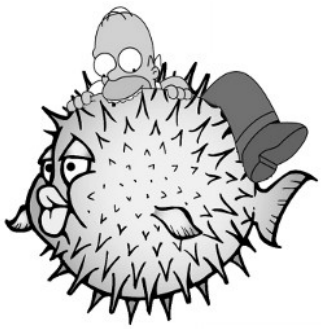
Книга о PF

by Peter N.M. Hansteen

Глава 2

PF: Основы конфигурирования

Перевод выполнил Евгений Дёмин
Homepage: unixzen.ru
Корректировка: Михайлов Алексей (iboxjo.h1.ru)



Глава 2

PF: Основы конфигурирования

В этой главе мы создадим очень простую настройку PF. Начнем с наипростейшей конфигурации, которая только возможна: единственная машина сконфигурирована для соединения с единственной сетью. Этой сетью вполне может быть Интернет.

Два основных инструмента для конфигурирования PF: Ваш любимый текстовый редактор и **pfctl -** инструмент администрирования командной строки. Конфигураця PF обычно хранится в файле **/etc/pf.conf** и называется набором правил, ппоскольку каждая строка в конфигурационном файле это правило, которое определяет, как фильтровать сетевой трафик. Повседневное управление состоит в том, чтобы редактировать свою конфигурацию в файле **/etc/pf.conf**, а после загружать изменения используя **pfctl.**

Существует веб интерфейс для управления правилами PF, но он не является частью базовой системы. Разработчики PF не враждебно настроены к этим опциям, но они не видели еще графического интерфейса для PF, который явно был бы более предпочтителен прямого редактирования pf.conf и использования команды **pfctl.**

Шаг первый: Активизация PF

Прежде чем мы перейдем к части конфигурирования сети и использования связанных с этим инструментов, нам необходимо, чтобы PF был доступен и активизирован. Детали зависят от вашей операционной системы: OpenBSD, FreeBSD или NetBSD. Проверьте вашу установку следуя инструкциям для вашей ОС и затем перейдите к разделу «Настройка простых правил PF: единственная, автономная машина» на странице 16.

Команда **pfctl** это программа, которой необходимо больше прав чем по умолчанию имеется у обычного пользователя. В оставшейся части книги, вы будете видеть команды, которым необходимы экстра привилегии, как выполняемые с префиксом sudo. Если вы ещё не использовали sudo, вам придётся это сделать. Sudo присутствует в базовой системе OpenBSD. В FreeBSD и NetBSD, sudo доступно через систему портов или систему пакетов, соответственно в **security/sudo**.

Вот несколько общих замечаний по использованию **pfctl**:

- Команда деактивации PF **pfctl –d**. Однажды введя эту команду, все основанное на pf фильтрации может быть пропущено в системе, поскольку pf будет отключен и весь трафик будет разрешен.
- Для удобства, pfctl позволяет производить несколько операций в одной командной строке. Для того, чтобы активизировать PF и загрузить установленные правила в единственной строке введите следующую команду:

sudo pfctl –ef /etc/pf.conf

Установка PF на OpenBSD

В OpenBSD 4.6 и позже, вам нет необходимости активизировать PF, поскольку PF активизирован и установлен в минимальной конфигурации по умолчанию[1].

1. Если вы устанавливаете первый раз PF конфигурацию на OpenBSD версии предыдущей чем эта, лучшим советом будет сделать модернизацию до самой последней стабильной версии. Если по какой-то причине вы должны остаться с более старой версией, вы должны будете проконсультироваться с первым изданием данной книги, а также с страницами мануалов и другой документацией для той версии, которую вы используете.

Если вы посмотрите сообщения выводимые на системную консоль в процессе загрузки системы, вы сможете заметить сообщение *pf enables*, которое появится вскоре после завершения сообщений ядра.

Если вы не увидели сообщения *pf enabled* в консоли во время загрузки, у вас есть несколько возможностей, позволяющих проверить, действительно ли PF активизирован.

