# Пакетный фильтр PF

Данный пакетный фильтр был разработан для OpenBSD, а в настоящий момент успешно портирован в FreeBSD, NetBSD и DragonFly BSD.
Для активации пакетного фильтра в FreeBSD понадобятся следующие строки в файле /etc/rc.conf:

```
# Включение PF
pf_enable="YES"
pf_rules="/etc/pf.conf"
pf_flags=""
pflog_enable="YES"
pflog_logfile="/var/log/pflog"
pflog_flags=""
```

# Управление пакетным фильтром

Так как пакетный фильтр реализован как модуль ядра, то для управления им используется специальная утилита pfctl.

Примеры использования:

```
pfctl -f /etc/pf.conf     #  Загрузить файл pf.conf
pfctl -nf /etc/pf.conf    # Проверить синтаксис, но не загружать
pfctl -Nf /etc/pf.conf    # Загрузить только правила NAT
pfctl -Rf /etc/pf.conf    # Загрузить только фильтрующие правила
pfctl -sn                 # Показать текущие правила NAT
pfctl -sr                 # Показать текущие правила фильтра
pfctl -ss                 # Показать текущую таблицу состояний (state table)
pfctl -si                 #  Показать статистику
pfctl -sa                 # Показать все что можно
```

Для удобства управления пакетным фильтром используется скрипт: /etc/rc.d/pf
Пакетный фильтр PF при запуске считывает правила из конфигурационного файла. По умолчанию это файл /etc/pf.conf. Далее будет рассмотрен  его синтаксис.

# Списки

Списки позволяют удобным образом задать несколько похожих критериев в одном правиле. Когда pfctl встречает в конфигурационном файле список, он автоматически заменяется на несколько правил. Например:

```
block out on fxp0 from { 192.168.0.1, 10.5.32.6 } to any
```

Заменяется на

```
block out on fxp0 from 192.168.0.1 to any
block out on fxp0 from 10.5.32.6 to any
```

Списки могут быть вложенными:

```
trusted = "{ 192.168.1.2 192.168.5.36 }"
pass in inet proto tcp from { 10.10.0.0/24 $trusted } to port 22
```

Будьте осторожны с отрицаниями в списках. Следующий пример демонстрирует распространённую ошибку:

```
pass in on fxp0 from { 10.0.0.0/8, !10.1.2.3 }
```

Эта запись означает не «любой адрес из сети 10.0.0.0/8 кроме 10.1.2.3», а раскрывается в следующие два правила:

```
pass in on fxp0 from 10.0.0.0/8
pass in on fxp0 from !10.1.2.3
```

# Макросы

Макросы, это определённые пользователем переменные, которые могут содержать IP-адреса, номера портов, имена интерфейсов и т.п. Имя макроса подчиняется традиционным для большинства языков программирования правилам: начинаться оно должно с буквы, а за ней должны идти буквы, цифры или символы подчерка. Имя не должно быть зарезервированным словом, таким как pass, out или queue.

```
ext_if = "fxp0"
block in on $ext_if from any to any
```

Здесь создан макрос ext_if. Когда надо сослаться на макрос, его имя начинают со знака $.

Макросы можно вкладывать друг в друга, но, поскольку в двойных кавычках макрос указывать нельзя, следует использовать следующий синтакс:

```
host1 = "192.168.1.1"
host2 = "192.168.1.2"
all_hosts = "{" $host1 $host2 "}"
```

Макрос $all_hosts в этом примере будет раскрыт в список { 192.168.1.1, 192.168.1.2 }.

# Таблицы

Таблицы используются для хранения адресов IPv4 и/или IPv6. Поиск в них осуществляется очень быстро, они расходуют значительно меньше памяти и процессорного времени, чем списки. Таблицы, таким образом, идеальны для хранения больших массивов адресов, поскольку поиск в таблице с 50 000 записей происходит не на много медленнее, чем в таблице с 50 адресами. Таблицы можно использовать следующим образом:

- Как IP адреса источника или назначения пакета в правилах фильтрации, NAT и правилах перенаправления
- Как адреса на которые происходит трансляция в правилах NAT
- Как адреса на которые происходит перенаправление трафика;
- Как адреса назначения в правилах фильтрации для опций route-to, reply-to...

