Министерство науки и высшего образования Российской Федерации

Рязанский государственный радиотехнический университет им. В.Ф. Уткина

Брандмауэр iptables

Методические указания к практическим занятиям

Рязань 2020

УДК 681.3.06

Брандмауэр iptables: методические указания к практическим занятиям / Рязан. гос. радиотехн. ун-т.; сост. А.А. Митрошин, В.Г. Псоянц. – Рязань, 2020. – 16 с.

Содержат описание практического занятия, используемого в курсе «Операционная система Linux». Могут использоваться при изучении других курсов, связанных с операционной системой Linux.

Предназначены для студентов очной, заочной и очно-заочной форм обучения направления подготовки «Информатика и вычислительная техника». Могут использоваться для студентов других направлений подготовки.

Могут использоваться как методические указания к лабораторным работам в курсах, связанных с изучением операционной системы Linux и свободно распространяемого программного обеспечения.

Ил. --. Библиогр.: -- назв.

Операционная система Linux, сетевые возможности, netfilter, iptables

Печатается по решению редакционно-издательского совета Рязанского государственного радиотехнического университета.

Рецензент: кафедра САПР вычислительных средств Рязанского государственного радиотехнического университета (зав. кафедрой засл. деят. науки и техники РФ В.П.Корячко)

Брандмауэр iptables

Составители: Митрошин Александр Александрович

Псоянц Владимир Грикорович

Редактор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Корректор \_\_\_\_\_\_\_\_\_\_\_\_\_

Подписано в печать \_\_\_\_\_\_\_\_. Формат бумаги 60×84 1/16.

Бумага газетная. Печать трафаретная. Усл. печ. л. 1,0.

Уч-изд. л. 1,0. Тираж 50 экз. Заказ

Рязанский государственный радиотехнический университет.

390005, Рязань, ул. Гагарина, 59/1.

Редакционно-издательский центр РГРТУ.

**Брандмауэры в Linux**

Термин «брандмауэр» (эквивалентом является термин «firewall») пришел из автомобилестроения. Этим термином обозначался огнеупорный щит, используемый в автомобилях для защиты пассажиров от огня, возникающего в моторном отделении. В вычислительной техники брандмауэром называют аппаратное или программное средство, которое защищает пользователей от внешних угроз.

В Linux можно использовать две базовых разновидности брандмауэров: пакетные фильтры (packet filters) и брандмауэры proxy. Каждая из базовых разновидностей обладает двумя подразновидностями. Пакетные фильтры могут быть пересылающими (forwarding), то есть принимающими решение о том, передавать ли пакет из сети в сеть, и маскирующими (masquerading), то есть модифицирующими адрес-источник и адрес-приемник. Брандмауэры proxy могут быть стандартными (standart), которые иногда называют непроницаемыми (opaque), в которых клиент подключается через специальный порт, а передаваемые данные перенаправляются в другой порт, или прозрачными (transparent), в которых клиент не использует специальный порт, а программное обеспечение брандмауэра перенаправляет пакеты из сети в сеть прозрачно для пользователя.

**Пакетные фильтры**

Функционирование брандмауэров фильтрации пакетов основано на принципе, что вся информация, которая необходима для принятия решения о том, что делать с пакетом, содержится в заголовке этого пакета. В заголовке пакета содержится информация об адресе-источнике, адресе-приемнике, времени жизни (time to live - TTL) и многие другие. Всего в заголовке IP-пакета содержится 13 полей.

В более старых версиях Linux в качестве пакетного фильтра использовался ipchains, который впоследствии был заменен на более мощный iptables.

**Пакетный фильтр iptables**

Суть работы iptables состоит в следующем. Обработка сетевого пакета системой представляется как конвейерная обработка. Пакет нужно получить из сетевого интерфейса или от системного процесса, затем следует выяснить предполагаемый маршрут этого пакета, после чего отослать его через сетевой интерфейс, либо отдать какому-либо процессу. То есть необходимы три конвейера обработки пакетов:

- «получить – маршрутизировать - отослать» (действие маршрутизатора);

- «получить – маршрутизировать - отдать» (действие при получении пакета процессом);

- «взять – маршрутизировать - отослать» (действие при отсылке пакета процессом).

Между каждым из этих действий помещается модуль брандмауэра (межсетевого экрана), называемый цепочкой. Цепочка обрабатывает пакет, исследуя, изменяя или уничтожая его. Если пакет не был уничтожен, он передается дальше по конвейеру.

В этой схеме есть два исключения:

- доступ к исходящему пакету предоставляется только после принятия решения о его маршрутизации, поэтому последовательность «взять - маршрутизировать» остается необработанной, а цепочка обрабатывающая исходящие пакеты вставляется после маршрутизации;

- ограничения на «чужие» пакеты, исходящие не от нас и предназначенные не для нас существенно отличаются от ограничений на «свои» пакеты, поэтому после маршрутизации транзитные пакеты обрабатываются еще одной цепочкой.

Цепочка, обслуживающая последовательность действий «получить - маршрутизировать» называется PREROUTING (перед маршрутизацией), цепочка, обрабатывающая последовательность «маршрутизировать - отдать» - INPUT, а цепочка, стоящая непосредственно перед отсылкой пакета – POSTROUTIG (после маршрутизации).

В ipchains каждая цепочка представляла собой одну таблицу правил. В правиле определяются свойства пакета и действие, которое нужно выполнять со всеми пакетами, обладающие указанными свойствами. Когда пакет попадает в цепочку, к нему последовательно применяются правила. Если пакет не имеет свойств, требуемых первым правилом, к нему применяется второе, и так вплоть до последнего, правила по умолчанию, которое применяется к любому пакету. Если свойства пакета удовлетворяют правилу, над ним совершается действие. Действие DROP уничтожает пакет, действие ASSEPT немедленно выпускает его из цепочки, после чего пакет движется дальше по конвейеру. Некоторые действия, например LOG, никак не влияют на дальнейшее продвижение пакета, после их выполнения он продолжает двигаться по цепочке правил. Из этого следует, что действия ASSEPT или DROP могут применяться к пакету лишь один раз.

Цепочки iptables состоят из нескольких таблиц. Выходя из одной таблицы, пакет попадает в следующую, а последняя таблица, если завершается действием ACCEPT, передает пакет дальше на конвейер.

Хотя таблицы одинаковы, принято их использовать по разному назначению. Таблицу mangle используют для внесения изменений в служебную информацию пакета, таблицу filter – для определения того, не следует ли пакет уничтожить, а таблицу nat – для подмены сетевых адресов. Существует таблица raw, которая имеет очень узкую функцию. Ее единственное назначение – предоставить механизм для маркировки пакетов для отказа от отслеживания соединения.

Процесс обработки пакетов таблицами iptables без учета таблицы raw показан на рис. 1а, с учетом raw – на рис. 1б.

Когда пакет приходит на брандмауэр, он сначала попадает на сетевое устройство, перехватывается соответствующим драйвером и передается в ядро. Далее пакет проходит ряд таблиц и затем передается либо локальному приложению, либо переправляется на другую машину. Порядок следования транзитных пакетов, то есть пакетов, которые должны быть направлены другому компьютеру, показан в табл. 1

Сеть

Routing decision

Local process

Routing decision

Сеть

Рисунок 1а. Обработка пакетов таблицами iptables без учета таблицы raw

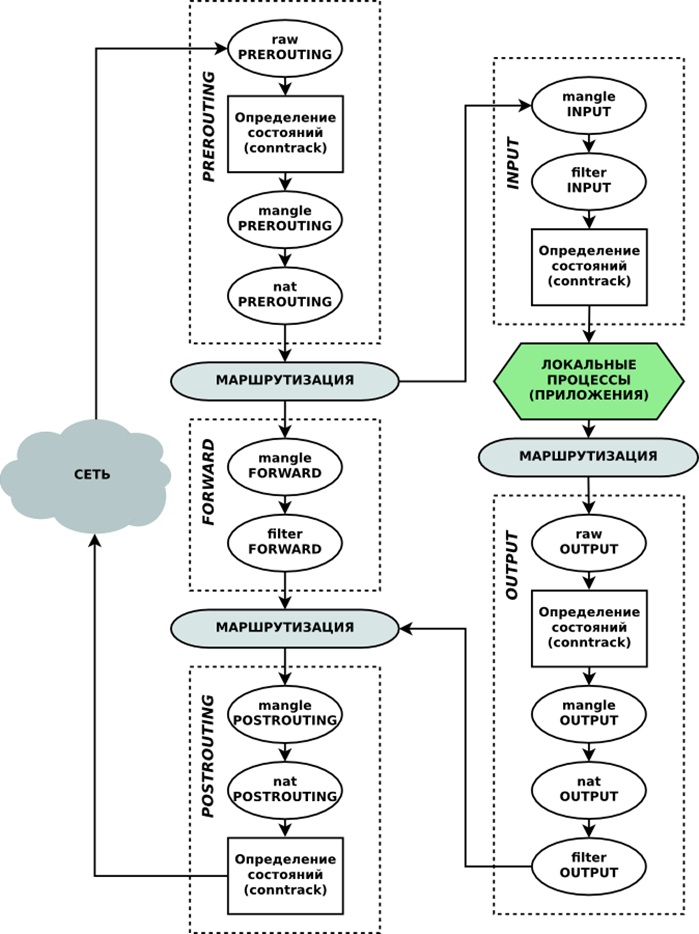


Рисунок 1б. Обработка пакетов таблицами iptables c учетом таблицы raw

Таблица 1. Порядок движения транзитных пакетов

|  |  |  |  |
| --- | --- | --- | --- |
| **Шаг** | **Таблица** | **Цепочка** | **Примечание** |
| 1 |  |  | Кабель |
| 2 |  |  | Сетевой интерфейс (например, eth0) |
| 3 | mangle | PREROUTING | Обычно эта цепочка используется для внесения изменений в заголовок пакета, например для изменения битов TOS и др.. |
| 4 | nat | PREROUTING | Эта цепочка используется для трансляции сетевых адресов (Destination Network Address Translation – трансляция адресов назначения). Source Network Address Translation (трансляция адресов источника) выполняется позднее, в другой цепочке. Любого рода фильтрация в этой цепочке может производиться только в исключительных случаях |
| 5 |  |  | Принятие решения о дальнейшей маршрутизации, т.е. в этой точке определяется, 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 предназначены для другого компьютера. Вся фильтрация транзитного трафика должна выполняться здесь. Через эту цепочку пакеты проходят в обоих направлениях. обязательно учитывайте это при написании правил фильтрации. |
| 8 | mangle | POSTROUTING | Цепочка предназначена для внесения изменений в заголовок пакета уже после того как принято последнее решение о маршрутизации. |
| 9 | nat | POSTROUTING | Цепочка предназначена в первую очередь для Source Network Address Translation. Не используйте ее для фильтрации без особой на то необходимости. Здесь же выполняется и маскарадинг (Masquerading). |
| 10 |  |  | Выходной сетевой интерфейс (например, eth1). |
| 11 |  |  | Кабель |

