ipfw - межсетевой экран (брандмауэр) IP и программа управления профилем передаваемой информации

НАЗВАНИЕ

ipfw - *межсетевой экран* (*брандмауэр*) IP и программа управления профилем передаваемой информации

СИНТАКСИС

```
ipfw [-q] [-p ??????????? [-D ??????[=????????]] [-U ??????]] ??????_???_?????
ipfw [-f | -q] flush
ipfw [-q] {zero | resetlog | delete} [????? ...]
ipfw [-s [????]] [-aftN] {list | show} [????? ...]
ipfw [-q] add [?????] ????_???????
ipfw pipe ????? config ?????_???????????????_??????
ipfw pipe {delete | list | show} [????? ...]
ipfw queue ????? config ?????_???????????????_???????
ipfw queue {delete | list | show} [????? ...]
```

ОПИСАНИЕ

ipfw - это пользовательский интерфейс для управления брандмауэром **ipfirewall(4)** и *формирователем трафика* (traffic shaper) **dummynet(4)** в ОС FreeBSD.
К каждому входящему и исходящему пакету применяются правила **ipfw**. Если хост работает как *шлюз* (gateway), пакеты, перенаправляемые шлюзом, обрабатываются программой **ipfw** дважды. Если хост работает как мост (bridge), пакеты, перенаправляемые мостом, обрабатываются программой **ipfw** один раз.
Конфигурация брандмауэра задается в виде списка пронумерованных правил, который просматривается для каждого пакета пока не будет найдено соответствие - тогда выполняется заданное соответствующим правилом действие. В зависимости от действия и некоторых установок в системе, пакеты могут возвращаться на брандмауэр для обработки правилами, начиная со следующего за сработавшим. Все правила применяются ко всем интерфейсам, так что задача написания набора правил с минимально необходимым количеством проверок возлагается на системного администратора.
Конфигурация всегда включает стандартное правило (**DEFAULT**) с номером 65535, которое нельзя изменять и которое соответствует любому пакету. С этим стандартным правилом может быть связано действие **deny** или **allow**, в зависимости от конфигурации ядра.
Если набор правил включает одно или несколько правил с опцией **keep-state**, то программа **ipfw** предполагает работу *с сохранением состояния* (stateful behaviour), т.е. при успешном сопоставлении будет создавать динамические правила, соответствующие конкретным параметрам (адресам и портам) сопоставившегося пакета.
Эти динамические правила с ограниченным временем существования проверяются, начиная с первого вхождения правила **check-state** или **keep-state**, и обычно используются для приоткрытия брандмауэра по требованию только для допустимого трафика. Подробнее о работе программы **ipfw** с сохранением состояния см. далее в разделах ФОРМАТ ПРАВИЛА и ПРИМЕРЫ.
Со всеми правилами (включая динамические) связано несколько счетчиков: счетчик пакетов, счетчик байтов, счетчик журнала (log count) и временная отметка (timestamp), хранящая время последнего сопоставления. Счетчики можно выдать или сбросить с помощью команд

ipfw.
Правила добавляются с помощью команды **add**; удаляются по одному - с помощью команды **delete**, и все сразу - с помощью команды **flush**; выдаются, в том числе, возможно, со значениями счетчиков, с помощью команд **show** и **list**. Наконец, значения счетчиков можно сбросить с помощью команд **zero** и **resetlog**.
Поддерживаются следующие опции:

-a
Выдавать значения счетчиков вместе с правилами. См. также описание команды **show**.
-f
Не запрашивать подтверждения для команд, которые при неправильном использовании могут вызвать проблемы, например, **flush**. Учтите, что если процесс не связан с терминалом, эта опция подразумевается.
-q
При добавлении, обнулении, сбросе журнала или списка правил не выдавать информации о действиях (эта опция подразумевает опцию **-f**). Это полезно при настройке списка правил с помощью нескольких команд **ipfw** в сценарии (например, **sh /etc/rc.firewall**), или при обработке файла с несколькими правилами **ipfw** в сеансе с удаленной регистрацией. Если команда **flush** выполняется в обычном (информативном) режиме (в стандартной конфигурации ядра), она выдает сообщение. Поскольку все правила сбрасываются, сообщение это не может быть передано сеансу. В результате, сеанс с удаленной регистрацией закрывается и остальные правила не обрабатываются. Для восстановления придется зарегистрироваться с консоли.
-t
При выдаче правил указывать временную отметку последнего сопоставления.
-N
При выдаче пытаться разрешать адреса и имена служб.
-s [**поле**]
При выдаче каналов, сортировать в соотетствии с одним из четырех счетчиков (общее/текущее количество пакетов или байтов).

