

# Filtrowanie pakietów w Linuksie 2.4

**Rusty Russell, lista pocztowa [netfilter@lists.samba.org](mailto:netfilter@lists.samba.org)**

Wersja oryginalna: 1.24, 2002/01/14 09:35:13

Oryginał tego dokumentu znajduje się pod adresem: <http://netfilter.filewatcher.org/>

**Tłumaczenie: Łukasz Bromirski, [l.bromirski@mr0vka.eu.org](mailto:l.bromirski@mr0vka.eu.org)**

Wersja tłumaczenia: 2.1, \$Date: 2002/03/19 22:34:43 \$

Oryginał tego tłumaczenia znajduje się pod adresem: <http://mr0vka.eu.org/tlumaczenia/linux24-pf.html>

---

*Dokument ten opisuje zastosowanie narzędzia iptables w filtrowaniu niepożądanych pakietów, w linuxie z jądrem 2.4.*

---

## 1. Wprowadzenie

Witam Szanownego czytelnika.

Zakładam, że wiesz co to adres IP, adres sieciowy, maska sieciowa, ruting i DNS. Jeśli nie, polecam przeczytanie HOWTO Network Concepts.

To HOWTO oscyluje na granicy delikatnego wprowadzenia (które sprawi, że poczujesz się zadowolony z siebie, ale niezabezpieczony przed Światem Rzeczywistym) i surowego, pełnego opisu (które oszczędzi tylko największych twardzieli, a reszta będzie zmieszana, w stanie paranoi i poszukiwać będzie ciężkiego uzbrojenia).

Twoja sieć nie jest **bezpieczna**. Problem umożliwienia szybkiej i wygodnej wymiany informacji a jednocześnie ograniczenia jej tylko do właściwych zastosowań, jest porównywalny do innych problemów takich jak zapewnienie wolność wypowiedzi i jednocześnie zapobieganiu wznoszenia okrzyków w stylu 'Pali się!' w zatłoczonym kinie. Ten problem nie zostanie rozwiązany w tym HOWTO.

Zatem tylko ty możesz zdecydować jak wyglądać będzie kompromis. Spróbuję poinstruować cię w użytkowaniu dostępnych narzędzi, poinformuję o pewnych słabościach z których trzeba sobie zdawać sprawę, a wszystko to w nadziei że użyjesz tych informacji dla dobrych, a nie złych celów. Kolejny problem porównywalny z tym powyżej.

(C) 2000 Paul 'Rusty' Russell. Na licencji GNU GPL.

## 2. Gdzie jest oficjalna strona WWW? Czy jest lista e-mail?

Są trzy oficjalne strony:

- Dzięki [Filewatcher](#).
- Dzięki [Zespołowi Samba i SGI](#).
- Dzięki [Harald'owi Welte](#).

Możesz osiągnąć je na zasadzie równoważenia obciążenia przez DNS round-robin, wybierając adres <http://www.netfilter.org/> i <http://www.iptables.org>.

Oficjalna lista pocztowa znajduje się na [serwerze list Samba](#).

## 3. To co to jest Filtr Pakietów?

Filtr pakietów to takie oprogramowanie, które sprawdza **nagłówki** (ang. *header*) pakietów w trakcie jak przechodzą przez maszynę na której działa i decyduje o ich losie. Może zdecydować, że pakiet zostanie **odrzucony** (ang. *DROP*, tzn. tak jakby pakiet nigdy nie dotarł), **zaakceptowany** (ang. *ACCEPT*, tzn. pozwoli mu przejść), lub coś bardziej skomplikowanego.

W Linuksie, filtrowanie pakietów jest wbudowane w kernel (jako moduł lub po prostu wbudowane) i jest parę jeszcze sprytniejszych rzeczy które możesz zrobić, ale generalnie idea polega na sprawdzaniu nagłówków i decydowaniu o losie pakietów.

## 3.1 Dlaczego mógłbym chcieć filtrować pakiety?

Kontrola. Bezpieczeństwo. Czujność.

### Kontrola:

kiedy używasz Linuksa by połączyć swoją wewnętrzną sieć z inną siecią (powiedzmy z Internetem) masz okazję wpuścić trochę różnych typów ruchu i odrzucić inne. Na przykład, nagłówek pakietu posiada adres docelowy pakietu, więc możesz odrzucać pakiety które podróżują do określonych części sieci zewnętrznej. Innym przykładem może być to: używam Netscape do oglądania archiwów Dilbert'a. Jest tam masa reklam pochodzących z adresu doubleclick.net, więc Netscape traci czas by je ładować. Pouczenie filtra pakietów by nie wpuszczał pakietów podróżujących do i z tego adresu rozwiązuje ten problem (jednakże jest parę innych sposobów by zrobić to lepiej, sprawdź Junkbuster).

### Bezpieczeństwo:

kiedy Twój linuks jest jedynym komputerem pomiędzy chaosem Internetu i twoją ładną, uporządkowaną siecią, miło jest wiedzieć że możesz obłożyć restrykcjami to co nadchodzi do twoich drzwi. Na przykład, możesz pozwolić by wszystko wychodziło z twojej sieci, ale możesz być zaniepokojony znanym atakiem 'Ping of Death' nadchodzącym od różnych złośliwych użytkowników sieci. Innym przykładem może być twoje życzenie, by nie zezwalać na telnet'owanie się na Twój komputer, mimo że wszystkie konta mają hasła; prawdopodobnie chcesz być (jak większość ludzi) raczej obserwatorem w Internecie a nie serwerem - po prostu nie dawać się nikomu do Ciebie dołączać, poprzez filtrowanie nadchodzących pakietów służących do ustanawiania połączeń.

### Czujność:

czasami źle skonfigurowana maszyna w sieci lokalnej zadecyduje o skierowaniu paru pakietów do sieci zewnętrznej. Miło jest móc poinstruować filtr pakietów by dał Ci znać o takich anormalnych zachowaniach; może będziesz chciał coś z tym zrobić, albo jesteś po prostu ciekawy z natury.

## 3.2 Jak filtrować pakiety pod Linuksem?

Kernele Linuksa miały wbudowane filtrowanie pakietów od serii 1.1. Pierwsza generacja, bazująca na ipfw z BSD, została przeniesiona przez Alana Cox'a pod koniec 1994. Została ona rozbudowana przez Jos'a Vos'a i innych dla Linuksa wersji 2.0; narzędzie działające w przestrzeni użytkownika 'ipfwadm' kontrolowało reguły filtrujące. W połowie 1998, dla Linuksa 2.2, zmieniłem dosyć mocno kernel, z pomocą Michael'a Neuling'a, i wprowadziłem narzędzie również działające w przestrzeni użytkownika nazwane 'ipchains'. Ostatecznie, w połowie 1999 dla Linuksa 2.4 stworzono kolejne narzędzie 'iptables'. Jest to właśnie to iptables na którym skoncentrowane jest to HOWTO.

Potrzebujesz kernel z infrastrukturą netfilter: netfilter to ogólny szkielet w kernelu Linuksa do którego mogą dołączać się inne moduły (takie jak moduł iptables). Oznacza to że potrzebujesz kernel w wersji 2.3.15 lub późniejszej, i w czasie konfiguracji kernela musisz zaznaczyć 'y' przy opcji 'CONFIG\_NETFILTER'.

