IPTABLES является утилитой (!), а не непосредственно фаерволом. Роль фаервола в **Linux** играет пакет**Netfilter** (в Linux с версии 2.4).

IPTABLES представляет собой утилиту для конфигурирования **Netfilter**.

Структура IPTABLES

Структуру можно представить так:

iptables -> Tables -> Chains -> Rules

Пакет приходит на **IPTABLES** > далее на *таблицы* (*Tables*) > попадает в *цепочки* (*Chains*) > проходит по правилам (*Rules*).

Фактически, **IPTABLES** состоит из трёх основных частей:

таблицы: фактически, являют собой набор цепочек;

цепочки: набор правил;

правила;

Таблицы IPTABLES

У **IPTABLES** имеется 4 встроенных типа таблиц.

1. Filter Table

Является таблицей по-умолчанию. Если при создании/изменении правила не указана таблица — используется именно `filter`. Используется в основном для фильтрации пакетов. К примеру, тут можно выполнить `DROP, LOG, ACCEPT` или `REJECT` без каких либо сложностей, как в других таблицах. Использует 3 встроенных цепочки:

- `INPUT` chain – входящие пакеты, используется только для пакетов, цель которых — сам сервер, не используется для транзитного (роутинга) трафика
- `OUTPUT` chain – исходящие пакеты, созданные локально и отправленные «за пределы» сервера;
- `FORWARD` chain – пакеты, предназначенные другому сетевому интерфейсу (роут на другие машины сети, например).

2. NAT table

Таблица `nat` используется главным образом для преобразования сетевых адресов (*Network Address Translation*). Через эту таблицу проходит только первый пакет из потока. Преобразования адресов автоматически применяется ко всем последующим пакетам. Это один из факторов, исходя из которых мы не должны осуществлять какую-либо фильтрацию в этой таблице.

- `PREROUTING` chain – преобразование адресов `DNAT` (*Destination Network Address Translation*), фильтрация пакетов здесь допускается только в исключительных случаях;
- `POSTROUTING` chain – выполняется преобразование адресов `SNAT` (*Source Network Address Translation),* фильтрация пакетов здесь крайне нежелательна;
- `OUTPUT` chain – `NAT` для локально сгенерированных пакетов;

3. Mangle table

Таблица `Mangle` предназначена только для внесения изменения в некоторые заголовки пакетов — `TOS`(*Type of Service*), `TTL` (*Time to Live*), `MARK` (особая метка для **IPTABLES** или других служб). **Важно**: в действительности поле `MARK` не изменяется, но в памяти ядра заводится структура, которая сопровождает данный пакет все время его прохождения через машину, так что другие правила и приложения на данной машине (и только на данной машине) могут использовать это поле в своих целях.

Включает в себя такие цепочки:

- `PREROUTING` chain
- `OUTPUT` chain
- `FORWARD` chain
- `INPUT` chain
- `POSTROUTING` chain

4. Raw table

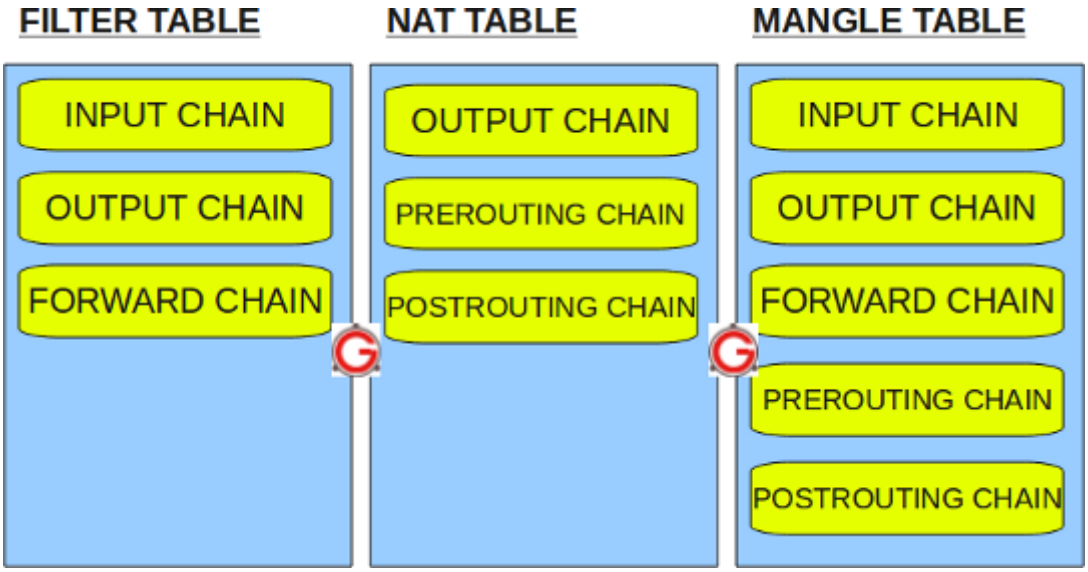
Применяется до передачи пакета механизму определения состояний (*state machine, connection tracking* — система трассировки соединений, при помощи которой реализуется межсетевой экран на сеансовом уровне (*stateful firewall*), позволяет определить, к какому соединению или сеансу принадлежит пакет, анализирует все пакеты кроме тех, которые были помечены `NOTRACK` в таблице raw).