Один простой путь для проверки - ввести команду, которая используется для активизации PF в командной строке, похожую на эту:

```
$ sudo pfctl –e
```

Если PF уже активизирован, система ответит на это сообщением:

```
pfctl: pf already enabled
```

Если PF не активизирован команда **pfctl –e** актвизирует PF и отобразит:

```
pf enabled
```

В версиях предшествующих OpenBSD 4.6, PF был не активизирован по умолчанию. Вы можете переопределить значение по умолчанию отредактировав ваш */etc/rc.conf.local* (или создав файл, если он не существует). Хотя в этом нет необходимости в последних версиях OpenBSD, не сложно добавить строку в ваш */etc/rc.conf.local* файл:

```
pf=YES          # enable PF
```

Если обратиться к содержимому файла */etc/pf.conf* на свежей инсталляции OpenBSD, первое, вы увидите свое первое работающее правило.

По умолчанию файл pf.conf в OpenBSD начинается с установки правила для интерфейса lo позволяющее сохранить уверенность, что трафик на интерфейсе петли не фильтруется. Следующая активная строка это правило *pass,* по умолчанию позволяющий проходить вашему сетевому трафику. И, наконец, правило *block*, блокирует удаленный трафик X11 к вашей машине.

Как вы возможно заметили, по умолчанию *pf.conf* файл также содержит несколько закомментированных строк начинающихся с хэш символа (#). В этих комментариях вы будете находить предлагаемые правила, которые намекают на полезные настройки, такие как FTP проксирование (смотрите Главу 3) и *spamd*, демон блокирования спама OpenBSD (смотрите Главу 6). Эти опции потенциально полезны в различных реальных сценариях, но, поскольку они не могут быть актуальны во всех конфигурациях, они закомментированы по умолчанию.

Если вы рассмотрите настройки связанные с PF в файле */etc/rc.conf*, вы найдете строку pf_rules=. В принципе, вы можете указывать в этом файле конфигурацию отличную от файла */etc/pf.conf*. Однако, изменение этого параметра на стоит траты времени. Используя установки по умолчанию вы даете преимущество для большого количества домашних приложений, таких как автоматическое ночное резервное копирование вашей конфигурации в */var/backups*. На OpenBSD скрипт */etc/rc* имеет встроенный механизм который поможет вам, если вы перезагрузили pf или при отсутствии файла *pf.conf* или если он содержит недопустимый набор правил. Перед активизацией любых сетевых интерфейсов, *rc* скрипт загружает правило позволяющее несколько базовых сервисов: SSH откуда угодно, базовое разрешение имен и монтирование NFS. Это позволяет вам залогиниться и исправить любые ошибки в вашем наборе правил, загрузить корректный набор правил и продолжить работу.

Установка PF на FreeBSD

Хороший код хорошо распространяется, и пользователи FreeBSD скажут вам, что хороший код откуда угодно, рано или поздно, найдет путь в FreeBSD. PF не исключение, и с FreeBSD 5.2.1 и 4.x серий и далее, PF и связанные с ним инструменты стали частью FreeBSD.

Если вы прочли предыдущий абзац по установке PF на OpenBSD, вы видели, что на OpenBSD, PF был активизирован по умолчанию. С FreeBSD дела обстоят не так, поскольку PF является только одним из трех доступных вариантов пакетного фильтра системы. В

FreeBSD вам необходимо сделать несколько явных действий для того, чтобы включить PF и при сравнении с OpenBSD, кажется, что нужно немного больше поколдовать в вашем */etc/rc.conf.* Загляните в файл */etc/defaults/rc.conf* - там показаны значения по умолчанию для FreeBSD связанные с PF :

```
pf_enable="NO"                  # Установить YES для активизации packet filter (PF)
pf_rules="/etc/pf.conf"         # файл для определения правил для PF
pf_program="/sbin/pfctl"        # путь, где находится исполняемый файл PF
pf_flags=""                     # дополнительные флаги для pfctl
pflog_enable="NO"               # установить YES для включения логирования packet filter
pflog_logfile="/var/log/pflog"  # где pflogd должен хранить logfile
pflog_program="/sbin/pflogd"    # где находится исполняемый файл pflogd
pflog_flags=""                  # дополнительные флаги для pflogd
pfsync_enable="NO"              # необходимо для синхронизации состояния pf для других хостов
pfsync_syncdev=""               # интерфейс через который работает pfsync
pfsync_ifconfig=""              # дополнительные опции для ifconfig(8) для pfsync
```

