Firewall na połączeniu modemowym w FreeBSD

Marc Silver

marcs@draenor.org

$FreeBSD: release/9.1.0/pl_PL.ISO8859-2/articles/dialup-firewall/article.sgml 38826 2012-05-17 19:12:14Z hrs $

W niniejszym artykule przedstawiono instrukcję konfiguracji firewalla przy dynamicznie przydzielanym adresie IP, a także przepis na uruchomienie takiego firewalla w FreeBSD, korzystając z IPFW. Artykuł nie zawiera instrukcji konfigurowania połączenia PPP.

1. Wstęp

Firewall na połączeniu modemowym w FreeBSD

Niniejszy artykuł opisuje konfigurację firewalla w FreeBSD w przypadku, gdy adres IP przydzielany jest dynamicznie przez dostawcę usług internetowych. Dołożono wszelkich starań, aby artykuł zawierał przydatne informacje i był wolny od błędów, jednakże wszelkie uwagi i sugestie są mile widziane, proszę kierować je do <marcs@draenor.org>.

2. Opcję jądra

Na początek będziemy musieli przekompilować jądro. Wiecej informacji na temat kompilowania jądra znaleźć można w części Podręcznika poświęconej konfiguracji jądra (../../books/handbook/kernelconfig.html). Do pliku konfiguracyjnego jądra dopisujemy następujące opcje:

`options IPFIREWALL`

Właczenie obsługi firewalla w jądrze.

`options IPFIREWALL_VERBOSE`

Wysyłanie informacji o pakietach do logów systemowych.

`options IPFIREWALL_VERBOSE_LIMIT=`*`100`*

Ograniczenie liczby pakietów zapisywanych w logach; dzięki temu plik loga nie zostanie zapchany wieloma powtarzającymi się wpisami. Wartość *`100`* jest sensowna, można jednak wstawić inną, odpowiednią dla własnych potrzeb.

`options IPDIVERT`

Włączenie gniazd divert.

Można dopisać jeszcze kilka wierszy *opcjonalnych*, które zwiększają poziom bezpieczeństwa. Firewall pracuje poprawnie również bez nich, jednakże bardziej ostrożni użytkownicy mogą zechcieć z nich skorzystać.

`options TCP_DROP_SYNFIN`

Ignorowanie pakietów TCP z ustawionymi flagami SYN i FIN. Zapobiega to możliwości identyfikacji stosu TCP/IP przy pomocy narzędzi takich jak nmap, jest to jednak wbrew ustaleniom dokumentu RFC1644. Nie powinno być stosowane, jeśli na maszynie ma działać serwer WWW.

Po kompilacji jądra nie trzeba od razu przeładowywać systemu. Jeśli wszystko pójdzie zgodnie z planem, wystarczy jedno przeładowanie po ukończeniu konfiguracji firewalla.

3. Uruchamianie firewalla w `/etc/rc.conf`

Trzeba teraz wprowadzić pewne zmiany w `/etc/rc.conf`, by uwzględniał on firewalla. Dodajemy następujące linijki:

```
firewall_enable="YES"
firewall_script="/etc/firewall/fwrules"
natd_enable="YES"
natd_interface="tun0"
natd_flags="-dynamic"
```

Informacje na temat działania powyższych poleceń można znaleźć w `/etc/defaults/rc.conf` oraz rc.conf(5).

4. Wyłączenie tłumaczenia adresów przez PPP

Jeżeli wykorzystywane jest tłumaczenie adresów sieciowych wbudowane w PPP, trzeba będzie je wyłączyć. W naszych przykładach tłumaczeniem zajmuje się natd(8).

Fragment pliku odpowiedzialny za automatyczne uruchomienie PPP wygląda zapewne tak:

```
ppp_enable="YES"
ppp_mode="auto"
ppp_nat="YES"
ppp_profile="profile"
```

Jeśli tak właśnie jest, trzeba będzie wyłączyć `ppp_nat` wpisując `ppp_nat="NO"` w `/etc/rc.conf`. Ponadto należy usunąć wpisy `nat enable yes` lub `alias enable yes` w `/etc/ppp/ppp.conf`.

