

Настраиваем firewall с if_bridge на базе FreeBSD 7.0



Владимир Ляшко

Поводом для написания статьи послужил реальный опыт настройки сетевого фильтрующего моста на связке if_bridge + IPFW. Для решения описываемой задачи это было самым простым и логичным решением.

Зачем нужен if_bridge

В каких же случаях и для решения каких задач можно использовать фильтрующий мост на основе if_bridge и сетевого фильтра IPFW во FreeBSD? Вот несколько примеров:

- Иногда возникает необходимость разделить одну физическую сеть на сегменты без создания подсе-

тей и установки маршрутизаторов, в случае если вы хотите отделить один из участков своей сети с фильтрацией входящего и исходящего трафика.

- Для создания сетевого экрана в случае, когда на вашу сеть выделены диапазон IP-адресов и подключение к шлюзу провайдера,

к которому вы не имеете доступа и который не выполняет никакой фильтрации, а вам необходимо установить собственные правила игры.

- Если в локальной сети есть публичные серверы, например WEB или FTP, и необходимо разделить их и локальную сеть на отдельные

части в целях безопасности, т.е. организовать DMZ – Demilitarized Zone. Как известно, суть DMZ заключается в том, что она не входит непосредственно ни во внутреннюю, ни во внешнюю сеть и доступ к ней может осуществляться только по заранее заданным правилам межсетевого экрана. В случае, когда необходимо это реализовать без какого-то либо изменения адресного пространства сети, можно использовать фильтрующий мост на основе `if_bridge`.

Сразу замечу, что максимальные возможности фильтрации представляются именно в связке `if_bridge` + IPFW, так как мост (сетевые интерфейсы моста) работает на канальном уровне, а брандмауэр IPFW может фильтровать на `layer2`.

В моем случае необходимо было изолировать четыре машины из сегмента 192.168.1.0/24 большой локальной сети, имеющей доступ в мир через NAT шлюза провайдера, одной небольшой фирмы. Проведя кабель к ближайшему свитчу и подключив эти машины через фильтрующий мост, я получил разделение сегмента сети без изменения сетевого пространства, единственное, что сделал провайдер по моей просьбе – зарезервировал (на шлюзе работал сервер DHCP) четыре IP-адреса, чтобы было легче писать правила пакетного фильтра. То есть моя за-

дача сводилась к фильтрации входящего и исходящего трафика для диапазона из четырех IP-адресов участка сети, который будет находиться за фильтрующим мостом, на **рис. 1** это PC5, PC6, PC7, PC8.

Теперь рассмотрим порядок настройки прозрачного моста – брандмауэра в операционной системе FreeBSD 7.0.

Для поднятия моста `if_bridge` необходимо пересобрать ядро с параметром:

device	if_bridge
--------	-----------

или загрузить код драйвера при загрузке системы, для этого добавляем вот такие строки в `/boot/loader.conf`:

```
if_bridge_load="YES"
bridgestp_load="YES"
```

Ставим «YES» для включения поддержки протокола STP, если присутствует кольцевая топология сети, основной задачей этого протокола является приведение сети Ethernet с множественными связями к древовидной топологии, исключающей циклы пакетов. Происходит это путем автоматического блокирования ненужных в данный момент для полной связности портов. Про логику работы протокола можно почитать тут: <http://www.nag.ru/2005/0517/0517.shtml>.

Параметры для поддержки ядром сетевого экрана IPFW:

- **options IPFWALL** – включает в ядро код для фильтрации пакетов;
- **options IPFWALL_VERBOSE** – включает возможность вести логи по правилам фильтрации и проходящих пакетов;
- **options IPFWALL_VERBOSE_LIMIT=10** – ограничение списка пакетов, записываемых в лог для ограничения флуда на `syslog` и быстрого роста размера файла лога.

Для тех, кто впервые настраивает брандмауэр на основе IPFW, возможно, полезной будет опция: `IPFWALL_DEFAULT_TO_ACCEPT`, она добавляет разрешающее правило за номером 65000: «add allow ip from any to any». Потому что без этой опции в итоге при перезагрузке системы мы получим firewall, который по умолчанию будет иметь одно запрещающее правило за номером 65535: «deny ip from any to any», которое закроет прохождение любого IP-трафика по всем интерфейсам, что может вызвать проблемы при удаленном администрировании.

