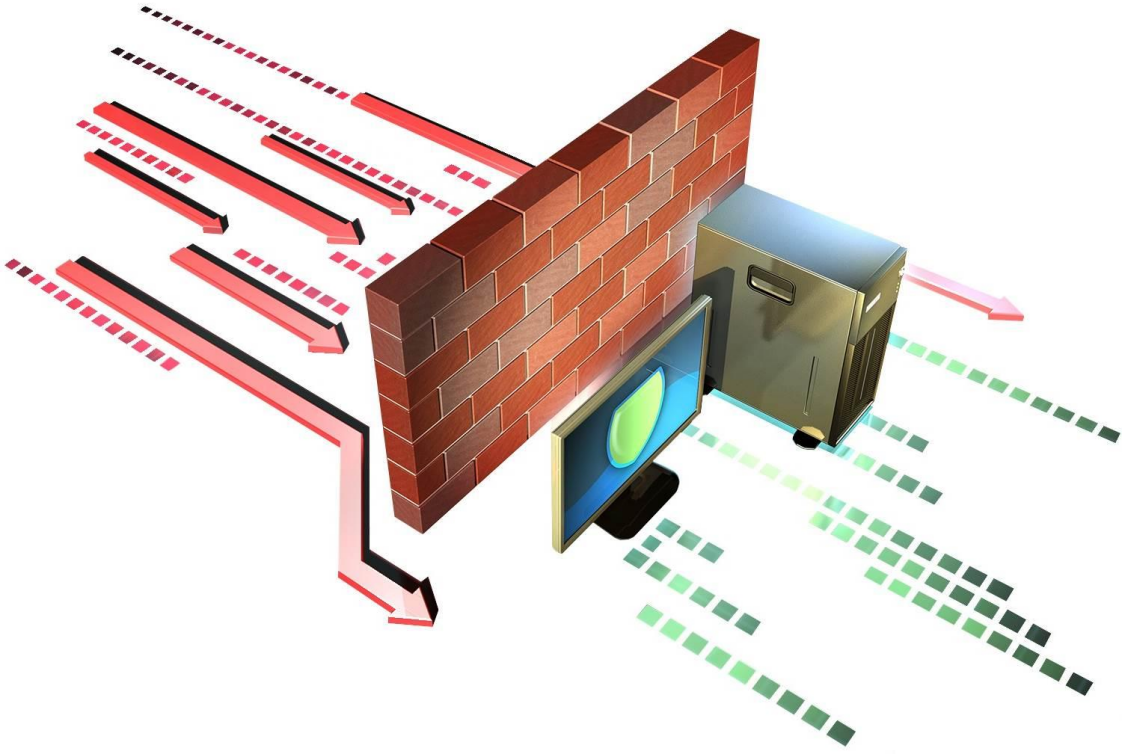
УНИВЕРСИТЕТ ИТМО

М.Б. Будько, М.Ю. Будько, А.В. Гирик

ИСПОЛЬЗОВАНИЕ МЕЖСЕТЕВОГО ЭКРАНА NETFILTER ДЛЯ ОБЕСПЕЧЕНИЯ СЕТЕВОЙ БЕЗОПАСНОСТИ В ОС LINUX



Санкт-Петербург
2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

М.Б. Будько, М.Ю. Будько, А.В. Гирик

ИСПОЛЬЗОВАНИЕ МЕЖСЕТЕВОГО ЭКРАНА NETFILTER ДЛЯ ОБЕСПЕЧЕНИЯ СЕТЕВОЙ БЕЗОПАСНОСТИ В ОС LINUX

УЧЕБНОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 10.03.01 Информационная безопасность
в качестве учебного пособия для реализации основных профессиональных
образовательных программ высшего образования бакалавриата

Санкт-Петербург

2020

Будько М.Б., Будько М.Ю., Гирик А.В. Использование межсетевого экрана Netfilter для обеспечения сетевой безопасности в ОС Linux : Учебное пособие. — СПб : Университет ИТМО, 2020. — 56 с.

Рецензент:
Алиев Тауфик Измайлович, доктор технических наук, профессор, профессор (квалификационная категория «профессор-эксперт») факультета программной инженерии и компьютерной техники, Университета ИТМО.

Учебное пособие посвящено различным аспектам применения системы Netfilter ядра ОС Linux, обеспечивающей функциональность межсетевого экрана с отслеживанием состояний соединений и рядом других полезных возможностей. В пособии рассмотрены общие сведения о сетевом стеке TCP/IP и межсетевых экранах, архитектура системы Netfilter, методы обеспечения фильтрации пакетов на сетевом и транспортном уровнях, работа механизма отслеживания соединений и возможности фильтрации пакетов на основе этого механизма, а также кратко рассмотрены требования, предъявляемые контролирующими государственными организациями к межсетевым экранам.

Учебное пособие предназначено для обучающихся по направлению подготовки 10.03.01 Информационная безопасность, изучающих дисциплины, связанные с информационной безопасностью и, в частности, с сетевой безопасностью и современными методами противодействия сетевым угрозам. Освоение изложенного материала требует знания основ информатики, компьютерных сетей, операционных систем, а также базовых знаний о структурах данных и алгоритмах.

УНИВЕРСИТЕТ ИТМО

Университет ИТМО — ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО — участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО — становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Содержание

Список аббревиатур

АСУТП — автоматическая система управления технологическим процессом.
АТС — автоматическая телефонная станция.
БД — база данных.
ГИС — геоинформационная система.
ИБ — информационная безопасность.
ИС — информационная система.
ИСПДн — информационная система персональных данных.
ЛВС — локальная вычислительная сеть.
МЭ — межсетевой экран.
ОС — операционная система.
ПО — программное обеспечение.
ПЗ — профиль защиты.
СЗИ — система защиты информации.
СПД — сеть передачи данных.

DPI — Deep Packet Inspection.
ICMP — Internet Control Message Protocol.
MTU — Maximum Transmission Unit.
OSI — Open Systems Interconnection.
TCP — Transmission Control Protocol.
UDP — User Datagram Protocol.

Введение

В настоящее время наблюдается рост количества информационных угроз и факторов, приводящих к нестабильному функционированию сетей передачи данных (СПД). Основными предпосылками к появлению уязвимостей являются массовость применения и усложнение иерархии вычислительных сетей и функциональности сетевых сервисов, увеличение гетерогенности программных и аппаратных средств.

При защите от сетевых атак в качестве обязательного элемента используют специальные программные и программно-аппаратные комплексы, называемые межсетевыми экранами. Межсетевые экраны представляют собой часть сетевой подсистемы узлов сети передачи данных, непосредственно реализующую функции перехвата, анализа и модификации сетевых пакетов на основе некоторой системы правил, заданной администратором или сформированной более высокоуровневым, чем межсетевой экран, приложением, например, системой обнаружения и противодействия вторжениям.

Также необходимо отметить важность освоения специалистами по информационной безопасности вопросов, связанных с применением операционной системы Linux, в решении прикладных задач сетевой безопасности, так как в настоящее время решения на основе ОС Linux (начиная от высокопроизводительных программно-аппаратных комплексов маршрутизации и фильтрации трафика и заканчивая контроллерами для киберфизических систем и Интернета вещей) приобрели большую популярность вследствие дешевизны, универсальности и широкого выбора совместимых пакетов программ. В ближайшем будущем популярность таких решений будет непрерывно расти. Таким образом, понимание того, как работает и настраивается межсетевой экран ОС Linux, является необходимой компетенцией обучающихся по направлению «Информационная безопасность».

Настоящее учебное пособие посвящено различным аспектам применения системы Netfilter ядра ОС Linux, обеспечивающей функциональность межсетевого экрана с отслеживанием состояний соединений и рядом других полезных возможностей. В пособии рассмотрены общие сведения о сетевом стеке TCP/IP и межсетевых экранах, архитектура системы Netfilter, методы обеспечения фильтрации пакетов на сетевом и транспортном уровнях, работа механизма отслеживания соединений и возможности фильтрации пакетов на основе этого механизма, а также кратко рассмотрены требования, предъявляемые контролирующими государственными организациями к межсетевым экранам.

Учебное пособие содержит 5 разделов и список рекомендуемой литературы. Каждый раздел сопровождается вопросами для контроля усвоения изученного материала.

Учебное пособие предназначено для обучающихся по направлению подготовки 10.03.01 Информационная безопасность, изучающих дисциплины, связанные с информационной безопасностью и, в частности, с сетевой безопасностью и современными методами противодействия сетевым угрозам. Освоение изложенного материала требует знания основ информатики, компьютерных сетей, операционных систем, а также базовых знаний о структурах данных и алгоритмах.

Раздел 1.
Общие сведения о межсетевых экранах

Информационная безопасность (*ИБ*) — такое состояние информационного поля, при котором исключается его случайный или преднамеренный перехват, декодирование и расшифровка, несанкционированное использование и копирование, внесение в информационное поле инородных информационных потоков, приводящих к его искажению либо разрушению.

Угроза (информационной безопасности) — это совокупность условий, создающих опасность нарушения информационной безопасности. С самой общей точки зрения информационные угрозы можно поделить на три категории: угрозы целостности информации, угрозы конфиденциальности информации и угрозы доступности информации.

Безопасность сети передачи данных — защита СПД от несанкционированного использования, приводящего к изменению, раскрытию или разрушению ее структуры, функциональных свойств или передаваемых данных. Задача обеспечения ИБ СПД должна решаться комплексно в силу следующих обстоятельств:

1) сеть представляет собой совокупность взаимодействующих программных и аппаратных средств разных производителей и построенных в соответствии с множеством различных стандартов, то есть вычислительную среду с высокой степенью гетерогенности. Специфика каждого сетевого элемента должна учитываться при проектировании системы управления безопасностью сети;

2) сетевое взаимодействие осуществляется по протоколам, принадлежащим определенному сетевому стеку, например, TCP/IP. Необходимо заботиться о том, чтобы на каждом уровне была обеспечена надежная защита передаваемых данных;

3) наличие разнородных механизмов защиты в сети, их взаимное влияние и влияние на производительность сети должны тщательно исследоваться. Работа различных подсистем безопасности должна быть согласована;

4) при проектировании системы защиты необходимо ориентироваться на статистику угроз и учитывать возможность появления потенциальных (неизвестных) угроз.

Сетевая атака представляет собой реализованную угрозу ИБ компонентов СПД, то есть деструктивное воздействие на информационную систему через СПД. Термин «вторжение» (intrusion) зачастую используется как синоним термина «атака», однако, говоря строго, вторжение может являться лишь частью атаки.

Для того чтобы успешно противостоять атакам на СПД (целью атак может быть как само управляемое оборудование СПД, так и ресурсы, доступ к которым предоставляется через СПД), необходимо обеспечить невозможность злонамеренного воздействия на компоненты СПД, в том числе невозможность эксплуатации уязвимостей компонентов СПД, под которыми понимаются ключевые узлы вычислительной сети: коммутаторы, маршрутизаторы, серверы, АТС и другое активное телекоммуникационное оборудование. В некоторых случаях компонентами СПД можно считать и программные системы, например, приложение веб-сервер или приложение DHCP-сервер.

В последнее десятилетие все чаще компоненты СПД реализуются на основе POSIX-совместимых операционных систем, наиболее распространенной из которых является ОС Linux. Таким образом, понимание того, как работает сетевая подсистема ОС Linux и механизмы сетевой защиты, становится совершенно необходимым для обеспечения информационной безопасности передачи данных в СПД и защиты данных, которые хранятся и обрабатываются узлами сети.

Для обнаружения и предупреждения сетевых атак используются специальные программные или программно-аппаратные комплексы — *системы обнаружения и предотвращения вторжений* (*СОВ*). Ключевым компонентом таких систем, который может использоваться и независимо от них, является *межсетевой экран* (*МЭ*) — *брандмауэр* (нем. brandmauer), *файерволл* (*firewall*). МЭ, как правило, представляет собой часть сетевой подсистемы компонента СПД, непосредственно реализующую функции перехвата, анализа и модификации сетевых пакетов на основе системы правил, заданной администратором компонента СПД или сформированной более высокоуровневым, чем МЭ, приложением (например, СОВ).

В ОС Linux МЭ реализован в виде подсистемы ядра под названием Netfilter.

1.1. Возможности межсетевых экранов

Рассмотрим классификацию МЭ и их основные возможности.

По типу реализации МЭ можно разделить на программные, программно-аппаратные и аппаратные. Последние встречаются редко и обладают ограниченной функциональностью. Граница между аппаратными и программно-аппаратными решениями в настоящее время достаточно размытая; иногда аппаратными МЭ называют программно-аппаратные решения, выполненные в виде отдельных изделий.

По уровню поддержки стека сетевых протоколов МЭ можно разделить на обеспечивающие фильтрацию на:

– канальном уровне;

– сетевом и более низких уровнях;
– транспортном и более низких уровнях;
– сеансовом и более низких уровнях;
– прикладном и более низких уровнях.

Наиболее распространены МЭ, обеспечивающие фильтрацию на транспортном и более низких уровнях. Часто они дополняются функциональными элементами, позволяющими выполнять ограниченный анализ или даже модификацию пакетов более высоких уровней. Полноценная реализация МЭ, обеспечивающих фильтрацию на прикладном уровне, возможна только в рамках оборудования, предназначенного для глубокого анализа потоков сетевых пакетов (*deep packet inspection, DPI*).

По возможности отслеживания потоков пакетов МЭ можно разделить на МЭ:

– без учета состояния соединений (*stateless firewall*);
– с учетом состояния соединений (*stateful firewall*).

Как было сказано ранее, МЭ зачастую является частью сетевого стека узла СПД (например, в случае хоста с ОС Linux), однако само по себе это не означает, что МЭ будет автоматически задействован для операций с пакетами, доставляемыми конкретному узлу. Как правило, если речь идет о рабочих станциях пользователей, МЭ на них может быть неактивен. С другой стороны, на сервере или узле, выступающем в качестве маршрутизатора, то есть находящемся на границе сетей, включение МЭ желательно или необходимо. Нужно понимать, что само по себе включение МЭ не увеличивает безопасность узла. Например, если на сервере нет запущенных сетевых сервисов и, следовательно, нет открытых портов и слушающих на этих портах приложений, то воспользоваться потенциальными уязвимостями в прикладном коде, работающем с сетью, злоумышленнику в любом случае не удастся. Однако на практике даже на персональных компьютерах пользователей с ОС общего назначения (к которым можно отнести рабочие станции с ОС Windows, Linux, Unix, OS X; мобильные устройства с ОС Android и iOS и т.д.) выполняется множество процессов, многие из которых ожидают приема данных по сети, и, следовательно, потенциально уязвимы для атак. Поэтому в последнее время считается целесообразным задействовать МЭ даже на узлах СПД, находящихся в локальной сети, для того, чтобы защищаться не только от атак снаружи сети, но и от атак, которые могут исходить от других узлов внутри периметра сети. Если ЛВС разбита на сегменты, МЭ могут быть размещены на границе этих сегментов.

