Расширенное администрирование FreeBSD.
Блок 6.

v 1.07

Оглавление

Фильтрация пакетов (firewall-ы) во FreeBSD

В ядро FreeBSD встроена возможность фильтрации сетевых пакетов. Для управления фильтрацией пакетов необходимо использовать специальное программное обеспечение, позволяющее администратору настраивать параметры фильтрации. В зависимости от предпочтений администратора, для управления используются различные программы:

- ipfw
- ip-filter
- pf (портирован из OpenBSD)

Несмотря на наличие трех разных программ управления фильтрацией пакетов, в современных версиях ядра Вы можете использовать любую из них, так как во FreeBSD все они поддерживаются. Наиболее востребованной реализацией пакетного фильтра является pf, позволяющий с удобством использовать все возможности фильтрации пакетов современных ядер FreeBSD.

В этом блоке будет рассмотрена организации фильтрации пакетов и создание NAT преобразований при помощи pf. Данный пакетный фильтр предлагает использование систем защиты сетей, трансляцию сетевых адресов (Network Address Translation), управление трафиком и управление полосой пропускания в единственной, гибкой и дружественной администратору системе.

Пакетный фильтр и межсетевой экран

PF работает в мире, который состоит из пакетов, протоколов, подключений и портов. Анализируя порт иточника или назначения, используемый протокол и адреса PF принимает решение о том, куда пропускать пакет и пропускать ли его вообще.

Также возможна фильтрация проходящего сетевого трафика, основанного на содержании пакета, обычно называемом фильтрацией прикладного уровня, но эта функция не является отличительной особенностью PF, в частности в этом большую помощь может оказать прокси-сервер squid, описанный в предыдущем блоке.

Трансляция сетевых адресов (NAT)

Другая вещь, которая будет рассмотрена в данном блоке - это "внутренние" и "внешние" адреса, "маршрутизируемые" и "не маршрутизирумые". Этот вопрос не связан непосредственно с системами сетевой защиты и фильтрации пакетов, но мы вынуждены будем немного его коснуться. Все началось в в начале 1990-ых, когда кто-то подсчитал, сколько всего возможно пользователей Интернета.

Во времена разработки системы адресации компьютеры были большими, очень дорогими и обычно обслуживали большое количество пользователей. Тогда подключенными к сети были лишь университеты и компании с заказами Пентагона. По сути дела 32 битная адресация в 4 октетах позволила бы подключить миллиарды сетевых устройств: активного сетевого оборудования, серверов и рабочих станций пользователей.

Но тут случилась коммерциализация Интернет и в сеть попали сотни тысяч дешевых маленьких машин, в результате чего адресное пространство стало таять с катастрофической скоростью. Для решения этой проблемы был разработан протокол IP version 6, если коротко - IPv6, использующий 128 битную адресацию. FreeBSD по умолчанию поддерживает IPv6 и PF способен работать с трафиком IPv6.

Кроме того, необходимо было временное решение, так как перевод сети на новую адресацию занял бы значительное время. Найденное решение состояло из двух частей. Одной частью был механизм, позволяющий перезаписывать адреса источника/назначения на шлюзе. Другой частью выделялись участки адресного пространства, которые будут использоваться только в сетях, непосредственно не соединенных с Интернетом. Это подразумевает то, что в

разных частях света некоторые машины могут иметь одинаковый адрес, но прежде чем их трафик попадет в сеть, эти адреса будут оттранслированы маршрутизаторами.
Если трафик с такого "не маршрутизируемого" адреса захотел бы попасть в Интернет, то маршрутизаторы должны были бы отбрасывать его, как имеющий недопустимый адрес источника. Это - то, что называют "Трансляцией сетевых адресов"(NAT), иногда упоминается "подмена IP адресов", маскарадинг или нечто подобное.
Узнать диапазоны не маршрутизитуемых сетей можно в "RFC 1918 addresses".

Внимание: *Два документа определяют работу NAT: RFC 1631, "The IP Network Address Translator (NAT)", датированный маем 1994 и RFC 1918, "Address Allocation for Private Internets", ддатированный февралем 1996.*

PF сегодня

С момента выхода PF и теперь, в составе FreeBSD 7.2 PF стал пакетным фильтром, способным на очень многие вещи.
С одной стороны, PF классифицирует пакеты, основываясь на типе протокола, порта, типе пакета, источнике или адресе назначения.
И даже если NAT не является самой необходимой частью пакетного фильтра, по практическим соображениям все же в функции PF включена логика NAT.
PF способен на основе протокола, адреса и других данных пропускать трафик адресатам, таким как удаленные машины или сервисные службы.
Еще до написания PF, FreeBSD содержал код altq для балансировки нагрузки и гибкого управления трафиком, в последствии они были интегрированы между собой, но в силу нетрвиальности настройки и невысокой востребованности, данный функционал отключен в ядре Generic.
В результате этого, все возможности становятся вам доступны через один единственный, легко читаемый файл конфигурации, обычно называемый pf.conf и находящийся в каталоге /etc.
В настоящее время PF доступен как составная часть FreeBSD, начиная с 5.3, PF одна из трех доступных систем пакетной фильтрации, PF также доступен в OpenBSD, NetBSD и DragonFlyBSD.