5. Reguły firewalla

Większość pracy mamy już za sobą. Pozostało już tylko ustalenie reguł firewalla, po czym będzie można dokonać przeładowania systemu i powinniśmy otrzymać działającego firewalla. Zdaję sobie sprawę, że zbiór reguł zależy od indywidualnych potrzeb, starałem się jednak przygotować reguły odpowiednie dla większości użytkowników łącz komutowanych. Można je oczywiście dostosować samodzielnie, traktując poniższe reguły jako punkt wyjścia.

Zacznijmy od zamkniętego firewalla: z założenia wszystkie pakiety są blokowane, przepuszczać będziemy jedynie to, co jest nam rzeczywiście potrzebne. Reguły powinny najpierw określać, co jest przepuszczane, potem co jest blokowane. Podajemy więc wszystkie reguły przepuszczające, a potem nakazujemy blokować całą resztę. :)

Stwórzmy teraz katalog `/etc/firewall`. W nim utwórzmy plik `fwrules`, zgodnie z tym, co napisaliśmy w `rc.conf`. Możemy oczywiście nazwać ten plik jak nam się żywnie podoba, proponowana tu nazwa jest jedną z możliwości.

Spójrzmy teraz na przykładowy plik firewalla, opatrzony komentarzami.

```
# Reguły firewalla
# Autor: Marc Silver (marcs@draenor.org)
# http://draenor.org/ipfw
#

# Definicja komendy firewalla (jak w /etc/rc.firewall) upraszcza
# jej wywoływanie i czyni plik bardziej czytelnym.
fwcmd="/sbin/ipfw"

# Wyczyszczenie aktualnie obowiązujących reguł.
$fwcmd -f flush

# Przekierowanie wszystkich pakietów przez interfejs tun0.
$fwcmd add divert natd all from any to any via tun0

# Przepuszczanie danych przesyłanych przez kartę sieciową i lokalnie.
# Upewnij się, że wpisałeś tu właściwą kartę (w moim przypadku fxp0)
# zanim przeładujesz system. :)
$fwcmd add allow ip from any to any via lo0
$fwcmd add allow ip from any to any via fxp0

# Przepuszczanie wszystkich połączeń nawiązywanych przez nas.
$fwcmd add allow tcp from any to any out xmit tun0 setup

# Pozwalamy, by połączenia nawiązane mogły pozostać otwarte.
$fwcmd add allow tcp from any to any via tun0 established

# Zezwolenie na połączenia z zewnątrz z określonymi usługami na
# naszej maszynie. Przykładowo dopuszczamy połączenia z ssh i apache.
$fwcmd add allow tcp from any to any 80 setup
$fwcmd add allow tcp from any to any 22 setup

# Wysyłamy RESET w odpowiedzi na pakiety ident.
$fwcmd add reset log tcp from any to any 113 in recv tun0

# Pozwalamy na wychodzące zapytania DNS do wybranych serwerów.
$fwcmd add allow udp from any to x.x.x.x 53 out xmit tun0

# I oczywiście pozwalamy im odpowiedzieć... :)
$fwcmd add allow udp from x.x.x.x 53 to any in recv tun0

# Dopuszczenie pakietów ICMP (dzięki którym działają ping i traceroute).
# Można zdecydować się na ich blokowanie, ja jednak myślę, że mi się
# przydadzą.
```

```
$fwcmd add allow icmp from any to any

# Odrzucenie całej reszty.
$fwcmd add deny log ip from any to any
```

Zbudowaliśmy w pełni sprawny firewall zezwalający na połączenia z portami 80 i 22, oraz rejestrujący próby połączenia z czymkolwiek innym. Po przeładowaniu systemu powinien już należycie funkcjonować. Jeżeli jakiekolwiek z podanych tu informacji okażą się błędne, bądź będą powodować problemy, proszę o zawiadomienie emailem. Mile widziane są również pomysły na ulepszenie niniejszej strony.

