Расширенное администрирование Linux.
Блок 6.

v 1.03

Оглавление

Фильтрация пакетов (firewall1) в Linux

В ядро Linux встроена возможность фильтрации сетевых пакетов. В ядрах версии 2.4.x и 2.6.x это программное обеспечение называется netfilter.

Для управления фильтрацией пакетов необходимо использовать специальное программное обеспечение, позволяющее администратору настраивать параметры фильтрации. В зависимости от версии ядра, для управления используются различные программы:

2.0.x — ipfwadm
2.2.x — ipchains
2.4.x и 2.6.x — iptables

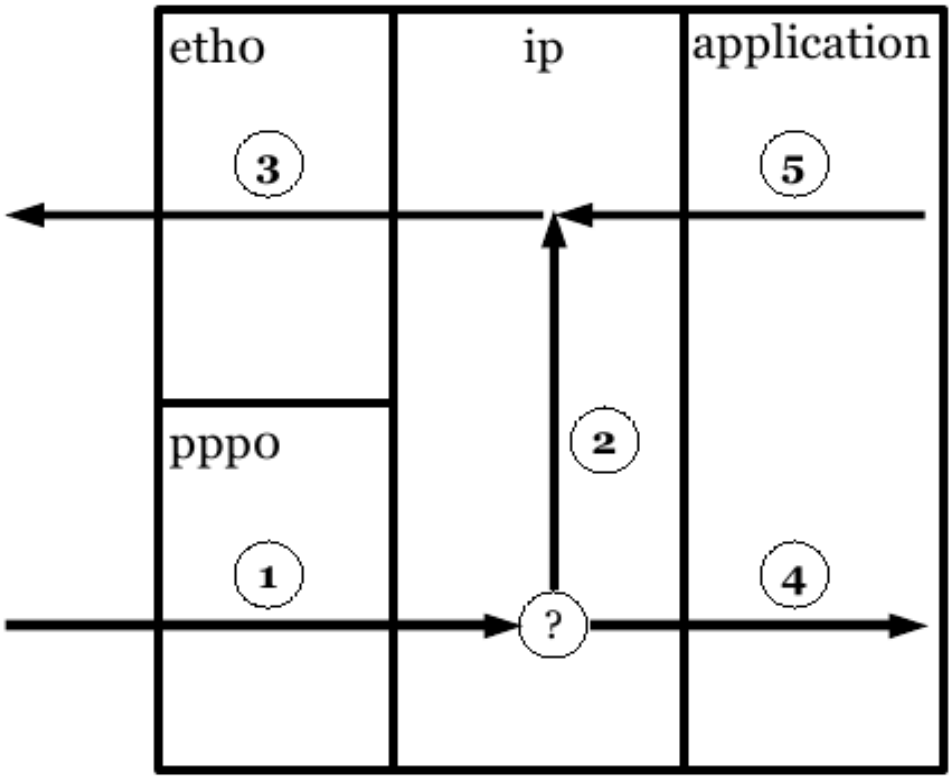
Несмотря на наличие трех разных программ управления фильтрацией пакетов, в современных версиях ядра Вы можете использовать любую из них, так как в Linux все они до сих пор поддерживаются. Но настоятельно рекомендуется пользоваться программой iptables, позволяющей использовать все возможности фильтрации пакетов современных ядер Linux.

В этом блоке будет рассмотрена организации фильтрации пакетов и создание NAT преобразований при помощи программы iptables.

Цепочки

С появление программы ipchains было введено понятие цепочки. Цепочка — это некоторый логический участок пути прохождения пакета, на который можно подключать правила фильтрации.

Для облегчения понимания организации правил фильтра- ции предположим, что все пакеты поступают через интерфейс ppp0 (1), а уходят через интерфейс eth0 (3). После поступления пакета на уровне IP принимается решение о его маршрутизации. После решения о маршрутизации пакет передаётся либо приложениям, работающим на этом компьютере (4), либо уходит через другой сетевой интерфейс (2,3). Так же пакеты могут отправляться приложениями, работающими на компьютере (5,3).



Стрелки, изображённые на рисунке, являются цепочками, на которые можно подключать правила фильтрации.

Название	**Номер**
REROUTING	1
POSTROUTING	3
INPUT	4
FORWARD	2
OUTPUT	5

Имена цепочек зарезервированы и пишутся с большой буквы.

В ipchains существует проблема с именами цепочек и прохождения пакетов через них. Если предположить, что пакет проходит транзитом через нашу машину, он будет последовательно проходить через цепочки INPUT (1), FORWARD (2) и OUTPUT (3). При этом в каждой цепочке пакет будет проверяться правилами, находящимися в каждой цепочке, что замедляет скорость прохождения пакета.

Iptables избавлена от этого недостатка, так как транзитные пакеты проходят только через правила цепочки FORWARD (2). Пакеты, предназначенные для приложений на компьютере, проходят только через правила цепочки INPUT (4). Исходящие пакеты приложений компьютера обрабатываются правилами фильтрации цепочки OUTPUT (5).

В ipchains и iptables пользователи могут определять свои собственные цепочки.

Таблицы

В iptables кроме цепочек используются три таблицы:
— filter
— nat
— mangle
Каждая таблица имеет индивидуальный набор цепочек. Пример, рассмотренный в предыдущей главе, описывал цепочки таблицы filter.

Таблица filter

Таблица предназначена для определения правил фильтрации пакетов.

В таблице имеются три цепочки:

— INPUT (4)
— FORWARD (2)
— OUTPUT(5)

Таблица nat

Таблица предназначена для описания правил NAT (Network Address Translations).

В таблице имеются три цепочки:

— PREROUTING (1)

— POSTROUTING (3)
— OUTPUT (5)

Таблица mangle

Таблица предназначена для внесения изменений в заголовки пакетов.