Для упрощения конфигурирования правила можно помещать в файл, который обрабатывается с помощью команды **ipfw** как показано в первой строке синтаксиса. Необходимо указывать полное имя файла. Файл будет читаться построчно и строки применяются как аргументы вызова утилиты **ipfw**.
Можно также указать препроцессор (с помощью конструкции **-p препроцессор**), через который будет пропущен заданный файл. Пригодятся препроцессоры **cpp(1)** и **m4(1)**. Если параметр **препроцессор** не начинается с косой черты ('/'), выполняется обычный поиск с учетом значения переменной среды **PATH**. Это надо учитывать в средах, где не все файловые системы еще смонтированы в момент выполнения команды **ipfw** (например, если они монтируются по NFS). Если была указана опция **-p**, можно также задавать спецификации **-D** и **-U**, которые будут передаваться препроцессору. Это позволяет создавать гибкие файлы конфигурации (например, включающие зависимости от имени локального хоста) и использовать макросы для часто используемых аргументов вроде IP-адресов.
Команды **ipfw pipe** используются для конфигурирования формирователя трафика, как описано в разделе КОНФИГУРИРОВАНИЕ ФОРМИРОВАТЕЛЯ ТРАФИКА ниже.

ФОРМАТ ПРАВИЛ

Правила **ipfw** имеют следующий формат:
[prob **вероятность_сопоставления**] **действие** [log [logamount **число**]] **протокол** from **исходный_адрес** to **целевой_адрес** [**спецификация_интерфейса**] [**опции**]
Каждый пакет можно фильтровать на основе следующей, связанной с ним информации:

- Интерфейс передачи и приема (по имени или адресу)
- Направление (входящий или исходящий)

- Исходный и целевой IP-адреса (возможно, замаскированные)
- Протокол (TCP, UDP, ICMP и т.п.)
- Исходный и целевой порт (можно указывать списки, диапазоны или маски)
- Флаги TCP
- Флаг IP-фрагмента
- Опции IP
- Типы ICMP
- Идентификатор пользователя/группы для сокета, связанного с пакетом

Учтите, что фильтровать пакеты на основе исходного IP-адреса или исходного порта TCP/UDP небезопасно, поскольку адрес и номер порта легко подделать.

prob вероятность_сопоставления
Сопоставление происходит только с заданной вероятностью (вещественное число в диапазоне от 0 до 1). Эта возможность может пригодиться для различных целей, например, для случайной потери пакетов или (при использовании совместно с **dummynet(4)**) для имитации эффекта доставки пакетов в другом порядке из-за использования нескольких путей.
действие:
allow
Пропускать пакеты, соответствующие правилу. Дальнейший поиск прекращается. У этого действия есть псевдонимы: **pass**, **permit** и **accept**.
deny
Отвергать пакеты, соответствующие правилу. Дальнейший поиск прекращается. У этого действия есть псевдоним - **drop**.
reject
(Не рекомендуется). Отвергать пакеты, соответствующие этому правилу, и пытаться посылать уведомление о недостижимости хоста по протоколу ICMP. Дальнейший поиск прекращается.
unreach код
Отвергать пакеты, соответствующие этому правилу, и пытаться посылать уведомление о недостижимости по протоколу ICMP с указанным **кодом**, где **код** - число от 0 до 255 или один из следующих псевдонимов: **net**, **host**, **protocol**, **port**, **needfrag**, **srcfail**, **net-unknown**, **host-unknown**, **isolated**, **net-prohib**, **host-prohib**, **tosnet**, **toshost**, **filter-prohib**, **host-precedence** или **precedence-cutoff**. Дальнейший поиск прекращается.
reset
Только для TCP-пакетов. Отвергать пакеты, соответствующие этому правилу, и попытаться послать уведомление TCP **reset** (RST). Дальнейший поиск прекращается.
count
Обновить счетчики для всех пакетов, соответствующих правилу. Поиск продолжается со следующего правила.
check-state
Проверяет пакет по динамическому набору правил. Если найдено соответствие, дальнейший поиск прекращается, иначе - переходим к следующему правилу. Если правило **check-state** не найдено, динамический набор правил проверяется с первого правила **keep-state**.
divert порт
Перенаправляет пакеты, соответствующие правилу, на сокет **divert(4)**, связанный с указанным **портом**. Дальнейший поиск прекращается.
tee порт
Посылает копию пакетов, соответствующих правилу, на сокет **divert(4)**, связанный с указанным **портом**. Дальнейший поиск прекращается и исходный пакет принимается (см., однако, раздел **ОШИБКИ** далее).
fwd ip-адрес[,порт]
Изменяет следующий переход (next-hop) для соответствующих пакетов, направляя их по указанному **ip-адресу**, который можно задавать либо в виде четверки чисел через точку, либо по имени хоста. Если **ip-адрес** непосредственно не доступен, вместо него используется соответствующий маршрут, полученный по локальной таблице маршрутизации. Если **ip-адрес** - локальный, при поступлении в систему пакета с удаленного хоста он будет перенаправлен на заданный **порт** на локальной машине, так что локальный адрес сокета