К счастью, вы можете игнорировать многие из них – по крайней мере сейчас. Следующие опции это только те, что вам необходимо добавить в вашу конфигурацию */etc/rc.conf*:

```
pf_enable="YES"                 # Включение PF (загружается модулем если нужно)
pflog_enable="YES"              # включаем pflogd(8)
```

Есть некоторые различия между FreeBSD 4.x, 5.x, и более поздних версиях в отношении PF. Ссылаясь на хэндбук FreeBSD, а именно раздел о PF главы Брандмауэры, доступный на http://www.freebsd.org ознакомьтесь с информацией которая применима в вашем случае. Код PF на FreeBSD 7 и 8 такой же, как код на OpenBSD 4.1 с некоторыми исправленными ошибками. Мы будем предполагать, что вы работаете на FreeBSD 7.0 или новей.

На FreeBSD, PF компилируется как модуль ядра по умолчанию. Если ваша система FreeBSD установлена с ядром GENERIC, вы можете запустить PF написав следующее:

```
$ sudo kldload pf
$ sudo pfctl –e
```

Предполагаю, что вы вставили строчки упомянутые в вашем */etc/rc.conf* и создали */etc/pf.conf* файл, а так же, вы должно быть используете *rc* скрипт для запуска PF. Следующие строки включают PF:

```
$ sudo /etc/rc.d/pf start
```

А эти выключают пакетный фильтр(packet filter):

```
$ sudo /etc/rc.d/pf stop
```

Замечание

На FreeBSD скрипту /etc/rc.d/pf необходима по крайней мере строка pf_enable="YES" в /etc/rc.conf и корректный /etc/pf.conf файл. Если какое либо из этих требований не выполняется, скрипт завершит работу с ошибкой. По умолчанию в инсталляции FreeBSD файла /etc/pf.conf нет, и вам нужно будет создать его перед тем как перезагружать систему с включенным PF. Для наших целей, создадим пустой /etc/pf.conf файл командой touch, но вы могли бы также работать если бы скопировали файл из /usr/share/examples/pf/pf.conf поставляемый с системой.

Файл примера */usr/share/examples/pf/pf.conf* содержит неактивные настройки. Он включает только закомментированные строки начинающиеся с символа # и закомментированные правила, что дает обзор того, как выглядят рабочие правила. Для примера, если вы удалите знак # перед строкой, которая говорит нам, что нужно пропускать все на loopback интерфейс, а затем сохраните файл как */etc/pf.conf,* вы включите PF и загрузите установленное правило на ваш loopback интерфейс, который не фильтрует трафик. Однако, если PF включен на системе FreeBSD, а мы не удосужились написать актуальные правила, PF ничего делать не будет и все пакеты будут пропущены.

На момент написания статьи (Сентябрь 2010) в *rc* скриптах FreeBSD не установлены правила по умолчанию как резервные и если конфигурация читается из */etc/pf.conf* то она не будет загружена. Это означает что включение PF без установленных правил или с содержимым *pf.conf*, которое синтаксически неверно оставит пакетный фильтр включенным с правилом по умолчанию ***пропускать все***.

Установка PF на NetBSD

На NetBSD 2.0, PF стал доступен как загружаемый модуль ядра, который может быть установлен посредством пакетов (*security/pflkm*) или скомпилирован статически в ядро. В NetBSD 3.0 и более поздних версиях, PF является часть базовой системы. На NetBSD PF - один из нескольких возможных систем пакетной фильтрации, и вам явно включить его.

Некоторые детали конфигурации PF изменились в разных NetBSD релизах. В этой книге предполагается что вы используете NetBSD 5.0[2] или более поздние версии.

