

Filtrowanie pakietów IP – miniHOWTO

Daniel Letkiewicz <dletkiew@pleple.ict.pwr.wroc.pl>

Abstrakt: Dokument zawiera podstawowe informacje o filtrowaniu pakietów IP za pomocą programu **ipf**. Przedstawiono możliwości tego pakietu, popierając je przykładami i opisując skrótkowo mechanizm działania.

Spis treści

1	Wstęp	3
2	Zasady filtrowania	3
2.1	Filtrowanie na podstawie adresu IP	4
2.2	Filtrowanie wybranego protokołu	4
2.2.1	Filtrowanie protokołu TCP	5
2.2.2	Filtrowanie protokołu ICMP	6
2.3	Filtrowanie na podstawie portu	6
2.4	Filtrowanie wybranego interfejsu	6
2.5	Dodatkowe możliwości filtrowania pakietów IP	7
3	Dynamiczne reguły filtrowania	8
3.1	Utrzymywanie informacji o stanie połączenia	8
3.1.1	keep state dla UDP	8
3.1.2	keep state dla ICMP	8
3.1.3	keep state dla TCP	9
3.2	Utrzymywanie danych o fragmentowanych pakietach	9
4	Generowanie pakietów zwrotnych	9
5	Grupowanie regułek	10
6	Raportowanie informacji	11
7	Podsumowanie	11
A	Dodatek	13

Niniejszy tekst został napisany na podstawie dokumentacji oraz własnych doświadczeń. Stanowi on jedynie wprowadzenie i skrócony opis możliwości filtru pakietów IP (IPF). Dodatkowe informacje można uzyskać na stronie IPF: <http://coombs.anu.edu.au/~avalon/ip-filter.html> oraz kierując pytanie do autora niniejszego tekstu.

1 Wstęp

Filtr pakietów IP, stosowany w systemach BSD, jest wygodnym narzędziem umożliwiającym kontrolę pakietów podróżujących (odbieranych/wysyłanych) przez interfejsy sieciowe. Bazując na prostym zapisie reguł, IPF potrafi zaakceptować lub odrzucić każdy wchodzący lub wychodzący pakiet spełniający podane kryteria. Może analizować pakiety na podstawie informacji znajdujących się w nagłówku pakietu IP, TCP, UDP i ICMP. Potrafi zapamiętywać m.in. dane dotyczące: fragmentacji pakietów IP, połączeń TCP, przepływu informacji dla UDP i ICMP.

2 Zasady filtrowania

Filtrowanie pakietów odbywa się na poziomie jądra systemu na podstawie zdefiniowanych przez administratora reguł filtrujących. Zarządzaniem regułami (aktywacja, dezaktywacja, usunięcie, wstawienie itd.) zajmuje się program **ipf** (zobacz dodatek). Regułki, które należy wstawić do systemu umieszcza się w pliku konfiguracyjnym, zazwyczaj jest to `/etc/ipf.rules`. Każda regułka rozpoczyna się słowem kluczowym określającym akcję jaką należy wykonać i kończy się znakiem końca linii. Akcja zostanie wykonana wówczas, gdy analizowany pakiet spełnia kryteria danej regułki.

Najczęściej wykorzystywane akcje to: **block** – odrzucić pakiet i **pass** – zaakceptuj pakiet¹. Słowa kluczowe **in** i **out** określają odpowiednio pakiet odbierany i wysyłany, **all** natomiast stosuje się do oznaczenia dowolnego pakietu.

Przykład 1. Zablokowanie wszystkich pakietów wysyłanych i akceptowanie wszystkich pakietów odbieranych na każdym interfejsie sieciowym w systemie.

```
block in all
```

¹Oprócz blokowania i przepuszczania pakietów można m.in. raportować informacje (**log**), zliczać pakiety (**count**) lub wywoływać funkcje umieszczone z jądrze systemu (**call**)

```
pass out all
```

Regułki analizowane są po kolei, tak jak występują w pliku konfiguracyjnym, i wykonywaną regułą jest pierwsza pasująca.

Przykład 2. Dwie pierwsze linie informują, że odebrany pakiet kwalifikuje się do odrzucenia ale trzecia linia decyduje o tym, aby go zaakceptować.