Цепочку FORWARD проходят все транзитные пакеты, которые движутся через брандмауэр. Не следует использовать цепочку INPUT для фильтрации транзитных пакетов, они туда не попадают. Через эту цепочку движутся только те пакеты, которые предназначены данному компьютеру.

Порядок движения пакета, предназначенного локальному процессу или приложению, показан в табл. 2.

Таблица 2. Порядок прохождения пакетов для локального приложения

|  |  |  |  |
| --- | --- | --- | --- |
| **Шаг** | **Таблица** | **Цепочка** | **Примечание** |
| 1 |  |  | Кабель |
| 2 |  |  | Входной сетевой интерфейс (например, eth0) |
| 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 | Производится фильтрация входящего трафика. Все входящие пакеты, адресованные нам, проходят через эту цепочку, независимо от того с какого интерфейса они поступили. |
| 8 |  |  | Локальный процесс или приложение |

В табл. 3 описан процесс движения пакетов, отправляемых локальными приложениями.

**Таблица 3. Движение пакетов от локальных процессов**

|  |  |  |  |
| --- | --- | --- | --- |
| **Шаг** | **Таблица** | **Цепочка** | **Примечание** |
| 1 |  |  | Локальный процесс ( программа-сервер или программа-клиент). |
| 2 |  |  | Принятие решения о маршрутизации. Определяется, куда пойдет пакет дальше - на какой адрес, через какой сетевой интерфейс и пр. |
| 3 | mangle | OUTPUT | Производится внесение изменений в заголовок пакета. Выполнение фильтрации в этой цепочке может иметь негативные последствия. |
| 4 | nat | OUTPUT | Цепочка используется для трансляции сетевых адресов (NAT) в пакетах, исходящих от локальных процессов брандмауэр. |
| 5 | filter | OUTPUT | Фильтруются исходящие пакеты. |
| 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. |
| 7 | nat | POSTROUTING | 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 (например, eth0) |
| 9 |  |  | Кабель |

**Назначение таблиц**

**Таблица mangle**

Таблица mangle (mangle - искажать, изменять) предназначена для внесения изменений в заголовки пакетов. В этой таблице не следует производить любого рода фильтрацию, маскировку или преобразование адресов (DNAT, SNAT, MASQUERADE).

В этой таблице допускается выполнять только следующие действия:

* TOS;
* TTL;
* MARK.

Действие **TOS** выполняет установку битов поля Type of Service (тип обслуживания) в пакете. Это поле используется для назначения сетевой политики обслуживания пакета, т.е. задает желаемый вариант маршрутизации. Однако, следует заметить, что данное свойство в действительности используется на незначительном количестве маршрутизаторов. Другими словами, не следует изменять состояние этого поля для пакетов, уходящих в Интернет, потому что на маршрутизаторах, которые обрабатывают это поле, может быть принято неправильное решение при выборе маршрута.

Действие **TTL** используется для установки значения поля TTL (Time To Live – время жизни) пакета. Есть одно неплохое применение этому действию. Мы можем присваивать определенное значение этому полю, чтобы скрыть наш брандмауэр от чересчур любопытных провайдеров (Internet Service Providers). Дело в том, что отдельные провайдеры очень не любят, когда одно подключение разделяется несколькими компьютерами. и тогда они начинают проверять значение TTL приходящих пакетов и используют его как один из критериев определения того, один компьютер "сидит" на подключении или несколько.

Действие **MARK** устанавливает специальную метку на пакет, которая затем может быть проверена другими правилами в iptables или другими программами. 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). 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. Другими словами, этим действием производится перенаправление пакетов на другие адреса, отличные от указанных в заголовках пакетов.

**Действие SNAT** (Source Network Address Translation) используется для изменения исходных адресов пакетов. С помощью этого действия можно скрыть структуру локальной сети, а заодно и разделить единственный внешний IP адрес между компьютерами локальной сети для выхода в Интернет. В этом случае брандмауэр, с помощью SNAT, автоматически производит прямое и обратное преобразование адресов, тем самым давая возможность выполнять подключение к серверам в Интернете с компьютеров в локальной сети.

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, т.е. когда происходит подключение к Интернет, скажем через PPP, SLIP или DHCP.

**Таблица filter**

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.

**Механизм определения состояний**

Механизм определения состояния (state machine) является отдельной частью iptables фактически является механизмом трассировки соединений. Трассировщик соединений создан для того, чтобы netfilter мог постоянно иметь информацию о состоянии каждого конкретного соединения. В пределах iptables, соединение может иметь одно из 4-х базовых состояний:

* NEW,
* ESTABLISHED,
* RELATED,
* INVALID.

Описание состояний приводится в табл. 4. Эти четыре состояния могут использоваться в критерии --state.

Таблица 4. Перечень состояний в пространстве пользователя

|  |  |
| --- | --- |
| 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 пакет SYN, 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. Это может породить определенные проблемы в отдельных случаях, но может оказаться полезным, например когда желательно "подхватить" соединения, "потерянные" другими брандмауэрами или в случаях, когда таймаут соединения уже истек, но само соединение не было закрыто. |
| 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. |
| ESTABLISHED | Состояние ESTABLISHED свидетельствует о том, что это не первый пакет в соединении. Для перехода в состояние ESTABLISHED необходимо чтобы узел сети передал пакет и получил на него ответ от другого узла (хоста). После получения ответа состояние соединения NEW или RELATED будет изменено на ESTABLISHED. |
| INVALID | Признак INVALID говорит о том, что пакет не может быть идентифицирован и поэтому не может иметь определенного статуса. Наилучшим вариантом было бы применение действия DROP к таким пакетам. |

Трассировка соединений производится в цепочке PREROUTING, исключая случаи, когда пакеты создаются локальными процессами на брандмауэре, в этом случае трассировка производится в цепочке OUTPUT.

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 далее. 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.

**TCP соединения**

TCP соединение всегда устанавливается передачей трех пакетов, которые инициализируют и устанавливают соединение, через которое в дальнейшем будут передаваться данные. Сессия начинается с передачи SYN пакета, в ответ на который передается SYN/ACK пакет и подтверждает установление соединения пакет ACK. После этого соединение считается установленным и готовым к передаче данных

Для всех типов соединений, трассировка проходит практически одинаково. На рис. 2 показаны стадии установления соединения.

Клиент

Firewall

Сервер

SYN

ACK

NEW

ESTABLISHED

SYN/ACK

Рисунок 2. Установление соединения TCP

Как только трассировщик видит первый (SYN) пакет, он присваивает ему статус NEW. Когда через трассировщик проходит второй пакет (SYN/ACK), то соединению присваивается статус ESTABLISHED.

Поэтому при построении набора правил можно позволить покидать локальную сеть пакетам со статусом NEW и ESTABLISHED, а во входящем трафике пропускать пакеты только со статусом ESTABLISHED. И наоборот, если бы трассировщик продолжал считать соединение как NEW, то фактически никогда не удалось бы установить соединение с «внешним миром», либо пришлось бы разрешить прохождение NEW пакетов в локальную сеть.

При закрытии, TCP соединение проходит через состояния, показанные на рис. 3. Соединение не закрывается до тех пор, пока не будет передан последний пакет ACK. Рис. 3 описывает нормальный процесс закрытия соединения. Кроме того, если соединение отвергается, то оно может быть закрыто передачей пакета RST (сброс). В этом случае соединение будет закрыто по истечение предопределенного времени.

Клиент

Firewall

Сервер

FIN/ACK

CLOSED

ESTABLISHED

ESTABLISHED

ACK

ACK

ESTABLISHED

CLOSED

FIN/ACK

CLOSED

Рисунок 3. Закрытие соединения TCP

**UDP соединения**

UDP соединения не имеют признака состояния. Этому имеется несколько причин, основная из них состоит в том, что этот протокол не предусматривает установления и закрытия соединения, но самый большой недостаток - отсутствие информации об очередности поступления пакетов. Приняв две датаграммы UDP, невозможно сказать точно в каком порядке они были отправлены. Однако, даже в этой ситуации все еще возможно определить состояние соединения. На рис. 4 показано, как выглядит установление соединения с точки зрения трассировщика.

Клиент

Firewall

Сервер

UDP Packet

ACK

NEW

ESTABLISHED

UDP Packet

Рисунок 3. Установление соединения UDP с точки зрения трассировщика

После того как сервер видит ответ на первый пакет, соединение считается ESTABLISHED (установленным).