В таблице имеется пять цепочек:

— PREROUTING (1)
— POSTROUTING (3)
— INPUT (4)
— FORWARD (2)
— OUTPUT (5)

Порядок прохождения таблиц и цепочек.

Несмотря на то, что в netfilter используется несколько та- блиц и цепочек, пакет не может параллельно проходить через все таблицы и цепочки. Он проходит через них последовательно.

Порядок прохождения транзитных пакетов

Таблица	Цепочка
mangle	PREROUTING
nat	PREROUTING
mangle	FORWARD
filter	FORWARD
mangle	POSTROUTING
nat	POSTROUTING

Порядок прохождения пакетов, предназначенных для приложений компьютера

Таблица	Цепочка
mangle	PREROUTING
nat	PREROUTING
mangle	INPUT
filter	INPUT

Порядок прохождения пакетов, отправленных приложениями компьютера.

Таблица	Цепочка
mangle	OUTPUT
nat	OUTPUT
filter	OUTPUT
mangle	POSTROUTING
nat	POSTROUTING

Вопросы

1. По каким основным параметрам можно осуществлять фильтрацию пакетов?

2. Какие таблицы существуют у netfilter?

3. Какие цепочки можно использовать в таблице mangle?

4. Какой порядок прохождения транзитных пакетов через таблицы и цепочки netfilter?

Программа iptables

iptables [опции] [-t таблица] [-Команда][цепочка][критерии отбора] [-j действие]

Программа iptables предназначена для управления правилами netfilter. Она позволяет:

- добавлять, заменять, удалять правила
- создавать, удалять цепочки пользователя
- просматривать, устанавливать и сбрасывать счётчики пакетов

Для определения, с какой таблицей будет работать программа, используется опция “-t”, после которой указывается соответствующая таблица: filter, nat, mangle. Если опция не указана, программа будет работать с таблицей filter.

Команды программы iptables

-A — добавить правило
-D — удалить правило
-R — заменить правило
-I — вставить правило
-L — показать список правил
-F — очистить цепочку или таблицу
-Z — обнулить счётчики
-N — создать цепочку пользователя
-X — удалить цепочку пользователя
-P — установить политику по умолчанию

Для определения действия, которое программа iptables должна выполнить, ей необходимо указать команду.

Команда	Описание
-A, --append	Добавляет правило в конец цепочки. При выполнении команды необходимо обязательно указать цепочку. Пример: `iptables -A INPUT -s 10.10.100.100 -j ACCEPT`
-D, --delete	Удаляет правило из цепочки. Существуют два способа удаления правила: по

Команда	Описание
	номеру правила или с указанием всех критериев отбора. Пример: **iptables -D INPUT 1 iptables -D INPUT --dport 80 -j DROP**
-R, --replace	Команда заменяет одно правило другим. Пример: **iptables -R INPUT 1 -s 192.168.0.1 -j DROP**
-I, --insert	Команда вставляет правило в указанное место в цепочке. Пример: **iptables -I INPUT 1 --dport 80 -j ACCEPT**
-L, --list	Если не указывать имя цепочки, команда показывает все правила текущей таблицы. Если указать имя цепочки — показывает все правила в указанной цепочке. Пример: **iptables -L INPUT**
-F, --flush	Если не указывать имя цепочки, команда удаляет все правила из текущей таблицы. Если указать имя цепочки — будут удалены все правила из указанной цепочки. Пример: **iptables -F INPUT**
-Z, --zero	Каждому правилу в таблице соответствуют два счётчика: количество пакетов и количество байт, сработавших на данном правиле. Команда сбрасывает счетчики в текущей таблице, или, если указано имя цепочки, в заданной цепочке. Пример: **iptables -Z INPUT**
-N, --new-chain	Команда создаёт цепочку с заданным именем в текущей таблице. Имя должно быть уникальным и не должно совпадать с зарезервированными именами цепочек и действий. Пример: **iptables -N allowed**
-X, --delete-chain	Команда удаляет цепочку пользователя из та- блицы. Удалять можно только цепочки, не содержащие правил. Так же на эти цепочки не должно быть ссылок из других цепочек. Пример: **iptables -X allowed**
-P, --policy	Команда задает политику по умолчанию для цепочки. Политика определяет, что следует сделать с пакетом, на котором не сработало ни одно правило цепочки. В качестве значения по- литики можно использовать значения ACCEPT и DROP. Пример: **iptables -P INPUT DROP**

Опции программы iptables

Программе iptables можно указывать дополнительные опции:

Опция	Описание
-v, --verbose	Опция заставляет программу выводить дополнительную информацию. Например, в сочетании с командой -L будут показаны не только правила, но и значения счётчиков.
-x, --exact	Опция заставляет программу выводить точные значения счётчиков.
-n, --numeric	Опция заставляет программу не делать обратного преобразования из IP адреса в имя машины, если в выводе программы встречаются IP адреса.
--line-numbers	Опция заставляет программу выводить номера правил в цепочке.
-c, --set- counters	Опция позволяет установить значения счётчиков при добавлении правила в цепочку.

Критерии отбора пакетов

При добавлении правила необходимо указывать критерии отбора пакетов, на которых это правило будет работать. Правила можно условно разделить на три группы:

- **Общие критерии** — не зависят от типа протокола и не требуют загрузки специальных модулей
- **Неявные критерии** — зависят от типа протокола и не требуют загрузки специальных модулей
- **Явные критерии** — перед использованием требуют явной загрузки специальных модулей

В одном правиле можно указывать сразу несколько критериев отбора пакетов. Если в правиле отбора пакета критерий не определён, значит возможно любое значение критерия.