Включение PF

Необходимо внести следующие строки в /etc/rc.conf:

```
pf_enable="YES"                 # Enable PF (load module if required)
pf_rules="/etc/pf.conf"         # rules definition file for pf
pf_flags=""                     # additional flags for pfctl startup
pflog_enable="YES"              # start pflogd(8)
pflog_logfile="/var/log/pflog"  # where pflogd should store the logfile
pflog_flags=""                  # additional flags for pflogd startup
```

Перезарузка правил PF:

```
/etc/rc.d/pf reload
```

Настройка фильтрации для рабочей станции

Теперь мы нуждаемся в некотором наборе правил фильтрации. Сейчас мы приведем пример самого простого варианта, для машины, на которой не выполняются никакие сервисы, которая подключена к локальной сети или сразу в Интернет. Итак, наш /etc/pf.conf пока будет выглядеть так:

```
block in all
pass out all keep state
```

Этими правилами будет запрещен любой входящий трафик, разрешен наш исходящий и пропущен входящий трафик, являющийся ответным на наши запросы. Если вы готовы к

использованию такого набора правил, то загружаем их:

```
/etc/rc.d/pf reload
```

Настройка фильтрации на сервере

Для написания несколько более структурированной и законченной системы правил фильтрации мы запретим весь трафик, чтобы впоследствии разрешить только необходимый. Делать это мы будем используя отличительные особенности PF: списки и макросы.
Теперь наш /etc/pf.conf будет выглядеть следующим образом:

```
block in all
# Макросы требуют определения перед использованием
tcp_services = "{ ssh, domain }"
udp_services = "{ domain }"
#Теперь мы можем продемонстрировать сразу несколько вещей - то, что макрос может быть
# списком и то, что PF понимает в качестве номера порта название службы. Именование
# служб берется из /etc/services.
pass proto tcp to any port $tcp_services keep state
pass proto udp to any port $udp_services keep state
```

Несмотря на то, что протокол UDP не предусматривает обратной связи, но PF, не смотря на это, может обработать такие соединения. Например, вы, скорее всего, захотите принять ответ от сервера имен DNS.
После выполненных изменений нам необходимо перезагрузить правила:

```
/etc/rc.d/pf reload
```

или

```
pfctl -f /etc/pf.conf
```

Если нет никаких синтаксических ошибок, то pfctl не выдаст никаких сообщений. Установленный флаг -v укажет pfctl сделать более подробный вывод.

Статистика из pfctl

В процессе работы PF возможен просмотр некоторой статистики, это делается с помощью утилиты pfctl. Для этого используется флаг "-s" и тип отображаемой информации. Следующий пример иллюстрирует возможный вывод утилиты pfctl:

```
pfctl -s info
   Status: Enabled for 6 days 01:30:14           Debug: Urgent
   Hostid: 0x9c6b095b
      Interface Stats for xl0              IPv4             IPv6
        Bytes In                      431807046                0
        Bytes Out                      40105602              352
        Packets In
          Passed                         362534                0
          Blocked                         45033                0
        Packets Out
          Passed                         285888                1
          Blocked                             1                4
      State Table                       Total             Rate
        current entries                     7
        searches                      1026325            2.0/s
        inserts                         26577            0.1/s
        removals                        26570            0.1/s
      Counters
        match                           48962            0.1/s
        bad-offset                          0            0.0/s
        fragment                           10            0.0/s
        short                              20            0.0/s
        normalize                           0            0.0/s
        memory                              0            0.0/s
        bad-timestamp                       0            0.0/s
```

В первой строке указывается, что PF работает уже несколько дней, отсчет ведется со времени последней перезагрузки. pfctl -s all даст вам еще более детальную информацию. Для получения дополнительных сведений, обратитесь к man 8 pfctl.