Для использования загружаемого модуля PF в NetBSD добавим следующие строки в */etc/rc.conf* для включения загружаемого модуля ядра, PF и интерфейса журналирования PF соответственно.

```
lkm="YES" # do load kernel modules
pf=YES
pflogd=YES
```

Загрузить модуль PF вручную и включить PF можно введя следующие команды:

```
$ sudo modload /usr/lkm/pf.o
$ sudo pfctl -e
```

Кроме того, вы можете запустить *rc.d* скрипт для включения PF и журналирования, как показано ниже:

```
$ sudo /etc/rc.d/pf start
$ sudo /etc/rc.d/pflogd start
```

Для автоматической загрузки модуля в процессе начальной загрузки, добавьте следующие строку в */etc/lkm.conf*

```
/usr/lkm/pf.o - - - - BEFORENET
```

Если ваша файловая система /usr находится в отдельном разделе, добавьте строку в файл */etc/rc.conf:*

```
critical_filesystems_local="${critical_filesystems_local} /usr"
```

Если ошибок в этой части не возникло, значит вы успешно включили PF на вашей системе и готовы перейти к создание законченной конфигурации.

Поставляемый файл */etc/pf.conf* не содержит активных установок. Там есть только закомментированые строки, начинающиеся с хэш символа (#) и закомментированные правила, но это дает вам обзор правил, которые могут работать. Для примера, если вы удалите хэш символ перед строкой, которая говорит пропускать на loopback интерфейс, раскомментировав ее, а затем сохранив файл, вы включите PF и загрузите установленное правило, а ваш интерфейс loopback не будет фильтровать трафик, проходящий через него. Однако, даже если PF включен на вашей NetBSD системе, и мы не удосужимся написать актуальные правила, то PF не будет выполнять правила, а будет пропускать пакеты.

В NetBSD реализованы набор правил по умолчанию или резервных правил в файле */etc/defaults/pf.boot.conf*. Это набор правил предназначен только для того, чтобы система завершила загрузку в случае если файл */etc/pf.conf* не существует или содержит некорректный набор правил. Вы можете переопределить набор правил по умолчанию установив собственные настройки в */etc/pf.boot.conf.*

2. Для инструкций по использованию PF в более ранних релизах, смотрите документацию для ваших релизов, а также смотрите литературу по поддержке в списке Приложения А этой книги.

Простой набор правил PF: Для единственной, или автономной машины

Главным образом, имея общую, минимальную базу, мы начнем строить набор правил с простейшей конфигурации.

Минимальный набор правил

Самая простейшая конфигурация правил PF — набор правил для автономной машины, которая не предоставляет сервисов и соединяется только с одной сетью (которая может быть и непосредственно Интернетом). Мы начнем работать с файлом */etc/pf.conf*, который выглядит примерно как этот:

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

Такой набор правил запрещает весь входящий трафик, позволяя исходящий, и сохраняет информацию о состоянии наших соединений. PF читает правила сверху вниз; выполняется последнее правило в наборе правил приложимо к пакету или соединению.

Любое входящее в нашу систему соединение, независимо от источника, будет подпадать под правило *block in all*. Даже с этой предварительной оценкой результата будет осуществлён переход к следующему правилу (*pass out all keep state*), если трафик не будет подпадать под первый критерий (направление *out*). Не имея больше правил для оценки, статус не будет меняться и трафик будет заблокирован. В похожей манере, любое инициированное соединение исходящее от машины с этим набором правил не будет совпадать с первым правилом, а будет совпадать со вторым (еще раз неправильном направлении); совпадение с *pass* правилом, позволило пропустить. Мы будем проверять путь, которым PF оценивает правила и в каком порядке, эти вопросы более детально мы будем рассматривать в Главе 3, в контексте чуть большего набора правил.

Для любого правила, которое относится к части keep state, PF хранит информацию о соединении (включая различные счётчики и порядковые номера) как запись в *таблице состояний (state table)*. Таблица состояний - это то место, куда PF сохраняет информацию о существующих соединениях, которые уже попали под правило, и новые пакеты которые сравниваются с существующей таблицей состояния в поисках первого вхождения в таблицу. Только если пакет не совпадает с любым существующим состоянием, PF начнёт полную *оценку набора правил*, проверяя, совпал ли пакет с правилом в загруженном наборе правил. Мы можем также инструктировать PF о действиях состояния информации различными путями, но в самом простом случае как этот, наша главная цель это позволить вернуть трафик для соединений, которые возвращаются к нам.

Обратите внимание, что в OpenBSD 4.1 и более поздних версиях, по умолчанию для *pass* правила хранится информации о состояние [3] и нам не нужно сохранять состояние явно в самом простом случае как этот. Это значит, что набор правил может быть записан так:

```
# минимальный набор правил, OpenBSD 4.1 и более поздних версиях сохраняет состояние # по умолчанию
block in all
pass out all
```

Фактически, вы можете даже оставить здесь ключевое слово *all,* если вам нравится.