остается установленным в соответствии с исходным IP-адресом, по которому направлялся пакет. Это действие предназначено для использования в прозрачных промежуточных серверах. Если IP-адрес - не локальный, дальнейший поиск прекращается; однако, при выходе из канала если переменная **sysctl(8) net.inet.ip.fw.one_pass** не установлена, пакет снова передается брандмауэру для проверки, начиная со следующего правила.
queue номер_очереди
Передает пакет в "очередь" **dummynet(4)** (для ограничения пропускной способности с помощью WF2Q).
skipto номер
Пропускает все последующие правила с номерами меньше указанного. Поиск продолжается с первого правила, номер которого равен указанному или больше его.

log [**logamount количество**]
Если ядро было скомпилировано с опцией **IPFIREWALL_VERBOSE**, то когда пакет соответствует правилу с ключевым словом **log** в журнал с помощью демона **syslogd(8)** записывается сообщение от источника **LOG_SECURITY**. *Примечание*: по умолчанию, сообщения добавляются в файл **/var/log/security** (см. **syslog.conf(5)**). Если ядро было скомпилировано с опцией **IPFIREWALL_VERBOSE_LIMIT**, то по умолчанию журнализация прекратится после того, как заданное в этой опции количество пакетов будет получено соответствующим правилом в цепочке, и параметр **net.inet.ip.fw.verbose_limit** будет установлен равным указанному количеству. Однако, если используется опция **logamount количество**, именно это **количество** будет ограничивать записываемые в журнал пакеты, а не **net.inet.ip.fw.verbose_limit**, причем, значение "0" снимает ограничение на количество записей в журнал. Затем журнализацию можно включить повторно путем сброса ограничения или счетчика пакетов для соответствующей записи.
Журнализация на консоль и ограничение количества записей в журнал настраивается динамически с помощью интерфейса **sysctl(8)** в базе MIB **net.inet.ip.fw**.

протокол
IP-протокол, заданный по номеру или по имени (полный список см. в файле **/etc/protocols**). Ключевые слова **ip** или **all** обозначают, что подходит любой протокол.

исходный_адрес и **целевой_адрес**:
any | **me** | [**not**] <**?????/?????**> [**?????**]
Если указать **any**, правилу будет соответствовать любой IP-адрес.
Если указать **me**, правилу будет соответствовать любой IP-адрес, сконфигурированный на любом из интерфейсов системы. Эта проверка требует достаточно много вычислительных ресурсов и ее надо использовать осторожно.
<**адрес**/**маску**> можно задавать в следующем виде:
ipno
IP-адрес в виде четверки чисел через точку, например, **1.2.3.4**. Правилу будет соответствовать только этот IP-адрес.
ipno/bits
IP-адрес в виде четверки чисел через точку плюс размер маски в битах, например, **1.2.3.4/24**. В данном случае соответствовать будут все IP-адреса в диапазоне от **1.2.3.0** до **1.2.3.255**.
ipno:mask
IP-адрес в виде четверки чисел с маской через точку в виде четверки чисел через точку, например, **1.2.3.4:255.255.240.0**. В данном случае соответствовать будут все IP-адреса в диапазоне от **1.2.0.0** до **1.2.15.255**.
Можно потребовать, чтобы правилу соответствовали все адреса, кроме указанных, добавив перед адресом модификатор **not**. Этот модификатор не влияет на выбор по номерам портов.
Для протоколов TCP и UDP можно также дополнительно указать порты в виде:
{**????**|**????**-**????**|**????**:**?????**}[,**????**[,...]]
Дефис (**-**) позволяет задать диапазон портов (включая границы).
Двоеточие (**:**) позволяет задать порт и маску - пакет считается соответствующим правилу,

если номер порта в нем соответствует указанному в правиле с точностью до битов, установленных в маске.
Вместо числовых значений портов можно указать имена служб (из файла **/etc/services**).
Диапазон можно указывать только в качестве первого элемента списка, а длина списка портов ограничена значением **IP_FW_MAX_PORTS** (которое задано в заголовочном файле **/usr/src/sys/netinet/ip_fw.h**). Обратная косая (\) позволяет замаскировать дефис (**-**) в имени службы:

```
ipfw add count tcp from any ftp\\-data-ftp to any
```

Фрагментированные пакеты с ненулевым смещением (т.е. не первый фрагмент) никогда не будут соответствовать правилу, в котором задан один или несколько портов. Подробнее о сопоставлении фрагментированных пакетов см. в описании опции **frag**.