1.2. Краткие сведения о компьютерных сетях

Международная распределенная сеть Интернет использует стек протоколов TCP/IP для объединения компьютерных ресурсов всей планеты. Достаточно часто эти протоколы используются также в частных и коммерческих сетях. Стек TCP/IP включает множество протоколов.

Ниже описаны только наиболее значимые для настоящего пособия, по мнению авторов, протоколы: IP, TCP, UDP, ICMP, — и технология трансляции сетевых адресов. Место указанных протоколов в различных моделях, включая модель взаимодействия открытых систем OSI (Open Systems Interconnection), показано на рис. 1.1.

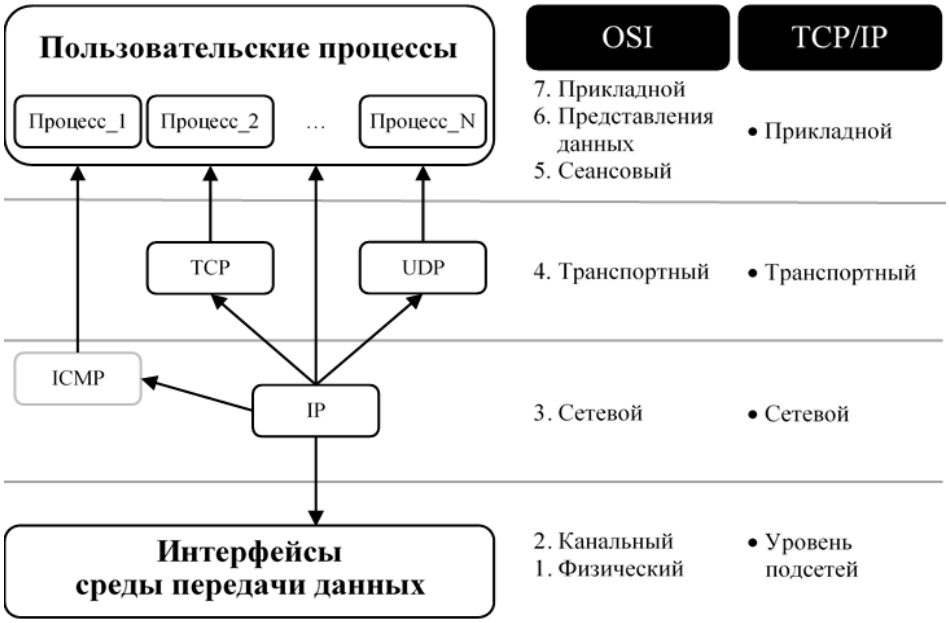


Рисунок 1.1. Стек TCP/IP в эталонной модели OSI

NAT (*Network Address Translation*) — это технология трансляции сетевых адресов, в которой продвижение пакета во внешней сети происходит с помощью адресов, отличающихся от используемых во внутренней сети. Это требуется, например, в случае недостатка IP-адресов или используется из соображений безопасности для скрытия узлов внутренней сети.

Технология NAT существует в нескольких вариантах, наиболее распространенной является традиционная технология. Она позволяет при организации сеансов связи, исходящих из частной сети, получать доступ к узлам внешних сетей. При этом на маршрутизаторе, который обеспечивает связь сетей, должно быть установлено программное обеспечение NAT. Оно выполняет динамическое отображение набора частных адресов на набор глобальных адресов, получаемых от поставщика услуг Интернета. Можно настроить NAT таким образом, что для внешнего мира будет существовать только один IP-адрес внутренней сети. NAT можно использовать для перенаправления трафика, а также для объединения двух локальных

сетей и балансировки нагрузки между одинаковыми серверами с разными IP-адресами.

ICMP (*Internet Control Message Protocol*) — это вспомогательный межсетевой протокол управляющих сообщений, реализующий мониторинг и диагностику сети. С его помощью передаются сообщения об ошибках и нештатных ситуациях, которые возникают в процессе передачи данных (недоступность запрашиваемой услуги, отсутствие ответа маршрутизатора).

Протокол ICMP в некотором смысле компенсирует ненадежность протокола IP, оповещая отправителя о возникших с пакетами проблемах. Протокол ICMP генерирует диагностические сообщения перед уничтожением пакета, если его дальнейшая передача невозможна. ICMP-сообщения инкапсулируются в поле данных IP-пактов. Протокол ICMP формирует сообщения единого формата, но различного типа в зависимости от решаемых задач.

TCP (*Transmission Control Protocol*) — это протокол транспортного уровня, обеспечивающий передачу потока данных с обязательным предварительным соединением, которое гарантирует, что передаваемые с помощью протокола байты потока данных будут доставлены получателю в целостности.

Особенностью протокола TCP является то, что он работает с неструктурированным потоком байтов, получаемых от протоколов высшего уровня. На нижний (сетевой) уровень отправляется часть (сегмент) потока байтов размером не более 64 Кбайт. Некоторые особенности этого протокола вызывают определенные проблемы при передаче:

– из-за фрагментации часть сегмента может оказаться утраченной;

– порядок поступления сегментов в узел назначения может быть произвольным;

– возможна задержка сегментов, превышающая интервал ожидания у отправителя, который может передать их повторно, что может потребовать сложной процедуры обработки для восстановления сегмента.

Если тот или иной пакет данных в процессе передачи будет потерян или искажен, происходит его повторная передача. Это обеспечивается тем, что со стороны получателя отправляется подтверждение полного и корректного получения байтов. До получения отправителем такого подтверждения следующий пакет не будет отправлен. Кроме того, протокол TCP предотвращает дублирование пакетов.

При организации связи между парой прикладных процессов протокол TCP обеспечивает надежную передачу и управление потоком данных.

UDP (*User Datagram Protocol*) — это протокол транспортного уровня, который обеспечивает передачу данных без установления соединений и

без гарантии доставки пакетов (дейтаграмм). В результате скорость передачи повышается, так как всеми узлами тратится меньше времени и ресурсов на работу механизмов обеспечения гарантированной доставки пакетов, однако надежность доставки снижается.

Чаще всего протокол UDP применяется при выполнении приложений «запрос-ответ». При этом UDP не обеспечивает контроль потока и ошибок, а также повторную передачу дейтаграмм.

Часто этот протокол используется для передачи потоковых данных (видео- и аудиоданных), поскольку в этом случае проще сбросить не пришедшие вовремя пакеты, чем дожидаться их получения (например, в системах реального времени) и потеря части пакетов является допустимой для данных такого типа.

1.3. Краткие сведения об ОС Linux

ОС Linux является Unix-подобной операционной системой на базе ядра Linux (kernel.org). ОС Linux работает на множестве аппаратных архитектур, включая x86/x86_64, ARM, PowerPC и MIPS. ОС Linux доступна пользователям в форме дистрибутивов, которые представляют собой наборы ядра, системных программ и прикладных программных пакетов. Дистрибутивы отличаются друг от друга составом этих наборов, необходимостью сборки из исходных кодов, региональной спецификой. Наиболее популярные дистрибутивы в настоящий момент: Linux Mint, Ubuntu, Debian, ArchLinux.

Ядро Linux монолитное, с возможностью загрузки дополнительных модулей. Начиная с версии ядра 2.3 поддерживается МЭ Netfilter. Большая часть административной работы в ОС Linux выполняется в терминале — интерфейсе командной строки (рис. 1.2).

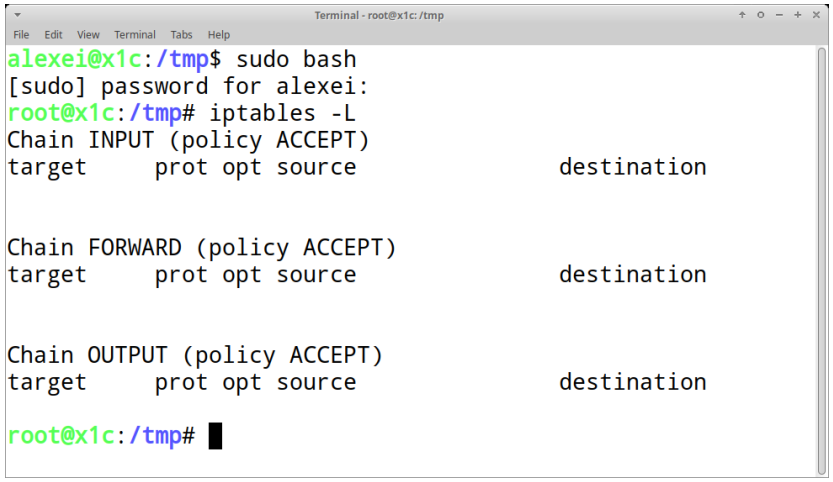


Рисунок 1.2. Терминал в ОС Linux

Для установки модулей системы Netfilter в конкретном дистрибутиве ОС Linux необходимо воспользоваться пакетным менеджером. В разных дистрибутивах пакетные менеджеры разные:

- в дистрибутивах, основанных на Debian: dpkg и apt (aptitude);
- в дистрибутивах RedHat — rpm;
- в Gentoo — Portage;
- в ArchLinux — Pacman и т.д.

Справочная система в ОС Linux состоит из множества компонентов, в терминале доступна справка с помощью команд `man` и `info`. Например, `man iptables` позволит вывести краткую справку по утилите `iptables` (для выхода нужно нажать <q>).

Некоторые команды для выполнения требуют прав суперпользователя (root):

```
$ iptables -L
iptables v1.6.1: can't initialize iptables table `filter': Permission denied (you must be root)
```

В большинстве дистрибутивов можно запустить их с помощью `sudo`:

```
$ sudo iptables -L
```

Если требуется вводить большое количество команд, требующих административных привилегий, можно запустить командный интерпретатор с правами root:

```
$ sudo bash
```

Для просмотра сетевых интерфейсов и сетевых настроек можно воспользоваться либо командой `ifconfig`, либо командой `ip addr`:

```
# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
            inet 127.0.0.1/8 scope host lo
                    valid_lft forever preferred_lft forever
```

Для просмотра соединений можно воспользоваться командой `netstat`:

```
# netstat -4plunt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:873   0.0.0.0:* LISTEN 4520/rsync
tcp 0 0 0.0.0.0:2605  0.0.0.0:* LISTEN 4279/bgpd
tcp 0 0 0.0.0.0:12783 0.0.0.0:* LISTEN 4661/sshd
tcp 0 0 0.0.0.0:8080  0.0.0.0:* LISTEN 4395/nginx: master
tcp 0 0 0.0.0.0:10050 0.0.0.0:* LISTEN 1578/zabbix_agentd
udp 0 0 109.239.132.98:123 0.0.0.0:* 4431/ntpd
```

Для проверки работы МЭ часто бывает необходимо сымитировать обращение на заданный IP-адрес и порт. Для этого проще всего использовать утилиту telnet. Для того, чтобы установить соединение с приложением, слушающим на порту 8080, на хосте с адресом 10.12.0.1:

```
$ telnet 10.12.0.1 8080
```

Также для этой цели можно использовать утилиту `netcat` (`nc`). Гораздо больше возможностей предоставляет утилита `hping3` (в том числе возможность имитировать простые DoS-атаки).

Для просмотра и изменения некоторых настроек ОС можно использовать утилиту `sysctl`. Например, для того чтобы разрешить на локальном хосте осуществлять транзит пакетов между интерфейсами (что необходимо для работы хоста в качестве маршрутизатора), следует выполнить команду

```
# sysctl -w net.ipv4.ip_forward=1
```

Для просмотра загруженных модулей ядра можно воспользоваться командой `lsmod`. Например, для того чтобы убедиться, что модуль ipt_NETFLOW загружен, можно выполнить следующую команду:

```
# lsmod | grep ipt_
ipt_NETFLOW 53772 1
```

Вопросы к разделу 1

1. Перечислите основные возможности межсетевых экранов.
2. Чем стек TCP/IP отличается от стека OSI?
3. Для чего нужны IP-адреса? Каких типов они бывают?
4. Для чего нужны MAC-адреса?
5. Что такое порт?
6. Для чего нужна трансляция сетевых адресов (NAT)?
7. Чем протокол UDP отличается от протокола TCP?
8. Для чего необходим протокол ICMP?
9. С помощью какой команды в ОС Linux можно узнать, какие IP-адреса присвоены сетевым интерфейсам?
10. Как в ОС Linux разрешить пропуск транзитного трафика?

Раздел 2.
Архитектура системы Netfilter

2.1. Основные компоненты системы Netfilter

МЭ в ОС Linux представляет собой совокупность программных компонентов в ядре ОС и набора утилит, которые работают в пространстве пользователя.

Принцип работы МЭ заключается в том, что при прохождении пакетом сетевого стека ОС в определенных точках, которые в Netfilter называются *ловушками* (*hooks*), он подвергается анализу и обработке в соответствии с *правилами* (*rules*), заданными администратором системы. В каждой ловушке может быть зарегистрировано несколько функций-обработчиков, которые вызываются в порядке, определенным специальным параметром, задающим приоритет. Существует пять основных ловушек:

– *NF_IP_PRE_ROUTING*. Компоненты ядра, зарегистрировавшие обработчик в настоящей ловушке, получают уведомление о пакете сразу, как только пакет принимается сетевым интерфейсом. Обработчики срабатывают до того, как принимается решение о маршрутизации пакета;

– *NF_IP_LOCAL_IN*. Настоящая ловушка позволяет компонентам ядра зарегистрировать обработчик и получать уведомления о пакетах, которые предназначены локальному хосту, то есть программам, работающим на нем (локальным приложениям). Обработка происходит после того, как принято решение о маршрутизации пакета;

– *NF_IP_FORWARD*. Настоящая ловушка позволяет компонентам ядра зарегистрировать обработчик и получать уведомления о пакетах, которые поступили локальному хосту, но предназначены другому хосту (это возможно, если локальный хост выступает в качестве маршрутизатора, то есть имеет несколько сетевых интерфейсов и передает пакеты между сетями). Обработка происходит после того, как принято решение о маршрутизации пакета;

– *NF_IP_LOCAL_OUT*. Компоненты ядра, зарегистрировавшие обработчик в настоящей ловушке, получают уведомление о пакете сразу, как только пакет поступает в сетевой стек локального хоста от локального приложения, то есть отправляется в сеть локальной программой. Обработчики срабатывают до того, как принимается решение о маршрутизации пакета;

– *NF_IP_POST_ROUTING*. Настоящая ловушка позволяет компонентам ядра зарегистрировать обработчик и получать уведомления об исходящих пакетах (сгенерированных локальными приложениями или проходящими через хост) перед тем, как они будут отправлены в сеть. Обработка происходит после того, как принято решение о маршрутизации пакета.