В других BSD системах дела обстоят примерно так же, и в оставшейся части книги, мы будем придерживаться нового стиля правил, с редкими напоминаниями, в том случае если вы используете старые системы. Само собой, мы будем считать, что весь трафик генерируемый от хоста, о котором идет речь, по сути заслуживает доверия.

Когда вы приготовитесь использовать этот набор правил, загрузите их следующей строкой:

```
$ sudo pfctl -ef /etc/pf.conf
```

3. На самом деле, новое значение по умолчанию соответствует хранению состояния флагов S/SA, гарантируя, что только инициированные SYN пакеты при установлении соединения создадут состояние, устраня некоторые загадочные сценарии ошибки.

Набор правил должен быть загружен без каких либо ошибок или предупреждений. Во всех системах, кроме самых медленных, вам должно сразу же вернуться приглашение командной строки.

Тестирование набора правил

Это неплохо протестировать ваш набор правил, это дает уверенность в том , что они работают как ожидается. Правильное тестирование станет существенным, как только вы перейдете к более сложным конфигурациям.

Для тестирования простого набора правил следует убедиться, может ли он выполнять разрешение доменных имен. Для примера, вы могли бы увидеть что хост nostarch.com вернул информацию такую как IP адрес хоста nostarch.com и имена почтовых хостов домена. Или просто проверьте можете ли вы выйти в сеть. Если вы можете соединиться с внешним веб сайтом по имени, значит набор правил выполняет разрешение доменных имен. В основном, любые службы к которым вы попытаетесь получить доступ со своей собственной системы должны работать, и любые службы, которые попытаются получить доступ к вашей система с другой машины получат сообщение отказа соединения (connection refused).

Более строгие: использование списков и макросов для удобства чтения

Набор правил в предыдущем разделе чрезвычайно прост – возможно слишком прост для практического использования. Но это полезная отправная точка для построения немного более структирированной и законченной установки. Мы начнем с запрещения всех служб и протоколов, а затем позволим только те, необходимость в которых мы имеется[4], используя спискои и макросы для лучшего удобства чтения и управления.

Список - два или более объектов одного типа, на которые вы можете сослаться в наборе правил, таких как:

```
pass proto tcp to port { 22 80 443 }
```

Здесь { 22 80 443 } - список.

Макрос является более читабельным инструментом. Если вы имеете объекты на которые нужно сослаться более одного раза в конфигурации, такие как IP адреса для важного хоста, может быть полезным определить макрос вместо их прямого указания. Для примера, вы должны определить определить этот макрос ранее в наборе правил:

```
external_mail = 192.0.2.12
```

Теперь вы можете ссылаться на этот хост как $external_mail в наборе правил:

```
pass proto tcp to $external_mail port 25
```

Эти две возможности имеют огромный потенциал для сохранения вашего набора правил более читабельными, и они являются важным фактором, который вносит вклад в общую цель - держать под контролем вашу сеть.

Базовый запрещающий набор правил

До этого момента, мы разрешали любой трафик в отношении себя. Разрешающий набор правил может быть очень полезен пока мы проверяем наши основные соединения находятся на месте, или проверяем какую-нибудь фильтрацию, которая является частью проблемы, которую мы видим. Но, как только фаза «У нас есть связь?» закончилась, пришло время начать ужесточение, чтобы создать базовую линию, которая обеспечит контроль.

4. Зачем писать набора запрещающих правил по умолчанию? Самый короткий ответ это дает вам лучший контроль. Задача пакетного фильтра держать под контролем, не дать попасться, что плохие парни могут сделать. Маркус Ранум написал очень увлекательную и информативную статью, которая называется “Шесть самых тупых идей в компьютерной безопасности” (*http://www.ranum.com/security/computer_security/editorials/dumb/index.html*)

Начнем, с добавления следующего правила в */etc/pf.conf:*

```
block all
```

Это правило полностью запрещающее и будет блокировать весь трафик во всех направлениях. Оно инициирует базовую линию правил фильтрации, которую мы будем использовать полностью во всех наборах правил в следующих нескольких главах. Мы начали с нуля, с конфигурации где нет ничего, разрешающей прохождение всего трафика. Позже мы добавим правила, которые еще немного сократят наш трафик, но мы будем делать это постепенно и таким образом это будет держать нас под твердым контролем.

