforclient@Relay1"
ProcessorURLContext on
ProcessorBias on
```

СОЗДАНИЕ БАЗЫ В MYSQL.

Для работы спам фильтра необходимо создать базу данных в MySQL.

Переходим в директории с исходными кодами программы. Затем переходим в директорию *src/tools.mysql_drv*.

```
# cd ~/dspam-3.8.0/src/tools.mysql_drv
```

Скопируем необходимые файлы в директорию ~/locals.

```
# cp ./{virtual_users.sql,mysql_objects-4.1.sql} ~/locals
```

И перейдем в эту директорию.

```
# cd ~/locals
# ls
build   dspam.m4  imap-user   mysql_setup.sql  virtual_users.sql
dspam.cron  dspam.rc  mysql_objects-4.1.sql  traindspam
#
```

Отредактируем файл *mysql_setup.sql*.

В строке

```
GRANT ALL PRIVILEGES ON dspam.* TO dspam@localhost IDENTIFIED BY 'isabel';
```

Поменяем пароль пользователя *dspam* с *isabel* на тот, который вы написали в конфигурационном файле *dspam.conf*.

Сохраните файл. Подключитесь к MySQL как пользователь *root* и запустите файл *mysql_setup.sql*.

```
# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> source mysql_setup.sql
mysql> quit
```

База данных подготовлена.

ВСПОМОГАТЕЛЬНЫЕ ФАЙЛЫ.

Из директории *~/locals* необходимо скопировать вспомогательные файлы, необходимые для работы спам фильтра.

```
# cp dspam.cron /etc/cron.daily/
```

```
# cp dspam.m4 /usr/share/sendmail-cf/mailer/
# cp ~/dspam-3.8.0/src/tools.mysql_drv/purge-4.1.sql /usr/local/bin
```

Создаем в системе пользователя *dspam*.

```
# useradd dspam -s /sbin/nologin
# passwd dspam
```

Открываем на редактирование файл *traindspam*. Эта программа необходима для обучения спам фильтра. Записываем туда пароль пользователя *dspam*, который вы только что ввели. Пароль надо менять в строке:

```
$imappasswd="mapsd";
```

Сохраните файл.

Теперь скопируем файл в общедоступную директорию, передадим его пользователю root и группе mail. Сменим права доступа.

```
# cp traindspam /usr/local/bin
# chown root.mail /usr/local/bin/traindspam
# chmod 0750 /usr/local/bin/traindspam
```

Для удобства создания почтовых ящиков пользователей в IMAP сервере, мы будем использовать программу *imap-user*. Программа создает почтовый ящик пользователя, необходимые папки и дает права пользователю *dspam* на доступ к папкам.

Мы немного отредактируем файл программы. Нам необходимо удалить строки, где вызывается программа saslpasswd2. Мы используем реальные учетные записи пользователей

Удалите из файла строки:

```
print "Will set '$user' password for SASL authentication\n";
system("/usr/sbin/saslpasswd2 -c $user");
```

Скопируйте файл в директорию */usr/sbin*.

```
# cp imap-user /usr/local/sbin/
```

В папке *quarantine* будет накапливаться спам. Что бы пользователю не приходилось самостоятельно очищать эту папку, можно заставить IMAP серер это делать самостоятельно. Для этого отредактируем файл */etc/cyrus.conf*.

В секцию EVENTS добавим следующую строку.

```
purgespam cmd="ipurge -f -d 3 user.*.quarantine" at=0300
```

Эта строка, заставляет сервер очищать папку *quarantine* раз в день, в 3 часа ночи.

В этом же файле добавляем строку, которая раз в 30 минут запускает программу обучения спам фильтра.

```
trainspam cmd="/usr/local/bin/traindspam" period=30
```

Файл можно сохранить.

Установим права доступа на директорию */var/dspam*.

```
chmod 750 /var/dspam
```

КОНФИГУРАЦИЯ SENDMAIL.

После того, как мы добавили мейлер *dspam*, в директорию */usr/share/sendmail-cf/mailer*. Нам необходимо заставить почтовый сервер передавать всю входящую почту не IMAP серверу, а спам фильтру.

Переходим в директорию */etc/mail*. И открываем на редактирование файл *sendmail.mc*.

Комментируем строки:

```
define(`confLOCAL_MAILER', `cyrusv2')dnl
define(`CYRUSV2_MAILER_ARGS', `FILE /var/lib/imap/socket/lmtp')dnl
MAILER(cyrusv2)dnl
```

Напомню, что комментарий в этом файле — dnl.

В конце файла добавляем:

```
define(`confLOCAL_MAILER',`dspam')dnl
MAILER(dspam)dnl
```

В результате последние строки файла должны выглядеть так:

```
define(`confLOCAL_MAILER',`dspam')dnl
MAILER(smtp)dnl
MAILER(dspam)dnl
```

Файл сохраняем и создаем конфигурационный файл.

```
# m4 sendmail.mc > sendmail.cf
```

ЗАПУСК ПРОГРАММ.

Мы создали и изменили все необходимые конфигурационные файлы. Теперь будем запускать программы.

Начнем со спам фильтра.

Сначала скопируем стартовый скрипт.

```
# cp dspam.rc /etc/rc.d/init.d/dspam
```

Проверим, запустился ли сервер и не выдавал ли он предупреждения.

```
# tail /var/log/messages
```

В идеале мы должны увидеть:

```
Dec  5 15:41:24 server dspam: 21799: [12/05/2008 15:41:24] Daemon process
starting
```

Запустим программу.

```
# service dspam start
```

Сделаем так, что бы фильтр запускался при старте сервера.

```
# chkconfig dspam on
# chkconfig dspam --list
dspam           0:выкл  1:выкл  2:вкл   3:вкл   4:вкл   5:вкл   6:выкл
#
```

Теперь перезапустим IMAP сервер.

```
# service cyrus-imapd restart
```

Удалим почтовые ящики всех пользователей, которые остались от предыдущих экспериментов.

Подключаемся к серверу пользователем cyrus.

```
# cyradm -u cyrus localhost
IMAP Password:
               server.st1.kryukov.biz>
```

Даем право на удаление всех ящиков пользователей, пользователю cyrus.

```
server.st1.kryukov.biz> sam user.* cyrus c
Setting ACL on user.student...OK.
Setting ACL on user.test...OK.
server.st1.kryukov.biz>
```

Удаляем все ящики.

```
server.st1.kryukov.biz> dm user.*
Deleting mailbox user.student...OK.
Deleting mailbox user.test...OK.
server.st1.kryukov.biz>
```

Выходим из программы.

Теперь создадим почтовый ящик пользователя student, при помощи программы imap-user, которую мы добавили в систему.

```
# imap-user student
Authenticating with the local Cyrus IMAP server
IMAP Password:
#
```

Программа спросит вас пароль, введите пароль пользователя cyrus.

Посмотрим, какие ящики были созданы.