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

Używając słowa kluczowego `quick` możemy zarządać, aby po dopasowaniu pakie tu do regułki analiza pozostałych reguł została przerwana.

Przykład 3. Wszystkie przychodzące pakiety zostaną zablokowane. Linia druga nigdy nie będzie analizowana.

```
block in quick all
pass in all
pass out all
```

2.1 Filtrowanie na podstawie adresu IP

Nadawca i odbiorca pakietu identyfikowany jest na podstawie numeru IP znajdującego się w nagłówku pakietu IP. IPF rozpoznaje pojedyncze numery IP, jak również zakresy, definiowane liczbą bitów przypadających na maskę podsieci. Nadawca określany jest za pomocą `from` a odbiorca `to`. Słowo kluczowe `any` oznacza natomiast dowolny adres.

Przykład 4. Wszystkie pakiety wysyłane z adresu od 192.168.0.0 do 192.168.255.255 zostaną zaakceptowane, pozostałe pakiety oraz pakiety pochodzące z 192.168.0.10 zostaną odrzucone.

```
block out quick from 192.168.0.10 to any
pass out quick from 192.168.0.0/16 to any
block out all
```

2.2 Filtrowanie wybranego protokołu

Stosując słowo kluczowe `proto` możemy wyodrębnić pakiety należące do wyspecyfikowanego protokołu. Protokół może być identyfikowany poprzez nazwę,

m.in.: `tcp`, `udp`, `icmp`, jak również poprzez numer dziesiętny odpowiadający danemu protokołowi². Słowo kluczowe `tcp/udp` oznacza pakiet TCP lub UDP.

Przykład 5. Pakiety UDP wysyłane z adresu 192.168.0.10 oraz wszystkie pakiety ICMP zostaną zablokowane a pakiety protokołu IGMP (nr 2) będą akceptowane.

```
block in quick proto udp from 192.168.0.10 to any
block in quick proto icmp from any to any
pass in quick proto 2 from any to any
```

2.2.1 Filtrowanie protokołu TCP

Filtr IP pozwala na dopasowanie pakietu na podstawie flag protokołu TCP. Słowem kluczowym `flags` można określić dokładnie jakie flagi należy brać pod uwagę:

- F – flaga FIN
- S – flaga SYN
- R – flaga RST
- P – flaga PUSH
- A – flaga ACK
- U – flaga URG

Łączenie flag oraz stosowanie maski (flagi/maska) pozwala na określenie dowolnej kombinacji flag. Pole „maska” określa flagi, które należy analizować, pole „flagi” natomiast identyfikuje te flagi, które mają być ustawione. Domyślna maska wynosi: FSRPAU.

Przykład 6. Poniższe regułki blokują przychodzące pakiety TCP.

– z ustawioną tylko i wyłącznie flagą SYN

```
block in quick proto tcp flags S
```

- z ustawionymi flagami SYN i FIN, pozostałe flagi mogą być ustawione lub nie

```
block in quick proto tcp flags SF/SF
```

- z ustawioną flagą SYN i wyzerowaną flagą ACK, pozostałe flagi nie są brane pod uwagę

```
block in quick proto tcp flags S/SA
```

²Mapowanie numeru protokołu na nazwę znajduje się w pliku `/etc/protocols`, dokładny opis można znaleźć w RFC1700.

2.2.2 Filtrowanie protokołu ICMP

Pakiety protokołu ICMP mogą być rozróżniane na podstawie numeru określającego typ pakietu ICMP. Stosuje się w tym celu słowo kluczowe `icmp-type` a typ definiuje się za pomocą nazwy lub numeru dziesiętnego³.

Przykład 7. Przychodzące pakiety ICMP: z zapytaniem o echo oraz informujące o nieosiągalności portu są akceptowane, pozostałe przychodzące pakiety ICMP są blokowane.

```
pass in quick proto icmp from any to any icmp-type echo
pass in quick proto icmp from any to any icmp-type 3
block in quick proto icmp all
```

2.3 Filtrowanie na podstawie portu

W nagłówku pakietu TCP i UDP znajduje się numer portu źródłowego i numer portu docelowego. Część numerów zarezerwowana jest dla ogólnie znanych usług⁴. Za pomocą słowa kluczowego `port` można określić nazwę usługi albo numer lub zakres numerów portów, które zostaną wzięte pod uwagę przy analizie regułki.