Общие критерии

-p — определяет протокол
-s — определяет IP источника
-d — определяет IP назначения
-i — определяет входной интерфейс
-o — определяет выходной интерфейс
-f — определяет фрагменты фрагментированного пакета

Общие критерии не зависят от типа протокола и для их использования не требуется загрузка специальных модулей при помощи опции -m.

Критерий	Описание
-p	Критерий служит для определения типа протокола. В качестве

Критерий	Описание

	дополнительного парамет- ра необходимо указывать имя, номер протокола (см. файл /etc/protocols) или ключевое слово ALL. При указании протокола можно использовать символ “!”, который инвертирует значение критерия. Примеры:
	`-p tcp`
	`-p ! icmp`

-s	Критерий определяет IP адрес источника. В качестве дополнительного параметра можно указывать IP адрес машины или сети. В по- следнем случае необходимо указывать маску подсети. Символ “!” инвертирует значение параметра.
	Примеры:
	`-s 192.168.0.1`
	`-s 10.10.100.0/24`
	`-s ! 10.10.100.0/255.255.255.0`

-d	Критерий определяет IP адрес получателя. В качестве дополнительного параметра можно указывать IP адрес машины или сети. В по- следнем случае необходимо указывать маску подсети. Символ “!” инвертирует значение параметра.
	Примеры:
	`-d 192.168.0.1`
	`-d 10.10.100.0/24`
	`-d ! 10.10.100.0/255.255.255.0`

-i	Критерий определяет интерфейс, с которого был получен пакет. Использование этого критерия допускается только в цепочках INPUT, FORWARD и PREROUTING, в любых других случаях будет вызывать сообщение об ошибке. Если имя интерфейса завершается символом +, то критерий задает все интерфейсы, начинающиеся с заданной строки. Символ “!” инвертирует значение параметра.
	Примеры:
	`-i ! eth0`
	`-i ppp+`
	`-i ! eth1`

-o	Критерий определяет выходной интерфейс. Этот критерий допускается использовать только в цепочках OUTPUT, FORWARD и POSTROUTING, в противном случае будет генерироваться сообщение об ошибке. Если имя интерфейса завершается символом +, то критерий задает все интерфейсы, начинающиеся с заданной строки. Символ “!” инвертирует значение параметра.
	Примеры:
	`-o eth0`
	`-o ppp+`
	`-o ! eth1`

-f	Критерий определяет все фрагменты фрагментированного пакета, кроме первого. Символ “!” инвертирует значение параметра. Примеры: `-f !`

Критерий	Описание
	`-f`

Неявные критерии

TCP критерии:
--sport — порт источника
--dport — порт назначения
--tcp-flags — определение TCP флагов
--syn — запрос на соединение
UDP критерии:
--sport — порт источника
--dport — порт назначения
ICPM критерии:
--icmp-type — определяет тип ICMP-пакета

Для того, чтобы воспользоваться явными критериями, необходимо при помощи критерия “-p” указать протокол и только после этого определять явные критерии.

TCP критерии

TCP критерии можно использовать только после определения протокола TCP: -p tcp.

Критерий	Описание

--sport	Критерий определяет порт, с которого был отправлен пакет. В качестве дополнительного па- раметра можно указывать номер порта или имя службы (см. файл /etc/services). Так же можно указывать диапазон портов, разделенных символом “:”. Символ “!” инвертирует значение параметра. Примеры:
	`--sport 1024:65535`
	`--sport 80`
	`--sport ! www`

--dport	Критерий определяет порт назначения. В каче- стве дополнительного параметра можно указывать номер порта или имя службы (см. файл /etc/services). Так же можно указывать диапазон портов, разделенных символом “:”. Символ “!” инвертирует значение параметра.
	Примеры:
	`--dport 1024:65535`
	`--dport 80`
	`--dport ! www`

--tcp-flags	Критерий позволяет контролировать флаги TCP пакета. В качестве дополнительных пара- метров указывают список интересующих флагов и через пробел список флагов, значения которых должны быть равны 1. Так же можно использовать ключевые слова ALL — все, NONE — ни одного. Символ “!” инвертирует значение параметра. Пример: `--tcp-flags SYN,ACK,FIN SYN`

Критерий	Описание

--syn	Критерий определяет TCP пакет с установлен- ным флагом SYN и со сброшенными флагами ACK и FIN, что соответствует запросу на соединение.
	Примеры:
	`--syn`
	`! --syn`

UDP критерии

UDP критерии можно использовать только после определения протокола UDP: -p udp.

Критерий	Описание

--sport	Критерий определяет порт, с которого был отправлен пакет. В качестве дополнительного параметра можно указывать номер порта или имя службы (см. файл /etc/services). Так же можно указывать диапазон портов, разделен- ных символом “:”. Символ “!” инвертирует значение параметра.
	Примеры:
	`--sport 1024:65535`
	`--sport 21`

--dport	Критерий определяет порт назначения. В качестве дополнительного параметра можно указывать номер порта или имя службы (см. файл /etc/services). Так же можно указывать диапазон портов, разделенных символом “:”. Символ “!” инвертирует значение параметра.
	Примеры:
	`--dport 1024:65535`
	`--dport 21`

ICMP критерий

ICMP критерий можно использовать только после определения протокола ICMP: -p icmp.

Существует всего лишь один icmp критерий: --icmp-type, определяющий тип ICMP пакета. В качестве дополнительного параметра можно указывать имя или номер ICMP пакета.

Например:
```
--icmp-type echo-request
```

Чтобы посмотреть список всех типов ICMP пакетов, можно воспользоваться программой iptables:
```
iptables -p icmp --help
```

Явные критерии

Наиболее часто используемыми явными критериями являются следующие четыре:

- limit - ограничивает количество срабатываний правила
- mac – позволяет использовать MAC адреса в качестве критерия отбора пакетов
- multiport – позволяет использовать в качестве параметра несколько портов
- state – определяет состояния пакетов

Для того, чтобы воспользоваться любым явным критерием, необходимо при помощи параметра “-m” загрузить необходимый модуль и только затем можно указывать явные критерии.

