Лабораторная работа 7. Межсетевое экранирование и трансляция сетевых адресов в GNU/Linux (nftables /netfilter).
Теоретический материал.

Маршрутизация — процесс определения лучшего пути, по которому пакет может быть доставлен получателю.

Возможные пути передачи пакетов называются маршрутами. Лучшие маршруты к известным получателям хранятся в таблице маршрутизации.

В зависимости от способа заполнения таблицы маршрутизации, различают два вида маршрутизации:

- Статическая маршрутизация
- Динамическая маршрутизация

Устройство, выполняющее маршрутизацию, именуется маршрутизатором. В качестве маршрутизатора могут использоваться специализированные устройства, такие, например, как Cisco Router, или это могут быть обычные компьютеры, оснащённые несколькими сетевыми картами (или принимающие тегированный трафик) и работающие под управлением универсальной операционной системы, такой как FreeBSD или GNU/Linux.

Для управления маршрутизацией в GNU/Linux применяются следующие команды: route, netstat, ip. При этом следует учитывать, что доступ ко всем возможностям дает только ip.

Linux позволяет разрешить или запретить пересылку пакетов между интерфейсами (forwarding). На рабочих станциях и серверах приложений ее можно запретить, на маршрутизаторах или межсетевых экранах она, очевидно, должна быть разрешена.

За этот параметр для IPv4 отвечает переменная net.ipv4.ip_forward (1 = «разрешить», 0 = «запретить»), для IPv6 используется переменная net.ipv6.conf.all.forwarding.

nftables и netfilter

Netfilter — компонент ядра Linux, обеспечивающий фильтрацию и модификацию трафика, межсетевой экран (брандмауэр). Netfilter часто ассоциируется с iptables — пользовательской утилитой, предназначенной для

управления системой netfilter. Netfilter встроен в ядро Linux с версии 2.4 (2001 год).

nftables – проект netfilter, целью которого ставится замена существующего набора программ iptables, ip6tables, arptables, ebtables. Была разработана новая система фильтрации пакетов, добавлена пользовательская утилита ntf, а также создан слой совместимости с iptables, ip6tables. nftables использует набор хуков, систему отслеживания соединений, систему очередей и подсистему логирования netfilter.

nftables включен в ядро Linux, начиная с версии 3.13 (2014 год).

nftables состоит из трёх основных частей: низкоуровневая реализация в составе ядра, библиотека libnl и пользовательский интерфейс nftables. Ядро занимается выполнением правил во время работы межсетевого экрана; для настройки правил в ядро добавлен интерфейс netlink. Библиотека libnl содержит низкоуровневые функции для взаимодействия с ядром, а в качестве пользовательского интерфейса nftables используется утилита nft.

Как и в случае с фреймворком iptables, nftables построен на правилах, определяющих действия. Эти правила прикреплены к цепочкам. Цепочка может содержать набор правил и регистрируется в хуках netfilter. Цепочки хранятся внутри таблиц. Таблица специфична для одного из протоколов уровня 3. Одно из основных отличий iptables заключается в том, что больше нет предопределенных таблиц и цепочек

Таблицы

Таблица (table) - это не что иное, как контейнер для цепочек. В nftables больше нет предопределенных таблиц (filter, raw, mangle …), но есть возможность воссоздать структуру, подобную iptables.

В настоящее время существует 5 различных семейств таблиц:

- **ip** : используется для цепочек, связанных с IPv4.
- **ip6** : используется для цепочек, связанных с IPv6.
- **arp** : используется для цепочек, связанных с ARP.
- **bridge** : используется для настройки мостов.
- **inet** : смешанные цепочки ipv4 / ipv6 (ядро 3.14 и выше).
- **netdev** : используется для создания базовых цепочек, прикрепленных к единому сетевому интерфейсу. Такие базовые цепочки видят весь

сетевой трафик на указанном интерфейсе без каких-либо предположений о протоколах L2 или L3 (ядро 4.2 и выше).

В этих таблицах нетрудно распознать старую структуру таблиц. Однако семейства таблиц inet и netdev не имеют аналогов в мире iptables. Семейство таблиц inet используется как для трафика IPv4, так и для трафика IPv6. Он должен упростить работу файрвола для хостов с двойным стеком, объединив правила для IPv4 и IPv6. Семейство таблиц netdev видит пакеты, которые драйвер только что передал в сетевой стек. Поэтому он используется для более эффективной фильтрации входящего трафика.

