Содержание

Основы

1. Основные операции и утилиты

Запуск/остановка сервисов

Сервисы в операционных системах семейства Unix/Linux называются демонами. Скрипты (сценарии) для запуска/остановки/перезапуска демонов находятся в каталоге /etc/rc.d. Чтобы запустить демона, наберите:

```
# /etc/rc.d/<имя_демона> start*
```

Чтобы остановить демона, наберите:

```
# /etc/rc.d/<имя_демона> stop
```

Чтобы перезапустить демона, наберите:

```
# /etc/rc.d/<имя_демона> restart
```

Для некоторых демонов (например, httpd) возможна перенастройка без остановки демона. Чтобы заставить демона во время его работы перечитать конфигурационный файл, наберите:

```
# /etc/rc.d/<имя_демона> reload
```

Для некоторых демонов возможны другие действия. Чтобы увидеть все доступные действия, наберите:

```
# /etc/rc.d/<имя_демона>
```

Автозапуск демонов при загрузке системы

Чтобы необходимые демоны запускались при старте системы, необходимо добавить их имена в файл /etc/rc.conf в секцию DAEMONS=():

DAEMONS=(syslog-ng network @httpd @mysqld @ntpd @acpid @snmpd @hal @sshd netfs crond)

Например, чтобы добавить демона динамической маршрутизации quagga, необходимо отредактировать эту строку следующим образом:

DAEMONS=(syslog-ng network @httpd @mysqld @ntpd @acpid @snmpd @hal @sshd netfs crond quagga)

Символ "@" перед именем демона означает, что система при загрузке не будет ждать завершения запуска этого демона, а перейдёт к запуску следующего (произойдёт фоновый запуск демона). Символ "!" перед именем демона означает, что этот демон не будет запускаться ни при каких обстоятельствах.

Утилита Midnight Commander

Данная утилита является консольным менеджером, ликвидирующим необходимость работы с командной строкой. Для того, чтобы запустить утилиту, запустите терминал и наберите «mc». Появится окно, представленное на рисунке 4.1.

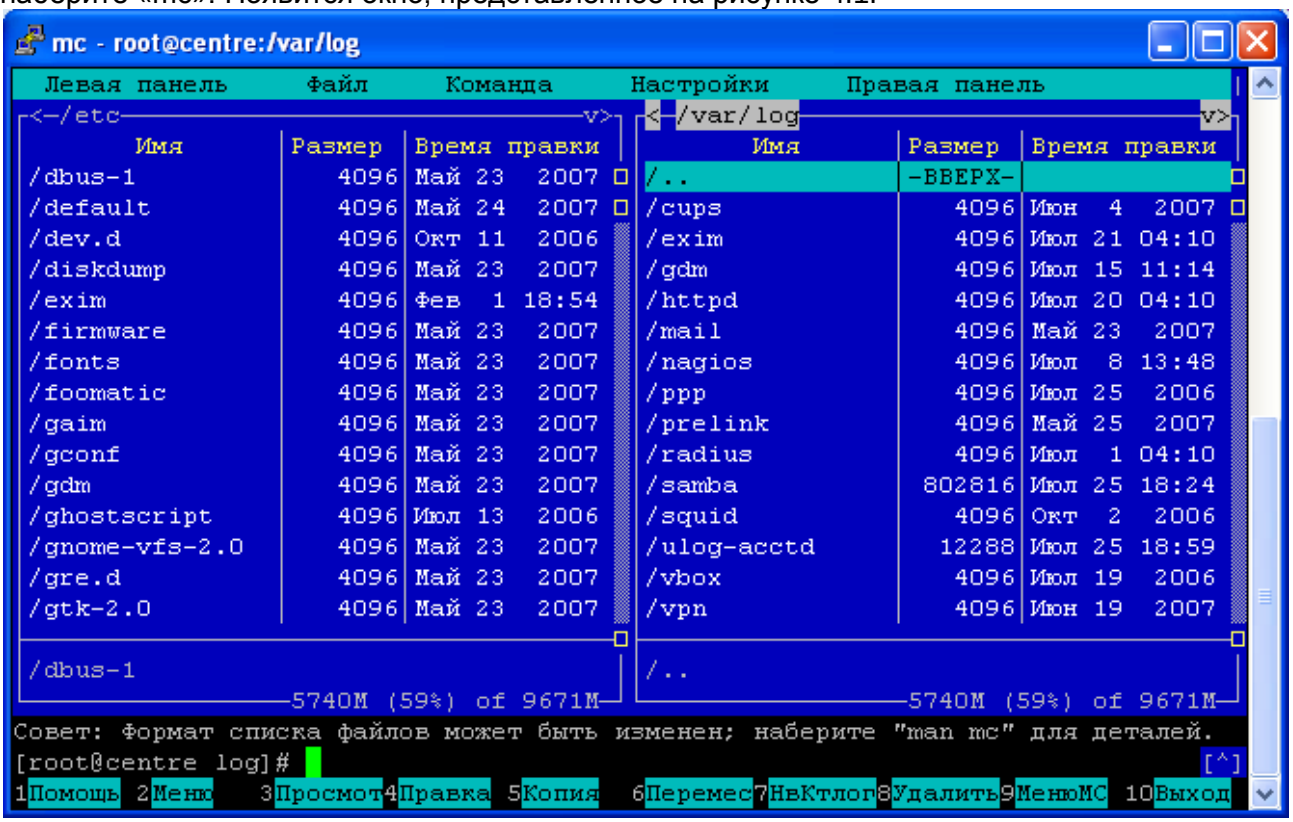


Рисунок 4.1.

Прелесть Midnight Commander заключается в том, что большинство «горячих» клавиш в нём совпадают с «горячими» клавишами в уже привычных для Вас менеджерах (Norton Commander, FAR).

2. Настройка сетевых параметров

Настройка сетевого интерфейса

Чтобы задать адрес сетевому интерфейсу, можно использовать команду ifconfig:

```
$ ifconfig <имя_интерфейса> <ip-адрес>
```

Для стандартных зарезервированных внутренних адресов сетевая маска и широковещательный адрес будут заданы автоматически. Для явного задания сетевой маски и широковещательного адреса можно использовать следующую команду:

```
$ ifconfig <имя_интерфейса> <ip-адрес> netmask
<сетевая_маска_записанная_октетами> broadcast
<широковещательный_адрес>
```

Например:

```
$ ifconfig eth0 10.10.10.10 netmask 255.255.255.0 broadcast
10.10.10.255
```

Для включения интерфейса (если он отключен) используется также команда ifconfig:

```
$ ifconfig <имя_интерфейса> up
```

Для отключения:

```
$ ifconfig <имя_интерфейса> down
```

Автонастройка сетевого интерфейса при запуске системы

Чтобы интерфейсам присваивались адреса (и прочие параметры) при старте системы, необходимо указать интерфейсы и параметры в файле /etc/rc.conf следующим образом:

```
<имя_интерфейса>="параметры_запуска"
```

Например, настройка интерфейса по протоколу DHCP:

```
eth0="dhcp"
```

Простое включение интерфейса, без дополнительной настройки:

```
eth0="eth0 up"
```

Присвоение адреса интерфейсу:

```
eth0="192.168.10.10"
```

Задание сетевой маски:

```
eth0="192.168.10.10 netmask 255.255.0.0"
```

Теперь эти интерфейсы необходимо перечислить в секции INTERFACES=():

```
INTERFACES=(eth0)
```

Маршруты настраиваются аналогично. Например, создание маршрута по умолчанию:

```
gateway="default gw 192.168.0.1"
```

Указываем его в секции ROUTES=():

```
ROUTES=(gateway)
```

Настройка DNS-клиента

Все настройки клиента системы разрешения имён находятся в файле «/etc/resolv.conf». Возможны следующие параметры:

- ✓ search <имя домена> – задаёт постфикс для неполностью определённых доменных имён (non-fqdn). Может встречаться только один раз;
- ✓ nameserver <адрес сервера> – задаёт адрес DNS-сервера. Можно указать сколь угодно много серверов, их опрос будет идти в порядке указания.

Настройка статических маршрутов

Для работы с таблицей маршрутизации ядра используется команда route. Команда route без параметров показывает текущую таблицу маршрутизации ядра (в том числе и маршруты, полученные от демонов динамической маршрутизации):

```
$ route
Destination   Gateway       Genmask        Flags  Metric  Ref  Use  Iface
192.168.10.0  *             255.255.255.0  U      0       0    0    eth0
169.254.0.0   *             255.255.0.0    U      0       0    0    eth0
default       server.oplab  0.0.0.0        UG     0       0    0    eth0
```

Здесь default – это маршрут по умолчанию. Символ «*» в колонке «Gateway» означает отсутствие шлюза для данного маршрута. Для отключения разрешения имен можно использовать ключ -n:

```
$ route –n
Destination   Gateway       Genmask        Flags  Metric  Ref  Use  Iface
192.168.10.0  0.0.0.0       255.255.255.0  U      0       0    0    eth0
169.254.0.0   0.0.0.0       255.255.0.0    U      0       0    0    eth0
0.0.0.0       192.168.10.1  0.0.0.0        UG     0       0    0    eth0
```

Здесь 0.0.0.0 – маршрут по умолчанию. 0.0.0.0 в колонке «Gateway» означает отсутствие шлюза для данного маршрута. Создание маршрутов при запуске системы рассмотрено выше.

Включение функции маршрутизации

Маршрутизация трафика между сетевыми интерфейсами – одна из возможностей ядра Linux. Управление всеми возможностями и настройками ядра осуществляется через файловую систему proc (обычно монтируется в каталог /proc, при этом настройки ядра оказываются в каталоге /proc/sys). Существует утилита, упрощающая управление настройками – sysctl. Её синтаксис следующий:

```
sysctl <путь до параметра от /proc/sys, каталоги разделяются
точкой>=<значение>
```

Посмотреть список параметров, доступных для изменения, можно, выполнив следующую команду:

```
sysctl –a
```

При этом также будут выведены и значения параметров.
За включение возможности маршрутизации отвечает параметр net.ipv4.ip_forward. Таким образом, чтобы включить маршрутизацию, выполните команду:

```
sysctl net.ipv4.ip_forward=1
```

Чтобы маршрутизация включалась при загрузке, добавьте в файл /etc/sysctl.conf следующую строку:

```
net.ipv4.ip_forward = 1
```

3. Утилиты мониторинга сети

Ниже приведён ряд утилит, использующихся в операционных системах семейства Linux для работы с сетью.

ping

Используется для проверки соединения с удаленным узлом. Утилита Ping использует пакеты эхо-запроса (echo request) и эхо-ответа (echo reply) протокола ICMP (Internet Control Message Protocol) для проверки доступности и работоспособности определенного узла TCP/IP. Действует посредством посылки ICMP пакетов и ожидания ответа в течение 1 секунды (значение по умолчанию). На экран выводится время в миллисекундах, затраченное на ожидание отклика.

Синтаксис командной строки:
ping IP-address или DNS-имя удаленного хоста

Пример:
```
ping 193.233.81.1
PING 193.233.81.1 (193.233.81.1): 56 data bytes
64 bytes from 193.233.81.1: icmp_seq=0 ttl=63 time=83.716 ms
64 bytes from 193.233.81.1: icmp_seq=6 ttl=63 time=1.949 ms
64 bytes from 193.233.81.1: icmp_seq=7 ttl=63 time=31.293 ms
^C
--- 193.233.81.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.949/18.160/83.716/26.597 ms
```

В поле time указывается, за какое время (в миллисекундах) посланный пакет доходит до удаленного узла и возвращается на ваш узел. Поле ttl указывает время жизни пакета. После приостановки выполнения утилиты она выдает статистику: сколько пакетов послано, сколько получено и утеряно, время задержки (минимальное, среднее и максимальное).

Вместо IP-адреса хоста может быть указан широковещательный адрес. В этом случае результатом работы утилиты будет список узлов, откликнувшихся на запрос. Откликнутся все узлы сети, активные в настоящий момент и имеющие IP-адрес, соответствующий указанной маске.

tcpdump

Одним из мощных средств анализа всей сетевой активности является утилита tcpdump. Она переводит сетевой интерфейс в режим приема всех пакетов (promiscuous) и выводит информацию на экран. В Linux такое переключение возможно только для суперпользователя, то есть для полноценного использования tcpdump необходимо зарегистрироваться под пользователем root. На других системах требования немного другие и они представлены в документации к tcpdump.

Синтаксис командной строки tcpdump следующий:
tcpdump [<опции...>] <выражение фильтра>

Наиболее используемые опции tcpdump:
-c <число пакетов>
Сколько пакетов считать. После считывания последнего пакета, tcpdump завершает работу. Например, *«tcpdump -c 50»* считывает только 50 пакетов

-i <интерфейс>

На каком интерфейсе осуществлять съём информации. Например, *«tcpdump -i eth1»* осуществляет съём данных на втором ethernet-интерфейсе eth1. Данная опция полезна, когда на используемом компьютере имеются 2 и более сетевых карт.

-s <число байт>
Сколько байт начала каждого пакета считывать. По умолчанию используется значение 68 байт. Этого должно хватать для расшифровки данных из заголовков пакетов большинства протоколов, однако может возникнуть необходимость использовать большее значение.

-w <имя файла>
Записывать содержимое пакетов в файл. Полезно для съёма информации в неурочное время или при больших объёмах передаваемой информации.

-r <имя файла>
Анализ информации записанной с помощью опции –w.

<выражение фильтра> позволяет отсеивать явно ненужную информацию, захватывая лишь пакеты, которые удовлетворяют условиям этого выражения. Полный синтаксис выражений можно найти в документации по tcpdump.

Пример:
tcpdump host 192.168.3.255
tcpdump: listening on eth0
12:23:19.493594 809-01.comp.chelcom.ru.netbios-ns>192.168.3.255.netbios-ns:
>>NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST

На экран выведены имя «слушающего» сетевого интерфейса, время с точностью до микросекунд, имя отправляющего узла и имя назначения.

arp

Для проверки ARP-таблиц, содержащих соответствие между IP-адресом и MAC-адресом, используется утилита ARP. В некоторых случаях бывает полезно просмотреть или изменить содержание ARP-таблицы, например, когда вы подозреваете, что двойной адрес является причиной сетевой неустойчивости. Одна из проблем, которая может потребовать, чтобы вы вручную добавили IP-адрес к ARP-таблице, это когда по некоторым причинам ARP-запросы для удаленного хоста не доходят, например, когда есть сбой ARP-драйвера, или имеется другой хост в сети, который ошибочно опознает себя с IP-адресом другого хоста. Твердая установка IP-адреса в ARP-таблице также является мерой защиты себя от хостов в вашем Ethernet, которые выдают себя за кого-то другого.

Синтаксис командной строки:
```
arp [-v] [-t hwtype] -a [hostname]
arp [-v] [-t hwtype] -s hostname hwaddr
arp [-v] -d hostname [hwaddr]
```

Аргумент hostname может быть как именем, так и IP адресом. Первая строка отображает ARP-запись для IP-адреса, указанного хоста или всех известных хостов, если hostname не задается.

Пример:
```
arp -a
IP address       HW type                  HW address
172.16.1.3       10Mbps Ethernet       00:00:C0:5A:42:C1
172.16.1.2       10Mbps Ethernet       00:00:C0:90:B3:42
```

```
172.16.2.4       10Mbps Ethernet          00:00:C0:04:69:AA
```

При использовании опции -t вы увидите информацию только о том типе аппаратных средств, который укажете. Это могут быть:

- *ether*
- *ax25*
- *pronet (Ethernet 10Mbps)*
- *AMPR AX.25*
- *IEEE 802.5*

Опция -s используется, чтобы добавить Ethernet-адрес хоста к ARP-таблицам.

Аргумент hwaddr определяет адрес аппаратных средств, который по умолчанию предполагается Ethernet-адресом, указанным как шесть шестнадцатеричных байт, разделяемых двоеточиями. Вы можете также устанавливать адреса для других типов аппаратных средств, используя опцию -t.

Вызов arp с использованием ключа -d удаляет все ARP-записи, касающиеся данного хоста. Это может быть необходимо, чтобы вынудить интерфейс повторно получить Ethernet-адрес для данного IP. Это полезно, когда переконфигурированная система имеет неправильную ARP-информацию.

4. Система безопасности ОС Linux

Пользователи

Linux — многопользовательская ОС. Информация об учётных записях пользователей хранится в файлах /etc/passwd, /etc/shadow, /etc/group, /etc/ghadow.

/etc/passwd

В этом файле перечислены учётные записи. Ранее в этом же фале содержался и пароль в зашифрованном виде, однако так как этот файл должен быть доступен для чтения любому пользователю, впоследствии была разработана система shadow и пароль с прочей информацией о безопасности били перенесены в /etc/shadow.

Учётные записи хранятся по одной в строке. Запись состоит из 7 полей, разделённых символом «:». Значение полей:

1. Регистрационное имя (логин). Регистрационные имена должны быть уникальными и представлять собой строки не длиннее 32-х символов (любые, кроме двоеточия и символа новой строки). В некоторых старых версиях UNIX существует ограничение длины в 8 символов, оно же будет действовать при использовании административной базы данных NIS.
2. пароль. Значение «x» указывает на то, что пароль хранится в /etc/shadow. Пустое значение указывает на отсутствие пароля.
3. Идентификатор пользователя (UID). Идентификатор пользователя — это число от 0 до $2^{32}-1$. В старых системах (Minix, ранние версии BSD) оно может быть не более 32 767. Пользователь с идентификатором 0 (обычно root) называется суперпользователем и имеет право на выполнение любых операций в системе. Принято соглашение о выделении «специальным» пользователям (bin, daemon), назначение которых — только запуск определённых программ, маленьких идентификаторов (<100 или, в некоторых дистрибутивах Linux, <500). В системе могут существовать несколько пользователей с одним идентификатором. Это нередко используется взломщиками, когда они после проникновения в систему создают себе учётную запись с UID=0. В результате они выглядят как обычные пользователи, но на самом деле имеют права root.
4. GID основной группы.
5. Поле GECOS.
6. Домашний каталог.
7. Командный интерпретатор.

Поле GECOS предназначено для хранения дополнительных сведений о пользователе. Не имеет чётко определённого синтаксиса, однако по умолчанию применяется синтаксис, воспринимаемый командой finger — четыре поля, разделённых символом «,»:

- ✓ полное имя;
- ✓ номер комнаты;
- ✓ рабочий телефон;
- ✓ домашний телефон.

Домашний каталог — это начальный рабочий каталог пользователя. Значение этого параметра будет присвоено переменной $HOME.

Если командный интерпретатор не указан, то по умолчанию используется /bin/sh. Значение последнего параметра записи о пользователе будет присвоено переменной $SHELL.

/etc/shadow

Для повышения безопасности секретные данные учётной записи (пароль, сведения об устаревании пароля и т.д.) были перенесены в файл, доступный для чтения только суперпользователю (UID=0). Каждая запись о пользователе из /etc/passwd, имеющая «x»

в поле пароля, обязана иметь соответствующую запись в /etc/shadow, иначе считается некорректной.
Записи безопасности хранятся по одной в строке. Каждая строка состоит из 9 полей:
- ✓ имя пользователя;
- ✓ шифрованный пароль;
- ✓ число дней с момента последнего изменения пароля, начиная с 1 января 1970 года;
- ✓ число дней, перед тем как пароль может быть изменён;
- ✓ число дней, после которых пароль должен быть изменён;
- ✓ число дней, за сколько пользователя начнут предупреждать, что пароль устаревает;
- ✓ число дней после устаревания пароля для блокировки учётной записи;
- ✓ день, отсчитанный от 1 января 1970 года, когда учётная запись была заблокирована;
- ✓ зарезервированное поле.

/etc/group
Пользователи объединяются в группы. Пользователи могут входить в несколько групп. В группе может присутствовать только один пользователь. Группы предназначены для более гибкого разделения прав доступа к объектам ФС.
Записи о группах хранятся по одной в строке. Каждая строка состоит из 4 полей:
- ✓ имя группы;
- ✓ групповой пароль. Пустое поле означает отсутствующий пароль;
- ✓ GID;
- ✓ имена пользователей, разделённые запятыми, являющихся членами группы.

/etc/gshadow
Аналог /etc/shadow, но для записей о группах.
Записи безопасности хранятся по одной в строке. Каждая строка состоит из 4 полей:
- ✓ имя группы;
- ✓ шифрованный групповой пароль. Заменяет соответствующее поле из /etc/group;
- ✓ имена пользователей, разделённые запятыми, являющихся администраторами группы;
- ✓ имена пользователей, разделённые запятыми, являющихся членами группы.

Работа с перечисленными файлами осуществляется следующими программами:
- ✓ vipw — запускает текстовый редактор, указанный в переменной среды EDITOR, загружая в него копию файла /etc/passwd. После закрытия редактора переносит временную копию в сам файл. Не позволяет двум пользователям выполнять редактирование одновременно;
- ✓ useradd — создаёт новую учётную запись;
- ✓ usermod — изменяет данные учётной записи;
- ✓ userdel — удаляет существующую учётную запись;
- ✓ chfn — изменяет поле GECOS;
- ✓ chsh — устанавливает новый командный интерпретатор;
- ✓ passwd — задаёт новый пароль пользователя.

Источники информации об учётных записях

Сведения о пользователях могут храниться не только в перечисленных файлах (однако информацию о суперпользователе настоятельно рекомендуется хранить в файле, иначе повышается риск получить систему, к которой нет локального доступа). За выбор источника учётных записей отвечает файл /etc/nsswitch.conf. Рассмотрим пример такого файла:

```
passwd: files
group: files
shadow: files

publickey: files
```

```
hosts: files dns
networks: files

protocols: db files
services: db files
ethers: db files
rpc: db files

netgroup: db files
```

Часть, идущая до двоеточия, указывает на тип информации, извлекаемой из источника:
- ✓ passwd = /etc/passwd
- ✓ group = /etc/group
- ✓ shadow = /etc/shadow
- ✓ hosts = /etc/hosts
- ✓ networks = /etc/networks
- ✓ protocols = /etc/services

После двоеточия указывается модуль или модули NSS, используемые для извлечения информации из соответствующих источников, а также правила реакции на некоторое событие. Примеры модулей:
- ✓ files — файл
- ✓ ldap — иерархическая база данных LDAP (требуется установка nss_ldap)
- ✓ nis — база сетевой информации NIS (Network Information Service)
- ✓ dns — обращение к серверу DNS

Существует также множество прочих модулей. Например, существует модуль, позволяющий объединять группы в группы.

События:
- ✓ SUCCESS — запрошенная запись была найдена в базе данных
- ✓ UNAVAIL — источник не сконфигурирован в данной системе или произошел внутренний сбой
- ✓ NOTFOUND — источник сообщил, что соответствующей записи нет
- ✓ TRYAGAIN — источник занят или не отвечает; возможно, ответ будет получен при следующих попытках

Действия:
- ✓ continue — обратиться к следующему источнику в списке
- ✓ return — завершить поиск

Для события TRYAGAIN доступны также следующие действия:
- ✓ forever — Повторять обращения к текущему источнику, пока не будет получен ответ
- ✓ n — повторить обращения к текущему источнику еще n раз, где n - целое число от 0 до MAX_INT (т.е., 2,14 миллиарда). После n попыток перейти к следующему источнику

Пример записи, использующей реакцию на события:

```
passwd: ldap [TRYAGAIN=10] files
```

Данная запись указывает, что при недоступности источника ldap следует повторить запрос ещё 10 раз, а потом перейти к источнику files. Для проверки работоспособности системы извлечения учётных записей можно воспользоваться командами getent passwd и getent group. Пример:

```
#getent passwd 0
root:x:0:0:root:/root:/bin/bash
#getent group 0
root::0:root
```

```
#getent hosts 127.0.0.1
127.0.0.1       localhost.localdomain localhost
```

Система аутентификации PAM

Классически аутентификация производится программой /bin/login посредством извлечения данных из источников, указанных в /etc/nsswitch.conf, с помощью системных функций getpwnam и getspnam и сравнения полученных данных с предоставленными пользователем. Для универсализации процесса аутентификации была внедрена система аутентификации PAM. Библиотека PAM является обобщённым API для служб, связанных с аутентификацией, которые позволяют системному администратору добавлять новые методы аутентификации простой установкой новых модулей PAM, и изменять политику аутентификации посредством редактирования конфигурационных файлов. Таким образом, суть PAM в добавлении дополнительных функций, позволяющих выполнять аутентификацию. Соответственно, для того, чтобы использовать PAM, некоторая программа должна использовать эти функции, то есть должны вноситься изменения в исходный код. В большинстве дистрибутивов поддержка PAM включена везде, где только возможно (в Slackware нет поддержки PAM до сих пор по идейным причинам).
PAM API предоставляет шесть различных примитивов для аутентификации, сгруппированных в четыре подсистемы, каждая из которых описывается ниже.

- ✓ auth — аутентификация. Эта подсистема, собственно говоря, реализует аутентификацию и выяснение полномочий учётной записи. Она предоставляет два примитива:
  - функция pam_authenticate(3) аутентифицирует аппликанта, обычно запрашивая аутентификационный ключ и сравнивая его со значением, хранящимся в базе данных или получаемым от сервера аутентификации;
  - функция pam_setcred(3) устанавливает полномочия учётной записи, такие, как идентификатор пользователя, членство в группах и ограничения на использование ресурсов.

- ✓ account — управление учётной записью. Эта подсистема обрабатывает вопросы доступности учетной записи, не связанные с аутентификацией, такие, как ограничения в доступе на основе времени суток или загрузки сервера. Он предоставляет единственный примитив:
  - функция pam_acct_mgmt(3) проверяет, доступна ли запрашиваемая учётная запись.

- ✓ session — управление сеансом. Эта подсистема отрабатывает задачи, связанные с установлением и закрытием сеанса, такие, как учет входов пользователей. Она предоставляет два примитива:
  - функция pam_open_session(3) выполняет действия, связанные с установлением сеанса: добавление записей в базы данных utmp и wtmp, запуск агента SSH и так далее;
  - функция pam_close_session(3) выполняет действия, связанные с закрытием сеанса: добавление записей в базы данных utmp и wtmp, завершение работы агента SSH и так далее.

- ✓ password — управление паролем. Эта подсистема используется для изменения ключа аутентификации, связанного с учетной записью, по причине истечения его срока действия или желания пользователя изменить его. Она предоставляет единственный примитив:
  - функция pam_chauthtok(3) изменяет ключ аутентификации, дополнительно проверяя, что он труден для подбора, не использовался ранее и так далее.

Модули являются центральной концепцией в PAM. Модуль PAM представляет собой самодостаточный кусок программного кода, который реализует примитивы одной или большего количества подсистем одного конкретного механизма; к возможным

механизмам для подсистемы аутентификации, к примеру, относятся базы данных паролей UNIX, системы NIS, LDAP или Radius.