Критерий limit

Критерий limit ограничивает количество срабатываний правила.

--limit-burst — определяет количество пакетов

--limit — задаёт единицу времени

Перед использованием критерия необходимо воспользоваться параметром “-m limit”.

Критерий	Описание
--limit-burst	Критерий определяет количество пакетов, на которых правило будет работать. Если это параметр не указан явно, его значение принимается равным 5. Пример: `--limit-burst 1`
--limit	Критерий определяет сколько раз в единицу времени будет срабатывать правило. В качестве дополнительного параметра, указывающего единицу времени, можно использовать: N/s (секунда), N/m (минута), N/h (час), N/d (день). Вместо буквы N необходимо подставить число. Пример: `--limit 1/s`

Например, установлены следующие критерии отбора пакетов:

```
-p tcp --dport 80 -m limit --limit 1/s --limit-burst 4 -j ACCEPT
```

Это значит, что четыре пакета (--limit-burst 4) протокола TCP (-p tcp), направленные на порт 80 (--dport 80), будут приниматься (-j ACCEPT) один раз в секунду (--limit 1/s).

Критерий limit применяют при защите от syn flood атак или подобных им. А так же его рекомендуют применять при отсылке пакетов в систему Syslog.

Лабораторная работа А.

.

Цель работы.

Научиться использовать критерий limit.

Задача.

Необходимо сделать так, чтобы ICMP пакеты типа echo- request принимались один раз в

секунду.

Задачи	Описание
1. Подготовка к выполнению работы.	1. Проверьте, какие правила фильтрации пакетов установлены на Вашей машине: **iptables -L** 2. Если какие-либо правила фильтрации были установлены, необходимо их удалить: **iptables -F** 3. Если политика по умолчанию на цепочках INPUT и OUTPUT не установлена в значение ACCEPT, политику по умолчанию необходимо установить в значение ACCEPT: **iptables -P INPUT ACCEPT iptables -P OUTPUT ACCEPT**
2. Добавление правила в цепочку INPUT.	1. Добавьте одной строкой следующее правило: **iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s --limit-burst 1 -j ACCEPT** 2. Посмотрите, как выглядит это правило в списке правил: **iptables -L 3.** 3. Проверьте, как работает это правило. Попросите соседа выполнить следующую команду: **ping -f Ваш_IP -c 10000** 4. Обратите внимание на процент потерянных пакетов. Он равен 0 или близкому к нулю значению, но это не значит, что правило не работает. Посмотрите количество пакетов, сработавших на правиле: **iptables -L -v -x -n** Как видите, количество пакетов действительно мало по сравнению с общим числом пакетов. Модуль limit ограничивает количество срабатываний правила, а если правило не сработало, выполняется следующее правило в цепочке. Поскольку следующего правила нет, выполняется политика по умолчанию ACCEPT — пакеты принимаются.
3. Ввод дополнительного правила.	1. Для реального ограничения количества принимаемых пакетов добавьте ещё одно правило: **iptables -A INPUT -p icmp --icmp- type echo-request -j DROP** 2. Проверьте, как работает это правило. Попросите соседа выполнить следующую команду: **ping -f Ваш_IP -c 10000** 3. Обратите внимание на количество потерянных пакетов.

Критерий MAC

Критерий MAC позволяет при отборе пакетов в качестве критерия отбора использовать MAC адрес Ethernet пакета.

Критерий	Описание
--mac-source	Критерий определяет MAC адрес сетевого узла, передавшего пакет. MAC адрес должен указываться в форме XX:XX:XX:XX:XX:XX. Этот критерий имеет смысл только в цепочках PREROUTING, FORWARD и INPUT Символ “!” инвертирует значение параметра.

Критерий	Описание
	Пример: `--mac-source 00:00:00:00:00:01`

Критерий multiport

Критерий multiport позволяет в качестве параметра указывать несколько портов в любой последовательности.

Критерий	Описание
--source-port	Критерий служит для указания списка исходящих портов. С помощью критерия можно указать до 15 различных портов. Названия портов в списке должны отделяться друг от друга запятыми, пробелы не допустимы. Данное расширение может использоваться только совместно с критериями -p tcp или -p udp. Пример: `--source-port 21,23,53`
--destination-port	Критерий служит для указания списка входных портов. В остальном этот критерий аналогичен критерию --source-port.

Критерий state

Netfilter позволяет отслеживать соединения, причем даже для таких протоколов, как ICMP и UDP. В пределах netfilter соедине-ние может иметь одно из 4-х базовых состояний: NEW, ESTABLISHED, RELATED и INVALID. Для управления прохождением пакетов, основываясь на их состоянии, используется критерий state.

Трассировка соединений производится специальным кодом в пространстве ядра – трассировщиком (conntrack). В большинстве случаев требуется более специфичная информация о соединении, чем та, которую поставляет трассировщик по умолчанию. Поэтому трассировщик включает в себя обработчики различных протоколов, например TCP, UDP или ICMP. Собранная ими информация затем
используется для идентификации и определения текущего состояния соединения. Например – соединение по протоколу UDP однозначно идентифицируется по IP-адресам и портам источника и приемника.

В предыдущих версиях ядра имелась возможность включения/выключения поддержки дефрагментации пакетов. Однако после того, как трассировка соединений была включена в состав netfilter, надобность в этом отпала. Причина в том, что трассировщик не в состоянии выполнять возложенные на него функции без поддержки дефрагментации и поэтому она включена постоянно. Её нельзя отключить иначе, как отключив трассировку соединений. Де-фрагментация выполняется всегда, если трассировщик включен.