```
# cyradm -u cyrus localhost
IMAP Password:
               server.st1.kryukov.biz> lm
user.student (\HasChildren)
user.student.Drafts (\HasNoChildren)
user.student.Sent (\HasNoChildren)
user.student.Templates (\HasNoChildren)
user.student.Trash (\HasNoChildren)
user.student.ham (\HasNoChildren)
user.student.quarantine (\HasNoChildren)
user.student.spam (\HasNoChildren)
server.st1.kryukov.biz> lam user.student*
user.student:
  student lrswipkxtecda
user.student.Drafts:
```

```
  student lrswipkxtecda
user.student.Sent:
  student lrswipkxtecda
user.student.Templates:
  student lrswipkxtecda
user.student.Trash:
  student lrswipkxtecda
user.student.ham:
  student lrswipkxtecda
  dspam lrswipkxtecd
user.student.quarantine:
  student lrswipkxtecda
  anonymous lrsp
user.student.spam:
  student lrswipkxtecda
  dspam lrswipkxtecd
server.st1.kryukov.biz>
```

Как видите, был создан почтовый ящик пользователя и несколько папок. В том числе, необходимые для работы спам фильтра. Так же был разрешен доступ пользователю *dspam* на папки *ham* и *spam*, что бы спам фильтр мог удалять сообщения из них.

Перезапускаем sendmail.

ДОПОЛНИТЕЛЬНЫЕ УТИЛИТЫ.

В комплект поставки спам фильтра ыходят дополнительные утилиты. Сейчас мы посмотрим некоторые из них.

DSPAM позволяет каждому пользователю вести свой список spam сообщений. После установки, фильтр надо обучать. На это тратиться некоторое количество времени. Если вы хотите *передать* накопленный опыт существующих пользователей вновь создаваемому пользователю, воспользуйтесь утилитой *dspam_merge*.

Сначала вы должны указать пользователей, чьи предпочтения вы собираетесь копировать. После параметра *–o* указывается пользователь, которому вы будуте передавать *накопленный опыт*.

```
# dspam_merge artur -o tester
Destination user: tester
Merging user: artur
processed 0 tokens
storing merged tokens...
completed.
#
```

Программа *dspam_stats* показывает статистику работы фильтра.

По умолчанию она выводит информацию обо всех почтовых ящиках, на которые приходили письма. Если хотите получить данные только по одному пользователю, укажите его имя в качестве аргумента.

Параметр *–H* заставляет выводить информацию в формате, понятном для человека.

Например:

```
# dspam_stats -H artur
artur:
       TP True Positives: 13018
       TN True Negatives: 3790
       FP False Positives: 97
       FN False Negatives: 802
       SC Spam Corpusfed: 367
       NC Nonspam Corpusfed: 0
       TL Training Left: 0
       SHR Spam Hit Rate 94.20%
       HSR Ham Strike Rate: 2.50%
       OCA Overall Accuracy: 94.92%
```

Для сброса накопленной статистики, используется параметр *–r*.

НАСТРОЙКА ПРОВЕРКИ SPF В SENDMAIL.

СБОРКА НЕОБХОДИМЫХ ПРОГРАММ.

К сожалению я не нашел готовых пакетов для CentOS, поэтому все необходимые компоненты мы соберем из исходных кодов.

Для того, что бы заставить sendmail учитывать spf записи, необходимо поставить специальный мильтер: *smf-spf*. Получить исходные коды программы можно тут: http://smfs.sourceforge.net/smf-spf.html. Так же, для работы нам потребуется библиотека *libspf2*. Исходные коды библиотеки можно получить на сайте проекта: http://www.libspf2.org.

Перейдите в домашнюю директорию и скачайте исходные коды библиотеки

```
# cd
# wget http://www.libspf2.org/spf/libspf2-1.2.9.tar.gz
```

Распакуйте полученный архив, и перейдите в появившуюся директорию.

```
# tar -xzf libspf2-1.2.9.tar.gz
# cd libspf2-1.2.9
```

Запустим программу конфигурации, затем соберем и установим программу.

```
# ./configure
# make
# make install
```

Для того, что бы другие программы «увидели» эту библиотеку, необходимо сконфигурировать кеш библиотек.

Открываем в текстовом редакторе файл /etc/ld.so.conf.d/smfspf.conf. Этого файла не существует, открыв его в редакторе, мы его создадим. Добавляем в файл одну строку:

```
/usr/local/lib
```

Это директория, в которой находится библиотека.

Пересоздадим кеш библиотек, и проверим, появилась ли в нем информация о библиотеке.

```
# ldconfig
# ldconfig -p | grep spf
        libspf2.so.2 (libc6) => /usr/local/lib/libspf2.so.2
        libspf2.so (libc6) => /usr/local/lib/libspf2.so
```

Теперь займемся мильтером. Перейдите в домашнюю директорию, скачайте исходные коды программы и распакуйте её.

```
# cd
# wget http://prdownloads.sourceforge.net/smfs/smf-spf-
2.0.2.tar.gz?download
# tar -xzf smf-spf-2.0.2.tar.gz
```

Устанавливаем пакет sendmail-devel, без него мильтер не соберется.

```
# yum install sendmail-devel
```

Перейдите в появившуюся директорию и запустите сборку программы.

```
# cd smf-spf-2.0.2
# make
# make install
```

Скопируйте стартовый скрипт мильтера в директорию */etc/rc.d/init.d*.

```
# cp init/smfspf.redhat /etc/rc.d/init.d/smfspf
```

КОНФИГУРАЦИЯ ПРОГРАММ.

Откройте в редакторе конфигурационный файл программы */etc/mail/smfs/smf-spf.conf*.

При помощи параметра *WhitelistIP* вы можете указать IP адреса сетей (в формате CIDR), откуда письма будут приниматься без проверки.

По умолчанию, в файле определены все сети, предназначенные для внутреннего использования и сеть 127.0.0.0.

```
WhitelistIP     127.0.0.0/8
WhitelistIP     10.0.0.0/8
WhitelistIP     172.16.0.0/12
WhitelistIP     192.168.0.0/16
```

Параметр *WhitelistPTR* используется для указания имени домена или машины, откуда можно принимать письма.

WhitelistFrom позволяет определить email, с которого можно принимать письма.

И т.д. Названия конфигурационных параметров в файле достаточно информативны.

Настройка *sendmail* сводиться в добавлении двух строк в файл с макросами *sendmail.mc*.

Перейдите в директорию */etc/mail* и откройте в редакторе файл с макросами. Перед определением милтера антивируса, добавьте две строки:

```
define(`confMILTER_MACROS_HELO', confMILTER_MACROS_HELO`, {verify}')dnl
INPUT_MAIL_FILTER(`smf-spf', `S=unix:/var/run/smfs/smf-spf.sock,
T=S:30s;R:1m')dnl
```

Создайте конфигурационный файл.

```
# m4 sendmail.mc > sendmail.cf
```

Запустим мильтер.

```
# chkconfig smfspf on
```

```
# service smfspf start
```

Перезапустим почтовый сервер.