Narzędzie iptables wstawia i kasuje reguły z tabeli filtrowania pakietów kernela. Oznacza to, że cokolwiek do niej wstawisz, zostanie stracone po restarcie; zajrzyj do [sekcji w której opisujemy zapisywanie reguł](#) po informacji jak upewnić się że po kolejnym starcie linuks odtworzy je.

iptables zastępuje ipfwadm i ipchains: zajrzyj do [sekcji opisującej używanie ipfwadm i ipchains](#) po informacje jak bezboleśnie uniknąć przesiadania się na iptables jeśli używasz jednego z tych narzędzi.

## Zapisywanie reguł na stałe

Twoje aktualne ustawienia ściany ogniowej zapisane są w kernelu i w związku z tym znikną po restarcie. Możesz wypróbować skrypty iptables-save i iptables-restore by odpowiednio zapisać je do i odtworzyć z pliku.

Innym sposobem może być umieszczenie komend wymaganych by ustawić twoje reguły w skrypcie wykonywanym w czasie startu. Upewnij się, że zrobi on coś inteligentnego w wypadku gdyby coś poszło nie tak (zwykle wywołanie 'exec /sbin/sulogin').

## 4. Kim u diabła jesteś i dlaczego bawisz się moim kernelem?

Jestem Rusty Russell; człowiek odpowiedzialny za ścianę ogniową IP Linuksa i jeszcze jeden koder który znalazł się we właściwym miejscu we właściwym czasie. Napisałem ipchains (zajrzyj do sekcji [Jak filtrować pakiety pod Linuksem?](#)) powyżej by sprawdzić kto tak naprawdę to wszystko stworzył i nauczyłem się wystarczająco dużo by tym razem filtrowanie pakietów wyszło dobrze. Tak sądzę.

Doskonała firma WatchGuard [WatchGuard](#) sprzedająca ściany ogniowe Firebox, zaoferowała że będzie płacić mi za nic, więc mogłem spędzać swój czas pisząc ten dokument i zajmować się poprzednimi rzeczami. Przewidywałem że zajmie to 6 miesięcy, zajęło 12, ale na końcu czuję że zostało to zrobione Właściwie. Wiele razy przepisywane od początku, padnięty twardy dysk, ukradziony laptop, parę uszkodzonych systemów plików i jeden zniszczony ekran, ale jest.

Póki jeszcze tu jestem, chciałbym wyklarować mylne wrażenia niektórych ludzi: nie jestem guru kernela. Wiem to wszystko, ponieważ moja praca spowodowała że kontaktowałem się z ludźmi którzy są guru: Davidem S. Millerem, Aleksiejem Kuzieczowem, Andi Kleenem, Alanem Coksem. Są zajęci uprawianiem głębokiej magii, a ja mogłem pobawić się na płytkich wodach, tam gdzie jest bezpiecznie.

## 5. Bardzo krótki przewodnik Rusty'ego do filtrowania pakietów

Większość ludzi ma pojedyncze połączenie PPP do Internetu, i nie chce by ktokolwiek mógł łączyć się do nich, lub do ściany ogniowej:

```
## Załaduj moduły śledzenia połączeń (niepotrzebne jeśli wbudowane w kernel)
# insmod ip_conntrack
# insmod ip_conntrack_ftp

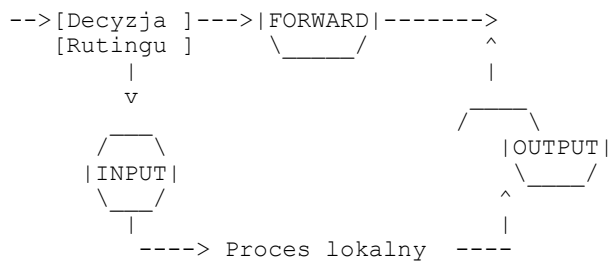
## Stwórz łańcuch blokujący nowe połączenia, z wyjątkiem tych od środka
# iptables -N block
# iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A block -j DROP

## Do łańcuchów INPUT i FORWARD dodaj skok do tego nowego łańcucha
# iptables -A INPUT -j block
# iptables -A FORWARD -j block
```

## 6. Jak pakiety podróżują przez filtry

Kernel rozpoczyna pracę z trzema listami reguł w tabeli filtrującej; nazywane są one **łańcuchami ściany ogniowej**, lub po prostu **łańcuchami**. Te trzy nazwane zostały **INPUT (wejściowy)**, **OUTPUT (wyjściowy)** i **FORWARD (przekazujący)**.

Dla fanów ASCII-art, łańcuchy ułożone są w następujący sposób: **(UWAGA: Wygląda to zupełnie inaczej niż w kernelach 2.0 i 2.2!)**



Trzy koła reprezentują trzy łańcuchy o których wspomniałem wyżej. Kiedy pakiet dociera do koła na diagramie, sprawdzany jest łańcuch reguł by zdecydować o losie pakietu. Jeśli łańcuch mówi że należy odrzucić (DROP) pakiet, jest on odrzucany tutaj, ale jeśli łańcuch mówi by zaakceptować pakiet (ACCEPT), kontynuuje on swoją podróż po diagramie.

Łańcuch to lista **reguł**. Każda reguła mówi 'jeśli nagłówek pakietu wygląda tak, to zrobimy z tym pakietem następującą rzecz'. Jeśli reguła nie pasuje do pakietu, sprawdzana jest następna. Na koniec, jeśli nie ma więcej reguł, kernel sprawdza politykę (ang. *policy*) danego łańcucha. W systemie w którym dba się o bezpieczeństwo, polityka mówi zwykle kernelowi by odrzucić (DROP) pakiet.

1. Kiedy pakiet dociera do maszyny (powiedzmy, przez kartę Ethernetową), kernel sprawdza najpierw adres przeznaczenia pakietu: nazywa się to routingiem.
2. Jeśli pakiet przeznaczony jest do tego komputera, pakiet zostaje przepuszczony do łańcucha INPUT (wejściowego). Jeśli przejdzie go, otrzymuje go proces do którego był adresowany.
3. W innym przypadku, jeśli kernel nie ma włączonego **przekazywania** (ang. *forwarding*), lub nie wie jak przekazać pakiet, jest on odrzucany. Jeśli przekazywanie jest włączone i pakiet jest przeznaczony do innego interfejsu sieciowego (jeśli w ogóle masz jeszcze jeden), pakiet przechodzi w prawo na naszym diagramie do łańcucha FORWARD (przekazującego). Jeśli zostaje zaakceptowany (ACCEPT), zostanie wysłany dalej.
4. Na koniec, program pracujący na tym komputerze może również wysyłać własne pakiety. Przejdą one od razu do łańcucha OUTPUT (wyjściowego): jeśli stwierdzi on że zaakceptuje pakiet (ACCEPT), pakiet przechodzi do właściwego interfejsu sieciowego.

## 7. Używanie iptables

iptables ma całkiem szczegółowy podręcznik (man iptables), do którego warto zajrzeć jeśli chodzi ci o coś konkretnego. Ci z was którzy znają ipchains mogą po prostu zajrzeć do [różnic pomiędzy iptables i ipchains](#); oba narzędzia są bardzo podobne.