Правила обработки пакетов объединяются в *цепочки* (*chains*), которые в Netfilter могут быть двух видов:

– *базовые*, или *встроенные* (*built-in*), которые существуют в системе по умолчанию и не могут быть удалены;

– *пользовательские* (*user-defined*), которые администратор может создавать и удалять.

Цепочки размещаются в *таблицах* (*tables*), которые можно условно считать компонентами ядра, регистрирующимися в ловушках. Таблицы объединяют правила по смыслу, то есть по типу операций, выполняемых с пакетом. Например, правила, которые связаны с трансляцией сетевых адресов, могут быть размещены только в таблице nat, а правила, с помощью которых принимается решение, пропускать пакет дальше или нет, могут быть размещены в таблице filter. Упрощенно говоря, от того, в какую таблицу добавлено правило, зависит, что будет сделано с пакетом, а от того, в какую цепочку — когда. Базовые цепочки присутствуют в нескольких таблицах.

Существуют следующие базовые цепочки (обратите внимание, что имена цепочек записываются в верхнем регистре):

– *PREROUTING*. Правила срабатывают при вызове обработчика, зарегистрированного в ловушке NF_IP_PRE_ROUTING;

– *INPUT*. Правила срабатывают при вызове обработчика, зарегистрированного в ловушке NF_IP_LOCAL_IN;

– *FORWARD*. Правила срабатывают при вызове обработчика, зарегистрированного в ловушке NF_IP_FORWARD;

– *OUTPUT*. Правила срабатывают при вызове обработчика, зарегистрированного в ловушке NF_IP_LOCAL_OUT;

– *POSTROUTING*. Правила срабатывают при вызове обработчика, зарегистрированного в ловушке NF_IP_POST_ROUTING.

Пользовательские цепочки создаются администратором и могут присутствовать только в одной таблице. Таким цепочкам рекомендуется давать имена, с помощью которых их легко отличить от базовых цепочек (если в верхнем регистре, то с определенным префиксом, например, USER_MARK или USER_CHECK). С помощью специальных правил переходов пользовательские цепочки встраиваются в базовые цепочки. Базовые цепочки также имеют *действие по умолчанию* (*default policy*), которое применяется к пакетам, которые не были обработаны ни одним правилом в базовой цепочке или правилами в пользовательских цепочках, вызванных из базовой.

Таблицы хранят правила базовых и пользовательских цепочек, объединенных общим функциональным назначением. Существуют следующие таблицы (обратите внимание, что имена таблиц записываются в нижнем регистре):

– *filter* — таблица, предназначенная для размещения правил, с помощью которых выполняются проверки атрибутов пакета и его принадлежности к соединению и принятие решения относительно дальнейшего продвижения пакета;

– *nat* — таблица, предназначенная для размещения правил, управляющих процессом трансляции сетевых адресов;

– *mangle* — таблица, предназначенная для размещения правил, модифицирующих пакет или его атрибуты (например, какие-либо поля заголовка IP-пакета), а также позволяющих добавить специальную метку (mark) к пакету, с помощью которой его можно идентифицировать в других таблицах или компонентах ядра;

– *raw* — таблица, предназначенная для размещения правил, позволяющих выполнить некоторые операции с пакетом в обход системы отслеживания соединений (conntrack);

– *rawpost* — таблица, предназначенная для размещения правил подмены адреса источника в обход системы отслеживания соединений. Настоящая таблица становится доступной после установки пакета xtables-addons.

– *security* — таблица, предназначенная для размещения правил, позволяющих установить отметку контекста безопасности на пакеты или соединения для дальнейшего использования в рамках подсистемы SELinux. Таблица становится доступной после установки пакета SELinux.

Сетевой пакет, в зависимости от того, предназначен ли он локальному хосту, сгенерирован им или является транзитным, проходит через определенный набор базовых цепочек различных таблиц. При прохождении пакетом цепочки к нему последовательно применяются все правила этой цепочки до тех пор, пока не найдется правило, под действие которого пакет подпадет, или не будет достигнут конец базовой цепочки в текущей таблице. В последнем случае к пакету применяется действие по умолчанию.

2.2. Порядок прохождения пакетов в системе Netfilter

Порядок прохождения пакетов по цепочкам следующий. Пакеты, предназначенные локальному хосту, проходят сначала правила цепочек PREROUTING в таблицах raw, mangle и nat, затем правила цепочек INPUT в таблицах mangle, filter и security. Транзитные пакеты сначала проходят правила цепочек PREROUTING в таблицах raw, mangle и nat, затем правила цепочек FORWARD в таблицах mangle, filter и security, после чего правила цепочек POSTROUTING в таблицах mangle и nat. Исходящие пакеты (сгенерированные приложениями, работающими на локальном хосте) сначала проходят правила цепочек OUTPUT в таблицах raw/rawpost, mangle, nat, filter и security, а затем правила цепочек POSTROUTING в таблицах mangle и nat. Порядок прохождения пакетов схематически изображен на рис. 2.

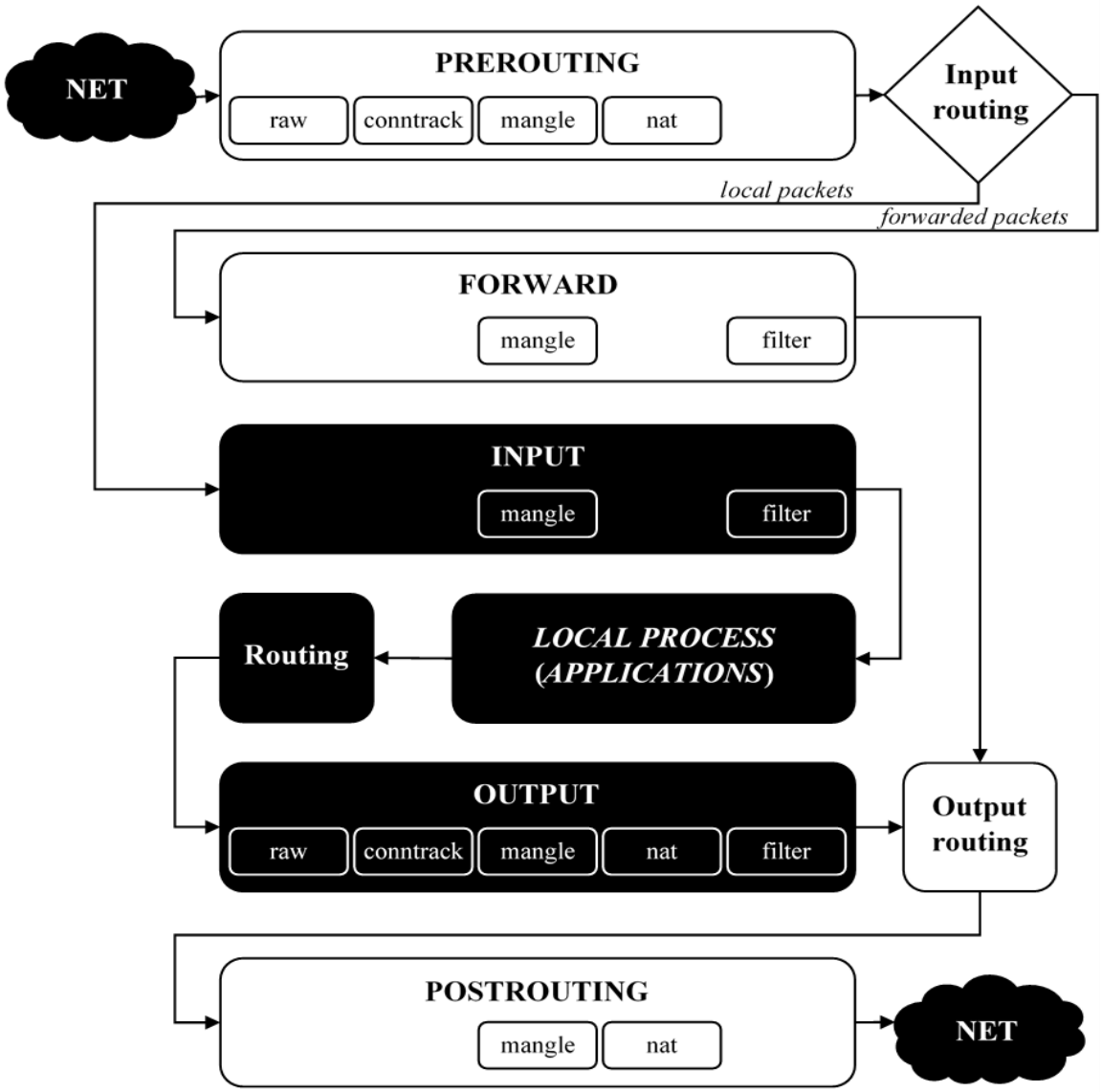


Рисунок 2. Порядок прохождения пакетов
по цепочкам и таблицам Netfilter

Что представляют собой правила, управляющие поведением МЭ? Каждое правило состоит из *критериев* и *действия*. Если пакет соответствует *критерию* (*criterion*), то выполняется указанное в правиле *действие*, или *цель* (*target*). Будем говорить, что такая ситуация соответствует *применению*, или *срабатыванию*, правила. Критерий представляет собой некоторое условие (предикат) или совокупность условий, с помощью которых проверяются атрибуты пакета и соединения и делается вывод о соответствии пакета текущему правилу. Если критерий не задан, то под действие правила подпадают все проходящие по рассматриваемой цепочке пакеты. Критериев в каждом правиле может быть указано несколько (при этом они объединяются неявной конъюнкцией). В случае, если правило сработало, увеличиваются *счетчики* пакетов и байтов (*counters*) для этого правила.

Действием, или *целью*, называется некоторая операция с пакетом, которая приводит либо к изменению порядка обработки настоящего пакета

(например, к прекращению проверок и пропуску пакета или передаче пакета в пользовательскую цепочку), либо к изменению его атрибутов, либо к его маркировке, либо к некоторому побочному эффекту (например, логированию пакета).

Некоторые действия являются *завершающими*, или *терминальными*. Это означает, что пакет перестает обрабатываться правилами МЭ в рассматриваемой базовой цепочке (тем не менее, это не значит, что пакет прекращает свое прохождение по сетевому стеку; например, к такому эффекту приводит действие ACCEPT).

Другие действия можно назвать *нетерминальными*. Это означает, что после выполнения операции с пакетом он проверяется следующим правилом в цепочке (так, например, работает действие MARK). Действие также может быть не задано. В этом случае правило считается примененным, и просто выполняется наращивание счетчиков.

В состав МЭ в ОС Linux входят следующие компоненты ядра:

– *ip_tables* и *ip*6*_tables* — данные модули обеспечивают основную функциональность МЭ, в том числе трансляцию сетевых адресов (для IPv4) и фильтрацию пакетов;

– *arp_tables* — модуль фильтрации для протоколов ARP/RARP;

– *ebtables* — модуль фильтрации канального уровня, выполняющий проверку и модификацию Ethernet-кадров;

– *x_tables* — общий модуль поддержки операций МЭ;

– *nf_conntrack* — модуль поддержки отслеживания соединений и классификации пакетов.

Для управления МЭ существуют различные пользовательские утилиты (все они для выполнения требуют наличия прав администратора системы):

– *iptables* — утилита настройки фильтрации на сетевом и транспортном уровне (для стека IPv4);

– *iptables-save* и *iptables-restore* — утилиты для сохранения и восстановления правил фильтрации на сетевом и транспортном уровнях (для стека IPv4);

– *ip*6*tables* — утилита настройки фильтрации на сетевом и транспортном уровне (для стека IPv6);

– *ip6tables-save* и *ip6tables-restore* — утилиты для сохранения и восстановления правил фильтрации на сетевом и транспортном уровнях (для стека IPv6);

– *arptables* — утилита настройки фильтрации протоколов ARP и RARP;

– *arptables-save* и *arptables-restore* — утилиты для сохранения и восстановления правил фильтрации протоколов ARP и RARP;

– *ebtables* — утилита настройки фильтрации на канальном уровне (для сетевых мостов);

– *ebtables-save* и *ebtables-restore* — утилиты для сохранения и восстановления правил фильтрации на канальном уровне (для сетевых мостов);

– *ipset* — утилита для манипулирования списками IP-адресов и подсетей для обеспечения более эффективной работы правил МЭ;

– *conntrack* — утилита управления таблицами состояния соединений и другими функциями подсистемы conntrack.

Вопросы к разделу 2

1. Перечислите основные ловушки, используемые МЭ Netfilter. Как они связаны с таблицами и цепочками?
2. Чем пользовательские цепочки отличаются от встроенных?
3. Есть ли у пользовательских цепочек действие по умолчанию?
4. Перечислите основные таблицы МЭ Netfilter.
5. Опишите порядок прохождения пакетов по цепочкам и таблицам Netfilter.
6. Приведите примеры терминальных и нетерминальных действий.
7. Какие основные компоненты ядра входят в состав МЭ Netfilter?
8. Поддерживает ли Netfilter отслеживание соединений?
9. Для чего нужны утилиты iptables-save и iptables-restore?
10. Что сделает команда “`iptables-save | grep PREROUTING`”?

Раздел 3.
Фильтрация на сетевом и транспортном уровнях

3.1. Общие сведения

Основная настройка МЭ осуществляется с помощью утилит iptables и ip6tables. В дальнейшем будет рассмотрена в основном функциональность iptables, так как с точки зрения конфигурирования МЭ отличий не так уж много. Детальное рассмотрение отличий протоколов IPv4 и IPv6 выходит за рамки настоящего пособия и может быть найдено в других работах.

В общем виде синтаксис команды iptables может быть определен следующим образом:

```
iptables [таблица] команда [цепочка [параметры]]
```

Квадратные скобки означают необязательность компонента, то есть, например, имя таблицы можно не указывать (в этом случае подразумевается работа с таблицей filter). При каждом вызове должна быть задана команда и в зависимости от нее цепочка, с которой необходимо работать, и дополнительные параметры, собственно определяющие критерии, действия и другую необходимую информацию. Команда в сокращенной форме обозначается заглавной буквой (например, `-L`), но может задаваться и в полной форме (например, `--list`; обратите внимание на два символа дефиса).

Для указания имени таблицы используется опция `-t`, например, `-t nat`. Вывести список правил можно с помощью команды `-L` (`--list`). Следующая команда выводит список правил таблицы filter, поскольку явно таблица не задается:

```
# iptables -L
Chain INPUT (policy ACCEPT)
target   prot opt source   destination

Chain FORWARD (policy ACCEPT)
target   prot opt source   destination

Chain OUTPUT (policy ACCEPT)
target   prot opt source   destination
```

Здесь и далее «машинным» шрифтом показан ввод в консоли, а символ “#” в приглашении командной строки показывает, что команда выполняется с правами администратора системы. В указанном случае можно видеть, что все цепочки таблицы filter не содержат правил.