Когда сервер инициирует PAM-транзакцию, библиотека PAM пытается загрузить политику для службы, указанной при вызове функции pam_start(3). Политика определяет, как должны обрабатываться запросы на аутентификацию, и задаётся в конфигурационном файле. Это составляет другую основополагающую концепцию PAM: возможность администратору настраивать политику безопасности системы (в самом широком её понимании) простым редактированием текстового файла.
Политика состоит из четырёх цепочек, по одной на каждый из методов PAM. Каждое звено представляет собой последовательность конфигурационных утверждений, задающих вызываемый модуль, некоторые (необязательные) параметры для передачи в модуль, и управляющий флаг, описывающий, как интерпретировать возвращаемый из модуля код.
Понимание смысла управляющего флага необходимо для понимания конфигурационных файлов PAM. Существуют четыре различных управляющих флага:

- ✓ binding — если модуль отработал успешно, и ни один из предыдущих модулей в цепочке не сработал отрицательно, то цепочка прерывается, а запрос подтверждается. Если же модуль отработает неудачно, то выполняется оставшаяся часть цепочки, однако запрос отвергается;
- ✓ required — если модуль возвратил положительный ответ, выполняется оставшаяся часть цепочки, запрос удовлетворяется, если никакой другой модуль не отработает отрицательно. Если же модуль возвратит отрицательный ответ, остаток цепочки тоже отрабатывается, но запрос отвергается;
- ✓ requisite — если модуль возвращает положительный ответ, выполняется оставшаяся часть цепочки, запрос удовлетворяется, если никакой другой модуль не отработает отрицательно. Если же модуль отрабатывает отрицательно, то отработка цепочки немедленно прекращается, а запрос отвергается;
- ✓ sufficient — если модуль возвратит положительный ответ, и ни один из предыдущих модулей в цепочке на отработал отрицательно, то отработка цепочки немедленно прекращается, а запрос удовлетворяется. Если модуль отработал отрицательно, то результат игнорируется и цепочка отрабатывается дальше;
- ✓ optional — модуль отрабатывается, но результат выполнения игнорируется. Если все модули в цепочке помечены как optional, то удовлетворяться будут все запросы.

Когда сервер вызывает один из шести PAM-примитивов, PAM запрашивает цепочку подсистемы, к которой принадлежит примитив, и запускает каждый модуль, перечисленный в цепочке в порядке их перечисления, пока список не будет исчерпан либо не будет определено, что дальнейшей обработки не нужно (по причине достижение модуля, вернувшего положительный ответ при условии binding или sufficient, либо отрицательный с условием requisite). Запрос подтверждается, если только был вызван по крайней мере один модуль, и все неопциональные модули вернули положительный ответ. Допустимо перечислять один и тот же модуль несколько раз в одной цепочке. К примеру, модуль, просматривающий имена и пароли пользователя в сервере каталога может быть вызван несколько раз с различными параметрами, задающими различные серверы каталогов для связи. PAM считает различные появления одного модуля в той же самой цепочке разными и не связанными модулями.

Политики PAM содержатся в каталоге /etc/pam.d. Обратите внимание, что для многих программ уже есть готовые политики. Рассмотрим типичную политику.

```
# Строка, начинающаяся символом #, считается комментарием
# Строка, начинающаяся с #%, является спецификатором версии
# формата файла
#%PAM-1.0
# требуется аутентификация через файл /etc/passwd и пр.
auth            required        pam_unix.so
# модуль, запрещающий вход не суперпользователям, если
```

```
# существует файл /etc/nologin. Его содержимое показывается
# пользователю
auth            required        pam_nologin.so
# данные пользователя берутся из /etc/passwd и пр.
account         required        pam_unix.so
# пароль сверяется с /etc/shadow
password        required        pam_unix.so
# ограничения на сеанс накладываются /etc/passwd и
# /etc/secutiry/limits.conf (определяет границы использования
# пользователем ресурсов системы)
session         required        pam_unix.so
session         required        pam_limits.so

# Изменим эту политику, добавив необязательную аутентификацию через
# LDAP (требуется модуль pam_ldap)
#%PAM-1.0
auth            sufficient      pam_ldap.so
auth            required        pam_unix.so use_first_pass
auth            required        pam_nologin.so
account         sufficient      pam_ldap.so
account         required        pam_unix.so
password        sufficient      pam_ldap.so
password        required        pam_unix.so
session         required        pam_unix.so
session         required        pam_limits.so
```

Параметр use_first_pass добавлен для предотвращения повторного запроса пароля при неудачной попытке аутентификации через LDAP.

5. Сетевая файловая система NFS

NFS – один из протоколов сетевой распределённой файловой системы (другие примеры – afs, sfs, cifs). Изначально разработан Sun Microsystems, с версии 4 разработка ведётся в IETF. Является стандартом де-факто для unix-подобных систем. Стандарты RFC – 1094, 1790, 1813, 3050.

Возможности:

1. В основном предназначена для организации файлового хранилища либо сервера бездисковой загрузки терминалов в локальной сети. Для передачи файлов через Интернет имеет смысл поискать другие протоколы.
2. Возможна работа по TCP или UDP.
3. Сохраняет атрибуты безопасности объектов экспортируемой файловой системы.
4. Позволяет варьировать размеры буферов чтения и записи.
5. При импортировании ветви (для unix-подобных ОС) становится частью дерева корневой файловой системы.
6. В ранних версиях разделение доступа реализуется на основе IP-адреса.
7. В версии 4 поддерживается авторизация через билеты Kerberos.
8. Работа осуществляется через RPC (Remote Procedure Call).

Следует отметить, что протокол NFS (а, точнее, передаваемые по сети объекты) создаёт ощутимую нагрузку на сеть, особенно в случае экспорта множества объектов малого информационного объёма (и при использовании UDP в качестве транспортного протокола). Для использования NFS-каталогов в качестве корневого или домашнего следует прокладывать каналы с достаточно высокой пропускной способностью (гигабитные Ethernet-линии или оптоволокно) либо выделять отдельный канал только пор NFS-трафик (однако, можно обойтись и без этого. Например, сеть на витой паре категории 5E, 100 Мбит/с, позволяет достаточно комфортно работать 14 пользователям с импортированными с сервера домашними каталогами, содержащими программные проекты пользователей, а также осуществлять синхронную запись в домашние каталоги и в общее файловое хранилище).

В Linux для поддержки NFS используются следующие объекты:

- ✓ демон nfsd (являющийся на самом деле потоком ядра), отвечающий за передачу файлов по протоколу NFS;
- ✓ файл /etc/exports, описывающий экспортируемые ветви;
- ✓ демон mountd, отвечающий за обслуживание удалённых запросов на подключение;
- ✓ демон statd, реализующий протокол NSM (Network Status Monitor) RPC, используемый для сохранения состояния блокировок между перезагрузками клиента или сервера;
- ✓ демон lockd, реализующий работу с файловыми блокировками;
- ✓ демон rpcbind, обеспечивающий преобразование номера порта DARPA в номер порта RPC и обратно для осуществления RPC-запросов. Является основой всех RPC-зависимых сервисов и должен запускаться раньше их;
- ✓ демон idmapd, обеспечивающий преобразование NFSv4 ID в имя пользователя. Требуется для работы протокола NFSv4;
- ✓ программа exportfs, заставляющая nfsd и mountd перечитать файл /etc/exports.

Набор демонов для запуска на клиентской и серверной стороне настраивается в файлах /etc/conf.d/nfs-common и /etc/conf.d/nfs-server соответственно.

В Arch Linux для запуска NFS-сервера требуется выполнить следующие команды (порядок важен, так как демоны зависимы друг от друга):

```
1./etc/rc.d/rpcbind start          # запуск rpcbind
2./etc/rc.d/nfs-common start       # запуск клиентского набора демонов
3./etc/rc.d/nfs-server start # запуск серверного набора демонов
```

Поток ядра, естественно, запускать не требуется.

Расмотрим синтаксис файла /etc/exports:

```
ветка_файловой_системы  IP-адрес_клиента/сетевая_маска(опции_экспортирования)
```

Внимание! Отсутствие пробела перед скобками, заключающими в себе опции экспортирования, является принципиальным. При наличии этого пробела последующие опции будут применяться для *всех* клиентов, запросивших разрешения на импорт данной ветки. Импортирование ветки, соответственно, будет разрешено для *всех* клиентов.

Сетевая маска может записываться в полном или сокращённом формате. Вместо адреса клиента можно не указывать ничего или указать *, тогда монтирование будет разрешено любому клиенту. Можно указать несколько клиентов с различными опциями монтирования (например, это полезно, если нужно обеспечить для любого пользователя только строго непривилегированный или анонимный доступ к файловой системе, а для компьютера администратора привилегии сохранить). При этом все клиенты должны быть записаны в одной строке, соответствующей экспортируемой ветке файловой системы. Вместо IP-адреса клиента может указываться DNS-имя, в имени могут использоваться маски (например, *.example.org). Адрес клиента может также являться NIS[1]-группой, в этом случае он записывается как @group.

Рассмотрим доступные опции экспортирования:

- ✓ secure — требует, чтобы запросы приходили с порта, имеющего привилегированный номер (менее 1024). Включена по умолчанию. Для отключения используйте опцию insecure;
- ✓ rw — разрешает запись в экспортируемую файловую систему. По умолчанию выключена. Для однозначного запрещения любых изменений можно использовать опцию ro;
- ✓ async — разрешает нарушать протокол NFS и отвечать на запросы до того, как изменения будут внесены в файловую систему. Повышает производительность, однако и увеличивает риск потери данных при сбое питания или самого сервера;
- ✓ sync — предписывает отвечать на запросы строго после внесения изменений. Включена по умолчанию в релизах nfs-utils до 1.0.0 включительно, для использования асинхронного обмена требуется явно указать опцию async (что, однако не рекомендуется разрпаботчиками[2]);
- ✓ no_wdelay — не имеет смысла, если задана опция async. Обычно NFS-сервер слегка задерживает отправку изменений на диск, если ожидает, что в скором времени может произойти или уже выполняется другой запрос на запись, связанный с обработанным, что позволяет производить вносить несколько изменений на диск за один раз, что увеличивает производительность. Если NFS-сервер в основном обрабатывает несвязанные маленькие запросы, то такое поведение замедляет работу. Эта опция позволяет отключить эту особенность. Для явного включения задержки записи используется опция wdelay;
- ✓ nohide — обычно, если сервер экспортирует 2 файловые системы, одна из которых монтируется в другую, клиент должен явно смонтировать обе ФС для получения доступа к ним. Если он смонтирует только родительскую ФС, то получит постой каталог. Такая ФС называется «скрытой» («hidden»). Установка данной опции делает экспортируемую ФС нескрытой, позволяя клиенту прозрачно перемещаться с родительской ФС на дочернюю. Однако, некоторые NFS-клиенты не справляются с данной ситуацией, так как, например, появляется возможность совпадения номеров inode у двух различных файлов. Поэтому данная опция эффективна только для ФС, экспортируемых для одного клиента. Может ббыть явно отменена с помощью опции hide;

---

1 NIS (Network Information Services) — устаревшая сетевая служба для хранения учётной информации о пользователяхсети (бывший Yellow Pages). Является низкопроизводительной и небезопасной, а также давно неразвиваемой. В большинстве мест, где NIS действительно была нужна, она заменена на LDAP, поэтому про NIS-группы можно забыть

2 Разработчики именуют асинхронный режим не иначе, как «ошибка администратора»

- ✓ crossmnt — выполняет действия, подобные опции nohide, но позволяет клиентам перемещаться с ФС, отмеченной как crossmnt, на экспортированную ФС, смонтированную на помеченную. Например, если ФС B смонтирована на ФС A, то установка опции crossmnt на A идентична установке опции nohide на B;
- ✓ no_subtree_check — отменяет проверку поддерева, что нарушает требования безопасности, однако может улучшить надёжность в некоторых случаях. В случае экспорта подкаталога ФС каждый NFS-запрос должен быть проверен на принадлежность не только конкретной файловой системе, но и на принадлежность экспортированному поддереву (что сложнее). Такая проверка может быть включена опцией subtree_check. Такая проверка обычно выполняется, чтобы убедиться, что к файлы внутри каталога, доступ к которому имеет только root, находятся на ФС, экспортированной с опцией no_root_squash. При экспорте домашнего каталога данную опцию следует использовать (так как производятся частые переименования и экспорт обычно производится от корня). Для каталогов только для чтения (например, /var, /usr), для которых может производиться экспорт подкаталогов, можно использовать subtree_check. По умолчанию используется subtree_check. С релиза nfs-utils 1.1.0 по умолчанию используется no_subtree_check, так как subtree_check, по мнению разработчиков, вызывает больше проблем, чем приносит пользы;
- ✓ insecure_locks, no_auth_nlm (синонимы) — заставляет NFS-сервер не требовать аутентификацию для выполнения блокировки (для запросов, использующих NLM). Обычно NFS-сервер требует, чтобы запрос на блокировку содержал пользовательские учётные данные для проверки клиентского доступа к файлам. При использовании этой опции такое требование не предъявляется. Ранние реализации NFS-клиентов не отправляли учётные данные с запросами на блокировку, да и многие современные клиенты основа ны на старых реализациях. Используйте этот флаг, чтобы разрешить блокирование только файлов, доступных на чтение для всех. Для отключения запроса учётных данных используйте secure_locks или auth_nlm;
- ✓ no_acl — на некоторых специально модифицированных ядрах при экспорте ФС данная опция предписывает не передавать ACL клиенту, таким образом, клиент увидит только подмножество актуальных разрешений на ФС. Эта опция безопасна для NFSv2-клиентов и старых NFSv3-клиентов, которые выполняют принятие решения о доступе локально. В текущих реализациях NFSv3-клиентов для этого используется RPC ACCESS. Опция имеет смысл только на поддерживающих её ядрах и только для ФС с поддержкой ACL. По умолчанию производится экспорт ФС вместе с ACL;
- ✓ mountpoint=path, mp (синонимы) — разрешает экспорт только тех каталогов, которые были успешно смонтированы. Если путь не указан, то экспортируемая ФС также длдна являться и точкой монтирования. Если это не так, тол экспортирование не производится. Это позволяет убедиться, что каталог внутри точки монтирования никогда не будет экспортирован по ошибке или в результате аварии (например, если ФС не смонтировалась из-за ошибки диска). Если задан путь, то он должен являться точкой монтирования для экспортируемой ФС;
- ✓ fsid=num|root|uuid — NFS должна иметь возможность идентифицировать каждую экспортируемую ФС. Обычно используется UUID (если у ФС имеется таковой) или номер устройства, содержащего ФС. Так как не все ФС содержатся на устройствах и не все ФС имеют UUID, иногда необходимо явно указать NFS, как идентифицировать ФС. Для NFSv4 существует ФС, являющаяся корневой для всех экспортируемых ФС. Она обозначается как fsid=root или fsid=0. Прочие ФС могут быть определены маленьким целым числом или UUID длиной 32 бита, который должен состоять только из шестнадцатиричных символов и иметь определённый формат.

Разграничение доступа выполняется на стороне сервера по установленным атрибутам (POSIX-разрешения, ACL, uid, gid). Соответственно, для этого у сервера и клиентов должна быть единая база пользователей, что не всегда возможно. Иногда возникает необходимость запретить доступ пользователю root на клиентской машине с полными правами на экспортируемую ФС. По умолчанию производится смена uid=0 на другой, анонимный (nobody). Это называется root squashing. NFS выбирает анонимный UID,

равный 65534 (может быть изменено). Также есть возможность смены любого UID на анонимный.

Для управления доступом используются следующие опции:
- ✓ root_squash — заменять в запросах UID=0 на анонимный;
- ✓ no_root_squash — отключить root squashing. Полезно для бездисковых клиентов;
- ✓ all_squash — заменять в запросах все UID на анонимный. Полезно для публично экспортируемых каталогов. По умолчанию используется обратная опция no_all_squash;
- ✓ anonuid и anongid — задают анонимные UID и GID. Полезно для тех клиентов, запросы от которых требуется выполнять от отпределённого UID и GID (например, при экспорте домашнего каталога пользователя при использован ии неодинаковой базы пользователей клиентом и сервером).

Рассмотрим пример файла /etc/exports:

```
/               master(rw) trusty(rw,no_root_squash)
/projects       proj*.local.domain(rw)
```

Здесь экспортируются две ФС — корневая и /projects. Корневая ФС экспортируется для клиентов master (полный доступ) и trusty (полный доступ, разрешён доступ с правами root). ФС /projects экспортируется для клиентов, имя которых совпадает с маской proj*.local.domain, с полным доступом.

Для того, чтобы файловые системы стали досутупны извне, после редактирования файла exports необходимо мыполнить команду exportfs -arv, где:
- ✓ exportfs — программа, управляющая экспортом ФС
- ✓ -a — экспортировать все каталоги
- ✓ -r — выполнить переэкспортирование всех каталогов. Удаляет устаревшие записи о клиентах из таблиц ядра
- ✓ -v — выводить подробную информацию о процессе экспортирования

Теперь экспортируемую ФС надо примонтировать на клиенте. Для этого используется команда mount следующим образом:

```
# mount fs_server:/exported/fs /local/mount/point –t nfs –o <опции_
  монтирования>
```

Для монтирования NFSv4 применяется ключ -t nfs4. Расмотрим доступные для всех версий NFS опции монтирования:
- ✓ soft / hard — определяет поведение NFS-клиента после окончания запроса по тайм-ауту. Если опции не указаны или указана опция hard, то запрос повторяется. Если указана опция soft, то клиент отменяет запрос и возвращает ошибку вызывающему приложению. ВНИМАНИЕ! Soft-тайм-аут может вызвать повреждение данных. Поэтому использование опции soft оправдано только тогда, когда отзывчивость клиента важнее целостности данных. Использование протокола TCP для передачи данных или увеличение значения опции retrans позволяет сократить риск повреждения данных
- ✓ timeo=n — время ожидания ответа на запрос в десятых долях секунды. Если опция не указана, то повторные попытки выполнения запроса предпринимаются каждые 60 секунд для TCP. Для UDP клиент использует адаптивный алгоритм для определения подходящей величины тайм-аута для часто используемых запросов (например, READ или WRITE), а опция timeo используется для редких запросов (например, FSINFO. По умолчанию — 1,1 секунда). После каждой передачи клиент удваивает тайм-аут до достижения порогового значения 60 секунд
- ✓ retrans=n — число попыток выполнить запрос до перехода к следующему действию. По умолчанию 3. Клиент генерирует сообщение «сервер не отвечает» после исчерпания попыток retrans, а потом приступает к выполнению заданной политики

реагирования на тайм-аут (soft или hard)

- ✓ rsize=n — максимальное количество байт в каждом сетевом запросе READ, которое может принять NFS-клиент. Максимальный размер для Linux-клиента — 1048576 байт (1 МБ). Параметр является положительных целым, кратным 1024. Значения, меньшие 1024, заменяются на 4096. Значения, большие 1048576, заменяются на 1048576. В случае указания числа, не кратного 1024, производится округление с недостатком. Если параметр не указан, то клиент и сервер согласуют значение этого параметра
- ✓ wsize=n — максимальное количество байт в каждом сетевом запросе WRITE, которе NFS-клиент может отправить. Максимальное значение — 1048576. Все параметры и замечания для rsize справедливы и для wsize
- ✓ ac / noac — определяет, может ли клиент кешировать файловые атрибуты. По умолчанию кеширование выполняется. Если кеширование не выполняется, то приложения, работающие с файлами на NFS-разделе, быстрее узнают о произошедших изменениях. Опция noac, кроме запрета не кеширование атрибутов, делает процесс записи синхронным. Использование noac даёт большой проигрыш в производительности, поэтому вместо отмены кеширования разработчики предлагают использовать файловые блокировки
- ✓ acregmin=n — минимальное время (в секундах), на которое клиент кеширует файловые атрибуты. По умолчанию — 3
- ✓ acregmax=n — максимальное время (в секундах), на которое клиент кеширует файловые атрибуты. По умолчанию — 60
- ✓ acdirmin=n — минимальное время (в секундах), на которое клиент кеширует атрибуты каталогов. По умолчанию — 30
- ✓ acdirmax=n — максимальное время (в секундах), на которое клиент кеширует атрибуты каталогов. По умолчанию — 60
- ✓ actimeo=n — использование этой опции устанавливает одинаковые значения параметров для acregmin, acregmax, acdirmin и acdirmax. Если эта опция не задана, то используются умолчания, описанные выше
- ✓ bg / fg — определяет, как mount реагирует на неудачную попытку монтирования экспортированной ФС. Опция fg принуждает завершить процесс с возвратом ошибки (используется по умолчанию). Опция bg позволяет по достижении тайм-аута выполнить новую попытку монтирования. Если отсутствует локальная точка монтирования, то mount ведет себя как в случае превышения тайм-аута запроса на монтирование. Это позволяет выполнять монтирование ФС, описанных в /etc/fstab в произвольном порядке в время запуска системы, даже если какие-то NFS-серверы ещё не доступны
- ✓ retry=n — время (в минутах), через которое mount предпринимает повторные попытки монтирования ФС. По умолчанию — 2 для режима fg и 10000 для режима bg
- ✓ sec=mode — определяет метод безопасности RPCGSS, который используется для доступа к файлам на данной ФС. Если опция не указана или sec=sys, то клиент использует AUTH_SYS. Доступные значения параметра: none, sys, krb5, krb5i, krb5p, lkey, lkeyi, lkeyp, spkm, spkmi, spkmp
- ✓ sharecache / nosharecache — определяет, как разделяются кеш данных и кеш атрибутов клиента, когда одна экспортируемая ФС монтируется несколько раз одновременно. Использование общего кеша уменьшает затртаты памяти на клиенте и предоставляет одинаковое содержимое файла, когда доступ к нему производится с разных точек монтирования. Если опция не указана или указана опция sharecache, то используется общий кеш для всем точек монтирования, содежащих одну экспортируемую ФС. Начиная с ядра 2.6.18 политика nosharecache объявлена устаревшей, так как при наличии нескольких кешей возможно появление разных копий одного файла на одном клиенте.

Рассмотрим опции монтирования, доступные только для NSFv3 и ранее:

- ✓ proto=netid — транспортный протокол, который используется клитентом для передачи запросов серверу. Может иметь значение udp или tcp. Если опция не определена, то mount определяет, какие протоклы поддерживаются, и использует подходящий транспорт для каждого сервиса

- ✓ udp — то же, что и proto=udp
- ✓ tcp — то же, что и proto=tcp
- ✓ port=n — номер порта NFS-сервера. Если на указанном порту служба NFS не обнаружена, то запрос на монтирование завершается неудачей. Если опция не указана или параметр равен 0, то клиент использует номер порта, предложенный службой rpcbind сервера. Запрос на монтирование оканчивается неудачей, если на сервере недоступен rpcbind, не зарегистрирована служба NFS или недоступна служба NFS на порту, указанном rpcbind
- ✓ mountport=n — номер порта сервера, на котором находится mountd. Если на указанном порту mountd не обнаружен, то запрос на монтирование завершается неудачей. Если опция не указана или параметр равен 0, то клиент использует номер порта, предложенный службой rpcbind сервера. Запрос на монтирование оканчивается неудачей, если на сервере недоступен rpcbind, не зарегистрирован mountd или недоступен mountd на порту, указанном rpcbind. Эта опция может быть использована при монгтировании NFS через межсетевой экран, на котором закрыт протокол rpcbind
- ✓ mountproto=netid — транспортный протокол, который будет использоваться для передачи запросов mount и unmount. Может иметь значение tcp или udp. Полезно при выполнении монтирования через межсетевой экран, на котором заблокирован один из протоколов. Если используется вместе с опцией proto, то для передачи запросов mount и для передачи данных могут быть использованы разные транспортные протоколы. Если удалённый сервер mountd не поддерживает выбранный протокол, то запрос mount оканчивается неудачей
- ✓ mounthost=name — имя сервера, на котором запущен mountd. Если опция не определена, то mount полагает, что mountd запущен на том же сервере, что и nfsd
- ✓ mountvers=n — Версия протокола RPC, используемая для взаимодействия с mountd. Если опция не определена, то клиент использует версию, соответствующую версии NFS. Полезно, когад на сервере запущено несколько NFS-серверов
- ✓ namlen=n — максимальная длина строки пути монтирования. Если опция не определена, то производится согласование с сервером. В большинстве случаев эта длина равна 255 символов
- ✓ nfsvers=n — версия протокола NFS, используемая для взаимодействия с сервером. Linux-клиент поддерживает версии 2 и 3. Если сервер не поддерживает выбранную версию, то запрос на монтирование заканчивается неудачей. По умолчанию используется версия 3, а при недоступности — версия 2
- ✓ vers=n — синоним nfsvers
- ✓ lock / nolock — определяет, надо ли использовать протокол NLM для осуществления блокировок. По умолчанию используется опция lock. Если используется опция nolock, то программы всё равно могут выполнять блокировки, однако они будут действительны только для одного клиента. При монтировании /var должна использоваться опция nolock, так как файлы, содержащиеся в /var, используются реализацией NLM в linux. Также опция nolock необходима, если сервер не поддерживает NLM
- ✓ intr / nointr — определяет, можно ли посылать сигналы для прерывания файловых операций. По умолчанию сигналы игнорируются. Использование опции intr предпочтительнее использования опции soft, так как менее вероятно приводит к повреждению данных
- ✓ cto / nocto — определяет, использовать ли семантику когерентности кеша close-to-open. По умолчанию используется cto. При указании опции nocto клиент использует нестандартную эвристику для определения, какие файлы были изменены. Опция nocto может увеличить производительность, но должна использоваться тогда, когда данные меняются редко
- ✓ acl / noacl — определяет, надо ли использовать протокол NFSACL. NFSACL — проприетарный протокол, реализованный в Sun Solaris, который управляет списками доступа. Не является частью стандарта NFS. Если опция не определена, то происходит согласование между клиентом и сервером. Если протокол поддерживается сервером, то он используется

- ✓ rdirplus / nordirplus — определяет, надо ли использовать запросы READDIRPLUS, появившиеся в 3 версии NFS. Если опция не определена, то запросы используются для чтения малых каталогов. Некоторые приложения работают быстрее, если такие запросы не используются.

6. Сервер журналов syslog-ng

Сервер syslog-ng – это сервер, на который отправляются журналы со всех устройств в сети, поддерживающих протокол syslog. Такое решение позволяет иметь все журналы в исходном виде, даже в случае взлома устройства, которое использует сервер журналирования, или его поломки (например, выхода из строя жесткого диска). Кроме того, снимается нагрузка на жесткий диск клиентской системы.

Основное понятие сервера журналов – это поток (Facility) – это класс событий, которые записываются в один определённый пункт назначения (например, файл). Существуют предопределённые потоки (например, mail — поток для событий почтовых серверов, auth — для событий авторизации) и потоки, которые пользователь может использовать по своему усмотрению (local0-local7).