```
# service sendmail restart
```

Для проверки, отправьте письмо по адресу student@ваш.домен. Письмо можно отправить с любого почтового сервера. Посмотрите логи системы электронной почты (*/var/log/maillog*).

Я послал письмо с *mail.ru*. Вот что было написано в логах.

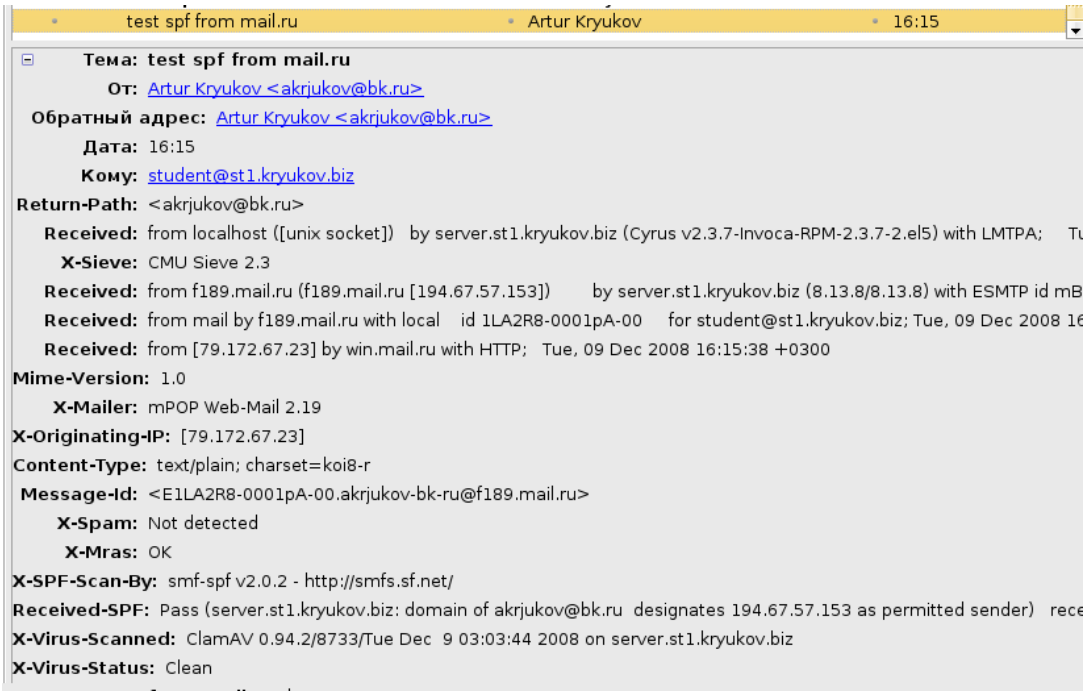
```
Dec  9 16:19:33 server smf-spf[26879]: SPF pass: 194.67.57.153,
f189.mail.ru, f189.mail.ru, <akrjukov@bk.ru>
```

```
Dec  9 16:19:33 server sendmail[26962]: mB9DJWdN026962:
from=<akrjukov@bk.ru>, size=678, class=0, nrcpts=1, msgid=<E1LA2R8-0001pA-
00.akrjukov-bk-ru@f189.mail.ru>, proto=ESMTP, daemon=MTA, re-
lay=f189.mail.ru [194.67.57.153]
```

```
Dec  9 16:19:33 server sendmail[26962]: mB9DJWdN026962: Milter add: header:
X-SPF-Scan-By: smf-spf v2.0.2 - http://smfs.sf.net/
```

```
Dec  9 16:19:33 server sendmail[26962]: mB9DJWdN026962: Milter add: header:
Received-SPF: Pass (server.st1.kryukov.biz: domain of akrju-
kov@bk.ru\n\tdesignates 194.67.57.153 as permitted send-
er)\n\treceiver=server.st1.kryukov.biz; client-
ip=194.67.57.153;\n\tenvelope-from=<akrjukov@bk.ru>; helo=f189.mail.ru;
```

Посмотрим письмо в почтовом ящике и его заголовки.



Кстати, у *mail.ru* определена запись *spf*, правда в более мягком режиме.

```
# dig mail.ru TXT
```

```
mail.ru.                3600    IN      TXT     "v=spf1 ip4:194.67.57.0/24
ip4:194.67.23.0/24 ip4:194.67.45.0/24 ip4:195.239.211.0/24
ip4:194.186.55.0/24 ip4:195.239.174.0/24 ip4:94.100.176.0/20 ~all"
```

СЕРЫЕ СПИСКИ.

Еще один способ борьбы со спамом — использование так называемых серых списков (gray list, graylisting).

Если бы спам рассылался только с почтовых серверов, для защиты от него было бы достаточно использование только черных списков. Не секрет, что обычно спам рассылается не с почтовых серверов, а с бот сетей. Т.е. для рассылки используются зараженные компьютеры обыкновенных пользователей. Этим можно воспользоваться.

Почтовый сервер может не принять письмо сразу, а выдать сообщение о временной ошибке. Отправляющий почту, почтовый сервер попытается повторить оправку почты через некоторое время, обычно через полчаса.

При использовании серых списков, почтовый сервер *всегда* не принимает первое письмо от отправителя, сославшись на внутреннюю ошибку. Но он записывает во внутреннюю базу данных информацию об этой попытке, и следующие письма от отправителя, находящегося в этой базе, принимает.

При отсылке писем спам ботами, обычно второй попытки уже не происходит.

Недостаток этого метода — большая задержка при получении первого письма. Но все остальные письма от отправителя будут приходить нормально.

ПОДКЛЮЧЕНИЕ К SENDMAIL.

Для включения серых списков в sendmail необходимо установить пакет milter-greylist. Он находится в подключенном нами репрозитарии rpmforge (dag).

```
# yum install milter-greylist
```

В файле с макросами /etc/mail/sendmail.mc надо добавить следующие строки:

```
INPUT_MAIL_FILTER(`greylist', `S=local:/var/milter-greylist/milter-
greylist.sock')dnl

define(`confMILTER_MACROS_CONNECT', `j, {if_addr}')dnl

define(`confMILTER_MACROS_HELO', `{verify}, {cert_subject}')dnl

define(`confMILTER_MACROS_ENVFROM', `i, {auth_authen}')dnl

define(`confMILTER_MACROS_ENVRCPT', `{greylist}')dnl
```

Эти строки необходимо добавить до определения других *INPUT_MAIL_FILTER*.

Если вы добавляли поддержку spf, то удалите старый вариант конфигурационного параметра *confMILTER_MACROS_HELO*:

```
define(`confMILTER_MACROS_HELO', confMILTER_MACROS_HELO`, {verify}')dnl
```

Создайте конфигурационный файл sendmail.

```
# m4 sendmail.mc > sendmail.cf
```

НАСТРОЙКА MILTER-GREYLIST.

Конфигурационный файл программы — */etc/mail/greylist.conf*.

Основная задача конфигурационного файла — определить белые списки, серверов, адресатов и пр. При помощи белых списков, мы указываем программе, окуда следует принимать почту без задержек.

В конфигурационном файле сначала определяются списки, которым присваиваются имена. Например,

```
list "my network" addr { 127.0.0.1/8 10.0.0.0/8 192.0.2.0/24 }
```

Это делается в большинстве случаев для удобства, что бы потом не перечислять все адреса в спсике контроля доступа.

В файле кроме локальных сетей, определен список неправильных почтовых серверов, которые не делают повторной попытки посылки почты или не корректно реагируют на сообщения о невозможности доставки почты.

```
list "broken mta" addr
```

В это список вы будете добавлять IP адреса других неправильных серверов, обнаруженых вами.

В конце файла определены списки контроля доступа:

```
acl whitelist list "my network"
acl whitelist list "broken mta"
acl greylist list "grey users" delay 30m autowhite 3d
acl whitelist default
```

В принципе вполне рабочая конфигурация. Но будет пропускать все письма ☺. Проблема в последней строке, она пропускает всю почту. Изначально, в конфигурационном файле существует список пользователей, при посылке почты которым включается механизм серых списков.