- `PREROUTING` chain
- `OUTPUT` chain

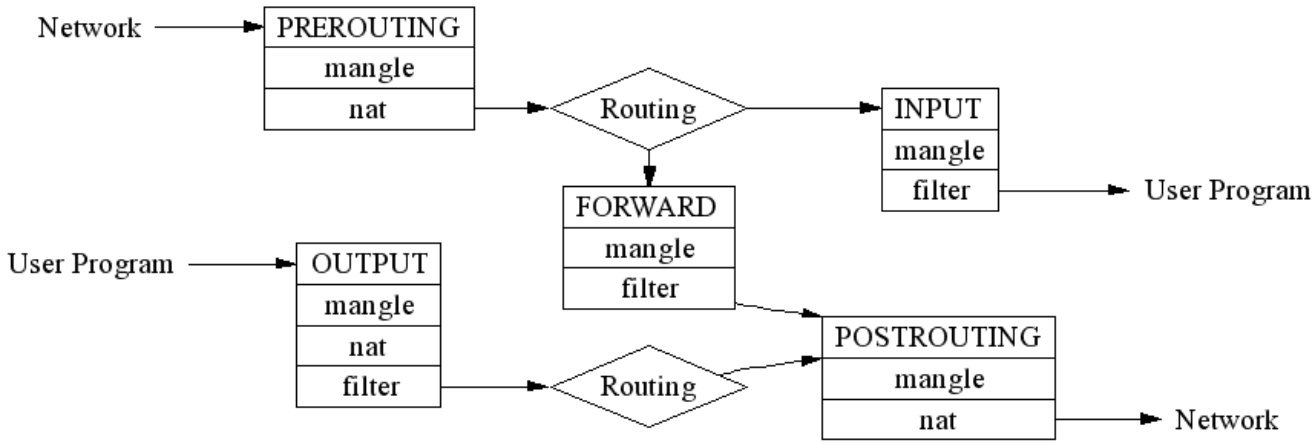
Цепочки IPTABLES

Существует 5 типов стандартных цепочек, встроенных в систему:

- `PREROUTING` — для изначальной обработки входящих пакетов;
- `INPUT` — для входящих пакетов адресованных непосредственно локальному процессу (клиенту или серверу);
- `FORWARD` — для входящих пакетов перенаправленных на выход (заметьте, что перенаправляемые пакеты проходят сначала цепь `PREROUTING`, затем `FORWARD` и `POSTROUTING`);
- `OUTPUT` — для пакетов генерируемых локальными процессами;
- `POSTROUTING` — для окончательной обработки исходящих пакетов.



Схематично путь пакетов через **IPTABLES** хорошо представлен на следующей схеме:



Правила IPTABLES

Правила имеют следующую структуру:

правило > цель > счётчик

Если пакет соответствует *правилу*, к нему применяется *цель*, и он учитывается *счетчиком*. Если *правило*(или *критерий*) не задан — то *цель* применяется ко всем проходящим через цепочку пакетам. Если не указаны ни *цель*, ни *правило* — для правила будет срабатывать только *счётчик* пакетов. Если пакет не попадает под *правило* и *цель* — он передаётся следующему правилу в списке.

Параметры для формирования правил смотрите в посте Linux: IPTABLES — руководство: часть 3 — параметры правил.

Цели (targets) IPTABLES

Правило может содержать одно из следующих целей (или действий):

- `ACCEPT` – принять пакет, и передать следующей цепочке (или приложению, или передать для дальнейшего роутинга);
- `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; действие `DNAT` может выполняться только в цепочках `PREROUTING` и `OUTPUT` таблицы `nat`, и во вложенных под-цепочках; важно запомнить, что вложенные подцепочки, реализующие `DNAT` не должны вызываться из других цепочек, кроме `PREROUTING` и `OUTPUT`;
- `DROP` – просто «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`; 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;
- `LOG` — действие, которое служит для журналирования отдельных пакетов и событий; в журнал могут заноситься заголовки **IP** пакетов и другая полезная информация; информация из журнала может быть затем прочитана с помощью `dmesg` или `syslogd`, либо с помощью других программ; обратите ваше внимание так же на цель `ULOG`, которое позволяет выполнять запись информации не в системный журнал, а в базу данных **MySQL** и т.п.;
- `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;
- `MASQUERADE` — в основе своей 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 у вас имеется