В Arch Linux по умолчанию используется демон syslog-ng, конфигурационный файл которого расположен по адресу «/etc/syslog-ng.conf». Обратите внимание: настройки по умолчанию достаточно хороши, но демон не настроен на работу в качестве сетевого сервера! Рассмотрим пример конфигурационного файла сервера журналирования. Жирным курсивом будут выделены строки, уже присутствующие в конфигурационном файле по умолчанию.

```
# Начало секции источников событий
source src {
# Приём событий из устройства
  unix-stream("/dev/log");
# Приём внутренних событий
  internal();
# Приём событий из файла
  file("/proc/kmsg");
# Приём событий через UDP-сокет – необходим для приёма событий из сети
  udp();
};

# Секция описания приёмников журналов
destination authlog { file("/var/log/auth.log"); };
destination syslog { file("/var/log/syslog.log"); };
destination cron { file("/var/log/crond.log"); };

# Создадим новый приёмник журналов с именем "des", который будет вести
# запись в файл /var/log/des-3838.log
destination des { file("/var/log/des-3828.log"); };

# Секция описания фильтров
filter f_auth { facility(auth); };
filter f_authpriv { facility(auth, authpriv); };
filter f_syslog { program(syslog-ng); };

# Создадим новый фильтр с именем f_des, выделяющий события от потока
# local0
filter f_des { facility(local0); };

# Секция связки источника, фильтра и приёмника
log { source(src); filter(f_acpid); destination(acpid); flags(final);
};
log { source(src); filter(f_authpriv); destination(authlog); };
log { source(src); filter(f_syslog); destination(syslog); };

# Создадим новую связку – источник src, фильтр f_des и приёмник des
```

```
log { source(src); filter(f_des); destination(des); };
```

Сохраните файл и перезапустите демон syslog-ng, выполнив команду

```
/etc/rc.d/syslog-ng restart
```

Проверьте ответ команды

```
netstat -lnup
```

Вы должны увидеть демон syslog-ng, ожидающий соединения на 518 порту по протоколу udp.

Сервера приложений

7. Веб-сервер Apache

Apache httpd – стандарт де-факто на рынке веб-серверов. В настоящее время этот сервер является, пожалуй, наиболее функциональным. Существуют также более лёгкие и модульные аналоги (например, lighttpd и nginx), которые в основном применяются на несложных сайтах или в качестве кеширующего веб-сервера перед httpd (обработка статических страниц происходит намного быстрее и меньше нагружает систему). Преимущество httpd заключается в наличии поставляемых с сервером модулей для обработки страниц на интерпретируемых языках (php, perl, python, прочие CGI). Как следствие, отсутствует необходимость настройки FastCGI и прочих CGI-прокси

В Arch Linux настройки httpd хранятся в каталоге /etc/httpd. Каталог имеет следующую структуру:

1. ./build – символическая ссылка на каталог с файлами, необходимыми для сборки внешнего динамически загружаемого модуля (DSO).
2. ./conf – содержит файлы настройки сервера.
3. ./conf/extra – содержит дополнительные настроки, подключаемые из основного файла настроек.
4. ./logs – ссылка на каталог с журналами сервера (/var/log/httpd).
5. ./modules – ссылка на каталог, содержащий имеющиеся загружаемые модули.
6. ./run – ссылка на каталог, в котором сервер хранит временные файлы, необходимые для работы.

Основные настройки содержатся в файле /etc/httpd/conf/httpd.conf.

Управление сервером

Запуск сервера:

```
# /etc/rc.d/httpd start
```

Останов сервера:

```
# /etc/rc.d/httpd stop
```

Перезапуск сервера:

```
# /etc/rc.d/httpd restart
```

Перечитывание конфигурационного файла (без остановки сервера):

```
# /etc/rc.d/httpd reload
```

Добавление поддержки PHP

Для добавление поддержки PHP-скриптов необходимо выполнить следующие действия:

1. Добавьте в файл conf/httpd.conf следующие строки:
   - ✓ в раздел описания модулей:
     ```
     LoadModule php5_module modules/libphp5.so
     ```
   - ✓ в конец файла:
     ```
     Include conf/extra/php5_module.conf
     ```
2. Перезапустите сервер.

Выбрать необходимые модули PHP можно в конце файла /etc/php/php.ini.

Добавление поддержки SSL

Для добавление поддержки протокола SSL необходимо выполнить следующие действия:

1. В файле conf/httpd.conf раскомментируйте строку:
   `Include conf/extra/httpd-ssl.conf`
2. Отредактируйте файл conf/extra/httpd-ssl.conf.

8. FTP-сервер ProFTPD

Конфигурационный файл сервера располагается по адресу /etc/proftpd.conf. Журнал аутентификации ведётся в файл /var/log/auth.log. Если вы заходите на сервер от имени пользователя ftp, обратите внимание, чтобы этот пользователь имел действительную оболочку (например, /bin/bash). Для запуска сервера используется команда:

```
$ /etc/rc.d/proftpd start
```

Если вы хотите указать, что сервер должен стартовать с определёнными ключами, укажите эти ключи в файле /etc/conf.d/proftpd. Пример настройки сервера:

```
# Имя сервера
ServerName              "ProFTPD Default Installation"
# Тип сервера
ServerType              standalone
# Является ли сервером по умолчанию. Для варианта, когда сервер не
# единственный
DefaultServer           on

# Порт, на котором сервер будет ожидать соединение
Port                    21

# Права, с которыми будут создаваться файлы и каталоги. Чтобы получить
# эффективные права в формате Unix, отнимите от 777 значение этого
# параметра
Umask                   022

# Максимальное количество одновременных соединений
MaxInstances            30

# Пользователь и группа, с правами которой будет запущен сервер
User                    nobody
Group                   nobody

# Эту строку можно раскомментировать, если есть необходимость в
# ограничении прав пользователя на просмотр структуры каталогов. Выше
# указанного каталога пользователь подняться не сможет
#DefaultRoot ~

# Разрешает перезапись файлов
AllowOverwrite          on



# Запрет использования SITE CHMOD
<Limit SITE_CHMOD>
  DenyAll
</Limit>

# Базовая настройка анонимного пользователя в режиме «только чтение»
```

```
<Anonymous ~ftp>

# Имя пользователя и группа, правами которой обладает анонимный
# пользователь
  User                    ftp
  Group                   ftp

  # Псевдонимы для имени anonymous
  UserAlias               anonymous ftp

  # Максимальное количество одновременных соединений от анонимного
  # пользователя
  MaxClients              10

  # Сообщение, показываемое при входе на сервер
  DisplayLogin            welcome.msg
  # Сообщение, показываемое при смене каталога
  DisplayChdir            .message

  # Запретить запись
  <Limit WRITE>
    DenyAll
  </Limit>
</Anonymous>
```

9. Служба Samba

Samba – реализация протокола SMB (Server Message Block), используемого в Windows в основном для обеспечения общего доступа к каталогам (существуют и другие применения). Также Samba может выступать в качестве замены серверу каталогов Active Directory (недостаток – создание и редактирование групповых политик реализовано только в Samba 4, стабильной версии которой пока нет). Также Samba предоставляет воможность аутентификации linux-пользователей в Active Directory посредством Winbind (можно проводить аутентификацию и через pam_ldap) и является WINS-сервером.

Настройка Samba осуществляется через файл /etc/samba/smb.conf. Samba работает со своим хранилищем пользователей (tdbsam, может также работать и с LDAP). Чтобы добавить пользователя в базу Samba, выполните команду:

```
# smbpasswd -a username
```

Для смены пароля выполните ту же команду, но без ключа –a.

Рассмотрим пример настройки общедоступного файлового сервера. Формат конфигурационного файла является INI-подобным.

```
# Секция, отвечающая за базовые настройки
[global]
# Рабочая группа, к которой принадлежит сервер
 workgroup = workgroup
# NetBIOS-имя сервера
 netbios name = fileserver
# NetBIOS-псевдоним
 netbios aliases = garbage
# Комментарий, отображаемый в «сетевом окружении». %v заменяется на
# версию Samba
 server string = Samba Server %v
# Расположение журнала. %m заменяется на адрес клиента (IP или, если
# доступен NetBIOS)
 log file = /var/log/samba/log.%m
# Максимальный размер журнала
 max log size = 500
# Включение WINS-сервера
 wins support = yes
# Публикация доступных каталогов на локальном браузере
 browse list = yes
# Samba будет выставлять свою кандидатуру на выборах локального
# браузера
 local master = yes
# Предлагать сервер как предпочтительный локальный браузер
 preferred master = yes
# «Уровень» операционно системы. Влияет на результат выборов. Это
# значение
# гарантирует победу над NT-серверами
 os level = 33
# Модель безопасности — общедоступный разделяемый ресурс
 security = share
# Отключение регистрозависимости имён файлов
 case sensitive = no
# Сохранять информацию о регистре имени
 preserve case = yes

# Общий ресурс с именем share
[share]
```

```
# Комментарий, отображаемый в «сетевом окружении»
 comment = shared data
# Каталог, соответствующий данному ресурсу
 path = /home/share
# Максимальный размер каталога в мегабайтах
 max disk size = 20000
# Разрешить публикацию на локальном браузере
 browseable = yes
# Разрешить запись в этот каталог
 writeable = yes
# Разрешить публичный гостевой доступ
 guest ok = yes
 public = yes
```

Можно также задавать ограничения по пользователям. Для этого модель безопасности необходимо изменить на user. За распределение прав отвечают следующие директивы в описании общего ресурса:

1. read users – список пользователей, которым разрешено чтение. Разделитель – запятая.
2. write users – список пользователей, которым разрешена запись. Разделитель – запятая.
3. admin users – список пользователей, имеющих абсолютные права. Разделитель – запятая.

Чтобы запустить сервер Samba, выполните команду:

```
# /etc/rc.d/samba start
```

Просмотр общих ресурсов может выполняться с помощью браузера Konqueror (smb://192.168.0.1) или консольным клиентом smbclient (smbclient -U username //192.168.0.1/share). Также существуют графические программы Smb4k и pyneighborhood для просмотра общих ресурсов. Файловые системы cifs и smbnetfs позволяют монтировать общие ресурсы к локальной файловой системе.

10. SMTP-сервер Postfix

Postfix – это SMTP-сервер, сочетающий в себе мощность, функциональность, гибкость и простоту настройки. Запуск Postfix для обслуживания одного домена занимает 5-10 минут (без настройки фильтрации). Настройки Postfix обеспечивают достаточную производительность и степень безопасности для сервера небольшой организации (по крайней мере, этот сервер не превратится в публичный). Postfix является модульным сервером (и число частей, на которые он разбит, достаточно велико).

Рассмотрим пример настройки Postfix для обслуживания домена example.com
Основной (и единственный, интересный нам) конфигурационный файл - /etc/postfix/main.cf.

```
# Каталог, в котором будут содержаться очереди сообщений
queue_directory = /var/spool/postfix

# Путь по умолчанию для команд вида post*
command_directory = /usr/sbin

# Каталог, содержащий всех демонов, составляющих Postfix. Владельцем
# каталога должен быть пользователь root
daemon_directory = /usr/lib/postfix

# Каталог, в котором хранятся данные Postfix (кеши, случайные числа и
# т.д.). Владельцем каталога должен быть пользователь, указанный в
# параметре mail_owner
data_directory = /var/lib/postfix

# Указывает пользователя-владельца почтовых очередей и большинства
# демонов, составляющих Postfix. Нельзя указывать уже используемую
# кем-то учётную запись
mail_owner = postfix

# Определяет имя сервера
myhostname = example.com

# Определяет доменную часть почтового адреса отправителя ЛОКАЛЬНЫХ
# сообщений. Так, почта, локально отправленная от имени пользователя
# user, будет содержать имя отправителя user@$myhostname
# (user@example.com). Все значения параметров, начинающиеся с символа
# $, вляются макросами и заменяются при запуске Postfix на значения
# одноимённых параметров
myorigin = $myhostname

# Ожидать соединения на всех доступных сетевых адресах
inet_interfaces = all

# Имена доменов, для которых надо принимать почту
mydestination = $myhostname, localhost.$mydomain, localhost

# Определяет локальных получателей почты
# В данном случае используется системная база пользователей
# (/etc/passwd) и база псевдонимов, заданная параметром alias_maps
local_recipient_maps = unix:passwd.byname $alias_maps

# Код, с которым будет отклоняться почта для неизвестного получателя
unknown_local_recipient_reject_code = 550
```

```
# Определяет сети, почта из которых будет отправляться без прохождения
# авторизации. Возможны следующие варианты:
# host — только с этого же хоста
# subnet — из подсети, которой принадлежит данный хост
# class — из сетей, относящихся к тому же классу, что и сеть, в
# которую входит данный хост
mynetworks_style = host

# Определяет путь до базы псевдонимов
alias_maps = hash:/etc/postfix/my_tables/aliases

# Определяет путь до базы псевдонимов, созданной командой newaliases.
# При изменении файла псевдонимов необходимо выполнить команду
# newaliases и заставить Postfix перечитать настройки. Необходим, так
# как необязательно все базы из параметра alias_maps находятся под
# контролем Postfix
alias_database = $alias_maps

# Определяет строку, выдаваемую сервером в начале SMTP-диалога
smtpd_banner = $myhostname ESMTP $mail_name

# Путь до команды sendmail из состава Postfix
sendmail_path = /usr/sbin/sendmail

# Путь до команды newaliases из состава Postfix
newaliases_path = /usr/bin/newaliases

# Путь до команды mailq из состава Postfix
mailq_path = /usr/bin/mailq
```

Запустите почтовый сервер, выполнив команду `/etc/rc.d/postfix start`. Чтобы заставить Postfix перечитать настройки, выполните команду `/etc/rc.d/postfix reload`.

11. POP/IMAP-сервер Dovecot

Dovecot является производительным и гибким POP/IMAP-сервером. Может работать с различными базами пользователей, поддерживает SSL и TLS.

Конфигурационный файл Dovecot разбит на несколько: основная часть находится в файле /etc/dovecot/dovecot.conf, а прочие настройки по группам содержатся в каталоге /etc/dovecot/conf.d, причём используются только файлы с расширением «.conf». Важно понимать, что все настройки, включаемые из других файлов, могут быть записаны и в один файл.

Рассмотрим пример файла /etc/dovecot/dovecot.conf:

```
# Протоколы, с которыми сервер будет работать
# LMTP (Local Mail Transfer Protocol) — протокол локальной доставки почты
# (например, между SMTP-сервером и сервером, обслуживающим почтовые ящики,
# например, Dovecot
protocols = imap pop3 lmtp

# Адреса, на которых будет работать Dovecot
# Элементы списка разделяются запятыми
# * - все адреса Ipv4
# :: - все адреса IPv6
listen = *, ::

# Приветствие, выдаваемое клиенту
login_greeting = Dovecot ready

На этом основные настройки закончены. Далее рассмотрим важные настройки из дополнительного набора.

10-auth.conf — настройки процесса аутентификации
# Управление функцией аутентификации прямым текстом (plaintext)
# Значение yes или no разрешает или запрещает такую аутентификацию
# при условии неиспользования SSL соответственно
disable_plaintext_auth = no

# Разрешённые механизмы аутентификации
# Учитывайте ключ disable_plain_auth
auth_mechanisms = plain

# Включение части файла, содержащего сведения о базе
# из которой берутся учётные данные пользователей
# Самый простой пример такой базы — база, формируемая системой PAM.
!include auth-system.conf.ext

auth-system.conf.ext
# Определение базы учётных данных пользователей, получаемой из PAM
# Через эту базу происходит аутентификация клиентов
# Настройки PAM хранятся в файле /etc/pam.d/dovecot
passdb {
  driver = pam
}

# Определение базы информации о пользователях, получаемой из NSS
```

```
# Через эту базу происходит поиск дополнительных сведений о
пользователе
# (например, домашнего каталога)
userdb {
  driver = passwd
}

10-mail.conf – настройки расположения почтовых ящиков
# Расположение почтового ящика пользователя
# Можно применять следующие переменные:
# %u – имя пользователя
# %n – часть имени пользователя до символа @
# %d – часть имени пользователя после символа @
# %h – домашний каталог пользователя
# Требуется указывать формат почтового ящика – mbox или Maildir
mail_location = mbox:/var/mail/%u

10-master.conf – настройки основного рабочего процесса
# Определение служб аутентификации для почтовых протоколов
# Можно изменить порт, на котором располагается та или иная служба
service imap-login {
  inet_listener imap {
    port = 143
  }
}

service pop3-login
  inet_listener pop3 {
    port = 110
  }
}

10-ssl.conf – настройки использования SSL
# Параметр, отключающий необходимость использования SSL
ssl = no
# Естественно, необходимо закомментировать строки, указывающие на
сертификаты
```

На этом базовая настройка простейшего POP/IMAP-сервера, обслуживающего системных пользователей, завершена. Так как все файлы конфигурации Dovecot снабжены подробными комментариями, несложно расширить функциональность сервера под ваши нужды.

Запуск сервера производится командой

```
# /etc/rc.d/dovecot start.
```

12. Почтовый клиент KMail

Утилита KMail – стандартный почтовый клиент из окружения рабочего стола KDE. Поддерживает локальные почтовые ящики (mbox и Maildir) и получение почты с удалённых серверов POP3 и IMAP. Поддерживает шифрование SSL и TLS. Рассмотрим добавление почтового ящика с удалённого сервера POP.

1. Запустите KMail из меню «К → Офис → KMail».
2. Если будет показано окно первого запуска — нажмите кнопку «Отмена».
3. Откроется главное окно (рисунок 4.5).

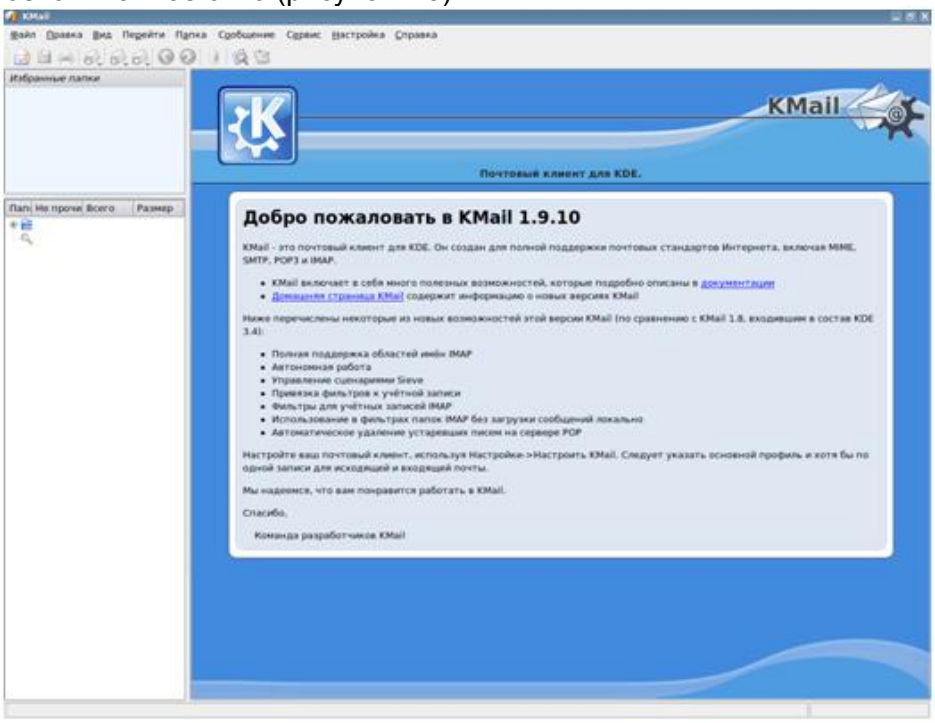


Рисунок 4.5.

4. Выберите пункт меню «Настройки → Настроить KMail». Откроется следующее окно (рисунок 4.6).

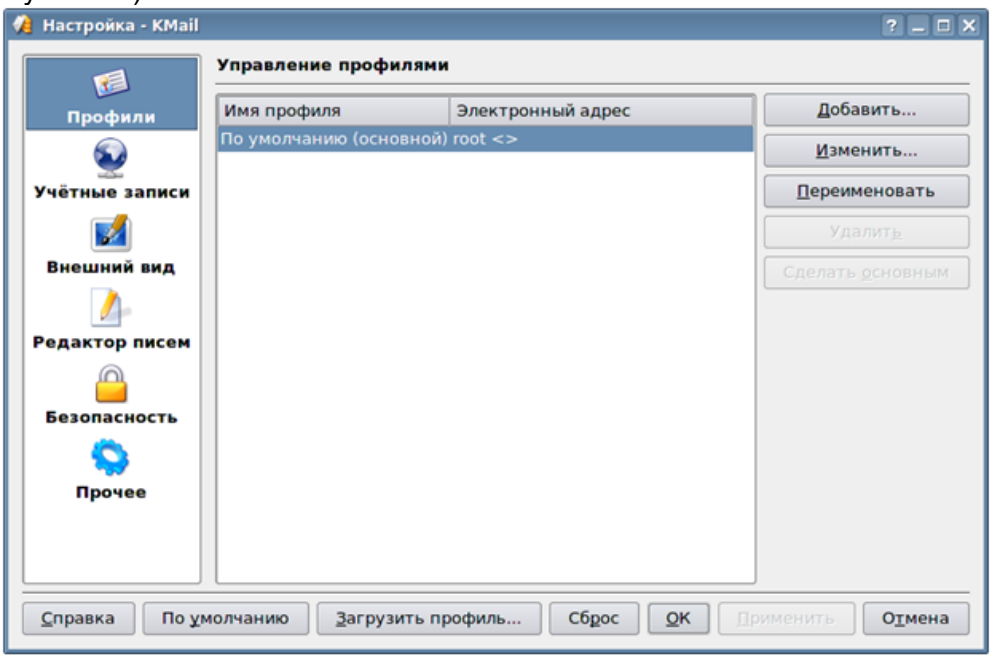


Рисунок 4.6.

5. Слева выберите раздел «Учётные записи» (рисунок 4.7).

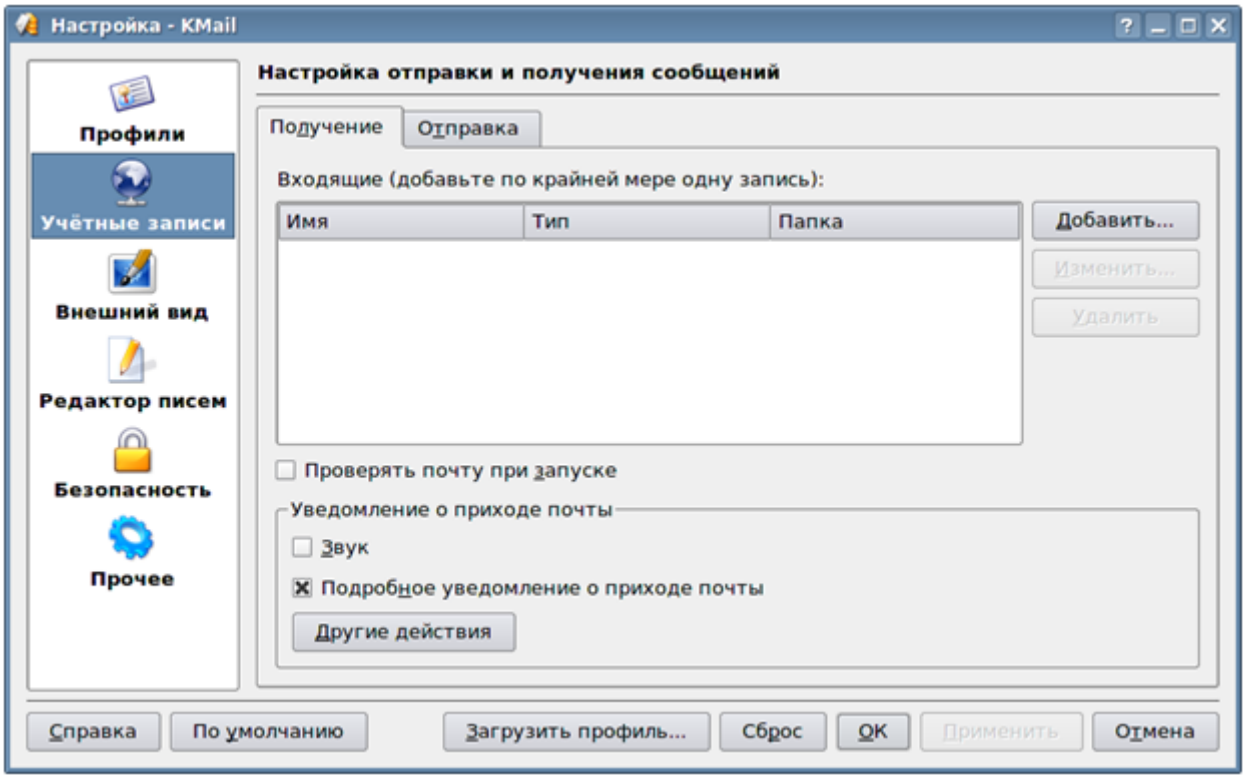


Рисунок 4.7.

6. Нажмите кнопку «Добавить».
7. Выберите тип почтового ящика POP3 (рисунок 4.8).

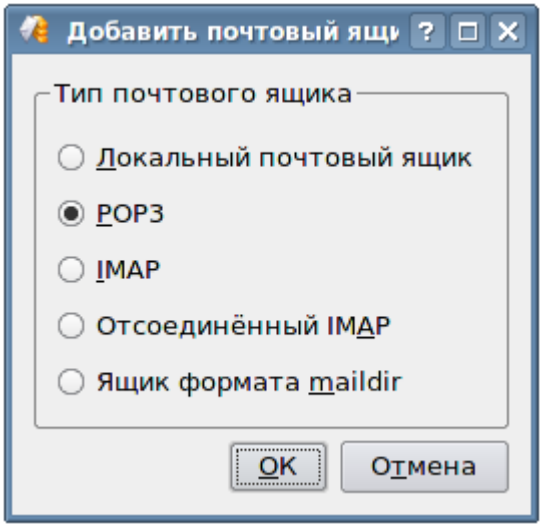


Рисунок 4.8.

8. Заполните поля следующим образом (рисунок 4.9):
   - ✓ Почтовый ящик — имя нового почтового ящика (например, me@my.com)
   - ✓ Учётное имя — логин на удалённом почтовом сервере
   - ✓ Пароль — пароль к учётной записи на удалённом почтовом сервере
   - ✓ Поставьте флаг «Сохранить пароль POP»
   - ✓ Поставьте флаг «Оставить полученные сообщения на сервере»

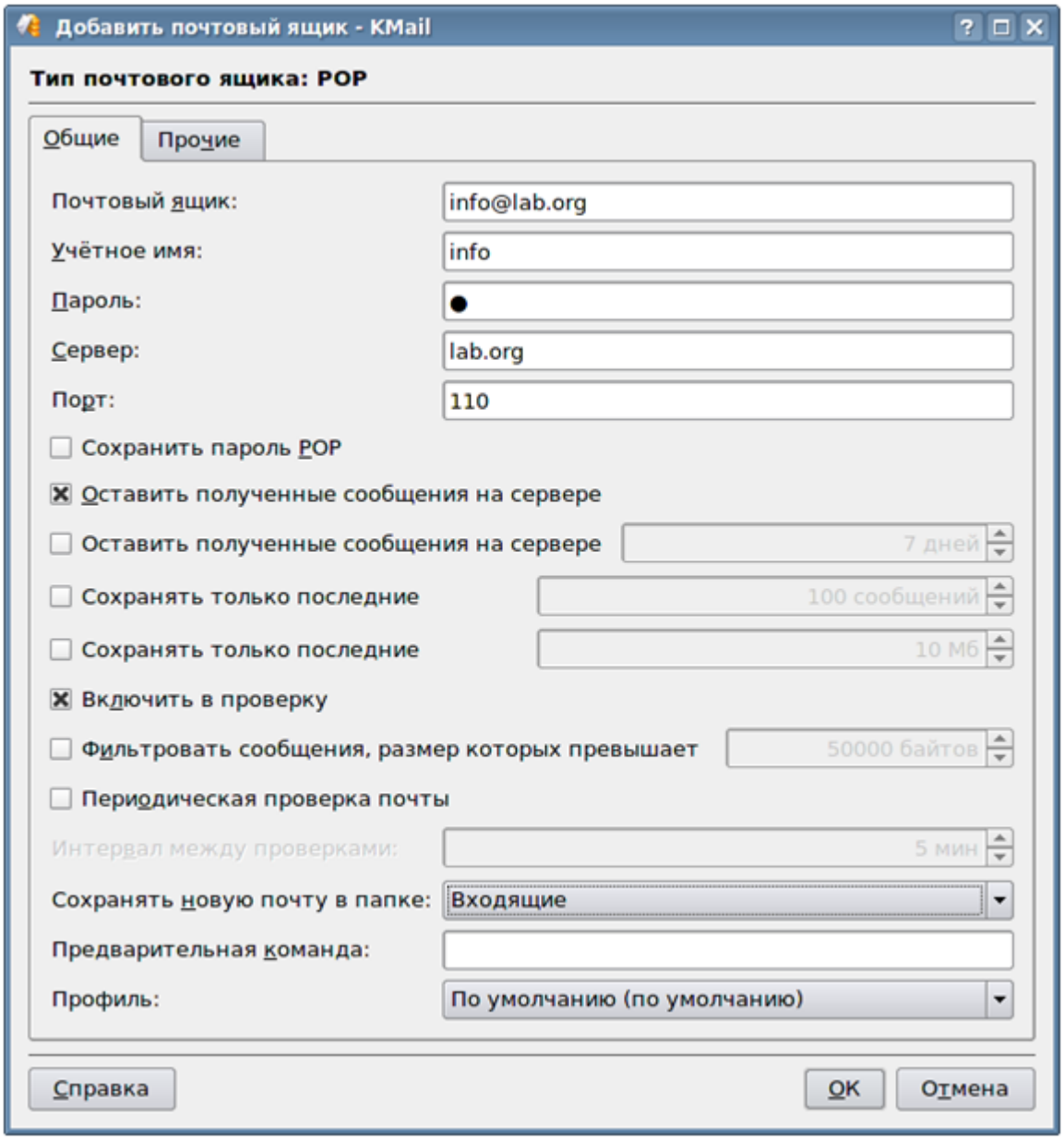


Рисунок 4.9.

