Основы iptables для начинающих. Часть 2. Таблица filter

Записки IT специалиста

Технический блог специалистов ООО"Интерфейс"

- Главная
- Основы iptables для начинающих. Часть 2. Таблица filter

Продолжение цикла статей о брандмауэре **iptables** начнем с изучения таблицы **filter**, как наиболее простой и часто используемой. Ее назначение ясно уже из названия и именно в ней выполняются основные действия по фильтрации трафика и именно с этими действиями у многих связано само понятие брандмауэра. Но несмотря на простоту, именно настройки этой таблицы отвечают за сетевую защиту узла или целой сети и поэтому к ее изучению следует подойти предельно внимательно.

Итак - таблица **filter**, это одна из пяти таблиц **iptables**, но при этом имеющая определенные отличия. Ее смело можно назвать таблицей по умолчанию, так как если мы не указываем таблицу, то подразумевается, что мы имеем дело с **filter**. Данная таблица содержит три цепочки: **INPUT**, **OUTPUT** и **FORWARD** - для входящего, исходящего и транзитного трафика узла. Более подробно о таблицах, цепочках и порядке их прохождения вы можете узнать из предыдущего материала цикла, здесь мы не будем уделять этому повышенного внимания.

Как мы должны помнить, таблицы содержат цепочки, цепочки - правила, а сами правила состоят из критериев, действия и счетчика. Начнем с **действий**. Основных действий в таблице **filter** три, все они терминальные:

- **ACCEPT** - пропускает пакет, прохождение пакета по цепочке прекращается
- **REJECT** - блокирует пакет и сообщает источнику об отказе
- **DROP** - блокирует пакет, не сообщая источнику об отказе

Два последних действия заслуживаю особого пояснения, так как многие часто используют просто **DROP**. Для чего нужно сообщать источнику об отказе? Чтобы он понимал, что запрашиваемый узел недоступен и прекращал попытки соединения с

ним. Если этого не сделать, то хост будет пытаться соединиться с узлом пока не истечет таймаут соединения. Скажем, если мы попробуем обратиться к заблокированному веб-узлу через браузер, то получив сообщение об отказе он практически сразу выведет сообщение, что узел недоступен, в противном случае будет достаточно долго пытаться установить соединение, что не добавит пользователю ясности и понимания происходящего.

Поэтому для внутренних сетей всегда следует использовать **REJECT**, а для внешних - **DROP**, так как это позволяет понизить информативность для злоумышленников, не передавая в их распоряжение лишней информации о работе вашей сети.

С действиями ясно, перейдем к критериям. Критерии - это логическое выражение которому должен соответствовать пакет, если критериев несколько, то они объединяются с помощью логического И. Также перед критерием может стоять знак отрицания - **!**, в этом случае критерий объединяется при помощи логического И-НЕ.

Критерии делятся на **универсальные**, которые могут быть применены к любым пакетам, **специфичные для протоколов** и **требующие подключения внешних модулей**.

Универсальные критерии:

- **-i, --in-interface** *имя_интерфейса* - определяет входящий сетевой интерфейс, если имя интерфейса указано как ppp+, то подразумеваются все интерфейсы, начинающиеся на указанное имя.
- **-o, --out-interface** *имя_интерфейса* - определяет исходящий сетевой интерфейс, по синтаксису аналогичен критерию **-i**.
- **-s, --src, --source** *адрес[/маска] ,адрес[/маска]...* - определяет адрес отправителя пакета, может быть указана в виде IP-адреса, подсети (в формате адрес/маска), в качестве имени из /etc/hosts или DNS-имени. В последнем случае имя будет разрешено в адрес в момент добавления правила. На практике не рекомендуется использовать эту возможность, так как на момент применения правил служба DNS может оказаться недоступной. Начиная с версии 1.4.6 iptables позволяет указывать несколько значений адресов источников, разделяя их запятыми.
- **-d, --dst, --destination** *адрес[/маска] ,адрес[/маска]...* - определяет адрес получателя пакета, синтаксис тот же.
- **-p, --protocol** *протокол* - определяет **протокол транспортного уровня**, можно указать номер либо наименование протокола в том виде, в котором они приведены в **/etc/protocols**.

Рассмотрим практические примеры. Начнем с самых простых. Разрешим весь трафик с интерфейса ens33 и запретим с ens34:

```
iptables -A INPUT -i ens33 -j ACCEPT
iptables -A INPUT -i ens34 -j DROP
```

Команда **iptables -A** добавляет новое правило в конец указанной цепочки. Обратите внимание, что критерий **-i** можно использовать только в цепочках **INPUT** и **FORWARD**, а **-o** - в **FORWARD** и **OUTPUT**.