```
list "grey users" rcpt {  \
        user1@example.com \
        user2@example.com \
        user3@example.com \
}
```

Предполагалось, что вы должны были вносить в список пользователей. Но если их много, то список получиться очень большой.

Поэтому, мы удалим из файла две строки:

```
acl greylist list "grey users" delay 30m autowhite 3d
acl whitelist default
```

А в самом конце файла добавим строку, заставляющую включать механизм серых списков для всех адресатов сервера.

```
acl greylist default
```

Это нужно писать именно последней строкой. Потому, что правила в конфигурационном файле рассматриваются в том порядке, в котором они написаны.

ЗАПУСК ПРОГРАММЫ.

Запускать программу будем при помощи стартового скрипта.

```
# service milter-greylist start
```

Незабудьте сдеалать так, что бы программа автоматически запускалась при старте компьютера.

```
# chkconfig milter-greylist on
```

И обязательно перезапустите sendmail.

Попробуйте отправить письмо на адрес *student* с любого почтового сервера. Помотрите логи сервера.

```
# tail /var/log/maillog
```

```
Dec 16 14:50:11 server smf-spf[26879]: SPF pass: 194.67.57.127,
f121.mail.ru, f121.mail.ru, <akrjukov@bk.ru>
```

```
Dec 16 14:50:11 server milter-greylist: mBGBoASk021492: addr
f121.mail.ru[194.67.57.127] from <akrjukov@bk.ru> to <stu-
dent@st1.kryukov.biz> delayed for 00:30:00 (ACL 104)
```

```
Dec 16 14:50:11 server sendmail[21492]: mBGBoASk021492: Milter:
to=<student@st1.kryukov.biz>, reject=451 4.7.1 Greylisting in action,
please come back later
```

Как видите, письмо не было принято. Если почтовый сервер на mail.ru работает согласно стандарта, то он повторит попытку через полчаса и мы получим письмо.

Сразу получения письма, еще раз повторите попытку отослать письмо с того же почтового сервера, что и раньше. На это раз письмо попадет в почтовый ящик сразу.

РЕШЕНИЕ ПРОБЛЕМ.

Если письмо не пришло через полчаса, то у нас проблемы. Напрмер, для отправки почты mail.ru может использовать несколько машин. Об этом свидетельствует запись TXT в их домене.

```
# dig mail.ru TXT
```

```
mail.ru.                3600    IN      TXT     "v=spf1 ip4:194.67.57.0/24
ip4:194.67.23.0/24 ip4:194.67.45.0/24 ip4:195.239.211.0/24
ip4:194.186.55.0/24 ip4:195.239.174.0/24 ip4:94.100.176.0/20 ~all"
```

Как видно из этой записи, существует много серевров, окуда может прийти почта.

Теперь посмотрим базу серых списков. Это обыкновенный текстовый файл, находящийся в директории */var/milter-greylist*.

```
# cat greylist.db
```

```
#
```

```
# greylisted tuples
```

```
#
```

```
# Sender IP     Sender e-mail   Recipient e-mail        Time accepted
```

```
194.67.57.127   <akrjukov@bk.ru>        <student@st1.kryukov.biz>
1229430011 # 2008-12-16 15:20:11
```

```
194.67.23.236   <akrjukov@bk.ru>        <student@st1.kryukov.biz>
1229431490 # 2008-12-16 15:44:50
```

Как видно из записей в этой базе, почтовый сервер mail.ru уже два раза пытался отправить нам письмо, но с разных машин: 194.67.57.127 и 194.67.23.236. С точки зрения программы — это разные машины. Велика вероятность того, что следующая попытка отправки письма будет произведена с другого почтового сервера и письмо еще раз не будет доставлено.

Что делать? Во-первых, не забывайте, что серые списки — это, прежде всего, защита от ботнетов. Поэтому вам придется вписать в белые списки все публичные почтовые службы (точнее их сервера), вроде mail.ru, google и прочих. Работенка та еще, но иначе вы вообще можете не дождаться письма.

Для этого определим список почтовых серверов. Например, для mail.ru этот список будет выглядеть так:

```
list "mail.ru" addr { \
```

```
194.67.57.0/24 194.67.23.0/24 194.67.45.0/24 \
195.239.211.0/24 194.186.55.0/24 195.239.174.0/24 \
94.100.176.0/20 }
```

В конце конфигурационного файла, но перед записью

```
acl greylist default
```

Добавляем acl:

```
acl whitelist list "mail.ru"
```

Таким образом, мы будем пропускать все письма, приходящие с машин с указанными IP адресами. Если на mail.ru заведется спамер, то с ним будет бороться антиспам фильтр.

Почему я не использовал запись MX для получения списка IP адресов серверов, а пользовался записью TXT? Ответ простой — MX определят сервера *принимающие* почту для домена. Эти сервера могут не использоваться для отправки почты.

WEB ИНТЕРФЕЙС.

После настройки почтового сервера можно поставить WEB интерфейс, для доступа к почтовым ящикам пользователей.

Существует огромное количество WEB интерфейсов, какой из них лучше или хуже, в итоге будете решать вы сами. Лично мне нравиться программа Round Cube. Именно его мы и будем ставить.

УСТАНОВКА ПРОГРАММЫ.

Для работы нам потребуется WEB сервер. Проверяем, установлен ли WEB сервер на вашей машине.

```
# rpm -qa '*httpd*'
```

Если он не установлен, ставим его.

```
# yum install httpd
```

Программа написана на PHP, поэтому должны быть установлены соответствующие пакеты: *php*, *php-ldap*, *php-mysql*, *php-pecl-Fileinfo*, *php-gd*, *php-mcrypt*, *php-mbstring*.

```
# yum install php-pecl-Fileinfo php-gd php-mcrypt php-mbstring
```

Переходим в директорию */var/www/html*. И создаем в ней директорию *round*.

```
# cd /var/www/html
# mkdir round
```

Передаем созданную директорию пользователю и группе *apache*. Переходим в эту директорию.

```
# chown apache.apache round
# cd round
```

Скачиваем архив с программой с сайта http://roundcube.net/downloads. Распаковываем в домашней директории.

```
# cd
# tar -xzf roundcubemail-0.1.1.tar.gz
```

Заходим в появившуюся директорию и копируем все файлы, находящиеся в ней в директорию *round*.

```
# cd roundcubemail-0.1.1
# cp -R * /var/www/html/round
```

Передаем пользователю и группе *apache* все файлы программы.

```
# chown -R apache.apache /var/www/html/round
```

ПОДГОТОВКА БАЗЫ ДАННЫХ.

Программа Round Cube использует базу данных. Поэтому мы должны эту базу подготовить.

Подключаемся к MySQL с правами пользователя root.

```
# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 45
```

```
Server version: 5.0.45 Source distribution


Type 'help;' or '\h' for help. Type '\c' to clear the buffer.


mysql>
```

Создаем базу с именем round, и даем пользователю round все права доступа на эту базу. Заодно устанавливаем пароль пользователю round.

```
mysql> CREATE DATABASE round;
mysql> GRANT ALL PRIVILEGES ON round.* TO 'round'@'localhost' IDENTIFIED BY
'password';
mysql> FLUSH PRIVILEGES;
mysql> quit
```

НАСТРОЙКА WEB СЕРВЕРА APACHE.

Открываем на редактирование конфигурационный файл */etc/httpd/conf/httpd.conf*.

Почти в самом конце этого файла ищем строку, содержащую параметр *NameVirtualHost*. Убираем символ комментария в начале строки.