Пример (имя таблицы указывается в угловых скобках <...>):

```
table <goodguys> { 192.0.2.0/24 }
table <rfc1918> const { 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 }
table <spammers> persist
block in on fxp0 from { <rfc1918>, <spammers> } to any
pass  in on fxp0 from <goodguys> to any
```

Адреса так же можно употреблять с отрицательным знаком (!), или с ключевым словом not:

```
table <goodguys> { 192.0.2.0/24, !192.0.2.5 }
```

Таблица <goodguys>, таким образом, включает в себя всю сеть 192.0.2.0/24 кроме адреса 192.0.2.5.

# Манипулирование таблицами с помощью утилиты pfctl

Таблицей можно манипулировать при помощи утилиты pfctl(8). Например, следующая команда добавляет в таблицу <spammers> ещё одну сеть:

```
pfctl -t spammers -T add 218.70.0.0/16
```

Кроме того, указанная команда создаст таблицу <spammers>, если её ещё нет. Чтобы перечислить все вхождения в таблицу можно использовать следующую команду:

```
pfctl -t spammers -T show
```

Опция -v может использваться вместе с -Tshow для того, чтобы увидеть статистику по каждому пункту в таблице.

```
pfctl -t crackers -Ts -v
```

Чтобы удалить адрес из таблицы можно использовать команду

```
pfctl -t spammers -T delete 218.70.0.0/16
```

# Использование адресов

При поиске в таблице находится «наиболее подходящий» адрес, т.е. сеть с самой большой маской. (Наибольшая маска у самой «узкой» сети.) Например:

```
table <goodguys> { 172.16.0.0/16, !172.16.1.0/24, 172.16.1.100 }
block in on dc0 all
pass  in on dc0 from <goodguys> to any
```

Адрес источника каждого пакета пришедшего через интерфейс dc0 будет искаться в таблице <goodguys>:

- 172.16.50.5 – сеть с наибольшей маской — 172.16.0.0/16; пакет соответствует таблице и будет пропущен.
- 172.16.1.25 – сеть с наибольшей маской — !172.16.1.0/24; пакет не соответствует таблице и будет отброшен.
- 172.16.1.100 – это сеть с наибольшей маской (а точнее хост) — 172.16.1.100; пакет соответствует таблице и будет пропущен.
- 10.1.4.55 – подходящих записей в таблице нет. Пакет не соответствует таблице и будет отброшен.

# Фильтрация пакетов

Фильтрация пакетов заключается в том, что пакеты пропускаются или отбрасываются при прохождении через сетевой интерфейс в соответствии с правилами. Правила основаны на заголовках пакетов сетевого и транспортного уровней модели OSI (см. OSI). Наиболее часто используемые критерии — адреса источника и назначения, номера портов источника и назначения, протоколы.

Правила фильтра состоят из критерия и действия, которое надо предпринять, если пакет соответствует критерию. Действие может быть или block или pass. Правила применяются по очереди от первого к последнему, при этом последнее правило выигрывает если только не встретится ключеве слово quick. Таким образом, если в самом начале конфигурационного файла задано правило «пропускать все пакеты», или «отбрасывать все пакеты», то это ни что иное, как политика по умолчанию — именно это правило будет применено к пакету, который не соответствует ни одному правилу ниже по ходу файла.

# Синтаксис правил фильтрации

Ниже приведена упрощённая синтаксическая схема:

```
action [direction] [log] [quick] [on interface] [af] [proto protocol] \
    [from src_addr [port src_port]] [to dst_addr [port dst_port]] \
    [flags tcp_flags] [state]
```

Полная синтаксическая схема приведена в справочной странице man по pf.conf.

Примеры:

```
# Разрешить GRE-протокол для тунелированных соединений
pass quick inet proto gre to any keep state
# Учет специфики FTP-протокола
pass quick inet proto { tcp, udp } from any to any port $ftp_ports
keep state
pass quick inet proto { tcp, udp } from any to any port > 18000
keep state
# Только разрешенные сервисы по TCP и UDP
pass quick inet proto udp to any port $udp_services keep state
pass quick inet proto tcp to any port $tcp_services keep state
```

# Политика

Рекомендуемая практика при написании брандмауэров — делать политику «default deny», т.е. по умолчанию отбрасывать все пакеты, а потом пропускать некоторые разрешённые пакеты. Для создания политики «default deny» надо сделать следующие первые два правила:

```
block in  all
block out all
```

или

```
block all
```

Эти правила блокируют весь трафик на всех интерфейсах вне зависимости от направления.