Теперь, мы определим несколько макросов в наборе правил, которые используем позже:

```
tcp_services = "{ ssh, smtp, domain, www, pop3, auth, https, pop3s }"
udp_services = "{ domain }"
```

Здесь вы можете видеть как комбинация списков и макросов может использоваться для получения некоторого преимущества. Макросы могут быть списками, и как демонстрировалось в примере, PF понимает правила которые используют имена служб также хорошо, как и номера портов, перечисленных в вашем файле */etc/services*. Мы позаботимся об использовании всех этих элементов и некоторых других трюках для большей читабельности в решении сложных ситуаций, которые требуют более сложных наборов правил. Имея эти макросы, мы можем использовать их в наших правилах, которые мы будем сейчас редактировать:

```
block all
pass out proto tcp to port $tcp_services
pass proto udp to port $udp_services
```

Замечание

Не забудьте добавить сохранение состояния для этих правил, если ваша система имеет версию PF старше чем OpenBSD 4.1

Строки $tcp_services и $udp_services являются ссылками на макрос. Макросы, которые появляются в наборе правил в этом месте расширяются, когда набор правил загружается и работающий набор правил будет иметь полные списки, которые вставляются в макросы на которые ссылаются. В зависимости от точной природы макросов, они могут вызываться одним правилом с ссылкой на макрос, расширяющуюся на несколько правил.

Даже в маленьких наборах правил, таких как этот, использование макросов делает правила легче для понимания и поддержки. Количество информации, которая необходима для появления сжатого набора правил, а также с осмысленными именами макросов, логика становится понятней. Следуя логике типичного набора правил, чаще всего, мы не должны видеть полный список IP адресов или номера портов в месте каждой ссылки на макрос. С практической точки зрения, с перспективой обслуживания набора правил, важно иметь ввиду каким службам и какой протокол позволять для того, чтобы сохранить одновременно и комфортный и жесткий режим. Сохранение отдельных списков разрешенных служб в соответствии с протоколом, будет сохранять ваш набор правил функциональным и читабельным.

TCP ПРОТИВ UDP

Мы позаботились, чтобы отделить TCP службы от UDP служб. Некоторые службы работают в основном на хорошо известных номерах портов и на TCP или UDP, а также несколько альтернативных между использованием TCP и UDP в соответствии с конкретными условиями.

Два протокола совершенно различных в некоторых отношениях. TCP является ориентированным на соединение и надежным, идеальный кандидат для динамической фильтрации. В отличие от UDP, который является не без основания не ориентированным на соединение, но PF создает и поддерживает данные эквивалентно о состоянии информации для UDP трафика, чтобы обеспечить возврат разрешенного UDP трафика назад, если он соответствует существующим состояниям.

Один общий пример где состояние информации для UDP полезно для обработки разрешения имен. Когда вы запрашиваете имя сервера разрешить доменное имя по IP адресу или разрешить IP адрес по имени хоста, то разумно предположить, что вы хотите получить в ответ. Сохранение информации или функциональный эквивалент о вашем UDP трафике сделает это возможным.

Перезагрузка набора правил и просмотр ошибок

После изменений в файле *pf.conf*, нам нужно загрузить новые правила как показано далее:

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

Если нет синтаксических ошибок, *pfctl* не должен отобразить никаких сообщение в ходе загрузки правил.
Если вы предпочитаете подробный вывод, используйте *–v* флаг:

```
$ sudo pfctl -vf /etc/pf.conf
```