Ключ `-n` заставляет выводить вместо символических имен хостов IP-адреса, `-v` — включить в вывод информацию о счетчиках, а

--line numbers — пронумеровать правила (номера правил используются в других командах; правила нумеруются с единицы):

```
# iptables -L -t raw -nv --line-numbers
Chain PREROUTING (policy ACCEPT 2612G packets, 3028T bytes)
num   pkts bytes target   prot opt in out source    destination
1 176G 311T CT all  --  * * 0.0.0.0/0  84.130.160.64/27  CT
notrack
2 205G 296T CT all  --  * * 0.0.0.0/0  0.0.0.0/0 match-set
notrack dst CT notrack
3 10G 526G CT tcp   --  * * 0.0.0.0/0  84.130.160.0/21 tcp
flags:0x17/0x02 CT notrack

Chain OUTPUT (policy ACCEPT 13G packets, 28T bytes)
Num pkts bytes target prot opt in out source  destination
```

Команда -A (--append) позволяет добавить новое правило в конец заданной цепочки в заданной таблице, команда -I (--insert) позволяет вставить правило в заданную цепочку заданной таблицы на заданную номером позицию (остальные правила сдвигаются вниз; если же номер не задан, правило добавляется в начало цепочки), команда -R (--replace) позволяет заменить правило с заданным номером, -D (--delete) позволяет удалить правило либо с заданным номером, либо с заданной спецификацией (то есть, например, можно при вызове iptables заменить -A на -D и соответствующее правило будет удалено), команда -F (--flush) удаляет все правила из заданной цепочки, либо правила из всех цепочек заданной таблицы (если цепочка не указана).

Команда -S (--list-rules) подобно команде -L выводит список правил, но в формате, пригодном для использования с утилитой iptables-restore:

```
# iptables -S -t raw
-P PREROUTING ACCEPT
-P OUTPUT ACCEPT
-A PREROUTING -d 94.230.160.64/27 -m comment --comment "notrack
incoming connections to nat users" -j CT --notrack
-A PREROUTING -m set --match-set notrack dst -m comment
--comment "notrack incoming connections to users with external
ip" -j CT --notrack
-A PREROUTING -d 94.230.160.0/21 -p tcp -m tcp --tcp-flags
FIN,SYN,RST,ACK SYN -j CT -notrack
```

С помощью команды -C (--check) можно проверить, существует ли правило с заданной спецификацией в заданной цепочке заданной таблицы. В случае успешности проверки iptables передает ОС код возврата 0 (узнать код возврата предыдущей команды можно, например, с помощью команды “echo $?”). Команда -Z (--zero) позволяет обнулить счетчики для заданного по номеру правила, либо для всех правил в цепочке, либо для правил во всех цепочках заданной таблицы.

Для того чтобы задать действие по умолчанию, в базовой таблице используется команда `-P` (`--policy`). Например, для того чтобы в качестве действия по умолчанию в цепочке `INPUT` таблицы filter отбрасывать пакеты, нужно выполнить следующую команду:

```
# iptables -P INPUT DROP
```

Для манипулирования пользовательскими цепочками существуют три команды: `-N` (`--new-chain`) позволяет создать новую цепочку, `-X` (`--delete-chain`) позволяет удалить цепочку с заданным именем, `-E` (`--rename-chain`) позволяет переименовать цепочку. Удалить можно только цепочку, не содержащую правил, и на которую нет ссылок из других цепочек.

3.2. Действия в таблице filter

Таблица filter предназначена для пропуска или запрета соединений и пакетов, то есть для того, что называется фильтрацией трафика. Настоящая таблица содержит цепочки `INPUT`, `FORWARD` и `OUTPUT`. Обычными действиями, которые применяются правилами в указанной таблице, являются `ACCEPT`, `DROP` и `REJECT`.

В том случае, если пакет надо пропустить (то есть прекратить его обработку в текущей базовой цепочке), применяется действие `ACCEPT`. В этом случае пакет в соответствии с алгоритмом обработки пакетов, описанным в главе 2, может продолжить свой путь по правилам других базовых цепочек в других таблицах:

```
# iptables -P INPUT DROP
# iptables -A INPUT -s 84.103.20.1 -j ACCEPT
```

В этом случае приложениям на локальном хосте запрещается получать любые пакеты, кроме тех, которые исходят от хоста 84.103.20.1. Действие `ACCEPT` может быть применено и в пользовательской таблице. В этом случае пакет точно так же покинет базовую таблицу (что отличается от действия `RETURN`, см. далее).

В том случае, когда требуется заблокировать пакет, применяется действие `DROP`. В этом случае пакет прекращает свое прохождение по правилам цепочки, а также по сетевому стеку хоста и отбрасывается.

```
# iptables -P INPUT ACCEPT
# iptables -A INPUT -s 84.103.20.1 -j DROP
```

Здесь приложениям на локальном хосте запрещается получать пакеты от хоста 84.103.20.1. Пакеты от любых других хостов не отбрасываются.

Действие `REJECT` похоже на `DROP`, но отличается от него тем, что источнику отброшенного пакета отправляется уведомление по протоколу ICMP об отказе в доставке пакета. По умолчанию об отказе сообщается

отправкой ICMP-пакета типа “icmp-port-unreachable”. При необходимости отправить ICMP-ответ другого типа можно воспользоваться опцией --reject-with:

```
# iptables -A INPUT -s 84.103.20.1 -j REJECT --reject-with
icmp-host-prohibited
```

Допустимые типы ICMP-ответов: “icmp-net-unreachable”, “icmp-host-unreachable”, “icmp-proto-unreachable”, “icmp-net-prohibited”, “icmp-host-prohibited”.

В случае, если необходимо прервать TCP-соединение средствами протокола TCP (отправив источнику пакет RST), также следует применить опцию --reject-with, но с другим типом ответа:

```
# iptables -A INPUT -p tcp -s 84.103.20.1 -j REJECT
--reject-with tcp-reset
```

В большинстве случаев при необходимости заблокировать трафик от хоста или сети используют действие DROP, так как отсутствие каких-либо ответов затрудняет процедуру сканирования хоста или сети, которую может проводить злоумышленник.

3.3. Действия в таблице mangle

Таблица mangle предназначена для модификации атрибутов пакетов, а также классификации и маркировки пакетов и соединений. Настоящая таблица содержит цепочки PREROUTING, INPUT, FORWARD, OUTPUT и POSTROUTING. Обычными действиями, которые применяются правилами в указанно таблице, являются TTL (HL для IP6v), TOS, DSCP, TCPMSS, ECN, TCPOPTSTRIP, TPROXY, MARK и CONNMARK. Рассмотрим некоторые из них подробнее.

Действие TTL позволяет задать значение полю TTL (Time To Live) пакета. В соответствии с правилами протокола IPv4 на каждом хопе (при прохождении очередного маршрутизатора) значение поля TTL уменьшается на единицу, и при достижении этим значением нуля пакет уничтожается. Широко известен прием, позволяющий отменить уменьшение значения поля TTL на маршрутизаторе, что приводит к его «прозрачности» для утилит типа traceroute:

```
# iptables -t mangle -I PREROUTING -j TTL --ttl-inc 1
```

Поддерживаются опции --ttl-set (установить поле TTL в заданное значение), --ttl-inc (увеличить значение поля на заданную величину) и --ttl-dec (уменьшить значение поля на заданную величину).

Аналогом действия TTL для IPv6 является действие HL, изменяющее значение поля HL (Hop Limit). Поддерживаются опции --hl-set (установить поле HL в заданное значение), --hl-inc (увеличить значение поля на

заданную величину) и `--hl-dec` (уменьшить значение поля на заданную величину).

Действие `DSCP` с помощью опций `--set-dscp` и `--set-dscp-class` позволяет управлять значением поля DS в заголовке IP-пакета, что может быть необходимо для управления и классификации трафика с целью поддержки механизмов качества обслуживания (QoS). Действие `TOS` с опциями `--set-tos`, `--and-tos`, `--or-tos` и `--xor-tos`, по сути, аналогично и позволяет манипулировать всеми битами поля TOS (а не только младшими шестью, как действие `DSCP`).

Предусмотрена возможность манипуляций с TCP-сегментами. Для этого применяются действия `TCPMSS`, `ECN` и `TCPOPSTRIP`. С помощью действия `TCPMSS` можно установить значение поля MSS (Maximum Segment Size) в исходящих или транзитных TCP сегментах. В тех случаях, когда по каким-либо причинам ICMP-сообщения о необходимости фрагментации не могут быть доставлены, возникает ситуация, известная как “path MTU discovery black hole”. Одним из возможных способов в этой ситуации может быть уменьшение максимального размера TCP-сегмента:

```
# iptables -t mangle -I FORWARD -p tcp -j TCPMSS --set-mss 1400
```

либо его «подгонка» под минимально допустимый MTU с помощью опции `--clamp-mss-to-pmtu`:

```
# iptables -t mangle -I FORWARD -p tcp --tcp-flags SYN,ACK SYN
-j TCPMSS --clamp-mss-to-pmtu
```

Действие `ECN` позволяет обнулить флаги CWR и ECE в TCP-заголовке (и, таким образом, не затрагивает ECN-биты в IP-заголовке), что может быть необходимо в редких случаях решения проблем с работой алгоритма управления насыщением TCP-соединения.

Действие `TCPOPTSTRIP` позволяет удалить заданные опции из заголовка TCP-пакета, что также является достаточно редко используемой возможностью. Например, для удаления опции MSS из TCP-сегментов, уходящих с хоста, можно выполнить команду:

```
# iptables -t mangle -A POSTROUTING -p tcp -j TCPOPTSTRIP
--strip-options mss
```

Действие `MARK` позволяет задать специальную отметку пакету (маркировку), которая не изменяет сам пакет, но позволяет идентифицировать пакет в других подсистемах ядра (например, в подсистеме ограничения скорости соединений). С помощью опций `--set-mark`, `--set-xmark`, `--or-mark`, `--and-mark` и `--xor-mark` можно установить и изменить 32-битную отметку. Целесообразно делать это в цепочке `PREROUTING`. Отметка действует в пределах сетевого стека локального хоста, то есть не может быть передана на другой хост:

```
# iptables -t mangle -A PREROUTING -d 84.130.10.11 -p tcp -m tcp
--dport 80 --tcp-flags FIN,SYN,RST,ACK SYN -j MARK --set-xmark
0x60/0xffffffff
```

Действие CONNMARK позволяет промаркировать соединение. С помощью опций --set-mark, --set-xmark, --or-mark, --and-mark и --xor-mark можно установить 32-битную отметку. Кроме этого, можно перенести отметку из пакета в соединение с помощью опции --save-mark и выполнить обратное действие с помощью опции --restore-mark.

Действие TPROXY позволяет обеспечить поддержку прозрачного проксирования (для работы этого механизма необходим соответствующий прокси-сервер, например, squid или nginx). Подробное описание работы настоящего механизма выходит за рамки настоящего пособия.

Как было указано в главе 2, правила цепочек таблицы mangle пакеты проходят раньше, чем правила цепочек таблиц nat и filter, что позволяет использовать классификацию соединений и постановку маркеров на пакеты в правилах этих таблиц (например, для управления трансляцией адресов).

3.4. Действия в таблице nat

Таблица nat предназначена для операций трансляции сетевых адресов. Настоящая таблица содержит цепочки PREROUTING, OUTPUT, POSTROUTING. Трансляция сетевых адресов имеет смысл, если выполняется на маршрутизаторе, то есть хосте, имеющем несколько интерфейсов, подключенным к разным сетям, и пропускающим трафик из одной сети в другую. Обычными действиями, которые применяются правилами в указанной таблице, являются SNAT, MASQUERADE, DNAT и REDIRECT.

Действие SNAT позволяет подменять адрес источника в уходящих с маршрутизатора транзитных пакетах и гарантирует обратную замену в транзитных пакетах, приходящих в качестве ответов. Его целесообразно применять в цепочке POSTROUTING. Пусть, например, у маршрутизатора есть два интерфейса: eth0 подключен к локальной сети 192.168.11.0/24 и имеет адрес 192.168.11.1, а eth1 подключен к сети 84.103.100.0/30 и имеет внешний (глобально маршрутизируемый) адрес 84.103.100.1 (то есть «смотрит» в Интернет). В этом случае для того, чтобы хосты локальной сети имели доступ к Интернету и могли получать ответы от хостов из Интернета, необходимо организовать трансляцию адресов:

```
# iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to-source
84.103.100.1
```

С помощью опции -o <имя_интерфейса> можно ограничить действие правила заданным сетевым интерфейсом. Действие MASQUERADE работает аналогично действию SNAT, но имеет смысл только в тех случаях, когда адрес на внешнем интерфейсе маршрутизатора не является статическим.

Действие DNAT позволяет подменять адрес назначения в пакетах, приходящих на маршрутизатор, что приводит, по сути, к переадресации, или «пробросу», запросов (в широком смысле), адресованных маршрутизатору, другому хосту. Как правило, «проброс» внутрь локальной сети выполняется только для некоторых портов. Например, если внутри сети работает веб-сервер на порту 8080 хоста 192.168.11.42, который нужно сделать доступным из Интернета, можно организовать трансляцию адресов следующим образом:

```
# iptables -t nat -A PREROUTING -d 84.103.100.1 -p tcp --dport
8080 -j DNAT --to-destination 192.168.11.42
```

При этом, разумеется, нужно удостовериться, что соединения на порт 8080 не запрещены в таблице filter и разрешена передача пакетов по уже установленным и связанным соединениям.

Действие REDIRECT позволяет «перенаправить» соединение на заданный порт локального хоста, то есть заменить номер порта в TCP- или UDP-пакете и заменить адрес назначения на один из собственных адресов (использовать адрес другого хоста нельзя). Например, команда

```
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j RE-
DIRECT --to-port 8080
```

перенаправит пакеты, приходящие из локальной сети и имеющие в качестве порта назначения 80 (HTTP-запросы), на порт 8080 маршрутизатора, на котором может «слушать» прокси-сервер. При этом также нужно удостовериться, что соединения на порты 80 и 8080 не запрещены в таблице filter и разрешена передача пакетов по уже установленным и связанным соединениям.

3.5. Действия в таблицах raw и rawpost

Таблица raw предназначена для выполнения действий с пакетами до принятия решения об их обработке системой отслеживания соединений (conntrack). В этой таблице критерии, связанные с подсистемой отслеживания соединений, не работают. Настоящая таблица содержит цепочки PREROUTING, OUTPUT. Обычными действиями, которые применяются к пакетам, являются NOTRACK, CT и RAWDNAT. Таблица rawpost предназначена для размещения правил изменения адреса источника в обход системы conntrack. Настоящая таблица содержит цепочку POSTROUTING и единственным допустимым действием в правилах этой таблицы является RAWSNAT.