**ICMP соединения**

ICMP пакеты используются только для передачи управляющих сообщений и не организуют постоянного соединения. Однако, существует 4 типа ICMP пакетов, которые вызывают передачу ответа, поэтому они могут иметь два состояния: NEW и ESTABLISHED. К этим пакетам относятся ICMP Echo Request/Echo Reply, ICMP Timestamp Request/Timestamp Reply, ICMP Information Request/Information Reply и ICMP Address Mask Request/Address Mask Reply (рис. 4).

Клиент

Firewall

Сервер

ICMP Echo

Reply

NEW

ESTABLISHED

ICMP Echo

Request

Client Processing

Рисунок 4. Состояния ICMP соединения

Как видно из рис. 4, сервер выполняет Echo Request (эхо-запрос) к клиенту, который распознается брандмауэром как NEW. На этот запрос клиент отвечает пакетом Echo Reply, и теперь пакет распознается как имеющий состояние ESTABLISHED.

Значительная часть ICMP используется для передачи сообщений о том, что происходит с тем или иным UDP или TCP соединением. В связи с этим они очень часто распознаются как связанные (RELATED) с существующим соединением. Простым примером могут служить сообщения ICMP Host Unreachable или ICMP Network Unreachable. Они всегда порождаются при попытке соединиться с узлом сети, когда этот узел или сеть недоступны, в этом случае последний маршрутизатор ответит соответствующим ICMP пакетом, который будет распознан как RELATED. На рис. 5 показано как это происходит.

Клиент

Firewall

Сервер

SYN

Client aborts

NEW

RELATED

ICMP Net Unreachable

Рисунок 5. Состояния ICMP соединения о происходящим с UDP или TCP соединением

В примере на рис. 5 некоторому узлу передается запрос на соединение (SYN пакет). Он приобретает статус NEW в брандмауэре. Однако, в этот момент времени, сеть оказывается недоступной, поэтому роутер возвращает пакет ICMP Network Unreachable. Трассировщик соединений распознает этот пакет как RELATED, благодаря уже имеющейся записи в таблице, так что пакет благополучно будет передан клиенту, который затем оборвет неудачное соединение. Тем временем, брандмауэр уничтожит запись в таблице, поскольку для данного соединения было получено сообщение об ошибке.

**Построение правил для iptables**

Каждое правило iptables - это строка, содержащая в себе критерии определяющие, подпадает ли пакет под заданное правило, и действие, которое необходимо выполнить в случае выполнения критерия. Правила записываются следующим образом:

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

Если правило не включает [-t table], то по умолчанию предполагается использование таблицы filter.

Наличие команды (command) обязательно.

Раздел match определяет критерии проверки, по которым определяется подпадает ли пакет под действие этого правила или нет. Могут быть указаны различные критерии: IP-адрес источника пакета или сети, IP-адрес места назначения, порт, протокол, сетевой интерфейс и т.д.

Раздел target указывает, какое действие должно быть выполнено при условии выполнения критериев проверки в правиле. Здесь можно передать пакет в другую цепочку правил, выдать на источник сообщение об ошибке и т.п.

**Команды iptables**

В табл. 5 приведен список команд и правила их использования. Посредством команд iptables получает инструкции о том, что требуется сделать. Обычно совершается одно из двух действий - добавление нового правила в цепочку или удаление существующего правила.

Таблица 5. Команды iptables

|  |
| --- |
| Команда: **-A**, **--append**  Добавляет новое правило в конец заданной цепочки.  Пример **iptables -A INPUT ...** |
| Команда: **-D**, **--delete**  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. Правила в цепочках нумеруются начинается с 1.  Примеры:  **iptables -D INPUT --dport 80 -j DROP**  **iptables -D INPUT 1** |
| Команда: **-R**, **--replace**  Заменяет одно правило другим. В основном она используется во время отладки новых правил.  Пример: **iptables -R INPUT 1 -s 192.168.0.1 -j DROP** |
| Команда: **-I**, **--insert**  Вставляет новое правило в цепочку. Число, следующее за именем цепочки, указывает номер правила, перед которым нужно вставить новое правило. В примере указывается, что данное правило должно быть 1-м в цепочке *INPUT*.  Пример: **iptables -I INPUT 1 --dport 80 -j ACCEPT** |
| Команда: **-L**, **--list**  Вывод списка правил в заданной цепочке. В 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.  Пример: **iptables -L INPUT** |
| Команда: **-F**, **--flush**  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.*  Пример: **iptables -F INPUT** |
| Команда: **-Z**, **--zero**  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.  Пример: **iptables -Z INPUT** |
| Команда: **-N**, **--new-chain**  Создается новая цепочка с заданным именем в заданной таблице В 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.)  Пример: **iptables -N allowed** |
| Команда: **-X**, **--delete-chain**  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.  Пример: **iptables -X allowed** |
| Команда: **-P**, **--policy**  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**.  Пример: **iptables -P INPUT DROP** |
| Команда: **-E**, **--rename-chain**  Команда **-E** выполняет переименование пользовательской цепочки. В примере цепочка allowed будет переименована в цепочку disallowed.  Пример: **iptables -E allowed disallowed** |

Команда должна быть указана всегда. Список доступных команд можно просмотреть с помощью команды iptables -h.

Некоторые команды могут использоваться совместно с дополнительными ключами. В табл. 6 приводится список дополнительных ключей и описывается результат их действия.

Таблица 6. Дополнительные ключи

|  |
| --- |
| Ключ: **-v**, **--verbose**  Используется для повышения информативности вывода и, как правило, используется совместно с командой **--list**. В случае использования с командой **--list**, в вывод этой команды включаются так же имя интерфейса, счетчики пакетов и байт для каждого правила. Формат вывода счетчиков предполагает вывод кроме цифр числа еще и символьные множители K (x1000), M (x1,000,000) и G (x1,000,000,000). Для того, чтобы заставить команду **--list** выводить полное число (без употребления множителей) требуется применять ключ **-x**, который описан ниже. Если ключ **-v**, **--verbose** используется с командами **--append**, **--insert**, **--delete** или **--replace**, то будет выведен подробный отчет о произведенной операции.  Команды, с которыми используется: **--list**, **--append**, **--insert**, **--delete**, **--replace** |
| Ключ: **-x**, **--exact**  Для всех чисел в выходных данных выводятся их точные значения без округления и без использования множителей K, M, G. Этот ключ используется только с командой **--list** и не применим с другими командами.  Команды, с которыми используется: **--list** |
| Ключ: **-n**, **--numeric**  Заставляет iptables выводить IP-адреса и номера портов в числовом виде предотвращая попытки преобразовать их в символические имена. Данный ключ используется только с командой **--list**.  Команды, с которыми используется: **--list** |
| Ключ: **--line-numbers**  Ключ **--line-numbers** включает режим вывода номеров строк при отображении списка правил командой **--list**. Номер строки соответствует позиции правила в цепочке. Этот ключ используется только с командой **--list**.  Команды, с которыми используется: **--list** |
| Ключ: **-c**, **--set-counters**  Этот ключ используется для установки начального значения счетчиков пакетов и байт в заданное значение при создании нового правила. Например, ключ --set-counters 20 4000 установит счетчик пакетов = 20, а счетчик байт = 4000.  Команды, с которыми используется: **--insert**, **--append**, **--replace** |
| Ключ: **--modprobe**  Ключ **--modprobe** определяет команду загрузки модуля ядра. Данный ключ может использоваться в случае, когда модули ядра находится вне пути поиска (search path). Этот ключ может использоваться с любой командой.  Команды, с которыми используется: Все |

**Критерии выделения пакетов**

Критерии выделения пакетов могут быть разбиты на пять групп:

- общие критерии, которые могут использоваться в любых правилах;

- TCP критерии, которые применяются только к TCP пакетам;

- UDP критерии, которые применяются только к UDP пакетам;

- ICMP критерии для работы с ICMP пакетами;

- специальные критерии, такие как state, owner, limit и пр.

**Общие критерии выделения пакетов**

Общие критерии допустимо употреблять в любых правилах, они не зависят от типа протокола.