Przykład 8. Tylko pakiety przychodzące na porty o numerach od 2000 do 2010 będą zaakceptowane.

```
blok in from any to any port < 2000
pass in from any to any port >= 2000
blok in from any to any port > 2010
```

Przykład 9. Taki sam efekt jak w poprzednim przykładzie

```
blok in from any to any port 2000 <> 2010
pass in from any to any port 1999 >< 2011
```

2.4 Filtrowanie wybranego interfejsu

Jeżeli w systemie znajduje się kilka interfejsów sieciowych to dla każdego z nich można stosować osobne reguły filtrowania.

³Spis wszystkich typów protokołu ICMP znajduje się na stronach podręcznika ekranowego (`man 5 ipf`) oraz w RFC792.

⁴Mapowanie numeru portu na nazwę usługi znajduje się w pliku `/etc/services`, dokładniejszy opis jest w RFC1700.

Przykład 10. Komputer posiada 3 interfejsy sieciowe: lo0 (loopback), xl0 (karta sieciowa 3Com Ethernet) i fxp0 (karta sieciowa Intel Ethernet). Na interfejsie fxp0 odbieramy pakiety adresowane tylko do podsieci 192.168.0.0, pozostałe pakiety blokujemy. Na interfejsie lx0 odbieramy pakiety pochodzące tylko z podsieci 192.168.0.0, pozostałe pakiety oczywiście blokujemy. Wszystko co przychodzi na interfejs lo0 oraz wszystko co wychodzi z dowolnego interfejsu akceptujemy.

```
pass in quick on fxp0 from any to 192.168.0.0/16
pass in quick on xl0 from 192.168.0.0/16 to any
block in all
pass out all
```

2.5 Dodatkowe możliwości filtrowania pakietów IP

Filtr potrafi rozpoznawać pakiety IP na podstawie typu usługi (type-of-service), czasu życia pakietu (time-to-live), bitów dotyczących fragmentacji, jak również opcji stosowanych w IP⁵.

Fragmentacja pakietów IP niesie ze sobą niebezpieczeństwo odebrania pakietu TCP lub UDP, którego nagłówek nie znajduje się w całości w jednym pakiecie IP. Możemy identyfikować pakiety, które są zbyt małe do kompletnej analizy.

Ze względu na dużą liczbę słów kluczowy po dokładny opis odsyłam do stron podręcznika ekranowego (man 5 ipf), podając w zamian kilka przykładów.

Przykład 11. Blokowanie pakietów z ustawioną opcją swobodnych punktów trasowania (loose source routing)

```
blok in all with opt lsrr
```

Przykład 12. Blokowanie pakietów z ustawioną minimalizacją połączeń w TOS i z maksymalnym czasem życia 255 przeskoków.

```
blok in tos 0x10 all
blok in ttl 255 all
```

Przykład 13. Blokowanie pofragmentowanych pakietów IP i pakietów TCP, które zawierają zbyt mało danych do analizy.

```
block in all with frag
block in proto tcp all with short
```

⁵Specyfikację protokołu IP można znaleźć w RFC791.

3 Dynamiczne reguły filtrowania

Oprócz reguł statycznych, zarządzanych za pomocą programu **ipf**, możliwe jest automatyczne generowanie reguł dynamicznych⁶. Filtr IP, po odebraniu pakietu przegląda – w pierwszej kolejności – regułki dynamiczne a następnie – jeżeli nie dopasuje analizowanego pakietu do żadnej z nich – przegląda regułki statyczne. Analogicznie postępuje z pakietem wysyłanym.

Reguły dynamiczne generowane są automatycznie po dopasowaniu pakietu do regułki statycznej ze słowem kluczowym `keep state` lub `keep frag`.