Применение действия NOTRACK к пакету приводит к тому, что он перестает отслеживаться системой conntrack, но продолжает свое прохождение по сетевому стеку и правилам в цепочках других таблиц. Например, если требуется запретить отслеживание пакетов (например, чтобы разгрузить ресурсы маршрутизатора, так как без специальной аппаратной под-

держки система conntrack требует времени центрального процессора и дополнительных объемов памяти на каждое новое соединение), то применяется:

```
# iptables -t raw -A PREROUTING -d 84.130.160.64/27 -j NOTRACK
```

Действие CT представляет собой расширенный вариант действия NOTRACK и позволяет изменять некоторые настройки системы conntrack. Функционально аналогичным действию `NOTRACK` является действие `CT` с ключом `--no-track`:

```
# iptables -t raw -A PREROUTING -d 84.130.160.64/27 -j CT
--no-track
```

С помощью ключа `--helper` можно подключить к обработке соединения один из специальных модулей поддержки (для протоколов FTP, SIP и других). С помощью ключа `--ctevents` можно задать список событий в системе conntrack, которые будет порождать пакет (как правило, опять же с целью уменьшения потребляемых вычислительных ресурсов).

Действия `RAWDNAT`/`RAWSNAT` позволяют выполнить трансляцию сетевых адресов и портов без учета состояний соединений, что при большом количестве соединений (например, на загруженном маршрутизаторе) позволяет существенно экономить ресурсы. Например, комбинация правил:

```
# iptables -t raw -A PREROUTING -d 84.130.100.64 -j RAWDNAT
--to-destination 10.12.0.0/21
# iptables -t rawpost -A POSTROUTING -s 10.12.0.0/21 -j RAWSNAT
--to-source 84.130.100.64
```

функционально эквивалентна следующему правилу

```
# iptables -t nat -A POSTROUTING -s 10.12.0.0/21 -j SNAT --to-
source 84.130.100.64
```

Однако на практике такой подход применяется редко, так как зачастую без использования возможностей (критериев), предоставляемых системой отслеживания соединений, обойтись не получается.

3.6. Пользовательские цепочки и переходы

О создании и удалении пользовательских цепочек было сказано ранее. Необходимость в пользовательских цепочках возникает в случаях, когда правил много (особенно, если они генерируются какими-либо вспомогательными программами) и необходимо поддерживать структуру правил в виде, пригодном для чтения и анализа человеком. Кроме этого, возможность организовать условные и безусловные переходы (jumps) позволяет превращать последовательно интерпретируемый набор правил в некоторое подобие программы (возможны даже циклы, хотя этого следует избегать, так как синтаксис iptables не может претендовать на удобство использования даже в качестве специализированного языка программирования).

Для перехода из одной цепочки в другую используется действие `-j <ИМЯ_ЦЕПОЧКИ>` (`--jump <ИМЯ_ЦЕПОЧКИ>`). Переходы в цепочки других таблиц невозможны. Переходы в произвольные базовые цепочки той же таблицы также невозможны. Внутри пользовательской таблицы правила проверяются последовательно, но этот процесс можно прервать с помощью действия `RETURN`, которое возвращает управление в «вызвавшую» таблицу. Действие `RETURN` может быть применено и в базовой цепочке, в этом случае просто выполняется действие по умолчанию.

Для перехода к пользовательской цепочке также может быть использовано действие `-g <ИМЯ_ЦЕПОЧКИ>` (`--goto ИМЯ_ЦЕПОЧКИ`). При использовании действия `RETURN` в той цепочке, переход на которую был сделан с помощью `-g`, возврата в эту цепочку не произойдет, а «управление» вернется в ту цепочку, откуда переход был сделан с помощью `-j` (либо выполнится действие по умолчанию, если был выполнен возврат в базовую цепочку).

Рассмотрим пример настройки правил для маршрутизатора, ограничивающего доступ из пользовательской сети (подключенных к интерфейсу eth0) в Интернет:

```
# iptable -S
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT ACCEPT
-N USER_ACCESS
-N USER_CHECK
-A INPUT -i lo -j ACCEPT
-A FORWARD -j USER_CHECK
-A FORWARD -j NETFLOW
-A FORWARD -m set --match-set allowed_networks src -j ACCEPT
-A FORWARD -m set --match-set out src -m set --match-set banlist
dst -j DROP
-A USER_CHECK -o eth0 -j USER_ACCESS
-A USER_CHECK -i eth0 -j USER_ACCESS
-A USER_CHECK -o eth0 -j RETURN
-A USER_CHECK -i eth0 -j RETURN
-A USER_CHECK -j ACCEPT
-A USER_ACCESS -m set --match-set allowed_users dst -j RETURN
-A USER_ACCESS -m set --match-set allowed_users src -j RETURN
-A USER_ACCESS -d 84.130.160.0/23 -j RETURN
-A USER_ACCESS -s 84.130.160.0/23 -j RETURN
-A USER_ACCESS -j DROP
```

Сначала устанавливаются политики по умолчанию для базовых цепочек таблицы filter. Далее создаются две пользовательских цепочки: `USER_ACCESS` и `USER_CHECK`. Первая из них содержит правила, определяющие, имеет ли право транзитный трафик проходить через маршрутиза-

тор. Вторая является вспомогательной цепочкой, из которой вызывается первая. В цепочке FORWARD первое правило отправляет пакеты на проверку в цепочку USER_CHECK. Далее содержатся еще несколько правил, обрабатывающих пакеты, вернувшиеся из USER_CHECK, то есть прошедшие проверку: учет данных с помощью ipt_netflow и проверка источников на вхождение в наборы IP-адресов (*ipsets*) с именами “allowed_networks” и “out”. Все пакеты, которые не были явно пропущены, удаляются. В цепочке USER_ACCESS делаются проверки относительно того, можно ли пропустить пакеты (первые четыре правила). Иначе пакеты удаляются. Работа с наборами IP-адресов подробнее рассмотрена в разделе 4.

Нужно понимать, что количество критериев в правиле увеличивает время проверки, то есть создает дополнительную вычислительную нагрузку на систему. За счет применения пользовательских цепочек можно сократить количество проверок. Например, пусть есть правила:

```
# iptables -A INPUT -p udp -s 172.16.1.1 -j ACCEPT
# iptables -A INPUT -p udp -s 172.16.1.11 -j ACCEPT
# iptables -A INPUT -p udp -s 172.16.1.111 -j ACCEPT
<...>
```

Каждое правило проверяет протокол и адрес источника. Организовать проверку можно иначе: проверку на протокол сделать один раз в правиле перехода к пользовательской цепочке, а проверять на адреса источника в правилах этой цепочки:

```
# iptables -N USER_UDP_SRC
# iptables -A INPUT -p udp -g USER_UDP_SRC
# iptables -A USER_UDP_SRC -s 172.16.1.1 -j ACCEPT
# iptables -A USER_UDP_SRC -s 172.16.1.11 -j ACCEPT
# iptables -A USER_UDP_SRC -s 172.16.1.111 -j ACCEPT
```

Тем не менее, если количество адресов велико, то гораздо более производительным подходом будет использование наборов IP-адресов.

3.7. Универсальные критерии

Как уже упоминалось, каждое правило, которое задается с помощью iptables, содержит один или несколько критериев. Рассмотрим один из приведенных ранее примеров:

```
# iptables -A FORWARD -m set --match-set out src -m set --match-
set banlist dst -j DROP
```

В этом правиле заданы два критерия (объединенных логическим И) для проверки принадлежности адресов источника и назначения различным спискам адресов: первый критерий “`-m set --match-set out src`” и второй критерий “`-m set --match-set banlist dst`”. Подробнее о

списках адресов и критериях проверки на принадлежность спискам адресов будет рассказано в разделе 4. Сейчас же рассмотрим несколько критериев, которые можно считать универсальными, то есть которые можно указывать почти в любых правилах (исключения, разумеется, есть) и которые не требуют подключения дополнительных модулей.

Критерий `-p`, `--protocol` уже встречался ранее. Как можно понять, он позволяет указать протокол транспортного уровня (например, tcp, udp, icmp или all). Например, команда

```
# iptables -A INPUT -p tcp -j ACCEPT
```

добавит правило, разрешающее принимать любые пакеты по протоколу TCP.

Если необходимо разрешить только пакеты, приходящие на определенный порт, то можно использовать дополнительную опцию рассматриваемого критерия — `--sport`. Команда

```
# iptables -A INPUT -p tcp --sport 5001 -j ACCEPT
```

разрешит принимать TCP-пакеты, отправленные с порта номер 5001 некоторого хоста. Аналогично если необходимо разрешить только пакеты, приходящие на определенный порт, то можно использовать дополнительную опцию `--dport`. Команда

```
# iptables -A INPUT -p tcp --sport 5001 --dport 80 -j ACCEPT
```

разрешит обращаться к 80-му порту на локальном хосте только с 5001-го порта на удаленном хосте.

Задать соответствие правила пакетам, имеющим определенный адрес источника, можно с помощью критерия `-s`, `--src`. Параметрами критерия являются перечисленные через запятую (если надо задать больше одного) IP-адреса или сети в формате CIDR. Например, команда

```
# iptables -A INPUT -s 84.130.110.0/24,84.130.3.1 -p tcp --sport
5001 --dport 80 -j ACCEPT
```

разрешит обращаться к 80-му порту на локальном хосте только с 5001-го порта удаленных хостов из подсети 84.130.110.0/24 или хоста 84.130.3.1.

Задать соответствие правила пакетам, имеющим определенный адрес назначения, можно с помощью критерия `-d`, `--dst`. Параметрами критерия являются перечисленные через запятую (если надо задать больше одного) IP-адреса или сети в формате CIDR. Команда

```
# iptables -A INPUT -s 84.130.110.0/24,84.130.3.1. -d 10.12.0.1
-p tcp --sport 5001 --dport 80 -j ACCEPT
```

позволит разрешить обращаться к 80-му порту на локальном хосте только с 5001-го порта удаленных хостов из подсети 84.130.110.0/24 или хоста 84.130.3.1, при этом один из интерфейсов локального хоста должен иметь адрес 10.12.0.1.

Следует обратить внимание, что выполнение приведенной выше команды фактически приведет к добавлению двух правил в цепочку INPUT:

```
# iptables -L
Chain INPUT (policy ACCEPT)
target   prot opt source  destination
ACCEPT   tcp -- 84.130.110.0/24  10.12.0.1  tcp spt:5001 dpt:80
ACCEPT   tcp -- 84.130.3.1       10.12.0.1  tcp spt:5001 dpt:80

Chain FORWARD (policy ACCEPT)
Target   prot opt source  destination

Chain OUTPUT (policy ACCEPT)
Target   prot opt source  destination
```

С помощью критерия `-i`, `--in-interface` можно ограничить действие правила только заданным входящим сетевым интерфейсом. Например, команда

```
# iptables -A INPUT -i eth0 -s 84.130.110.0/24 -p tcp --dport 80
-j ACCEPT
```

добавит правило, которое сработает только в том случае, когда TCP-пакет на порт 80, исходящий от хостов из подсети 84.130.110.0/24, прибудет на интерфейс eth0. Настоящий критерий допустимо указывать в правилах, которые добавляются в цепочки `PREROUTING`, `INPUT` и `FORWARD`.

С помощью критерия `-o`, `--out-interface` можно ограничить действие правила только заданным исходящим сетевым интерфейсом. Например, команда

```
# iptables -A OUTPUT -o eth1 -d 84.130.42.0/24 -j DROP
```

запретит локальным приложениям отправлять любые пакеты хостам из подсети 84.130.42.0/24 с интерфейса eth1. Этот критерий допустимо указывать в правилах, которые добавляются в цепочки `FORWARD`, `OUTPUT` и `POSTROUTING`.

Многие критерии (не только универсальные) можно инвертировать, поставив перед ними восклицательный знак. Например, команда

```
# iptables -A OUTPUT -o eth1 ! -d 84.130.42.0/24 -j DROP
```

запретит локальным приложениям отправлять с интерфейса eth1 любые пакеты любым хостам, кроме тех, которые входят в подсеть 84.130.42.0/24. Какие критерии поддерживают инвертирование, а какие нет, следует уточнять в документации.

Вопросы к разделу 3

1. Чем опция `-S` команды `iptables` отличается от опции `-L`?
2. Запишите команду, позволяющую удалить четвертое по счету правило из цепочки `PREROUTING` таблицы raw.
3. С помощью какой команды можно задать действие по умолчанию?
4. Чем действие `REJECT` отличается от действия `DROP`?
5. Для чего используется действие `MARK`? В каких таблицах его допустимо использовать?
6. Верно ли, что с помощью действия `TPROXY` можно осуществлять прозрачную трансляцию соединений?
7. Чем действие `SNAT` отличается от действия `DNAT`?
8. В каких таблицах допустимо использовать действие `CT`?
9. Для чего нужны пользовательские цепочки правил?
10. Какие команды необходимо выполнить, чтобы запретить приложениям, работающим на локальном хосте, обращаться в сеть на любые порты, кроме 80?
11. Что сделает команда “`iptables -I INPUT -i lo -j ACCEPT`”? Какой смысл в добавлении этой команды в цепочку `INPUT`?
12. Что сделает команда “`iptables -I INPUT -i eth0 -s 192.168.0.0/16 -j DROP`”? Какой смысл в добавлении этой команды в цепочку `INPUT`?

Раздел 4.
Отслеживание состояния соединений

Как упоминалось ранее, Netfilter является МЭ с отслеживанием (или определением) состояния соединений (*stateful firewall*). При этом под соединением понимается не только TCP-соединение, но вообще любая последовательность пакетов, объединенных некоторыми общими признаками или связанных по смыслу. Модуль ядра nf_conntrack (*connection tracking, отслеживание соединений*) позволяет принимать решения о прохождении пакетов в зависимости от накопленной информации о соединениях.