спецификация_интерфейса
Можно указывать комбинации следующих спецификаций:

in
Соответствует только входящим пакетам.
out
Соответствует только исходящим пакетам.
via *ifX*
Пакет должен пройти через интерфейс ***ifX***.
via *if**
Пакет должен пройти через интерфейс ***ifX***, где ***X*** - любой номер устройства.
via any
Пакет должен пройти через *любой* интерфейс.
via *ipno*
Пакет должен пройти через интерфейс с IP-адресом **ipno**.
Ключевое слово **via** приводит к обязательной проверке интерфейса. Если вместо **via** использовано ключевое слово **recv** или **xmit**, то проверяется только интерфейс получения или интерфейс передачи, соответственно. Задав оба этих ключевых слова, можно отбирать пакеты на основе обоих интерфейсов, получения и отправки, например:

```
ipfw add 100 deny ip from any to any out recv ed0 xmit ed1
```

Интерфейс **recv** можно проверять на входящие и исходящие пакеты, а интерфейс **xmit** - только на исходящие пакеты. Поэтому ключевое слово **out** - обязательно (а **in** - недопустимо), если используется **xmit**. Указывать **via** вместе с **xmit** или **recv** нельзя.
Пакет может не иметь интерфейса получения или передачи: пакеты, поступающие с локального хоста, не имеют интерфейса получения, а пакеты, предназначенные для локального хоста, не имеют интерфейса передачи.

опции:
keep-state [*метод*]
При сопоставлении брандмауэр создаст динамическое правило, по умолчанию соответствующее двунаправленному обмену данными между исходным и целевым IP-адресом/портом по тому же самому протоколу. Это правило имеет ограниченное время существования (определяемое набором переменных **sysctl(8)**), и это время изменяется при выявлении каждого соответствующего пакета.
Фактическое поведение можно изменить, задав другой ***метод***.

bridged
Соответствует только пакетам, прошедшим через мост. Это может пригодиться для отбора пакетов многоадресной или широковещательной передачи, которые в противном случае будут проходить через брандмауэр дважды: один раз при прохождении через мост, а второй - когда пакет попадает в локальный стек.
Помимо небольшого снижения производительности, это приводит к проблемам при использовании *каналов*, поскольку один и тот же пакет будет учитываться дважды при определении пропускной способности (bandwidth), использования очередей и при изменении счетчиков.

frag
Соответствует пакету, являющемуся не первым фрагментом датаграммы. Опцию **frag** нельзя использовать в сочетании с **tcpflags** или спецификациями порта TCP/UDP.
ipoptions *спецификация*
Соответствует пакету, если его IP-заголовок содержит указанный в ***спецификации*** список оцпий через запятую. Поддерживаются следующие опции IP:

ssrr (strict source route), **lsrr** (loose source route), **rr** (record packet route) и **tsОтсутствие определенной опции может обозначаться восклицательным знаком (!).** (timestamp).

tcpoptions *спецификация*
Соответствует пакету, если его TCP-заголовок содержит указанный в ***спецификации*** список оцпий через запятую. Поддерживаются следующие опции TCP:
mss (maximum segment size - максимальный размер сегмента), **window** (tcp window advertisement - предлагаемое окно tcp), **sack** (selective ack - избирательное подтверждение), **ts** (rfc1323 timestamp - временная отметка по стандарту RFC 1323) и **cc** (rfc1644 t/tcp connection count - счетчик подключения по стандарту RFC 1644). Отсутствие определенной опции может обозначаться восклицательным знаком (**!**).

established
Только для TCP-пакетов. Соответствует пакетам с установленными битами **RST** или **ACK**.
setup
Только для TCP-пакетов. Соответствует пакетам с установленным битом **SYN**, но со сброшенным битом **ACK**.
tcpflags *спецификация*
Только для TCP-пакетов. Соответствует пакету, если его TCP-заголовок содержит заданный в ***спецификации*** список флагов через запятую. Поддерживаются следующие флаги TCP:
fin, **syn**, **rst**, **psh**, **ack** и **urg**. Отсутствие определенной опции может обозначаться восклицательным знаком (**!**). Правило, содержащее спецификацию **tcpflags**, не может сопоставиться с фрагментированным пакетом, имеющим ненулевое смещение. Подробнее о сопоставлении фрагментированных пакетов см. в описании опции **frag**.