3.1 Utrzymywanie informacji o stanie połączenia

3.1.1 `keep state` dla UDP

Jeżeli hostA wysyła pakiet UDP adresowany do hostaB na portB i pakiet ten zostanie dopasowany do regułki z `keep state`, wówczas utworzona zostanie reguła dynamiczna. Będzie ona akceptowała pakiety UDP pochodzące z hostaB, z portuB i adresowane do hostaA, na portA. Zostanie usunięta po upływie 60 sekund nieaktywności.

Przykład 14. Akceptowane są tylko zapytania UDP do zewnętrznej usługi DNS. Wszystkie pakiety UDP, oprócz tych które są wysyłane w odpowiedzi na zapytania, będą blokowane.

```
block in proto udp all
pass out proto udp from any to any port = 53 keep state
```

3.1.2 `keep state` dla ICMP

Dla protokołu ICMP regułki dynamiczne działają podobnie jak dla UDP. Zapamiętywane są informacje pozwalające na zaakceptowanie właściwego pakietu zwrotnego ICMP.

Przykład 15. Akceptowane są jedynie przysyłane zapytania o echo i odpowiedzi na te zapytania.

```
pass in quick proto icmp from any to any icmp-type echo keep state
block in proto icmp all
block out proto icmp all
```

⁶Nazwa została wprowadzona przez autora i nie znajduje odzwierciedlenia w dokumentacji.

3.1.3 keep state dla TCP

Dzięki regułom dynamicznym, IPF potrafi kontrolować połączenie TCP od chwili jego rozpoczęcia aż do zakończenia. Reguła dynamiczna, odpowiedzialna za dane połączenie jest modyfikowana w taki sposób, aby akceptować tylko te pakiety, które w danym momencie są właściwe z punktu widzenia poprawności protokołu TCP. Oprócz sprawdzania adresów IP i portów (tak jak ma to miejsce w przypadku UDP), brane są również pod uwagę flagi, numery: sekwencyjny i potwierdzenia oraz rozmiar okna.

Przykład 16. Akceptowane są jedynie zewnętrzne połączenia do usługi WWW.

```
pass in quick proto tcp from any to any port = 80 flags S keep state
block in proto tcp all
block out proto tcp all
```

3.2 Utrzymywanie danych o fragmentowanych pakietach

Poniższa regułka zaakceptuje pakiety UDP przychodzące na port 2049.

```
pass in proto udp from any to any port = 2049 keep state
```

Jeżeli jednak po drodze zostanie wykonana fragmentacja pakietu IP to filtr nie będzie wiedział, że odebrane, fragmenty należą do pakietu UDP (który jest kierowany do portu 2049) i ich nie zaakceptuje.

Stosując `keep frag` możemy zarządzać aby filtr IP pamiętał informacje o fragmentacji pakietów i wykorzystywał je przy analizie regułek.

Przykład 17. Akceptuje pakiety przychodzące na port 2049, również te które zostały pofragmentowane.

```
pass in proto udp from any to any port = 2049 keep state keep frag
```

4 Generowanie pakietów zwrotnych

Domyślnie, po odrzuceniu pakietu, nie jest wysyłana żadna informacja zwrotna do nadawcy. Standardowe zachowanie się protokołu IP jest inne. Po odebraniu „nieoczekiwanego” pakietu TCP, wysłany jest pakiet z ustawioną flagą RST. Jeżeli natomiast zostanie odebrany pakiet UDP adresowany do portu,

na którym nie nasłuchuje żadna aplikacja, to wysyłany jest pakiet ICMP o typie i kodzie równym 3. Łatwo jest zatem wykryć filtr, który jedynie odrzuca pakiety.

Za pomocą `return-rst` można zarządzać, aby do nadawcy został wysłany pakiet TCP resetujący połączenie. Zastosowanie `return-icmp(port-unr)` spowoduje wysłanie pakietu ICMP informującego o nieosiągalności portu a `return-icmp-as-dest(port-unr)` dodatkowo użyje źródłowego adresu IP z odebranego pakietu, zamiast właściwego adresu IP z interfejsu sieciowego. Korzystając z `return-icmp(kod-icmp)` można wygenerować dowolny pakiet ICMP ale najbardziej odpowiednim, w tej sytuacji, jest `port-unr`.