```
NameVirtualHost *:80
```

Таким образом мы включили виртуальных хостинг. Т.е. наш WEB сервер сможет одновременно поддерживать несколько WEB сайтов.

Файл сохраняем и закрываем.

Переходим в директорию */etc/httpd/conf.d*. При подключении к серверу мы будем шифровать соединение. Подумайте сами, вы будете вводить логины пользователей и их пароли. Если эти данные при передаче не будут зашифрованы, потенциально они смогут попасть в руки злоумышленникам.

Открываем на редактирование файл *ssl.conf*. Нас интересует строки с параметра:

```
<VirtualHost _default_:443>
```

Все строки ниже, включая указанную строку, мы должны перенести из этого файла в файл *round.conf*. Скопируйте файл *ssl.conf*.

```
# cp ssl.conf round.conf
```

В обеих файлах удалите не нужные строки. В файле *ssl.conf* все строки, начиная с:

```
<VirtualHost _default_:443>
```

В файле *round.conf* все строки, до указанной выше.

В результате файл *round.conf* должен содержать следующие параметры:

```
<VirtualHost *:443>
```

```
DirectoryIndex index.html index.php
```

Параметр определяет файлы по умолчанию.

```
ServerAdmin student@st1.kryukov.biz
```

Параметр определяет email пользователя, отвественного за сервер.

```
DocumentRoot "/var/www/html/round"
```

Параметр определяет директорию, откуда сервер будет брать файлы.

```
ServerName round.st1.kryukov.biz:443
```

По этому имени, Apache понимает какому виртуальному серверу пришел запрос.

```
ErrorLog /var/log/httpd/round_error.log
CustomLog /var/log/httpd/round_access.log common
```

Два параметра, определяющие индивидуальные файлы журнальной регистрации для виртуального сервера.

```
LogLevel warn
SSLEngine on
```

Включение поддержки SSL.

```
SSLProtocol all -SSLv2
SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
SSLCertificateFile /etc/pki/tls/certs/round-cert.pem
SSLCertificateKeyFile /etc/pki/tls/private/round-key.pem
```

Два параметра, определяющие ключ и сертификат сервера.

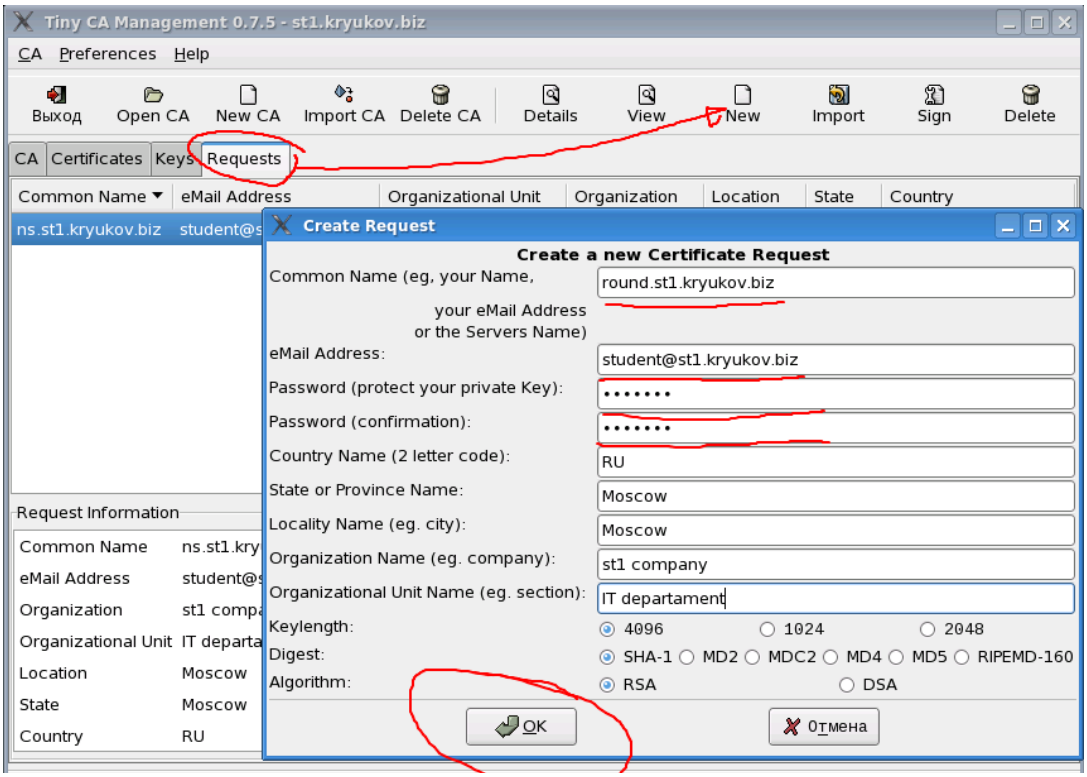
Остальные параметры не изменялись.

```
<Files ~ "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>
<Directory "/var/www/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>
SetEnvIf User-Agent ".*MSIE.*" \
         nokeepalive ssl-unclean-shutdown \
         downgrade-1.0 force-response-1.0
CustomLog logs/ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

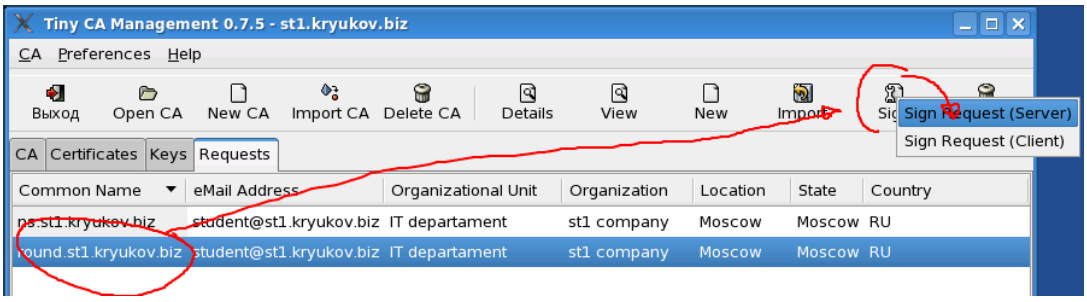
СЕРТИФИКАТ ДЛЯ WEB СЕРВЕРА.

Подключитесь к виртуальной машине, запустите графическую оболочку. Запустите программу TinyCA2.

Создаем запрос на сертификацию.