Для того, чтобы сделать эти изменения постоянными внесем следующие изменения в /etc/rc.conf:

```
gateway_enable="YES"
ipv6_gateway_enable="YES" # для поддержки ipv6
```

Если вы намереваетесь пропускать трафик из вашей внутренней сети наружу, то ваш /etc/pf.conf будет выглядеть следующим образом:

```
ext_if = "em0"  # macro for external interface - use tun0 for PPPoE
int_if = "le0"  # macro for internal interface
nat on $ext_if from $int_if:network to any -> ($ext_if)
block in all
tcp_services = "{ ssh, domain }"
udp_services = "{ domain }"
pass proto tcp to any port $tcp_services keep state
pass proto udp to any port $udp_services keep state
pass from { lo, $int_if:network } to any keep state
```

Обратите внимание на использование макросов для определения сетевых карт. В дальнейшем мы убедимся, что тип интерфейса не играет особой роли. Использование макросов позволяет абстрагироваться от типа интерфейса, сетевой карты или ее адреса.
Обратите внимание на правило NAT. В этом правиле мы транслируем "немаршрутизируемые" адреса нашей локальной сети в один внешний адрес сетевого интерфейса нашего маршрутизатора.
Использование выражения ($ext_if) позволяет нам использовать правило в случае, если адрес назначается динамически.
С другой стороны, это позволит проходить большему трафику, нежели могло хотеться. Ниже приведен пример макроса:

```
client_out = "{ ftp-data, ftp, ssh, domain, pop3, http, https, 139, 143, 445, 3128 }"
```

Совместно с правилом:

```
pass inet proto tcp from $int_if:network to any port $client_out flags S/SA keep state
```

Ваши реальные потребности могут отличаться и требовать разрешения других номеров портов. Кроме того, у нас имеется в запасе еще несколько интересных правил разрешения трафика, их мы покажем ниже. Например, для доступа к нашей машине из интернета по ssh:

```
pass in inet proto tcp from any to any port ssh
```

или так:

```
pass in inet proto tcp from any to $ext_if port ssh
```

Вместе со службой DNS мы разрешим службу сетевого времени(NTP), которая используется для синхронизации системных часов. Отличительная особенность обоих протоколов заключается в том, что они посылают данные и по tcp и по udp. Определим макрос:

```
udp_services = "{ domain, ntp }"
```

И добавим само правило:

```
pass quick inet proto { tcp, udp } to any port $udp_services keep state
```

Обратите внимание на ключевое слово quick. Если пакет соответствует правилу с этим ключевым словом то пакет прекращает дальнейшее прохождение по цепочке правил и к нему применяется то действие, которое указано в правиле. Использование ключевого слова quick весьма удобно, когда вам требуется сделать несколько правил-исключений.

Внимание: Для пользователей модемного соединения, ADSL или PPP over Ethernet (PPPoE) внешним интерфейсом будет tun0, вместо интерфейса типа ethernet.

Решение проблем - утилиты ping и traceroute

Написанный нами набор правил имеет один существенный недостаток - не работают средства диагностики и поиска неисправностей в сети, а именно утилиты ping и traceroute. Это не будет иметь значение для большинства ваших пользователей, особенно работающих с Windows и тех, кто считает, что утилита ping - опасная команда и что icmp должен быть вне закона. Но вам скорее всего эти утилиты будут нужны. Для успешного решения это проблемы выполним несколько простых шагов. Добавляем макрос:

```
icmp_types = "echoreq"
```

И соответствующее правило:

```
pass in inet proto icmp all icmp-type $icmp_types keep state
```

Если вам необходимо попускать и другие типы icmp пакетов, то добавьте их в макрос icmp_types.
traceroute/tracert - другая команда, которая является весьма полезной, когда ваши пользователи утверждают, что Internet не работает. Правило ниже работает с командой tracert на всех версиях Windows:

```
# allow out the default range for traceroute(8):
pass out on $ext_if inet proto udp from any to any port 33433 >< 33626 keep state
```

Если в каких-либо других операционных системах это правило не будет работать, то вам придется откорректировать его. Это решение было найдено в списке рассылки openbsd-misc и в случаях проблем с PF советую вам обратиться к архиву рассылки на http://marc.theaimsgroup.com/.

Работа с внутренними web и почтовыми серверами

Довольно обычной является ситуация, когда необходимо предоставлять внешние сервисы, а свободных маршрутизируемых адресов дря размешения серверов попросту нет, а выполнение на одной машине нескольких сервисов не самое безопасное решение. Механизмы перенаправления в PF довольно просты и позволяют легко разместить серверы внутри локальной сети. Если разместить в локальной сети сервера http, https и почтовый сервер для приема и отправки почты, то этот отрезок списка правил будет выглядеть так:

```
webserver = "172.16.1.7"
webports = "{ http, https }"
emailserver = "172.16.1.5"
email = "{ smtp, pop3, imap, imap3, imaps, pop3s }"
rdr on $ext_if proto tcp from any to any port $webports -> $webserver
rdr on $ext_if proto tcp from any to any port $email -> $emailserver
….........................
# уже имеющиеся правила
….........................
pass in on $ext_if proto tcp from any to $webserver port 80 flags S/SA synproxy state
pass in on $ext_if proto tcp from any to $emailserver port $email flags S/SA synproxy state
pass out on $ext_if proto tcp from $emailserver to any port smtp flags S/SA synproxy state
```