Istnieje wiele rzeczy które możesz zrobić przy użyciu iptables. Zaczynasz z trzema wbudowanymi łańcuchami INPUT, OUTPUT i FORWARD, których nie możesz skasować. Spójrzmy na listę możliwych operacji na całych łańcuchach:

1. Stworzenie nowego łańcucha (-N).
2. Skasowanie pustego łańcucha (-X).
3. Zmiana polityki dla wbudowanego łańcucha (-P).
4. Wylistowanie reguł w łańcuchu (-L).
5. Wyczyszczenie łańcucha z reguł (-F).
6. Wyzerowanie liczników pakietów i bajtów we wszystkich regułach w łańcuchu (-Z).

Jest również parę sposobów na manipulowanie regułami w obrębie łańcucha:

1. Dodanie nowej reguły do łańcucha (-A).
2. Wstawienie nowej reguły na pewnej pozycji w łańcuchu (-I).
3. Zamiana reguły na pewnej pozycji w łańcuchu (-R).
4. Skasowanie reguły na pewnej pozycji w łańcuchu, lub pierwszej która pasuje (-D).

### 7.1 Co zobaczysz gdy wystartujesz komputer

iptables może być modulem, ('iptables\_filter.o'), który powinien być automatycznie ładowany gdy po raz pierwszy uruchomisz iptables. Może być również skompilowany w kernelu.

Zanim nie zostaną wykonane jakieś komendy (zwróć uwagę, że niektóre dystrybucje wykonują już operacje z iptables w skryptach startowych) nie ma żadnych reguł we wbudowanych łańcuchach ('INPUT', 'FORWARD' i 'OUTPUT') i wszystkie mają domyślną politykę ACCEPT. Możesz zmienić domyślną politykę łańcucha FORWARD przez podanie opcji 'forward=0' do modułu iptable\_filter.

## 7.2 Operacje na pojedynczej regule

Manipulowanie regułami to bułka z masłem filtrowania pakietów. Najczęściej będziesz zapewne dodawał (-A) i kasował (-D). Inne komendy (-I dla wstawiania i -R do zamieniania) są prostymi rozwinięciami tych koncepcji.

Każda reguła jest zestawem warunków które pakiet musi spełnić i zawiera informację co zrobić jeśli tak się stało (czyli **cel**, ang. *target*). Na przykład, możesz chcieć odrzucać wszystkie pakiety ICMP nadchodzące z adresu IP 127.0.0.1. W tym przypadku naszymi warunkami są zatem: protokołem musi być ICMP, a adresem źródłowym 127.0.0.1. Naszym celem będzie 'DROP'.

127.0.0.1 to adres **pętli zwrotnej** (ang. *loopback*), który posiadasz nawet wtedy gdy nie masz żadnego połączenia sieciowego. Możesz użyć programu 'ping' by wygenerować pakiety takie jak powyżej. Polecenie wysyła pakiety ICMP typ 8 - żądanie echa (ang. *echo request*) na które wszystkie współpracujące hosty odpowiedzą pakietem ICMP typu 0 - echo. To sprawia, że polecenie to jest wygodne dla testów.

```
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
# iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
```

Widać powyżej że pierwszy ping dociera (opcje '-c 1' powoduje wysłanie tylko jednego pakietu).

Dodajemy teraz (-A) do łańcucha 'INPUT' regułę, która mówi że pakiety z 127.0.0.1 ('-s 127.0.0.1') protokołu ICMP ('-p icmp') powinny zostać przeznaczone do wyrzucenia ('-j DROP').

Testujemy następnie działanie naszej reguły, wykonując drugi ping. Nastąpi pauza po której program się podda, po krótkim oczekiwaniu na odpowiedź która nigdy nie nadejdzie.

Możemy skasować naszą regułę na dwa sposoby. Po pierwsze, ponieważ wiemy że jest to jedyna reguła w łańcuchu wejściowym, możemy użyć numerowania reguł, tak jak poniżej:

```
# iptables -D INPUT 1
#
```

Co spowoduje skasowanie reguły numer 1 w łańcuchu INPUT.

Drugi sposób to dokładne przepisanie poleceń po opcji -A, ale zamiast opcji -A podajemy opcję -D. Przydaje się to w przypadku gdy masz skomplikowany zestaw reguł i nie chce ci się liczyć ich wszystkich by ustalić w końcu że chcesz pozbyć się reguły numer 37. W tym wypadku użyjemy:

```
# iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
#
```

Składnia polecenia -D musi dokładnie odpowiadać opcjom które podałeś przy poleceniu -A (lub -I czy -R). Jeśli istnieje wiele identycznych reguł w tym samym łańcuchu, tylko pierwsza pasująca zostanie skasowana.

## 7.3 Specyfikacja filtrowania

Widzieliśmy już jak używać opcji `-p` by wskazać protokół i `-s` by wskazać adres źródłowy, ale są jeszcze inne opcje których możemy użyć by scharakteryzować pakiet. Poniżej znajdziesz wyczerpujące kompendium.

### Wskazanie adresów IP: źródłowego i docelowego

Adres IP źródłowy (`-s`, `--source` lub `--src`) i docelowy (`-d`, `--destination` lub `--dst`) mogą być podane na cztery sposoby. Najczęściej robi się to przez podanie pełnej nazwy, takiej jak `localhost` czy `www.linuxhq.com`. Drugim sposobem jest podanie adresu IP, tak jak np. `127.0.0.1`.

Trzeci i czwarty sposób pozwalają na wskazanie grupy adresów IP, tak jak na przykład `199.95.207.0/24` lub `199.95.207.0/255.255.255.0`. Oba wskazują na zakres adresów IP od 199.95.207.0 do 199.95.207.255 włącznie; cyfry po `/` mówią która część adresu IP ma znaczenie. `/32` czy inaczej `/255.255.255.255` jest domyślne (i mówi że wszystkie liczby w adresie IP są ważne). By wskazać dowolny adres, można użyć `/0` tak jak poniżej:

```
[ UWAGA: -s 0/0 jest tu całkowicie zbędne. ]
# iptables -A INPUT -s 0/0 -j DROP
#
```

Ale takich konstrukcji używa się rzadko, ponieważ efekt jest dokładnie taki sam jak w przypadku nie podania opcji `-s` w ogóle.

### Inwersja

Wiele flag, włączając `-s` (lub `--source`) i `-d` (`--destination`) można poprzedzić znakiem `!` (wymawianym 'nie') by wskazać adresy **nie** pasujące do tych podanych. Na przykład `-s ! localhost` będzie pasować do wszystkich pakietów nie pochodzących z localhost.

### Protokół

Protokół podaje się po parametrze `-p` (lub `--protocol`). Protokół może być numerem (jeśli znasz wartości numeryczne protokołów IP) lub nazwą dla `TCP`, `UDP` i `ICMP`. Wielkość liter nie ma znaczenia, więc `tcp` działa tak samo jak `TCP`.

Nazwa protokołu może być poprzedzona przez znak `!` by wskazać na wszystkie oprócz wymienionego, tak jak na przykład `-p ! TCP` (warunek dotyczy wszystkich protokołów prócz TCP).

### Interfejs

Opcje `-i` (lub `--in-interface` czyli interfejs wejściowy) i `-o` (lub `--out-interface` czyli interfejs wyjściowy) używane są dla wskazania interfejsu. Interfejs to fizyczne urządzenie do którego pakiety przychodzą (`-i`) i z którego są wysyłane (`-o`). Możesz użyć programu `ifconfig` by wylistować interfejsy które są 'podniesione' (tzn. aktualnie pracujące).