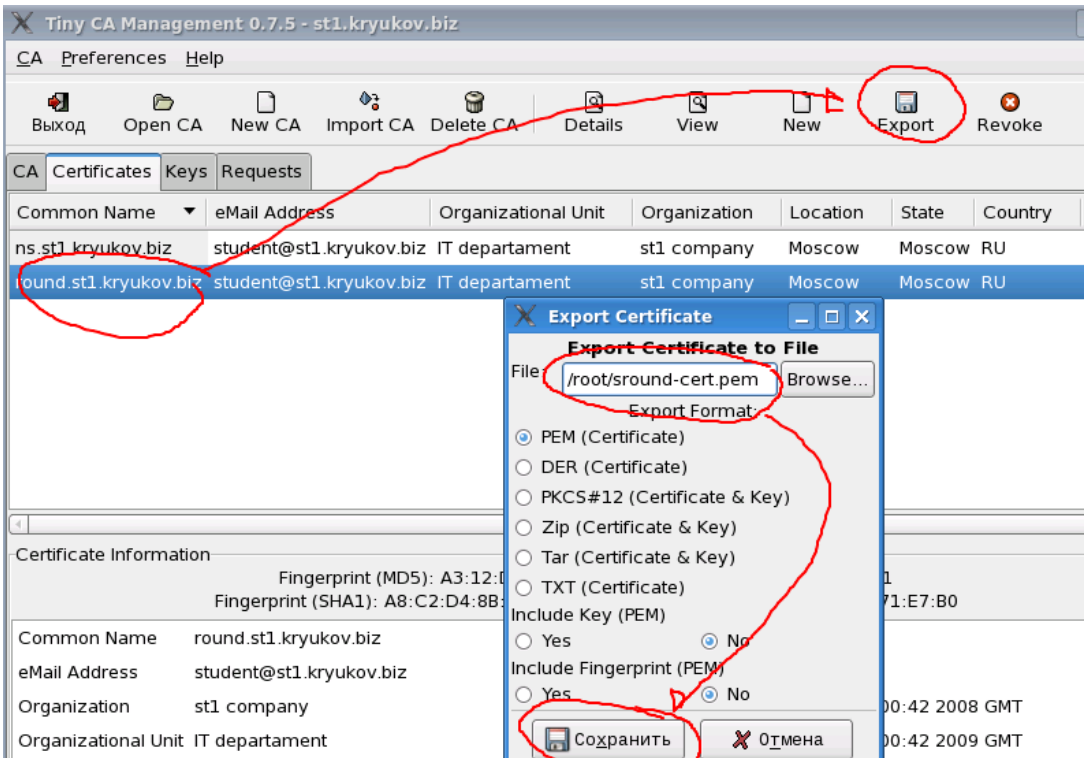


Теперь подпишем сертификат.

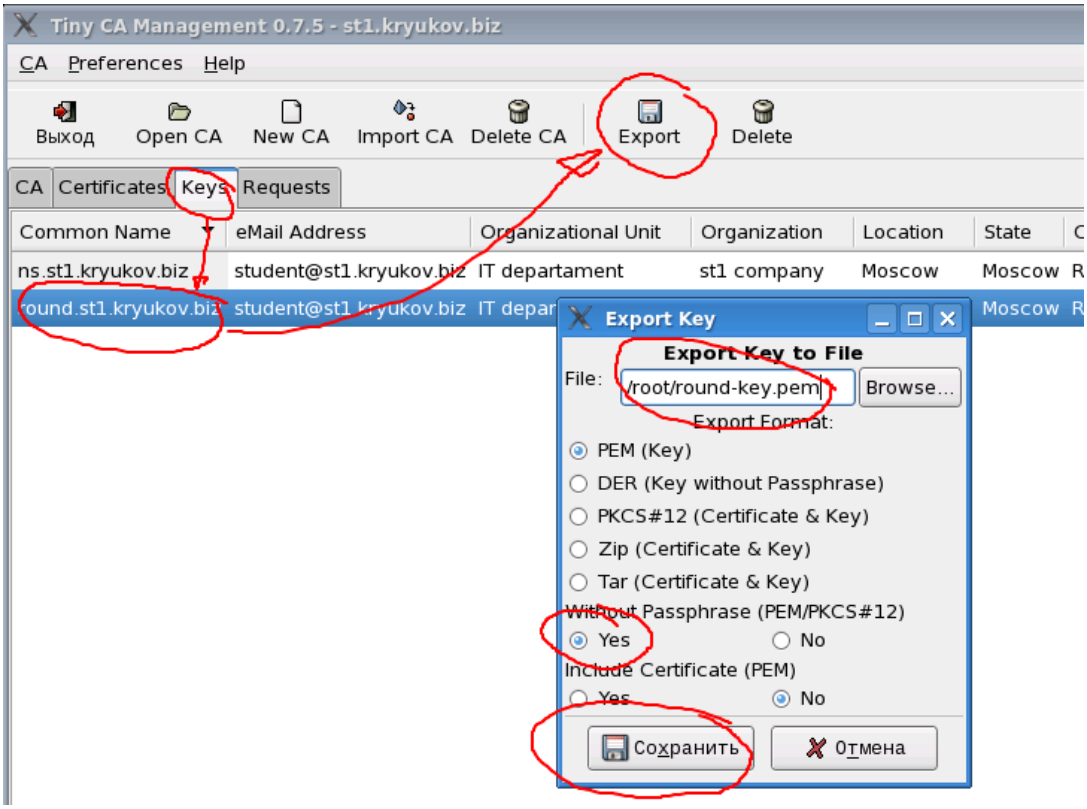


Выбираем сертификат для сервера. В следующем окне вводим пароль CA.

Экспортируем сертификат сервера.



Экспортируем ключ.



Вводим пароль ключа. Закрываем программу, закрываем графическую оболочку.

Переносим файл *round-cert.pem* в директорию */etc/pki/tls/certs*:

```
# mv round-cert.pem /etc/pki/tls/certs
```

Переносим файл round-key.pem в директорию /etc/pki/tls/private:

```
# mv round-key.pem /etc/pki/tls/private
```

Скопируем файл сертификата CA в корневую директорию нашего WEB сервера, что бы клиенты могли его скачивать и устанавливать.

```
# cp /etc/pki/CA/st1.kryukov.biz-cacert.pem /var/www/html/round/CA.crt
```

DNS.

Добавляем запись о машине round в DNS сервер.

Открываем на редактирование файл зоны */var/named/chroot/var/named/master.st1.kryukov.biz*. Разумеется, вы должны отредактировать файл вашей зоны.

Добавляем запись:

```
round   IN      CNAME   ns
```

Не забудьте увеличить серийный номер зоны.

Файл сохраняем, закрываем.

Проверяем синтаксис файла описания зоны.

```
# named-checkzone st1.kryukov.biz
/var/named/chroot/var/named/master.st1.kryukov.biz
```

Заставляем DNS сервер перечитать файл зоны.

```
# killall -HUP named
```

FIREWALL.

Теперь мы должны разрешить подключение к WEB серверу в firewall.

Открываем на редактирования файл ~/bin/rc.fw и добавляем следующие строки, в функции init.

```
### WEB
$IPT -A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
```

Файл сохраняем, закрываем. Применяем изменения.

НАСТРОЙКА WEB ИНТЕРФЕЙСА.