Таблица 7. Общие критерии выделения пакетов

|  |
| --- |
| Критерий: **-p**, **--protocol**  Критерий используется для указания типа протокола. Примерами протоколов могут быть *TCP*, *UDP* и *ICMP*. Список протоколов можно посмотреть в файле [/etc/protocols](http://www.opennet.ru/docs/RUS/iptables/misc/iptables-tutorial/other/protocols.txt). Прежде всего, в качестве имени протокола в данный критерий можно передавать один из трех вышеупомянутых протоколов, а также ключевое слово ALL. В качестве протокола допускается передавать число - номер протокола, например, протоколу ICMP соответствует число 1, TCP - 6 и UDP - 17. Соответствия между номерами протоколов и их именами можно посмотреть в файле /etc/protocols. Если данному критерию передается числовое значение 0, то это эквивалентно использованию спецификатора ALL, который подразумевается по умолчанию, когда критерий **--protocol** не используется. Для логической инверсии критерия, перед именем протокола используется символ **!**, например **--protocol ! tcp** подразумевает пакеты протоколов, *UDP* и *ICMP*.  Пример: **iptables -A INPUT -p tcp** |
| Критерий: **-s**, **--src**, **--source**  IP-адрес(а) источника пакета. Адрес источника может указываться так, как показано в примере, тогда подразумевается единственный IP-адрес. Можно указать адрес в виде address/mask, например как *192.168.0.0/255.255.255.0*, или *192.168.0.0/24*, т.е. фактически определяя диапазон адресов. Символ !, установленный перед адресом, означает логическое отрицание, т.е. **--source ! 192.168.0.0/24** означает любой адрес кроме адресов *192.168.0.x*.  Пример: **iptables -A INPUT -s 192.168.1.1** |
| Критерий: **-d**, **--dst**, **--destination**  IP-адрес(а) получателя. Имеет синтаксис схожий с критерием **--source**, за исключением того, что подразумевает адрес места назначения. Может определять как единственный IP-адрес, так и диапазон адресов. Символ ! используется для логической инверсии критерия.  Пример: **iptables -A INPUT -d 192.168.1.1** |
| Критерий: **-i**, **--in-interface**  Интерфейс, с которого был получен пакет. Использование этого критерия допускается только в цепочках *INPUT*, *FORWARD* и *PREROUTING*, в других случаях будет вызывать сообщение об ошибке. При отсутствии этого критерия предполагается любой интерфейс, что равносильно использованию критерия **-i +**. Символ ! инвертирует результат совпадения. Если имя интерфейса завершается символом **+**, то критерий задает все интерфейсы, начинающиеся с заданной строки, например **-i PPP+** обозначает любой PPP интерфейс, а запись **-i ! eth+** -- любой интерфейс, кроме любого eth.  Пример: **iptables -A INPUT -i eth0** |
| Критерий: **-o**, **--out-interface**  Задает имя выходного интерфейса. Этот критерий допускается использовать только в цепочках *OUTPUT*, *FORWARD* и *POSTROUTING*, в противном случае будет генерироваться сообщение об ошибке. При отсутствии этого критерия предполагается любой интерфейс, что равносильно использованию критерия **-o +**. Символ ! инвертирует результат совпадения. Если имя интерфейса завершается символом **+**, то критерий задает все интерфейсы, начинающиеся с заданной строки, например **-o eth+** обозначает любой eth интерфейс, а запись **-o ! eth+** - любой интерфейс, кроме любого eth.  Пример: **iptables -A FORWARD -o eth0** |
| Критерий: **-f**, **--fragment**  Правило распространяется на все фрагменты фрагментированного пакета, кроме первого. Это сделано потому, что нет возможности определить исходящий/входящий порт для фрагмента пакета, а для ICMP-пакетов определить их тип. С помощью фрагментированных пакетов могут производиться атаки на ваш брандмауэр, так как фрагменты пакетов могут не отлавливаться другими правилами. Как и раньше, допускается использования символа ! для инверсии результата сравнения. только в данном случае символ ! должен предшествовать критерию **-f**, например **! -f**. Инверсия критерия трактуется как "все первые фрагменты фрагментированных пакетов и/или нефрагментированные пакеты, но не вторые и последующие фрагменты фрагментированных пакетов".  Пример: **iptables -A INPUT -f** |

**Неявные критерии выделения пакетов**

Неявные критерии - это критерии, которые подгружаются неявно и становятся доступны, например, при указании критерия --protocol tcp. Существует три автоматически подгружаемых расширения, это TCP критерии, UDP критерии и ICMP критерии. Загрузка этих расширений может производиться и явным образом с помощью ключа -m, -match, например -m tcp.

**Неявные критерии выделения пакетов TCP**

Этот набор критериев работает только с TCP пакетами. Чтобы использовать их потребуется в правилах указывать тип протокола --protocol tcp. Критерий --protocol tcp обязательно должен стоять перед специфичным критерием. Неявные критерии для протокола TCP приведены в табл. 8.

Таблица 8. Неявные критерии для протокола TCP

|  |
| --- |
| Критерий: **--sport**, **--source-port**  Исходный порт, с которого был отправлен пакет. В качестве параметра может указываться номер порта или название сетевой службы. Соответствие имен сервисов и номеров портов вы сможете найти в файле /etc/services. При указании номеров портов правила отрабатывают несколько быстрее. однако это менее удобно при разборе листингов скриптов. Если же вы собираетесь создавать значительные по объему наборы правил, скажем порядка нескольких сотен и более, то предпочтительнее использовать номера портов. Номера портов могут задаваться в виде интервала из минимального и максимального номеров, например **--source-port 22:80**. Если опускается минимальный порт, т.е. когда критерий записывается как **--source-port :80**, то в качестве начала диапазона принимается число 0. Если опускается максимальный порт, т.е. когда критерий записывается как **--source-port 22:**, то в качестве конца диапазона принимается число 65535. Допускается такая запись **--source-port 80:22**, в этом случае **iptables** поменяет числа 22 и 80 местами, т.е. подобного рода запись будет преобразована в **--source-port 22:80**. Как и раньше, символ **!** используется для инверсии. Так критерий **--source-port ! 22** подразумевает любой порт, кроме 22. Инверсия может применяться и к диапазону портов, например **--source-port ! 22:80**. За дополнительной информацией обращайтесь к описанию критерия multiport.  Пример: **iptables -A INPUT -p tcp --sport 22** |
| Критерий: **--dport**, **--destination-port**  Порт или диапазон портов, на который адресован пакет. Аргументы задаются в том же формате, что и для **--source-port**.  Пример: **iptables -A INPUT -p tcp --dport 22** |
| Критерий: **--tcp-flags**  Определяет маску и флаги tcp-пакета. Пакет считается удовлетворяющим критерию, если из перечисленных флагов в первом списке в единичное состояние установлены флаги из второго списка. Так для вышеуказанного примера под критерий подпадают пакеты у которых флаг *SYN* установлен, а флаги *FIN* и *ACK* сброшены. В качестве аргументов критерия могут выступать флаги *SYN, ACK, FIN, RST, URG, PSH*, а так же зарезервированные идентификаторы *ALL* и *NONE*. *ALL* -- значит *ВСЕ* флаги и *NONE* - НИ ОДИН флаг. Так, критерий **--tcp-flags ALL NONE** означает - *"все флаги в пакете должны быть сброшены"*. Символ **!** означает инверсию критерия Важно: имена флагов в каждом списке должны разделяться запятыми, пробелы служат для разделения списков.  Пример: **iptables -p tcp --tcp-flags SYN,FIN,ACK SYN** |
| Критерий: **--syn**  Критерию **--syn** соответствуют пакеты с установленным флагом *SYN* и сброшенными флагами *ACK* и *FIN*. Этот критерий аналогичен критерию **--tcp-flags SYN,ACK,FIN SYN**. Такие пакеты используются для открытия соединения *TCP*. Заблокировав такие пакеты, можно надежно заблокировать все входящие запросы на соединение, однако этот критерий не способен заблокировать исходящие запросы на соединение. Допускается инвертирование критерия символом **!**. Так критерий **! --syn** означает *«все пакеты, не являющиеся запросом на соединение»*, т.е. все пакеты с установленными флагами *FIN* или *ACK*.  Пример: **iptables -p tcp --syn** |
| Критерий: **--tcp-option**  Удовлетворяющим условию данного критерия будет пакет *TCP,* параметр которого равен заданному числу. *TCP Option* - это часть заголовка пакета. Она состоит из 3 различных полей. Первое 8-ми битовое поле содержит информацию об опциях, используемых в данном соединении. Второе 8-ми битовое поле содержит длину поля опций. Пакет, который не будет иметь полного *TCP* заголовка, будет сброшен автоматически при попытке изучения его *TCP* параметра. Как и ранее, допускается использование флага инверсии условия **!**.  Пример: **iptables -p tcp --tcp-option 16** |

**Неявные критерии для протокола UDP**

Неявные критерии для протокола UDP приведены в табл. 9.

Таблица 9. Неявные критерии для протокола UDP

|  |
| --- |
| Критерий: **--sport**, **--source-port**  Исходный порт, с которого был отправлен пакет. В качестве параметра может указываться номер порта или название сетевой службы. Соответствие имен сервисов и номеров портов вы сможете найти в файле other/services.txt. При указании номеров портов правила отрабатывают несколько быстрее. однако это менее удобно при разборе листингов скриптов. Если же вы собираетесь создавать значительные по объему наборы правил, скажем порядка нескольких сотен и более, то тут предпочтительнее использовать номера портов. Номера портов могут задаваться в виде интервала из минимального и максимального номеров, например **-source-port 22:80**. Если опускается минимальный порт, т.е. когда критерий записывается как **--source-port :80**, то в качестве начала диапазона принимается число 0. Если опускается максимальный порт, т.е. когда критерий записывается как **--source-port 22:** , то в качестве конца диапазона принимается число 65535. Допускается такая запись **--source-port 80:22** , в этом случае iptables поменяет числа 22 и 80 местами, т.е. подобного рода запись будет преобразована в **--source-port 22:80** . Как и раньше, символ **!** используется для инверсии. Так критерий **--source-port ! 22** подразумевает любой порт, кроме 22. Инверсия может применяться и к диапазону портов, например **--source-port ! 22:80**.  Пример: **iptables -A INPUT -p udp --sport 53** |
| Критерий: **--dport**, **--destination-port**  Порт, на который адресован пакет. Формат аргументов полностью аналогичен принятому в критерии **--source-port**.  Пример: **iptables -A INPUT -p udp --dport 53** |

**Неявный критерий для протокола ICMP**

Существует только один специфичный критерий для ICMP пакетов. Это расширение загружается автоматически, когда указывается критерий --protocol icmp. Для проверки ICMP пакетов могут использоваться и общие критерии, поскольку известны адрес источника, адрес назначения и др.

Неявный критерий для протокола ICMP приведен в табл. 10.