Трассировка соединений производится в цепочке PREROUTING, исключая случаи, когда пакеты создаются локальными процессами на брандмауэре. В этом случае трассировка производится в цепочке OUTPUT. Это означает, что netfilter производит все вычисления, связанные с определением состояния, в пределах этих цепочек. Когда локальный процесс на

брандмауэре отправляет первый пакет из потока, то в цепочке OUTPUT ему присваивается состояние NEW, а когда возвращается пакет ответа, то состояние соединения в цепочке PREROUTING изменяется на ESTABLISHED, и так да- лее. Если же соединение устанавливается извне, то состояние NEW присваивается первому пакету из потока в цепочке PREROUTING. Таким образом, определение состояния пакетов производится в пределах цепочек PREROUTING и OUTPUT таблицы nat.

Критерий	Описание
--limit	Критерий определяет состояние: NEW, EASTABLISHED, RELATED и INVALID. Пример: `--state ESTABLISHED,RELATED`

Состояния

Состояние	Описание
NEW	Признак NEW сообщает о том, что пакет является первым для данного соединения. Это означает, что это первый пакет в данном соединении, который увидел модуль трассировщика. Например, если получен SYN пакет, являющийся первым пакетом для данного соединения, то он получит статус NEW. Однако пакет может и не быть SYN пакетом и тем не менее получить статус NEW.
ESTABLISHED	Признак ESTABLISHED говорит о том, что пакет принадлежит уже установленному соединению.
RELATED	Соединение получает статус RELATED, если оно связано с другим соединением, имеющим признак ESTABLISHED. Это означает, что соединение получает признак RELATED тогда, когда оно инициировано из уже установленного соединения, имеющего признак ESTABLISHED. Хорошим примером соединения, которое может рассматриваться как RELATED, является соединение FTP-data, которое является связанным с портом FTP control, а так же DCC соединение, запущенное из IRC. Обратите внимание на то, что большинство протоколов TCP и некоторые из протоколов UDP весьма сложны и передают информацию о соединении через область данных TCP или UDP пакетов и поэтому требуют наличия специальных вспомогательных модулей для корректной работы.
INVALID	Признак INVALID говорит о том, что пакет не может быть идентифицирован и поэтому не может иметь определенного статуса. Это может происходить по разным причинам, например при нехватке памяти или при получении ICMP-сообщения об ошибке, которое не соответствует какому-либо известному соединению.

Критерий state позволяет сократить количество правил фильтрации, необходимых для работы.

Обычно самым первым в цепочках INPUT и FORWARD определяют правила, запрещающие приём пакетов с состояним INVALID:

```
iptables -A INPUT -m state --state INVALID -j
iptables -A FORWARD -m state --state INVALID -j
```

Затем определяют правила, принимающие пакеты уже установленных соединений:

```
iptables -A INPUT -m state –state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state –state ESTABLISHED, RELATED -j ACCEPT
```

Теперь достаточно разрешить выход в Internet на необходимые порты или машины, например:

```
iptables -A FORWARD -p tcp --dport 80 -j ACCEPT
```

Действия и переходы

ACCEPT — принять пакет DROP — сбрасывает пакет REJECT — сбрасывает пакет с сообщением об ошибке RETURN — возврат из цепочки. LOG — помещает информацию в журнальные файлы

Действия и переходы сообщают правилу, что необходимо выполнить, если пакет соответствует заданному критерию.

Переходы позволяют перейти в цепочку пользователя. В этом случае достаточно после параметра “-j” указать имя цепочки.

Действие – это предопределенная команда, описывающая действие, которое необходимо выполнить, если пакет совпал с задан- ным критерием. Например, можно применить действие DROP или ACCEPT. Существуют и другие типы действий, некоторые из них будут рассмотрены ниже.

Действие ACCEPT

Если над пакетом выполняется действие ACCEPT, то пакет прекращает движение по цепочке (и всем вызвавшим цепочкам, если текущая цепочка была вложенной) и считается ПРИНЯТЫМ.

Действие DROP

Действие просто “сбрасывает” пакет и netfilter “забывает” о его существовании. “Сброшенные” пакеты прекращают свое движение полностью, т.е. 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 REJECT

REJECT используется, как правило, в тех же самых ситуациях, что и DROP, но в отличие от

DROP, команда REJECT выдает сообщение об ошибке на хост, передавший пакет. Действие REJECT может использоваться только в цепочках INPUT, FORWARD и OUTPUT (и во вложенных в них цепочках).

Параметр	Описание
--reject-with	Указывает, какое сообщение необходимо передать в ответ, если пакет совпал с заданным критерием. При применении действия REJECT к пакету, сначала на хост отправителя будет отослан указанный ответ, а затем пакет будет “сброшен”. Допускается использовать следующие типы ответов: • icmp-net-unreachable • icmp-host-unreachable • icmp-port-unreachable • icmp-proto-unreachable • icmp-net-prohibited • icmp-host-prohibited По умолчанию передается сообщение port-unreachable.
--tcp-reset	Ещё один тип ответа, который используется только для протокола TCP. Если указано значение tcp-reset, то действие REJECT передаст в ответ пакет TCP RST. Пакеты TCP RST используются для закрытия TCP соединений. Пример: `-j REJECT --reject-with icmp-host-unreachable`

Действие RETURN

Действие RETURN прекращает движение пакета по текущей цепочке правил и производит возврат в вызывающую цепочку, если текущая цепочка была вложенной, или, если текущая цепочка лежит на самом верхнем уровне (например INPUT), то к пакету будет применена политика по умолчанию.

Действие LOG

Действие LOG передаёт информацию демону syslogd. В качестве информации передаются основные заголовки пакетов. Содержимое пакета в систему журнальной регистрации не попадает. Если необходимо передавать содержимое пакета, следует воспользоваться действием ULOG. Описание действия ULOG не входит в рамки данного курса.