# Разрешающие правила

Надо явно разрешить прохождение трафика, чтобы он не был заблокирован политикой. Здесь понадобятся такие критерии как номера портов источника и назначения, адреса источника и назначения, протоколы. Правила должны быть настолько строгими, насколько это возможно, для того, чтобы через брандмауэр проходил только нужный трафик.

Например:

```
# Пропускать трафик из локальной сети (192.168.0.0/24) идущий через
# интерфейс dc0 на машину OpenBSD (FreeBSD, NetBSD) с адресом 192.168.0.1,
# а так же выпускать ответный трафик.
pass in  on dc0 from 192.168.0.0/24 to 192.168.0.1
pass out on dc0 from 192.168.0.1 to 192.168.0.0/24
# Пропускать TCP трафик на интерфейсе fxp0 направленный на web сервер
# запущенный на нашей машине. Имя интерфейса
# использовано в качестве адреса назначения, поэтому правилу
# соответствуют только пакеты направленные к нам.
pass in on fxp0 proto tcp from any to fxp0 port www
```

# Ключевое слово quick

Как было замечено выше, пакеты проходят через все правила от начала до конца. Пакет, который пометился правилом как проходящий (pass) может многократно сменить действие с pass на block и обратно, в процессе прохождения через правила. Последнее правило выигрывает, если только пакет не встретит правило с ключевым словом quick, которое останавливает прохождение пакета по правилам.

Пример:

```
block in quick on fxp0 proto tcp from any to any port ssh
pass  in all
```

В данном примере трафик ssh будет отброшен немедленно, так как встретилось ключевое слово quick, и все другие правила относящиеся к трафику ssh будут проигноророваны.

Неоднозначный пример:

```
pass  in all
block in on fxp0 proto tcp from any to any port ssh
```

Такие два правила возможно приведут к блокированию трафика, а может быть и нет. В зависимости от того, какие правила окажутся ниже по тексту.

# Отслеживание состояния соединения

Одна из важных возможностей пакетного фильтра — отслеживание состояния соединений («stateful inspection»). Stateful inspection возможна благодаря возможности пакетного фильтра отслеживать состояния сетевых соединений. Состояние каждого соединения хранится в таблице состояний. Пакетный фильтр может быстро определить принадлежит ли пакет уже открытому соединению. Если пакет принадлежит уже открытому соединению, то его пропускают, не направляя на другие фильтрующие правила.

Использование таблицы состояний имеет много преимуществ: от упрощения правил фильтрации, до увеличения производительности брандмауэра. Пакетный фильтр может определить принадлежность пакета открытому соединению независимо от направления в котором идёт пакет. Это освобождает от необходимости написания правил пропускающих ответный трафик. А поскольку пакет не направляется ни на какие правила, время прохождения пакета через брандмауэр существенно уменьшается.

# Опция keep state

Если в правиле присутсвует опция keep state, первый пакет соответствующий правилу создаёт запись в таблице состояний связывающую источник и получателя пакета. Теперь не только пакеты идущие от источника к получателю, но и обратные пакеты будут соответствовать созданной записи в таблице состояний и не будут подвергаться проверке. Например:

```
pass out on fxp0 proto tcp from any to any keep state
```

Это правило разрешает любой исходящий трафик на интерфейсе fxp0, а так же разрешает прохождение всего ответного трафика через брандмауэр. Функция keep state замечательна так же тем, что значительно улучшает производительность брандмауэра, так как поиск в таблице состояний намного быстрее чем проверка пакета при помощи правил фильтрации.

# Опция modulate state

Функция modulate state  работает так же как keep state, но применима только к пакетам TCP. При использовании modulate state, начальный номер последовательности портов  исходящих соединений (ISN) выбирается случайным образом.  Правило modulate state можно употреблять не только для протокола TCP (рандомизация последовательности при этом будет работать для TCP протокола, а для UDP и ICMP будет работать keep state).

Сохранять состояние TCP, UDP и ICMP трафика, и рандомизировать ISN в TCP:

```
pass out on fxp0 proto { tcp, udp, icmp } from any to any modulate state
```

Другое преимущество таблицы состояний состоит в том, что трафик ICMP соответствующий открытому соединению тоже пропускается через брандмауэр. Например: если keep state указан для соединения TCP, и получено сообщение ICMP source-quench относящееся к данному соединению TCP, то оно будет пропущено через брандмауэр (ICMP пакет source-quench уменьшает скорость отправки пакетов через маршрутизатор и отсылается маршрутизатором или конечным хостом если, например, у них переполнен буфер).