Когда вы используете подробный режим вывода, *pfctl* должен расширять свои макросы в отдельные правила, прежде чем вернут вас в командную строку как показано далее:

```
$ sudo pfctl -vf /etc/pf.conf
tcp_services = "{ ssh, smtp, domain, www, pop3, auth, https, pop3s }"
udp_services = "{ domain }"
block drop all
pass out proto tcp from any to any port = ssh flags S/SA keep state
pass out proto tcp from any to any port = smtp flags S/SA keep state
pass out proto tcp from any to any port = domain flags S/SA keep state
pass out proto tcp from any to any port = www flags S/SA keep state
pass out proto tcp from any to any port = pop3 flags S/SA keep state
pass out proto tcp from any to any port = auth flags S/SA keep state
pass out proto tcp from any to any port = https flags S/SA keep state
pass out proto tcp from any to any port = pop3s flags S/SA keep state
pass proto udp from any to any port = domain keep state
$ _
```

Сравните этот вывод с содержанием файла */etc/pf.conf*, который вы на самом деле написали. Для нашей единственной TCP службы правило разворачивается в восьми различных контекстах: по одному для каждой службы в списке. Одно правило UDP заботиться только об одной службе, и оно расширяется от того, что мы включили опцию по умолчанию. Обратите внимание, что правила отображаются в полном объеме, с значениями по умолчанию, такие как флаги S/SA хранящие состояние применяемое в этом месте для любой опции не указаной в явном виде. Эта конфигурация та, которая загружается на самом деле.

Проверка ваших правил

Если вы сделали внешние изменения в вашем наборе правил, проверьте их перед тем, как попытаетесь загрузить набор правил :

```
$ pfctl -nf /etc/pf.conf
```

Опция *–n* говорит PF о том, что нужно только проверить правила, без их загрузки – покажет вам что вы имеете в сухом остатке и позволит вам просмотреть и исправить возможные ошибки. Если *pfctl* найдет какие-либо синтаксические ошибки в вашем наборе правил, он выйдет с сообщением об ошибке, которая указывает на номер строки, где произошла ошибка.
Некоторые руководства по брандмауэрам посоветуют вам убедиться что ваша старая конфигурация действительно работает, или, если вы столкнулись с проблемами, брандмауэр может быть в каком-то промежуточном состоянии, которое не соответствует состоянию ни до, ни после. Это просто не верно, когда вы используете PF. Последний допустимый загруженный набор правил будет активным до тех пор пока вы не отключите PF или пока не загрузите новый набор правил. *pfctl* проверяет синтакс, а затем загружает новый набор правил полностью перед переходом на новые. Как только допустимый набор правил был загружен, нет промежуточного состояния с частичным набором правил или вообще с незагруженными правилами. Одним из следствий этого является то, что трафик, который соответсвует состоянию которые действительны в обоих, старого и нового набора правил не будет нарушена. Если вы на самом деле следовали советам из некоторых этих старых руководств и покраснели за существующие правила (это возможно, используя pfctl –F all или аналогичного) прежде чем пытаться загрузить новые из вашего конфигурационного файла, последняя допустимая конфигурация будет оставаться

загруженной. На самом деле, сброс набора правил иногда хорошая идея, так как это эффективно устанавливает ваш пакетный фильтр в режим разрешить все, и с достаточно высоким риском нарушения полезного трафика пока вы готовитесь загрузить ваши правила.

Тестирование измененного набора правил

Если у вас есть набор правил, которые *pfctl* загрузил без каких-либо ошибок, пришло время посмотреть работают ли ваши правила так, как ожидалось. Проверка разрешения имен с командой такой как $ host nostarch.com (как мы делали ранее) должно все еще работать, но выберите домен, который недоступен в последнее время (например, одной из политических партий, которая не прошла голосование), чтобы убедиться, что вы не тянете DNS-информацию из кеша. Вы должны способны серфить Web и использовать несколько связанных с почтой служб, но из-за природы этого обновленного набора правил, пытается получить доступ к TCP службам другие, чем те, которые определены (SSH, SMTP и т.д.) на любой удаленной системе не сможет. И как и наш простой набор правил, ваша система должна отказаться от всех соединений, которые не соответствуют существующим записям в таблице состояний; только обратный входящий трафик будет разрешено инициировать на этой машине.