Обратите внимание на флажок "synproxy" в новых правилах. Это означает, что pf будет перехватывать входящие соединения и отправлять их на внутренние сервера, выполняя роль прокси-сервера.
Правила, описывающие работу с демилитаризованной зоной(DMZ), когда локальная сеть отделяется от сети, в которой размещены серверы, не будут сильно отличаться от вышеприведенных, достаточно только определить интерфейсы локальной сети и сети DMZ.

Таблицы, делающие вашу жизнь легче

Таблица - это структура, главным образом полезная как список правил, которые могут управляться, не вызывая перезагрузки основного набора правил. Имена таблиц всегда

включаются в < >, подобно этому:
```
table <clients> { 172.16.1.0/24, !172.16.1.254 }
```

Здесь в таблице содержатся адреса сети 172.16.1.0/24 за исключением адреса 172.16.1.254, что указано оператором !(логическое NOT). Также возможна загрузка таблицы из выделенного файла, например /etc/clients, значения в котором указываются по одному в строке:
```
172.16.1.0/24
!172.16.1.254
```

Имя файла в свою очередь используется, чтобы инициализировать таблицу в /etc/pf.conf:
```
table <clients> persist file «/etc/clients»
```

Тогда, например, Вы можете заменить одно из наших более ранних правил на следущее:
```
pass out inet proto tcp from <table> to any port $client_out flags S/SA keep state
```

Имея в руках такой инструмент, вы можете оперативно изменять правила фильтрации. Например, для редактирования таблицы достаточно воспользоваться командой:
```
pfctl -t clients -T add 192.168.1/16
```

Вы могли бы решить сохранять копию таблицы на диск, используя планировщик заданий cron. В качестве альтернативы, вы можете редактировать файл /etc/clients и заменить им находящуюся в памяти содержание таблицы с данными файла:
```
pfctl -t clients -T replace -f /etc/clients
```

Регистрация событий

PF позволяет вести регистрацию с помощью ключевого слова "log", указываемого после интересующего нас правила. Сделать это можно правилом:
```
pass in log (all, to pflog0) on $ext_if inet proto tcp to $ext_if port 22 keep state
```

В результате этого, весь трафик, подходящий под это правило будет зарегистрирован в формате вывода утилиты tcpdump.
```
tcpdump -n -e -ttt -i pflog0
```

Регистрация событий - очень полезная функция, но она в любом случае должна быть выборочной, так как есть опасность неконтролируемого разрастания файлов журналов. Для отслеживания текущей сетевой активности можно использовать утилиту pftop.

Пример настройки /etc/pf.conf

```
# Transparent proxy
#rdr proto tcp from any to any port 80 -> 127.0.0.1 port 3128
# Настройка макросов для интерфейсов
ext_if = "em0"  # macro for external interface - use tun0 for PPPoE
int_if = "le0"  # macro for internal interface
# Разрешаем выход из внутренней сети в интернет
nat on $ext_if from $int_if:network to any -> ($ext_if)
# Макросы требуют определения перед использованием
tcp_services = "{ ssh, smtp, domain, http, https, 821, 1723, nfsd, rpcbind }"
ftp_ports    = "{ ftp, ftp-data }"
udp_services = "{ domain, ntp, rpcbind, 821, 1723, nfsd }"
# Блокировать весь остальной входящий траффик
block in all
# 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
# Разрешить весь icmp-трафик
pass quick inet proto icmp from any to any
# Разрешить tracert из Windows-клиентов
pass out on $ext_if inet proto udp from any to any port 33433 >< 33626 keep state
# разрешить запросы к серверу NFS и RPCBIND
pass quick inet proto { tcp, udp } from any to port { nfsd, rpcbind } keep state
# mountd -p 883
pass quick inet proto { tcp, udp } from any to port 883 keep state
# rpc.lockd -p 884
pass quick inet proto { tcp, udp } from any to port 884 keep state
# rpc.statd -p 885
pass quick inet proto { tcp, udp } from any to port 885 keep state
# Протоколирование отвергнутых пакетов
block in log all
```

Вопросы

1. По каким основным параметрам можно осуществлять фильтрацию пакетов?

2. Какие задачи стоят перед брандмауэром?

3. Какие возможности открывает использование таблиц?

4. Каким образом осуществляется протоколирование пакетов ?

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