Немного усложним задачу:

```
iptables -A INPUT -i ens33 -s 192.168.0.0/24 -j ACCEPT
iptables -A INPUT -i ens33 -s 192.168.2.51 -j DROP
iptables -A FORWARD -i ens33 -d 192.168.1.0/24 -j ACCEPT
iptables -A FORWARD -o ens34 -s 192.168.1.0/24 -d 8.8.8.8 -j REJECT
```

Первое правило разрешает доступ к хосту через интерфейс **ens33** любым пакетам из сети **192.168.0.0/24**, второе запрещает доступ от узла **192.168.2.51**. Третье разрешает транзитный трафик, который пришел на интерфейс **ens33** и предназначен сети **192.168.1.0/24**. Последнее запретит трафик из сети **192.168.1.0/24** к узлу **8.8.8.8** через интерфейс **ens34** и уведомит об отказе источник.

Таким образом, комбинируя критерии можно строить достаточно сложные правила, позволяющие достаточно гибко управлять трафиком. Отдельно коснемся возможности указывать в критериях **-s** и **-d** сразу нескольких IP-адресов. Указанные таким образом адреса объединяются при помощи логического ИЛИ, т.е. запись с несколькими адресами будет эквивалентна нескольким записям с одним адресом. Например:

```
iptables -A INPUT -i ens33 -s 192.168.0.0/24,  192.168.1.0/24,  192.168.2.0/24 -j
ACCEPT
```

Эквивалентна набору записей:

```
iptables -A INPUT -i ens33 -s 192.168.0.0/24 -j ACCEPT
iptables -A INPUT -i ens33 -s 192.168.1.0/24 -j ACCEPT
iptables -A INPUT -i ens33 -s 192.168.2.0/24 -j ACCEPT
```

При использовании сразу нескольких адресов в обоих критериях следует проявлять разумную осторожность, так как такая запись будет являться набором записей со всеми возможными комбинациями указанных адресов.

Скажем:

```
iptables -A FORWARD -o ens34 -s 192.168.1.0/24, 192.168.2.0/24  -d 1.1.1.1,
8.8.8.8 -j REJECT
```

Фактически является набором записей:

```
iptables -A FORWARD -o ens34 -s 192.168.1.0/24  -d 1.1.1.1 -j REJECT
iptables -A FORWARD -o ens34 -s 192.168.1.0/24  -d 8.8.8.8 -j REJECT
iptables -A FORWARD -o ens34 -s 192.168.2.0/24  -d 1.1.1.1 -j REJECT
iptables -A FORWARD -o ens34 -s 192.168.2.0/24  -d 8.8.8.8 -j REJECT
```

В ряде случаев, как в данном примере, это удобно. В некоторых иных может приводить к появлению неожиданных комбинаций, которые не предполагались изначально и могут либо нарушить работу сети, либо создать уязвимые участки, предоставляя доступ там, где это не нужно.

Все примеры, которые мы рассмотрели выше, являются универсальными и применяются к любому транспортному протоколу. Для указания протокола используется критерий **-p**, например:

```
iptables -A INPUT -i ens33 -p tcp -j ACCEPT
```

Также это можно записать как:

```
iptables -A INPUT -i ens33 -p 6 -j ACCEPT
```

Используя вместо **tcp** номер протокола - **6**.

Но само по себе указание протокола обычно не имеет большого значения без использования **специфичных для протоколов критериев**. Разберем наиболее часто используемые из них:

--sport, --source-port *порт[:порт]* - указывает порт или диапазон портов источника.

--dport, --destination-port *порт[:порт]* - аналогично, но для порта или диапазона портов назначения.

Например:

```
iptables -A FORWARD -o ens34 -p udp --dport 53 -s 192.168.2.0/24 -d 8.8.8.8 -j
REJECT
```

Мы уточнили уже существующее правило, которое теперь блокирует не все соединения к узлу **8.8.8.8** из сети **192.168.2.0/24**, а только DNS-запросы (порт 53) по протоколу UDP.

А вот такое правило:

```
iptables -A INPUT -s 192.168.2.0/24 -p tcp --dport 62000:62200 -j ACCEPT
```

Разрешит любые входящие подключения к диапазону портов **62000 - 62200** по протоколу **TCP** из сети **192.168.2.0/24**.

А как быть, если нужно указать несколько портов? В этом случае нам понадобится подключение внешнего модуля **multiport**. Это делается при помощи указания параметра **-m**. Специфичными для этого модуля критериями являются **--sports, --source-ports** и **--dports, --destination-ports**, не путать с **--sport** и **--dport**, которые имеют такой же синтаксис, но позволяют указывать порты или диапазоны портов через запятую. Порты внутри критерия объединяются через логическое ИЛИ.