Цепочки

Цепочки (chains) используются для группировки правил. Как и в случае с таблицами, nftables не имеет предопределенных цепочек. Цепочки сгруппированы по базовым и небазовым типам. Базовые цепочки регистрируются в одном из хуков netfilter, небазовые цепочки - нет. Таким образом, базовая цепочка имеет зарегистрированный хук, тип и приоритет. Напротив, небазовые цепочки не привязаны к хуку и по умолчанию не видят никакого трафика. Их можно использовать для организации набора правил в дереве цепочек.

В настоящее время существует три типа цепочек:

- **filter** : для фильтрации пакетов.
- **route** : для перенаправления пакетов.
- **nat** : для выполнения трансляции сетевых адресов. В эту цепочку попадает только первый пакет потока, что делает невозможным его использование для фильтрации.

Можно использовать следующие хуки:

- **prerouting** : входящий трафик, обрабатываемый до принятия решения о маршрутизации;
- **input** : все входящие пакеты для локальной системы;
- **forward** : пакеты не для локальной системы, т.е. которые необходимо переадресовать;
- **output** : пакеты, исходящие из локальной системы;
- **postrouting** : исходящий трафик, обрабатываемый после принятия решения о маршрутизации.

Правила

Правила (rules) определяют, какое действие необходимо выполнить для каких пакетов. Правила прикреплены к цепочкам. Каждое правило может иметь выражение для сопоставления пакетов и одно или несколько действий, выполняемых при сопоставлении. Одно из основных отличий от iptables заключается в том, что для каждого правила можно указать несколько действий. Другое отличие заключается в том, что по умолчанию выключены счетчики. Счетчик должен быть указан явно в каждом правиле, для которого требуются счетчики пакетов или счетчики байтов.

Список возможных параметров для правил:

- **ip** : IP-протокол.
- **ip6** : протокол IPv6.
- **tcp** : протокол TCP.
- **udp** : протокол UDP.
- **udplite** : протокол UDP-lite.
- **sctp** : протокол SCTP.
- **dccp** : протокол DCCP.
- **ah** : протокол Authentication Header (AH).
- **esp** : протокол Encapsulating Security Payload (ESP).
- **ipcomp** : заголовки IPcomp.
- **icmp** : протокол icmp.
- **icmpv6** : протокол icmpv6.
- **ct** : отслеживание соединений.
- **meta** : метасущности, напр. интерфейсы.

Действия

Действие (statement) представляет собой операцию, которая должна быть выполнена при срабатывании правила. Действия бывают двух видов: завершающие — безоговорочно завершающие оценку текущих правил, и незавершающие — либо условно, либо никогда не завершающие текущие

правила. Может быть произвольное количество незавершающих действий, но должно быть только одно завершающее.

Завершающие действия :

- **accept** : принять пакет и остановить оценку набора правил.
- **drop** : отбросить пакет и остановить оценку набора правил.
- **reject** : отклонить пакет с сообщением icmp.
- **queue** : поставить пакет в очередь в пользовательское пространство и остановить оценку набора правил.
- **continue** : продолжить оценку набора правил со следующим правилом. Это поведение по умолчанию, если правило не выносит вердикта.
- **return** : возврат из текущей цепочки и переход к следующему правилу последней цепочки. В базовой цепочке это эквивалентно accept.
- **jump *цепочка***: продолжить с первого правила *цепочки*. Оно будет продолжено в следующем правиле после того, сработает return.
- **goto *цепочка***: аналогично действию jump, но после новой цепочки оценка продолжится с последней цепочки вместо той, которая содержит goto.

Множества

Множества (sets) бывают именованные и анонимные. Множество объявляется фигурными скобками, в которых перечислены разделённые запятыми элементы. Анонимные множества "встраиваются" в правило и не могут быть изменены отдельно от него. Если анонимное множество необходимо модифицировать, то придётся удалить правило целиком и добавить его заново. Именованное множество можно изменить, а также присвоить ему тип и установить флаги.

Примеры:

- набор портов: {22, 23}
- набор протоколов: {telnet, http, https}

Применение

Все команды nftables выполняются с помощью утилиты nft.

Действующие правила можно узнать командой:

```
# nft list ruleset
```

В пакет nftables входит простая и безопасная конфигурация экрана, сохранённая в файле /etc/nftables.conf. Служба nftables.service загрузит эти правила при запуске или включении.

Работа с таблицами

Добавить новую таблицу:

```
# nft add table семейство таблица
```

Название таблицы может быть произвольным.