9. Нажмите кнопку «ОК».
10. Ещё раз нажмите кнопку «ОК».
11. Для получения почты воспользуйтесь меню «Файл → Проверить почту».

Сервисы

13. Сервер точного времени ISC NTPD

Протокол Network Time Protocol позволяет поддерживать одинаковое время на всех компьютерах и прочих сетевых устройствах. Одинаковое время необходимо, если в сети используются сервисы авторизации, основанные на взаимной проверке сервера авторизации и клиента (например, Kerberos). Ещё один плюс одинакового времени на всех устройствах – вы точно знаете, когда произошло некоторое событие (например, при чтении журналов). В любой более-менее крупной сети использование этого протокола вполне оправданно и даже необходимо (равно как и прочие синхронизации).

Существует несколько реализаций серверов NTP, но стандартом де-факто в настоящее время является ISC NTPD (как и многие другие сетевые сервисы от ISC). Рассмотрим пример настройки сервера точного времени на основе ISC NTPD. Конфигурационный файл единственный - /etc/ntp.conf.

```
# Указываем вышестоящие серверы точного времени (если к таковым есть
# доступ). Хотя бы одна действительная директива server обязана
# присутствовать в файле! Желательно указывать не менее трёх серверов,
# если такая возможность есть.
# Имеет смысл только в случае доступности указанного сервера
server ru.pool.ntp.org

# Путь до файла, в котором NTPD хранит смещение времени относительно
# эталонного
driftfile       /var/lib/ntp/ntp.drift

# Указываем системный таймер в качестве источника точного времени
# При наличии более точных источников делать такое не рекомендуется
server 127.127.1.1
fudge 127.127.1.1 stratum 0 refid NIST

# Запрещаем изменять конфигурацию сервера отовсюду, кроме локальной
# машины
restrict default nomodify nopeer
restrict 127.0.0.1

# Разрешаем нашей подсети снимать показания с данного сервера, но
# запрещаем изменять настройку сервера
restrict 192.168.10.0 mask 255.255.255.0 nomodify nopeer notrap
```

Запустите NTPD, выполнив команду `/etc/rc.d/ntpd start`. Проверить состояние связи созданного сервера с источниками точного времени можно, выполнив команду

```
# ntpq -c peers -n
     remote           refid      st t when poll reach   delay   offset
jitter
==============================================================================
 127.127.1.1     .NIST.           0 l   53   64  377    0.000    0.000
0.001
+193.233.85.131  147.45.15.34     3 u   16 1024  377    1.641   -3.145
5.337
 193.233.85.60   193.233.85.132   4 u  512 1024   17    0.502   -4.567
1.506
*193.233.85.132  147.45.15.34     3 u   23 1024  377    4.922  -35.893
0.097
```

Созданный сервер синхронизируется с теми серверами, записи о которых в выводе этой утилиты отмечены знаком + или *.

15. DHCP-сервер ISC DHCPD

Для редактирования настроек DHCP-сервера имеется единственный файл настроек /etc/dhcpd.conf. Формат файла приведён ниже.

```
ddns-update-style параметр;
# Данная опция задаёт режим обновления DNS-сервера и может принимать три значения:
# none – отсутствует режим обновления;
# ad-hoc – режим непосредственного обновления DNS;
# interim – режим взаимодействия DNS-DHCP.

# Далее идёт раздел объявления сети, настройки которой будут получать клиенты
subnet 192.168.1.0 netmask 255.255.255.0
{
      # Необязательная опция. Задаёт шлюз по умолчанию.
      option routers                  192.168.1.254;

      # Необязательная опция. Задаёт маску сети.
      option subnet-mask              255.255.255.0;

      # Необязательная опция. Задаёт имя домена.
      option domain-name              "example.com";

      # Необязательная опция. Задаёт DNS-сервера.
      option domain-name-servers      192.168.1.1;

      # Необязательная опция. Задаёт смещение времени.
      option time-offset              -18000;

      # Необязательная опция. Задаёт диапазон выдаваемых адресов.
      range                             192.168.1.10 192.168.1.100;

      # Задаёт время аренды, то есть на какое время клиент получает от сервера
      # настройки. По истечению данного времени клиент будет запрашивать настройки
      # заново.
      default-lease-time                  секунды;

      # Время, через которое клиент должен освободить (вернуть) выданные ему
      # настройки.
      max-lease-time                secунды;

      # После объявления глобальных параметров, которые будут общими для вех
      # клиентов, можно создать необязательные статические записи, то есть присвоить
      # конкретным машинам конкретные IP-адреса.
      host имя_узла
      {
         hardware ethernet MAC-адрес;
         fixed-address IP-адрес;
      }
}
```

После внесения изменений в файл настроек необходимо перезапустить сервер:

```
$ /etc/rc.d/dhcpd restart
```

Если в результате запуска произошёл сбой, то причину ошибки можно посмотреть в файле /var/log/messages.log.

16. DNS-сервер ISC BIND

ISC BIND является стандартом де-факто в Интернет. Этот DNS-сервер является очень гибким, конфигурируемым, производительным и надёжным. Для работы серверу необходимы следующие данные:

1. Основные параметры (адрес, порт, расположение журналов).
2. Описание обслуживаемых зон.
3. Собственно зоны.

BIND может являться как первичным, так и вторичным сервером. Может содержать так называемые зоны-заглушки (stub). Может являться авторитетным или кеширующим сервером. В Arch Linux BIND запускается от имени пользователя named (в целях безопасности), поэтому все файлы, содержащие зоны, должны быть доступны для чтения этому пользователю. Рассмотрим пример настройки BIND как первичного сервера зоны localhost.

Основная настройка сервера

Все базовые настройки BIND хранятся в файле /etc/named.conf. Формат файла является C-подобным.

```
// начало основных настроек
options
{
// корневой каталог для зон
        directory "/var/named";

// файл, содержащий PID процесса
        pid-file "/var/run/named/named.pid";

// указание, что данный сервер — авторитетный (в случае невозможности
// выполнения запроса по причине отсутствия сведений об объекте из
// запроса будет возвращён ответ non-existing domain (NX-DOMAIN) с
// установленным битом авторитетности ответа)
        auth-nxdomain yes;

// способ ограничения памяти, потребляемой сервером
        datasize default;

// хосты, для которых разрешены рекурсивные запросы к вышестоящим
// серверам
        allow-recursion { 127.0.0.1; };

// адрес, на котором сервер будет ожидать запросы. any — любой
// доступный адрес
        listen-on { any; };

// то же, что и listen-on, но для протокола IPv6
        listen-on-v6 { any; };
};

//описание зоны localhost
zone "localhost" IN
{
//тип обслуживания. master — значит, этот сервер первичный для этой
//зоны
        type master;

//файл, в котором содержится зона
```

```
        file "localhost.zone";

//хосты, которым разрешено динамически изменять зону (например, через
//связку DNS-DHCP)
        allow-update { none; };

//список подчиннных, вторичных серверов
        allow-transfer { any; };
};

//описание обратной зоны 127.0.0.0/24
zone "0.0.127.in-addr.arpa" IN
{
        type master;
        file "127.0.0.zone";
        allow-update { none; };
        allow-transfer { any; };
};

//описание корневых серверов
zone "." IN
{
        type hint;
        file "root.hint";
};

//описание обратной зоны 192.168.100.0/24
zone "100.168.192.in-addr.arpa" IN
{
        type master;
        file "192.168.100.zone";
        allow-update { none; };
        allow-transfer { none; };
};

//настройки журналирования
logging
{
//создаём новый канал
        channel xfer-log
        {
//файл, в который будут записываться события
                file "/var/log/named.log";

//указывать категорию события
                print-category yes;

//указывать степень опасности события
                print-severity yes;

//указывать время события
                print-time yes;

//записывать события со степенью опасности не ниже информационных
                severity info;
        };

//записывать события приёма зоны с первичного сервера в канал xfer-log
```

```
        category xfer-in { xfer-log; };

//записывать события передачи зоны на вторичный сервер в канал xfer-
//log
        category xfer-out { xfer-log; };

//записывать оповещения в канал xfer-log
        category notify { xfer-log; };
};
```

Зоны

Важно понимать, что для BIND все строки, не оканчивающиеся точкой – это не полностью определённые имена домена (non-fqdn), поэтому эти имена будут дополнены до fqdn путём дописывания в конец доменной части, определяемой директивой $ORIGIN. Если такой директивы нет – в файле зоны все имена доменов должны быть указаны в формате fqdn, то есть быть полностью определёнными.

Если несколько записей подряд относятся к одному и тому же домену (например, подряд идущие записи SOA, A, NS и MX для зоны), то имя домена можно указывать один раз. Все записи имеют следующий вид:

```
имя_домена TTL тип тип_записи параметры
```

Ниже приведён пример зоны loclahost, определённой в файле /var/named/localhost.zone.

```
// определяем общую доменную часть адреса. Символ @ означает ссылку на
// эту часть
$ORIGIN localhost.

// SOA-запись
// Параметры:
// Имя первичного DNS-сервера, обслуживающего эту зону
// Почтовый адрес человека, ответственного за сервер. Вместо символа @
//в качестве разделителя используется точка
@                       1D IN SOA       @ root (

// серийный номер зоны. Обычно в формате <год><месяц><число><номер ревизии>
42              ; serial (yyyymmdd##)

// период между обновлениями зоны на клиентских кешах
3H              ; refresh

// период между повторными запросами, если сервер недоступен
15M             ; retry

// время жизни зоны, если сервер недоступен
1W              ; expiry

// время жизни зоны, если сервер доступен
1D )            ; minimum ttl

// NS-запись. Перечисляются все сервера (первичные и вторичные),
// ответственные за зону
1D IN NS        @

// A-запись. Требуется, если данный сервер является авторитетным для
```

```
// данной зоны
1D IN A         127.0.0.1
```

Запуск DNS-сервер осуществляется командой

```
/etc/rc.d/named start
```

При изменении настройки сервера или зон, используйте команду `rndc reload`, чтобы BIND перечитал свои настройки и файлы зон. Журнал BIND по умолчанию пишется в файл «/var/log/messages.log». Просмотреть записи, относящиеся только к BIND, можно, выполнив команду

```
grep named /var/log/messages.log.
```

Удалённое управление

17. Утилиты управления сетью по протоколу SNMP

Утилита iReasoning MIB Browser
Данная утилита является стандартным обозревателем базы данных MIB, поддерживаемой технологией SNMP. Утилита является кросс-платформенной, так как написана на языке Java. Для запуска утилиты запустите файл root/Desktop/SNMP/mibbrowser/browser.sh. Появится окно, представленное на рисунке 4.2.

Рисунок 4.2. Главное окно обозревателя SNMP.

По умолчанию в программе загружаются две базы MIB. Если необходимо загрузить дополнительные базы, то используйте пункт меню «File Load MIBs».

Для работы с определенным сетевым устройством необходимо в поле «Address» ввести IP-адрес данного устройства. Для того, чтобы получить значение записи в базе MIB устройства, необходимо выбрать нужную запись и нажать «CTRL-G» или выбрать пункт меню «Operations Get» или нажав правую кнопку мыши выбрать команду «Get». Для того, чтобы получить значение всей базы выберите пункт меню «Operations Walk». Для того, чтобы просмотреть содержимое таблицы необходимо выбрать команду «Table View» (рисунок 4.3).

Рисунок 4.3. Просмотр содержимого таблицы.

Утилита mbrowse
Данная утилита входит в стандартный пакет программного обеспечения операционной системы Arch Linux и используется для просмотра и изменения параметров удалённой системы по протоколу SNMP. Для её запуска откройте терминал и введите:

```
$ mbrowse
```

Окно программы представлено на рисунке 4.4 и состоит из следующих областей:
1 — Поле ввода адреса (имени) транслятора SNMP
2 — Поле ввода имени группы для чтения
3 — Поле ввода имени группы для чтения/записи
4 — Кнопка получения значения параметра
5 — Кнопка рекурсивного обхода дерева параметров, начиная с выделенного раздела
6 — Поле просмотра дерева доступных параметров
7 — Поле просмотра значений параметров

Для получения данных с определённого агента SNMP необходимо:
1. В поле 1 указать адрес или имя агента.
2. В поле 2 указать имя группы для чтения.
3. В поле 6 выбрать нужный параметр и нажать кнопку 4.
4. Либо в поле 6 выбрать нужную ветку дерева и нажать кнопку 5.

В поле 7 отобразится запрошенная информация (при правильной работе агента и наличии запрошенных параметров).
Рисунок 4.4.

18. Пакет OpenSSH

OpenSSH наиболее известен как безопасное средство удалённого администрирования. Пакет OpenSSH был разработан в рамках проекта OpenBSD как замена проприетарному пакету SSH. Позволяет помимо предоставления удалённого командного интерпретатора создавать защищенные туннели и организовывать безопасную передачу файлов (Secure FTP). Реализует версию 2 протокола ssh. Порт по умолчанию – 22. Шифрование производится по алгоритму DSA[3] (SSHv1) или по алгоритму RSA (SSHv2) парой ключей (открытым и закрытым) произвольной длины. Авторизация может производиться по имени/адресу клиента[4], по паролю (через системную базу пользователей или через PAM), по билету Kerberos или по ключам. Могут использоваться ключи, как защищенные парольной фразой, так и незащищенные[5]. Позволяет сжимать передаваемый трафик. Позволяет осуществлять проброс протокола X11 через шифрованный туннель, что позволяет создавать удалённые X-терминалы с безопасным подключением. Ключи шифрования могут быть проверены через DNS (при наличии в зоне записи SSHFP). Рассмотрим все роли по порядку.

Удалённое администрирование

Это, пожалуй, основное применение ssh. Заменяет устаревшие и абсолютно небезопасные telnet и r-службы (а именно rsh). Используется следующим образом:

```
$ ssh myhost
```

Эта команда выполнит безопасное подключение к серверу myhost. Вместо имени может быть указан адрес. При первом подключении будет выдано следующее предупреждение:

```
$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is
a6:53:ca:72:f5:64:e1:dd:52:f5:0b:c5:5c:55:db:ff.
Are you sure you want to continue connecting (yes/no)?
```

Это происходит, так как в вашей базе известных ключей (есть общесистемная база и база, отдельная для каждого пользователя) нет ключа данного сервера. Если вы уверены, что сервер не поддельный, то наберите «yes». Далее в зависимости от выбранного способа авторизации может быть выдан запрос на ввод пароля или парольной фразы. Обратите внимание, что при смене ключа сервера клиент в зависимости от настроек либо повторно запросит разрешение на добавление ключа, либо категорически запретит подключение к данному серверу. В такой ситуации надо быть предельно внимательным, так как если смена ключа сервера произошла без вашего ведома, это может означать, что сервер был взломан или вы подверглись атаке «человек

3 Алгоритм шифрования DSA сравнительно легко поддаётся атакам методом перебора, поэтому не рекомендуется использовать этот алгоритм при работе с важными данными

4 Использовать такой метод авторизации категорически нельзя, даже если ssh-сервер располагается на компьютере внутренней сети и принимает пакеты только из этой сети. Технология подделки IP-адреса стала слишком публичной, чтобы можно было полагаться на IP-адрес в качестве однозначного идентификатора клиента

5 Предпочтительней использовать парольную фразу, однако для некоторых целей это порождает неудобства, поэтому ключ с парольной фразой не всегда применим. Однако ключ без парольной фразы также является весьма надёжным средством аутентификации (если закрытый ключ не был утерян)

посередине». Если Вы абсолютно уверены в своих действиях, повторно ответьте «yes» или удалите соответствующую запись (в выводе клиента указан номер) из файла ~/.ssh/known_keys (пользовательская база) или /etc/ssh/ssh_known_keys (общесистемная база).

Если требуется выполнить вход от имени другого пользователя, можно воспользоваться следующими командами:

```
$ ssh -l user myhost
$ ssh -luser myhost
$ ssh myhost -l user
$ ssh user@myhost
```

Все эти команды делают одно и то же – выполняют подключение к серверу myhost от имени пользователя user. Если сервер находится на нестандартном порту, воспользуйтесь следующей командой:

```
$ ssh -p 2222 myhost
```

Чтобы включить сжатие трафика, используйте следующую команду:

```
$ ssh -C myhost
```

Чтобы включить проброс X11, используйте следующую команду:

```
$ ssh -X myhost
```

Шифрованные туннели

Внимание! Для успешного туннелирования на удалённом сервере в файле /etc/ssh/sshd_config должна быть включена опция GatewayPorts (то есть выставлена в «yes»). Чтобы выполнить перенаправление пакетов с некоторого локального порта на некоторый порт удалённой машины, выполните следующую команду:

```
$ ssh -L 3333:localhost:4444 user@remotehost
```

Теперь все пакеты, приходящие на локальную машину на порт 3333, будут перенаправлены на машину remotehost на порт 4444. Имя пользователя можно не указывать, если вы выполняете команду от имени этого пользователя. Внимание! Проброс привелегированных портов (до 1024) можно выполнять только от имени суперпользователя (root). Откроется ssh-сессия. Не закрывайте её, иначе перенаправление порта будет закончено!

Если вы хотите, чтобы все пакеты, пришедшие на некоторый порт удалённого сервера, приходили на некоторый порт локальной машины, выполните следующую команду:

```
$ ssh -R 4444:localhost:3333 user@remotehost
```

Теперь все пакеты, приходящие на машину remotehost на порт 4444, будут перенаправлены на порт 3333 локальной машины.

Возможна и организация VPN с помощью ssh. Делается это следующим образом:
На клиенте:

```
# ssh -f -w 0:1 192.168.1.15 true
# ifconfig tun0 10.1.1.1 10.1.1.2 netmask 255.255.255.252
# route add 10.0.99.0/24 10.1.1.2
```

На сервере:

```
# ifconfig tun1 10.1.1.2 10.1.1.1 netmask 255.255.255.252
# route add 10.0.50.0/24 10.1.1.1
```

где:

- 10.0.50.0/24 — клиентская подсеть
- 10.0.99.0/24 — серверная подсеть
- 10.1.1.1 и 10.1.1.2 — PtP-адреса клиента и сервера
- 192.168.1.15 — адрес шлюза в серверную подсеть

Внимание! На сервере в файле /etc/ssh/sshd_config опция PermitTunnel должна иметь значение «yes», «point-to-point» (для туннеля 3 уровня) или «ethernet» (для туннеля 2 уровня). Система должна поддерживать создание tun-устройств (должен присутствовать модуль ядра tun).

Безопасная передача файлов

Для передачи файла на удалённый компьютер выполните следующую команду:

```
$ scp somefile remotehost:/path/to/save/file
```

Если требуется указать имя пользователя, то выполните следующую команду:

```
$ scp somefile user@remotehost:/path/to/save/file
```

Если требуется передать каталог, то воспользуйтесь ключом -r:

```
$ scp -r somedir remotehost:/path/to/save/dir
```

Существует также файловая система sshfs пространства пользователя (fuse, Filesystem in USErspace), позволяющая подмонтировать уделённую файловую систему через ssh. Используется следующим образом:

```
$ sshfs [user@]remotehost:/fs/to/mount /mount/point
```

Эта команда выполняет монтирование удалённнной точки /fs/to/mount к точке монтирования /mount/point. Далее с удалёнными файлами можно работать как с частью локальной файловой системы.

Настройка сервера ssh

OpenSSH использует библиотеку tcp_wrappers для управления списком IP-адресов, с которых разрешено использовать SSH-сервер. Настройка производится в файле /etc/hosts.allow и /etc/hosts.deny:

Разрешающая запись:

```
sshd : 192.168.0.0/255.255.255.0 127.0.0.1 : ALLOW
sshd : ALL : ALLOW
```

Запрещающая запись:

```
sshd : ALL : DENY
```

Если невозможно составить список адресов, с которых вход разрешен, то приходится разрешать доступ с любого (ALL) адреса. Обратите внимание, что массовая атака на ssh-серверы – это обычное дело, поэтому позаботьтесь об усиленной защите ssh-сервера, принимающего запросы извне:

- ✓ следите за мощностью паролей пользователей, которым разрешено использовать ssh-сервер! Слабый пароль — по-прежнему основная уязвимость любой системы;
- ✓ запретите использование паролей и перейдите на использование 2048-битных RSA-ключей с парольными фразами;
- ✓ установите демон knockd, реализующий технологию Port Knocking, открытие определённого порта после определённой последовательности поступления запросов на другие порты;
- ✓ установите демон denyhosts, активно следящий за журналом аутентификации и вносящий изменения в файл hosts.deny;
- ✓ смените номер порта, на котором принимает соединения ssh-сервер;
- ✓ никогда, ни при каких условиях, ни в коем случае не включайте аутентификацию по имени или адресу клиента;
- ✓ при использовании ключей выбирайте большую длину ключа и мощную парольную фразу;
- ✓ запомните: основная брешь в безопасности системы – это администратор и пользователь. Подходите к настройке ssh-сервера с большой внимательностью и повышенным уровнем паранойи, так как ни один другой сервис при взломе не даёт моментального доступа к командному интерпретатору. Получение взломщиком приглашения интерпретатора можно считать полным крахом системы безопасности;
- ✓ следите за лентами обнаруженных уязвимостей; в OpenSSH порой обнаруживаются критические уязвимости (хотя как и любой другой демон, написанный в рамках OpenBSD /«Всего 2 удалённых уязвимости в поставке по умолчанию за 10 лет!»/, OpenSSH является сравнительно безопасным в плане удалённых уязвимостей)
- ✓ до сих пор не улеглись волнения после обнаружения модификации пакета OpenSSL в Debian, после которой openssl мог сгенерировать только 65536 уникальных ключей. Внимательно следите за «вменяемостью» команды, создающей дистрибутив, и за политикой создания пакетов (поставка «как есть», внесение незначительных изменений, полный аудит и пересмотр кода). Например, не стоит на сервер устанавливать Linspire или Linux XP. В то же время, в Slackware или Arch Linux по умолчанию на программы не накладываются патчи, изменяющие их поведение;
- ✓ в любом случае, устанавливайте программы только из официального репозитория или официального зеркала;

Конфигурационный файл — /etc/ssh/sshd_config. Рассмотрим пример такого файла[6]:

```
# Номер порта
Port 22
# Адрес, на котором ожидается соединение
ListenAddress 0.0.0.0
# Строгое разрешение использования только SSHv2
Protocol 2
# Время неактивности, через которое сессия будет сброшена
LoginGraceTime 2m
# Разрешить удалённый вход суперпользователя[7]
PermitRootLogin yes
# Максимальное количество попыток авторизации
MaxAuthTries 6
# Максимальное число одновременных ssh-сеансов
MaxSessions 10
# Запретить аутентификацию по имени клиента
HostbasedAuthentication no
# Не игнорировать пользовательские базы отпечатков ключей удалённых
# серверов
IgnoreUserKnownHosts no
# Запретить аутентификацию через файлы .rhosts и .shosts
IgnoreRhosts yes
# Включить аутентификацию по паролю
PasswordAuthentication yes
# Не разрешать пустые пароли
PermitEmptyPasswords no
# Использовать PAM для аутентификации
UsePAM yes
# Разрешить перенаправление портов
AllowTcpForwarding yes
# Разрешить удалённым машинам использовать перенаправление портов
GatewayPorts no
# Разрешить проброс X11
X11Forwarding no
# Проверять ключи через DNS (аписи SSHFP)
UseDNS yes
# Разрешать построение туннелей
PermitTunnel yes
# Включение SFTP-сервера
Subsystem       sftp    /usr/lib/ssh/sftp-server
```

Также возможно создание набора опций для конкретного пользователя или группы. Такие наборы предваряются следующей строкой:

```
Match User anoncvs
```

6 Будут рассмотрены только те ключи, значение которых следует изменить или может потребоваться изменить

7 Зачастую возможность удалённого входа суперпользователя отключают, используя непривилегированную учётную запись и sudo. Это правильно с точки зрения безопасности, однако сильный пароль суперпользователя, хранящийся в строгой тайне, это не отменяет

Далее идёт перечисление опций со значениями

Настройка авторизации по ключам

Для авторизации по ключам выполните следующие команды (от имени пользователя, для которого необходимо разрешить авторизацию по ключу):

```
# генерируем пару ключей
$ ssh-keygen -t rsa -b 2048
```

Будет запрошена парольная фраза. Нажмите «Enter», чтобы сгенерировать ключ без парольной фразы[8]. После генерации выполните следующую команду:

```
$ ssh-copy-id -i ~/.ssh/id_rsa remotehost
```

Эта команда скопирует ваш открытый ключ на компьютер remotehost и зарегистрирует его в ~/.ssh/authorized_keys пользователя, от имени которого был произведён удалённый вход.

8 Отсутствие парольной фразы может быть полезно, когда требуется выполнять некоторые действия на компьютере по расписанию без участия человека. Во всех остальных случаях отсутствие парольной фразы иметь оправдания не может. Ключ суперпользователя ОБЯЗАН иметь парольную фразу (а лучше вообще запретить удалённый вход суперпользователя и настроить sudo)

Сервисы безопасности

19. RADIUS-сервер freeradius

FreeRADIUS — это сервис, реализующий процедуры аутентификации, авторизации и аккаунтинга, предусмотренные протоколом RADIUS. FreeRADIUS может использоваться как сервер AAA в архитектуре IEEE 802.1X.