Pakiety podróżujące w łańcuchu `INPUT` nie mają interfejsu wyjściowego, więc podanie opcji `-o` w regule w tym łańcuchu spowoduje że nie będzie ona pasować do żadnego pakietu. Podobnie, pakiety podróżujące przez łańcuch `OUTPUT` nie mają interfejsu wejściowego, więc podanie opcji `-i` w regule tego łańcucha spowoduje że nie będzie ona nigdy pasowała.

Tylko pakiety podróżujące przez łańcuch `FORWARD` mają zarówno interfejs wejściowy i wyjściowy.

Jest całkowicie poprawne wskazanie interfejsu który aktualnie nie istnieje; reguła nie będzie pasowała do niczego, dopóki interfejs nie zostanie podniesiony. Jest to bardzo przydatne dla połączeń wdzwanianych PPP (zwykle interfejs ma nazwę `ppp0`) i podobnych.

Można również wskazać interfejs kończąc jego nazwę przez `+` co spowoduje że reguła będzie pasowała do wszystkich interfejsów których nazwa zaczyna się od podanego ciągu znaków (bez znaczenia czy interfejs aktualnie

istnieje czy nie). Na przykład, by wskazać regułę która pasuje do wszystkich interfejsów PPP użyć należy polecenia '-i ppp+'.

Interfejs może być poprzedzony przez '!' ze spacjami wokół, co spowoduje że pasować będą pakiety które nie są powiązane ze wskazanym interfejsem (tzn. pakiet nim nie dotarł do systemu, lub nie zamierza nim go opuścić), np. -i ! ppp+.

## Fragmenty

Czasami pakiet jest zbyt duży by zmieścić się cały w jednostce transmisji. Kiedy się tak dzieje, jest on dzielony na **fragmenty** i wysyłany jako osobne pakiety. Komputer docelowy składa fragmenty by zrekonstruować cały pakiet.

Problem z fragmentami polega na tym, że pierwszy fragment posiada komplet pól nagłówka (IP+TCP, UDP i ICMP) który można sprawdzać, ale następne fragmenty mają tylko podzbiór nagłówków (IP bez dodatkowych pól specyficznych dla protokołów które przenosi pakiet). W związku z tym niemożliwe jest analizowanie pewnych informacji z nagłówków fragmentów (tak jak robi się to dla typowych pakietów TCP, UDP i ICMP).

Jeśli prowadzisz śledzenie połączeń lub NAT, to wszystkie fragmenty zostaną najpierw złożone a dopiero później przekazane do kodu filtrującego pakiety, więc nie powinieneś się martwić fragmentami.

W każdym innym przypadku ważne jest by zrozumieć jak fragmenty traktowane są przez reguły filtrujące. Każda reguła która ma sprawdzić informacje których nie posiadamy dla danego pakietu, **nie będzie** pasowała. Oznacza to, że tylko pierwszy fragment traktowany jest tak, jak można by się tego spodziewać. Drugi pakiet i kolejne już nie będą. W związku z tym reguła '-p TCP --sport www' (podająca port źródłowy 'www') nigdy nie będzie pasowała do fragmentu pakietu (innego niż pierwszy). Nie będzie również pasować reguła odwrotna '-p TCP --sport ! www'.

Możesz jednak dodać reguły specjalnie dla drugiego i następnych fragmentów, poprzez użycie opcji '-f' (lub '--fragment'). Poprawne jest również dodanie reguły która nie pasuje do drugiego i następnych fragmentów, przez poprzedzenie opcji '-f' opcją '!'.  
-f !

Zwykle uważa się za bezpieczne umożliwienie drugiemu i następnym fragmentom przejść, ponieważ filtrowanie zajmie się pierwszym fragmentem i w związku z tym zapobiegnie złożeniu pakietu na maszynie docelowej; z drugiej strony znane były pluskwy które powodowały zawieszanie się maszyn tylko poprzez wysyłanie do nich fragmentów. To twoja decyzja.

Mała uwaga do speców od sieci: pakiety zniekształcone (TCP, UDP i ICMP które są zbyt krótkie by kod ściany ogniowej mógł odczytać porty, czy w przypadku ICMP kod i typ) są również wyrzucane gdy prowadzone są takie analizy. Dokładnie tak samo jest z fragmentami TCP które rozpoczynają się od pozycji 8.

Jako przykład, poniższa reguła wyrzuci wszystkie fragmenty przeznaczone dla 192.168.1.1:

```
# iptables -A OUTPUT -f -d 192.168.1.1 -j DROP
#
```

## Rozszerzenia do iptables: Nowe cele i nowe testy

iptables są **rozszerzalne**, co oznacza że funkcjonalność zarówno kernela jak i narzędzia iptables może być rozszerzana by dodać nowe opcje.

Niektóre rozszerzenia są standardowe, inne są trochę bardziej egzotyczne. Oczywiście, mogą być one dodawane przez innych ludzi i dystrybuowane niezależnie dla użytkowników niszowych.

Fizycznie rozszerzenia znajdują się zwykle w podkatalogu modułów kernela, tak jak na przykład '/lib/modules/2.4.0-test10/kernel/net/ipv4/netfilter'. Ładowane są na żądanie gdy kernel został skompilowany z opcją CONFIG\_KMOD, więc nie powinno być potrzeby ładowania ich ręcznie.

Rozszerzenia do narzędzia iptables są współdzielonymi bibliotekami, które znajdują się zwykle w '/usr/local/lib/iptables/', choć dystrybucje mogą umieścić je w katalogach takich jak '/lib/iptables/' czy '/usr/lib/iptables'.

Rozszerzenia mogą należeć do jednego z dwóch typów: nowych celów, lub nowych testów (porozmawiamy o nowych celach za moment). Niektóre protokoły oferują automatycznie nowe testy: aktualnie są nimi TCP, UDP i ICMP tak jak pokażemy to poniżej.

Możesz dla nich podać nowe testy w linii poleceń po opcji '-p', który ładuje rozszerzenie. Dla samodzielnych nowych testów, używa się opcji '-m' by załadować rozszerzenie, po której dostępne są nowe opcje.

By uzyskać pomoc do rozszerzenia, użyj opcji ładującej go ('-p', '-j' lub '-m') po której podaj '-h' lub '--help', np.:

```
# iptables -p tcp --help
#
```

## Rozszerzenia TCP

Rozszerzenia TCP ładowane są automatycznie gdy podano opcję '-p tcp'. Dodają one następujące opcje (z których żadna nie pasuje do fragmentów).

### --tcp-flags

Po której następuje opcjonalny znak '!', a następnie dwa ciągi flag które pozwalają wskazać zestaw flag do zbadania. Drugi ciąg mówi, które mają być ustawione. Na przykład:

```
# iptables -A INPUT --protocol tcp --tcp-flags ALL SYN,ACK -j DROP
```

mówi że sprawdzone powinny zostać wszystkie flagi ('ALL' to synonim dla 'SYN,ACK,FIN,RST,URG,PSH'), ale tylko flagi SYN i ACK powinny być ustawione. Istnieje również argument 'NONE' który oznacza że żadna flaga nie może być ustawiona.

### --syn

opcjonalnie poprzedzona przez '!', jest skrótem dla '--tcp-flags SYN,RST,ACK SYN'.