# Network Address Translation

Когда клиент из внутренней сети пытается послать IP пакет в Интернет в этом пакете подменяется исходящий IP адрес на адрес шлюза, а так же, при необходимости, подменяется номер порта источника у пакетов TCP и UDP. Делается запись в таблице состояний.

Обратные пакеты находятся в таблице состояний и с ними проделывается аналогичное обратное преобразование.

Network Adress Translation — трансляция сетевых адресов. Один из способов скрыть целую сеть за одним адресом. Использование NAT необходимо в ситуации когда провайдер выделил адресов меньше чем компьютеров в сети. NAT описан в [RFC-1631].

Ни внутренняя машина, ни внешняя не знают о существовании NAT. Всё происходит прозрачно. Для внутренней машины NAT это просто шлюз, а внешняя машина ничего не знает о внутренней и считает, что соединение открыто шлюзом. Обнаружить NAT можно только по косвенным признакам.

# IP forward, проброс пакетов

Поскольку NAT всегда используется на шлюзах и маршрутизаторах, необходимо включить проброс пакетов. Для этого надо выставить переменную ядра net.inet.ip.forwarding в истину. Для этого во всех системах BSD используется программа sysctl(8)

```
sysctl -w net.inet.ip.forwarding=1
sysctl -w net.inet6.ip6.forwarding=1
```

Чтобы сделать эти изменения постоянными следует добавить в файл /etc/sysctl.conf такие строки:

```
net.inet.ip.forwarding=1
net.inet6.ip6.forwarding=1
```

# Конфигурирование NAT

Синтаксическая диаграмма правила NAT в pf.conf(5) выглядит следующим образом:

```
nat [pass [log]] on interface [af] from src_addr [port src_port] to \
    dst_addr [port dst_port] -> ext_addr [pool_type] [static-port]
```

В большинстве случаев для NAT трансляции годится примерно такая строка:

```
nat on tl0 from 192.168.1.0/24 to any -> 24.5.0.5
```

Это правило указывает, что надо осуществить NAT трансляцию на интерфейсе tl0 для каждого пакета пришедшего из сети 192.168.1.0/24 подменив адрес источника на 24.5.0.5.

Предыдущая строка корректна, но рекомендуется для облегчения поддержки брандмауэра использовать другую форму записи (в примере dc0 внутренний интерфейс, а tl0 внешний):

```
nat on tl0 from dc0:network to any -> tl0
```

При использовании имени интерфейса, как указано выше, адрес будет определён и подставлен когда pf.conf загружается, но не на лету. Это может вызвать проблемы, если адрес интерфейсу присваивается по DHCP и меняется во время работы. Чтобы избежать этой проблемы надо указывать адрес интерфейса в круглых скобках.

```
nat on tl0 from dc0:network to any -> (tl0)
```

Трансляция работает как для IPv4 так и для IPv6.

# Проверка состояния правил NAT

Чтобы увидеть состояния соединений подвергаемых NAT трансляции можно воспользоваться командой pfctl(8) с аргументом -s state.

```
pfctl -s state
fxp0 TCP 192.168.1.35:2132 -> 24.5.0.5:53136 -> 65.42.33.245:22 TIME_WAIT:TIME_WAIT
fxp0 UDP 192.168.1.35:2491 -> 24.5.0.5:60527 -> 24.2.68.33:53   MULTIPLE:SINGLE
```

Этот отчёт (первая срока) означает следующее:

- fxp0 – интерфейс на котором происходит трансляция.
- TCP – протокол
- 192.168.1.35:2132 – IP -адрес внутренней машины (192.168.1.35) и порт открытый на этой машине (2132). Эти параметры подменяются при трансляции.
- 24.5.0.5:53136 – внешний IP-адрес шлюза (24.5.0.5) и порт открытый на шлюзе (53136). Эти параметры подставляются при трансляции на место исходного адреса и исходного порта.
- 65.42.33.245:22 – IP -адрес назначения (65.42.33.245) и порт назначения (22).
- TIME_WAIT:TIME_WAIT – пакетный фильтр считает, что TCP соединение пребывает в указанном состоянии.