Конфигурационные файлы сервера находятся в каталоге /etc/raddb, копия этих настроек — в /etc/raddb.default.

Начиная с версии 2.0, FreeRADIUS стал поддерживать несколько сайтов (site, realm). Все настроенные сайты располагаются в каталоге sites-available, а в каталоге sites-enabled находятся символические ссылки на сайты из sites-available, которые требуется обслуживать.

FreeRADIUS — модульный сервер, что означает, что его функциональность разбита на несколько модулей, которые можно комбинировать при настройке сервера. Список настроечных файлов для модулей находится в каталоге modules.

Основные настройки сервера находятся в файле radiusd.conf, список локальных пользователей — в файле users, список RADIUS-клиентов (то есть серверов, обращающихся к FreeRADIUS для выполнения процедур AAA, например, точки доступа, VPN-серверы и так далее) — в файле clients.conf.

На постоянную работу сервер запускается с помощью команды /etc/rc.d/radiusd start, а в целях отладки можно запускать сервер командами radiusd -x или radiusd -X (меняется глубина отладки).

По умолчанию все журналы хранятся в каталоге /var/log/radius.

Рассматривая конфигурационные файлы, будем останавливаться только на самых важных настройках, так как остальные могут быть оставлены в состоянии по умолчанию.

Файл radiusd.conf:

```
# Секция, определяющая порт для пакетов аутентификации/авторизации

listen {

type = auth

# Адрес, на котором запущена аутентификационная часть FreeRADIUS

ipaddr = *

# Порт, на котором запущена аутентификационная часть FreeRADIUS

# Если 0, то используется порт, указанные в /etc/services (1812)

port = 0

}
```

```
# Секция, определяющая порт для пакетов аккаунтинга

listen {

type = acct

# Адрес, на котором запущена аккаунтинговая часть FreeRADIUS

ipaddr = *

# Порт, на котором запущена аккаунтинговая часть FreeRADIUS

# Если 0, то используется порт, указанные в /etc/services (1812)

port = 0

}

# Ключ, отвечающий за работу сервера в качестве прокси. Обычно не требуется

proxy_requests = no
```

Далее происходит подключение настроек из каталога sites-enabled. Рассмотрим пример такой настройки (файл default):

```
# Секция, определяющая процесс авторизации

# Содержит перечень модулей, которые последовательно выполняются

# влияя на процесс

# Ненужные вам модули комментируйте

authorize {

# Очистка данных от разного рода некорректных атрибутов

preprocess

# Если ключ Auth-Type не был установлен и мы имеем дело с CHAP-запросом,

# установить значение Auth-Type в CHAP

chap

# Если ключ Auth-Type не был установлен и мы имеем дело с MS-CHAP-запросом,

# установить значение Auth-Type в MS-CHAP

mschap

# Применить правила, зависимые от сайта (realm)

suffix

# Выполнить авторизацию через системные вызовы
```

```
# обращающиеся к файлам passwd и shadow
unix
# Выполнить авторизацию через файл users
files
}

# Секция, определяющая процесс аутентификации
# Указывается список модулей, доступных для проведения аутентификации
# Из них выбирается один в соответствии с типом,
# установленным на этапе authorize
authenticate {
# Для Auth-Type == PAP использовать модуль pap
Auth-Type PAP {
pap
}

# Для Auth-Type == CHAP использовать модуль chap
# Применяется чаще всего
# База паролей должна хранить пароли в прямом тексте (plaintext)
Auth-Type CHAP {
chap
}

# Для Auth-Type == MS-CHAP использовать модуль mschap
Auth-Type MS-CHAP {
mschap
}
```

```
# Использовать PAM

pam

# Использовать UNIX-аутентификацию

# Применяется только для обеспечения совместимости со старыми настройками

# Рекомендуется использовать PAP вместо UNIX-аутентификации

unix

}


# Секция, определяющая процесс выбора способа аккаунтинга

preacct {

preprocess

# Создать уникальный идентификатор для каждого запроса

acct_unique

# Применить правила, зависимые от сайта (realm)

suffix

# Прочитать файл acct_users

files

}


# Секция, определяющая процесс ведения журнала

accounting {

# Вести детальный журнал

detail

# Обновить информацию в файле wtmp

# Файл содержит сведения о пользователях, вошедших в систему

# Эту информацию можно получить с помощью команды last

unix

}
```

Файл clients.conf:

```
# Есть много способов описать RADIUS-клиента

# Опишем самый простой — явное указание адреса клиента

client 192.168.0.1 {

# Используемое для аутентификации ключевое слово

secret secret

# Краткое имя клиента

shortname testclient

}
```

Файл users:

Является обычным текстовым файлом, содержащим базу данных пользователей. Формат файла очень простой:

```
<имя_пользователя> Cleartext-Password := "пароль_пользователя"
```

Например:

```
student Cleartext-Password := "qwerty"
```

20. Клиент 802.1x wpa_supplicant

Утилита wpa_supplicant позволяет клиентам аутентифицироваться в проводных и беспроводных сетях, защищённых протоколами WEP, WPA, WPA2, WPA2-TKIP, WPA2-PSK, 802.1X (PAP, CHAP, EAP, EAP-TLS, EAP-MD5, LEAP). Поддерживает аппаратные ускорители генерации псевдослучайных последовательностей (через OpenSSL). Настройка клиента wpa_supplicant может осуществляться двумя способами:

1. Напрямую через конфигурационный файл.
2. С использованием графического интерфейса.

Настройка wpa_supplicant через конфигурационный файл

При данном способе работы сначала необходимо настроить файл конфигурации /etc/wpa_supplicant.conf, а затем только запустить клиент wpa_supplicant. Ниже приведён листинг конфигурационного файла wpa_supplicant, содержащий необходимый минимум параметров для аутентификации в сети, защищенной по протоколу 802.1X.

```
# Интерфейс управления клиентом для внешних программ
ctrl_interface=/var/run/wpa_supplicant
# Версия протокола IEEE 802.1X/EAPOL. Поддерживается также версия 2
eapol_version=1
# Режим автоматического сканирования и выбора точки доступа
ap_scan=1
# Включение быстрой повторной аутентификации EAP при обрыве связи
fast_reauth=1
# Блок описания сети
network={
      # Идентификатор сети
      ssid="mynetwork"
      # Протокол аутентификации
      key_mgmt=IEEE8021X
      # Вариант механизма аутентификации
      eap=MD5
      # Идентификатор пользователя
      identity="student"
      # Пути к пользовательским сертификатам
      ca_cert="/etc/cert/ca.pem"
      client_cert="/etc/cert/user.pem"
      private_key="/etc/cert/user.prv"
      # Пароль на закрытый ключ
      private_key_passwd="password"
      # Вместо сертификатов можно использовать пароль
      password="1234"
      eapol_flags=3
}
```

Чтобы запустить wpa-supplicant, выполните команду:

```
$ wpa_supplicant –i eth0 –D wired –c /etc/wpa_supplicant.conf
```

Описание ключей:

i – имя интерфейса, который подключен к защищенной сети
D – тип драйвера для работы с интерфейсом
c – расположение конфигурационного файла

Настройка wpa_supplicant с использованием графического интерфейса

Графический интерфейс запускается при работающей утилите wpa_supplicant командой wpa_gui (рисунок 4.10).

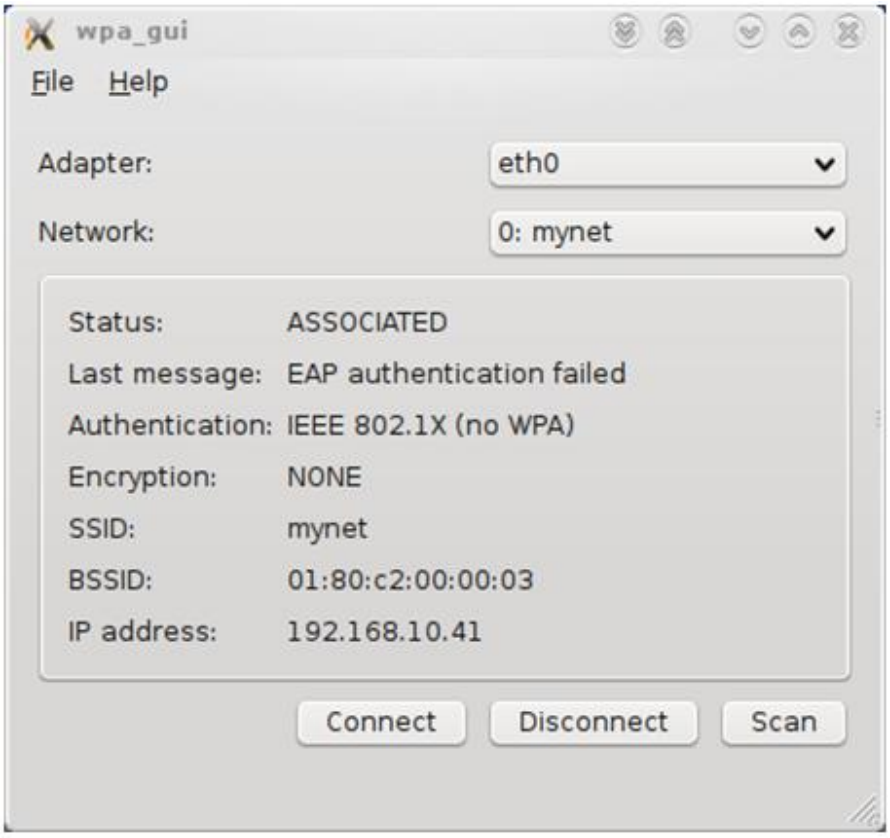


Рисунок 4.10. Главное окно wpa_gui.

Параметр «Adapter» задаёт сетевой интерфейс, через который будет происходить аутентификация. Параметр «Network» указывает, какой набор настроек следует использовать для аутентификации. Ниже расположено окно статуса, которое отображает:

- ✓ Status – текущий статус аутентификации
- ✓ Last message – последнее сообщение от демона wpa_supplicant
- ✓ Authentification – выбранный механизм аутентификации
- ✓ Encryption – выбранный механизм шифрования
- ✓ SSID – идентификатор сети (актуально для беспроводных сетей)
- ✓ BSSID – идентификатор сети (при построении беспроводной сети типа ad-hoc)
- ✓ IP address – адрес интерфейса, через который происходит аутентификация

Чтобы создать новый набор настроек аутентификации, воспользуйтесь меню File → Add Network (рисунок 4.11). Пункт меню File → Edit Network может использоваться для редактирования уже имеющихся наборов настроек. Пункт меню Event History покажет журнал событий wpa_supplicant.

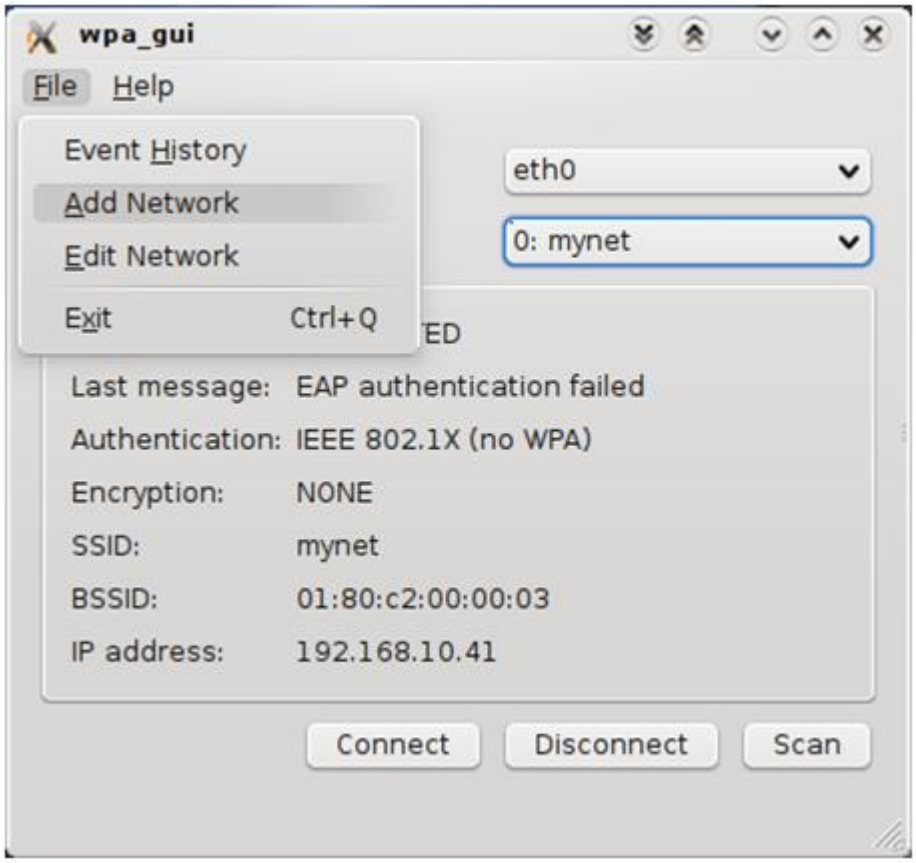


Рисунок 4.11. Структура меню File.

В открывшемся окне (рисунок 4.12) задаётся новый набор настроек аутентификации:

- ✓ SSID – идентификатор сети (актуально для беспроводных сетей)
- ✓ Authentification – метод аутентификации
- ✓ Encryption – метод шифрования
- ✓ PSK – секретный ключ (для метода WPA-PSK)
- ✓ EAP method – вариант EAP-аутентификации (для IEEE 802.1x)
- ✓ Identity – имя учётной записи
- ✓ Password – пароль
- ✓ CA Sertificate – корневорй сертификат (при использовании аутентификации по X.509-сертификатам)
- ✓ WEP keys – ключи для WEP-шифрования (при использовании метода шифрования WEP)

После того, как все необходимые поля будут заполнены, нажмите кнопку «Add».

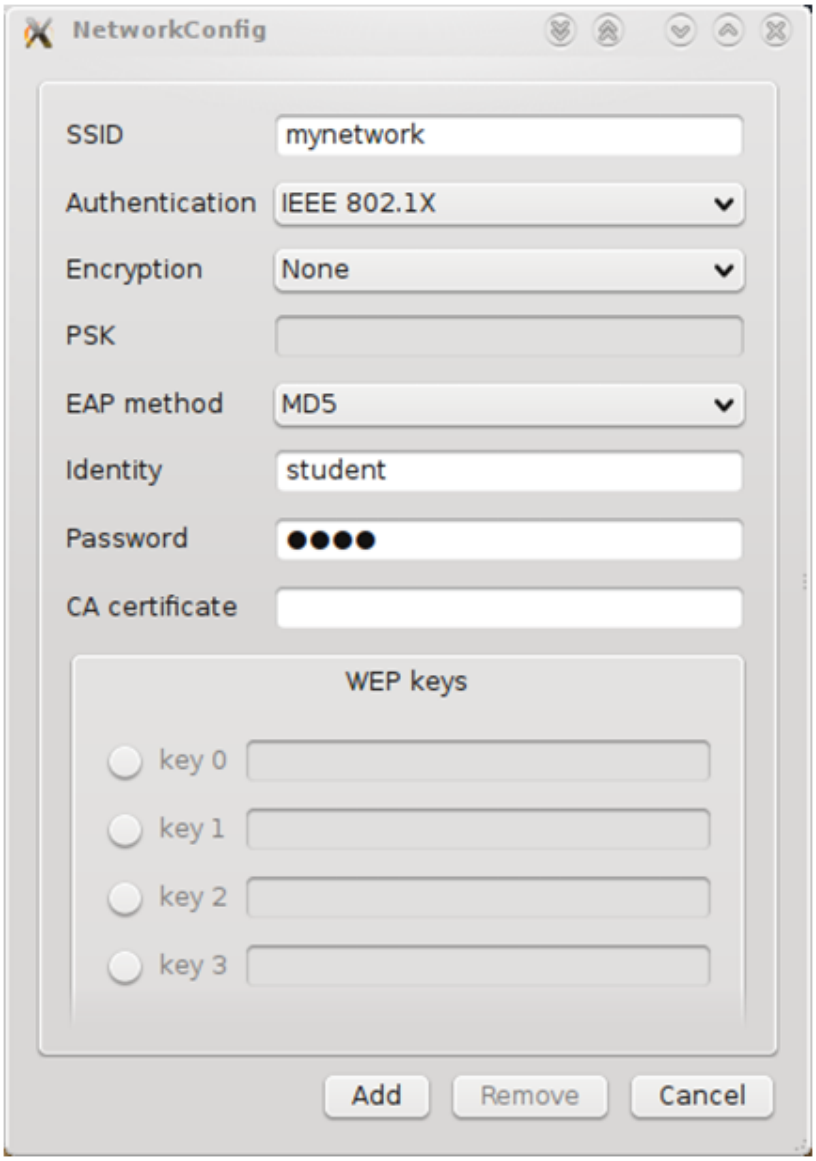


Рисунок 4.12. Диалог добавления сети.

Ниже на рисунке 4.13 схематично указана взаимосвязь параметров, настраиваемых на сервере freeradius, клиенте wpa_supplicant и коммутаторе.

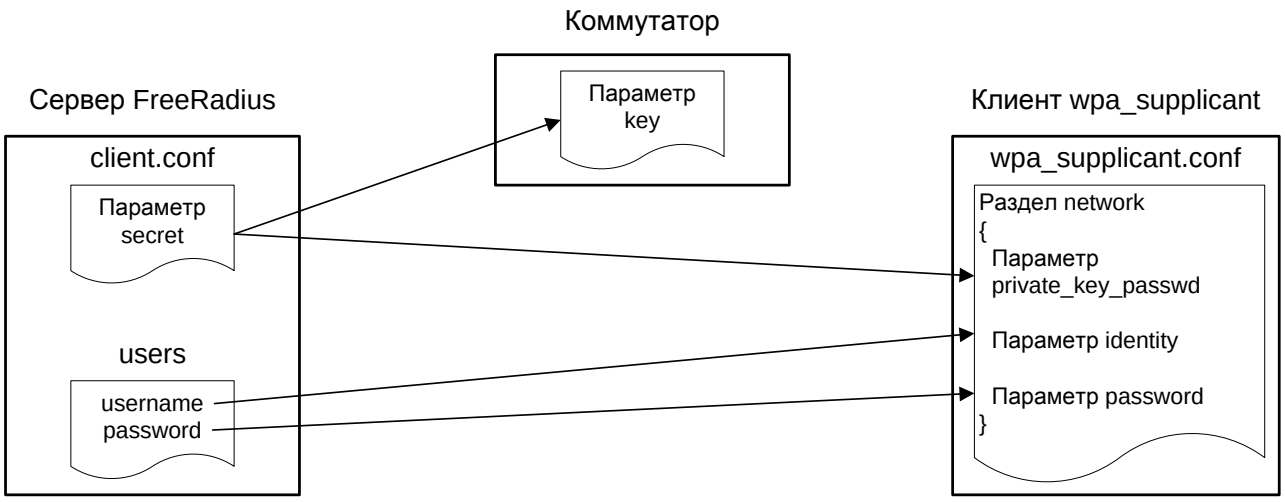


Рисунок 4.13.

21. Центр управления сертификатами на базе пакета OpenSSL

Центр управления сертификатами (Certificate Authority, CA) – это сервер, предназначенный для выдачи цифровых сертификатов для различных целей (например, для создания туннелей L2TP/IPSec или применения протокола HTTPS). Центр сертификации ведёт учёт выданных сертификатов и предоставляет возможность проверки действительности выданного сертификата. Для этого существуют 2 механизма:

1. Certificate Revocation Lists (CRL) – списки отозванных сертификатов. Для проверки сертификата на действительность необходимо запросить эти списки с центра сертификации. Проверка выполняется на стороне клиента.
2. Online Certificate Status Protocol (OCSP) – протокол проверки статуса сертификата. Для проверки сертификата необходимо связаться с сервером проверки по этому протоколу и передать индекс сертификата. Проверка выполняется на стороне сервера. Обычно диалог происходит поверх протокола HTTP. Все сообщения кодируются алгоритмом ASN.1.

Алгоритм создания нового подписанного сертификата:

1. Клиент создаёт запрос на подпись сертификата (Certificate Signing Request, CSR) и передаёт его в центр сертификации.
2. Центр сертификации подписывает запрос и выдаёт сертификат (публичный ключ в терминах алгоритмов асимметричного шифрования) и ключ (закрытый ключ), подписанные своим ключом (закрытым ключом).

При обращении клиента к некоторому сервису, защищённому выданным сертификатов, клиент может проверить истинность сертификата с помощью сертификата центра сертификации (естественно, этот сертификат должен иметься у клиента). Существуют так называемые самоподписанные сертификаты (Self-signed Certificate); это сертификаты, подписаные самим создателем.

Если требуются сертификаты исключительно для внутренних нужд организации, то самоподписанные сертификаты вполне жизнеспособны. Однако если вашими защищёнными SSL сервисами будут пользоваться сторонние лица, то они каждый раз будут получать предупреждение о самоподписанном сертификате, так как не имеют сертификата вашего центра сертификации. Существуют международные центры сертификации (VeriSign, Thawte и пр.), чьи сертификаты распознаются практически всеми ОС, однако их услуги платные. Также существует некоммерческий центр сертификации cacert.org, но выданные им сертификаты принимаются не очень большой частью клиентов.

Рассмотрим создание собственного центра сертификации:

1. Создание структуры каталогов:
   - ✓ перейдите в каталог /etc/ssl/ (cd /etc/ssl);
   - ✓ создайте каталог CA (mkdir CA).
2. Редактирование конфигурационного файла openssl.cnf:

```
# Задание домашнего каталога и файла, содержащего энтропию
HOME            = .
RANDFILE        = $ENV::HOME/.rnd
```

```
# Дополнительная информация о OBJECT IDENTIFIER:
[ new_oids ]
oid_section = new_oids

####################################################################
[ ca ]
default_ca = CA_default          # Секция, описывающая CA по умолчанию

####################################################################
[ CA_default ]

dir             = /etc/ssl/CA    # Каталог, содержащий CA
certs           = $dir/certs     # Каталог для хранения подписанных
                                 # сертификатов
crl_dir         = $dir/crl       # Каталог для хранения CRL
database        = $dir/index.txt# Индексный файл базы сертификатов
new_certs_dir   = $dir/newcerts # Каталог для новых сертификатов

certificate     = $dir/cacert.pem # Корневой сертификат
serial          = $dir/serial    # Текущий серийный номер сертификата
crlnumber       = $dir/crlnumber# Текущий серийный номер CRL
crl             = $dir/crl.pem  # CRL
private_key     = $dir/private/cakey.pem   # Ключ
RANDFILE        = $dir/private/.rand  # Файл,содержащий энтропию для
                                      # ключа
x509_extensions = usr_cert       # Расширения, добавляемые к
                                 # сертификату
default_days    = 365            # Время в днях, на которое выдаётся
                                 # сертификат
default_crl_days= 30             # Время в днях, через которое можно
                                 # создавать следующий CRL
default_md      = sha1           # Алгоритм, используемый для
                                 # формирования дайджеста

# здесь находятся настройки, значения которых по умолчанию оптимальны
# задание значения по умолчанию для некоторых полей сертификата
[ req_distinguished_name ]

# Страна
countryName           = Country Name (2 letter code)
countryName_default   = RU
countryName_min       = 2
countryName_max       = 2

# Штат/область
stateOrProvinceName          = State or Province Name (full name)
stateOrProvinceName_default = Chelyabinskaya Oblast

# Город
```

```
localityName                = Locality Name (eg, city)
localityName_default        = Chelyabinsk

# Название организации
0.organizationName          = Organization Name (eg, company)
0.organizationName_default = Uchtex-Profi

# Название подразделения
organizationalUnitName          =   Organizational   Unit   Name   (eg,
section)
organizationalUnitName_default  = telecom_department

# Имя субъекта сертификации (например, название службы)
# Для сертификатов, используемых в интернет-службах, значение этого
# поля должно совпадать с fqdn службы, поэтому значение по умолчанию
# не задаётся
commonName            = Common Name (eg, YOUR name)
commonName_max        = 64

# Почтовый адрес
emailAddress          = Email Address
emailAddress_max      = 454080, Chelyzbinsk, Lenin pr., 83

# Теперь перейдите в каталог /etc/ssl/CA. Создайте следующие объекты:
# Каталоги для хранения сертификатов, ключей и CRL
mkdir certs private crl newcerts

# Задание прав доступа к каталогу с ключами — доступ только для root
chmod g-rwx,o-rwx private

# Создание файла с серийным номером
echo '01' > serial

# Создание файла для индекса
touch index.txt
```

3. Создание корневого сертификата

Перейдите в каталог /etc/ssl/CA и выполните команду:

```
# openssl req -new -x509 -extensions v3_ca -keyout private/cakey.pem \
  -out cacert.pem -config ../openssl.cnf
```

(«\» - символ, соответствующий переносу строки в интерпретаторе bash)

Будет выдан запрос:

```
Enter PEM pass phrase:
```

Введите секретный ключ для защиты корневого сертификата. Далее будет выдан запрос на повторный ввод. Позднее будет выдана серия запросов на ввод описательных полей сертификата, значения по умолчанию к большей части которых были заданы в

конфигурационном файле. Для корневого сертификата полю Organizational Unit Name можно присвоить имя домена или имя ответственного за домен/центр сертификации. Соответственно заполните и поле Email Address. Полю Common Name можно присвоить значение Certificate Authority.

4. Создание сервисом запроса на сертификацию в ЦУС

Перейдите в каталог /etc/ssl/CA и выполните команду:

```
# openssl req -new -nodes -out req.pem -config ../openssl.cnf
```

Не забудьте указать в поле Common Name fqdn службы, для которой создаётся сертификат. Запрос на подписание будет записан в файл req.pem.

5. Подписание запроса

Перейдите в каталог /etc/ssl/CA и выполните команду:

```
# openssl ca -out cert.pem -config ./openssl.cnf -infiles req.pem
```

Будет выдан запрос парольной фразы для секретного ключа центра сертификации. Введите его. Будет выведен запрос на сертификацию в текстовом виде. Если вы хотите подписать его, введите «y» и нажмите «Enter». Новый сертификат — CA/newcerts/01.pem или CA/cert.pem. Закрытый ключ к сертификату — CA/privkey.pem. Теперь эти файлы можно использовать для защиты какого-либо сервиса с помощью SSL/TLS. Учтите, что если вы не сохраните закрытый ключ, то при выдаче следующего сертификата он будет перезаписан. Следующий выданный сертификат будет иметь номер 02 и так далее.

Некоторые программы (например, Stunnel) требуют, чтобы сертификат и закрытый ключ находились в одном файле. Чтобы получить такой файл, выполните следующую команду:

```
# cat privkey.pem cert.pem > key-cert.pem
```

Теперь файл key-cert.pem можно использовать в Stunnel, например:

```
# stunnel -p /etc/ssl/CA/key-cert.pem <прочие аргументы>
```

22. Шифратор TCP-соединения Stunnel

Программа Stunnel применяется в тех случаях, когда требуется зашифровать TCP-соединение, однако программа, работающая через это соединение, не поддерживает SSL/TLS. Рассмотрим пример запуска SSL-туннеля с помощью stunnel. Для функционирования stunnel требует конфигурационный файл stunnel.conf. Он имеет следующий формат:

```
[имя_туннеля1]
параметр = значение
параметр = значение
…

[имя_туннеля2]
параметр = значение
параметр = значение
…
```

Приведём простейший пример такого файла – инкапсулируем сервер IMAP в SSL:

```
[imapd]
# порт, на котором надо ожидать соединение
accept = 993

# программа, которую следует запустить при установлении соединения
# ВНИМАНИЕ! Таким образом, можно запустить только программу, которую
# можно запустить через inetd/xinetd
exec = /usr/sbin/imapd

# аргументы для запускаемой программы
# первым должно быть имя программы
execargs = imapd
```

Теперь запустим stunnel с этим конфигурационным файлом:

```
# stunnel stunnel.conf
```

При подключении к серверу на порт 993 после выполнения согласования будет запущена программа imapd, которая и обслужит imap-клиента.