Для примера приведем типовое правило, разрешающее доступ к веб-серверу:

```
iptables -A INPUT -p tcp -m multiport --dports 80, 443 -j ACCEPT
```

Немного иное правило, запретим доступ к веб-серверу всем, кроме узлов локальной сети:

```
iptables -A INPUT !-s 192.168.2.0/24 -p tcp -m multiport --dports 80, 443 -j DROP
```

Здесь мы использовали критерий с отрицанием, который буквально читается как: запретить доступ к TCP-портам 80 и 443 всем, кроме сети 192.168.2.0/24. Без отрицания нам бы пришлось написать два правила:

```
iptables -A INPUT -s 192.168.2.0/24 -p tcp -m multiport --dports 80, 443 -j ACCEPT
iptables -A INPUT -p tcp -m multiport --dports 80, 443 -j DROP
```

Указанные нами критерии являются наиболее универсальными, в тоже время существуют критерии, относящиеся к какому-то отдельному протоколу, мы не будем подробно на них останавливаться, при необходимости обратитесь к документации по iptables. Так для TCP это могут быть критерии **--tcp-flags** для проверки TCP-флагов пакета, а для ICMP - **--icmp-type**, определяющий тип ICMP пакета, скажем правило:

```
iptables -I INPUT -p icmp --icmp-type echo-request -j ACCEPT
```

Разрешит только входящие эхо-запросы (пинги).

К полезным в повседневной деятельности внешним модулям, кроме **multiport**, следует отнести **iprange** и **mac**, которые позволяют указывать диапазоны IP-адресов не являющиеся сетью и MAC-адреса. Для **iprange** критериями являются **--src-range** и **--dst-range**, которые задают диапазон адресов источника или назначения, а для **mac** - **--mac-source**, определяющий MAC-адрес источника.

Например:

```
iptables -A FORWARD -o ens34 -p udp --dport 53 -m iprange --src-range
192.168.2.100-192.168.2.199 -d 8.8.8.8 -j REJECT
```

Данное правило запретит DNS-запросы к узлу **8.8.8.8** для устройств с адресами в диапазоне **192.168.2.100-192.168.2.199**.

А вот такое правило:

```
iptables -A INPUT -i ens33 -p UDP --sport 68 --dport 67 -m mac --mac-source
00:0C:29:77:E0:29 -j ACCEPT
```

Разрешает получение сетевых настроек только от доверенного DHCP-сервера с физическим адресом **00:0C:29:77:E0:29**.

Еще одним важным внешним модулем является **conntrack**, позволяющий контролировать состояние соединения, в частности нас будет интересовать критерий **--ctstate**, ранее в этом качестве использовался модуль **state** с с критерием **--state**, но в современных системах его использование не рекомендуется.

Критерий **--ctstate** позволяет отслеживать состояние соединения, которое может принимать следующие значения:

- **NEW** - пакет является первым в соединении, соединение еще не установлено.
- **ESTABLISHED** - пакет относится к уже установленному соединению.
- **RELATED** - первый пакет для соединения связанного с уже установленным, например, открытие канала передачи данных для FTP.
- **INVALID** - пакет, не принадлежащий ни одному соединению в системе и не являющийся первым пакетом соединения.

Также можно дополнительно отслеживать следующие состояния:

- **DNAT** - соединение подверглось операции подмены адреса назначения
- **SNAT** - соединение подверглось операции подмены адреса источника

Рассматривать работу этого модуля лучше на реальных примерах, поэтому попробуем создать базовую настройку брандмауэра для некого условного сервера, имеющего на борту веб-сервер, пассивный FTP и SSH для удаленного управления.

Существует два варианта настройки брандмауэра: **нормально открытый** (разрешено все, что не запрещено) и **нормально закрытый** (запрещено все, что не разрешено). Нормально открытый брандмауэр обычно используется только со стороны доверенных сетей, во всех остальных случаях следует использовать нормально закрытый брандмауэр.

Исходя из этих соображений на ум приходит достаточно просто набор правил:

```
iptables -A INPUT -i ens33 -p tcp --dport 21 -j ACCEPT
iptables -A INPUT -i ens33 -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -i ens33 -p tcp -m multiport --dports 80, 443 -j ACCEPT
iptables -A INPUT -i ens33 -j DROP
```

Казалось бы, все верно, но FTP-сервер с такой настройкой работать не будет, так как в пассивном режиме он открывает второе соединение на динамических портах, которых в разрешающих правилах нет. Можно, конечно, указать этот диапазон отдельно, но лучше поступить иначе и вспомнить о таком состоянии соединения как RELATED. Второй момент - правил может быть много, а пакет последовательно проходит все правила в цепочке до совпадения условий, вызывая повышенную нагрузку на процессор, что может быть очень актуально для маломощных устройств. Поэтому вспоминаем об еще одном состоянии соединения - ESTABLISHED.