Таблица 10. Неявный критерий для протокола ICMP

|  |
| --- |
| Критерий: **--icmp-type**  Тип сообщения *ICMP* определяется номером или именем. Чтобы получить список имен *ICMP* значений выполните команду **iptables --protocol icmp --help**. Как и ранее, символ **!** инвертирует критерий, например **--icmp-type ! 8**.  Пример: **iptables -A INPUT -p icmp --icmp-type 8** |

**Явные критерии**

Перед использованием этих расширений, они должны быть загружены явно, с помощью ключа -m или --match. Так, например, если необходимо использовать критерии state, то нужно явно указать это в строке правила: -m state левее используемого критерия.

**Критерий limit**

Критерий определяет предельное число пакетов в единицу времени, которое способно пропустить правило. Критерий можно представить как некоторую емкость с выходным отверстием, через которое проходит определенное число пакетов за единицу времени (скорость вытекания). Скорость вытекания определяет величина --limit. Величина --limit-burst задает общий объем емкости.

Представим себе правило --limit 3/minute --limit-burst 5, тогда после поступления 5 пакетов, емкость наполнится и каждый последующий пакет будет вызывать переполнение емкости, т.е. срабатывание критерия. Через 1/3 минуты (в соответствии с величиной --limit) уровень в емкости будет понижен на 1, таким образом она готова будет принять еще один пакет, не вызывая переполнения емкости, т.е. не вызывая срабатывания критерия.

Таким образом, --limit-burst - это максимальное значение счетчика пакетов, при котором срабатывает ограничение, а --limit - это скорость, с которой счетчик burst limit уменьшает свое значение.

Критерий должен подгружаться явно ключом -m limit. Он хорошо подходит для правил, производящих запись в системный журнал (logging) и т.п. Ключи критерия limit приведены в табл. 11.

Таблица 11. Ключи критерия limit

|  |
| --- |
| Ключ: **--limit**  Устанавливается средняя скорость «освобождения емкости» за единицу времени. В качестве аргумента передаются число пакетов и единица измерения времени. Допустимы следующие единицы измерения времени: **/second** (секунда) **/minute** (минута) **/hour** (час) **/day** (день). По умолчанию принято значение 3 пакета в час, или **3/hour**. Использование флага инверсии условия **!** в данном критерии недопустимо.  Пример: **iptables -A INPUT -m limit --limit 3/hour** |
| Ключ: **--limit-burst**  Устанавливает максимальное значение числа *burst limit* для критерия **limit**. Это число увеличивается на единицу если получен пакет, подпадающий под действие данного правила, и при этом средняя скорость (задаваемая ключом **--limit**) поступления пакетов уже достигнута. Так происходит до тех пор, пока число *burst limit* не достигнет максимального значения, устанавливаемого ключом **--limit-burst**. После этого правило начинает пропускать пакеты со скоростью, задаваемой ключом **--limit**. Значение по умолчанию принимается равным 5.  Пример: **iptables -A INPUT -m limit --limit-burst 5** |

**Критерий MAC**

MAC (Ethernet Media Access Control) критерий используется для проверки исходного MAC-адреса пакета. Расширение -m mac требуется указывать явно. Ключи критерия MAC приведены в табл. 11.

Таблица 11. Ключи критерия MAC

|  |
| --- |
| Ключ: **--mac-source**  MAC адрес сетевого узла, передавшего пакет. MAC адрес должен указываться в форме XX:XX:XX:XX:XX:XX. Как и ранее, символ ! используется для инверсии критерия, например --mac-source ! 00:00:00:00:00:01, что означает – «пакет с любого узла, кроме узла, который имеет MAC адрес 00:00:00:00:00:01» . Критерий имеет смысл только в цепочках PREROUTING, FORWARD и INPUT.  Пример: **iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01** |

**Критерий Mark**

Критерий mark предоставляет возможность пометить пакеты специальным образом. Mark - специальное поле, которое существует только в области памяти ядра и связано с конкретным пакетом. Может использоваться в самых разнообразных целях, например, для ограничения трафика и его фильтрации. Поле mark представляет собой беззнаковое целое число в диапазоне от 0 до 4294967296 для 32-битных систем.

Ключ критерия mark приведен в табл. 12.

Таблица 12. Ключи критерия Mark

|  |
| --- |
| Ключ: **- -mark**  Критерий производит проверку пакетов, которые были предварительно помечены. Метки устанавливаются действием **MARK**, которое будет рассмотрено при описании действий и переходов. Все пакеты, проходящие через netfilter имеют специальное поле **mark**. Допускается использовать маску метки. В данном случае критерий будет выглядеть подобным образом: **--mark 1/1**. Если указывается маска, то выполняется логическое AND метки и маски.  Пример: **iptables -t mangle -A INPUT -m mark --mark 1** |

**Критерий multiport**

Расширение multiport позволяет указывать в тексте правила несколько портов и диапазонов портов.

Ключи критерия multiport приведены в табл. 13.

Таблица 13. Ключи критерия multiport

|  |
| --- |
| Ключ: **--source-port**  Служит для указания списка исходящих портов. С помощью данного критерия можно указать до 15 различных портов. Названия портов в списке должны отделяться друг от друга запятыми, пробелы в списке не допустимы. Данное расширение может использоваться только совместно с критериями **-p tcp** или **-p udp**. Главным образом используется как расширенная версия обычного критерия **--source-port**.  Пример: **iptables -A INPUT -p tcp -m multiport - -source-port 22,53,80,110** |
| Ключ: **- -destination-port**  Служит для указания списка входных портов. Формат задания аргументов полностью аналогичен **-m multiport --source-port**.  Пример: **iptables -A INPUT -p tcp -m multiport - -destination-port 22,53,80,110** |
| Ключ: **--port**  Критерий проверяет как исходящий, так и входящий порт пакета. Формат аргументов аналогичен критерию **--source-port** и **--destination-port**. Критерий проверяет порты обеих направлений, т.е. под критерий **-m multiport --port 80** подпадают пакеты, идущие с порта 80 на порт 80.  Пример: **iptables -A INPUT -p tcp -m multiport --port 22,53,80,110** |

**Критерий owner**

Расширение owner предназначено для проверки «владельца» пакета. Допускается использовать этот критерий только в цепочке OUTPUT. Такое ограничение наложено потому, что на сегодняшний день нет реального механизма передачи информации о владельце по сети. Справедливости ради следует отметить, что для некоторых пакетов невозможно определить владельца в этой цепочке. К такого рода пакетам относятся различные ICMP responses. Поэтому не следует применять этот критерий к ICMP responses пакетам.

Ключи критерия owner приведены в табл. 14.

Таблица 14. Ключи критерия owner

|  |
| --- |
| Ключ: **--uid-owner**  Производится проверка "владельца" по User ID (UID). Подобного рода проверка может использоваться, к примеру, для блокировки выхода в Интернет отдельных пользователей.  Пример: **iptables -A OUTPUT -m owner --uid-owner 500** |
| Ключ: **--gid-owner**  Производится проверка "владельца" пакета по Group ID (GID).  Пример: **iptables -A OUTPUT -m owner --gid-owner 0** |
| Ключ: **--pid-owner**  Производится проверка "владельца" пакета по *Process ID* (PID). Этот критерий достаточно сложен в использовании, например, если мы хотим позволить передачу пакетов на *HTTP* порт только от заданного демона, то нам потребуется написать небольшой сценарий, который получает *PID* процесса (хотя бы через ps) и затем подставляет найденный *PID* в правила.  Пример: **iptables -A OUTPUT -m owner --pid-owner 78** |
| Ключ: **--sid-owner**  Производится проверка *Session ID* пакета. Значение *SID* наследуются дочерними процессами от родителя, так, например, все процессы *HTTPD* имеют один и тот же *SID*.  Пример: **iptables -A OUTPUT -m owner --sid-owner 100** |

**Критерий state**

Критерий state используется совместно с кодом трассировки соединений и позволяет получать информацию о признаке состояния соединения, что позволяет судить о состоянии соединения, причем даже для таких протоколов как ICMP и UDP. Данное расширение необходимо загружать явно, с помощью ключа -m state.

Ключи критерия state приведены в табл. 15.

Таблица 15. Ключи критерия state

|  |
| --- |
| Ключ: **--state**  Проверяется признак состояния соединения (state). Используются 4 состояния: **INVALID**, **ESTABLISHED**, **NEW** и **RELATED**. **INVALID.** Подразумевает, что пакет связан с неизвестным потоком или соединением и, возможно содержит ошибку в данных или в заголовке. Состояние **ESTABLISHED** указывает на то, что пакет принадлежит уже установленному соединению, через которое пакеты идут в обеих направлениях. Признак **NEW** подразумевает, что пакет открывает новое соединение или принадлежит однонаправленному потоку. Признак **RELATED** указывает на то что пакет принадлежит уже существующему соединению, но при этом он открывает новое соединение Примером тому может служить передача данных по *FTP*, или выдача сообщения *ICMP* об ошибке, которое связано с существующим *TCP* или *UDP* соединением. Признак **NEW** это не то же самое, что установленный бит *SYN* в пакетах *TCP*, посредством которых открывается новое соединение, и, подобного рода пакеты, могут быть потенциально опасны в случае, когда для защиты сети используется один сетевой экран.  Пример: **iptables -A INPUT -m state --state RELATED,ESTABLISHED** |

**Критерий tos**

Критерий TOS предназначен для проведения проверки битов поля TOS. TOS - Type Of Service -- представляет собой 8-ми битовое, поле в заголовке IP-пакета. Модуль должен загружаться явно, ключом -m tos.