23. Протокол PPPoE

Реализацией PPPOE-сервера/клиента в Linux является пакет rp-pppoe.

Настройка PPPOE-сервера

1. Конфигурационный файл /etc/ppp/pppoe-server-options:

```
# Включаем chap-аутентификацию
require-chap
# разрешаем изменять настройки ядра в случае необходимости (разрешение
# маршрутизации и пр.)
ktune
# Если пользователь есть в системной базе учётных записей – записать
# событие входа в журнал wtmp
login
# Время бездействия в секундах, после которого соединение закрывается
idle 1200
# Промежуток между отправками пакетов LCP echo-request
lcp-echo-interval 10
# Количество неполученных пакетов LCP echo-reply, после которого
# соединение сбрасывается
lcp-echo-failure 5
# Указываем адрес DNS-сервера, передаваемый клиенту
ms-dns 83.142.162.36
# Отключаем ненужные алгоритмы сжатия
nobsdcomp
novj
# Не отправляем и не получаем от второй точки туннеля endpoint
discriminator
# Позволяет избежать ошибок при ошибочном поведении противоположной
стороны
noendpoint
# Отключаем ненужный протокол IPX
noipx
# Указываем расположение файла с журналом
logfile /var/log/pppoe.log
```

2. Конфигурационный файл /etc/ppp/chap-secrets

Добавляем запись о пользователе в формате «пользователь * пароль»

```
user3   *  pass3
```

3. Запуск сервера

Запускаем PPPOE-сервер в режиме ядра на интерфейсе eth0. Адрес серверной точки туннеля – 192.168.0.1, адреса клиентских начинаются со 192.168.0.100. Выставляем имя службы в «PPPOE»:

```
# pppoe-server -k -I eth0 -L 192.168.0.1 -R 192.168.0.100 -S PPPOE
```

Проверить факт запуска сервера можно, выполнив команду:

```
# ps ax | grep pppoe
```

Также эта команда покажет активные в данный момент PPPOE-сессии.

Настройка PPPOE-клиента

Клиентская часть пакета rp-pppoe представлена следующими программами:

- ✓ pppoe-discovery – позволяет обнаружить доступные PPPOE-серверы
- ✓ pppoe-setup – позволяет в диалоговом режиме настроить PPPOE-сеодинение
- ✓ pppoe-start – запускает настроенное PPPOE-соединение
- ✓ pppoe-stop – прерывает PPPOE-сеодинение
- ✓ pppoe-status – выводит информацию о соединении и туннельном интерфейсе
- ✓ pppoe-connect – выполняет ту же функцию, что и pppoe-start, однако после выполнения не уходит в фоновый режим и выводит отладочные сообщения на экран

Запустите программу pppoe-setup для настройки соединения. Будут заданы следующие вопросы:

- ✓ Enter your PPPoE user name — введите имя пользователя для аутентификации на сервере (зависит от файла chap-secrets на сервере. Например, user1)
- ✓ Enter the Ethernet interface connected to the DSL modem — укажите интерфейс, на котором необходимо искать PPPOE-сервер (зависит от того, через какой интерфейс соединены клиент и сервер. Например, eth0)
- ✓ Enter the demand value — укажите, следует ли подключаться только в случае необходимости использования туннеля (yes) или необходимо сохранять постоянно активное соединение (no)
- ✓ Enter the DNS information here — либо введите «server» для использования настроек, переданных с сервера, либо укажите адрес необходимого сервера. Если оставить строку пустой, rp-pppoe не будет изменять системные настройки разрешения имён
- ✓ Please enter your PPPoE password — введите пароль, соответствующий имени пользователя (зависит от файла chap-secrets на сервере. Например, pass1). Запрос пароля будет выдан дважды
- ✓ Choose a type of firefall — rp-pppoe может задать некоторые настройки межсетевого экрана при установке соединения:
  - 0 — не изменять настройки межсетевого экрана
  - 1 — настройки для рабочей станции
  - 2 — настройки для интернет-шлюза
- ✓ Будут выведены сделанные настройки. Если вы согласны с ними — введите y и нажмите Enter

Теперь для установки PPPOE-соединия выполните команду pppoe-start.

24. Сервис динамической маршрутизации quagga

Quagga позволяет использовать протоколы динамической маршрутизации RIP, OSPF и BGP. Каждый протокол реализуется и поддерживается отдельным демоном (например, демон ospfd реализует протокол OSPF). Все демоны управляются общим демоном zebra и имеют telnet-интерфейс, позволяющий менять настройки маршрутизации во время работы демона. Конфигурационные файлы демонов хранятся в каталоге /etc/quagga.d/. Язык конфигурации сервисов quagga практически идентичен языку Cisco IOS, Запуск демонов производится следующим образом:

$ /etc/rc.d/<имя_демона> start

Имя_демона может быть следующим:

- ✓ ripd — протокол RIP
- ✓ ripngd — новая реализация протокола RIP
- ✓ ospfd — протокол OSPF
- ✓ ospf6d — протокол OSPF для Ipv6
- ✓ bgpd — протокол BGP

Для запуска демона необходимо составить минимальный конфигурационный ! файл, состоящий из директив hostname и password (см. ниже).

Любой демон, реализующий протокол маршрутизации, требует настроенного и запущенного демона zebra. Любая пустая строка в конфигурационном файле любого демона из набора quagga должна начинаться с символа "!". Любые символы после этого не рассматриваются демоном как значащие.

```
Пример настройки демона zebra:
! Задаём имя хоста, на котором выполняется демон
! Параметр обязательный, но значение является осведомительным
! для администратора и не влияет на демон
hostname Router
! Задаём пароль для доступа к демону через telnet
password zebra
! Задаём пароль для доступа к секции настроек
enable password zebra
! Секция описания интерфейсов
! Здесь можно задать текстовое описание интерфейса
! Зададим описание интерфейсу eth0
interface eth0
 description Internal Network
! Здесь же можно задать IP-адрес интерфейса
 ip address 192.168.0.1 255.255.255.0
! Здесь можно указать статические маршруты
! Зададим статический маршрут по умолчанию
ip route 0.0.0.0/0 203.181.89.241
! Здесь указывается имя журнала
log file zebra.log

Пример настройки демона ripd:
! Задаём имя хоста, на котором выполняется демон
! Параметр обязательный, но значение является осведомительным
! для администратора и не влияет на демон
hostname ripd
! Задаём пароль для доступа к демону через telnet
password zebra
! Указываем, что журнал должен выводиться на экран
log stdout
```

```
! Задаём параметры демона для интерфейса eth0
interface eth0
! Посылаем анонсы по протоколу RIPv1 и RIPv2
 ip rip send version 1 2
! Принимаем анонсы по протоколу RIPv1 и RIPv2
 ip rip receive version 1 2
! Задаём параметры RIP-маршрутизации
router rip
! Принимаем анонсы из указанной сети
 network 192.168.0.0/24
! Принимаем анонсы с указанного интерфейса
 network eth0

Пример настройки демона ospfd:
! Задаём имя хоста, на котором выполняется демон
! Параметр обязательный, но значение является осведомительным
! для администратора и не влияет на демон
hostname ospfd
! Задаём пароль для доступа к демону через telnet
password zebra
! Задаём пароль для доступа к секции настроек
enable password zebra
! Задаём параметры OSPF-маршрутизации
router ospf
! Принимаем анонсы из указанной сети для указанной зоны
 network 192.168.1.0/24 area 0
! Указываем, что журнал должен выводиться на экран
log stdout
```

После запуска управление демонами возможно через telnet. Порты, которые прослушивают демоны:

zebra 2601/tcp
ripd 2602/tcp (RIP)
ripngd 2603/tcp (RIPng)
ospfd 2604/tcp (OSPF)
bgpd 2605/tcp (BGP)
ospf6d 2606/tcp (OSPF для IPv6)

Подключиться к интерфейсу управления демоном можно, выполнив следующую команду:

$ telnet <имя_хоста> <порт>

Например, для подключения к демону ospfd на локальной машине, выполните:

$ telnet localhost 2604

После подключения демон запросит пароль, указанный в конфигурационном файле.

25. Протокол IPSec

В Linux реализацией L2TP/IPSec-сервера и клиента является OpenSwan (http://www.openswan.org). Вследствие природы протокола IPSec (ориентированность на одно соединение «точка-точка») настройки сервера и клиента не отличаются друг от друга. Надо отметить, что существуют 2 режима создания ipsec-канала:

- ✓ с фиксированными IP-адресами – режим для двух серверов, имеющих статические адреса;
- ✓ так называемый «road warrior» – режим для сервера с статическим адресом и клиента с динамическим адресом.

Отличие режима «road warrior» в том, что при настройке openswan для стороны, имеющей динамически адрес, необходимо указывать 0.0.0.0.

Рассмотрим настройку IPSec в режиме PSK (Pre-Shared Keys) для статических адресов. Не забывайте, что PSK ВСЕГДА используется только для аутентификации сторон, но не для шифрования.

Настройка ядра

С openswan поставляется файл (/etc/ipsec.d/examples/sysctl.conf), в котором описаны настройки ядра, не мешающие работе ipsec. Рассмотрим эти настройки:

```
# Включение маршрутизации
net.ipv4.ip_forward = 1

# rp_filter в некоторых случаях сбрасывает легальные пакеты после их
# дешифрования. Отключим его
net.ipv4.conf.default.rp_filter = 0

# Отключаем некоторые возможности ядра для предотвращения ошибок,
# например,в случае настройки нескольких подсетей на одном интерфейсе
# Отключение отправки сообщений ICMP redirect
net.ipv4.conf.all.send_redirects = 0
# Отключение журналирования некорректных (не по RFC1122) сообщений об
# ICMP-ошибках
net.ipv4.icmp_ignore_bogus_error_responses = 1
# Отключение журналирования пакетов с невозможными адресами источника
net.ipv4.conf.all.log_martians = 0

# Настройки, повышающие безопасность
# Отключаем маршрутизация от источника
net.ipv4.conf.default.accept_source_route = 0

# Отключаем реакцию на сообщение ICMP redirect
net.ipv4.conf.all.accept_redirects = 0
```

Применить эти настройки можно, записав их в конец файла /etc/sysctl.conf и выполнив sysctl -w, либо выполнять sysctl <настройка> для каждой вышеприведённой строки, например:

```
# sysctl net.ipv4.conf.all.accept_redirects=0
```

Также все эти настройки можно сделать с помощью команды echo и перенаправления вывода в файловой системе /proc, например:

```
# echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
```

Конфигурационный файл /etc/ipsec.conf

Содержит основные настройки openswan – базовые настройки и информацию о соединениях:

```
# начало секции общих настроек
config setup
        # отключаем NAT Traversal
        nat_traversal=no
        # Количество потоков, на которое необходимо распараллелить
        # сервер
        nhelpers=0

conn L2TP-PSK-noNAT
        # Режим аутентификации по ключу
        authby=secret
        # Отключение Perfect Forward Secrecy
        pfs=no
        # Задаём поведение сервера
        # auto=add - не устанавливать соединение автоматически
        # auto=start — устанавливать соединение при запуске openswan
        auto=add
        # Количество попыток согласования соединения
        keyingtries=3
        # Не производить переинициализацию соединения при истчении
        # максимального срока действия туннеля
        rekey=no
        # Тип туннеля — соединение хост-хост
        type=transport
        # Адрес стороны left
        left=192.168.0.100
        # Протокол и порт для стороны left
        leftprotoport=17/0
        # Адрес стороны right
        right=192.168.0.101
        # Протокол и порт для стороны right
        rightprotoport=17/0
# Отключаем оппортунстическое шифрование (opportunistic encryption)
include /etc/ipsec.d/examples/no_oe.conf
```

Конфигурационный файл /etc/ipsec.secrets

В этом файле содержится общий ключ. Рекомендуется генерировать этот файл командой ipsec ranbits, Следующая команда сгенерирует произвольный ключ длинной 128 бит:

```
# ipsec ranbits 128 > /etc/ipsec.secrets
```

Теперь откройте /etc/ipsec.secrets любым текстовым редактором и добавьте в самое начало файла (в одну строку с ключом) символы «: PSK» и заключите ключ в кавычки. Файл должен выглядеть следующим образом:

```
: PSK "0xf2c28329_2b679d28_714c05f5_a6949859"
```

Запуск openswan и установление соединения

Перед установлением соединения запустите openswan следующей командой:

```
# /etc/rc.d/openswan start
```

Теперь установите соединение, выполнив на одной из сторон команду:

```
# ipsec auto --up L2TP-PSK-noNAT
```

Вы должны увидеть примерно следующие строки:

```
117 "L2TP-PSK-noNAT" #5: STATE_QUICK_I1: initiate
004 "L2TP-PSK-noNAT" #5: STATE_QUICK_I2: sent QI2, IPsec SA
established {ESP=>0xb91c643a <0xc9cf00a3 xfrm=AES_0-HMAC_SHA1
NATD=none DPD=none}
```

26. Протокол PPTP

Настройка сервера PPTP

В Linux реализацией сервера PPTP является poptop (http://www.poptop.org/). Рассмотрим его настройку.

1. Конфигурационный файл /etc/pptpd.conf:

```
# Расположение файла с настройками
option /etc/ppp/options.pptp

# IP-адрес сервера из подсети, в которой находятся адреса для клиентов
localip 192.168.1.1

# Диапазон адресов для клиентов
remoteip 192.168.1.200-254
```

2. Конфигурационный файл /etc/ppp/options.pptp:

```
# Даём название серверу
name PPTP

# Даём название набору настроек
ipparam pptpd

# Самоаутентификация не требуется
noauth
# Блокируем порт
lock
# Запрещаем использование ненадёжных протоколов аутентификации
refuse-pap
refuse-eap
refuse-chap
refuse-mschap

# отключаем алгоритмы сжатия, которые не будут использоваться
nobsdcomp
nodeflate
 Используем шифрование
reuire-mschap-v2
require-mppe-128
# Передаём адрес DNS-сервера клиенту
ms-dns 192.168.1.1
```

3. Конфигурационный файл /etc/ppp/chap-secrets:

```
# Если пользователь должен динамически получать IP-адрес
# из диапазона remoteip в pptpd.conf:
user1    PPTP    password1        "*"

# Если мы хотим привязать определённый IP к логину:
user2    PPTP    password2        "192.168.1.101"
```

4. Если на сервере установлен межсетевой экран, то необходимо сделать следующие добавки:

```
# Разрешаем протокол GRE для всех;
iptables -A INPUT -p gre -j ACCEPT

# Разрешаем соединение с PPTP-сервером для всех;
```

```
iptables -A INPUT -p tcp --dport 1723 -j ACCEPT
```

5. Запускаем сервер:

```
# /etc/rc.d/pptpd start
```

Настройка клиента PPTP

В Linux реализацией клиента PPTP является pptpclient (http://pptpclient.sourceforge.net/). Рассмотрим его настройку.

1. Конфигурационный файл /etc/ppp/options.pptp

```
lock
noauth
refuse-pap
refuse-eap
refuse-chap
refuse-mschap
require-mschap-v2
require-mppe-128
nobsdcomp
nodeflate
defaultroute     # эта опция не даёт эффекта, если уже есть маршрут по
                 # умолчанию
```

2. Конфигурационный файл /etc/ppp/peers/pptpvpn

```
name user1
ipparam pptpvpn
remotename PPTP
lock
noauth
require-mschap-v2
require-mppe-128
# 192.168.0.1 — адрес PPTP-сервера
pty «pptp 192.168.0.1 --nolaunchpppd» usepeerdns
# использование серверов DNS, переданных PPTP-сервером
file /etc/ppp/options.pptp
```

3. Запускаем PPTP-соединение

```
# pon pptpvpn
```

4. Разрываем PPTP-соединение

```
# poff
```

27. Утилита iptables

Утилита iptables предназначена для управления межсетевым экраном netfilter, встроенным в ядро Linux.

Порядок прохождения таблиц и цепочек

Когда пакет приходит на наш брандмауэр, то он сперва попадает на сетевое устройство, перехватывается соответствующим драйвером и далее передается в ядро. Далее пакет проходит ряд таблиц и затем передается либо локальному приложению, либо переправляется на другую машину. Порядок следования пакета приводится в таблице 5.1.

Таблица 5.1 – Порядок движения транзитных пакетов

Шаг	Таблица	Цепочка	Примечание
1			Кабель
2			Сетевой интерфейс
3	mangle	PREROUTING	Обычно эта цепочка используется для внесения изменений в заголовок пакета, например, для изменения битов *TOS* и пр.
4	nat	PREROUTING	Эта цепочка используется для трансляции сетевых адресов (*Destination Network Address Translation*). *Source Network Address Translation* выполняется позднее, в другой цепочке. Любого рода фильтрация в этой цепочке может производиться только в исключительных случаях.
5			Принятие решения о дальнейшей маршрутизации, то есть в этой точке решается, куда пойдет пакет – локальному приложению или на другой узел сети.
6	mangle	FORWARD	Далее пакет попадает в цепочку *FORWARD* таблицы mangle, которая должна использоваться только в исключительных случаях, когда необходимо внести некоторые изменения в заголовок пакета между двумя точками принятия решения о маршрутизации.
7	Filter	FORWARD	В цепочку *FORWARD* попадают только те пакеты, которые идут на другой хост. Вся фильтрация транзитного трафика должна выполняться здесь. Не забывайте, что через эту цепочку проходит трафик в обоих направлениях, обязательно учитывайте это обстоятельство при написании правил фильтрации.
8	mangle	POSTROUTING	Эта цепочка предназначена для внесения изменений в заголовок пакета уже после того как принято последнее решение о маршрутизации.
9	nat	POSTROUTING	Эта цепочка предназначена в первую очередь для *Source Network Address Translation*. Не используйте ее для фильтрации без особой на то

Шаг	Таблица	Цепочка	Примечание
			необходимости. Здесь же выполняется и маскарадинг (Masquerading).
10			Выходной сетевой.
11			Кабель.

Как вы можете видеть, пакет проходит несколько этапов, прежде чем он будет передан далее. На каждом из них пакет может быть остановлен. Заметьте, что нет каких либо цепочек, специфичных для отдельных интерфейсов. Цепочку *FORWARD* проходят ВСЕ пакеты, которые движутся через устройство. Не используйте цепочку INPUT для фильтрации транзитных пакетов, они туда просто не попадают! Через эту цепочку движутся только те пакеты, которые предназначены данному хосту!

В таблице 5.2 рассмотрен порядок движения пакета, предназначенного локальному процессу/приложению.

Таблица 5.2 – Для локального приложения

Шаг	Таблица	Цепочка	Примечание
1			Кабель.
2			Входной сетевой интерфейс.
3	mangle	PREROUTING	Обычно используется для внесения изменений в заголовок пакета, например для установки битов *TOS* и пр.
4	nat	PREROUTING	Преобразование адресов (*Destination Network Address Translation*). Фильтрация пакетов здесь допускается только в исключительных случаях.
5			Принятие решения о маршрутизации.
6	mangle	INPUT	Пакет попадает в цепочку *INPUT* таблицы mangle. Здесь вносятся изменения в заголовок пакета перед тем, как он будет передан локальному приложению.
7	filter	INPUT	Здесь производится фильтрация входящего трафика. Помните, что все входящие пакеты, адресованные нам, проходят через эту цепочку, независимо от того с какого интерфейса они поступили.
8			Локальный процесс/приложение.

Важно помнить, что на этот раз пакеты идут через цепочку INPUT, а не через FORWARD.
В заключение в таблице XX приведён порядок движения пакетов, созданных локальными процессами.

Таблица X – от локальных процессов

Шаг	Таблица	Цепочка	Примечание
1			Локальный процесс.
2			Принятие решения о маршрутизации. Здесь решается, куда пойдет пакет дальше – на какой адрес, через какой сетевой интерфейс и пр.
3	mangle	OUTPUT	Здесь производится внесение изменений в

Шаг	Таблица	Цепочка	Примечание
			заголовок пакета. Выполнение фильтрации в этой цепочке может иметь негативные последствия.
4	nat	OUTPUT	Эта цепочка используется для трансляции сетевых адресов (NAT) в пакетах, исходящих от локальных процессов брандмауэра.
5	Filter	OUTPUT	Здесь фильтруется исходящий трафик.
6	mangle	POSTROUTING	Цепочка *POSTROUTING* таблицы mangle в основном используется для правил, которые должны вносить изменения в заголовок пакета перед тем, как он покинет брандмауэр, но уже после принятия решения о маршрутизации. В эту цепочку попадают все пакеты, как транзитные, так и созданные локальными процессами брандмауэра.
7	nat	POSTROUTING	Здесь выполняется *Source Network Address Translation*. Не следует в этой цепочке производить фильтрацию пакетов во избежание нежелательных побочных эффектов. Однако и здесь можно останавливать пакеты, применяя политику по-умолчанию DROP.
8			Сетевой интерфейс.
9			Кабель.

Таким образом, есть три различных варианта прохождения пакетов. Рисунок 4.14 более наглядно демонстрирует это.

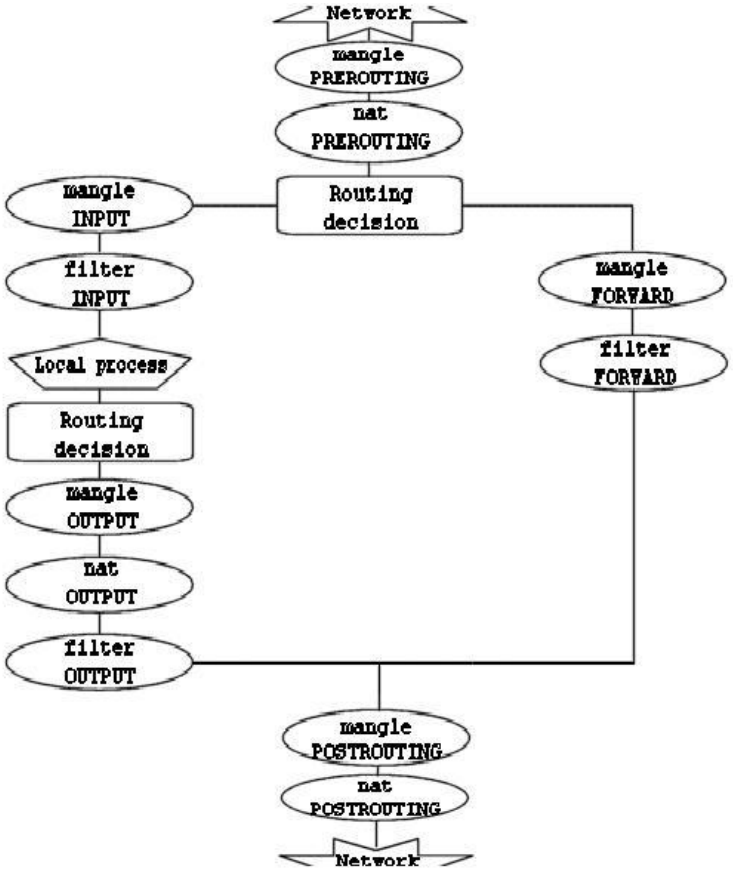


Рисунок 4.14.

Пакеты, с адресом назначения на сетевое устройство, могут претерпеть изменение сетевого адреса назначения (DNAT) в цепочке *PREROUTING* таблицы nat и соответственно дальнейшая маршрутизация в первой точке будет выполняться в зависимости от произведенных изменений.

Таблица Mangle

Эта таблица предназначена главным образом для внесения изменений в заголовки пакетов («mangle» с англ. «искажать», «изменять»). То есть в этой таблице вы можете устанавливать биты *TOS* (Type Of Service) и т.д. В этой таблице не следует производить любого рода фильтрацию, маскировку или преобразование адресов (DNAT, SNAT, MASQUERADE).

В этой таблице допускается выполнять только нижеперечисленные действия:

- ✓ TOS
- ✓ TTL
- ✓ MARK

Действие TOS выполняет установку битов поля *Type of Service* в пакете. Действие TTL используется для установки значения поля *TTL* (Time To Live) пакета. Действие MARK устанавливает специальную метку на пакет, которая затем может быть проверена другими правилами в iptables или другими программами, например iproute2. С помощью "меток" можно управлять маршрутизацией пакетов, ограничивать трафик и т.п.

Таблица Nat

Эта таблица используется для выполнения преобразований сетевых адресов *NAT* (Network Address Translation). Как уже упоминалось ранее, только первый пакет из потока проходит через цепочки этой таблицы, трансляция адресов или маскировка применяются ко всем последующим пакетам в потоке автоматически. Для этой таблицы характерны действия:

- ✓ DNAT
- ✓ SNAT
- ✓ MASQUERADE

Действие DNAT (Destination Network Address Translation) производит преобразование адресов назначения в заголовках пакетов. SNAT (Source Network Address Translation) используется для изменения исходных адресов пакетов. Маскировка (MASQUERADE) применяется в тех же целях, что и SNAT, но в отличие от последней, MASQUERADE дает более сильную нагрузку на систему. Происходит это потому, что каждый раз, когда требуется выполнение этого действия – производится запрос IP адреса для указанного в действии сетевого интерфейса, в то время как для SNAT IP адрес указывается непосредственно. Однако, благодаря такому отличию, MASQUERADE может работать в случаях с динамическим IP адресом.

Таблица Filter

Как следует из названия, в этой таблице должны содержаться наборы правил для выполнения фильтрации пакетов. Пакеты могут пропускаться далее, либо отвергаться (действия ACCEPT и DROP соответственно), в зависимости от их содержимого. Конечно же, можно отфильтровывать пакеты и в других таблицах, но эта таблица существует именно для нужд фильтрации. В этой таблице допускается использование большинства из существующих действий, однако ряд действий, которые рассмотрен выше в этой главе, должен выполняться только в присущих им таблицах.

Как строить правила

Каждое правило – это строка, содержащая в себе критерии определяющие, подпадает ли пакет под заданное правило, и действие, которое необходимо выполнить в случае выполнения критерия. В общем виде правила записываются примерно так:

```
iptables [-t table] command [match] [target/jump]
```

Нигде не утверждается, что описание действия (target/jump) должно стоять последним в строке, однако, такая нотация более удобочитаема. Как бы то ни было, но чаще всего будет встречаться именно такой способ записи правил.

Если в правило не включается спецификатор [-t table], то по умолчанию предполагается использование таблицы *filter*, если же предполагается использование другой таблицы, то это требуется указать явно. Спецификатор таблицы так же можно указывать в любом месте строки правила, однако более или менее стандартом считается указание таблицы в начале правила.