### --source-port

po której następuje opcjonalny '!', a następnie pojedynczy port lub grupa portów TCP. Porty mogą być wskazywane przez nazwy takie jak w /etc/services lub przez numery. Grupy portów wskazuje się albo przez dwie nazwy portów podzielone przez ':', lub (by wskazać większe lub równe wskazaniemu) przez port z dodanym ':', lub (by wskazać mniejsze lub równe wskazaniemu), port poprzedzany przez ':'.

### --sport

to synonim '--source-port'.

### --destination-port

i

### --dport

mają takie same opcje jak powyżej, ale określają port przeznaczenia.

### --tcp-option

po którym następuje opcjonalny znak ! i numer, które wskazują na opcję TCP równą wskazanemu numerowi. Pakiet który nie posiada kompletnego nagłówka TCP jest automatycznie odrzucany jeśli wykonana zostanie próba sprawdzenia jego opcji TCP.

## Parę słów wyjaśnienia o flagach TCP

Czasami użyteczne jest, by zezwolić na połączenia TCP w jednym kierunku ale nie w drugim. Na przykład, możesz chcieć zezwolić na połączenia do zewnętrznego serwera WWW, ale nie połączenia od tego serwera.

Naiwnym rozwiązaniem byłoby blokowanie pakietów TCP nadchodzących z tego serwera. Niestety, połączenia TCP wymagają by pakiety mogły poruszać się w jedną i w drugą stronę.

Rozwiązaniem jest blokowanie tylko pakietów używanych do nawiązania połączenia. Nazywa się je pakietami **SYN** (dobrze, technicznie rzecz biorąc są to pakiety z ustawioną flagą SYN i zgaszonymi flagami RST i ACK, ale nazywamy je pakietami SYN by było krócej). Poprzez zabronienie ruchu tylko tym pakietom, możemy zapobiec takim połączeniom u samego ich źródła.

Używa się do tego opcji '--syn': która jest dozwolona tylko dla reguł które wskazują na protokół TCP. Na przykład, by wskazać połączenia TCP z 192.168.1.1:

```
-p TCP -s 192.168.1.1 --syn
```

Działanie flagi można odwrócić poprzedzając ją '!', co oznacza, że chodzi nam o pakiety różne od tych które inicjują połączenie.

## Rozszerzenia UDP

Są one ładowane automatycznie po podaniu '-p udp'. Udostępniają opcję '--source-port', '--sport', '--destination-port' i '--dport', dokładnie takie same jak dla TCP powyżej.

## Rozszerzenia ICMP

Ładowane automatycznie po podaniu '-p icmp'. Udostępniają one tylko jedną nową opcję:

### --icmp-type

po której następuje opcjonalny znak '!', a następnie albo nazwa typu pakietu ICMP (np. 'host-unreachable', czyli *komputer nieosiągalny*), lub numer typu (np. '3'), lub numer typu i kod oddzielone przez '/' (np. '3/3'). Lista dostępnych nazw typów ICMP dostępna jest po podaniu '-p icmp --help'.

## Inne rozszerzenia testowe

Inne rozszerzenia w paczce netfiltera są rozszerzeniami demonstracyjnymi, które (jeśli je zainstalowano) mogą być wywołane poprzez opcję '-m'.

### mac

Moduł ten musi być wskazany przez '-m mac' lub '--match mac'. Używa się go do sprawdzania źródła Ethernetowego (tzw. adresu MAC) adresu nadchodzącego pakietu i w związku z tym działa tylko w łańcuchach PREROUTING i INPUT. Udostępnia on tylko jedną opcję:

### --mac-source

po którym następuje opcjonalny '!' a następnie adres ethernetowy w formacie heksdecymalnym oddzielanym dwukropkami, np. '--mac-source 00:60:08:91:CC:B7'.

### limit

Moduł ten musi być wskazany przez '-m limit' lub '--match limit'. Używa się go do ograniczania częstotliwości pasowania reguły, tak jak na przykład do ograniczanie wiadomości generowanych do logów. Spowoduje że pakiety będą pasować z taką częstotliwością jak podana w tej opcji w czasie jeden sekundy

(domyślnie 3 krotnie na godzinę, z serią 5). Moduł umożliwia podanie dwóch argumentów opcjonalnych:

### --limit

po którym następuje numer; określa maksymalny średni numer testów pozytywnych na sekundę. Numer może również wskazywać wprost jednostki, przez użycie '/second', '/minute', '/hour' lub '/day', albo ich skrótów (np. '5/second' to to samo co '5/s').

### --limit-burst

po którym następuje numer, określający maksymalną serię po której powyższy limit się włącza.

Używa się tego testu zwykle w połączeniu z celem LOG by zrealizować ograniczone logowanie. By zrozumieć jak to działa, popatrzmy jak działa następująca reguła, która loguje pakiety z domyślnymi limitami:

```
# iptables -A FORWARD -m limit -j LOG
```

Gdy pierwszy raz dochodzimy do reguły, pakiet jest logowany; tak naprawdę, ponieważ domyślną serią jest 5, pierwsze pięć pakietów zostanie zalogowane. Następnie, minie dwadzieścia minut zanim zalogowany zostanie następny pakiet pasujący do tej reguły, niezależnie od tego ile pakietów do niej dotrze. Jednocześnie, każde dwadzieścia minut które minie bez pakietu który pasowałby do tej reguły, spowoduje odnowienie jednego numeru z serii; jeśli żaden pakiet nie dotrze do reguły w ciągu 100 minut, seria zostanie w pełni odnowiona; do 5 sztuk, tak jak zaczęliśmy.

Nota: nie możesz aktualnie stworzyć reguły które ma czas odnawiania się powyżej 59 godzin, więc jeśli ustawisz średnią częstotliwość na jeden pakiet na dzień, numer serii musi być mniejszy niż 3.

Możesz również użyć tego modułu by zapobiec rozmaitym atakom Odmowy Usługi (ang. *Denial of Service*), z większym współczynnikiem częstotliwości by zwiększyć szybkość reakcji.

Zabezpieczenie przed powodzią pakietów SYN (ang. *Syn-flood*):

```
# iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
```

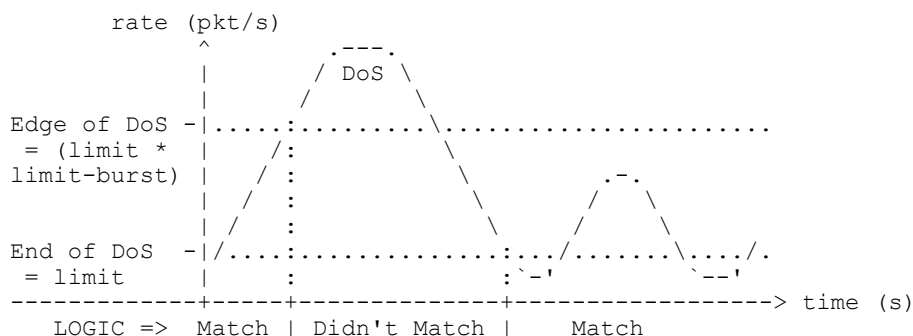
Skaner portów Furtive:

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit
```

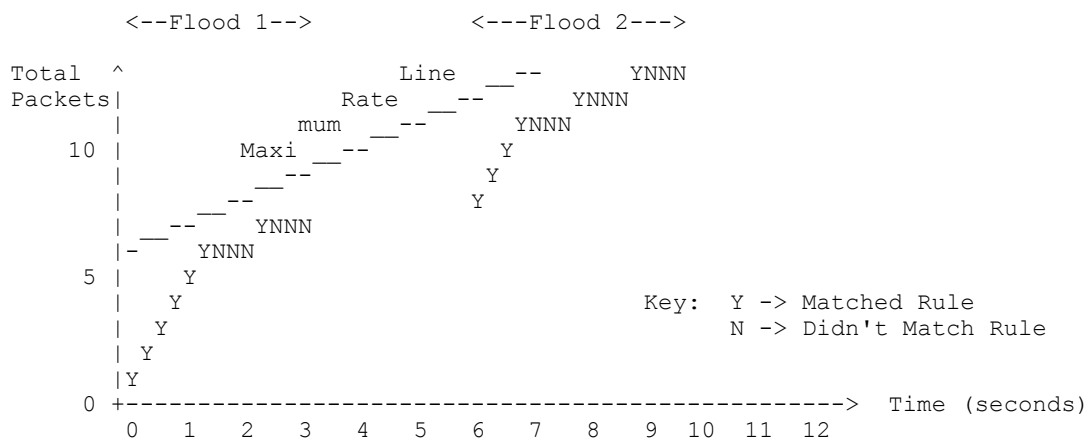
Ping of death:

```
# iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j
```

Moduł ten działa jak "drzwi hysterii", tak jak pokazano to na diagramie poniżej.



Powiedzmy że pasuje jeden pakiet na sekundę z serią pięciu pakietów, ale pakiety zaczynają dochodzić w ilości czterech na sekundę przez trzy sekundy, a następnie znowu po trzech sekundach.



Widać, że pierwsze pięć pakietów przekracza limit jednego pakietu na sekundę, a następnie włącza się ograniczanie (limit). Jeśli nastąpi pauza, kolejna seria zostanie wpuszczona ale nie powyżej maksymalnej częstotliwości określonej przez regułę (1 pakiet na sekundę po tym jak dotarła seria).

## owner (ang. właściciel)

Ten moduł stara się ustalić pewne charakterystyki twórcy pakietu, dla pakietów generowanych lokalnie. Jego użycie jest możliwe tylko w łańcuchu OUTPUT, a nawet nie dla wszystkich pakietów (takich jak odpowiedzi na ICMP ping) które mogą nie mieć właściciela, a w związku z tym nie będą pasowały.

### --uid-owner userid

Pasuje dla pakietów stworzonych przez proces ze wskazanym efektywnym (numerycznym) identyfikatorem użytkownika.

### --gid-owner groupid

Pasuje dla pakietów stworzonych przez proces ze wskazanym efektywnym (numerycznym) identyfikatorem grupy.

### --pid-owner processid

Pasuje dla pakietów stworzonych przez proces ze wskazanym efektywnym (numerycznym) identyfikatorem procesu.

### --sid-owner sessionid

Pasuje dla pakietów stworzonych przez proces we wskazanej grupie sesji.

## unclean (ang. brudny)

Jest to eksperymentalny moduł którego używa się przez podanie '-m unclean' lub '--match unclean'. Wykonuje różne losowe testy sprawdzające na pakiecie. Moduł ten nie był sprawdzany i nie powinien być używany jako test związany z bezpieczeństwem (prawdopodobnie sprawia że wszystko wygląda jeszcze gorzej, ponieważ sam może mieć pluskwy). Nie udostępnia żadnych opcji.

## Test Stanu

Najbardziej użytecznym testem jest ten dostarczany przez rozszerzenie 'state', który interpretuje analizę śledzenia połączeń modułu 'ip\_conntrack'. Generalnie bardzo zaleca się jego wykorzystanie.

Podanie w regule opcji '-m state' udostępnia dodatkową opcję '--state', która jest listą oddzielonych stanów do

przetestowania (opcja '!' wskazuje na pakiety **nie pasujące** do wskazanych stanów). Stanami które można sprawdzać są:

### NEW (NOWY)

Pakiet który tworzy nowe połączenie.

### ESTABLISHED (NAWIĄZANY)

Pakiet który należy do istniejącego połączenia (np. pakiet odpowiedzi, lub pakiet wychodzący w połączeniu które otrzymało już odpowiedzi).

### RELATED (POWIĄZANY)

Pakiet który jest powiązany z istniejącym połączeniem, ale nie jest jego częścią, tj. np pakiet z błędem ICMP, lub (jeśli załadowany jest moduł FTP) pakiet ustanawiający połączenie ftp dla danych.

### INVALID (BŁĘDNY)

Pakiet który nie może być zidentyfikowany z jakiś powodów: mogą to być wyczerpanie się pamięci, lub błędy ICMP które nie należą do żadnego znanego połączenia. Generalnie, pakiety tego typu powinno się odrzucać.

Przykładem wykorzystania tego potężnego rozszerzenia mogłoby być:

```
# iptables -A FORWARD -i ppp0 -m state ! --state NEW -j DROP
```

## 7.4 Cel

Znamy już testy które możemy przeprowadzić na pakiecie, potrzebujemy zatem sposobu by wskazać co robić z pakietami które pasują do naszych testów. Nazywa się to **celem** (ang. *target*) reguły.

Są dwa proste wbudowane cele : DROP (wyrzucić) i ACCEPT (zaakceptować). Już je widzieliśmy. Jeśli reguła pasuje do pakietu a jej cel jest jednym z tych dwóch, nie analizuje się już innych reguł: los pakietu został już określony.

Istnieją jeszcze dwa inne typy celów: rozszerzenia i łańcuchy zdefiniowane przez użytkownika.

### Łańcuchy zdefiniowane przez użytkownika

Bardzo potężną własnością którą iptables dziedziczy z ipchains jest możliwość tworzenia przez użytkownika nowych łańcuchów, oprócz wbudowanych (INPUT, FORWARD i OUTPUT). Zgodnie z przyjętą konwencją, wszystkie łańcuchy generowane przez użytkownika pisane są małymi literami by odróżnić je od wbudowanych (opiszemy jak tworzyć nowe łańcuchy użytkownika poniżej, w sekcji [Operacje na całym łańcuchu](#)).

Kiedy do reguły dociera pakiet który pasuje, a cel tej reguły zdefiniowany jest jako łańcuch zdefiniowany przez użytkownika, rozpoczyna on testy w tym właśnie łańcuchu. Jeśli w obrębie tego łańcucha los pakietu nie zostanie zdecydowany, przemierzanie reguł rozpoczyna się w pierwotnym łańcuchu, w miejscu w którym zostało przerwane (dokładnie od następnej reguły).

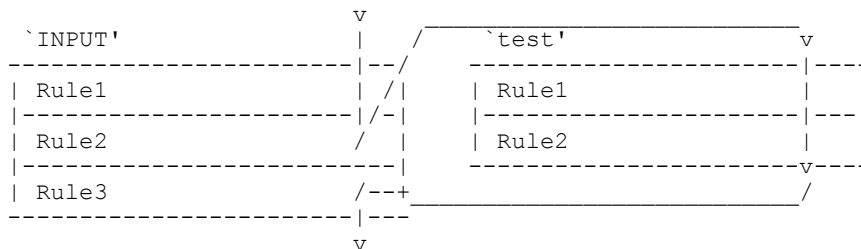
Czas na trochę rysunków ASCII. Rozważmy dwa (śmiesznie proste) łańcuchy: INPUT (łańcuch wbudowany) i test (łańcuch zdefiniowany przez użytkownika).

`INPUT'	`test'
Rule1: -p ICMP -j DROP	Rule1: -s 192.168.1.1
Rule2: -p TCP -j test	Rule2: -d 192.168.1.1