6. Pytania

1. Dlaczego korzystasz z natd(8) i ipfw(8), a nie z filtrów wbudowanych w ppp(8)?

Mówiąc szczerze, nie ma konkretnego powodu dla którego zdecydowałem się na `ipfw` i `natd`, a nie filtrowanie wbudowane w `ppp`. Dyskusje przeprowadzone z różnymi osobami doprowadziły do stwierdzenia, iż `ipfw` jest z pewnością bardziej rozbudowany i ma większe możliwości konfiguracji niż filtry `ppp`, jest jednak trudniejszy w używaniu. Jednym z powodów mojego wyboru jest to, że wolę, by firewall działał na poziomie jądra systemu, a nie programu użytkownika.

2. Otrzymuję komunikat w rodzaju `limit 100 reached on entry 2800`, po którym w logach nie pojawiają się informacje o zablokowanych pakietach. Czy mój firewall nadal działa?

Taki komunikat oznacza jedynie, że osiągnięty został limit rejestrowania reguły. Sama reguła wciąż obowiązuje, nie będzie już jednak rejestrowana, dopóki liczniki rejestrowania nie zostaną wyzerowane; można to zrobić poleceniem `ipfw resetlog`. Innym rozwiązaniem jest zwiększenie limitu w konfiguracji jądra przy pomocy opcji `IPFIREWALL_VERBOSE_LIMIT`, tak jak jest to opisane wcześniej. Limit można także ustawić zmieniając wartość net.inet.ip.fw.verbose_limit przy pomocy sysctl(8).

3. W sieci wewnętrznej korzystam z adresów z puli prywatnej, np. z zakresu 192.168.0.0, czy mogę uzupełnić reguły firewalla wpisem w rodzaju `$fwcmd add deny all from any to 192.168.0.0:255.255.0.0 via tun0` aby uniemożliwić próby połączeń z zewnątrz z lokalnymi maszynami?

Nie, ponieważ *wszystko* co przechodzi przez `tun0` podlega tłumaczeniu adresów realizowanemu przez `natd`. Pakiety przychodzące z zewnątrz trafiają wyłącznie do dynamicznie przydzielonego adresu IP, a *nie* do sieci wewnętrznej. Zauważmy jednak, że można dodać regułę w rodzaju `$fwcmd add deny all from 192.168.0.4:255.255.0.0 to any via tun0`, która zabroni maszynie w sieci wewnętrznej komunikowania się ze światem przez firewall.

4. Coś musi być nie tak. Postępowałem dokładnie według wskazówek i jestem w kropce.

W artykule przyjmujemy, że korzystamy z *userland-ppp*, reguły obowiązują więc dla interfejsu `tun0`, odpowiadającemu pierwszemu połączeniu nawiązanemu przez ppp(8) (zwanemu także *user-ppp*). Dodatkowym połączeniom odpowiadać będą interfejsy `tun1`, `tun2` itd.

W przypadku pppd(8) jest z kolei wykorzystywany interfejs `ppp0`, jeśli więc połączenie jest realizowane za pośrednictwem pppd(8), w miejscu `tun0` trzeba wstawić `ppp0`. Poniżej przedstawiona jest szybka metoda uwzględnienia tej zmiany w regułach firewalla. Oryginalny plik z regułami zachowywany jest pod nazwą `fwrules_tun0`.

```
% cd /etc/firewall
```

```
/etc/firewall% su
Password:
/etc/firewall# mv fwrules fwrules_tun0
/etc/firewall# cat fwrules_tun0 | sed s/tun0/ppp0/g > fwrules
```

By przekonać się, czy w użyciu jest ppp(8), czy pppd(8), można po nawiązaniu połączenia posłużyć się ifconfig(8). W przypadku połączenia nawiązanego przez pppd(8) zobaczylibyśmy coś w rodzaju (pomijając nieistotne informacje):

```
% ifconfig
(nieistotne...)
ppp0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1524
        inet xxx.xxx.xxx.xxx --------------> xxx.xxx.xxx.xxx netmask 0xff000000
(nieistotne...)
```

Natomiast gdy nawiązanie połączenia odbyło się za pośrednictwem ppp(8) (*user-ppp*), ujrzymy coś takiego:

```
% ifconfig
(nieistotne...)
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
(nieistotne...)
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1524
        (nieistotne IPv6...)
        inet xxx.xxx.xxx.xxx --------------> xxx.xxx.xxx.xxx netmask 0xffffff00
        Opened by PID xxxxx
(nieistotne...)
```