Przykład 18. Wszystkie pakiety pochodzące z adresu 192.168.0.10 zostaną odrzucone. Jeżeli był to pakiet TCP kierowany do portu 80 to zostanie wysłany pakiet resetujący połączenie, jeżeli natomiast był to pakiet UDP na port 111 to zostanie wysłany pakiet ICMP informujący o nieosiągalności portu.

```
block return-rst in proto TCP from 192.168.0.10 to any port = 80
block return-icmp(port-unr) in proto UDP from 192.168.0.10 to any port = 111
block in from 192.168.0.10 to any
```

5 Grupowanie regułek

Wadą sekwencyjnego przeglądania regułek jest to, że wraz ze wzrostem ich liczby rośnie czas jaki filtr musi poświęcić na ich analizę. Można próbować poprawić wydajność stosując `quick` i umieszczając najczęściej dopasowywane regułki na początku pliku konfiguracyjnego.

Innym, bardziej efektywnym rozwiązaniem jest grupowanie regułek. Po dopasowaniu pakietu do reguły ze słowem kluczowym `head` numer-grupy filtr będzie analizował jedynie te regułki, które zostały przydzielone do tej samej grupy za pomocą `group` numer-grupy. Domyślnie wszystkie regułki należą do grupy 0.

Przykład 19. Pakiety odbierane na interfejsie `x10` będą analizowane przez regułki w liniach 1,2,3,4 a pakiety wysyłane z interfejsu `x11` zostaną poddane analizie przez regułki w liniach 5,6,7.

```
block in quick on x10 all head 1
block in quick on x10 from 192.168.0.0/16 to any group 1
block in quick on x10 from 10.0.0.0/8 to any group 1
pass in on x10 all group 1
```

```
block out quick on xl1 all head 2
pass out quick on xl1 from any to any port = 80 group 2
pass out quick on xl1 from any to any port = 53 group 2
```

6 Raportowanie informacji

Ważną cechą każdego filtra pakietów jest odnotowywanie wydarzeń, które uznajemy za istotne. Za pomocą słowa kluczowego **log** możemy uzyskać informację o dopasowaniu pakietu do danej regułki. Filtr IP może odnotować również pierwsze 128 bajtów pakietu (**body**) lub zanotować tylko pierwszy pakiet połączenia (**first**) zapamiętanego za pomocą regułki dynamicznej.

Jeżeli dane mają być przetwarzane przez demona `syslog` to możemy zdefiniować nazwę podsystemu i/lub poziom, z jakim zostaną one wygenerowane. (`loglevel nazwa-podsystemu.poziom`)

Domyślnie logowane dane zapisywane są do urządzenia `/dev/ipl`. Do odczytywania i przetwarzania tych danych można użyć programu **ipmon** (zobacz dodatek).

Przykład 20. Odnotowanie informacji o pakietach UDP przychodzących na port 2049

```
blok in log proto udp from any to any port = 2049
```

Przykład 21. Zapisanie pierwszych 128 bajtów każdego pakietu TCP z ustawioną jednocześnie flagą SYN i RST.

```
pass in log body proto tcp from any to any flags SR/SR
```

Przykład 22. Poinformowanie o nawiązaniu połączenia TCP z portem 79.

```
pass in log first proto tcp from any to any port = 79 keep state
```