Запускаем WEB сервер.

```
service httpd start
```

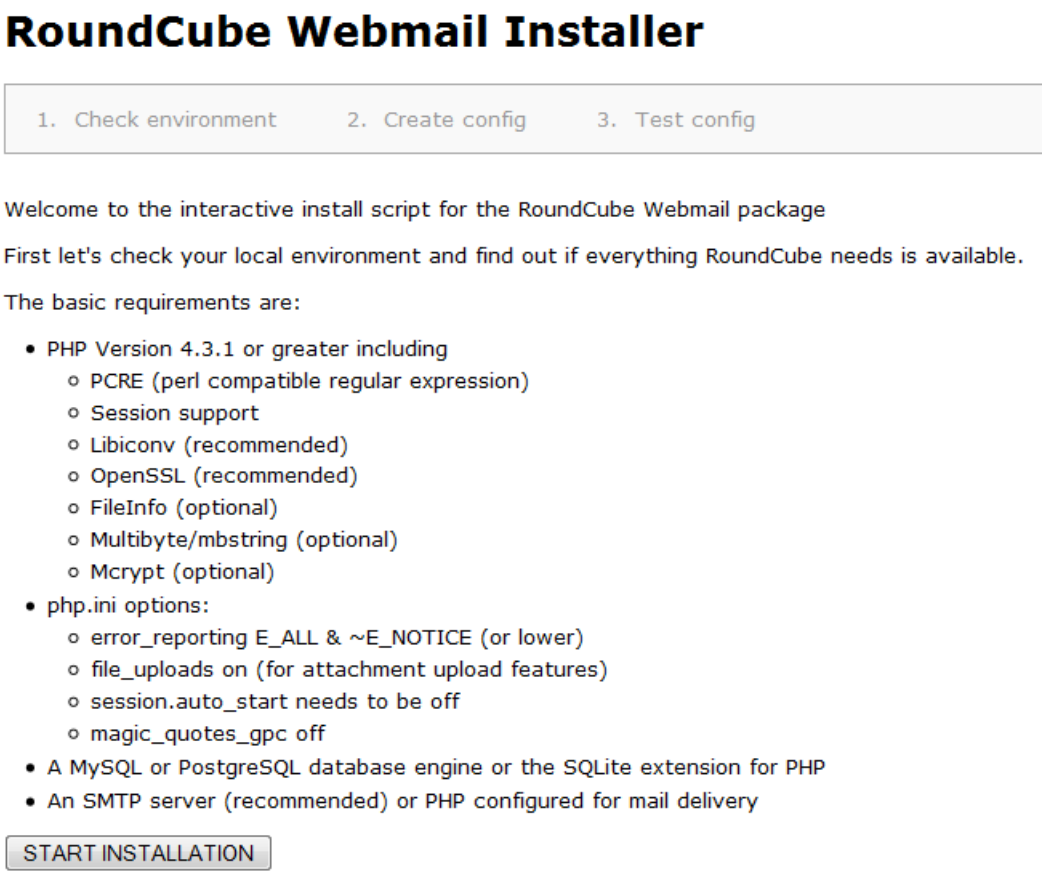
В первую очередб скачаем и установим сертификат нашего CA. Для этого открываем браузер и вводим следующий URL

```
https://round.st1.kryukov.biz/CA.crt
```

Разумеется, вы будете использовать имя вашего домена.

После установки сертификата, можно приступать к настройке Round Cube. Наберите следующий URL:

```
https://round.st1.kryukov.biz/installer
```



Нажмите кнопку START INSTALLATION.

Checking PHP version

Version: **OK** (PHP 5.1.6 detected)

Checking PHP extensions

The following modules/extensions are *required* to run RoundCube:

PCRE: **OK**
Session: **OK**

The next couple of extensions are *optional* but recommended to get the best performance:

FileInfo: **OK**
Libiconv: **OK**
Multibyte: **OK**
OpenSSL: **OK**
Mcrypt: **OK**
GD: **OK**

Checking available databases

Check which of the supported extensions are installed. At least one of them is required.

MySQL: **OK**
MySQLi: **OK**
PostgreSQL: **NOT AVAILABLE** (Not installed)
SQLite (v2): **NOT AVAILABLE** (Not installed)

Check for required 3rd party libs

This also checks if the include path is set correctly.

PEAR: **OK**
DB: **OK**
MDB2: **OK**
Net_SMTP: **OK**
Mail_mime: **OK**
iilConnection: **OK**

Checking php.ini/.htaccess settings

file_uploads: **OK**
session.auto_start: **OK**
magic_quotes_gpc: **OK**
magic_quotes_sybase: **OK**

NEXT

Нажимаем кнопку NEXT.

Database setup

db_dsnw

Database settings for read/write operations:

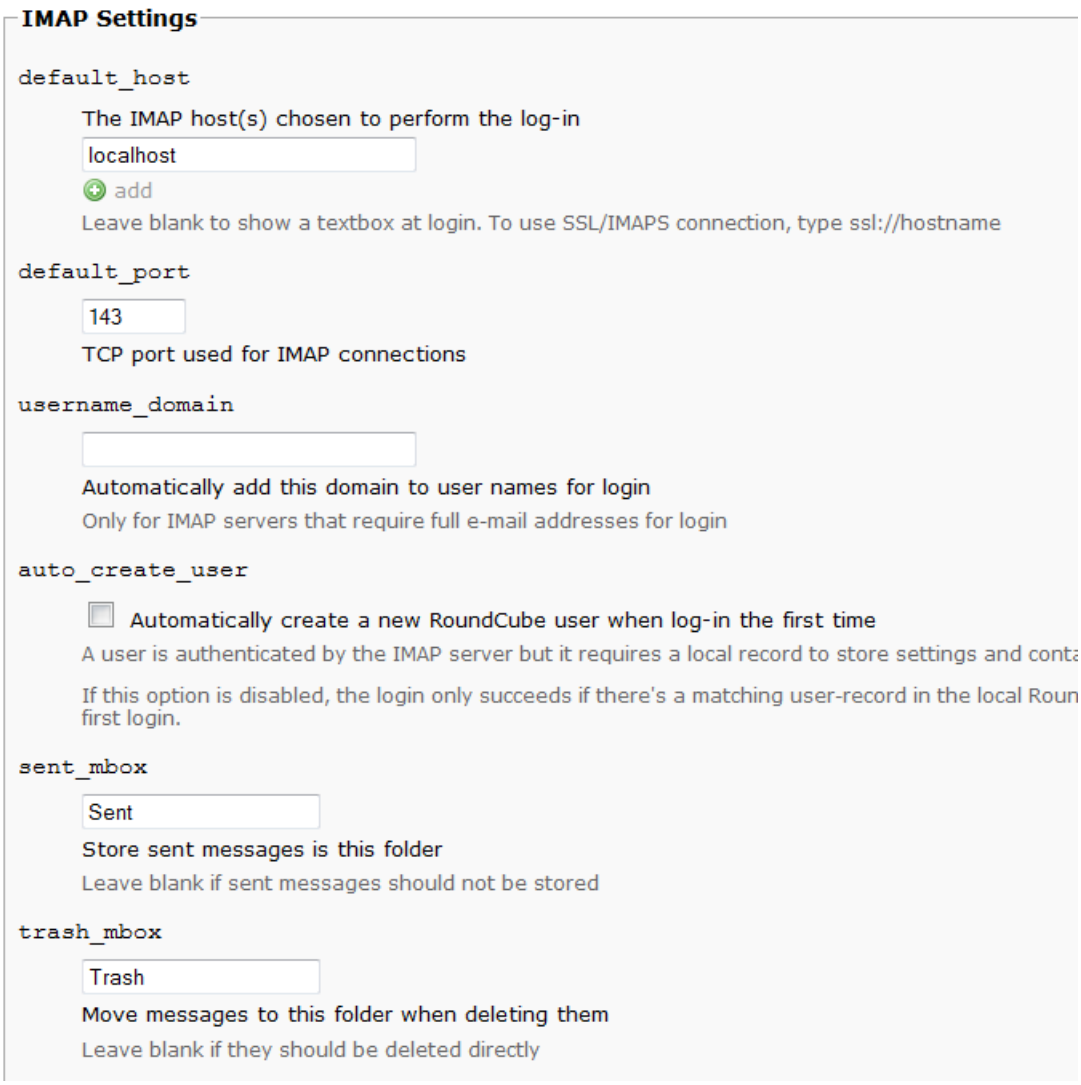
MySQL — Database type
localhost — Database server
round — Database name
round — Database user name (needs write permissions)
password — Database password