Пример команды для создания таблицы с именем filter для IP(v4):

```
#  nft add table ip filter
```

Вывести список таблиц:

```
# nft list tables
```

Вывести цепочки и правила выбранной таблицы:

```
# nft list table семейство таблица
```

Например, чтобы вывести на экран правила таблицы filter семейства inet, выполните команду:

```
# nft list table inet filter
```

Удалить таблицу со всеми цепочками:

```
# nft delete table семейство таблица
```

Стереть все правила таблицы:

```
# nft flush table семейство таблица
```

Работа с цепочками

Добавить обычную цепочку цепочка в таблицу *таблица*:

```
# nft add chain семейство таблица цепочка
```

Например, чтобы добавить обычную цепочку tcpchain к таблице filter семейства адресов inet, выполните:

```
# nft add chain inet filter tcpchain
```

Чтобы добавить базовую цепочку, укажите хук и значение приоритета:

```
# nft add chain семейство таблица цепочка '{ type тип hook хук priority приоритет ; }'
```

тип может принимать значения filter, route или nat.

Для семейств адресов IPv4/IPv6/Inet *хук* может принимать значения prerouting, input, forward, output или postrouting.

Параметр *приоритет* задаётся названием приоритета или его значением. Цепочки с более низкими значениями обрабатываются раньше. Значение приоритета может быть отрицательным.

Например, добавим базовую цепочку для фильтрации входящих пакетов:

```
# nft add chain inet filter input '{ type filter hook input priority 0; }'
```

Если заменить команду add на create в любом из правил выше, то цепочка тоже будет создана, но при этом вы получите сообщение об ошибке, если цепочка с таким названием уже существует.

Следующая команда выводит на экран все правила в цепочке:

```
# nft list chain семейство таблица цепочка
```

Например, следующая команда выведет правила цепочки output таблицы filter семейства inet:

```
# nft list chain inet filter output
```

Для редактирования цепочки укажите её название и правила, которые нужно изменить:

```
# nft chain семейство таблица цепочка '{ [ type тип hook хук device устройство priority приоритет ; policy политика ; ] }'
```

Например, для изменения политики обработки входящих пакетов цепочки исходной таблицы с accept на drop выполните команду:

```
# nft chain inet filter input '{ policy drop ; }'
```

Удалить цепочку:

```
# nft delete chain семейство таблица цепочка
```

Цепочка должна быть пустой (не содержать правил) и не должна быть целью перехода из другой цепочки.

Чтобы стереть все правила цепочки, выполните:

```
# nft flush chain семейство таблица цепочка
```

Работа с правилами

Чтобы добавить правило в цепочку, выполните:

```
# nft add rule семейство таблица цепочка handle маркер действие
```

Правило будет прикреплено после *маркера*, который можно не указывать. Если маркер (handle) не задать, то правило добавится в конец цепочки.

Чтобы добавить правило перед определённой позицией, выполните

```
# nft insert rule семейство таблица цепочка handle маркер действие
```

Если маркер не указан, то правило добавится в начало цепочки.

Следующая команда добавляет правило в цепочку input таблицы filter с семейством ip, отбрасывая весь трафик, входящий на порт 80:

```
# nft add rule ip filter input tcp dport 80 drop
```

Отдельные правила могут быть удалены только с помощью маркера. Команда nft --handle list выведет список маркеров.

Ниже приведён пример правила с маркером и команда для его удаления. Аргумент --numeric позволяет выводить IP-адреса в численном виде.

```
# nft --handle --numeric list chain inet filter input
table inet fltrTable {
    chain input {
        type filter hook input priority 0;
        ip saddr 127.0.0.1 accept # handle 10
    }
}
# nft delete rule inet fltrTable input handle 10
```

Стирание всех цепочек таблицы выполняется командой nft flush table. Отдельные цепочки могут быть стёрты командами nft flush chain или nft delete rule.

```
# nft flush table foo
# nft flush chain foo bar
# nft delete rule ip6 foo bar
```

Первая команда стирает все цепочки в ip-таблице foo. Вторая - цепочку bar в ip-таблице foo. Третья удаляет все правила в цепочке bar в ip6-таблице foo.

Порт

Порт (англ. port) — целое неотрицательное число, записываемое в заголовках протоколов транспортного уровня сетевой модели OSI (TCP, UDP, SCTP, DCCP).

Обычно на хосте под управлением ОС в пространстве пользователя исполняется несколько процессов, в каждом из которых выполняется какая-либо программа. В случае если несколько программ используют компьютерную сеть, то ОС периодически получает по сети IP-пакет, предназначенный для одной из программ.

Процесс программы, желающей обмениваться данными по сети, может (например, при создании socket):

- запросить у ОС в своё распоряжение порт с определённым номером. ОС может либо предоставить порт с этим номером, либо отказать программе (например, в случае, когда порт с этим номером уже отдан в распоряжение другому процессу);
- запросить у ОС в своё распоряжение свободный порт с любым номером. ОС в этом случае сама выберет свободный порт, ещё не занятый никаким процессом, и предоставит его в распоряжение запрашивающей программе.