Поле tos служит для нужд маршрутизации пакета. Установка любого бита может привести к тому, что пакет будет обработан маршрутизатором не так как пакет со сброшенными битами TOS. Каждый бит поля TOS имеет свое значение. В пакете может быть установлен только один из битов этого поля, поэтому комбинации не допустимы. Каждый бит определяет тип сетевой службы:

* ***минимальная задержка*** - используется в ситуациях, когда время передачи пакета должно быть минимальным, т.е., если есть возможность, то маршрутизатор для такого пакета будет выбирать более скоростной канал. Например, если есть выбор между оптоволоконной линией и спутниковым каналом, то предпочтение будет отдано более скоростному оптоволокну;

- ***максимальная пропускная способность*** - указывает, что пакет должен быть переправлен через канал с максимальной пропускной способностью. Например, спутниковые каналы, обладая большей задержкой имеют высокую пропускную способность;

- ***максимальная надежность*** - выбирается максимально надежный маршрут во избежание необходимости повторной передачи пакета;

- ***минимальные затраты*** - применяется в случаях, когда важно минимизировать денежные затраты на передачу данных. Например, при передаче на другой континент аренда спутникового канала может оказаться дешевле, чем аренда оптоволоконного кабеля;

- ***обычный сервис*** - в данной ситуации все биты поля TOS сброшены. Маршрутизация такого пакета полностью отдается на усмотрение провайдера.

Ключи критерия tos приведены в табл. 16.

Таблица 16. Ключи критерия TOS

|  |
| --- |
| Ключ: **--tos**  Критерий предназначен для проверки установленных битов *TOS*. Как правило поле используется для нужд маршрутизации, но вполне может быть использовано с целью "маркировки" пакетов для использования с iproute2 и дополнительной маршрутизации в linux. В качестве аргумента критерию может быть передано десятичное или шестнадцатиричное число, или мнемоническое описание бита, мнемоники и их числовое значение можно получить выполнив команду **iptables -m tos -h**. Далее приводятся мнемоники и их значения: Minimize-Delay 16 (0x10) (Минимальная задержка), Maximize-Throughput 8 (0x08) (Максимальная пропускная способность), Maximize-Reliability 4 (0x04) (Максимальная надежность), Minimize-Cost 2 (0x02) (Минимальные затраты), Normal-Service 0 (0x00) (Обычный сервис)  Пример: **iptables -A INPUT -p tcp -m tos --tos 0x16** |

**Критерий TTL**

TTL (Time To Live) является числовым полем в IP заголовке. При прохождении очередного маршрутизатора, это число уменьшается на 1. Если число становится равным нулю, то отправителю пакета будет передано ICMP сообщение типа 11 с кодом 0 (TTL equals 0 during transit) или с кодом 1 (TTL equals 0 during reassembly) . Для использования этого критерия необходимо явно загружать модуль ключом -m ttl.

С ключами критерия ttl можно познакомиться, используя справочную систему.

**Критерий "мусора" (Unclean match)**

Критерий unclean не имеет дополнительных ключей и для его использования достаточно явно загрузить модуль. Данная проверка производится для вычленения пакетов, которые имеют расхождения с принятыми стандартами, это могут быть пакеты с поврежденным заголовком или с неверной контрольной суммой и т.д., однако использование этой проверки может привести к разрыву и вполне корректного соединения.

**Действия и переходы**

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

Действие - это предопределенная команда, описывающая действие, которое необходимо выполнить, если пакет совпал с заданным критерием. Чаще всего используются действия ACCEPT и DROP.

Описание переходов в правилах производится также, как и описание действий, т.е. ставится ключ -j и указывается название цепочки правил, на которую выполняется переход. На переходы накладывается ряд ограничений:

- цепочка, на которую выполняется переход, должна находиться в той же таблице, что и цепочка, из которой этот переход выполняется,

- цепочка, являющаяся целью перехода должна быть создана до того как на нее будут выполняться переходы. Например, создадим цепочку tcp\_packets в таблице filter с помощью команды

iptables -N tcp\_packets

Теперь можно выполнять переходы на эту цепочку:

iptables -A INPUT -p tcp -j tcp\_packets

Встретив пакет протокола tcp, iptables произведет переход на цепочку tcp\_packets и продолжит движение пакета по этой цепочке. 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 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 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 бишь пропускается), тем не менее, пакет продолжит движение по цепочкам в других таблицах и может быть отвергнут там. Действие задается с помощью ключа -j ACCEPT.

**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. Данное действие может использоваться для предоставления доступа к web-серверу, находящемуся в локальной сети, и не имеющему реального IP адреса. Для этого создается правило, которое перехватывает пакеты, идущие на HTTP порт брандмауэра и выполняя DNAT передаются их на локальный адрес web-сервера. 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. Вложенные подцепочки, реализующие DNAT не должны вызываться из других цепочек, кроме PREROUTING и OUTPUT.

Ключ действия DNAT описан в табл. 17.

Таблица 17. Ключ действия DNAT

|  |
| --- |
| Ключ: **--to-destination**  Ключ **--to-destination** указывает, какой IP адрес должен быть подставлен в качестве адреса места назначения. В выше приведенном примере во всех пакетах, пришедших на адрес 15.45.23.67, адрес назначения будет изменен на один из диапазона от 192.168.1.1 до 192.168.1.10. Как уже указывалось выше, все пакеты из одного потока будут направляться на один и тот же адрес, а для каждого нового потока будет выбираться один из адресов в указанном диапазоне случайным образом. Можно также определить единственный IP адрес. Можно дополнительно указать порт или диапазон портов, на который (которые) будет перенаправлен траффик. Для этого после ip адреса через двоеточие укажите порт, например **--to-destination 192.168.1.1:80**, а указание диапазона портов выглядит так: **--to-destination 192.168.1.1:80-100**. Как вы можете видеть, синтаксис действий **DNAT** и **SNAT** во многом схож. Не забывайте, что указание портов допускается только при работе с протоколом TCP или UDP, при наличии опции **--protocol** в критерии.  Пример: **iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.10** |

Действие DNAT достаточно сложно. Рассмотрим следующий пример. Есть WEB сервер и необходимо разрешить доступ к нему из Интернет. Мы имеем только один реальный IP адрес, а WEB-сервер расположен в локальной сети. Реальный IP адрес $INET\_IP назначен брандмауэру, HTTP сервер имеет локальный адрес $HTTP\_IP и, наконец брандмауэр имеет локальный адрес $LAN\_IP.

Сначала добавим простое правило в цепочку PREROUTING таблицы nat:

iptables -t nat -A PREROUTING --dst $INET\_IP -p tcp --dport 80 -j DNAT --to-destination $HTTP\_IP

В соответствии с этим правилом, все пакеты, поступающие на порт 80 адреса $INET\_IP перенаправляются на наш внутренний WEB-сервер. Если теперь обратиться к WEB-серверу из Интернет, то все будет работать. Но при попытке соединиться с ним из локальной сети соединение не установится. Рассмотрим, как маршрутизируются пакеты, идущие из Интернет на WEB-сервер. Примем адрес клиента в Интернет равным $EXT\_BOX.

1) Пакет покидает клиентский узел с адресом $EXT\_BOX и направляется на $INET\_IP.

2) Пакет приходит на брандмауэр.

3) Брандмауэр, в соответствии с вышеприведенным правилом, подменяет адрес назначения и передает его дальше, в другие цепочки.

4) Пакет передается на $HTTP\_IP.

5) Пакет поступает на HTTP сервер и сервер передает ответ через брандмауэр, если в таблице маршрутизации он обозначен как шлюз для $EXT\_BOX. Как правило, он назначается шлюзом по умолчанию для HTTP сервера.

6) Брандмауэр производит обратную подстановку адреса в пакете, теперь все выглядит так, как будто бы пакет был сформирован на брандмауэре.

7) Пакет передается клиенту $EXT\_BOX.

Теперь рассмотрим, что произойдет, если запрос посылается с узла, расположенного в той же локальной сети. Примем адрес клиента в локальной сети равным $LAN\_BOX.

1. Пакет покидает $LAN\_BOX.
2. Пакет поступает на брандмауэр.

3) Производится подстановка адреса назначения, однако адрес отправителя не подменяется, т.е. исходный адрес остается в пакете без изменения.

4) Пакет покидает брандмауэр и отправляется на HTTP сервер.

5) HTTP сервер, готовясь к отправке ответа, обнаруживает, что клиент находится в локальной сети (поскольку пакет запроса содержал оригинальный IP адрес, который теперь превратился в адрес назначения) и поэтому отправляет пакет непосредственно на $LAN\_BOX.

6) Пакет поступает на $LAN\_BOX. Клиент «путается», поскольку ответ пришел не с того узла, на который отправлялся запрос. Поэтому клиент сбрасывает пакет ответа и продолжает ждать настоящий ответ.

Проблема решается довольно просто с помощью SNAT. Ниже приводится правило, которое выполняет эту функцию. Это правило вынуждает HTTP сервер передавать ответы на наш брандмауэр, которые затем будут переданы клиенту.

iptables -t nat -A POSTROUTING -p tcp --dst $HTTP\_IP --dport 80 -j SNAT --to-source $LAN\_IP

Цепочка POSTROUTING обрабатывается самой последней и к этому моменту пакет уже прошел процедуру преобразования DNAT, поэтому критерий строится на базе адреса назначения $HTTP\_IP.

Теперь представим себе ситуацию, когда в качестве клиента выступает сам брандмауэр. Тогда пакеты будут передаваться на локальный порт с номером 80 самого брандмауэра, а не на $HTTP\_IP. Чтобы разрешить и эту проблему, добавим правило:

iptables -t nat -A OUTPUT --dst $INET\_IP -p tcp --dport 80 -j DNAT --to-destination $HTTP\_IP

**Действие DROP**

Данное действие сбрасывает пакет и iptables 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. Это 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 интересующая информация. Информация из журнала может быть затем прочитана с помощью команд dmesg или syslogd.