Собираем ядро, делая, как пишется в `handbook` (http://www.freebsd.org/doc/ru_RU.KOI8-R/books/handbook/kernelconfig-building.html):

```
#cd /usr/src/sys/i386/conf
#mkdir /root/kernels
#cp GENERIC /root/kernels/MOST
#ln -s /root/kernels/MOST
```

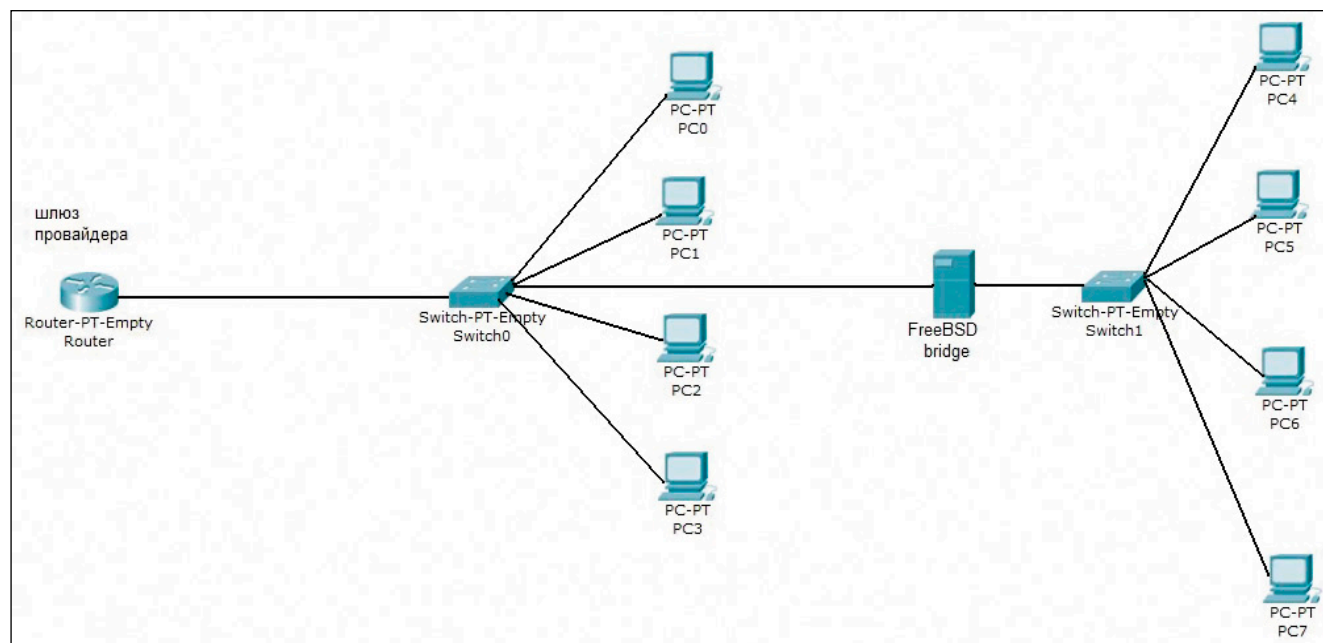


Рисунок 1. Настройка ядра и конфигурационные файлы

Правим конфиг нового ядра:

```
#cd /usr/src
#make buildkernel KERNCONF=MOST
```

Собираем:

```
#make installkernel KERNCONF=MOST
```

Инсталлируем новое ядро:

```
#ee MOST
```

Настраиваем параметры загрузки брандмауэра IPFW, редактируя файл /etc/rc.conf:

- **firewall_enable="YES"** – требуется как для загрузки IPFW в виде модуля (иначе запускать придется вручную), так и для IPFW, компилированном в ядро. «YES» установит значение системной переменной `sysctl net.inet.ip.fw.enable=1`, что укажет системе, использовать IPFW или нет, без скрипта с набором правил firewall загрузится с одним правилом по умолчанию;
- **firewall_logging="YES"** – включаем возможность логирования прохождения пакетов через брандмауэр, системная переменная `sysctl net.inet.ip.fw.verbose=1`, лог по умолчанию будет писаться в файл /var/log/security;
- **firewall_script="/etc/ip.rules"** – будем писать собственный скрипт с набором правил.

Для поднятия `if_bridge` добавляем следующие строки, назначая для одного сетевого интерфейса сетевой адрес для возможности удаленного администрирования, как рекомендуют нам здесь – http://www.freebsd.org/doc/ru_RU.KO18-R/books/handbook/network-bridging.html:

```
ifconfig_nfe0="inet 192.168.1.179 netmask 255.255.255.0"
ifconfig_r10="up"
cloned_interfaces="bridge0"
ifconfig_bridge0="addm r10 addm nfe0 up"
```

Перезагружаемся.

```
#reboot
```

Смотрим по выводу `ifconfig`, поднялся ли наш мост: должны появиться строки `member` с нашими сетевыми картами.