динамическое подключение, то нужно использовать маскарадинг, если же у вас статическое **IP** подключение, то лучше будет использование `SNAT`;

- `MIRROR` — может использоваться вами 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 и пакет отправляется в сеть; допускается использовать только в цепочках `INPUT, FORWARD` и `PREROUTING`, и в цепочках, вызываемых из этих трех;
- `QUEUE` – ставит пакет в очередь на обработку пользовательскому процессу; оно может быть использовано для нужд учета, проксирования или дополнительной фильтрации пакетов;
- `REDIRECT` — выполняет перенаправление пакетов и потоков на другой порт той же самой машины; можно пакеты, поступающие на **HTTP** порт перенаправить на порт HTTP-proxy; удобен для выполнения «прозрачного» проксирования (*transparent proxy*), когда машины в локальной сети даже не подозревают о существовании прокси; может использоваться только в цепочках `PREROUTING` и `OUTPUT` таблицы `nat`;
- `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;
- `RETURN` — прекращает движение пакета по текущей цепочке правил и производит возврат следующему правилу в вызывающей (предыдущей) цепочке, 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`;
- `SNAT` — используется для преобразования сетевых адресов (*Source Network Address Translation*), т.е. изменение исходящего **IP** адреса в **IP** в заголовке пакета; `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;
- `TOS` — используется для установки битов в поле *Type of Service* **IP**заголовка; поле `TOS` содержит 8 бит, которые используются для маршрутизации пакетов; важно помнить, что данное поле может обрабатываться различными маршрутизаторами с целью выбора маршрута движения пакета; в отличие от `MARK`, сохраняет свое значение при движении по сети, а поэтому может использоваться для маршрутизации пакета; лучше всего использовать это поле для своих нужд только в пределах **WAN** или **LAN**;
- `TTL` — используется для изменения содержимого поля *Time To Live* в **IP**заголовоке;
- `ULOG` — система журналирования пакетов, которая 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.), млжет формировать отчёты в **CSV, XML, Netfilter's LOG, Netfilter's conntrack**; подробнее смотрите на домашней странице проекта;

Параметры для формирования действий смотрите в посте Linux: IPTABLES — руководство: часть 4 — цели для правил.

Критерий так же может использовать состояние пакета для принятия решений:

- `NEW` — пакет открывает новый сеанс, например — пакет `TCP` с флагом `SYN`;
- `ESTABLISHED` — пакет является частью уже существующего сеанса;
- `RELATED` — пакет открывает новый сеанс, связанный с уже открытым сеансом, например, во время сеанса пассивного **FTP**, клиент подсоединяется к порту 21 сервера, сервер сообщает клиенту номер второго, случайно выбранного порта, после чего клиент подсоединяется ко второму порту для передачи файлов; в этом случае второй сеанс (передача файлов по второму порту) связан с уже существующим сеансом (изначальное подсоединение к порту 21);
- `INVALID` — все прочие пакеты.

Управление IPTABLES

Просмотр правил

Просмотр текущих правил в таблице `filter`:

```
# iptables -t filter --list
```

Тоже самое — для остальных:

```
# iptables -t raw --list
```

```
# iptables -t nat --list
```

```
# iptables -t mangle --list
```

Или с опцией `-L`:

```
# iptables -t filter -L
```

Колонки тут:

- `num` – номер правила текущей цепочке (см. дальше);
- `target` – действие;
- `prot` – протокол — **TCP, UDP, ICMP** и т.д.;
- `opt` – специальные опции для этого правила;
- `source` – исходный **IP**-адрес пакетов;
- `destination` – **IP**-адрес назначения пакетов.

Далее я буду придерживаться коротких опций, вида `-L`, вместо длинных вида `--list`.