Далее, непосредственно за именем таблицы, должна стоять команда. Если спецификатора таблицы нет, то команда всегда должна стоять первой. Команда определяет действие iptables, например: вставить правило, или добавить правило в конец цепочки, или удалить правило и т.п.

Раздел match задает критерии проверки, по которым определяется подпадает ли пакет под действие этого правила или нет. Здесь можно указать самые разные критерии – IP-адрес источника пакета или сети, IP-адрес места назначения, порт, протокол, сетевой интерфейс и т.д.

target указывает, какое действие должно быть выполнено при условии выполнения критериев в правиле. Здесь можно заставить ядро передать пакет в другую цепочку правил, "сбросить" пакет и забыть про него, выдать на источник сообщение об ошибке и т.п.

Таблицы

Опция -t указывает на используемую таблицу. По умолчанию используется таблица *filter*. С ключом -t применяются следующие опции.

Таблица 5.3 – Таблицы

Таблица	Описание
nat	Таблица *nat* используется главным образом для преобразования сетевых адресов (*Network Address Translation*). Через эту таблицу проходит только первый пакет из потока. Преобразования адресов автоматически применяется ко всем последующим пакетам. Это один из факторов, исходя из которых нельзя осуществлять какую-либо фильтрацию в этой таблице. Цепочка *PREROUTING* используется для внесения изменений в пакеты на входе в брандмауэр. Цепочка *OUTPUT* используется для преобразования адресов в пакетах, созданных приложениями внутри брандмауэра, перед принятием решения о маршрутизации. И последняя цепочка в этой таблице – *POSTROUTING*, которая используется для преобразования пакетов перед выдачей их в сеть.
mangle	Эта таблица используется для внесения изменений в заголовки пакетов. Примером может служить изменение поля TTL, TOS или MARK. Важно: в действительности поле MARK не изменяется, но в памяти ядра заводится структура, которая сопровождает данный пакет все время его прохождения через брандмауэр, так что другие правила и приложения на данном брандмауэре (и только на нём) могут использовать это поле в своих целях. Таблица имеет пять цепочек *PREROUTING*, *POSTROUTING*, *INPUT*, *OUTPUT* и *FORWARD*. *PREROUTING* используется для внесения изменений на входе в брандмауэр, перед принятием решения о

Таблица	Описание
	маршрутизации. *POSTROUTING* используется для внесения изменений на выходе из брандмауэра, после принятия решения о маршрутизации. *INPUT* – для внесения изменений в пакеты перед тем как они будут переданы локальному приложению внутри брандмауэра. *OUTPUT* – для внесения изменений в пакеты, поступающие от приложений внутри брандмауэра. *FORWARD* – для внесения изменений в транзитные пакеты после первого принятия решения о маршрутизации, но перед последним принятием решения о маршрутизации. Таблица *mangle* ни в коем случае не должна использоваться для преобразования сетевых адресов или маскарадинга (*Network Address Translation, Masquerading*), поскольку для этих целей имеется таблица nat.
filter	Таблица *filter* используется главным образом для фильтрации пакетов. Здесь можно выполнить DROP, LOG, ACCEPT или REJECT без каких либо ограничений, которые имеются в других таблицах. Имеется три встроенных цепочки. Первая – *FORWARD*, используемая для фильтрации пакетов, идущих транзитом через брандмауэр. Цепочку *INPUT* проходят пакеты, которые предназначены локальным приложениям. Цепочка *OUTPUT* – используется для фильтрации исходящих пакетов, сгенерированных приложениями на самом брандмауэре.

Команды

Посредством команд администратор сообщает iptables, что он хочет сделать. Обычно предполагается одно из двух действий: добавление нового правила в цепочку или удаление существующего правила из той или иной таблицы. Далее приведены основные команды, которые используются в iptables**.**

Таблица 5.4 – Команды

Команда	-A**,** --append
Пример	iptables -A INPUT ...
Описание	Добавляет новое правило в конец заданной цепочки.
Команда	-D**,** --delete
Пример	iptables -D INPUT --dport 80 -j DROP**,** iptables -D INPUT 1
Описание	Удаление правила из цепочки. Команда имеет два формата записи, первый – когда задается критерий сравнения с опцией -D (см. первый пример), второй – порядковый номер правила. Если задается критерий сравнения, то удаляется правило, которое имеет в себе этот критерий, если задается номер правила, то будет удалено правило с заданным номером. Счет правил в цепочках начинается с 1.
Команда	-R, --replace
Пример	iptables -R INPUT 1 -s 192.168.0.1 -j DROP
Описание	Эта команда заменяет одно правило другим. В основном она используется во время отладки новых правил.
Команда	-I, --insert
Пример	iptables -I INPUT 1 --dport 80 -j ACCEPT
Описание	Вставляет новое правило в цепочку. Число, следующее за именем цепочки, указывает номер правила, перед которым нужно вставить новое правило, другими словами число задает номер для вставляемого правила. В примере выше, указывается, что данное правило должно быть 1-м в цепочке *INPUT*.
Команда	-L, --list

Пример	iptables -L INPUT
Описание	Вывод списка правил в заданной цепочке, в данном примере предполагается вывод правил из цепочки *INPUT*. Если имя цепочки не указывается, то выводится список правил для всех цепочек. Формат вывода зависит от наличия дополнительных ключей в команде, например -n, -v, и пр.
Команда	-F, --flush
Пример	iptables -F INPUT
Описание	Сброс (удаление) всех правил из заданной цепочки (таблицы). Если имя цепочки и таблицы не указывается, то удаляются все правила, во всех цепочках. Если не указана таблица ключом *-t (--table)*, то очистка цепочек производится только в таблице filter.
Команда	-Z, --zero
Пример	iptables -Z INPUT
Описание	Обнуление всех счетчиков в заданной цепочке. Если имя цепочки не указывается, то подразумеваются все цепочки. При использовании ключа -v совместно с командой -L, на вывод будут поданы и состояния счетчиков пакетов, попавших под действие каждого правила. Допускается совместное использование команд -L и -Z. В этом случае будет выдан сначала список правил со счетчиками, а затем произойдет обнуление счетчиков.
Команда	-N, --new-chain
Пример	iptables -N allowed
Описание	Создается новая цепочка с заданным именем в заданной таблице. В выше приведенном примере создается новая цепочка с именем allowed. Имя цепочки должно быть уникальным и не должно совпадать с зарезервированными именами цепочек и действий (такими как DROP, REJECT и т.п.)
Команда	-X, --delete-chain
Пример	iptables -X allowed
Описание	Удаление заданной цепочки из заданной таблицы. Удаляемая цепочка не должна иметь правил и не должно быть ссылок из других цепочек на удаляемую цепочку. Если имя цепочки не указано, то будут удалены все цепочки заданной таблице, кроме встроенных.
Команда	-P, --policy
Пример	iptables -P INPUT DROP
Описание	Задает политику по-умолчанию для заданной цепочки. Политика по-умолчанию определяет действие, применяемое к пакетам, не попавшим под действие ни одного из правил в цепочке. В качестве политики по умолчанию допускается использовать DROP и ACCEPT.
Команда	-E, --rename-chain
Пример	iptables -E allowed disallowed
Описание	Команда -E выполняет переименование пользовательской цепочки. В примере цепочка allowed будет переименована в цепочку disallowed. Эти переименования не изменяют порядок работы, а носят только косметический характер.

Команда должна быть указана всегда. Список доступных команд можно просмотреть с помощью команды iptables -h. Некоторые команды могут использоваться совместно с дополнительными ключами.

Действия и переходы

Действия и переходы сообщают правилу, что необходимо выполнить, если пакет соответствует заданному критерию. Чаще всего употребляются действия ACCEPT и DROP.

Описание переходов в правилах выглядит точно так же как и описание действий, то есть ставится ключ -j и указывается название цепочки правил, на которую выполняется переход. На переходы накладывается ряд ограничений: первое – цепочка, на которую выполняется переход, должна находиться в той же таблице, что и цепочка, из которой этот переход выполняется; второе – цепочка, являющаяся целью перехода должна быть создана до того как на нее будут выполняться переходы.

Если пакет достиг конца цепочки то он будет возвращен в вызывающую цепочку и движение пакета продолжится с правила, следующего за правилом, вызвавшем переход. Если к пакету во вложенной цепочке будет применено действие ACCEPT, то автоматически пакет будет считаться принятым и в вызывающей цепочке и уже не будет продолжать движение по вызывающим цепочкам. Однако пакет пойдет по другим цепочкам в других таблицах.

Действие – это предопределенная команда, описывающая действие, которое необходимо выполнить, если пакет совпал с заданным критерием. Например, можно применить действие DROP или ACCEPT к пакету. В результате выполнения одних действий, пакет прекращает свое прохождение по цепочке, например DROP и ACCEPT, в результате других, после выполнения неких операций, продолжает проверку, например, LOG, в результате работы третьих даже видоизменяется, например DNAT и SNAT, TTL и TOS, но также продолжает продвижение по цепочке.

Действие ACCEPT

Данная операция не имеет дополнительных ключей. Если над пакетом выполняется действие ACCEPT, то пакет прекращает движение по цепочке (и всем вызвавшим цепочкам, если текущая цепочка была вложенной) и считается ПРИНЯТЫМ (то есть пропускается), тем не менее, пакет продолжит движение по цепочкам в других таблицах и может быть отвергнут там.

Действие DNAT

DNAT (Destination Network Address Translation) используется для преобразования адреса места назначения в IP заголовке пакета. Если пакет подпадает под критерий правила, выполняющего DNAT, то этот пакет, и все последующие пакеты из этого же потока, будут подвергнуты преобразованию адреса назначения и переданы на требуемое устройство, хост или сеть. Для этого действия так же можно указать диапазон адресов, тогда выбор адреса назначения для каждого нового потока будет производиться случайным образом.

Действие DNAT может выполняться только в цепочках PREROUTING и OUTPUT таблицы nat, и во вложенных подцепочках. Важно запомнить, что вложенные подцепочки, реализующие DNAT не должны вызываться из других цепочек, кроме PREROUTING и OUTPUT. Действие DNAT используется совместно с ключом --to-destination.

Действие DROP

Данное действие просто "сбрасывает" пакет и iptables "забывает" о его существовании. "Сброшенные" пакеты прекращают свое движение полностью, то есть они не передаются в другие таблицы, как это происходит в случае с действием ACCEPT. Следует помнить, что данное действие может иметь негативные последствия, поскольку может оставлять незакрытые "мертвые" сокеты как на стороне сервера, так и на стороне клиента, наилучшим способом защиты будет использование действия REJECT особенно при защите от сканирования портов.

Действие LOG

LOG – действие, которое служит для журналирования отдельных пакетов и событий. В журнал могут заноситься заголовки IP пакетов и другая информация. Информация из журнала может быть затем прочитана с помощью dmesg или syslogd, либо с помощью других программ. Обратите ваше внимание так же на действие ULOG.

Действие MARK

Используется для установки меток для определенных пакетов. Это действие может выполняться только в пределах таблицы *mangle*. Установка меток обычно используется для нужд маршрутизации пакетов по различным маршрутам, для ограничения трафика и т.п.

Действие MASQUERADE

Маскарадинг в основе своей представляет то же самое, что и SNAT только не имеет ключа --to-source. Если у вас имеется динамическое подключение, то нужно использовать маскарадинг, если же у вас статическое IP подключение, то бесспорно лучшим выходом будет использование действия SNAT.

Маскарадинг подразумевает получение IP адреса от заданного сетевого интерфейса, вместо прямого его указания, как это делается с помощью ключа --to-source в действии SNAT.

Действие MASQUERADE допускается указывать только в цепочке *POSTROUTING* таблицы nat, так же как и действие SNAT.

Действие REDIRECT

Выполняет перенаправление пакетов и потоков на другой порт той же самой машины. К примеру, можно пакеты, поступающие на HTTP порт перенаправить на порт HTTP proxy. Действие REDIRECT очень удобно для выполнения "прозрачного" проксирования (transparent proxying), когда машины в локальной сети даже не подозревают о существовании прокси-сервера. REDIRECT может использоваться только в цепочках *PREROUTING* и *OUTPUT* таблицы nat. Для действия REDIRECT предусмотрен только один ключ –to-ports.

Действие REJECT

REJECT используется, как правило, в тех же самых ситуациях, что и DROP, но в отличие от DROP, команда REJECT выдает ICMP-сообщение об ошибке на хост, передавший пакет. Действие REJECT на сегодняшний день может использоваться только в цепочках *INPUT*, *FORWARD* и *OUTPUT* (и во вложенных в них цепочках).

Действие SNAT

SNAT используется для преобразования сетевых адресов (Source Network Address Translation), SNAT допускается выполнять только в таблице nat, в цепочке *POSTROUTING*. Если первый пакет в соединении подвергся преобразованию исходящего адреса, то все последующие пакеты, из этого же соединения, будут преобразованы автоматически и не пойдут через эту цепочку правил. Используется с ключом --to-source.

28. Система обнаружения вторжений Snort

Сетевая система обнаружений Snort была разработана Мартином Рош (Martin Roesch) как упрощенная СОВ в виде сетевого анализатора (www.snort.org). Благодаря постоянным усовершенствованиям и расширениям Snort все более превращается в полнофункциональную систему анализа IP-трафика и записи пакетов в реальном времени. По своему замыслу Snort является переносимой системой и работает на многих операционных системах. В настоящее время существует Snort для архитектур x86 Linux, free BSD, NetBSD, Open BSD и Windows.

Система Snort может работать в трех основных режимах: анализатор пакетов, регистратор пакетов и сетевая система обнаружения вторжений. Кратко рассмотрим основные компоненты Snort.

Сначала входные данные (сетевые пакеты) обрабатываются декодером пакетов. Если Snort используется только как анализатор пакетов, то декодированные данные форматируются и выводятся на консоль. Если Snort используется как регистратор пакетов, то декодированные данные, в зависимости от того что задано в командной строке, или сохраняются в двоичном формате, или преобразуются в формат ASCII и записываются в файл на диске. Если Snort используется как сетевая СОВ, то обработка продолжается. Декодированные данные передаются препроцессорам, которые указаны в файле конфигурации *snort.conf.* Затем данные передаются процессору обнаружения, который осуществляет их сравнение с параметрами наборов правил, которые заданы в файле конфигурации. В случае совпадения данных происходит пересылка компонентам регистрации и сигнализации (в зависимости от настроек происходит регистрация или генерация сигналов тревоги).

Декодер пакетов

Декодер пакетов определяет протокол, содержащийся в рассматриваемом пакете, и проверяет соответствие данных этому протоколу. Декодер может генерировать свои собственные сигналы в случае неправильно сформированных заголовков пакетов, слишком больших пакетов, наличия необычных или неправильных опций TCP, установленных в заголовке, и в других аналогичных случаях. Посредством изменения файла конфигурации можно включать или отключать различные наборы сигнатур для этих полей пакетов.

Препроцессоры

Препроцессорами являются подключаемые модули Snort, которые позволяют различными способами анализировать входные данные. Если в файле конфигурации не задан никакой препроцессор, то можно только просматривать каждый пакет в том виде, в каком он передается.

Snort содержит широкий выбор препроцессоров, которые можно использовать в различных режимах работы. Все препроцессоры по выполняемым функциям можно условно разбить на три группы:

- ✓ сборка (реассемблирование) пакетов (препроцессоры frag2, frag3, stream4);
- ✓ нормализация протоколов (препроцессоры http_nspect, rpc_decode, telnet_decode, ftp_telnet, ftp_telnet_protocol, smtp, arpspoof);
- ✓ обнаружение известных аномалий в трафике (препроцессоры sfPortscan, bo, performance).

Рассмотрим представителей каждой из этих групп.

Препроцессоры сборки пакетов

Препроцессор frag3.

Новый препроцессор дефрагментации frag3 позволяет осуществлять обработку, ориентированную на целевую систему. Он предназначен для замены препроцессора frag2 и имеет дополнительные свойства: более быстрое выполнение и обработка, ориентированная на целевую систему для защиты от методов обхода и обмана СОВ. Идея учета целевой системы состоит в учете особенностей построения сетевых стеков реальных хостов защищаемой сети и снабжение СОВ топологической информацией. Поэтому данный препроцессор содержит два препроцессора: препроцессор глобальной конфигурации frag3_global и препроцессор обработки frag3_engine.

Препроцессор глобальной конфигурации использует следующие параметры:

- ✓ max_fragment – максимальное число одновременно отслеживаемых фрагментов (по умолчанию – 8192);
- ✓ memcap – объем памяти, зарезервированный для работы препроцессора (по умолчанию – 4Мб);
- ✓ prealloc_memcap – альтернативный режим управления памятью для повышения скорости обработки (использует заранее определенные узлы фрагментов в памяти);
- ✓ prealloc_frags – альтернативный режим управления памятью для повышения скорости обработки (использует заранее определенные узлы фрагментов на основе статистики).

Препроцессор обработки использует следующие параметры:

- ✓ timeout – допустимое время нахождения фрагмента в памяти препроцессора (по умолчанию – 60 с);
- ✓ ttl_limit – устанавливает границу значения ttl, превышение которойбудет не будет вызывать генерацию тревоги;
- ✓ min_ttl – устанавливает минимально допустимое значение TTL (время жизни пакета);
- ✓ detect_anomalies – обнаружение аномалий фрагментов (в текущей версии определяются восемь типов аномалий);
- ✓ bind_to – список IP-адресов, которые будут обрабатываться этим препроцессором (по умолчанию значение «all»);
- ✓ policy – выбор режима учета целевых систем. Допустимыми типами являются first, last, bsd, bsd-right, linux, windows и Solaris (по умолчанию – bsd).

Пример:

```
preprocessor frag3_global: max_frags 65536
prealloc_frags 262144 preprocessor frag3_engine: policy linux
bind_to { 10.1.1.11/32, 10.1.1.13/32 }
detect_anomalies
```

Препроцессор stream4.

Этот препроцессор предназначен для формирования состояний и полного реассемблирования TCP-потоков, что позволяет обнаруживать различные типы

сканирования портов, определение «отпечатков» ОС и т.д. Препроцессор stream4 использует следующие опции:

- ✓ detect_scans – обнаружение скрытого сканирования портов без обычного TCP-квитирования;
- ✓ detect_state_problems – обнаружение проблем с сохранением состояний TCP соединений (возможно большое число ложных срабатываний, что вызывается различными подходами в реализации стека TCP/IP);
- ✓ disable__evasion_alerts – запрещение выдачи сигналов тревоги, когда атакующий пытается обмануть систему сборки пакета;
- ✓ min_ttl [number] – установление минимального значения ttl, которое допускается snort при реассемблировании потока;
- ✓ ttl_limit [number] – устанавливает максимально допустимую разницу между пакетами одного и того же сеанса (начальным ttl сеанса и текущим);
- ✓ keepstats [machine|binary] – сохранение статистики сеанса (в текстовом или двоичном формате);
- ✓ noinspect – выключение инспекции состояний для всех портов, исключая те, для которых осуществляется сборка пакетов;
- ✓ timeout [number] – установка счетчика времени для сеанса (по умолчанию – 30 с);
- ✓ max_sessions [number] – ограничение числа сеансов, которые будет отслеживать данный препроцессор;
- ✓ memcap [number] – установка объема памяти, которую будет использовать препроцессор (по умолчанию – 8 388 608 байт);
- ✓ log_flushed_streams – указывает препроцессору, что каждый раз, когда создаваемый для потока над-пакет генерирует сигнал тревоги, его необходимо сохранять на диске (эта опция работает только в режиме pcap);
- ✓ server_inspect_limit [bytes] – установка количества байтов, которые будут контролироваться на стороне сервера.

Пример:

```
preprocessor stream4:   disable_evasion_alerts
```

Препроцессоры нормализации протоколов

Препроцессор http_inspect.

Этот препроцессор просматривает буфер данных в поисках заголовков HTTP и нормализует его поля. Препроцессор http-inspect распознает трафик сервера (ответы) и трафик клиента (запросы). Существующая версия препроцессора не осуществляет контроль состояний, то есть анализируются поля пакетов HTTP в каждом пакете. Поэтому, если пакеты не реассемблированы, то препроцессор может быть обманут. В будущем планируется ввести контроль состояний и для протокола HTTP. Данный препроцессор также содержит в себе два препроцессора: глобальной конфигурации и конфигурации сервера.

Препроцессор глобальной конфигурации позволяет задавать различные таблицы кодирования (параметр iis_unicode_map), контролировать наличие HTTP-трафика на портах, не сконфигурированных для HTTP (параметр detect_anomalous_servers), контролирует использование пользователями прокси-сервера и попытки использовать другие неуполномоченные прокси-сервера (параметр proxy_alert).

Препроцессор конфигурации сервера может работать с двумя типами серверов: конфигурация по умолчанию и конфигурация для списка IP-адресов. Конфигурация по умолчанию применяется к каждому серверу, который индивидуально не сконфигурирован. В случае задания адреса (адресов) указываются опции конфигурации для каждого сервера, содержащегося в списке. В число опций конфигурирования сервера входит profile [all/apache/iis] – профиль конфигурирования, включающий в себя следующие параметры: ports, iis_unicode_map, allow__proxy_use,flow_depth, no_alerts, inspect_uri_only, oversize_dir_length.
Пример:

```
preprocessor http_inspect_server: server 1.1.1.1 profile all ports {80 3128}
```

В текущей версии реализовано три вида профилей:

1. all – нормализация URI для предупреждения известных методов атак.
2. apache – для серверов Apache. Отличается от профиля iis только использованием UTF-8 стандарта Unicode, не допуская использования обратных слэшей.
3. iis – этот профиль предназначен для серверов IIS. В этом случае используется таблица кодирования IIS Unicode и допускаются кодирование, двойное декодирование, обратные слэши и т.п. для защиты от известных атак на IIS.

Тогда следующий URI

```
/scripts/.. %c0%af../winnt/system32/cmd.exe?/c+ver
```

при применении профиля all будет представлен (нормализован) следующим образом:

```
/winnt/system32/cmd.exe?/c+ver
```

Примеры:

- ✓ глобальное конфигурирование:

```
preprocessor http_inspect: global iis_unicode_map Unicode.map 1252
```

- ✓ конфигурирование сервера:

```
preprocessor http_inspect_server: server default profile all ports { 80 8080 } oversize_dir_length 500
```

- ✓ конфигурирование уникального сервера:

```
preprocessor http_inspect_server: server 1.1.1.1
ports { 80 3128 8080 } flow_depth 0
ascii no double_decode yes
non_rfc_char { 0x00 } chunk_length 500000
non_strict oversize_dir_length 300 no_alerts
```

Препроцессоры обнаружения аномалий

Препроцессор sfPortscan.

Этот препроцессор предназначен для обнаружения различных методов сканирования портов и использует следующие опции:

- ✓ proto {tcp udp icmp ip all} – указывает типы протоколов сканирования;
- ✓ scanjype {portscan portsweep decoy_portscan distributed_portscan all} – указывает тип сканирования, который будет обнаруживаться препроцессором;
- ✓ Sense_level {low|medium|high} – задает уровень чувствительности обнаружения. Низкий уровень позволяет обнаруживать общие методы сканирования (наблюдением за ответами ошибок, такими как TCP RST или ICMP Unreachable). Средний уровень обнаруживает сканирование портов, включая фильтруемое сканирование (сканирование без получения ответов). Этот уровень используется при использовании функции NAT или кэшировании DNS-серверов. Высокий уровень имеет наименьшие

пороги для обнаружения сканирования портов и значительно большее временное окно наблюдения;

- ✓ memcap {positive integer} – задает максимальный размер памяти в байтах, предназначенной для обнаружения сканирования портов;
- ✓ logfile {filename} – эта опция задает имя файла, в который будут записываться сигналы тревоги и отброшенные пакеты.

Пример:

```
preprocessor sfportscan: proto {all} memcap {1000000} sense_level {low}
```

Препроцессор bo.

Препроцессор предназначен для обнаружения трафика средства Back Orifice, которое было разработано хакерской группой «Cult of the Dead Cow» в 1998 г. Одной из особенностей этого средства удаленного управления является использование шифрования. Атакующий выбирает пароль, который преобразуется в шестнадцатиразрядный хэш-код. Трафик обмена шифруется путем выполнения операции XOR с полученным хэш-кодом. Исследователями обнаружено, что все запросы атакующего начинаются с «магической» строки *!*QWTY?, что позволяет определить наличие обмена и найти используемую для шифрования гамму.

Препроцессор обнаруживает Back Orifice, проверяя каждый UDP-пакет, размер которого не менее 18 байт, в ходе сравнения первых 8 символов данных с предварительно подготовленной таблицей зашифрованных вариантов магической строки (на самом деле проверяются только первые и последние два символа этой строки). Такая таблица формируется при запуске Snort на этапе инициализации препроцессора.

Препроцессор bo не требует аргументов и может сигнализировать о наличии следующих случаев: обнаружение трафика Back Orifice, обнаружение трафика клиента, обнаружение трафика сервера и обнаружение атаки буфера Snort.

Пример:

```
preprocessor bo: noalert {general server} drop {snort_attack}
```

Процессор обнаружения

Основой работы механизма обнаружения является сравнение с имеющимися правилами. Этот компонент Snort принимает данные от декодера пакетов и препроцессоров (если они активированы) и сравнивает значения полей этих данных с правилами, заданными в файле конфигурации. Процессор обнаружения сначала пытается определить, какой набор правил следует использовать для конкретного фрагмента данных. В первую очередь это определяется по соответствующему протоколу (TCP, UDP, ICMP или IP), а затем идентифицируются характеристики в рамках соответствующего протокола. Для TCP и UDP – это номера портов источника и назначения, для ICMP – это тип сообщения.

При сравнении с правилами обычно используется вариант, когда Snort работает по принципу первого совпадения – срабатывает первое правило, условия которого удовлетворены. В текущую версию Snort включена возможность выполнять несколько сравнений для одного события и генерировать несколько сигналов тревоги для одного пакета. Кроме того, существует возможность выбора последовательности применения правил, связанная с принципиальной возможностью атак обмана СОВ.

Для реагирования на множественные сигналы тревог предусмотрена возможность сигнализации об определенном количестве появления некоторого набора данных в пределах

заданного интервала времени. Эта возможность называется пороговой классификацией. При этом можно выбрать следующие варианты: выдавать сигнал тревоги после получения первых n-сигналов о данном событии, или на каждые n-экземпляров данного события.

Модули вывода

Модули вывода позволяют администратору формировать и представлять выходные данные Snort. Модули вывода включаются в работу всякий раз, когда вызываются подсистемы регистрации и сигнализации, после завершения работы декодера и препроцессоров. Модули вывода можно рассматривать как модули расширения системы, так как они могут быть написаны пользователями системы и включены в Snort в процессе компиляции.

В каждом из модулей вывода можно выделить семь ключевых аспектов: информация о заголовках и копирайте, include-файлы, зависимости (dependencies) и глобальные переменные, регистрация ключевых слов, анализ аргументов и включение в список функций, форматирование, обработка и хранение данных, препроцессорная обработка, очистка областей памяти и завершение работы приложения.

Snort предлагает пользователю несколько способов регистрации как генерируемых сигналов тревоги, так и связанных с ними данных пакетов. Эти данные могут регистрироваться в коде ASCII или в двоичном формате и записываться с помощью различных модулей вывода. Записи в двоичном формате могут быть восстановлены с помощью любого анализатора пакетов, например Ethereal, TCPdump или IRIS. Для текущей версии Snort разработаны модули, позволяющие посылать сигналы тревоги в виде запросов SNMP (Simple Network Management Protocol) удаленному серверу SNMP, а также модуль вывода SMB Alerting, посылающий сигналы тревоги удаленным Windows-системам в реальном масштабе времени.