При использовании действия LOG настоятельно рекомендуется включать критерий limit для ограничения информации, попадающей в систему журнальной регистрации.

Параметр	Описание
--log-level	Используется для задания уровня важности передаваемого сообщения. Все сообщения передаются средством kernel. Пример: `-j LOG --log-level debug`

Параметр	Описание
--log-prefix	Параметр задает текст, которым будут предваряться все сообщения iptables. Сообщения со специфичным префиксом затем легко можно найти, к примеру, с помощью grep. Префикс может содержать до 29 символов, включая пробелы. Пример: `-j LOG --log-prefix “Alert: “`

Лабораторная работа Б.

Цель работы.
Научиться использовать действие LOG.

Задача.
Необходимо помещать в систему журнальной регистрации информацию на уровне важности debug о пакетах протокола ICMP типа echo-request но не более, чем два пакета один раз в минуту.

Задачи	Описание
1. Подготовка к выполнению работы.	1. Очистите цепочку INPUT от всех правил: `iptables -F`
2. Добавление правила.	1. Добавьте в цепочку INPUT следующее правило: `iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/m --limit-burst 2 -j LOG --log-level debug –log-prefix “Alert: “`
3. Проверка работоспособности.	1. Попросите соседа, чтобы он выполнил следующую команду: `ping -c 10 Ваш_IP` 2. Обратите внимание на поле icmp_seq, последовательность не нарушается. Это значит, что несмотря на то, что правило сработало, пакет продолжает своё движение по цепочке. 3. Для просмотра передаваемой информации выполните программу dmesg или настройте syslog таким образом, чтобы сообщения на уровне debug попадали в журнальные файлы.

Вопросы.

1. При помощи какой команды можно добавить правило в конец цепочки?

2. При помощи какого параметра указывается таблица, с которой будет работать программа iptables?

3. На какие типы можно разделить критерии отбора пакетов? Чем они отличаются друг от друга?

4. Для чего предназначен критерий limit?

5. При помощи какого действия можно сбросить пакет с воз- вратом ICMP сообщения?

6. Какое действие можно применить для посылки информации в журнальные файлы?

NAT преобразования

NAT преобразования позволяют заменить в заголовках пакетов IP адрес источника или назначения, а также порт источника или назначения.

В netfilter существуют два основных действия, позволяющие осуществить NAT преобразования:

- SNAT — замена адреса и/или порта источника
- DNAT — замена адреса и/или порта назначения

Кроме того, у SNAT и DNAT существуют частные случаи:

- MASQUERADE — частный случай SNAT
- REDIRECT — частный случай DNAT

SNAT

Предположим, что существует сеть 10.10.100.0/24 и необходимо сделать так, чтобы компьютеры, находящиеся в этой сети, могли получить доступ в Internet.

Недостаточно просто разрешить хождение пакетов через роутер. Все пакеты, отправляемые с машин клиентов, в поле IP источника будут содержать IP адрес внутренней сети. Первый же роутер Вашего провайдера такие пакеты будет отбрасывать. Даже если пред- положить, что такой пакет дойдёт, например, до WEB сервера, сервер не будет знать, как ответить клиенту.

Ведь на роутерах в Internet не прописаны маршруты к частным сетям.

Наша задача при выходе пакета из нашего роутера заменить IP адрес частной сети на реальный IP внешнего интерфейса роутера, и обеспечить механизм обратного преобразования, когда пакеты будут возвращаться на роутер из Internet для передачи их машинам во внутренней сети.

Если выходной интерфейс имеет статический IP адрес, мож- но применять действие SNAT, которое позволяет осуществлять заме- ны IP адреса и/или порта источника. Предположим, что роутер под- ключён к Internet через интерфейс eth0 с IP адресом 1.2.3.4, а все пакеты посылаются из внутренней сети с машины с адресом 10.10.100.1. Тогда для замены IP адреса источника воспользуемся следующей командой:

```
iptables -t nat -A POSTROUTING -p tcp -o eth0-j SNAT --to-source 1.2.3.4
```

Все действия с NAT преобразованиями следует производить только в таблице nat (-t nat). SNAT преобразования можно делать

только в цепочке POSTROUTING (-A POSTROUTING). При использовании действия SNAT обязательно следует указывать протокол (-p tcp). Действия по замене будем производить только над теми пакетами, которые должны выходить в Internet, т.е. пакеты, выходящие че- рез интерфейс eth0 (-o eth0). IP адрес источника будем менять на IP адрес внешнего интерфейса — 1.2.3.4 (--to-source 1.2.3.4).

Для иллюстрации того, что будет происходить с пакетами можно представить следующую таблицу:

До SNAT		**После SNAT**	
IPs	Ports	IPs	Ports
10.10.100.1	4000	1.2.3.4	4000

Теперь предположим, что с другой машины в сети открывается ещё одно соединение с сервером в Internet и тоже с порта 4000. Получается неприятная ситуация. Все пакеты, которые будут выходить с роутера в Internet, будут видны как пакеты соединения откры- ваемого роутером с определённого порта. И получается, что два соединения открываются с одного и того же порта. Такого быть не должно. Поэтому при SNAT необходимо менять не только IP адрес источника, но и порт. Например, следующим образом:

```
iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to-source 1.2.3.4:30000-35000
```

Теперь при открытии соединения будет меняться и порт ис- точника. Причём будет выбираться следующий свободный порт из указанного диапазона:

До SNAT		**После SNAT**	
IPs	Ports	IPs	Ports
10.10.100.1	4000	1.2.3.4	30000

10.10.100.2	4000	1.2.3.4	30001

Поскольку netfilter отслеживает соединения, во всех пакетах приходящих обратно, IP адрес назначения автоматически меняется на IP адрес внутренней сети, и пакет попадает по назначению.