Отображение информации о вашей системе

Тесты, которые вы выполняли должны были показать, что PF запущен и что ваши правила ведут себя так как и ожидалось. Есть несколько способов, чтобы отслеживать, что происходит в вашей работающей системе. Один из наиболее простых способов получения информации о PF это использовать уже знакомую программу *pfctl.* После того как PF включен и работает, система обновлений различных счетчиков и статистики ответственна за сетевую активность. Для подтверждения того, что PF действительно работает и просматривает статистику своей деятельности, вы можете использовать *pfctl –s*, а затем выведется информация, которую вы хотели видеть. Довольно длинный список информации которую нужно отобразить (смотрите man 8 pfctl найдите опцию -s). Мы будем возвращаться к некоторым из этих параметров в главе 8 и вдаваться в дальнейшие подробности о некоторой статистике, которую она предоставляет в главе 9, когда мы используем данные для оптимизации конфигурации, которую мы строим. Ниже приведен пример верхней части информации вывода команды *pfctl –s* (взято с моего домашнего шлюза). Информация высокого уровня о том, что система действительно пропускает трафик может быть найдена в этой верхней части.

```
$ sudo pfctl -s info
Status: Enabled for 24 days 12:11:27 Debug: err
Interface Stats for nfe0 IPv4 IPv6
Bytes In 43846385394 0
Bytes Out 20023639992 64
Packets In
Passed 49380289 0
Blocked 49530 0
Packets Out
Passed 45701100 1
Blocked 1034 0
State Table Total Rate
current entries 319
searches 178598618 84.3/s
inserts 4965347 2.3/s
removals 4965028 2.3/s
```

Первая строка pfctl вывода показывает, что PF включен и работает уже в течение чуть более трех недель, равный времени с тех пор как я последний раз производил обновление системы, которое требовало перезагрузки.

Состояние статистики интерфейса, отображается количество байт входящего и исходящего трафика обрабатываемого интерфейсом. Следующие несколько пунктов, вероятно, будут более интересны в нашем контексте, показывая количество пакетов, заблокированных или переданных в каждом направлении. Именно здесь мы находим ранние признаки фильтрующих правил, блокирующих весь трафик, которые мы писали. В этом случае, либо набор правил ожидаемого трафика хорош, либо мы имеем пользователей и гостей, которые достаточно хорошо себя ведут, при этом число пакетов

пропущенных гораздо больше тех, что были заблокированы в обоих направлениях. Следующий важный показатель работающей системы, которая обрабатывает трафик это блок статистики таблицы состояний. Состояние текущей строки таблицы показывает, что есть 319 активных состояний или соединений, в то время как в таблице состояний проходил поиск на совпадение существующих состояний в среднем чуть больше чем 84 раза в секунду, в обще сложности чуть более 178 миллионов раз счетчики были сброшены.

Вставки и удаленные счетчики показывают, сколько раз состояния были созданы и удалены, соответсвенно. Как и ожидалось, количество вставок и удалений отличаются по количеству активных состояний, счетчики показывают, что за время прошедшее со времени последнего сброса счетчиков, скорость состояний создаваемых и удаляемых с точностью совпадают до этого просмотра.

Информация здесь примерно соответствует статистике, которую вы ожидаете увидеть на шлюзе для маленькой сети сконфигурированной только с IPv4. Не существует никаких оснований для беспокойства по пакетам, пришедшим в IPv6 колонке. OpenBSD поставляется с встроенным IPv6. Во время сетевой конфигурации интерфейса, по умолчанию, TCP/IP стэк посылает IPv6 с запросом на ссылку соседнего локального адреса. В нормальной только IPv4 конфигурации, только первые несколько пакетов на самом деле пройдет, и к тому времени набор правил PF из */etc/pf.conf* полностью загружены, IPv6 пакеты блокируются блоком набора правил по умолчанию. (В данном примере они не отображаются в статистике nfe0 потому, что протокол IPv6 туннелируется через другой интерфейс).

Забегая вперед

Теперь у вас есть машина, которая может общаться с другими соединенными с интернет машинами, используя очень простой набор правил, который служит отправной точкой для контроля сетевого трафика. Как вы прочитаете эту книгу, вы узнаете, как добавить правила, которые делают различные полезные вещи.

В главе 3, мы будем расширять конфигурацию выступающую в качестве шлюза для небольшой сети. Обслуживание потребностей нескольких компьютеров имеет некоторые последствия, и мы увидим, как разрешить некоторый ICMP и UDP трафик на основе наших собственных потребностей устранения неполадок, если ничего больше нам не нужно.