```
| Rule3: -p UDP -j DROP |
-----
```

Rozważmy pakiet TCP nadchodzący z 192.168.1.1 i wysłany do 1.2.3.4. Wchodzi on do łańcucha `INPUT` i rozpoczyna się sprawdzanie. Reguła 1 (Rule1) nie pasuje, natomiast druga tak. Ponieważ cel zdefiniowany jest jako `'test'`, następna reguła która jest sprawdzana pochodzi z łańcucha `'test'`. Pierwsza reguła w tym łańcuchu pasuje ale nie podaje celu, więc sprawdzana jest następna reguła. Ona nie pasuje i osiągany jest koniec łańcucha `'test'`. Wracamy do łańcucha `INPUT`, w którym ostatnio sprawdzaliśmy regułę drugą, teraz więc sprawdzamy trzecią która również nie pasuje.

Zatem droga pakietu wygląda w sposób następujący:



Łańcuchy zdefiniowane przez użytkownika mogą jako cel wskazywać inne łańcuchy również zdefiniowane przez użytkownika (ale nie mogą tworzyć pętli: twój pakiet zostanie wyrzucony jeśli okaże się że jest sprawdzany w pętli).

## Rozszerzenia do iptables: Nowe Cele

Innym typem rozszerzenia jest cel. Cel składa się z modułu kernela i opcjonalnych rozszerzeń `iptables`, które zapewniają opcje dla linii poleceń. Jest kilka takich rozszerzeń w standardowej dystrybucji `netfilter`:

### LOG

Moduł ten zapewnia logowanie w kernelu pasujących pakietów. Udostępnia następujące opcje:

#### **--log-level**

Po którym następuje nazwa poziomu logowania lub odpowiednik numeryczny. Prawidłowymi nazwami są (wielkość liter nie jest ważna) `'debug'`, `'info'`, `'notice'`, `'warning'`, `'err'`, `'crit'`, `'alert'` i `'emerg'`, którym odpowiadają cyfry od 7 do 0. Sprawdź stronę podręcznika `syslog.conf` by dowiedzieć się co oznaczają poszczególne poziomy. Domyślnym jest `'warning'`.

#### **--log-prefix**

po którym następuje ciąg do 29 znaków, który dodawany jest do logowanej informacji by umożliwić jej jednoznaczną identyfikację.

Moduł ten używany jest najczęściej z testem `limit`, dzięki czemu nie zaśmiecasz sobie logów.

### REJECT (ODRZUĆ)

Moduł ten ma takie samo działanie jak `'DROP'`, poza tym że to nadawcy pakietu odsyłany jest pakiet ICMP `'port unreachable'`. Weź jednak pod uwagę fakt, że błąd ICMP nie zostanie odesłany jeśli (sprawdź RFC 1122):

- Pakiet którego dotyczy ta reguła jest wiadomością ICMP o błędzie, lub nieznanym typem wiadomości ICMP
- Pakiet którego dotyczy ta reguła był drugim lub dalszym fragmentem.
- Wysłaliśmy już ostatnio zbyt dużo wiadomości o błędach ICMP do tego nadawcy (zajrzyj do `/proc/sys/net/ipv4/icmp_ratelimit`).

Do `REJECT` można również dodać opcjonalny argument `'--reject-with'` który pozwala na zadeklarowanie jaki dokładnie pakiet ICMP ma zostać odesłany zamiast domyślnego `'port unreachable'`. Sprawdź stronę

podręcznika.

## Specjalne cele wbudowane

Są dwa wbudowane specjalne cele: `RETURN` (POWRÓT) i `QUEUE` (KOLEJKA).

`RETURN` ma dokładnie ten sam efekt jak zakończenie sprawdzania łańcucha: dla reguły w łańcuchu wbudowanym sprawdzana jest wtedy polityka. Dla reguły w zdefiniowanym przez użytkownika łańcuchu, oznacza to powrót do poprzedniego łańcucha, zaraz po regule która spowodowała skok do tego łańcucha.

`QUEUE` to cel który kolejkuje pakiety dla przetwarzania w przestrzeni użytkownika. Żeby można było ten cel zastosować, potrzebne są jeszcze dwa składniki:

- program **obsługujący kolejkę** (ang. *queue handler*), który zajmie się mechaniką przekazywania pakietów pomiędzy kerneliem i **przestrzenią użytkownika** (ang. *userspace*); oraz
- aplikacja działająca w przestrzeni użytkownika która będzie potrafiła obsłużyć przyjęcie, ewentualną modyfikację i opcjonalne wydanie werdyktu w sprawie pakietu.

Standardowym programem obsługującym kolejkę dla IPv4 jest w iptables moduł `ip_queue`, który dystrybuowany jest z kerneliem i oznaczony jako eksperymentalny.

Poniżej przedstawiono krótki przykład jak użyć iptables z kolejką pakietów do przetwarzania w przestrzeni użytkownika:

```
# modprobe iptable_filter
# modprobe ip_queue
# iptables -A OUTPUT -p icmp -j QUEUE
```

W powyższych regułach, pakiety ICMP generowane lokalnie (tak jak na przykład przy użyciu polecenia ping) przekazywane są do modułu `ip_queue`, który stara się dostarczyć pakiety do aplikacji działającej w przestrzeni użytkownika. Jeśli nie ma takiej aplikacji, pakiety są wyrzucane.

By napisać taką aplikację, należy użyć API `libipq`. Dystrybuowana jest ona razem z iptables. Kod przykładowy znajduje się w narzędziach testuite (`np.redirect.c`) z CVS.

Status modułu `ip_queue` może być sprawdzony przez wywołanie:

```
/proc/net/ip_queue
```

Maksymalna długość kolejki (tzn. ilość pakietów dostarczonych do przestrzeni użytkownika bez odpowiedzi) może być kontrolowana przez:

```
/proc/sys/net/ipv4/ip_queue_maxlen
```

Domyślną wartością jest 1024. Kiedy zostaje osiągnięty limit, nowe pakiety będą wyrzucane dopóki długość kolejki nie spadnie poniżej wartości maksymalnej. Dobre protokoły takie jak TCP interpretują wyrzucane pakiety jako tłok i prawdopodobnie dadzą sobie spokój gdy kolejka się wypełni. Można oczywiście trochę poeksperymentować by wyznaczyć idealną maksymalną długość kolejki dla określonej sytuacji, jeśli domyślna wartość jest zbyt mała.

## 7.5 Operacje na całym łańcuchu

Bardzo przydatną opcją w iptables jest możliwość grupowania reguł w łańcuchy. Możesz je nazwać jak chcesz, ale zalecam raczej używanie małych liter by nie pomylić ich z wbudowanymi łańcuchami i celami. Nazwy ograniczone są do 31 liter.

### Tworzenie nowego łańcucha

Stwórzmy nowy łańcuch. Ponieważ jestem kolesiem z wyobraźnią, nazwijmy go `'test'`. Możemy użyć albo `'-N'` albo `'--new-chain'`:

```
# iptables -N test
#
```

Proste. Możesz teraz dodać do niego swoje reguły tak jak to już opisano.

## Kasowanie łańcucha

Kasowanie łańcucha również jest proste, używa się do tego opcji '-X' lub '--delete-chain'. A dlaczego '-X'? Cóż, wszystkie dobre literki były już zajęte.