7 Podsumowanie

W ramach podsumowania prosty przykład pliku konfiguracyjnego dla programu **ipf**. Komputer wyposażony jest w kartę Ethernet i udostępnia usługi SSH i WWW, użyty adres IP (111.111.111.111) ma znaczenie jedynie symboliczne.

```
### Zezwolenie na ruch na interfejsie loopback
pass in quick on lo0
pass out quick on lo0

### Blokowanie pakietów fragmentowanych
block in log quick proto tcp all with frag
### Blokowanie pakietów z ustawioną opcją swobodnych
### lub dokładnych punktów trasowania
block in log quick on fxp0 all with opt ssrr
block in log quick on fxp0 all with opt lsrr
### Blokowanie pakietów, które nie powinny się pojawić
### źródłowy adres IP zarezerwowany dla sieci lokalnych
block in log quick on fxp0 from 10.0.0.0/8 to any
block in log quick on fxp0 from 127.0.0.0/8 to any
block in log quick on fxp0 from 172.16.0.0/12 to any
block in log quick on fxp0 from 192.168.0.0/16 to any
block in log quick on fxp0 from 111.111.111.111 to any
### docelowy adresy IP zarezerwowany dla sieci lokalnych
block out log quick on fxp0 from any to 10.0.0.0/8
block out log quick on fxp0 from any to 127.0.0.0/8
block out log quick on fxp0 from any to 172.16.0.0/12
block out log quick on fxp0 from any to 192.168.0.0/16

### zezwolenie na odbieranie informacji o niedostępnych portach
pass in quick on fxp0 proto icmp from any to 111.111.111.111 icmp-type unreachable
### zezwolenie na zapytanie o echo za pomocą ICMP
pass in quick on fxp0 proto icmp from any to 111.111.111.111 icmp-type echo keep state
### zezwolenie na nawiązanie połączenia z usługą SSH i WWW
pass in quick on fxp0 proto tcp from any to 111.111.111.111 port = 22 flags S keep state
pass in quick on fxp0 proto tcp from any to 111.111.111.111 port = 80 flags S keep state

### Blokowanie TCP z wysyłaniem pakietu resetującego połączenie
block return-rst in log quick on fxp0 proto tcp all
### Blokowanie UDP z wysyłaniem pakietu ICMP typu host nieosiągalny
block return-icmp(3) in log quick on fxp0 proto udp all
### Blokowanie ICMP
block in quick on fxp0 proto icmp all

### Zezwolenie na nawiązywanie dowolnych połączeń TCP
pass out quick on fxp0 proto tcp from 111.111.111.111 to any keep state keep frags
### Zezwolenie na korzystanie z dowolnych usług opartych na UDP
pass out quick on fxp0 proto udp from 111.111.111.111 to any keep state keep frags
### Zezwolenie na wysyłanie pakietów ICMP
pass out quick on fxp0 proto icmp from 111.111.111.111 to any keep state
### Zablokowanie wszystkich pozostałych pakietów.
block out log quick on fxp0 all
```

A Dodatek

Rozdział zawiera skrótowe informacje o opcjach programów: **ipf**, **ipmon** i **ipfstat**.

Program **ipf** pozwala na zarządzanie regułkami. Oto kilka użytecznych opcji:

- A zastosuj zmiany w zbiorze aktywnych reguł
- I zastosuj zmiany w zbiorze nieaktywnych reguł
- D Zablokuj filtr
- E Odblokuj filtr
- F [i|o|a] usuń regułki: 'i' – wejściowe, 'o' – wyjściowe, 'a' – wszystkie
- F [S|s] usuń regułki dynamiczne: 's' – dla nieustalonych połączeń TCP, 'S' – wszystkie
- f nazwa-pliku czytaj regułki z podanego pliku
- s przełącz się pomiędzy zbiorami aktywnych i nieaktywnych regułek

Program **ipmon** odczytuje raportowanie dane i przetwarza je do czytelnej postaci. Potrafi czytać informacje pochodzące m.in. z regułek statycznych (/dev/ipl) i regułek dynamicznych (/dev/ipstate). Może pracować jako demon oraz jako proces „standardowy”. Użyteczne opcje:

- D utwórz proces potomny i pracuj jako demon
- n jeżeli jest to możliwe zamieniaj numery IP i numery portów na nazwy.
- o [N|S|I] czytaj dane pochodzące z: 'N' – /dev/ipnat, 'S' – /dev/ipstate, 'I' – /dev/ipl
- s wysyłaj informacje do demona syslog
- X pokazuj zawartość pakietu w postaci szesnastkowej
- x pokazuj nagłówki pakietu w postaci szesnastkowej

Program **ipfstat** wyświetla bieżące informacje statystyczne dotyczące filtrowanych pakietów. Przydatne opcje:

- I przełącz się pomiędzy regułkami aktywnymi i nieaktywnymi
- f pokaż statystykę fragmentowanych pakietów
- i pokaż listę regułek wejściowych
- n ponumeruj regułki
- o pokaż listę regułek wyjściowych
- s pokaż statystykę regułek dynamicznych