icmptypes *типы*
Только для пакетов ICMP. Соответствует пакету, если его ICMP-тип входит в указанный список типов. Список можно задавать как любую комбинацию диапазонов или отдельных типов через запятые. Поддерживаются следующие ICMP-типы:
echo reply (**0**, эхо-ответ), destination unreachable (**3**, целевой адрес недоступен), source quench (**4**), redirect (**5**, перенаправление), echo request (**8**, эхо-запрос), router advertisement (**9**, представление маршрутизатора), router solicitation (**10**), time-to-live exceeded (**11**, превышено время существования), IP header bad (**12**, неверный IP-заголовок), timestamp request (**13**, запрос временной отметки), timestamp reply (**14**, временная отмета в ответ), information request (**15**, запрос информации), information reply (**16**, ответ на запрос информации), address mask request (**17**, запрос маски адреса) и address mask reply (**18**, маска адреса в ответ).

uid *пользователь*
Соответствует всем TCP- или UDP-пакетам, посланным или полученным указанным ***пользователем***. ***Пользователя*** можно задавать по имени или с помощью числового идентификатора.
gid *группа*
Соответствует всем TCP- или UDP-пакетам, посланным или полученным для указанной ***группы*** . ***Группу*** можно задавать по имени или с помощью числового идентификатора.

КОНФИГУРАЦИЯ ФОРМИРОВАТЕЛЯ ТРАФИКА

Утилита **ipfw** также обеспечивает пользовательский интерфейс для формирователя трафика **dummynet(4)**. Формирователь работает путем деления пакетов на *потоки* в соответствии с заданной пользователем маской по различным полям заголовка IP. Пакеты, принадлежащие к

одному потоку, затем передаются двум различным объектам - *каналу* или *очереди*.
Канал эмулирует связь с заданными пропускной способностью, задержкой при передаче, размером очереди и процентом потери пакетов. Пакеты проходят по каналу в соответствии с его параметрами.
Очередь - абстракция, используемая для реализации правил WF2Q+. Очередь связывает с каждым потоком вес и соответствующий канал. Далее, все потоки, связанные с одним и тем же каналом, переключаются с частотой, определяемой каналом в соответствии с правилами WF2Q+.
Формат конфигурации канала **ipfw** следующий:
pipe *число* **config** [**bw** *пропускная_способность* | *устройство*] [**delay** *задержка_в_миллисекундах*] [**queue** {*слоты* | *размер*}] [**plr** *вероятность_потери*] [**mask** *спецификатор_маски*] [**buckets** *размер_хеш-таблицы*] [**red** | **gred** *w_q/min_th/max_th/max_p*]
Формат конфигурации очереди **ipfw** следующий:
queue *число* **config** [**pipe** *номер_канала*] [**weight** *вес*] [**queue** {*слотов* | *размер*}] [**plr** *вероятность_потери*] [**mask** *спецификатор_маски*] [**buckets** *размер_хеш-таблицы*] [**red** | **gred** *w_q/min_th/max_th/max_p*]
Для канала можно сконфигурировать следующие параметры:

bw *пропускная_способность* | *устройство*
Пропускная способность, измеренная в [**K**|**M**]{**bit/s**|**byte/s**}.
Значение 0 (принятое по умолчанию) означает неограниченную пропускную способность.
Единицу измерения необходимо указывать сразу после числового значения, например:
`ipfw pipe 1 config bw 300Kbit/s queue 50KBytes`
Если вместо числового значения указано имя устройства, то скорость передачи задается указанным устройством. В настоящее время только устройство **tun(4)** поддерживает такую возможность, для использования совместно с **ppp(8)**.
delay *задержка_в_миллисекундах*
Задержка при передаче, задаваемая в миллисекундах. Значение округляется до следующего кратного минимальному временному интервалу системных часов (обычно - 10 миллисекунд, но обычно имеет смысл запускать ядра с опцией "options HZ=1000", чтобы можно было задавать время с точностью до 1 миллисекунды или менее). Стандартное значение, 0, означает отсутствие задержки.
queue {*слотов* | *размер*Kbytes}
Размер очереди, в *слотах* или **Кбайтах**. По умолчанию используется 50 слотов, что является типичным размером очереди для устройств Ethernet. Учтите, что для низкоскоростных подключений следует использовать очереди небольшого размера или при передаче будут возникать существенные задержки пакетов в очереди. Так, 50 пакетов ethernet максимального размера (1500 байтов) означает 600 Кбит, или нахождение в очереди в течение 20 секунд для канала с пропускной способностью 30Kbit/s. Эффект может быть еще хуже, если пакеты принимаются с интерфейса с намного большим значением MTU, например, с интерфейса закольцовывания с его пакетами размером 16 Кбайт.
plr *вероятность_потери*
Вероятность потери пакета. Аргумент ***вероятность_потери*** - вещественное число от 0 до 1, так что 0 означает отсуствие потерь, а 1 - 100% потерю. Вероятность потери внутренне представлена 31 битом.
mask *спецификатор_маски*
Интерфейс **dummynet(4)** позволяет создавать отдельнЁе очереди для каждого потока. Идентификатор потока создается путем маскировки IP-адресов, портов и типов протоколов, заданных в конфигурации канала. Пакеты с одинаковым идентфикатором после маскировки попадают в одну очередь. Спецификаторы маски строятся как комбинации следующих компонентов: **dst-ip *маска***, **src-ip *маска***, **dst-port *маска***, **src-port *маска***, **proto *маска*** или **all**, что означает существенность всех битов во всех полях. При использовании в конфигурации *канала*, каждому потоку присваивается частота (rate), равная частоте канала. При использовании в конфигурации *очереди*, каждый поток получает вес, равный весу очереди, и все потоки, связанные с одним каналом, используют канал пропорционально их весу.