# Перенаправление пакетов, проброс портов

Если у вас работает NAT, вам доступен весь Интернет, но как быть если за шлюзом с NAT, в приватной сети находится машина доступ к которой нужен снаружи? Здесь нам поможет проброс портов. С его помощью мы можем перенаправлять входящий трафик на машину расположенную за шлюзом с NAT.

Пример:

```
rdr on tl0 proto tcp from any to any port 80 -> 192.168.1.20
```

С помощью этого правила все обращения к 80 порту будут пробрасываться на машину 192.168.1.20. Так же можно пробрасывать диапазоны портов:

```
rdr on tl0 proto tcp from any to any port 5000:5500 ->  192.168.1.20
```

Запросы пришедшие на порты с 5000 по 5500 включительно перенаправлены на хост 192.168.1.20 один к одному т.е. 5000 на 5000, 5001 на 5001 и т.д.

```
rdr on tl0 proto tcp from any to any port 5000:5500 -> 192.168.1.20 port 6000
```

Запросы пришедшие на порты с 5000 по 5500 включительно перенаправлены на хост 192.168.1.20, причём все на порт 6000.

```
rdr on tl0 proto tcp from any to any port 5000:5500 -> 192.168.1.20 port 7000:*
```

Запросы пришедшие на порты с 5000 по 5500 включительно перенаправлены на хост 192.168.1.20 со сдвигом номера порта, т.е. 5000 на 7000, 5001 на 7001 и т.д.

# Нормализация трафика (Scrub)

Нормализация трафика нужна для того, чтобы исключить неопределённость с тем куда направляется пакет. Кроме того, при нормализации собираются вместе фрагментированные пакеты, происходит защита операционных систем от некоторого вида атак и отбрасываются TCP пакеты с невозможным сочетанием флагов. Простейшая директива выглядит так:

```
scrub in all
```

Это приводит к нормализации всего входящего трафика на всех интерфейсах.
Одна из возможных причин для неиспользования нормализации — использование NFS, но ту проблему можно разрешить при использовании опции no-df. Другая причина может состоять в том, что некоторые многопользовательские сетевые игры блокируются пакетным фильтром с запущенным нормализатором. Во всех остальных случаях, кроме приведённых весьма необычных ситуаций, нормализация трафика крайне желательна.
Более сложный ример:

```
scrub in on fxp0 all fragment reassemble min-ttl 15 max-mss 1400
scrub in on fxp0 all no-df  # не трогать NFS-траффик
scrub on fxp0 all reassemble tcp
```

# Маркирование пакетов

Маркирование пакетов — способ пометить пакет внутренним идентификатором для дальнейшего использования в качестве критерия в правилах трансляции и фильтрации. Маркирование позволяет создать «доверие» между интерфейсами, а так же помогает ределить был ли пакет обработан правилами трансляции. Также становится возможным переход от фильтрации, основанной на правилах, к фильтрации, основанной на политиках. Для присвоения маркера используется ключевое слово tag:

```
pass in on $int_if all tag INTERNAL_NET keep state
```

Маркер INTERNAL_NET будет присвоен любому пакету соответствующему правилу.

Для проверки установленных маркеров используется ключевое слово tagged:

```
pass out on $ext_if tagged INT_NET keep state
```

# Источники информации


- OpenBSDs web http://www.openbsd.org/
- OpenBSDs FAQ, http://www.openbsd.org/faq/index.html
- PF User Guide http://www.openbsd.org/faq/pf/index.html
- Daniel Hartmeier's PF pages, http://www.benzedrine.cx/pf.html
- Daniel Hartmeier: Design and Performance of the OpenBSD Stateful Packet Filter (pf), http://www.benzedrine.cx/pf-paper.html
- Nate Underwood: HOWTO: Transparent Packet Filtering with OpenBSD, http://ezine.daemonnews.org/200207/transpfobsd.html
- Randal L. Schwartz: Monitoring Net Traffic with OpenBSD's Packet Filter, http://www.samag.com/documents/s=9053/sam0403j/0403j.htm
- Unix.se: Brandvдgg med OpenBSD, http://unix.se/Brandvägg_med_OpenBSD
- Randal L. Schwartz: Blog for Thu, Jan 29, 2004, http://use.perl.org/~merlyn/journal/17094
- RFC 1631, "The IP Network Address Translator (NAT)", May 1994 http://www.ietf.org/rfc/rfc1631.txt?number=1631
- RFC 1918, "Address Allocation for Private Internets", February 1996 http://www.ietf.org/rfc/rfc1918.txt?number=1918