Превосходное средство для отладки ваших правил. Желательно на период отладки правил вместо действия DROP использовать действие LOG, чтобы убедиться, что брандмауэр работает правильно.

Существует действие ULOG, которое позволяет выполнять запись журналируемой информации не в системный журнал, а в базу данных MySQL и т.п..

Действие LOG имеет ряд ключей, которые перечислены в табл. 18.

Таблица 18. Ключи действия LOG

|  |
| --- |
| Ключ: **--log-level**  Используется для задания уровня журналирования (log level). Полный список уровней можно найти в man по syslog.conf. Обычно используются следующие уровни: *debug*, *info*, *notice*, *warning*, *warn*, *err*, *error*, *crit*, *alert*, *emerg* и *panic*. Ключевое слово *error* означает то же самое, что и *err*, *warn* - *warning* и *panic* - *emerg*. В последних трех парах слов не следует использовать *error*, *warn* и *panic*. Приоритет определяет различия в том, как будут заноситься сообщения в журнал. Все сообщения заносятся в журнал средствами ядра. Если установить **kern.=info /var/log/iptables** в файле syslog.conf, то все сообщения из **iptables**, использующие уровень *info*, будут заноситься в файл /var/log/iptables. Однако, в этот файл попадут и другие сообщения, поступающие из других подсистем, которые используют уровень *info*.  Пример: **iptables -A FORWARD -p tcp -j LOG --log-level debug** |
| Ключ: **--log-prefix**  Ключ задает текст (префикс), которым будут предваряться все сообщения **iptables**. Сообщения со специфичным префиксом затем легко можно найти с помощью **grep**. Префикс может содержать до 29 символов, включая и пробелы.  Пример: **iptables -A INPUT -p tcp -j LOG --log-prefix "INPUT packets"** |
| Ключ: **--log-tcp-sequence**  Ключ позволяет заносить в журнал номер *TCP Sequence* пакета. Номер *TCP Sequence* идентифицирует каждый пакет в потоке и определяет порядок сборки потока.  Пример: **iptables -A INPUT -p tcp -j LOG --log-tcp-sequence**  Ключ: **--log-tcp-options**  Этот ключ позволяет заносить в системный журнал различные сведения из заголовка TCP пакета. Такая возможность может быть полезна при отладке. Этот ключ не имеет дополнительных параметров, как и большинство ключей действия **LOG**.  Пример: **iptables -A FORWARD -p tcp -j LOG --log-tcp-options** |
| Ключ: **--log-ip-options**  Ключ позволяет заносить в системный журнал различные сведения из заголовка IP пакета. Во многом схож с ключом **--log-tcp-options**, но работает только с IP заголовком.  Пример: **iptables -A FORWARD -p tcp -j LOG --log-ip-options** |

**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.. Метка пакета существует только в период времени пока пакет не покинул брандмауэр, т.е. метка не передается по сети.

Ключ действия MARK описан в табл. 19.

Таблица 19. Ключи действия MARK

|  |
| --- |
| Ключ: **--set-mark**  Ключ устанавливает метку на пакет. После ключа **--set-mark** должно следовать целое беззнаковое число.  Пример: **iptables -t mangle -A PREROUTING -p tcp --dport 22 -j MARK --set-mark 2** |

**Действие MASQUERADE**

Маскарадинг (MASQUERADE) представляет то же самое, что и SNAT только не имеет ключа --to-source. Причиной тому то, что маскарадинг может работать, например, с dialup подключением или DHCP, т.е. в тех случаях, когда IP адрес присваивается устройству динамически. Если имеется динамическое подключение, то нужно использовать маскарадинг, если же статическое IP подключение, то 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 имеет 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. 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, поэтому не целесообразно хранить трассировочную информацию.

Действие MASQUERADE может быть использовано вместо SNAT, даже если имеется постоянный IP адрес, однако, маскарадинг не следует считать предпочтительным в этом случае, поскольку он создает большую нагрузку на систему.

Действие MASQUERADE допускается указывать только в цепочке POSTROUTING таблицы nat, так же, как и действие SNAT. MASQUERADE имеет ключ, описываемый в табл. 20, использование которого необязательно.

Таблица 20. Ключ действия MASQUERADE

|  |
| --- |
| Ключ: **--to-ports**  Ключ **--to-ports** используется для указания порта источника или диапазона портов исходящего пакета. Можно указать один порт, например: **--to-ports 1025**, или диапазон портов как здесь: **--to-ports 1024-3000**. Этот ключ можно использовать только в правилах, где критерий содержит явное указание на протокол TCP или UDP с помощью ключа **--protocol**.  Пример: **iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE --to-ports 1024-31000** |

**Действие MIRROR**

Действие MIRROR может использоваться только для экспериментов и в демонстрационных целях, поскольку это действие может привести к зацикливанию пакета и в результате к отказу от обслуживания. В результате действия MIRROR в пакете, поля source и destination меняются местами (invert the source and destination fields) и пакет отправляется в сеть. Использование этой команды может иметь весьма забавный результат, наверное, со стороны довольно потешно наблюдать, как кто-то пытается взломать свой собственный компьютер.

Данное действие допускается использовать только в цепочках INPUT, FORWARD и PREROUTING, и в цепочках, вызываемых из этих трех. Пакеты, отправляемые в сеть действием MIRROR больше не подвергаются фильтрации, трассировке или NAT, избегая тем самым зацикливания и других неприятностей. Однако это не означает, что проблем с этим действием нет.

**Действие QUEUE**

Действие QUEUE ставит пакет в очередь на обработку пользовательскому процессу. Оно может быть использовано для нужд учета или дополнительной фильтрации пакетов. Более подробную информацию можно получить в справочной системе.

**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 удобно для выполнения прозрачного проксирования (transparent proxying), когда машины в локальной сети даже не подозревают о существовании прокси.

REDIRECT может использоваться только в цепочках PREROUTING и OUTPUT таблицы nat. Это действие можно выполнять в подцепочках, вызываемых из вышеуказанных. Для действия REDIRECT существует ключ, описанный в табл. 21.

Таблица 21. Действие REDIRECT

|  |
| --- |
| Ключ: **--to-ports**  Ключ **--to-ports** определяет порт или диапазон портов назначения. Без указания ключа **--to-ports**, перенаправления не происходит, т.е. пакет идет на тот порт, куда и был назначен. В примере, приведенном выше, **--to-ports 8080** указан один порт назначения. Если нужно указать диапазон портов, то мы должны написать нечто подобное **--to-ports 8080-8090**. Этот ключ можно использовать только в правилах, где критерий содержит явное указание на протокол TCP или UDP с помощью ключа **--protocol**.  Пример: **iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080** |

**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 в цепочках INPUT, FORWARD и OUTPUT (и во вложенных в них цепочках). Существует ключ, управляющий поведением команды REJECT, описанный в табл. 22.

Таблица 22. Ключ действия REJECT

|  |
| --- |
| Ключ: **--reject-with**  Указывает, какое сообщение необходимо передать в ответ, если пакет совпал с заданным критерием. При применении действия **REJECT** к пакету, сначала на хост-отправитель 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 и icmp-host-prohibited. По умолчанию передается сообщение port-unreachable. Все вышеуказанные типы ответов являются *ICMP error messages*. Тип ответа **tcp-reset** 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: **iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset** |

**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. Обычно, в качестве политики по умолчанию назначают действия ACCEPT или DROP.

Допустим, что пакет движется по цепочке INPUT и встречает правило, которое производит переход во вложенную цепочку - --jump EXAMPLE\_CHAIN. Далее, в цепочке EXAMPLE\_CHAIN пакет встречает правило, которое выполняет действие --jump RETURN. Тогда произойдет возврат пакета в цепочку INPUT. Другой пример, пусть пакет встречает правило, которое выполняет действие --jump RETURN в цепочке INPUT. Тогда к пакету будет применена политика по умолчанию цепочки INPUT.

**Действие SNAT**

SNAT используется для преобразования сетевых адресов (Source Network Address Translation), т.е. изменение исходящего IP адреса в IP заголовке пакета. Например, это действие можно использовать для предоставления выхода в Интернет другим компьютерам из локальной сети, имея лишь один уникальный IP адрес. Для этого. необходимо включить пересылку пакетов (forwarding) в ядре и затем создать правила, которые будут транслировать исходящие IP адреса нашей локальной сети в реальный внешний адрес. В результате, внешний мир ничего не будет знать о нашей локальной сети, он будет считать, что запросы пришли с брандмауэра.

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.

Ключ действия SNAT описан в табл. 23.

Таблица 23. Ключ действие SNAT

|  |
| --- |
| Ключ: **--to-source**  Ключ **--to-source** используется для указания адреса, назначаемого пакету. Указывается IP адрес, который будет подставлен в заголовок пакета в качестве исходящего. Если необходимо перераспределять нагрузку между несколькими брандмауэрами, то можно указать диапазон адресов, где начальный и конечный адреса диапазона разделяются дефисом, например: 194.236.50.155-194.236.50.160. Тогда, конкретный IP адрес будет выбираться из диапазона случайным образом для каждого нового потока. Дополнительно можно указать диапазон портов, которые будут использоваться только для нужд **SNAT**. Все исходящие порты будут после этого преобразовываться в заданный диапазон. iptables старается избегать преобразования (перекартирования) портов, однако не всегда это возможно, и тогда производится перекартирование. Если диапазон портов не задан, то исходные порты ниже 512 перекартируются в диапазоне 0-511, порты в диапазоне 512-1023 перекартируются в диапазоне 512-1023, и, наконец порты из диапазона 1024-65535 перекартируются в диапазоне 1024-65535. Порты назначения не подвергаются перекартированию.  Пример: **iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to-source 194.236.50.155-194.236.50.160:1024-32000** |

**Действие TOS**

Действие TOS используется для установки битов в поле Type of Service IP заголовка. Поле TOS содержит 8 бит, которые используются для маршрутизации пакетов. Это поле может обрабатываться различными маршрутизаторами с целью выбора маршрута движения пакета.