Реализованы модуль регистрации в формате PCAP (Packet Capture Library), модуль регистрации согласно стандарту XML IDMEF. Кроме того, Snort имеет возможность записывать сигналы тревоги и пакеты в различные базы данных, включая MySQL, PostgreSQL, SQL Server и Oracle.

Правила Snort

Одной из самых сильных сторон Snort является возможность для пользователя разрабатывать и использовать свои собственные правила. Для построения правил используется простой язык описаний, с помощью которого можно написать собственные правила, исходя из специфики применения Snort. Большинство правил записывается в одной строке. В текущей версии допускается написание правил, занимающих несколько строк, при этом в конце каждой строки пишется символ обратного слэша (\).

Правило Snort состоит из двух основных частей: заголовка правила и опций правила. Заголовок правила содержит действие правила, протокол, адреса и маски IP-адресов источника и назначения, номера портов источника и назначения. Опции правила содержат сообщения, выдаваемые при генерации тревоги, и информацию о том, какую часть пакета необходимо просматривать для сравнения с правилом. Опции правила заключаются в скобки. Пример простого правила (для наглядности правило записано двумя строками с использованием знака \):

```
alert top any any -> 192.168.1.0/24 111 \
(content: "|00 01 86 a5|"; msg: "mountd access";)
```

В этом случае текст до открывающей скобки представляет собой заголовок правила. Часть правила, заключенная в круглые скобки, содержит опции правила. Имя опции (ключевое слово) заканчивается двоеточием, после которого следует содержимое опции, которое, в свою очередь, заканчивается точкой с запятой.

Заголовок правила.

Первым элементом заголовка является действие, которое указывает Snort, что делать при совпадении данных пакета с опциями правила и перечень действий, которые можно указывать в заголовке правила:

- ✓ alert – генерировать сигнал тревоги, используя заданный метод, и записать пакет в журнал;
- ✓ log – записать пакет в журнал;
- ✓ pass – игнорировать пакет;
- ✓ activate – генерировать сигнал тревоги и активировать динамическое правило;
- ✓ dynamic – действие данного правила игнорируется до активации правилом активации (опцией activate), после чего записывается в журнал;
- ✓ drop – посредством iptables удалить пакет и записать его в журнал;
- ✓ reject – посредством iptables удалить пакет, записать его в журнал и послать пакет TCP RST (для протокола TCP) или ICMP port unreachable (для протокола ICMP);
- ✓ sdrop – позволить iptables удалить пакет, но не записывать его в журнал.

Следующим полем заголовка является протокол. В текущей версии Snort для анализа подозрительного поведения используются четыре протокола: TCP, UDP, ICMP и IP. Планируется введение следующих протоколов: ARP, IGRP, GRE, OSPF, RIP, IPX и др.

Очередными полями заголовка являются адрес и порт. Может использоваться ключевое слово *any* для определения любого адреса. Адрес может сопровождаться блоком (CIDR, Classless Inter-Domain Routing) , обозначающим маску. Блок маски /24 показывает на сеть класса C, блок /16 на сеть класса B, а блок /32 на определенный адрес (например, комбинация 192.168.1.1/24 указывает на блок адресов от 192.168.1.1 до 192.168.1.255).

Предусмотрен оператор отрицания — восклицательный знак. Этот оператор указывает Snort рассматривать все адреса, за исключением отмеченных оператором отрицания. В заголовке правила можно указать диапазон адресов, который заключается в квадратные скобки:

```
alert top ! [192.168.1.0/24, 10.1.1.1/24] any -> \
[192.168.1.0/24, 10.1.1.1/24] 111 \
(content:   "| 00  01   86 a5|"; msg:   "mountd access";)
```

Очередным полем заголовка является номер порта. Порт может быть задан множеством способов: ключевое слово *any* указывает на любой номер порта, просто номер порта (в примере – 111), диапазоном номеров (например, 8000:8080). При задании номера порта можно также использовать оператор отрицания (за исключением применения оператора отрицания к ключевому слову *any).*

Далее в заголовке Snort используется оператор направления –>, показывающий направление трафика, к которому необходимо применить правило. Если необходимо

применять правило к двунаправленному трафику, используется символ **<>.** Символ направления <– не используется, так как оно заменяется символом –> с соответствующей заменой адресных частей заголовка. После оператора направления указываются адрес, маска и порт назначения,

Опции правил.

Опции правил являются основой механизма обнаружения, предоставляя возможности для обнаружения. Существует четыре категории опций:

- ✓ meta-data – предоставление информации о правиле, которая не используется в процессе обнаружения;
- ✓ payload – обеспечивает просмотр данных внутри пакета;
- ✓ non-payload – просмотр данных, не содержащихся в нагрузке пакета;
- ✓ post-definition – указывает на специфические переходы, выполняемые при срабатывании правила.

Далее кратко рассмотрим опции правил по категориям.

Опции meta-data. К опциям этой категории относятся опции: msg, reference, sid, rev, classtype и priority:

- ✓ msg – указывает механизмам регистрации и генерации тревоги содержимое соответствующего сообщения, которое задается в виде текстовой строки, заключенной в кавычки, например, (msg: «IMAP buffer overflow»);
- ✓ reference – позволяет сделать ссылку на внешние системы идентификации атаки. Поддерживаются следующие ссылки:
  - bugtraq (http://www.securityfocus.com/bid/);
  - cve (http://cve.mitre.org/cgi-bin/cvename.cgi?name=);
  - nessus (http://cgi.nessus.org/plugins/dump.php3?id=);
  - arachnids (http://www.whitehats.com/info/IDS);
  - mcafee (http://vil.nai.com/vil/dispVirus.asp?virus_k=);
  - url (http://).
- ✓ sid – используется для уникальной идентификации правил Snort. В настоящее время все sid разделены на группы: до 100 – зарезервированы, от 100 до 1000000 – включаются в дистрибутив Snort, более 1000000 – для локального применения;
- ✓ rev – позволяет перезаписывать сигнатуры и их описания с новой информацией (используется совместно с sid);
- ✓ classtype – характеризует категорию тревоги (указывает класс атаки). Пользователь может определить приоритет для каждого типа правил. Существующие категории и их приоритеты находятся в файле classification.config;
- ✓ priority – приоритет устанавливает уровень важности правила, например, в теле правила можно указать: (content: «/cgi-bin/phf»; priority: 10;).

Опции обнаружения содержимого (payload detection). Эти опции являются наиболее значимыми и интересными. К числу этих опций относятся: content, uricontent, iisdataat, pcre и др.:

- ✓ content – позволяет использовать правило, которое ищет определенные данные в содержимом пакета. Данные могут представлять собой текстовую строку или двоичные данные. Например, content:!"GET" (представляет текстовую строку), content: "|5c 00|P|00|1|00|P|00|00 5c|"; (представляет смесь текстовых и двоичных данных). Для ключевого слова content, в свою очередь, можно указать модифици-

рующие ключевые слова: depth, offset, distance, within, nocase и rawbytes, которые предоставляют различные варианты возможностей поиска заданных строк:

- depth – определяет количество байтов, которые данное правило должно анализировать при поиске заданного содержимого;
- offset – указывает препроцессору, что поиск заданной строки надо начинать с байта offset;
- distance – указывает количество байтов, которые должны игнорироваться до начала сравнения;
- within – позволяет удостовериться, что заданное опцией количество байтов находится между указанными образцами, например, (content:«ABC»; content:«EFG»; within: 10;);
- nocase – указание на игнорирование регистра текста в содержании правила, например, (content:«USER root»; nocase;);
- rawbytes – это ключевое слово позволяет Snort рассматривать содержимое пакета в шестнадцатеричном представлении, не учитывая тип кодирования, например, (content: «|FF Fl|»; rawbytes;);
- uricontent – эта опция позволяет анализировать трафик запрашивающей системы только в URI-разделе запроса, нормализуя найденные строки, например URI:

  /scripts/..%c0%af../winnt/system32/cmd.exe?/c+ver и

  /cgi-bin/aaaaaaaaaaaaaaaaaaaaaaaaa/.. %252fp%68f?

  будут нормализованы в

  /winnt/system32/cmd.exe?/c+ver и /cgi-bin/phf?

- ✓ iisdataat – определяет, что содержимое находится в определенном месте, соответствуя концу данных, полученных в предыдущем пакете;
- ✓ pcre – позволяет записать правило, используя язык Perl.

Учитывая значительное число атак, использующих специфику реализации стека TCP\IP, Snort содержит большое число опций, которые не связаны с содержимым пакета. Рассмотрим некоторые из этих опций, задаваемых следующими ключевыми словами:

- ✓ fragoffset – позволяет сравнивать значение поля offset фрагмента с заданным значением, например для просмотра первых фрагментов (fragbits: M; fragoffset: 0;);
- ✓ ttl – позволяет проверять время жизни пакета, например (ttl: <3;);
- ✓ tos – проверяет поля TOS на соответствие заданному значению, например (tos:!4;);
- ✓ id – позволяет сравнить значение поля ID IP-пакета с заданным значением, например (id:31337;);
- ✓ ipopts – используется для проверки наличия определенных опций IP. Контролю могут подвергаться следующие опции: rr – Record route, eol – End of list, nop – No operation, ts – Time Stamp, sec – IP security option, lsrr – Loose source routing, ssrr – Strict source routing, satid – Stream identifier, any – установка любой опции. В правиле может содержаться только одно ключевое слово ipopts, например (ipopts:lsrr;);
- ✓ fragbits – позволяет контролировать биты фрагментации и зарезервированные биты, для чего можно использовать обозначения битов (M – есть еще фрагменты, D – запрет фрагментирования, R – зарезервированные биты) и простые операции сравнения (знак «+» означает соответствие заданного бита и любые значения других битов, знак «*» означает срабатывание, если любой из заданных битов установлен, знак «!» вызовет срабатывание, если ни один из указанных битов не установлен). Например, опция (fragbits:MD+;)вызовет срабатывание, если будут установлены биты More fragments и Do not Fragment;

- ✓ dsize – позволяет контролировать длину пакета, например (dsize:300<>400;);
- ✓ flags – позволяет контролировать установку флагов TCP (F – FIN, S – SYN, R – RST, P – PSH, A – ACK, U – URG, 1 – зарезервированный бит 1, 2 – зарезервированный бит 2, 0 – отсутствие флагов TCP) и простые операции сравнения (знак «+» означает соответствие заданного бита и любые значения других битов, знак «*» означает срабатывание, если любой из заданных битов установлен, знак «!» вызывает срабатывание, если ни один из указанных битов не установлен). Например, для контроля наличия флагов SYN и FIN и игнорирования зарезервированных битов можно задать опцию в виде (flags:SF,12;).

Кроме того, возможны проверки значений и других параметров пакетов, для чего служат следующие опции: seq (последовательный номер TCP-пакета, ack (значение подтверждения), window (значение окна передачи TCP), itype (тип сообщения ICMP), icode (код сообщения ICMP), icmp_id (значение идентификатора ICMP), icmp_seq (последовательный номер ICMP-сообщения), ip_proto (имя протокола в заголовке IP), sameip (проверка равенства адреса источника и адреса назначения) и др. Рассмотренные опции позволяют реализовать достаточно сложные правила обнаружения различных атак и их вариантов.

Примеры правил

В качестве примера применения рассмотренных опций рассмотрим несколько реальных правил Snort. Правило для определения TCP-пакета, в котором установлен только флаг FIN. Такие пакеты используются атакующими для определения операционной системы целевого хоста, поскольку машины с ОС Windows отвечают на него пакетом с установленными флагами АСК и RST (независимо от того, открыт или закрыт соответствующий порт), а системы Unix отвечают пакетом с такими же флагами только в случае, когда порт закрыт. Соответствующее правило имеет следующий вид:

```
alert top $EXTERNAL_NET any -> $HOME_NET any \
(msg:»SCAN FIN»; flags: F; reference:arachnids,27; \
classtype:attempted-recon; sid:621; rev:1;)
```

Рассмотрим правило, определяющее поведение атакующего после получения доступа к Windows-серверу путем использования IIS-уязвимости, когда атакующий пытается использовать командный интерпретатор cmd.exe. Затем атакующий пытается просматривать директорию, используя команду dir. Строка «Volume Serial Number» обычно содержится в списке директорий для Windows NT\2000\XP:

```
alert top $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET \
any (msg:»ATTACK RESPONSES http dir listing»; \
content: «Volume Serial Number»; \
flow:from_server,established; classtype :bad-unknown ; \
sid:1292; rev:4;)
```

Следующее правило предназначено для обнаружения сетевого червя Slammer, пытающегося использовать уязвимость переполнения буфера в Resolution Service MS SQL Server 2000:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 1434 \
(msg: »MS-SQL Worm propagation attempt»; \
content:"| 04|»; depth:1; \
content:»| 81 F1 03 01 04 9B 81 F1 01|»; \
content:»sock»; content:»send»; \
reference:bugtraq,5310; class type:misc-attack; \
reference:bugtraqf5311; \
reference:url,vil.nai.com/vil/content/v_99992.htm; \
sid:2003; rev:2;)
```

29. Медиаплеер VLC

Приложение VideoLAN является транслятором медиапотока. Для его использования необходимо зарегистрироваться в системе, используя учётную запись «vlc» и пароль «vlc».

Настройка с использованием командного интерфейса

На сервере вещания выполните команду:

```
cvlc -vvv input_video.file --sout udp:239.0.0.1:1234 --ttl 12
```

где:

- cvlc – команда для запуска VideoLAN без графического интерфейса;
- -vvv – ключ запуска для включения вывода отладочной информации и прочих сообщений – наиболее подробный режим
- input_video.file – имя файла с входным медиапотоком;
- --sout – ключ запуска, определяющий тип выходного потока;
- udp:239.0.0.1:1234 – выходной поток будет передаваться по протоколу UDP на групповой адрес 239.0.0.1 на порту 1234;
- --ttl – ключ запуска, определяющий время жизни IP-пакета.

На потребителе потока выполните команду:

```
vlc -vvv udp://@239.0.0.1:1234
```

где:

- vlc – команда для запуска VideoLAN;
- -vvv – ключ запуска для включения вывода отладочной информации и прочих сообщений – наиболее подробный режим;
- udp://@239.0.0.1:1234 – получать поток для группы с адресом 239.0.0.1 по протоколу UDP на порту 1234.

Либо выполните следующую команду (для использования mplayer):

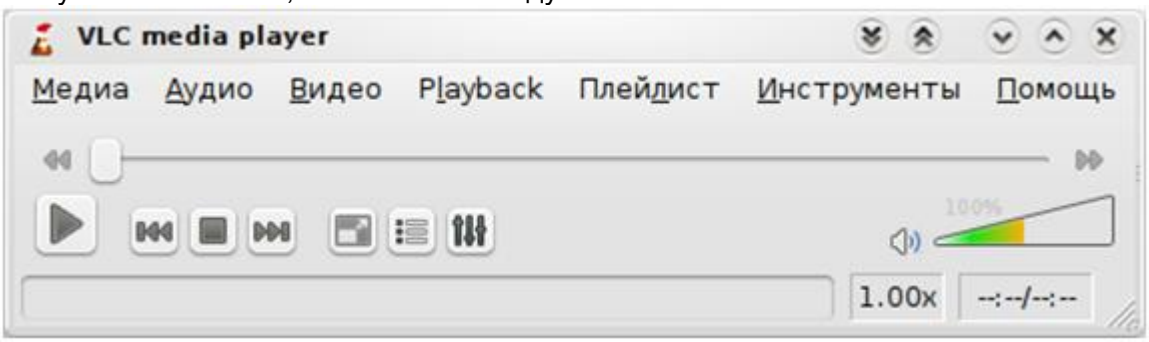
```
mplayer udp://239.0.0.1:1234
```

где:

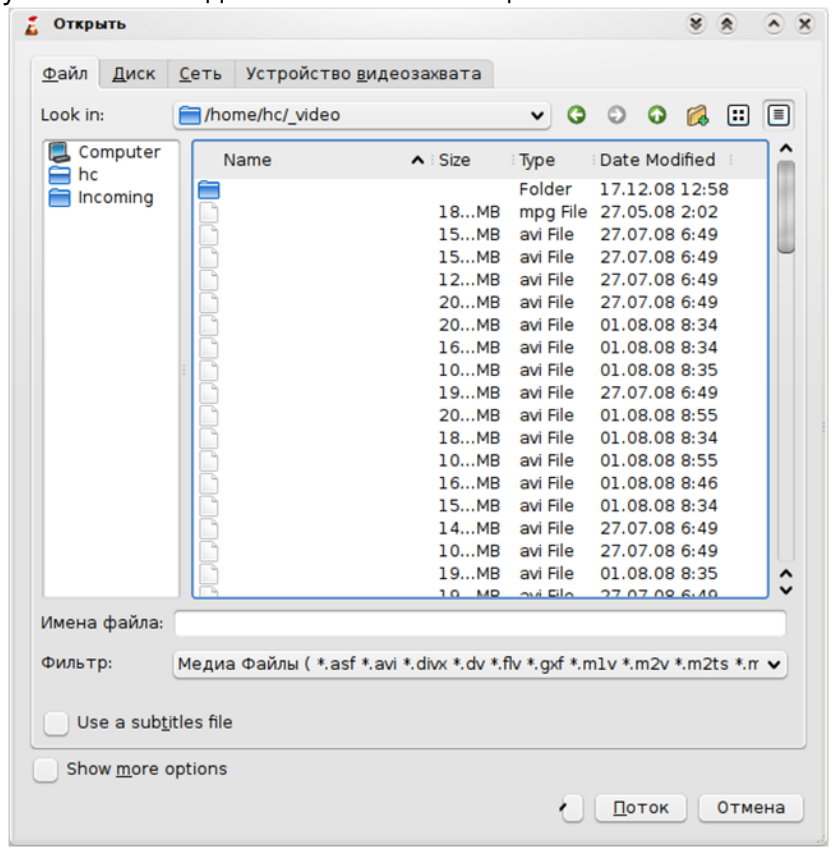
- mplayer – команда для запуска mplayer;
- udp://239.0.0.1:1234 – получать поток для группы с адресом 239.0.0.1 по протоколу UDP на порту 1234.

Работа с графическим интерфейсом

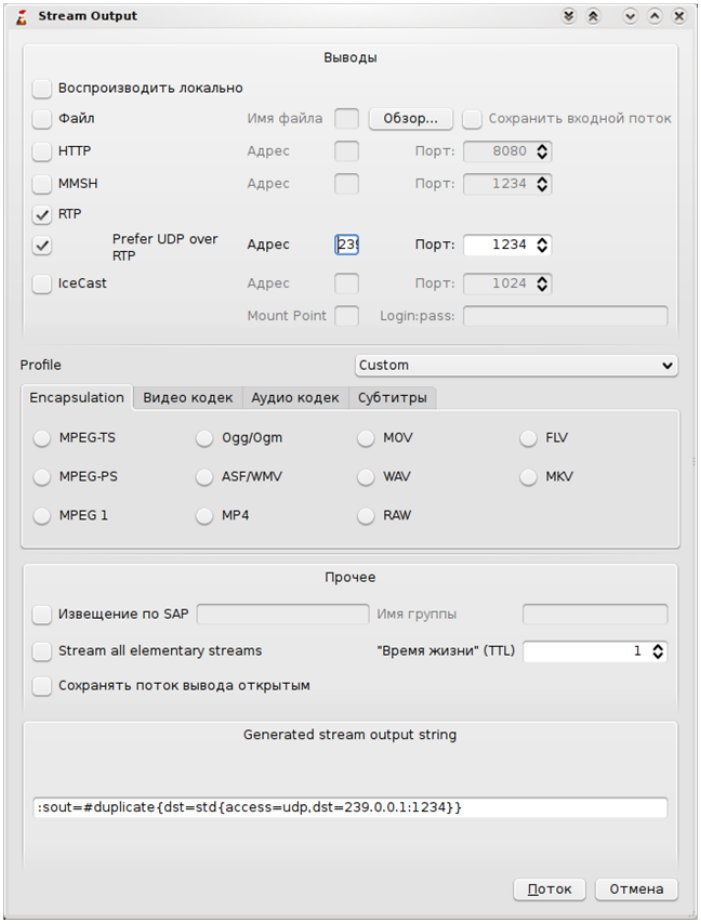
1. Запустите VideoLAN, выполнив команду vlc. Появится окно:



2. Откройте пункт меню «Медиа → Потоковое вещание». Появится окно:

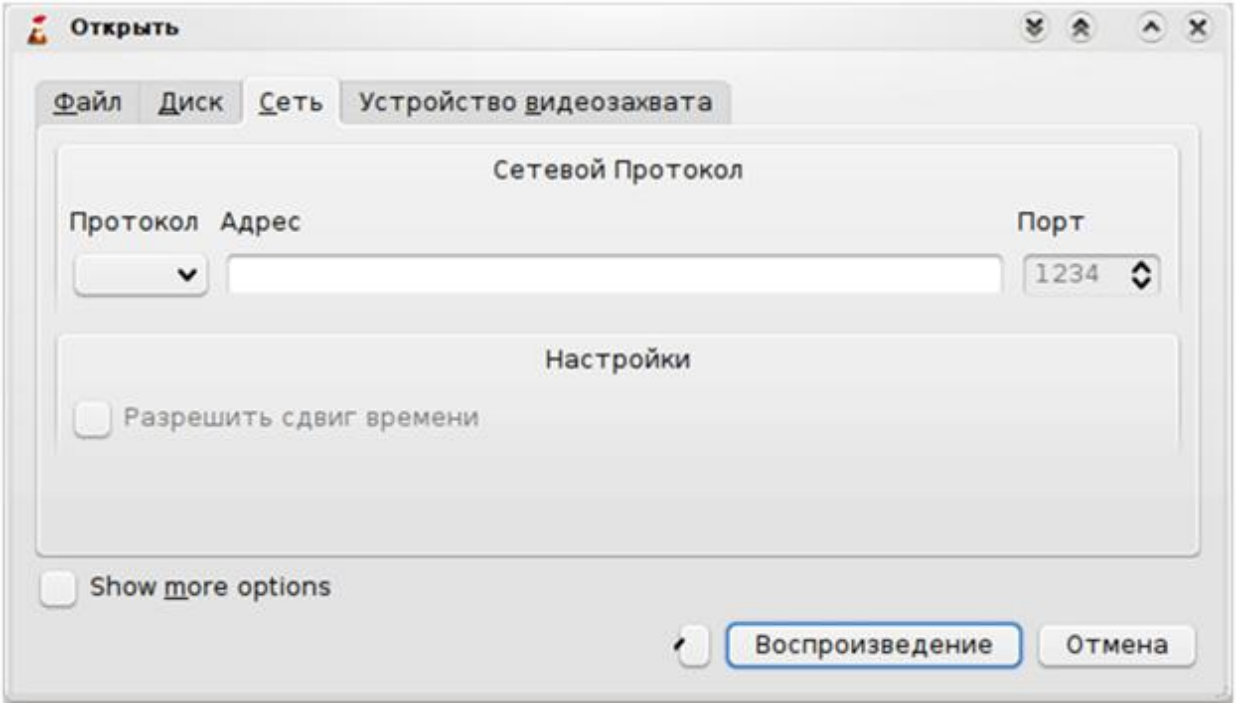


3. Выберите нужный файл и нажмите кнопку «Поток». Появится окно:



4. Отметьте пункты «RTP» и «Prefer UDP Over RTP». В поле «Адрес» напротив опции «Prefer UDP...» впишите требуемый групповой адрес, а в поле «Порт» укажите порт, на котором будет вестись трансляция. Нажмите кнопку «Поток». Вещание запущено.

5. На стороне потребителя запустите VideoLAN командой vlc. Выберите пункт меню «Медиа → Open Network». Откроется окно:



6. В списке протоколов выберите UDP, в поле «Адрес» введите требуемый групповой адрес, в поле «Порт» — порт, через который ведётся вещание. Нажмите кнопку «Воспроизведение». Либо воспользуйтесь командой запуска mplayer из консольного варианта

Анализ multicast-трафика с помощью утилиты tcpdump

Для перехвата пакетов IGMP выполните команду:

```
# tcpdump -i eth0 -n ip proto 2
```

где:

- -i eth0 – выбор интерфейса, с которого будут захватываться пакеты;
- -n – отключение функции преобразования IP-адресов в имена с помощью DNS;
- ip proto 2 – фильтр IP-пакетов с полем «Протокол», равным 2.

Дополнительно можно указать ключ -v для увеличения количества выводимой информации (чем больше букв v — тем подробнее) и ключ -X для вывода пакета в шестнадцатиричном виде (-XX для подробного режима).

Пример перехвата:

```
01:26:10.391015 IP 192.168.0.2 > 239.0.0.1: igmp v2 report 239.0.0.1
01:26:37.005141 IP 192.168.0.1 > 224.0.0.1: igmp query v2 [max resp time 226]
01:26:56.294347 IP 192.168.0.2 > 239.0.0.1: igmp v2 report 239.0.0.1
01:27:10.985580 IP 192.168.0.2 > 224.0.0.2: igmp leave 239.0.0.1
```

30. Утилита vconfig

Программа vconfig является программой настройки VLAN 802.1Q.

Синтаксис
vconfig [опции]

Опции
add [*interface-name*] [*vlan-id*]

Создаёт устройство *vlan-device* на интерфейсе *interface-name*. Созданное устройство будет названо в соответсвии с настройками именования.

rem [*vlan-device*]

Удаляет заданый *vlan-device*.

set_flag [*vlan-device*] 0 | 1

Если установить равным 1, выполняется reorder заголовка Ethernet. При выполнении прослушивания трафика (tcpdump) на устройстве, он будет выглядеть как обычное ethernet-устройство без VLAN'ов. Если задан 0 (по умолчанию), при просмотре трафика видны теги. Обычно никаких проблем не возникает и при настройках по умолчанию, но иногда бывает -- с программами фильтрации трафика.

set_egress_map [*vlan-device*] [*skb-priority*] [*vlan-qos*]

Говорит, что исходящие пакеты с заданным *skb-priority* должны маркироваться тегом с заданным приоритетом *vlan-qos*. По умолчанию устанавливается приоритет 0.

set_ingress_map [*vlan-device*] [*skb-priority*] [*vlan-qos*]

Говорит, что входящие пакеты маркированные тегом с приоритетом *vlan-qos' должны ставиться в очередь с приоритетом* skb-priority. *По умолчанию приоритет равен 0.*

set_name_type VLAN_PLUS_VID | VLAN_PLUS_VID_NO_PAD | DEV_PLUS_VID | DEV_PLUS_VID_NO_PAD

Задаёт режим в котором будет создаваться *vlan-device*. Для того чтобы посмотреть какие режимы поддерживаются, нужно вызвать **vconfig** без параметров.

Если сетевое устройство поддерживает интерфейс Broadcom NICE, он используется. Это важно, поскольку такие устройства сами удаляют тег VLAN на Ethernet-пакетах. В этом случае опция set_flag игнорируется. При просмотре дампа трафика, увидеть теги уже не получится.

Файлы

/proc/net/vlan/config

/proc/net/vlan/[vlan-device]