Просмотр списка правил и счётчиков:

```
# iptables -L -v
```

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
```

```
pkts bytes target     prot opt in     out     source               destination
```

```
3313  199K DROP       all  --  any    any     37.9.53.253          anywhere
```

Кроме того, `-v` (`verbose`) выводит:

— имена интерфейсов для правил;
— заголовки `TOS`;
— дополнительные опции правила.

`verbose` так же можно использовать при создании-удалении правил и т.д.

Не переводить **IP** в имена хостов (**FQDN**):

```
# iptables -L -n
```

Выводить номер правила:

```
# iptables -L --line-numbers
```

Вывести правила только для `INPUT`, с отображением счётчиков, **IP**вместо имён хостов, и номерами правил:

```
# iptables -L INPUT -v -n --line-numbers
```

Просмотреть только список правил:

```
# iptables -S
```

Пример:

```
# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
```

Управление chains

Создать цепочку:

```
# iptables -N new_chain
# iptables -L new_chain
Chain new_chain (0 references)
target     prot opt source               destination
```

Переименовать цепочку:

```
# iptables -E new_chain old_chain
# iptables -L old_chain
Chain old_chain (0 references)
target     prot opt source               destination
```

Удалить цепочку:

```
# iptables -X old_chain
# iptables -L old_chain
iptables: No chain/target/match by that name.
```

Управление правилами

Добавить правило:

```
<em># iptables -A chain</em>
```

Выполнит *append* к указанной *chain*, добавляя правило в конец списка. Пример:

```
# iptables -A INPUT -s 192.168.1.102 -j ACCEPT
# iptables -L INPUT
...
ACCEPT     all  --  192.168.1.102        anywhere
```

Вставить правило:

```
# iptables -I <em>chain</em> <em>rulenum</em>
```

Добавит правило в цепочку *chain* под номером *rulenum* (если такой номер уже есть — то на его место, а существующее — сдвинется «вниз»). Пример:

```
# iptables -I INPUT 1 -s 192.168.1.102 -j ACCEPT
# iptables -L --line-numbers
```

```
2      Chain INPUT (policy ACCEPT)
3  num  target     prot opt source               destination
4  1    ACCEPT     all  --  192.168.1.102        anywhere
```

Заменить правило:

```
1      # iptables -R chain rulenum
```

Добавит правило в цепочку *chain* под номером *rulenum* (если такой номер уже есть — то на его место, а существующее -будет удалено). Пример:

```
1   # iptables -R INPUT 1 -s 192.168.1.102 -j DROP
1      # iptables -L --line-numbers
2      Chain INPUT (policy ACCEPT)
3  num  target     prot opt source               destination
4  1    DROP       all  --  192.168.1.102        anywhere
```

Удалить правило — вариант 1:

```
1   # iptables -D <em>chain</em> <em>rulenum</em>
```

Удалит правило в цепочке *chain* с номером *rulenum*. Пример:

```
1        # iptables -D INPUT 1
1      # iptables -L --line-numbers
2      Chain INPUT (policy ACCEPT)
3  num  target     prot opt source               destination
```

Удалить правило — вариант 2:

```
1   # iptables -D <em>chain</em> <em>rule</em>
```

Удалит правило, которое соответствует *rule*. Пример:

```
1   # iptables -I INPUT 1 -s 192.168.1.102 -j DROP
1      # iptables -L --line-numbers
2      Chain INPUT (policy ACCEPT)
3  num  target     prot opt source               destination
4  1    DROP       all  --  192.168.1.102        anywhere
1   # iptables -D INPUT -s 192.168.1.102 -j DROP
1      # iptables -L --line-numbers
2      Chain INPUT (policy ACCEPT)
3  num  target     prot opt source               destination
4  1    ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
```

Другие опции

Изменить правило по-умолчанию:

```
1      # iptables -P chain target
```

Например, для `INPUT` изменить `ACCEPT` на `DROP`:

```
1           # iptables -L
2      Chain INPUT (policy ACCEPT)
1       # iptables -P INPUT DROP
1           # iptables -L
```

```
2 Chain INPUT (policy DROP)
```

(будьте осторожны с этой опцией — можете лишиться доступа к серверу)

Сбросить (удалить) все правила во всех цепочка:

```
1 # iptables -F
```

Удалить правила только для цепочки `INPUT`:

```
1 # iptables -F INPUT
```

(будьте осторожны с этой опцией — можете лишиться доступа к серверу)

Обнулить все счётчики во всех правилах:

```
1 # iptables -Z
```

Пример:

```
1 # iptables -L -v
2 Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
3 pkts bytes target prot opt in out source destination
4 25 2352 ACCEPT all -- any any anywhere anywhere state RELATED,ESTABLISHED
1 # iptables -Z
1 # iptables -L -v
2 Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
3 pkts bytes target prot opt in out source destination
4 5 356 ACCEPT all -- any any anywhere anywhere state RELATED,ESTABLISHED
```

Сохранение и восстановление правил

В **CentOS** правила хранятся в файле `/etc/sysconfig/iptables`:

```
01 # head /etc/sysconfig/iptables
02 # Generated by iptables-save v1.4.7 on Thu Oct 16 18:56:56 2014
03 *filter
04 :INPUT ACCEPT [0:0]
05 :FORWARD ACCEPT [0:0]
06 :OUTPUT ACCEPT [66:27636]
07 -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
08 -A INPUT -p icmp -j ACCEPT
09 -A INPUT -i lo -j ACCEPT
10 -A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
11 -A INPUT -j REJECT --reject-with icmp-host-prohibited
```

Для сохранения правил — выполните:

```
1 # service iptables save
2 iptables: Saving firewall rules to /etc/sysconfig/iptables:[ OK ]
```

Что бы сохранить правила в отдельный файл — используйте:

```
1 # iptables-save > /root/iptables_bkp
```

Что бы восстановить из него:

```
1 # iptables-restore < /root/iptables_bkp
```

С помощью опции `-c` команды `iptables-save` можно так же сохранить значения счётчиков, а с помощью `-t <em>tablename</em>` — сохранить определённую таблицу.

Основные параметры правил

`-p, --protocol` — правило сработает для протокола; имя протокола должно быть указано явно, в виде**TCP, UDP** или **ICMP**, и должно присутствовать в файле `/etc/protocols`;

Пример:

```
1       iptables -A INPUT -p tcp
```

`-s, --src, --source` — фильтр по исходному (source — источник) адресу; можно указать как один адрес — так и подсеть; при установленном флаге ! перед указанием фильтра — значение будет инвертировано (обратное значение: —source адрес *является* > —*source ! адрес не является*);

Примеры:

```
1     iptables -A INPUT -s 192.168.1.1
2     iptables -A INPUT -s ! 192.168.1.1
3
4
5   <code>-d, --dst, --destination</code> - фильтр по адресу назначения (<em>destination</em> - назначение); осталь
6
7                 Пример:
1     iptables -A INPUT -d 192.168.1.1
```

`-i, --in-interface` — правило для входящего трафика на указанном интерфейсе; используется только в цепочках `INPUT, FORWARD` и `PREROUTING`; допустимо использовать маски, например `+` — для всех интерфейсов, или `eth+` — для всех интерфейсов `eth0`, `eth1` и т.д.; допустимо использование инвертации;

Примеры:

```
1       iptables -A INPUT -i eth0
2       iptables -A INPUT -i eth+
3      iptables -A INPUT -i ! eth0
```

`-o, --out-interface` — правило для исходящего трафика на указанном интерфейсе; остальные правила аналогичны `--in-interface`;

Параметры правил TCP

`--sport, --source-port` — правило для порта-источника; если не указан порт — сработает для всех портов; можно указывать имя сервиса, либо порт в виде цифр; при указании имени сервиса — оно должно присутствовать в файле `/etc/services`; использование портов вместо имён сервисов уменьшит нагрузку и ускорит обработку; допустимо указать диапазон портов, например 80:1024 — для всех портов от 80 до 1024 включительно, либо :80 — для всех портов от 0 до 80 включительно; допустимо инвертирование значения (!);

Примеры:

```
1     iptables -A INPUT -p tcp --sport 22
2     iptables -A INPUT -p tcp --sport ssh
3    iptables -A INPUT -p tcp --sport 80:1024
4     iptables -A INPUT -p tcp --sport :80
5     iptables -A INPUT -p tcp --sport ! 80
```

`--dport, --destination-port` — правило для порта назначения; остальные правила аналогичны `--source-port`;

Пример:

```
1    iptables -A INPUT -p tcp --dport 22
```

`--tcp-flags` — правило сработает при наличии **TCP**-флагов;

Пример:

```
1    iptables -p tcp --tcp-flags SYN,FIN,ACK SYN
```

Параметры правил UDP

`--sport, --source-port` — аналогично `--source-port` **TCP**;

`--dport, --destination-port` — аналогично `--destination-port` **TCP**;

Параметры правил ICMP

`--icmp-type` — проверяет соответствие типа **ICMP**-пакета; список смотрите тут>>>,

```
1    iptables -A INPUT -p icmp --icmp-type 8
```

Дополнительные модули и параметры

Перечислю только наиболее интересные и используемые.

Модули подключаются с помощью опции `-m`.

Addrtype — фильтрация на основе типа адреса.

`--src-type` — фильтр на основе типа адреса источника; можно указать один или несколько типов, разделив их запятой; допустимо инвертирование значения (!);

Примеры:

```
1    iptables -A INPUT -m addrtype --src-type UNICAST
2    iptables -A INPUT -m addrtype --src-type ! UNICAST
3    iptables -A INPUT -m addrtype --src-type MULTICAST, BROADCAST
```

—dst-type — то же самое, но для адреса получателя пакета;

Модуль **Comment** — добавление своего комментария к фильтру.

`--comment` — добавление своего комментария к фильтру; максимальное значение — 256 символов;

Пример:

```
1    iptables -A INPUT -m comment --comment "This is comment"
```

Модуль **Connmark** — действие на основе метки, установленной действием `CONNMARK` или `MARK --set-mark` (*действия* смотрите в следующей части).

`--mark` — фильтр на основе наличия указанной метки;

Пример:

```
1    iptables -A INPUT -m connmark --mark 12 -j ACCEPT
```

Модуль **IP range** — расширенный аналог `--source` и `--destination`. Позволяет указывать даиапазон**IP**.

`--src-range` — диапазон **IP** источников; в отличии от `--source` и `--destination` позволяет указать диапазон между двумя конкретными адресами, а не подсетью; допустимо инвертирование (!);

Пример:

```
1 iptables -A INPUT -p tcp -m iprange --src-range 192.168.1.13-192.168.2.19
```

`--dst-range` — аналогично `--src-range`, но для адресов назначения;

Модуль **Length** — фильтрация по длине пакета.

`--length` — допустимо указание диапазона размеров, например 1400:1500; допустимо инвертирование значения;

Примеры:

```
1 iptables -A INPUT -p tcp -m length --length 1400
2
3 iptables -A INPUT -p tcp -m length --length 1400:1500
```

Модуль **Limit** — ограничение количества пакетов в единицу времени. Хорошо описан тут>>>.

`--limit` — средняя скорость заполнения «счётчика»; это же значение учитывается для уменьшения счётчика в `--limit-burst`; указывается в виде *число пакетов/время*, где время указывается в виде /second /minute /hour /day; по-умолчанию 3 пакета в час, или 3/hour; использование инвертирования невозможно;

Пример:

```
1 iptables -A INPUT -m limit --limit 3/hour
```

`--limit-burst` — максимальное значение счётчика для срабатывания правила; каждый раз, когда значение `--limit` превышается — счётчик увеличивается на единицу, пока не достигнет `--limit-burst`, после чего срабатывает правило, в котором используется `--limit-burst`;

Пример:

```
1 iptables -A INPUT -m limit --limit-burst 5
```

Модуль **Mac** — фильтрация по **MAC** (*Ethernet Media Access Control*).

`--mac-source` — **MAC**-адрес источника; допустимо использование инвертирования (!); используется только в цеочках P`REROUTING, FORWARD` и `INPUT`;

Пример:

```
1 iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01
```

Модуль **Mark** — фильтр на основе метки (действия `CONNMARK` и `MARK` см. в следующем посте).

`--mark` — метка, установленная `MARK`;

Пример:

```
1 iptables -t mangle -A INPUT -m mark --mark 1
```

Модуль **Multiport** — фильтр по нескольким портам. В отличии от `--sport` допускает использование нескольких независимых портов, а не диапазона.

`--source-port` — список портов источника; максимум 15 значений, разделённых запятыми; должен использоваться только с `-p tcp` или `-p udp`;

Пример:

```
1 iptables -A INPUT -p tcp -m multiport --source-port 22,53,80,110
```

`--destination-port` — аналогично `--source-port`, но для портов назначения;

Пример:

```
1 iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110
```

`--port` — аналогично для обоих значений `--source-port` и `--destination-port`, для обоих направлений; сработает только в случае, если порт источника и порт назначения идентичны;

Пример:

```
1 iptables -A INPUT -p tcp -m multiport --port 22,53,80,110
```

Модуль **Owner** — фильтры на основе владельца пакета.

`--cmd-owner` — фильтр по имени сервиса, который сгенерировал пакет; допустимо использование инвертации с помощью(!);

Пример:

```
1 iptables -A OUTPUT -m owner --cmd-owner httpd
```

`--uid-owner` — фильтр по **UID** (*User ID*) пользователя, который сгенерировал пакет;

Пример:

```
1 iptables -A OUTPUT -m owner --uid-owner 500
```

`--gid-owner` — аналогично, но для **GID** (*Group ID*);

Пример:

```
1 iptables -A OUTPUT -m owner --gid-owner 0
```

`--pid-owner` — аналогично, но для **PID** (*Process ID*);

Пример:

```
1 iptables -A OUTPUT -m owner --pid-owner 78
```

`--sid-owner` — аналогично, но для **SID** (`Service ID`);

Пример:

```
1 iptables -A OUTPUT -m owner --sid-owner 100
```

Модуль **Packet type** — фильтры на основе типа адресации пакета.

`--pkt-type` — допустимо использование *unicast , broadcast*или *multicast* типов; допустим использование инвертации;

Пример:

```
1 iptables -A OUTPUT -m pkttype --pkt-type unicast
```

Модуль **State** — фильтры на основе состояния соединения.

`--state` — указывает состояние пакета в соединении; могут быть использованы четыре тип состояния — `INVALID, ESTABLISHED, NEW` и `RELATED`. `INVALID` означает, что пакет ассоциируется с неизвестным стримом или соединением и может содержать некорретные данные или заголовки. `ESTABLISHED` — пакет принадлежит к уже установленному в обоих направлениях соединению и полностью валиден. `NEW` — пакет устанавливает или будет устанавливать новое соединение, или пакет соединения,у которого ещё не было передачи данных в обоих

направлениях; `RELATED` — пакет, который устанавливает новое соединение, которое ассциировано с уже установленным соединением (пример — пассивный режим **FTP**, или ошибка**ICMP**, связанная с каким-то **TCP** или **UDP** соединением)

Содержание:

Действие над попавшим под правило пакетом выполняется с помощью опий `-j (--jump)` или `-g (--goto`).

`-j, --jump <i>цель</i>` — *цель* (или *действие*) для пакетов, попадающих по действие правила; целью может быть цепочка, определённая пользователем (отличная от цепочки правила), одна из встроенных целей, определяющая окончательное действие над пакетом; если опция не задана в правиле (и ключ `-g` не использован), то сработает только счётчик количества правила;

`-g, --goto <i>цепочка</i>` — продолжить обработку в цепочке, определённой пользователем; в отличие от опции —jump, после действия RETURN из вызванной цепочки, применение правил будет продолжено не в текущей цепочке, а в той цепочке, которая вызвала текущую через `--jump`.

Для использования `-j <em>chain</em>` или `-g <em>chain</em>` — цепочка должна быть уже создана.

Пример:

```
1    iptables -A INPUT -p tcp -j tcp_packet
```

Цели IPTABLES

ACCEPT – принять пакет, и передать следующей цепочке (или приложению, или передать для дальнейшего роутинга);

У данного действия нет дополнительных опций. После выполнения **ACCEPT** пакет не будет более отслеживаться никаким правилом этой цепочки (но может попасть под правило в другой, например в `POSTROUTING`).

Пример:

```
1    iptables -A INPUT -s SOME_IP_HERE -j ACCEPT
```

CONNMARK — схожа с **MARK**. Позволяет установить метку на пакеты одной сессии или соединения. Можно использовать в любой цепочке, но при этом учтите, что в таблице `nat` проверяется только первый пакет, а потому метки на всех пакетах будут бесполезны.

`--set-mark` — устанавливает метку на паккет; значение может быть *long int* (от 0 до 4294967295l); можно установить метку на каждый бит пакета, например `--set-mark 12/8`;

Пример:

```
1 iptables -t nat -A PREROUTING -p tcp --dport 80 -j CONNMARK --set-mark 4
```

`--save-mark` — используется для сохранения метки пакета на всё соединение; например, если вы установили метку на пакет с помощью цели **MARK** — с помощью `--save-mark` вы можете переместить её на всё подключение целиком;

Пример:

```
1 iptables -t mangle -A PREROUTING --dport 80 -j CONNMARK --save-mark
```

`--restore-mark` — восстановить метку пакета из метки всего соединения, заданного **CONNMARK**; допустимо использование только в таблице `mangle` ;

Пример:

```
1 iptables -t mangle -A PREROUTING --dport 80 -j CONNMARK --restore-mark
```

`--mask` — используется вместе с `--save-mark` и `--restore-mark`;

Пример:

```
1 iptables -t mangle -A PREROUTING --dport 80 -j CONNMARK --restore-mark --mask 12
```

DNAT — используется для *Destination Network Address Translation*, перезаписывает *Destination IP address*пакета. Допустимо использование только в цепочках `PREROUTING` и `OUTPUT` таблицы `nat`.

`--to-destination` — указывает механизму **DNAT** какой **IP** задать в **IP**-заголовке пакета, и куда его пересылать; в примере ниже все пакеты на адрес 15.45.23.67 будут перенаправляться на диапазон локальных адресов 192.168.1.1 — 192.168.1.10; можно указать порт назначения, например 192.168.1.1:80, или диапазон портов — 192.168.1.1:80-100; обязательно указание протокола (`-p tcp` или `-p udp`):

Пример:

```
1 iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.1
```

Больше информации тут>>>.

DROP — просто «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**. 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.

LOG — действие, которое служит для журналирования отдельных пакетов и событий. В журнал могут заноситься заголовки **IP**пакетов и другая полезная информация. Информация из журнала может быть затем прочитана с помощью `dmesg` или `syslogd`, либо с помощью других программ. Обратите ваше внимание так же на цель **ULOG**, которое позволяет выполнять запись информации не в системный журнал, а в базу данных **MySQL** и т.п.

`--log-level` — указание log-level при логировании события;

Пример:

```
1 iptables -A FORWARD -p tcp -j LOG --log-level debug
```

`--log-prefix` — добавляет префикс к сообщениям в лог-файле; допустимо максимум 29 символов,включая пробелы и другие специальные символы;

Пример:

```
iptables -A INPUT -p tcp -j LOG --log-prefix "INPUT packets"
```

`--log-tcp-sequence` — записывать в лог номер *TCP Sequence*;

Пример:

```
iptables -A INPUT -p tcp -j LOG --log-tcp-sequence
```

`--log-tcp-options` — записывать влог различные опции из заголовка **TCP**-пакета;

Пример:

```
iptables -A FORWARD -p tcp -j LOG --log-tcp-options
```

MARK — устанавливает метку **Netfilter**, которая ассоциирована с определённым пакетом. Допустимо использование только в таблице `mangle`. Учтите — «метка» устанавливается не на сам пакет, а является неким значением для ядра, которое ассоциируется с пакетом. Т.е., если вы ожидаете увидеть данную метку на это пакете на другом хосте — у вас это не выйдет. Для таких целей лучше использовать **TOS** (в данном посте не описано, смотрите тут>>>), который устанавливает значение в **IP**-заголовок пакета.

`--set-mark` — устанавливается в виде int (целого числа), например для дальнейшей расширенной маршрутизации;

Пример:

```
iptables -t mangle -A PREROUTING -p tcp --dport 22 -j MARK --set-mark 2
```

MASQUERADE — в целом, сход по назначению с целью **SNAT**, но без ключа `--to-source`, так как в основном используется при динамических соединения — **DHCP**, **DialUp**, когда **IP** присваивается динамически и получает информацию об **IP** непосредственно с сетевого интерфейса. Использование **MASQUERADE** допустимо приналичии постоянного **IP**, но использует больше системных ресурсов, потому в таком случае предпочтительнее использование **SNAT**. Допускается использование только в цепочке `POSTROUTING`таблицы nat.

`--to-ports` — может быть использована для указания исходных порта или портов для исходящих пакетов; обязательно указание протокола (`p- tc`p или `-p udp`);

Пример:

```
iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE --to-ports 1024-31000
```

NOTRACK — установка метки **NOTRACK** на пакеты. Не имете дополнительных опций. Подробнее смотрите тут>>>.

REDIRECT — выполняет перенаправление пакетов и потоков на другой порт той же самой машины. Можно пакеты, поступающие на **HTTP** порт, перенаправить на порт **HTTP**-proxy. Удобен для выполнения «прозрачного» проксирования (*transparent proxy*), когда машины в локальной сети даже не подозревают о существовании прокси. Может использоваться только в цепочках `PREROUTING` и `OUTPUT` таблицы `nat.`

`--to-ports` — указывает порт или диапазон портов назначения; что бы указать диапазон — укажите их через тире; обязательно указание протокола (`-p tcp` или `-p udp`);

Пример:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080
```

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080-9090
```

REJECT — используется, как правило, в тех же самых ситуациях, что и **DROP**, но в отличие от **DROP**, цель **REJECT** выдает сообщение об ошибке на хост, передавший пакет. Допустимо использование в цепочках `INPUT, FORWARD` и `OUTPUT`, и их под-цепочках.

`--reject-with` — выполняет **REJECT** с указанным ответом; сначала будет выполнен **REJECT**, отправлен ответ, после чего — **DROP**пакета; допустимые значения:`icmp-net-unreachable, icmp-host-unreachable, icmp-port-unreachable, icmp-proto-unreacha`

Пример:

```
1 iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset
```

RETURN — прекращает премещение пакета далее по правилам в текущей цепочке. Если это вложенная цепочка другой, вышестоящей, цепочки — пакет будет перемещён в неё, если это главная цепочка — он будет обработан согласно действия по-умолчанию для данной цепочки. Не имеет дополнительных опций.

SNAT — используется для преобразования сетевых адресов (*Source Network Address Translation*), т.е. изменение исходящего **IP**адреса в **IP** в заголовке пакета. **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.

`--to-source` — используется для указания исходного адреса пакета; допустимо указать диапазон адресов, разделённые тире — 194.236.50.155-194.236.50.160, в таком случае конкретный **IP**адрес будет выбираться случайным образом для каждого нового потока;

Пример:

```
1 iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to-source 194.236.50.155-194.236.50.160:1024-32000
```

ULOG — лучше ознакомится с информацией на домашней странице проекта.