db_backend

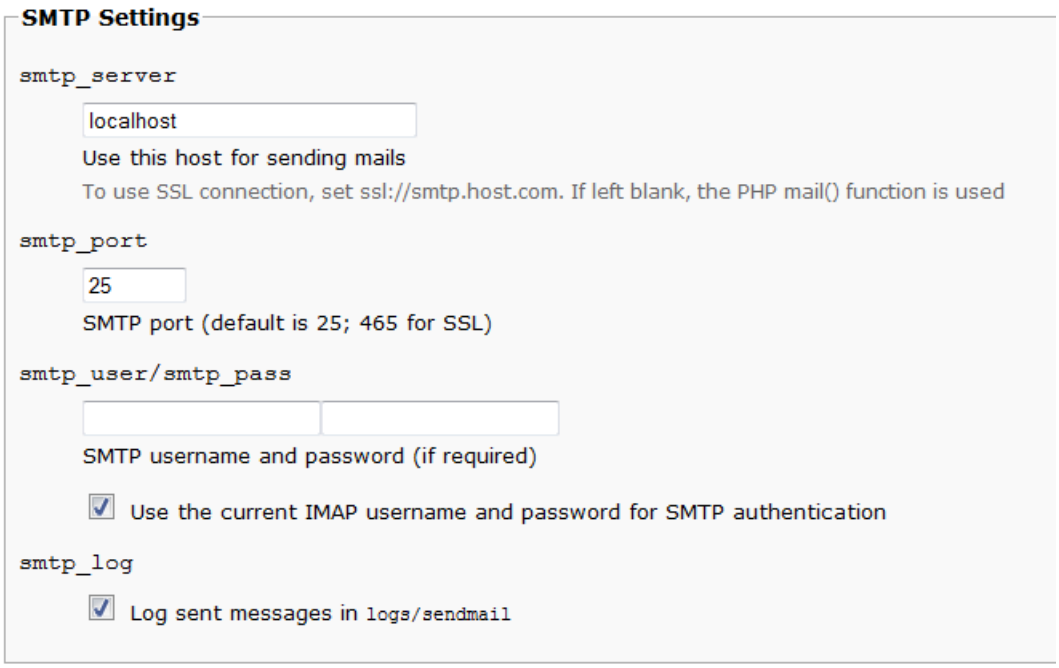
DB

PEAR Database backend to use

Обязательно выберите DB в db_backend!



Обязательно разрешите автоматическое создание почтовых ящиков (*auto_create_user*).



При указании параметров почтового сервера используйте имя *localhost* или IP *127.0.0.1*.

Программа сформирует содержимое двух файлов: *main.inc.php* и *db.inc.php*. Вы должны создать их в директории config и скопировать содержимое, которое показано в окне браузера, в эти файлы.

В последнем окне вам предложат инициализировать базу данных и проверить правильность подключения к почтовому серверу.

После проверки обязательно удалите папку *installer*.

Для доступа к WEB интерфейсу введите URL:

https://round.st1.kryukov.biz

ПОДДЕРЖКА НЕСКОЛЬКИХ ПОЧТОВЫХ ДОМЕНОВ.

Почтовый сервер, который мы настроили в предыдущих разделах, может поддерживать несколько почтовых доменов.

Предположим, что мы должны добавить поддержку почты для домена any.com. Что нам потребуется изменить в конфигурации сервера?

1. В DNS сервере, отвечающем за домен any.com добавить запись *MX*, ссылающуюся на наш почтовый сервер.

```
any.com. IN MX 5 ns.st1.kryukov.biz.
```

2. В файле */etc/mail/local-host-names* на новой строке добавить имя домена *any.com*.
3. Завести почтовый ящик пользователей.

Но если ограничиться только перечисленными выше действиями, то могут возникнуть неприятные ситуации. Предположим, что в доменах *st1.kryukov.biz* и *any.com* есть почтовые ящики с одинаковым названием:

- student@st1.kryukov.biz
- student@any.com

Почтовый адрес состоит из двух частей:

- Правая часть (после символа @) — определяет пчотовый сервер, куда следует отсылать почту. Тут может быть имя машины или имя домена. В последнем случае имя почтового сервера определяется при помощи записи MX.
- Левая часть (перед символом @) — определяет имя почтового ящика, куда следует доставить почту.

Если вернуться к нашему примеру, то правая часть адреса в обоих случаях указывает на машину *ns.st1.kryukov.biz*. Таким образом, письма отправление по первому и второму адресу будут попадать в один и тот же почтовый ящик — student.

Как разнести почту по разным почтовым ящикам?

1. Во первых, создаем дополнительный почтовый ящик any_student.
2. Во вторых используем таблицу виртуальных пользователей /etc/mail/virtusertable.

Наша задача, следать так, что бы письма, приходящие на *student@any.com* попадали в почтовый ящик *any_student*.

МАКРОС VIRTUSERTABLE.

Если посмотерть содержимое файла *sendmail.mc*, то там можно обнаружить следующую строку:

```
FEATURE(`virtusertable', `hash -o /etc/mail/virtusertable.db')dnl
```

Макрос заставляет sendmail использовать таблицу виртуальных пользователей, выполненную в виде хеш файла */etc/mail/virtusertable.db*. Параметр *–o* говорит, что этот файл не обязательный (optional), и sendmail запуститься даже если его не будет.

Сам файл virtusertable.db — это бинарный файл и никто его напрямую не редактирует. Обычно создается текстовый файл, который затем преоьразуется в бинарную базу.

В директории /etc/mail необходимо создать или, если он уже был, отредактировать файл virtusertable. Файл будет состоять из строк, в которых будет два поля:

- Уникальный ключ поиска[1].
- Значение, которое по этому ключу будет найдено.

В нашем случае в качестве ключа поиска можно использовать email *student@any.com*, а в поле значение написать имя локального почтового ящика, куда следует отправить почту.

```
student@any.com     any_student
```

Но это не очень удачное решение. Предположим, что у нас в двух доменах встречается около пятиста одинаковых email. Вам придется написать пятьсот строк в файле virtusertable! Что бы этого не делать, можно воспользоваться следующей записью.

```
@any.com      any_%1
```

В первом поле мы говорим, что всю почту, предназначенную для домена any.com следует направлять в указанный далее почтовый ящик. А вот во втором поле у нас указан хитрый ящик.

Давайте разберемся, что такое *%1*. Email состоит из нескольких частей. *%1* — это имя почтового ящика. Таким образом, мы говорим, что имя локального почтового ящика будет начинаться с *any_*, а дальше будет использоваться имя, указанное в email.

Т.е. адрес будет *student@any.com*, то письмо попадет в локальный почтовый ящик *any_student*. Письмо для адресата *director@any.com* будет помещено в почтовый ящик *any_director* и так далее.

В дальнейшем, если вы захотите создать почтовый ящик для пользователя в домене *any.com*, имя ящика должно начинаться с *any_*.

[1] Обратите внимание на слово уникальный. Это значит, что первое поле в этом файле не должно повторяться.