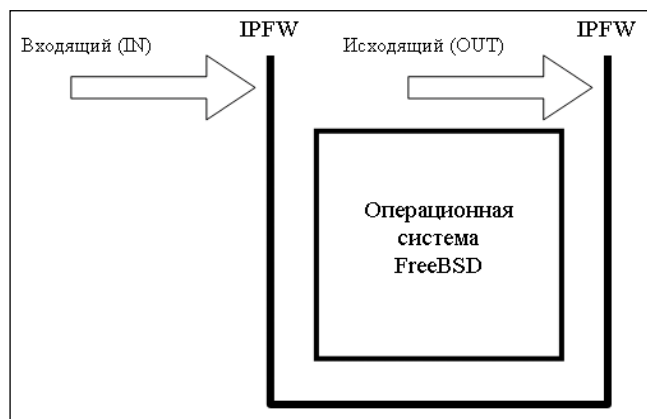


Рисунок 2. Порядок прохождения пакетов и логика работы IPFW

```
# ifconfig
```

```
r10: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=8<VLAN MTU>
ether 00:e0:4c:50:31:2d
media: Ethernet autoselect (100baseTX <full-duplex>)
status: active
nfe0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=8<VLAN MTU>
ether 00:1d:92:3f:e0:f2
inet 192.168.1.179 netmask 0xfffff000 broadcast 192.168.1.255
media: Ethernet autoselect (100baseTX <full-duplex>)
status: active
plip0: flags=108810<POINTOPOINT,SIMPLEX,MULTICAST,NEEDSGIANT> metric 0 mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
inet6 fe80::1%lo0 prefixlen 64 scopeid 0x4
inet6 ::1 prefixlen 128
inet 127.0.0.1 netmask 0xff000000
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
ether 5e:79:06:7c:cb:5e
id 00:00:00:00:00:00 priority 32768 hellotime 2 fwddelay 15
maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
root id 00:00:00:00:00:00 priority 32768 ifcost 0 port 0
member: nfe0 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
member: r10 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
```

Устанавливаем и проверяем следующие значения переменных системы:

- **#sysctl net.link.ether.ipfw=1** – переменная относится к опциям работы IPFW, значение, равное единице, включает возможность фильтрации IPFW на канальном уровне;
- **#sysctl net.link.bridge.ipfw=1** – имеет смысл только при использовании `if_bridge`, для реализации фильтрации мостом на канальном уровне необходимо установить значение, равное единице, при условии, что переменная `net.link.ether.ipfw` имеет такое же значение;
- **#sysctl net.link.bridge.pfil_member=0** – включает возможность фильтрации на сетевых интерфейсах моста, в нашем случае пакеты будут идти транзитом, через поднятый `bridge0`, и при написании правил будем описывать именно этот интерфейс, поэтому ставим значение в ноль. При тестировании правил брандмауэра было обнаружено, что, если поставить значение `net.link.bridge.pfil_member=1`, параметр `via` в правилах firewall будет работать только с `member`-сетевыми картами, а параметр вида `via bridge0` – нет.

Остальные переменные системы, относящиеся к `if_bridge`, у меня имели такой вид:

```
# ifconfig
```

```
net.link.bridge.ipfw: 1
net.link.bridge.log_stp: 0
net.link.bridge.pfil_local_phys: 0
net.link.bridge.pfil_member: 0
net.link.bridge.pfil_bridge: 0
net.link.bridge.ipfw_arp: 0
net.link.bridge.pfil_onlyip: 0
```

Чтобы после перезагрузки сохранились значения этих переменных, необходимо прописать их в /etc/sysctl.conf.

Настройка правил для firewall

Для начала вспомним три главных правила, которые определяют порядок прохождения пакетов и логику работы IPFW (см. рис. 2):

- пакет, попадая в firewall IPFW, следует согласно порядку расположения его правил до первого удовлетворяющего, где над этим IP-пакетом согласно данному

правилу совершаются какие-либо действия (пропускается, отбрасывается, возвращается обратно в firewall и т. д.);

- входящие (IN) и исходящие (OUT) пакеты следует рассматривать относительно операционной системы, а не относительно сетевых интерфейсов;
- каждый маршрутизированный IP-пакет попадает в firewall как на входе в операционную систему, так и не выходе из неё.