MASQUERADE

Действие MASQUERADE является частным случаем SNAT преобразования. Его рекомендуют применять в тех случаях, когда IP адрес получается динамически: при PPP соединении или от DHCP сервера.

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 имеет хорошее свойство – “забывать” соединения при остановке сетевого интерфейса. В случае же SNAT, в этой ситуации, в таблице трассировщика остаются данные о потерянных соединениях, и эти данные могут сохраняться до суток, поглощая ценную память. Эффект “забывчивости” связан с тем, что при остановке сетевого интерфейса с динамическим IP адресом есть вероятность на следующем запуске получить другой IP адрес, но в этом случае любые соединения все равно будут потеряны, и было бы глупо хранить трассировочную информацию.

Невзирая на положительные черты, маскарадинг не следует считать предпочтительным, поскольку он дает большую нагрузку на систему.

Для примера, описанного в предыдущей главе, вместо SNAT можно использовать следующую команду:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Обратите внимание на то, что в этом случае нет необходимости указывать протокол.

Дополнительно можно использовать необязательный ключ --to-ports — порт или диапазон портов.

DNAT

Предположим, что в DMZ (10.10.100.0/24) находится WEB-сервер (10.10.100.1) и необходимо организовать доступ к этому серверу из Internet. Внешний интерфейс роутера (eth0) имеет IP адрес 1.2.3.4. В DNS, машине www соответствует IP 1.2.3.4, т.е. все пакеты предназначенные нашему WEB серверу, будут приходить на этот IP на порт 80. Нам необходимо организовать пересылку этих пакетов на машину 10.10.100.1 на порт 80.

Для решения этой задачи можно воспользоваться действием DNAT, при помощи которого у проходящих пакетов меняются IP адрес и/или порт назначения.

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 10.10.100.1
```

Обязательный ключ --to-destination — определяет IP адрес и/или порт, на которые будут заменены соответствующие поля.

Теперь у всех пакетов TCP, направленных на порт 80, IP ад- реса назначения будут меняться на IP 10.10.100.1. Поскольку дей- ствие DNAT можно определять только на цепочке PREROUTING (до принятия решения о маршрутизации), после замены IP адреса, пакет будет перенаправлен по необходимому маршруту согласно стандарт- ной таблицы маршрутизации. Так как netfilter отслеживает соедине- ния, в памяти организуется таблица, похожая на таблицу, которую мы рассматривали в случае SNAT. У пакетов идущих обратно, IP адрес источника меняется автоматически.

При помощи DNAT можно организовать равномерное распределение соединений между несколькими серверами, обслуживающими одну и ту же службу. Например, в DMZ существуют три WEB сервера (10.10.100.1 — 10.10.100.3) и необходимо распределить между ними соединения.

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 10.10.100.1 10.10.100.3
```

Теперь новое соединение будет перенаправлено на сервер, у которого в данный момент времени наименьшее количество открытых соединений. Правда у этого метода есть один существенный недостаток — если один из серверов перестанет работать, netfilter это не “увидит” и будет пытаться новые соединения перенаправлять на неработающий сервер.

Лабораторная работа B.

Цель работы.
Научиться использовать NAT преобразования.

Задача.
Лабораторная работа выполняется двумя слушателями. Машина с нечётным IP будет выступать в роли клиента во внутренней сети. Машина с чётным IP адресом в роли роутера, на котором будут производиться NAT преобразования.

Задачи	Описание

1. Изменения на машине клиенте.	1. Отключите сетевой интерфейс eth0:
	`ifconfig eth0 down`
	2. Добавьте маршрут по умолчанию на сетевой интерфейс роутера, с которым Ваша машина связана по интерфейсу eth1:
	`route add default gw 192.168.2.2`

2. Включение NAT преобразования.	1. На роутере включите возможность пересылки пакетов между интерфейсами:
	`echo 1 > /proc/sys/net/ipv4/ip_forward`
	2. Добавьте действие MASQUERADING в таблицу nat:
	`iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE`

3. Проверка работоспособности.	1. На роутере запустите программу iptraf и включите режим просмотра пакетов на всех интерфейсах.

Задачи	Описание
	2. У слушателя на нечётной машине запустите браузер. 3. Отключите браузер от proxy сервера. 4. Откройте любую WEB страницу. 5. В программе iptraf посмотрите соединения и обратите внимание на то, что на любое соединение показаны два экземпляра соединения, одно на интерфейсе eth1, другое на интерфейсе eth0.

Вопросы

1. Какую разновидность SNAT рекомендуется использовать в случае получения IP адреса динамически?

2. Каким образом при определении действия SNAT можно указать IP адрес и диапазон портов, которые будут подстав- ляться во время преобразования?

3. Каким образом можно равномерно распределить соедине- ния между серверами?

Управление netfilter

После создания правил фильтрации необходимо сделать так, чтобы эти правила загружались автоматически после старта компьютера.

С iptables поставляются две программы, при помощи которых можно сохранять правила фильтрации в специальном файле, а затем восстанавливать их из этого файла.

```
iptables-save [-c] > file
```

Программа выводит текущие правила на стандартный вывод. Если указать опцию “-c”, с каждым правилом будут показаны значения счётчиков.

```
iptables-restore [-c] file
```

Программа устанавливает текущие правила фильтрации из указанного файла. Опция “-c” восстанавливает значения счётчиков.

Стартовые скрипты

Для автоматического включения правил фильтрации пристарте компьютера и для его автоматического выключения, при остановке компьютера, необходимо написать или использовать готовые стартовые скрипты.

В случае Ubuntu Linux стартовый скрипт уже существует, но я бы рекомендовал его удалить и написать свой собственный вариант.