Общий смысл следующего действия такой: если соединение уже установлено, следовательно оно было разрешено брандмауэром и дополнительно проверять его на совпадение условий не нужно. Все связанные с уже установленными новые соединения также можно разрешать, дополнительных проверок они не требуют. Поэтому **самым первым** правилом в цепочке мы должны поставить:

```
iptables -A INPUT -i ens33 -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Это позволит резко снизить нагрузку на систему, так как все относящиеся к установленным соединениям пакеты будут пропускаться этим правилом, без предварительных проверок, также сразу снимется проблема связанных соединений, они также будут разрешаться автоматически.

Вторым после него правилом можно указать:

```
iptables -A INPUT -i ens33 -m conntrack --ctstate INVALID -j DROP
```

Что позволит сразу отбрасывать все не принадлежащие существующим соединениям пакеты, а также исключить некоторые виды атак.

Если следовать данной логике и дальше, то можно изменить разрешающие правила следующим образом:

```
iptables -A INPUT -i ens33 -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT
```

В этом случае будут пропускаться только первые пакеты соединений. На первый взгляд особой разницы с обычной записью, без уточнения состояния тут нет, в любом случае по правилу будет проходить только первый пакет, а остальные пойдут как ESTABLISHED, но в ряде случаев через просто правило может проскочить и соответствующим образом подготовленный INVALID пакет, переведя соединение в ESTABLISHED, с явным указанием соединения такого уже не произойдет.

Если наш сервер является еще и маршрутизатором (роутером), то потребуется настроить также цепочку FORWARD, в простейшем случае это два правила:

```
iptables -A FORWARD -i ens33 -o ens34 -m conntrack --ctstate ESTABLISHED,RELATED -
j ACCEPT
iptables -A FORWARD -i ens33 -o ens34 -j DROP
```

Отдельного разговора заслуживает проброс портов. Многие ошибочно считают, что разрешать такие подключения следует в цепочке INPUT, однако это неверно, так как прежде такой пакет попадет в цепочку PREROUTING таблицы nat и его адрес назначения будет изменен на адрес конечного узла в локальной сети. Поэтому разрешающее правило должно выглядеть так:

```
iptables -A FORWARD -i ens33 -o ens34 -d 192.168.2.220 -p tcp --dport 3389 -j
ACCEPT
```

Где в качестве узла назначения указываем внутренний адрес узла, на который мы пробрасываем порт. Также стоит обратить внимание на еще один момент, если мы делаем проброс с изменением номера порта, скажем для 192.168.2.220 делаем проброс 3389 -> 3389, а для 192.168.2.221 3390 -> 3389, то в обоих правилах мы должны будем указать один и тот же порт - порт назначение, т.е. 3389:

```
iptables -A FORWARD -i ens33 -o ens34 -d 192.168.2.220 -p tcp --dport 3389 -j
ACCEPT
iptables -A FORWARD -i ens33 -o ens34 -d 192.168.2.221 -p tcp --dport 3389 -j
ACCEPT
```

Либо можно сделать короче:

```
iptables -A FORWARD -i ens33 -o ens34 -d 192.168.2.220, 192.168.2.221 -p tcp --
dport 3389 -j ACCEPT
```

Таким образом минимальная конфигурация брандмауэра будет выглядеть так:

```
#Разрешаем уже установленные и связанные соединения
iptables -AINPUT -i ens33 -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT



#Разрешаем входящий SSH
iptables -AINPUT -i ens33 -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT



#Запрещаем остальные входящие соединения
iptables -AINPUT -i ens33 -j DROP



#Разрешаем уже установленные и связанные транзитные соединения  WAN -> LAN
iptables -A FORWARD -i ens33 -o ens34 -m conntrack --ctstate ESTABLISHED,RELATED -
j ACCEPT

#Запрещаем остальные транзитные соединения WAN -> LAN
iptables -A FORWARD -i ens33 -o ens34 -j DROP
```

Это тот, необходимый каркас вокруг которого будут выстраиваться все остальные правила. При их создании следует придерживаться следующих принципов: все правила, относящиеся к одной цепочке, должны быть записаны рядом, это облегчает чтение и понимание конфигурации. Правила описывающие более частные случаи должны располагаться выше более общих. При создании правил всегда обращайте внимание на последнее действие в цепочке и действие по умолчанию.

В заключение хотелось бы коснуться распространенных ошибок. Одна из них - это **правила-пустышки**, которые разрешают или запрещают то, что уже разрешено/ запрещено. Яркий пример:

```
iptables -A FORWARD -i ens33 -o ens34 -m conntrack --ctstate ESTABLISHED,RELATED -
j ACCEPT
iptables -A FORWARD -i ens34 -o ens33 -j ACCEPT
iptables -A FORWARD -i ens33 -o ens34 -j DROP
```

Второе правило представляет типичную пустышку. Запрещающего правила для трафика от ens34 к ens33 нет, по умолчанию цепочка FORWARD имеет действие ACCEPT, т.е. мы разрешаем то, что и так разрешено. Особого вреда от этого нет,

кроме повышенной нагрузки на оборудование.

Единственная пустышка, которую следует применять - это разрешение доступа к устройству первой строкой конфигурации, даже если это разрешено по умолчанию. Таким образом вы страхуетесь от того, что, изменив какие-либо правила, либо добавив новые вы потеряете контроль над устройством.

Вторая ошибка - **лжеправила**, обычно это вычитанные где-то в интернете конструкции, содержащие массу непонятных параметров, скажем TCP-флаги и позиционирующиеся как "защита от атак" и т.д. и т.п. В лучшем случае это будет очередная пустышка, в худшем такое правило может нарушить нормальную работу сети. То же самое касается правил с лимитами, квотами и т.п. конструкциями. Общий принцип здесь простой: если вы не понимаете, что делает то или иное правило - оно вам не нужно.

Ну и классика жанра - **неверное расположение правил**. Часто это происходит не сознательно, а от неправильно или неявно заданных параметров. Допустим:

```
iptables -A FORWARD -i ens33 -o ens34 -j ACCEPT
iptables -A FORWARD -s 192.168.1.0/24 -j REJECT
```

На первый взгляд все должно работать, но если сеть **192.168.1.0/24** расположена за адаптером **ens33**, то данная конструкция работать не будет, так как прежде запрещающего правила по адресу у вас стоит разрешающее правило по интерфейсу.

Поэтому вспоминаем совет: более частные правила ставить выше более общих. Сеть - более узкий критерий, нежели интерфейс (за которым может быть несколько сетей), поэтому такие правила должны быть указаны раньше.

К этой же категории относится и составление правил без учета последнего правила в цепочке или действия по умолчанию. Допустим перед нами стоит задача разрешить транзитный трафик из сети за **ens33** в сеть за **ens34** только для подсети **192.168.2.0/24**. Для правильного ее решения нам нужно знать тип брандмауэра со стороны исходящей сети и проверить весь набор правил. Если брандмауэр нормально закрытый, т.е. у нас написано что-то наподобие

```
iptables -A FORWARD -i ens33 -o ens34 -m conntrack --ctstate ESTABLISHED,RELATED -
j ACCEPT
...
iptables -A FORWARD -i ens33 -o ens34 -j DROP
```

То мы просто можем добавить:

```
iptables -A FORWARD -i ens33 -o ens34 -m conntrack --ctstate ESTABLISHED,RELATED -
j ACCEPT
iptables -A FORWARD -i ens33 -o ens34 -s 192.168.2.0/24 -j ACCEPT
...
iptables -A FORWARD -i ens33 -o ens34 -j DROP
```

Иначе, если брандмауэр нормально открытый, что в порядке вещей для доверенных сетей, такое правило окажется пустышкой (запрещающего правила нет, а по умолчанию и так стоит ACCEPT) и работать не будет. В этом случае вам понадобится совсем иная конструкция:

```
iptables -A FORWARD -i ens33 -o ens34 ! -s 192.168.2.0/24 -j REJECT
```

Если же применить ее в предыдущей конфигурации, то данное правило тут же станет пустышкой. Это хороший пример того, как работающие в одной ситуации правила в другом контексте полностью теряют свой смысл. Это всегда следует учитывать при копировании правил из чужих конфигурации или примеров в интернете, а также обращаясь за помощью на форумах, потому что рассматривать конфигурацию брандмауэра нужно с учетом всех правил в цепочке и их взаимного расположения.

В данном материале мы рассмотрели только базовые возможности таблицы **filter**, оставив за скобками многие продвинутые вещи, скажем, лимиты. Это обусловлено предназначением статьи прежде всего для начинающих, а применение сложных параметров требует определенного уровня знаний, поэтому мы будем следовать в подаче материала от простого к сложному и непременно коснемся этих параметров в будущих статьях цикла.