Действие TOS имеет ключ, который описан в табл. 24.

Таблица 24. Ключ действия TOS

|  |
| --- |
| Ключ: **--set-tos**  Ключ **--set-tos** определяет числовое значение в десятичном или шестнадцатеричном виде. Поскольку поле TOS является 8-битным, то можно указать число в диапазоне от 0 до 255 (0x00 - 0xFF). Однако, большинство значений этого поля никак не используются. Вполне возможно, что в будущих реализациях TCP/IP числовые значения могут быть изменены, поэтому, во-избежание ошибок, лучше использовать мнемонические обозначения: Minimize-Delay (16 или 0x10), Maximize-Throughput (8 или 0x08), Maximize-Reliability (4 или 0x04), Minimize-Cost (2 или 0x02) или Normal-Service (0 или 0x00). По-умолчанию большинство пакетов имеют признак Normal-Service, или 0. Список мнемоник вы сможете получить, выполнив команду **iptables -j TOS -h**.  Пример: **iptables -t mangle -A PREROUTING -p TCP --dport 22 -j TOS --set-tos 0x10** |

**Действие TTL**

Действие TTL используется для изменения содержимого поля Time To Live в IP заголовке. Один из вариантов применения этого действия - это устанавливать значение поля Time To Live во всех исходящих пакетах в одно и то же значение. Существуют провайдеры, которые очень не любят, когда одним подключением пользуется несколько компьютеров, если мы начинаем устанавливать на все пакеты одно и то же значение TTL, то тем самым мы лишаем провайдера одного из критериев определения того, что подключение к Интернету разделяется несколькими компьютерами. Для примера можно привести число TTL = 64, которое является стандартным для ядра Linux.

Действие TTL можно указывать только в таблице mangle. Для данного действия предусмотрены ключи, описанные в табл. 25.

Таблица 25. Ключи действия TTL

|  |
| --- |
| Ключ: **--ttl-set**  Устанавливает поле TTL в заданное значение. Оптимальным считается значение около 64. Это не слишком много, но и не слишком мало Не задавайте слишком большое значение, это может иметь неприятные последствия для вашей сети. Представьте себе, что пакет зацикливается между двумя неправильно сконфигурированными маршрутизаторами, тогда, при больших значениях TTL, есть риск потерять значительную долю пропускной способности канала.  Пример: **iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-set 64** |
| Ключ: **--ttl-dec**  Уменьшает значение поля TTL на заданное число. Например, пусть входящий пакет имеет значение TTL равное 53 и мы выполняем команду **--ttl-dec 3**, тогда пакет покинет хост с полем TTL равным 49. Сетевой код автоматически уменьшит значение TTL на 1, поэтому, фактически мы получаем 53 - 3 - 1 = 49.  Пример: **iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-dec 1** |
| Ключ: **--ttl-inc**  Увеличивает значение поля TTL на заданное число. Возьмем предыдущий пример, пусть к нам поступает пакет с TTL = 53, тогда, после выполнения команды **--ttl-inc 4**, на выходе с нашего хоста, пакет будет иметь TTL = 56, не забывайте об автоматическом уменьшении поля TTL сетевым кодом ядра, т.е. фактически мы получаем выражение 53 + 4 - 1 = 56. Увеличение поля TTL может использоваться для того, чтобы сделать наш брандмауэр менее заметным для трассировщиков (traceroutes).  Пример: **iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-inc 1** |

**Действие ULOG**

Действие ULOG предоставляет возможность журналирования пакетов в пользовательское пространство. Оно заменяет традиционное действие LOG, базирующееся на системном журнале. При использовании этого действия пакет через сокеты netlink передается специальному демону который может выполнять очень детальное журналирование в различных форматах (обычный текстовый файл, база данных MySQL и пр.) и к тому же поддерживает возможность добавления плагинов для формирования различных выходных форматов и обработки сетевых протоколов.

Ключи действия ULOG описаны в табл. 26.

Таблица 6-25. Ключи действия ULOG

|  |
| --- |
| Ключ: **--ulog-nlgroup**  Ключ **--ulog-nlgroup** сообщает **ULOG** в какую группу netlink должен быть передан пакет. Всего существует 32 группы (от 1 до 32). Если вы желаете передать пакет в 5-ю группу, то можно просто указать **--ulog-nlgroup 5**. По-умолчанию используется 1-я группа.  Пример: **iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-nlgroup 2** |
| Ключ: **--ulog-prefix**  Ключ **--ulog-prefix** имеет тот же смысл, что и аналогичная опция в действии **LOG**. Длина строки префикса не должна превышать 32 символа.  Пример: **iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-prefix "SSH connection attempt: "** |
| Ключ: **--ulog-cprange**  Ключ **--ulog-cprange** определяет, какую долю пакета, в байтах, надо передавать демону **ULOG**. Если указать число 100, как показано в примере, то демону будет передано только 100 байт из пакета, это означает, что демону будет передан заголовок пакета и некоторая часть области данных пакета. Если указать 0, то будет передан весь пакет, независимо от его размера. Значение по умолчанию - 0.  Пример: **iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-cprange 100** |
| Ключ: **--ulog-qthreshold**  Ключ **--ulog-qthreshold** устанавливает величину буфера в области ядра. Например, если задать величину буфера равной 10, как в примере, то ядро будет накапливать журналируемые пакеты во внутреннем буфере и передавать в пользовательское пространство группами по 10 пакетов. По умолчанию размер буфера равен 1 из-за сохранения обратной совместимости с ранними версиями ulogd, которые не могли принимать группы пакетов.  Пример: **iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-qthreshold 10** |

**Утилита iptables-save**

Утилита iptables-save предназначена для сохранения текущего набора правил в файл, который затем может быть использован утилитой iptables-restore. Эта команда имеет два аргумента.

**iptables-save** [-c] [-t *table*]

Первый аргумент -c (допустимо использовать длинный вариант --counters) заставляет iptables-save сохранить значения счетчиков байт и пакетов. Это делает возможным рестарт брандмауэра без потери счетчиков, которые могут использоваться для подсчета статистики. По умолчанию, при запуске без ключа -с, сохранение счетчиков не производится.

С помощью ключа -t (более длинный вариант --table) можно указать имя таблицы для сохранения. Если ключ -t не задан, то сохраняются все таблицы.

**Утилита iptables-restore**

Утилита iptables-restore используется для восстановления (загрузки из файла) набора правил, который ранее был сохранен утилитой iptables-save. Набор правил утилита получает со стандартного ввода и не может загружать его из файла напрямую. Команда имеет следующий синтаксис:

**iptables-restore** [-c] [-n]

Ключ -c (более длинный вариант --counters) заставляет восстанавливать значения счетчиков.

Указание ключа -n (более длинный вариант --noflush) сообщает iptables-restore о том, что правила должны быть добавлены к имеющимся. По умолчанию утилита iptables-restore (без ключа -n) очистит содержимое таблиц и цепочек перед загрузкой нового набора правил.

Для загрузки набора правил утилитой iptables-restore из файла можно использовать несколько вариантов, но наиболее часто используется следующий:

**cat /etc/iptables-save | iptables-restore -c**

В результате выполнения этой команды содержимое файла /etc/iptables-save будет прочитано утилитой cat и перенаправленно на стандартный ввод утилиты iptables-restore.

**Порядок выполнения работы**

1. Изучите теоретический материал.
2. Выполните практическое задание.
3. Ответьте на контрольные вопросы.

**Практическое задание**

1. Сохраните в файл текущие настройки iptables.
2. Запретите доступ к некоторому выбранному сайту.
3. Запретите ответы на ping.
4. Запретите некоторому пользователю отправлять запросы на порт 80.
5. Сохраните полученную конфигурацию брандмауэра в файл и продемонстрируйте работу преподавателю.
6. Восстановите из файла исходные настройки iptables.

**Контрольные вопросы**

1. Опишите процесс обработки пакетов таблицами iptables.

2. Опишите порядок движения транзитных пакетов.

3. Опишите порядок прохождения пакетов для локального приложения.

4. Опишите порядок движения пакетов от локальных процессов.

5. Назначение таблицы mangle.

6. Назначение таблицы nat.

7. Назначение таблицы filter.

8. Опишите работу механизма определения состояний для TCP-соединения.

9. Опишите работу механизма определения состояний для UDP.

10. Опишите работу механизма определения состояний для ICMP.

11. Опишите команды iptables.

12. Общие критерии выделения пакетов.

13. TCP критерии выделения пакетов.

14. UDP критерии выделения пакетов.

15. ICMP критерии выделения пакетов.

16. Критерий limit.

17. Критерии mac, mark, multiport, owner.

18. Критерии state, tos, ttl.

19. Действия ACCEPT, DNAT, DROP, REJECT.

20. Действия LOG, ULOG, MARK, MASQUERADE, REDIRECT.

21. Действия RETURN, SNAT, TOS, TTL.

22. Утилита iptables-save.

23. Утилита iptables-restore.

**Библиографический список**

1. Стахнов А.А. Linux. – СПб.: БХВ-Петербург, 2009.

2. Бэндл Д. Защита и безопасность в сетях Linux. Для профессионалов. – СПб.: Питер, 2002.

3. Андерссон О. Руководство по iptables (Iptables Tutorial 1.1.19)