4.1. Работа системы conntrack и состояния соединений

В системе conntrack МЭ Netfilter соединением в общем случае считается последовательность пакетов некоторого протокола с одними и теми же адресами источника и назначения и одними и теми же портами источника и назначения, промежуток между которыми не превышает некоторого установленного значения (которое для некоторых протоколов можно менять с помощью настроек ядра). Приведем пример просмотра настроек, имеющих отношение к conntrack:

```
# sysctl -a | grep conntrack
net.netfilter.nf_conntrack_acct = 0
net.netfilter.nf_conntrack_buckets = 1048576
net.netfilter.nf_conntrack_checksum = 1
net.netfilter.nf_conntrack_count = 82263
net.netfilter.nf_conntrack_events = 1
net.netfilter.nf_conntrack_expect_max = 4096
net.netfilter.nf_conntrack_frag6_high_thresh = 4194304
net.netfilter.nf_conntrack_frag6_low_thresh = 3145728
net.netfilter.nf_conntrack_frag6_timeout = 60
net.netfilter.nf_conntrack_generic_timeout = 600
net.netfilter.nf_conntrack_helper = 1
net.netfilter.nf_conntrack_icmp_timeout = 30
net.netfilter.nf_conntrack_icmpv6_timeout = 30
net.netfilter.nf_conntrack_log_invalid = 0
net.netfilter.nf_conntrack_max = 1048576
net.netfilter.nf_conntrack_sctp_timeout_closed = 10
net.netfilter.nf_conntrack_sctp_timeout_cookie_echoed = 3
net.netfilter.nf_conntrack_sctp_timeout_cookie_wait = 3
net.netfilter.nf_conntrack_sctp_timeout_established = 432000
net.netfilter.nf_conntrack_sctp_timeout_heartbeat_acked = 210
net.netfilter.nf_conntrack_sctp_timeout_heartbeat_sent = 30
net.netfilter.nf_conntrack_sctp_timeout_shutdown_ack_sent = 3
net.netfilter.nf_conntrack_sctp_timeout_shutdown_recd = 0
```

```
net.netfilter.nf_conntrack_sctp_timeout_shutdown_sent = 0
net.netfilter.nf_conntrack_tcp_be_liberal = 1
net.netfilter.nf_conntrack_tcp_loose = 1
net.netfilter.nf_conntrack_tcp_max_retrans = 3
net.netfilter.nf_conntrack_tcp_timeout_close = 10
net.netfilter.nf_conntrack_tcp_timeout_close_wait = 60
net.netfilter.nf_conntrack_tcp_timeout_established = 1800
net.netfilter.nf_conntrack_tcp_timeout_fin_wait = 120
net.netfilter.nf_conntrack_tcp_timeout_last_ack = 30
net.netfilter.nf_conntrack_tcp_timeout_max_retrans = 300
net.netfilter.nf_conntrack_tcp_timeout_syn_recv = 60
net.netfilter.nf_conntrack_tcp_timeout_syn_sent = 120
net.netfilter.nf_conntrack_tcp_timeout_time_wait = 120
net.netfilter.nf_conntrack_tcp_timeout_unacknowledged = 300
net.netfilter.nf_conntrack_timestamp = 0
net.netfilter.nf_conntrack_udp_timeout = 30
net.netfilter.nf_conntrack_udp_timeout_stream = 180
net.netfilter.nf_conntrack_udplite_timeout = 30
net.netfilter.nf_conntrack_udplite_timeout_stream = 180
net.nf_conntrack_max = 1048576
```

Видно, что помимо общих настроек (таких, как максимальное количество отслеживаемых одновременно соединений `net.nf_conntrack_max`), присутствует большое количество настроек, связанных с конкретными протоколами, в частности, с таймаутами (`net.netfilter.nf_conntrack_<протокол>_timeout_*`).

В специальную таблицу (хеш-таблицу) в модуле nf_conntrack для каждого отслеживаемого соединения добавляются в качестве ключей два кортежа: для прямого поиска (`proto`, `src_ip`, `src_port`, `dst_ip`, `dst_port`) и для обратного поиска (`proto`, `dst_ip`, `dst_port`, `src_ip`, `src_port`). Поиск по этим ключам в таблице возвращает структуру nf_conn, описывающую соединение.

Утилита conntrack из пакета conntrack-tools позволяет просматривать таблицу отслеживаемых соединений и события, связанные с созданием, изменением и удалением соединений. Для того, чтобы наблюдать за событиями, можно запустить утилиту conntrack с ключом `-E`:

```
# conntrack -E
[UPDATE] tcp   6 120 FIN_WAIT src=10.12.44.178
dst=212.59.127.213 sport=60425 dport=443 src=212.59.127.213
dst=84.130.161.69 sport=443 dport=60425 [ASSURED]
[NEW] tcp      6 120 SYN_SENT src=10.12.44.178
dst=212.59.127.213 sport=60426 dport=443 [UNREPLIED]
src=212.59.127.213 dst=84.130.161.69 sport=443 dport=60426
[UPDATE] tcp   6 30 LAST_ACK src=10.12.44.178 dst=212.59.127.213
sport=60425 dport=443 src=212.59.127.213 dst=84.130.161.69
sport=443 dport=60425 [ASSURED]
```

```
[DESTROY] udp  17 src=84.130.163.139 dst=176.212.42.205
sport=6881 dport=6881 [UNREPLIED] src=176.212.42.205
dst=84.130.163.139 sport=6881 dport=6881
<...>
conntrack v1.4.4 (conntrack-tools): 1689 flow events have been
shown.
```

Просмотреть таблицу отслеживаемых соединений можно с помощью команды conntrack `-L`:

```
# conntrack -L
udp      17 8 src=94.230.164.36 dst=5.189.160.21 sport=51413
dport=20221 [UNREPLIED] src=5.189.160.21 dst=94.230.164.36
sport=20221 dport=51413 mark=0 secctx=null use=1
tcp      6 291 ESTABLISHED src=94.230.167.58 dst=183.84.5.205
sport=43367 dport=1885 [UNREPLIED] src=183.84.5.205
dst=94.230.167.58 sport=1885 dport=43367 mark=0 secctx=null
use=1
unknown  50 597 src=94.230.167.85 dst=136.243.136.142
[UNREPLIED] src=136.243.136.142 dst=94.230.167.85 mark=0
secctx=null use=1
<...>
```

Размер таблицы задается параметром `net.netfilter.nf_conntrack_buckets` и в тех случаях, когда таблица переполняется, в системном логе можно обнаружить сообщения вида “`ip_conntrack: table full, dropping packet`”.

Дополнительная информация о работе системы conntrack может быть получена из псевдофайлов `/proc/net/nf_conntrack` и `/proc/net/nf_conntrack_expect`.

Каждый пакет, проходящий по сетевому стеку, с точки зрения системы отслеживания соединений может принадлежать соединению, находящемуся в одном из следующих состояний:

– *NEW* (новое). Поступает пакет, не относящийся ни к одному из существующих соединений, и при этом не являющийся недопустимым в качестве первого пакета в соединении (например, пакет SYN может являться первым пакетом в TCP-соединении, а пакет RST не может), то создается новая запись в таблице conntrack;

– *ESTABLISHED* (установленное). Соединение переходит из состояния NEW в состояние ESTABLISHED в том случае, если зафиксирован «ответ», то есть пакет, проходящий в обратном направлении. Для некоторых протоколов, поддерживающих свое понятие состояния, «ответом» может быть не любой пакет, а только допустимый (например, в случае TCP — пакет SYN,ACK);

– *RELATED* (связанное). Соединение считается связанным с существующим соединением, если оно появляется в результате взаимодействия по существующему соединению. Поясним на примере протокола FTP. Управляющее соединение по этому протоколу устанавливается на порт 21.

Затем в ходе обмена между клиентом и сервером определяется порт, который будет использоваться сервером для другого соединения, по которому будут передаваться данные. Обработчик (helper) системы conntrack для протокола FTP извлекает номер порта данных из проходящего по управляющему соединению пакета (тем самым условно поднимаясь по стеку протоколов до прикладного уровня) и в момент, когда соединение устанавливается, считает его не просто новым соединением, но ассоциированным с уже установленным ранее соединением. Кроме FTP, системой conntrack поддерживаются обработчики для протоколов SIP, H.323, SCTP и некоторых других. Как RELATED будут отмечены также пакеты ICMP, которые возвращаются в качестве ответов на TCP- и UDP-пакеты.

Таким образом, пакет может быть отмечен как NEW (устанавливает новое соединение), ESTABLISHED (проходит по стеку в рамках установленного соединения) или RELATED (проходит по стеку в рамках связанного с некоторым установленным соединения). Кроме этого, допустимы следующие отметки пакетов:

– *INVALID* (недопустимый). Пакет может быть отмечен как INVALID, если он не может быть отнесен ни к одному из существующих соединений в состоянии ESTABLISHED или RELATED, и не может открывать соединение (то есть быть NEW). Также пакет отмечается как IDVALID, если в отношении этого пакета не может быть принято решение о маршрутизации;

– *UNTRACKED* (неотслеживаемый). Как уже упоминалось ранее, в некоторых случаях может быть целесообразно исключить некоторые пакеты из учета системой conntrack. Это делается с помощью действия CT и опции `--no-track`. При прохождении по сетевому стеку такие пакеты имеют отметку UNTRACKED.

Соединения могут также находиться в двух «виртуальных» состояниях:

– *SNAT*. Соединение переходит в указанное состояние, если адрес источника был изменен с помощью действия SNAT. Система conntrack для пакетов, следующих в обратном направлении в рамках такого соединения, выполняет замену адреса назначения на сохраненный ранее оригинальный адрес источника из пакетов, обработанных действием SNAT;

– *DNAT*. Соединение переходит в настоящее состояние, если адрес источника был изменен с помощью действия DNAT. Система conntrack для пакетов, следующих в обратном направлении в рамках такого соединения, выполняет замену адреса назначения на сохраненный ранее оригинальный адрес источника из пакетов, обработанных действием DNAT.

Кроме таблицы conntrack, системой отслеживания соединений используются еще три таблицы. Таблица expect содержит записи об ожидаемых связанных соединениях:

```
# conntrack -L expect
```

```
135 proto=17 src=188.120.242.37 dst=84.130.161.84 sport=0
dport=5060 mask-src=255.255.255.255 mask-dst=255.255.255.255
sport=0 dport=65535 master-src=10.12.160.5 master-
dst=188.120.242.37 sport=5060 dport=5060 PERMANENT class=0 help-
er=sip
107 proto=17 src=193.201.229.35 dst=84.130.167.101 sport=0
dport=5781 mask-src=255.255.255.255 mask-dst=255.255.255.255
sport=0 dport=65535 master-src=94.230.167.101 master-
dst=193.201.229.35 sport=6812 dport=5060 class=1 helper=sip
69 proto=17  src=81.88.86.37 dst=84.130.166.231 sport=0
dport=45063 mask-src=255.255.255.255 mask-dst=255.255.255.255
sport=0 dport=65535 master-src=94.230.166.231 master-
dst=81.88.86.37 sport=45063 dport=5060 PERMANENT,INACTIVE
class=0 helper=sip
<...>
conntrack v1.4.4 (conntrack-tools): 29 expectations have been
shown.
```

Таблица dying в течение непродолжительного времени хранит соединения, которые были по таймауту или вручную удалены из таблицы conntrack:

```
# conntrack -L dying
tcp      6 0 TIME_WAIT src=10.12.42.25 dst=188.242.228.213
sport=2898 dport=21 src=188.242.228.213 dst=94.230.161.69
sport=21 dport=2898 [ASSURED] mark=0 secctx=null helper=ftp
use=1
tcp      6 0 TIME_WAIT src=10.12.42.25 dst=188.242.228.213
sport=2906 dport=21 src=188.242.228.213 dst=94.230.161.69
sport=21 dport=2906 [ASSURED] mark=0 secctx=null helper=ftp
use=1
conntrack v1.4.4 (conntrack-tools): 2 flow entries have been
shown.
```

Таблица unconfirmed хранит записи, которые после прохождения по сетевому стеку и достижению ловушки NF_IP_POST_ROUTING, добавляются в таблицу conntrack. Таблицы dying и unconfirmed используются в основном разработчиками для отладки и не представляют интереса для системных администраторов.

4.2. Критерии состояния соединения

Рассмотрим использование критерия conntrack в правилах iptables. Для того, чтобы выделить пакеты, относящиеся к соединению в заданных состояниях, используется опция `--ctstate`. Например, команды:

```
# iptables -F INPUT
# iptables -P INPUT DROP
# iptalbes -A INPUT -p tcp -m conntrack --ctstate ESTAB-
LISHED,RELATED -j ACCEPT
# iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW
-j ACCEPT
```

позволят принимать соединения от клиентов на порт 80 и пропускать пакеты, относящиеся к установленным и связанным (например, ICMP-ответы с удаленного хоста) соединениям, все остальные пакеты будут отброшены.

Обратите внимание, что приведенная выше комбинация правил — не то же самое, что:

```
# iptables -F INPUT
# iptables -P INPUT DROP
# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

потому что во втором случае как минимум не будет обеспечена возможность нормального обмена по протоколу ICMP. Именно поэтому в большинстве практических случаев администраторы предпочитают в начало цепочки INPUT добавить:

```
# iptables -A INPUT -p icmp -j ACCEPT
```

Уже упоминавшимся классическим примером, когда необходимо разрешить связанные соединения, является настройка МЭ на FTP-сервере:

```
# iptables -F INPUT
# iptables -P INPUT DROP
# iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -
j ACCEPT
# iptables -A INPUT -p tcp --dport 21 -m conntrack --ctstate NEW
-j ACCEPT
```

В таком случае упоминание RELATED важно для обеспечения нормальной работы пассивного режима передачи данных протокола FTP. В управляющем соединении от сервера клиенту будет передан номер порта для передачи данных, и при попытке клиента (ожидаемой) установить соединение на этот новый порт будет создано связанное соединение и пакеты будут пропущены. Убедиться в том, что обработчик для протокола FTP загружен, можно с помощью команды:

```
# lsmod | grep ftp
nf_conntrack_ftp      20480 0
nf_conntrack 131072 6 xt_conntrack, nf_conntrack_ipv4,
nf_conntrack_pptp, nf_conntrack_netlink, nf_conntrack_proto_gre,
nf_conntrack_ftp
```

Если модуль не найден, его можно загрузить с помощью команды

```
# modprobe nf_conntrack_ftp
```

Прочие опции текущего критерия (`--ctstatus`, `--ctdir`, `--ctorigsrc`, `--ctorigdst`, `--ctreplsrc`, `--ctrepldst`, `--ctexpire` и другие) применяются редко. Например, с помощью опции `--ctexpire` можно выделить соединения, таймер которых находится в нужном диапазоне (время указывается в секундах). Команда:

```
# iptables -A INPUT -p udp -m conntrack --ctexpire 30:100
-j REDIRECT --to-port 8090
```

позволит перенаправить на порт 8090 «задержавшиеся» пакеты в потоке (в рамках соединения, которое будет удалено из таблицы conntrack через 30–100 секунд). Применение этой опции имеет смысл в основном для отладки сетевых приложений. Для каждого протокола устанавливаются свои таймауты (см. раздел 4.1).

4.3. Наборы IP-адресов

Когда необходимо добавить в цепочку МЭ большое количество однотипных правил (с одинаковыми критериями), которые отличаются только критериями проверки на адреса источников/назначения или порты, имеет смысл использовать модуль ядра ip_set и критерий `-m set` для замены группы правил на одно правило, использующее для проверки набор адресов (*ipset*).

Для манипулирования наборами адресов предназначена утилита ipset. С ее помощью можно:

- создать и удалить набор;
- добавить и удалить элемент набора;
- вывести содержимое наборов в заданном формате.

После того как наборы созданы и заполнены, их можно использовать в правилах, которые задаются с помощью критерия `-m set` в iptables.

Что представляет собой наборы IP-адресов? Существует несколько типов, которые отличаются методом поиска и типом сохраняемых данных.

Первый метод поиска — *битовая карта* (*bitmap*), он предназначен для ускорения проверки вхождения элемента в набор, когда возможные значения элементов представляют собой сплошной диапазон значений. Примером может быть IP-подсеть или диапазон портов. Каждому значению элемента соответствует один бит битовой карты.

Второй метод поиска — *хеш-таблица* (*hash*), он предназначен для ускорения проверки вхождения элемента в набор, когда сплошным диапазоном возможные значения элементов представить нет возможности, например, в случае составления набора из непоследовательных IP-адресов или комбинаций IP-адреса и MAC-адреса.

Рассмотрим использование наборов адресов на конкретном примере, а затем кратко опишем существующие типы. Предположим, что перед администратором локального хоста, работающего в качестве маршрутизатора и предоставляющего доступ в Интернет, стоит задача заблокировать обращения хостов ЛВС к некоторым ресурсам в Интернете, заданным IP-адресами и IP-подсетями. Пусть ЛВС подключена к интерфейсу eth0, имеющему адрес 192.168.1.1, а интерфейс eth1 имеет глобально маршрутизируемый адрес 84.130.200.2. Список содержит большое количество значе-

ний, поэтому создавать на каждый адрес по одному запрещающему правилу нецелесообразно.

Сначала необходимо понять, какой тип наборов адресов нужно использовать. Поскольку адреса заданы не диапазоном и включают подсети, следует использовать тип `hash:net`. Создание набора выполняется командой:

```
# ipset create ban hash:net
```

После этого для всех адресов и подсетей из списка блокировки следует выполнить команду вида `# ipset add ban <адрес>`. Например, если в списке подлежащих блокировке адресов есть сеть 88.13.1.0/24, то:

```
# ipset add ban 88.13.1.0/24
```

Проверить вхождение адреса в набор в командной строке можно с помощью команды test:

```
# ipset test ban 88.13.1.42
88.13.1.42 is in ban
```

В результате набор адресов с именем ban будет заполнен значениями и готов к использованию. Теперь в цепочке FORWARD таблицы filter можно добавить проверку:

```
# iptables -A FORWARD -m set --match-set ban dst -j DROP
```

Транзитные пакеты любых протоколов, направленные на адреса из набора ban, будут заблокированы.

Доступны следующие типы наборов адресов:

– `bitmap:ip` позволяет задать сплошной диапазон IP-адресов, включающий до 65536 элементов. Пример добавления и проверки:

```
# ipset create foo bitmap:ip range 192.168.0.0/16
# ipset add foo 192.168.1/24
# ipset test foo 192.168.1.1
192.168.1.1 is in set foo
```

– `bitmap:ip,mac` позволяет задать сплошной диапазон IP-адресов, причем каждый элемент можно дополнить MAC-адресом:

```
# ipset create foo bitmap:ip,mac range 192.168.11.0/8
# ipset add foo 192.168.11.1,12:34:56:78:9A:BC
# ipset test foo 192.168.11.1
192.168.11.1 is in set foo
```

– `bitmap:port` позволяет задать сплошной диапазон портов:

```
# ipset create foo bitmap:port range 0-1024
# ipset add foo 80
```

– `hash:ip` позволяет задать перечень IP-адресов, которые добавляются в качестве ключей в хеш-таблицу. Наряду с `hash:net` это один из наиболее часто используемых типов наборов адресов. Пример использования настоящего типа был приведен выше;

– hash:net позволяет задать перечень IP-подсетей, которые добавляются в качестве ключей в хеш-таблицу. Отдельные IP-адреса также можно хранить в наборе этого типа, если задавать их с cidr-суффиксом:

```
# ipset create foo hash:net
# ipset add foo 192.168.11.0/24
# ipset add foo 172.16.0.0/16
# ipset add foo 192.168.42.1/32
```

– hash:mac позволяет задать перечень MAC-адресов, которые добавляются в качестве ключей в хеш-таблицу:

```
# ipset create foo hash:mac
# ipset add foo 11:22:33:44:55:66
# ipset test foo 11:22:33:44:55:66
11:22:33:44:55:66 is in set foo
```

– hash:ip,port и hash:net,port позволяют хранить IP-адреса или IP-подсети вместе с портами (можно указать даже диапазон портов). При проверке порты учитываются. Дополнительно можно указать протокол:

```
# ipset create foo hash:ip,port
# ipset add foo 192.168.1.0/24,5060-5070
# ipset add foo 192.168.1.1,tcp:8080
```

– hash:ip,port,ip, hash:ip,port,net и hash:net,port,net позволяют хранить кортежи, состоящие из IP-адреса или IP-подсети, порта и IP-адреса или IP-подсети. Пример создания и добавления элементов в набор типа hash:ip,port,net:

```
# ipset create foo hash:ip,port,net
# ipset add foo 192.168.11.1,80,10.0.0.0/24
# ipset add foo 192.168.22.1,25,10.1.0.0/16
```

– hash:net,iface. С помощью такого типа наборов адресов можно задать перечень кортежей, состоящих из IP-подсети и имени сетевого интерфейса:

```
# ipset create foo hash:net,iface
# ipset add foo 192.168.1.0/24,eth0
# ipset add foo 12.1.0.0/16,eth1
# ipset test foo 192.168.1.0/24,eth0
192.168.1.0/24,eth0 is in set foo
```

– hash:ip,mark. С помощью указанного типа наборов адресов можно задать перечень кортежей, состоящих из IP-адреса и 32-битного значения отметки пакета. Этот тип наборов часто применяется при работе с системой ограничения скорости соединений (см. раздел 5). Пример создания и добавления элементов в набор типа hash:ip,port,net:

```
# ipset create foo hash:ip,mark
# ipset add foo 192.168.1.1,0xdeadbeef
# ipset add foo 192.168.11.0/24,777
```

Вопросы к разделу 4

1. Что считается соединением в системе conntrack?
2. Опишите принцип работы системы отслеживания соединений.
3. Для чего необходимо отслеживание соединений МЭ?
4. Какую команду необходимо выполнить, чтобы просмотреть текущее содержимое таблицы conntrack?
5. Какие таблицы доступны для просмотра и манипуляции в системе conntrack?
6. В каких случаях может понадобиться увеличить размер таблицы conntrack?
7. В каких состояниях с точки зрения системы отслеживания соединений может находиться пакет, проходящий по сетевому стеку хоста?
8. Для чего нужны обработчики (хелперы) системы conntrack? Что они собой представляют (как реализуются)? Что необходимо сделать для подключения обработчика?
9. Для чего нужны наборы IP-адресов?
10. Чем метод bitmap отличается от метода hash в типах наборах IP-адресов?
11. Какие типы наборов IP-адресов на самом деле не хранят IP-адреса?
12. Какой тип наборов IP-адресов имеет смысл использовать, если нужно хранить пары <*IP-подсеть*, *MAC-адрес*>?
13. Что сделает команда “`iptables -I INPUT -m conntrack --ctstate INVALID -j DROP`”? Какой смысл в добавлении этой команды в цепочку INPUT?
14. Почему правило “`iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED`” имеет смысл размещать как можно ближе к началу цепочки?

Раздел 5. Требования к межсетевым экранам

5.1. Нормативная база

До сравнительно недавнего времени в нашей стране организация защиты информации (ЗИ) в части использования МЭ более 20 лет определялась нормативными документами 1992–1997 гг.

С 2012 г. ФСТЭК России проводил большую работу по актуализации, развитию и совершенствованию нормативной и методической базы сертификации СЗИ. Были исследованы современные угрозы информационной безопасности, а также изучены мировой опыт стандартизации и нормативного регулирования в области разработки и сертификации СЗИ, в том числе, МЭ. Учитывались реальный уровень технического исполнения МЭ и практика их применения. Итогом этой деятельности стал новый документ ФСТЭК России «Требования к межсетевым экранам», утвержденный Приказом № 9 от 9.02.2016 и зарегистрированный Минюстом РФ 25.03.2016 (регистрационный № 41564). Новые «Требования к межсетевым экранам» вступили в силу 1.12.2016. «Требования к межсетевым экранам», прежде всего, адресованы разработчикам и производителям СЗИ, специалистам, занятым в сфере испытаний и сертификации СЗИ, а также заявителям на сертификацию МЭ.

Требования для отдельного типа средств защиты информации оформлены в виде комплекта документов:

– «Требования к межсетевым экранам» (имеет пометку «для служебного пользования» и определяет классы и типы для отдельного типа изделий);

– профили защиты (ПЗ), определяющие номенклатуру функциональных требований безопасности и требования доверия к безопасности в зависимости от типа и класса изделия. Профили защиты разработаны на базе стандартов ГОСТ Р ИСО/МЭК 15408-2-2013 и 15408-3-2013.

Полный текст «Требований к межсетевым экранам» открыто не публикуется (имеет гриф «Для служебного пользования»). Профили защиты доступны на официальном сайте ФСТЭК России.

В «Требованиях к межсетевым экранам» устанавливаются требования к функциям безопасности МЭ по следующим направлениям:

– требования к составу функций безопасности МЭ и сред, в которых эти средства функционируют;

– требования к составу функциональных возможностей МЭ, обеспечивающих реализацию функций безопасности;

– требования к реализации функциональных возможностей МЭ;

– требования доверия к безопасности МЭ.

Функциональные требования безопасности для МЭ составлены на основе требований ГОСТ Р ИСО/МЭК 15408-2-2013 «Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 2. Функциональные компоненты безопасности». Наиболее важная для пользователей МЭ информация, содержащаяся в «Требованиях к межсетевым экранам», изложена в выпущенных ФСТЭК информационных сообщениях и методических документах. Новая типизация МЭ представлена в табл. 5.1.

Таблица 5.1. Типы межсетевых экранов согласно «Требованиям к межсетевым экранам»

Тип МЭ	**Определение**
Тип «А» (уровня сети)	МЭ, применяемые на физической границе (периметре) ИС или между физическими границами сегментов ИС. Могут иметь только программно-техническое исполнение
Тип «Б» (уровня логических границ сети)	МЭ, применяемые на логической границе (периметре) ИС или между логическими границами сегментов ИС. Могут иметь программное или программно-техническое исполнение
Тип «В» (уровня узла)	МЭ, применяемые на узле (хосте) ИС. Могут иметь только программное исполнение и устанавливаются на мобильных или стационарных технических средствах конкретного узла ИС
Тип «Г» (уровня веб-сервера)	МЭ, применяемые на серверах, обслуживающих сайты, веб-службы и веб-приложения, или на физической границе сегмента таких серверов). Могут иметь программное или программно-техническое исполнение и должны обеспечивать контроль и фильтрацию информационных потоков по протоколу передачи гипертекста, проходящих к веб-серверу и от веб-сервера
Тип «Д» (уровня промышленной сети)	МЭ, применяемые в АСУТП. Могут иметь программное или программно-техническое исполнение и должны обеспечивать контроль и фильтрацию промышленных протоколов передачи данных (Modbus, Profibus, CAN, HART, Industrial Ethernet и (или) иные протоколы)

Для МЭ выделяют 6 классов (самый низкий — 6-й, самый высокий — 1-й). В информационном сообщении ФСТЭК России от 12.09.2016 № 240/24/4278 «Об утверждении методических документов, содержащих профили защиты межсетевых экранов» представлена спецификация профилей защиты для МЭ всех типов и классов защиты. Профили защиты имеют идентификаторы следующей структуры:

ИТ.МЭ.<тип><класс>.ПЗ,

где *<тип>* — это тип МЭ, *<класс>* — класс защиты МЭ (от 1 до 6).

Для типов А, Б и В существуют требования к МЭ от 1 до 6 класса защиты, для типов Г и Д — только от 4 до 6. Установлено соответствие между классами защиты МЭ и классами защищенности ИС (табл. 5.2).

Таблица 5.2. Соответствие между классами защиты МЭ
и классами защищенности ИС

Класс защиты МЭ	**Классы защищенности ИС**		
6	ГИС 3 и 4 классов защищенности	АСУТП 3 класса защищенности	ИСПДн 3 и 4 уровней защищенности
5	ГИС 2 класса защищенности	АСУТП 2 класса защищенности	ИСПДн 2 уровня защищенности
4	ГИС 1 класса защищенности	АСУТП 1 класса защищенности	ИСПДн 1 уровня защищенности ИС общего пользования II класса
3	ИС обработки информации, которая содержит сведения, составляющие государственную тайну		
2			
1			

Ниже приведены другие варианты представления соответствия классов защиты МЭ и типов и классов защищенности ИС классификации МЭ (табл. 5.3):

Таблица 5.3. Соответствие между классами защиты МЭ
и уровнями защищенности ГИС, АСУТП и ИСПДн

	МЭ 1	**МЭ 2**	**МЭ 3**	**МЭ 4**	**МЭ 5**	**МЭ 6**
ИСПДн 4 УЗ	+	+	+	+	+	+
ИСПДн 3 УЗ	+	+	+	+	+	+
ИСПДн 2 УЗ	+	+	+	+	+	
ИСПДн 1 УЗ	+	+	+	+		
ГИС К4	+	+	+	+	+	+
ГИС К3	+	+	+	+	+	+
ГИС К2	+	+	+	+	+	
ГИС К1	+	+	+	+		
АСУТП К3	+	+	+	+	+	+
АСУТП К2	+	+	+	+	+	
АСУТП К1	+	+	+	+		
ИС с гос. тайной	+	+	+			

В последней таблице легко увидеть, что МЭ 4 класса может использоваться для всех ИС, кроме систем, работающих с государственной тайной.

Основные требования ФСТЭК к МЭ можно охарактеризовать следующим образом:

1) установка пяти типов МЭ;
2) учет актуальных угроз безопасности информации;
3) предъявление дополнительных требований доверия;
4) установка требований к среде функционирования;
5) учет современных технологий межсетевого экранирования;
6) установка требований по самозащите межсетевых экранов»;
7) возможность глубокого анализа протоколов при фильтрации.

Угрозы безопасности, которые должен нейтрализовать МЭ, можно подразделить на:

1) *внешние*:
- несанкционированный доступ к информации, содержащейся в ИС;
- отказ в обслуживании ИС;
- вызов нарушения функционирования МЭ;
- несанкционированное получение сведений о сети и ее узлах;

2) *внутренние*:
- несанкционированная передача информации из ИС;
- отказ в обслуживании ИС;
- вызов нарушения функционирования МЭ.

5.2. Профили защиты и политики безопасности

Приведем далее несколько выдержек из методического документа «Профиль защиты межсетевых экранов типа «Д» шестого класса защиты ИТ.МЭ.Д6.ПЗ».

Согласно Профилю ИТ.МЭ.Д6.ПЗ МЭ типа «Д» шестого класса защиты — это «программное или программно-техническое средство, реализующее функции контроля и фильтрации в соответствии с заданными правилами проходящих через него информационных потоков и используемое в целях обеспечения защиты (некриптографическими методами) информации, содержащей сведения, составляющие государственную тайну, и иной информации ограниченного доступа».

МЭ «должен обеспечивать нейтрализацию следующих угроз безопасности информации:

– несанкционированный доступ к информации, содержащейся в автоматизированной системе управления;

– отказ в обслуживании автоматизированной системы управления и (или) ее отдельных компонентов;

– несанкционированная передача информации из автоматизированной системы управления в информационно-телекоммуникационные сети или иные информационные системы;

– несанкционированное воздействие на МЭ, целью которого является нарушение его функционирования, включая преодоление или обход его функций безопасности».

Отмечается, что в МЭ не должно содержаться программ, не выполняющих функций безопасности или не предназначенных для обеспечения функционирования МЭ (сторонних программ). Далее перечисляются функции безопасности, которые должны быть реализованы МЭ:

– контроль и фильтрация;

– идентификация и аутентификация;
– регистрация событий безопасности (аудит);
– обеспечение бесперебойного функционирования и восстановление;
– управление (администрирование).

Приводятся также требования к среде, в которой функционирует МЭ. В ней должны быть реализованы следующие функции безопасности МЭ:

– исключение каналов связи в обход правил фильтрации;
– обеспечение доверенного канала;
– обеспечение доверенного маршрута;
– физическая защита;
– обеспечение безопасного функционирования;
– обеспечение взаимодействия с сертифицированными средствами защиты информации.

Функции безопасности МЭ должны обладать функциональными возможностями (функциональными требованиями безопасности), которые обеспечивают реализацию этих требований.

В ПЗ перечислены следующие виды требований безопасности, предъявляемые к МЭ МЭ:

– функциональные требования безопасности МЭ;
– требования доверия к безопасности МЭ.

Функциональные требования безопасности МЭ, изложенные в ПЗ, включают:

– требования к управлению потоками информации;
– требования к идентификации и аутентификации субъектов межсетевого взаимодействия;
– требования к регистрации событий безопасности (аудиту);
– требования к обеспечению бесперебойного функционирования МЭ и восстановлению;
– требования к управлению МЭ.

Состав функциональных требований безопасности, включенных в ПЗ, обеспечивает следующие функциональные возможности МЭ типа «Д» МЭ:

– возможность осуществлять фильтрацию сетевого трафика для отправителей информации, получателей информации (в том числе исполнительных устройств) и всех операций передачи контролируемой МЭ информации к узлам автоматизированной системы управления и от них;
– возможность обеспечения фильтрации для всех операций перемещения через МЭ информации к узлам автоматизированной системы управления и от них;
– возможность осуществлять фильтрацию, основанную на следующих типах атрибутов безопасности субъектов: сетевой адрес узла отправи-

теля, сетевой адрес узла получателя, сетевой протокол, который используется для взаимодействия;

– возможность явно разрешать информационный поток, базируясь на устанавливаемых администратором МЭ наборе правил фильтрации, основанном на идентифицированных атрибутах;

– возможность явно запрещать информационный поток, базируясь на устанавливаемых администратором МЭ наборе правил фильтрации, основанном на идентифицированных атрибутах;

– возможность регистрации и учета выполнения проверок информации сетевого трафика;

– возможность читать информацию из записей аудита уполномоченным администраторам;

– возможность выбора совокупности событий, подвергающихся аудиту, из совокупности событий, в отношении которых возможно осуществление аудита;

– возможность оповещения уполномоченных лиц о критичных видах событий безопасности, в том числе сигнализация о попытках нарушения правил межсетевого экранирования;

– возможность регистрации возникновения событий, которые в соответствии с национальным стандартом Российской Федерации ГОСТ Р ИСО/МЭК 15408-2-2013 «Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 2. Функциональные компоненты безопасности» включены в минимальный уровень аудита;

– возможность (для предусмотренных сценариев функционирования) выполнения определенных действий по управлению МЭ без прохождения процедуры идентификации;

– возможность идентификации администратора МЭ до разрешения действий (по администрированию), выполняемого при посредничестве МЭ от имени этого администратора;

– возможность (для предусмотренных сценариев функционирования) выполнения определенных действий по управлению МЭ без прохождения процедуры аутентификации;

– возможность аутентификации администратора МЭ до разрешения действий (по администрированию), выполняемого при посредничестве МЭ от имени этого администратора;

– поддержка определенных ролей по управлению МЭ;

– возможность со стороны администраторов МЭ управлять режимом выполнения функций безопасности МЭ;

– возможность со стороны администраторов МЭ управлять данными МЭ, используемыми функциями безопасности МЭ;

– возможность со стороны администраторов МЭ управлять атрибутами безопасности;

– возможность обеспечения перехода в режим аварийной поддержки, который предоставляет возможность возврата МЭ к штатному режиму функционирования.

Существенным отличием профилей защиты МЭ класса 6 от профилей защиты более высоких классов является отсутствие описания угроз безопасности, которые должен нейтрализовывать МЭ, требований к политике безопасности и перечня предположений безопасности. Например, требования к политике безопасности сформулированы в виде набора правил и предположений.

Политика безопасности-1. Должно обеспечиваться блокирование передачи защищаемой информации, сетевых запросов и трафика, несанкционированно исходящих из информационной системы и (или) входящих в информационную систему, путем фильтрации информационных потоков.

Политика безопасности-2. Должно осуществляться присвоение информации состояния соединения только допустимых значений.

Политика безопасности-3. Должна обеспечиваться интерпретация управляющих сигналов от средств защиты информации и блокирование соответствующего трафика.

Политика безопасности-4. Должно осуществляться разграничение доступа к управлению МЭ и параметрами МЭ на основе ролей уполномоченных лиц.

Политика безопасности-5. Должна обеспечиваться возможность управления работой МЭ и параметрами МЭ со стороны администраторов МЭ.

Политика безопасности-6. Должны обеспечиваться идентификация и аутентификация администраторов МЭ.

Политика безопасности-7. Должны обеспечиваться механизмы регистрации возможных нарушений безопасности.

Политика безопасности-8. Должны обеспечиваться установка безопасного состояния функций безопасности объекта или предотвращение их перехода в опасное состояние после сбоев, прерывания функционирования или перезапуска.

Политика безопасности-9. Должно осуществляться ведение отдельных профилей проверок для типовых мест расположения узла информационной системы с установленным МЭ.

Политика безопасности-10. Должна осуществляться выдача предупреждающих сообщений пользователю об обнаружении возможного нарушения безопасности и предоставлять пользователю возможность осуществить определенные действия при обнаружении возможного нарушения безопасности.

Предположение безопасности-1. Должна обеспечиваться физическая защита средства вычислительной техники, на котором функционирует МЭ.

Предположение безопасности-2. Должно обеспечиваться исключение каналов связи защищаемой информационной системы с иными информационными системами в обход МЭ.

Предположение безопасности-3. Должен обеспечиваться доверенный канал передачи данных между защищаемой информационной системой и МЭ, а также между МЭ и терминалом, с которого выполняется его управление.

Предположение безопасности-4. Должен обеспечиваться доверенный маршрут между МЭ и администраторами МЭ.

Предположение безопасности-5. Должно обеспечиваться взаимодействие МЭ с сертифицированными на соответствие требованиям безопасности информации по соответствующему классу защиты средствами защиты информации (системами обнаружения вторжений, средствами антивирусной защиты и другими), от которых МЭ получает управляющие сигналы.

Предположение безопасности-6. Должны быть обеспечены совместимость компонентов МЭ с компонентами средств вычислительной техники информационной системы, а также необходимые ресурсы для выполнения функций безопасности МЭ (в том числе изоляция данных и процессов МЭ от иных данных и процессов средства вычислительной техники, на котором он функционирует).

Предположение безопасности-7. Должно быть обеспечено функционирование МЭ в среде сертифицированной на соответствие требованиям безопасности информации по соответствующему классу защиты операционной системы или в среде, защищенной путем принятия мер защиты информации, соответствующих классу защищенности информационной системы (автоматизированной системы управления), для использования в которой предназначается МЭ.

Предположение безопасности-8. Должны быть обеспечены тестирование и контроль целостности аппаратных средств, а также программного обеспечения базовой системы ввода-вывода, загрузчика и операционной системы МЭ или средства вычислительной техники, на котором он функционирует.

Предположение безопасности-9. Персонал, ответственный за функционирование ОО, должен обеспечивать установку, настройку и эксплуатацию МЭ в соответствии с правилами по безопасной настройке и руководством пользователя (администратора).

Сделаем некоторые замечания относительно определения типа и класса конкретного МЭ. Рассмотрим, например, встроенный в Linux МЭ Netfilter. Существуют отдельные сертифицированные ОС специального назначения, основанные на дистрибутивах ОС Linux, например, Astra Linux SE, «Заря», МСВС, но входящие в их состав МЭ отдельно не сертифицированы. Дистрибутивы ОС Linux, используемые на обычных рабочих

местах (не используемые в ГИС, ИСПДн и так далее), не проходят сертификацию ФСТЭК. Соответственно, встроенный в них МЭ не является сертифицированным. Теоретически (согласно представленным выше требованиям) МЭ Netfilter мог бы быть отнесен к типу В (если компьютер с ОС Linux рассматривать как узел информационной системы).

Вопросы к разделу 5

1. Охарактеризуйте нормативную базу, определяющую требования к межсетевым экранам в Российской Федерации.
2. Что учитывалось в новых требованиях по межсетевым экранам?
3. Как классы защиты межсетевых экранов соотносятся с классами защищенности информационных систем?
4. Что такое профили защиты? Охарактеризуйте их.
5. Какие требования выдвигаются к среде, в которой функционирует межсетевой экран?
6. Перечислите основные требования ФСТЭК к межсетевым экранам.
7. Какие угрозы должен нейтрализовывать межсетевой экран типа «Д» шестого класса защиты?

Заключение

Правильная настройка межсетевого экрана является важным шагом в построении надежной и безопасной IT-инфраструктуры. Понимание основных принципов фильтрации трафика и отслеживания соединений сетевой подсистемой ОС позволяет специалисту по информационной безопасности обнаруживать потенциальные уязвимости и своевременно устранять возможность реализации атак. В настоящем пособии указанные принципы рассмотрены на примере системы Netfilter ядра ОС Linux как наиболее популярного решения в различных прикладных областях вычислительных систем и сетей связи.

Многие вопросы, имеющие отношение к Netfilter, но требующие более глубокого погружения в предмет, не были рассмотрены в настоящем пособии. К их числу относятся, прежде всего, специфичные для IPv6 и редко применяемых транспортных протоколов (DCCP, SCTP) критерии фильтрации пакетов, использование BPF, возможность связи правил с подсистемой cgroup, особенности настройки правил для протоколов туннелирования (таких как IPSec, GRE и других), поддержка прозрачного проксирования и сбора статистики соединений. Также не были рассмотрены вопросы ограничения скорости соединений, которые в большинстве систем, использующих ядро Linux, решаются с помощью системы Netfilter: особенности организации шейпинга и полисинга, отличия дисциплин очередей пакетов и т.д. Читателю предлагается самостоятельно ознакомиться с этими вопросами.

Система Netfilter и ядро ОС Linux постепенно развиваются и совершенствуются для того, чтобы адекватно решать новые проблемы и служить надежным фундаментом обеспечения сетевой безопасности в ОС Linux. В то же время значительное внимание уделяется обеспечению обратной совместимости. С 2009 года ведется работа по замене значительной части подсистем Netfilter более функциональными и производительными реализациями, которые доступны с ядра Linux 3.13 в виде пакета nftables. Есть основания полагать, что в ближайшее время эти разработки будут включены в основную ветку ядра.

Список литературы

1. Алиев Т.И. Сети ЭВМ и телекоммуникации. — СПб : СПбГУ ИТМО, 2011. — 400 с.
2. Лапонина О.Р. Межсетевое экранирование. — М. : «Бином. Лаборатория знаний», 2007. — 343 с. — ISBN 5-94774-603-4.
3. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы / Римицан Н.А. — 5-е изд. — СПб : Изд-во «Питер», 2019. — 992 с. — ISBN 978-5-4461-1343-9.
4. Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. — М. : «ДМК-Пресс», 2012. — 592 с. — ISBN 978-5-94074-833-5.
5. ГОСТ Р ИСО/МЭК 15408-2-2013. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 2. Функциональные компоненты безопасности.
6. ГОСТ Р ИСО/МЭК 15408-3-2013. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Компоненты доверия к безопасности.
7. ГОСТ Р ИСО/МЭК 15408-3-2013 Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Компоненты доверия к безопасности.
8. Документы по сертификации средств защиты информации и аттестации объектов информатизации по требованиям безопасности информации. — URL: https://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty-po-sertifikatsii/120-normativnye-dokumenty.
9. Информационное сообщение «Об утверждении Требований к межсетевым экранам», № 240/24/1986, 28.04.2016.
10. Информационное сообщение по вопросам разработки, производства, поставки и применения межсетевых экранов, сертифицированных ФСТЭК России по требованиям безопасности информации, 24.03.2017.
11. Информационное сообщение ФСТЭК России «Об утверждении методических документов, содержащих профили защиты межсетевых экранов», №240/24/4278, 12.09.2016.
12. Методический документ «Профиль защиты межсетевых экранов типа «В» четвертого класса защиты ИТ.МЭ.Д6.ПЗ», 12.09.2016.
13. Методический документ «Профиль защиты межсетевых экранов типа «д» шестого класса защиты ИТ.МЭ.Д6.ПЗ», 12.09.2016.

14. Методический документ ФСТЭК «Меры защиты информации в государственных информационных системах».
15. СОИБ. Проектирование. Новые требования к межсетевым экранам. — URL: https://www.securitylab.ru/blog/personal/sborisov/294638.php.

Будько Марина Борисовна
Будько Михаил Юрьевич
Гирик Алексей Валерьевич

Использование межсетевого экрана Netfilter для обеспечения сетевой безопасности в ОС Linux

Учебное пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49