Теперь напишем скрипт с правилами для нашего firewall:

```
#!/bin/sh

# Объявляем переменные
fwcmd="/sbin/ipfw"
mylan="192.168.1.203, 192.168.1.201, 192.168.1.202, 192.168.1.200"
macrcp1=00:17:31:a9:e9:9b
macrcp2=00:0c:42:1c:09:bd
macrcp3=00:1d:92:3f:e0:f2
macrcp4=00:e0:4c:50:31:2d
macgate=00:0e:2e:a9:6f:9e

# MAC-адрес сетевого интерфейса моста, которому мы
# присвоили IP-адрес
ipbridge=00:50:da:4e:ba:63

# Сбрасываем все правила
${fwcmd} -f flush

# Проверяем, соответствует ли пакет динамическим правилам
${fwcmd} add check-state

# Разрешаем все установленные соединения
${fwcmd} add allow tcp from any to any established

# Разрешаем работу протоколу ARP
${fwcmd} add allow all from any to any layer2 mac-type arp

# Так как у нас есть возможность фильтровать пакеты
# на канальном уровне, воспользуемся этим

# Разрешаем трафик только от наших машин к роутеру
# провайдера и обратно
${fwcmd} add allow all from any to any MAC $macrcp1 $macgate
${fwcmd} add allow all from any to any MAC $macgate $macrcp1
${fwcmd} add allow all from any to any MAC $macrcp2 $macgate
${fwcmd} add allow all from any to any MAC $macgate $macrcp2
${fwcmd} add allow all from any to any MAC $macrcp3 $macgate
${fwcmd} add allow all from any to any MAC $macgate $macrcp3
${fwcmd} add allow all from any to any MAC $macrcp4 $macgate
${fwcmd} add allow all from any to any MAC $macgate $macrcp4

# Блокируем весь трафик на lo0
${fwcmd} add deny all from any to 127.0.0.0/8
${fwcmd} add deny all from 127.0.0.0/8 to any

# Блокируем мультикастовые рассылки
${fwcmd} add deny all from any to 240.0.0.0/4

# Запрещаем фрагментированные пакеты icmp
${fwcmd} add deny log icmp from any to 255.255.255.255

# Запрещаем адреса, которые используются в протоколах
# автоконфигурации
${fwcmd} add deny ip from 169.254.0.0/16 to any

# Разрешаем http-трафик
${fwcmd} add allow ip from $mylan to any 80 via bridge0 1
keep-state

# Разрешаем DNS-трафик
${fwcmd} add allow ip from $mylan to any 53 via bridge0 1
keep-state

# Разрешаем вход по ssh с хоста 192.168.1.200 нашей сети
${fwcmd} add allow ip from $mylan to me 22 keep-state
${fwcmd} add allow all from any to any MAC $macrcp1 1
$ipbridge
```

```
${fwcmd} add allow all from any to any MAC $ipbridge 1
$macrcp1

# Разрешаем синхронизацию времени
${fwcmd} add allow udp from any to any 123 keep-state

# Разрешаем ftp-трафик (активный режим)
${fwcmd} add allow tcp from $mylan to any 21 via bridge0
${fwcmd} add allow tcp from any 20 to $mylan via bridge0


# Разрешаем icmp - эхо-запрос, эхо-ответ и время
# жизни пакета истекло
${fwcmd} add allow icmp from any to any 1
icmp types 0,8,11 via bridge0

# Разрешаем работать с почтой
${fwcmd} add allow tcp from $mylan to any 110,143,25,995 keep-state

# Разрешаем icq
${fwcmd} add allow ip from $mylan to any 5190 keep-state

# Блокируем весь трафик не с наших MAC-адресов
${fwcmd} add deny all from any to any layer2

# Это правило добавится автоматически
${fwcmd} add deny all from any to any
```

Следует отметить, что это только пример, а какие правила писать в вашем конкретном случае, решать вам. Также следует заметить, что в данном случае (всего четыре компьютера) фильтрующий мост никак не виден в сети, задержки при прохождении пакетов через мост минимальны и ими можно пренебречь. 

1. man if_bridge.
2. man ipfw.
3. http://www.freebsd.org/doc/ru_RU.KOI8-R/books/handbook.



VPS хостинг

Удобный. Надежный. Мощный.
Для веб-сайтов, баз данных и электронной почты.




➔ PHP5, MySQL5, MS SQL, .NET, FTP,
Mail-сервер, root-доступ.

24 ЧАСА
ТЕХПОДДЕРЖКА

ДОМЕН
ЗА НАШ СЧЕТ!

(495) 799-00-18
<http://www.rusonyx.ru>