buckets *размер_хеш-таблицы*
Задает размер хеш-таблицы, используемой для хранения различных очередей. Стандартное

значение, 64, задается переменной **sysctl(8) net.inet.ip.dummynet.hash_size**, и может быть в диапазоне от 16 до 1024.

pipe *номер_канала*

Связывает очередь с указанным каналом. С одним каналом можно связать несколько очередей (обычно, с разными весами), который задает суммарную скорость передачи для набора очередей.

weight *вес*

Задает вес, который будет использоваться для потоков, соответствующих данной очереди. Вес должен быть в диапазоне 1..100, и имеет стандартное значение 1.

red | gred *w_q/min_th/max_th/max_p*

Использует алгоритм управления очередью RED. ***w_q*** и ***max_p*** - вещественные числа в диапазоне от 0 до 1 (не включая 0), а ***min_th*** и ***max_th*** - целые числа, задающие пороговые значения для алгоритма управления очередью (пороговые значения задаются в байтах, если размер очереди задавался в байтах, и в слотах в противном случае). Интерфейс **dummynet(4)** поддерживается также более умеренную (gentle) версию RED (**gred**). Для управления работой алгоритма RED можно использовать три переменных **sysctl(8)**:

net.inet.ip.dummynet.red_lookup_depth

задает точность при вычислении средней очереди, когда связь простаивает (стандартное значение - 256, должно быть больше нуля)

net.inet.ip.dummynet.red_avg_pkt_size

задает ожидаемый средний размер пакета (стандартное значение - 512, должно быть больше нуля)

net.inet.ip.dummynet.red_max_pkt_size

задает ожидаемый максимальный размер пакета, который используется только когда пороговые значения для очереди задаются в байтах (стандартное значение - 1500, должно быть больше нуля).

СПИСОК ДЛЯ ПРОВЕРКИ

При разработке правил надо учитывать следующие существенные моменты:

- Помните, что фильтруются как **входящие**, так и **исходящие** пакеты. Для большинства подключений необходим двунаправленный обмен пакетами.
- Помните о необходимости очень внимательно тестировать набор правил. При этом крайне желателен доступ к консоли. Если консоль недоступна, используйте сценарий автоматического восстановления типа **/usr/share/examples/ipfw/change_rules.sh**.
- Не забывайте про интерфейс закольцовывания (loopback interface).

ПОЛОЖИТЕЛЬНЫЕ МОМЕНТЫ

Одну разновидность пакетов брандмауэр отвергает всегда - это фрагмент TCP-пакета со смещением один. Это допустимый пакет, но используется он с единственной целью - для обходя брандмауэров. Если включена регистрация, такие пакеты регистрируются как отвергнутые правилом -1.

- При подключении по сети загрузка версии **kld(4)** утилиты **ipfw** не так проста, как может показаться. Я рекомендую использовать следующую командную строку:

```
kldload /modules/ipfw.ko && \
```

```
ipfw add 32000 allow ip from any to any
```

В этой же ситуации выполнение команды

```
ipfw flush
```

тоже не поможет.

- Список фильтров **ipfw** нельзя изменять если уровень защиты системы установлен равным 3 или выше (подробнее об уровнях защиты системы см. на странице справочного руководства **init(8)**).

ПЕРЕНАПРАВЛЕНИЕ ПАКЕТОВ

Сокет **divert(4)**, связанный с указанным портом, будет получать все пакеты, перенаправленные на этот порт. Если с целевым портом не связан ни один сокет или если ядро не было скомпилировано с поддержкой сокетов перенаправления, пакеты удаляются.