Скрипт следует писать с учетом особенностей системы инициализации UpStart. То есть, он должен поддерживать как минимум две функции:

- start — запуск firewall
- stop — останов firewall

Кроме перечисленных функций рекомендуется использовать еще две:

- reset — сбрасывает firewall в какое либо первоначальное состояние и применяется когда вы редактируете правила firewall на удаленной машине
- init — используется для первоначальной инициализации firewall

Прежде чем начинать писать скрипт, вы должны четко представлять себе какие задачи должен решать firewall. Лучше всего нарисовать графически план прохождения пакетов. И только после этого приступать к написанию скрипта.

Напишем стартовый скрипт для Ubuntu Linux. Несмотря на то, что в этом дистрибутиве используется система инициализации UpStart, скрипт будет написан с учетом системы инициализации System V. Таким образом с минимальными переделками он может быть использован в дистрибутиве SuSE и RedHat Linux. В RedHat Linux следует использовать только две последних функции.

Сначала укажем интерпретатор и для удобства определим несколько переменных.

```
#!/bin/bash
IPT=/usr/sbin/iptables
IPTR=/usr/sbin/iptables-restore
IPTS=/usr/sbin/iptables-save
```

Функция старт должна организовать первоначальную инициализацию firewall. Для этого будет использована программа iptables-restore и файл /etc/iptables, который будет создаваться автоматически.

```
start()
{
echo -n "Starting firewall... "
$IPT -F
$IPT -t nat -F
$IPT -t mangle -F
$IPTR -c /etc/iptables echo "Done"
}
```

Функция stop должна записывать текущие правила фильтрации в файл /etc/iptables.

```
stop()
{
echo -n "Stop firewall... "
```

```
$IPTS -c > /etc/iptables echo "Done"
}
```

Функция reset должна сбрасывать правила в первоначальное состояние. В этом состоянии должен быть запрещен доступ ко всем службам, кроме удаленного соединения по ssh.

Как уже говорилось выше, эту функцию необходимо использовать в случае удаленного управления компьютером, при изменении в правилах фильтрации. Запуск скрипта с параметром reset (будет описан ниже) необходимо производить раз в 10-15 минут, при помощи CRON. Это необходимо для того, что бы в случае ошибки, пра- вила фильтрации вернулись в первоначальное состояние и вы получили доступ к удаленному компьютеру. После изменения правил, не забудьте убрать из CRON вызов скрипта.

```
reset()
{
$IPT -F
$IPT -t nat -F
$IPT -t mangle -F
$IPT -P INPUT DROP
$IPT -P FORWARD DROP
#===================
# STATE RULES ======
$IPT -A INPUT -m state --state INVALID -j DROP
$IPT -A FORWARD -m state --state INVALID -j
$IPT -A INPUT -m state –state ESTABLISHED,RELATED -j ACCEPT
$IPT -A FORWARD -m state –state ESTABLISHED,RELATED -j ACCEPT
# LOCALHOST ===========
$IPT -A FORWARD -i ! lo -s 127.0.0.1 -j DROP
$IPT -A INPUT -i ! lo -s 127.0.0.1 -j DROP
$IPT -A FORWARD -i ! lo -d 127.0.0.1 -j DROP
$IPT -A INPUT -i ! lo -d 127.0.0.1 -j DROP
$IPT -A INPUT -d 127.0.0.1 -j ACCEPT
$IPT -A INPUT -s 127.0.0.1 -j ACCEPT
# ICMP ENABLED =====
$IPT -A INPUT -p icmp -j ACCEPT
# SSH enable =======
$IPT -A INPUT -p tcp -i eth0 -m state –state NEW --dport 22 -m recent --update --seconds 20 -j DROP
$IPT -A INPUT -p tcp -i eth0 -m state –state NEW --dport 22 -m recent --set -j ACCEPT
$IPT -A INPUT -p tcp --dport 22 -j ACCEPT
}
```

В функции сбрасываются все правила, устанавливаются политики по умолчанию. Затем устанавливаются правила, связанные с модулем state. Разрешается хождение пакетов через интерфейс lo. Разрешается трафик icmp.

В заключении разрешается подключение к 22 порту (ssh). Но для разрешения используется модуль recent, который позволяет ограничить количество соединений с одного IP адреса. В нашем примере не более одного соединения в 20 секунд. Таким способом закрывается подбор паролей по ssh «роботами».

В функции init вы должны написать все остальные правила.

```
init()
{
```

```
echo -n "Init firewall... "
reset
# DNS ENABLE ==========
$IPT -A INPUT -p tcp --dport 53 -j ACCEPT
$IPT -A INPUT -p udp --dport 53 -j ACCEPT
# Apache ==============
$IPT -A INPUT -p tcp --dport 80 -j ACCEPT
# SMTP ================
$IPT -A INPUT -p tcp --dport 25 -j ACCEPT
# SPOP ================
$IPT -A INPUT -p tcp -i eth0 -m state –state NEW --dport 995 -m recent --update --seconds 20 -j DROP
$IPT -A INPUT -p tcp -i eth0 -m state –state NEW --dport 995 -m recent --set -j ACCEPT
# SAVE rules in file ==
$IPTS -c > /etc/iptables echo "Done"
}
```

Какие правила использовать в функции init зависит от задач, решаемых вашим сервером. В приведенном примере был открыт до ступ к серверам: DNS, WEB, SMTP и SPOP3 (порт 995). В конце функции происходит запись текущих правил в файл /etc/iptables.

В заключении необходимо написать код, который будет обрабатывать параметры командной строки и вызывать соответствующие функции.

```
case $1 in
        start) start ;;
        stop) stop ;;
        init) init ;;
        reset) reset ;;
*)
echo "Usage ufw start|stop|init|reset"
exit 88
esac
```

Обратите внимание на то, что запуск firewall из соображений безопасности рекомендуется выполнять до инициализации сетевых интерфейсов. А его останов, только после выключения интерфейсов.