Обмен данными по сети ведётся между двумя процессами по определённому протоколу. Для установки соединения необходимы:

- номер протокола;
- два IP-адреса (адрес хоста-отправителя и адрес хоста-получателя для построения маршрута между ними);
- два номера порта (порт процесса-отправителя и порт получателя).

Порт процесса-отправителя (источника) может быть постоянным (статическим) или назначаться динамически для каждого нового сеанса связи.

При соединении по протоколу TCP порт процесса-отправителя используется:

- операционной системой хоста-получателя для отправки пакета-подтверждения о получении данных;
- процессом-получателем для отправки пакета-ответа.

При соединении по протоколу UDP допустимо вместо порта процесса-отправителя указывать число ноль, означающее «порт не указан».

При соединении по протоколу SCTP в рамках ассоциации может использоваться:

- несколько портов процесса-отправителя (источника)
- несколько портов процесса-получателя.

Так как IP-адрес хоста-отправителя и номер порта процесса-отправителя являются аналогом обратного адреса, записываемого на почтовых конвертах (позволяют получателю отправить ответ отправителю), номер порта процесса-отправителя иногда называют «обратным» портом.

Если на хосте какой-либо процесс постоянно использует один номер порта (например, процесс программы, реализующей web-сервер, может использовать порт 80 для приёма и передачи данных), говорят, что порт является «открытым».

Термины «открытый порт» и «закрытый порт» (заблокированный) также используются, когда речь идёт о фильтрации сетевого трафика.

Если процесс получил номер порта у ОС («открыл порт») и «держит его открытым» для приёма и передачи данных, говорят, что процесс «прослушивает» (разг. слушает, от англ. listen) порт.

Обычно прослушиванием порта занимается процесс программы, реализующей сервер для какого-либо протокола. Процесс программы, реализующей клиент для того же протокола, часто позволяет ОС указать номер порта для подключения к серверу.

Если хост получит пакет, порт процесса-отправителя называется «удалённым» (англ. remote) портом или "открытым на другом хосте", а порт процесса получателя — «локальным» портом, то есть открытым на текущем хосте. Если хост отправил пакет, порт процесса-отправителя называется «локальным» портом (открытым на текущем хосте), а порт процесса-получателя — «удалённым» портом (открытым на другом хосте).

Номера портов для протоколов прикладного уровня модели TCP/IP (HTTP, SSH и др.) обычно назначаются организацией IANA (англ. internet assigned numbers authority). Однако на практике в целях безопасности номера портов могут выбираться произвольно.

Термин «порт» чаще всего применяется по отношению к протоколам TCP и UDP ввиду популярности этих протоколов. В протоколах SCTP и DCCP

используются номера, соответствующие понятию «номер порта» для протоколов TCP и UDP.

В заголовках протоколов TCP и UDP для хранения номеров портов выделены поля размером 16 бит. Для протокола TCP порт с номером 0 зарезервирован и не может использоваться. Для протокола UDP указание порта процесса-отправителя («обратного» порта) не является обязательным, и порт с номером 0 означает отсутствие порта. Таким образом, номер порта — число в диапазоне от 1 до 2^{16}-1 = 65 535.

Номера портов

Номера портов, используемые для конкретных специфических целей, выделяет и регистрирует IANA (Internet Assigned Numbers Authority), однако на практике часто встречаются случаи их неофициального применения.

Все порты разделяются на три диапазона — общеизвестные (или системные, 0—1023), зарегистрированные (или пользовательские, 1024—49151) и динамические (или частные, 49152—65535).

Примеры общеизвестных портов:

- 21/TCP – FTP
- 22/TCP,UDP – SSH
- 25/TCP,UDP – SMTP
- 80/TCP,UDP – HTTP

Примеры зарегистрированных (пользовательских) портов:

- 3306/TCP,UDP – MySQL
- 3389/TCP,UDP – MS WBT Server (RDP)
- 5060/TCP,UDP — SIP

Многие версии ядра Linux используют в качестве динамических порты 32768 — 60999 (диапазон задаётся в файле /proc/sys/net/ipv4/ip_local_port_range).

Microsoft Windows начиная с версии Vista и Server 2008 использует стандартный диапазон портов IANA.

FreeBSD, начиная с релиза 4.6, использует стандартный диапазон портов IANA.

Список источников:

1. http://xgu.ru/wiki/Маршрутизация

2. http://xgu.ru/wiki/Маршрутизация_в_Linux

3. https://wiki.archlinux.org/index.php/Nftables_(Русский)

4. https://wiki.debian.org/ru/nftables

5. https://wiki.gentoo.org/wiki/Nftables

6. https://ru.wikipedia.org/wiki/Порт_(компьютерные_сети)