ПЕРЕМЕННЫЕ SYSCTL

Набор переменных **sysctl(8)** управляет поведением брандмауэра. Стандартные значения и описания этих переменных представлены далее:

net.inet.ip.fw.debug: 1
Управляет отладочными сообщениями, которые выдает **ipfw**.
net.inet.ip.fw.one_pass: 1
При установке значения 1 пакет, выходящий из канала **dummynet(4)** не проходит снова через брандмауэр. В противном случае, после выполнения действия канала, пакет повторно попадает на брандмауэр и проверяется, начиная со следующего правила.
net.inet.ip.fw.verbose: 1
Включает выдачу подробных сообщений.
net.inet.ip.fw.enable: 1
Включает брандмауэр. При установке значения 0 машина будет работать без брандмауэра, даже если он скомпилирован в ядро.
net.inet.ip.fw.verbose_limit: 0
Ограничивает количество сообщений, выдаваемых брандмауэром в режиме выдачи подробных сообщений.
net.inet.ip.fw.dyn_buckets: 256
net.inet.ip.fw.curr_dyn_buckets: 256
Первоначально сконфигурированный и текущий размер хеш-таблицы, использующейся для хранения динамических правил. Значение должно быть степенью 2. Размер таблицы можно менять только когда она пуста, так что для изменения размера на ходу, вероятно, придется сбросить (**flush**) и повторно загрузить набор правил.
net.inet.ip.fw.dyn_count: 3
Текущее количество динамических правил (доступна только для чтения).
net.inet.ip.fw.dyn_max: 1000
Максимальное количество динамических правил. При достижении этого предела нельзяч больше добавлять динамические правила, пока прежние не устареют.
net.inet.ip.fw.dyn_ack_lifetime: 300
net.inet.ip.fw.dyn_syn_lifetime: 20

net.inet.ip.fw.dyn_fin_lifetime: 20
net.inet.ip.fw.dyn_rst_lifetime: 5
net.inet.ip.fw.dyn_short_lifetime: 30
Эти переменные задают время существования динамических правил в секундах. При перовначальном обмене SYN-пакетами время существования устанавливается небольшим, а после получения обоих SYN-пакетов увеличивается, и снова уменьшается в ходе обмена завершающими пакетами FIN или RST.

ПРИМЕРЫ

Следующая команда добавляет запись, которая отвергает перенаправление текущим хостом всех tcp-пакетов от хоста **cracker.evil.org** на порт **telnet** хоста **wolf.tambov.su**:
```
ipfw add deny tcp from cracker.evil.org to wolf.tambov.su telnet
```
Следующая команда запрещает любое подключение из сети взломщиков к моему хосту:
```
ipfw add deny ip from 123.45.67.0/24 to my.host.org
```
Основной и эффективный способ ограничить доступ (без использования динамических правил) - использовать правила следующего типа:
```
ipfw add allow tcp from any to any established
ipfw add allow tcp from net1 portlist1 to net2 portlist2 setup
ipfw add allow tcp from net3 portlist3 to net3 portlist3 setup
...
ipfw add deny tcp from any to any
```
Первому правилу соответствуют обычные пакеты TCP, но ему не соответствует исходный пакет SYN, который будет соответствовать правилам **setup** только для избранных пар иходный адрес/целевой адрес. Все остальные пакеты SYN будут отвергаться завершающим правилом **deny**.
Для защиты сайта от атак шквалом поддельных пакетов TCP, безопаснее использовать динамические правила:
```
ipfw add check-state
ipfw add deny tcp from any to any established
ipfw add allow tcp from my-net to any setup keep-state
```
Это позволит брандмауэру устанавливать динамические правила только для подключения, которое начинается с обычного пакета SYN, поступающего из нашей сети. Динамические правила проверяются при обработке первого правила **check-state** или **keep-state**. Правило **check-state** обычно надо помещать ближе к началу набора правил, чтобы уменьшить объем работы при просмотре набора правил. Но возможны и другие решения.
УЧТИТЕ: правила, учитывающие состояние, уязвимы для атак на службы (denial-of-service attacks) путем забрасывания шквалом SYN-пакетов, что приводит к созданию множества динамических правил. Ущерб от таких атак можно частично ограничить путем установки соответствующих переменных **sysctl(8)**, которые управляют работой брандмауэра.
Вот хороший пример использования команды **list** для просмотра учетных записей и информации о временных отметках:
```
ipfw -at list
```
или в сокращенном виде, без временных отметок:
```
ipfw -a list
```
Следующее правило перенаправляет все входящие пакеты от **192.168.2.0/24** на порт 5000:
```
ipfw divert 5000 ip from 192.168.2.0/24 to any in
```
Следующие правила показывают некоторые варианты применения средств **ipfw** и **dummynet(4)** для имитации и моделирования.
Это правило теряет случайные входящие пакеты с вероятностью 5%:
```
ipfw add prob 0.05 deny ip from any to any in
```
Аналогичного эффекта можно добиться с помощью каналов **dummynet**:
```
ipfw add pipe 10 ip from any to any
ipfw pipe 10 config plr 0.05
```
Можно использовать каналы для искусственного ограничения пропускной способности, например, если на машине, работающей как маршрутизатор, необходимо ограничить объем данных, передаваемых локальными клиентами в сети **192.168.2.0/24**, можно задать правила:
```
ipfw add pipe 1 ip from 192.168.2.0/24 to any out
```

```
ipfw pipe 1 config bw 300Kbit/s queue 50KBytes
```

Обратите внимание на использование модификатора **out** - в результате правило не используется дважды. Запомните, что правила **ipfw** проверяются как для входящих, так и для исходящих пакетов.
При необходимости сымитировать двунаправленную связь с ограниченной пропускной способностью, это следует делать так:

```
ipfw add pipe 1 ip from any to any out
ipfw add pipe 2 ip from any to any in
ipfw pipe 1 config bw 500Kbit/s queue 100 red 0.002/30/80/0.1
```

Еще одно типичное приминение формирователя трафика связано с добавлением некоторых задержек при взаимодействии. Это может повлиять на многие приложения, выполняющие много вызовов удаленных процедур, когда время обмена данными (round-trip-time) по подключению часто становится намного более существенным ограничением, чем пропускная способность:

```
ipfw add pipe 1 ip from any to any out
ipfw add pipe 2 ip from any to any in
ipfw pipe 1 config delay 250ms bw 1Mbit/s
ipfw pipe 2 config delay 250ms bw 1Mbit/s
```

Отдельные очереди для потоков могут пригодиться для различных целей. Например, для подсчета объема переданных данных:

```
ipfw add pipe 1 tcp from any to any
ipfw add pipe 1 udp from any to any
ipfw add pipe 1 ip from any to any
ipfw pipe 1 config mask all
```

Представленный выше набор правил создаст очереди (и будет собирать статистическую информацию) для всех передаваемых данных. Поскольку каналы не накладывают никаких ограничений, в результате будет только собираться статистическая информация. Учтите, что необходимы 3 правила, а не только последнее, поскольку когда программа **ipfw** пытается сопоставить IP-пакеты, она не учитвает порты, поэтому нельзя будет различить подключения к разным портам.
Более сложный пример связан с ограничением объема исходящих из сети данных с ограничениями по отдельным хостам, а не по сетям:

```
ipfw add pipe 22.06.031 ip from 192.168.2.0/24 to any out
ipfw add pipe 2 ip from any to 192.168.2.0/24 in
ipfw pipe 1 config mask src-ip 0x000000ff bw 200Kbit/s queue 20Kbytes
ipfw pipe 2 config mask dst-ip 0x000000ff bw 200Kbit/s queue 20Kbytes
```

ссылки

cpp(1), **m4(1)**, **bridge(4)**, **divert(4)**, **dummynet(4)**, **ip(4)**, **ipfirewall(4)**, **protocols(5)**, **services(5)**, **init(8)**, **kldload(8)**, **reboot(8)**, **sysctl(8)**, **syslogd(8)**

ошибки

Ситаксис складывался исторически и не слишком прозрачен.
ПРЕДУПРЕЖДЕНИЕ!!ПРЕДУПРЕЖДЕНИЕ!!ПРЕДУПРЕЖДЕНИЕ!!
Эта программа может сделать работу с компьютером практически невозможной. При первом использовании работать надо за консолью и *НЕ* делать ничего, что вы не понимаете.
При изменении/добавлении записей в цепочки имена служб и протоколов испольбзовать нельзя.
Фрагменты входящих пакетов, перенаправленные с помощью **divert** или **tee**, повторно собираются перед доставкой на соответствующий сокет.
Пакеты, соответствующие правилу **tee**, не должны приниматься немедленно, а должны проходить дальше по списку правил. Эта ошибка может быть исправлена в следующей версии.

АВТОРЫ

Юрген Анцилевич (Ugen J. S. Antsilevich),
Пол-Хёнинг Камп (Poul-Henning Kamp),
Алекс Нэш (Alex Nash),
Арчи Коббс (Archie Cobbs),
Луиджи Риццо (Luigi Rizzo).
Функциональный интерфейс (API) основан на коде, написанном Дэниэлем Буле (Daniel Boulet) для BSDI.
Работу над формирователем трафика **dummynet(4)** поддерживала корпорация Akamba Corp.

ИСТОРИЯ

Утилита **ipfw** впервые появилась в ОС FreeBSD 2.0. Интерфейс **dummynet(4)** был добавлен в FreeBSD 2.2.8. Расширения, связанные с запоминанием состояния, появились в ОС FreeBSD 4.0.

FreeBSD 4.4, 16 февраля 2000 года

Источник (получено *2025-07-17 06:50*):
http://muff.kiev.ua/content/ipfw-mezhsetevoi-ekran-brandmauer-ip-i-programma-upravleniya-profilem-peredavaemoi-informats