```
# iptables -X test
#
```

Jest jednak parę ograniczeń dotyczących kasowania łańcuchów: muszą być puste (sprawdź sekcję [Opróżnianie łańcucha](#) poniżej) i nie mogą być wskazywane jako cel w innej regule. Nie możesz również skasować żadnego z trzech wbudowanych łańcuchów.

Jeśli nie podasz nazwy łańcucha, skasowane zostaną w miarę możliwości wszystkie łańcuchy zdefiniowane przez użytkownika.

## Opróżnianie łańcucha

Istnieje oczywiście również sposób by wykasować wszystkie reguły z łańcucha. Używa się do tego opcji '-F' (lub '--flush').

```
# iptables -F FORWARD
#
```

Jeśli nie wskażesz konkretnego łańcucha, opróżnione zostaną wszystkie.

## Listowanie zawartości łańcucha

Możesz wylistować reguły w łańcuchu, używając opcji '-L' (lub '--list').

Pozycja 'refcnt' przy każdym łańcuchu zdefiniowanym przez użytkownika podaje numer reguł które odwołują się do tego łańcucha. Wartość ta musi być równa zero (a łańcuch musi być pusty), by taki łańcuch można było skasować.

Jeśli pominięto nazwę łańcucha, wylistowane zostaną wszystkie łańcuchy, nawet te puste.

Istnieją trzy opcje które mogą towarzyszyć opcji '-L'. Opcja '-n' (numerycznie) jest o tyle przydatna, że zapobiega sprawdzaniu nazw odpowiadającym adresom IP przez `iptables`, co może spowodować duże zwłoki jeśli twój DNS (a zakładamy że używasz DNS jak większość ludzi) nie jest prawidłowo skonfigurowany, lub odfiltrowałś zapytania DNS. Powoduje ona również podanie portów TCP i UDP numerycznie zamiast nazw.

Opcja '-v' pokazuje wszystkie detale reguł, takie jak liczniki pakietów i bajtów, porównania TOS (Typu Usługi, ang. *Type of Service*) i interfejsy. Bez tej opcji wszystkie te informacje zostaną pominięte.

Zwróć uwagę że liczniki pakietów i bajtów drukowane są przy użyciu suffiksów 'K', 'M' lub 'G', dla odpowiednio 1000, 1,000,000 i 1,000,000,000. Poprzez użycie opcji '-x' (rozwiń liczby) można uzyskać pełne liczby, bez względu na to jak są duże.

## Resetowanie (zerowanie) liczników

Czasami przydatne jest móc wyzerować liczniki. Wykonuje się to przez użycie opcji '-z' (lub '--zero').

Rozważ poniższy przykład:

```
# iptables -L FORWARD
# iptables -Z FORWARD
#
```

Pewna liczba pakietów mogłaby przejść pomiędzy wydaniem polecenia z opcją '-L' a '-Z'. W związku z tym, możesz tych opcji używać **razem**, by wyzerować liczniki dokładnie w momencie ich wyświetlenia.

## Określanie polityki

Wspomnieliśmy już co dzieje się gdy pakiet dociera do końca wbudowanego łańcucha, kiedy rozmawialiśmy o tym jak pakiet podróżuje przez łańcuchy. W tym przypadku o losie pakietu decyduje **polityka** dla łańcucha. Tylko wbudowane łańcuchy (INPUT, OUTPUT i FORWARD) mają przypisaną politykę, ponieważ jeśli pakiet dociera do końca łańcucha zdefiniowanego przez użytkownika, sprawdzanie wraca do poprzedniego łańcucha.

Polityką może być ACCEPT lub DROP, na przykład:

```
# iptables -P FORWARD DROP
#
```

## 8. Używanie ipchains i ipfwadm

W dystrybucji netfilter znajdują się moduły ipchains.o i ipfwadm.o. Można załadować je do kernela (UWAGA: są niekompatybilne z modułem ip\_tables.o!). Możesz następnie używać ich tak jak za starych dobrych czasów.

Będą one wspierane jeszcze przez jakiś czas. Wydaje mi się że sensownym wzorem jest 2\*[notka o zastąpieniu-pierwsza stabilna dystrybucja], po którym to czasie dostępna jest stabilna wersja nowego odpowiednika. Oznacza to że wsparcie dla tych modułów skończy się wraz z pojawieniem się linuxa 2.6 lub 2.8.

## 9. Łączenie filtrowania pakietów i NAT

Zwykle używa się jednocześnie Translacji Adresów Sieciowych (NAT, zajrzyj do NAT HOWTO) i filtrowania pakietów. Dobra wiadomość jest taka że łączą się je bardzo dobrze.

Projektujesz swój filtr pakietów kompletnie ignorując NAT. Adresy źródłowe i przeznaczenia które będzie sprawdzał filtr pakietów będą 'prawdziwymi' adresami. Na przykład, jeśli prowadzisz DNAT i wysyłasz połączenia do 1.2.3.4 na port 80 przez 10.1.1.1 port 8080, filtr pakietów zobaczy pakiety podróżujące do 10.1.1.1 na port 8080 (prawdziwy adres przeznaczenia), a nie 1.2.3.4 na port 80. Podobnie, możesz zignorować masquerading (maskaradę): z punktu widzenia filtra pakietów pakiety będą nadchodziły z prawdziwych wewnętrznych adresów IP (powiedzmy 10.1.1.1) a odpowiedzi wracały gdzie powinny.

Możesz używać testu stanu ('state') bez dostarczania dodatkowej pracy filtrowi pakietów, ponieważ NAT wymaga i tak śledzenia połączeń (ang. *connection tracking*). By rozszerzyć prosty przykład maskarady z NAT HOWTO o zabronienie kreowania nowych połączeń z interfejsu ppp0, zrobiłbyś tak:

```
# Masquerade out ppp0
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

# Disallow NEW and INVALID incoming or forwarded packets from ppp0.
iptables -A INPUT -i ppp0 -m state --state NEW,INVALID -j DROP
iptables -A FORWARD -i ppp0 -m state --state NEW,INVALID -j DROP

# Turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

## 10. Różnice pomiędzy iptables i ipchains

- Po pierwsze, nazwy wbudowanych łańcuchów zostały zmienione z małych na DUŻE litery, ponieważ łańcuchy INPUT i OUTPUT otrzymują tylko pakiety kierowane do maszyny lokalnej lub pakiety generowane lokalnie. Do tej pory używano ich dla wszystkich pakietów przychodzących i wychodzących.

- Opcja '-i' oznacza teraz interfejs wejściowy i działa tylko w łańcuchach INPUT i FORWARD. Reguły w łańcuchach FORWARD i OUTPUT zamiast dotychczasowego '-i' używają '-o'.
- Porty TCP i UDP podaje się teraz przez opcje --source-port lub --sport (czy też --destination-port/--dport) i zapisuje po wskazaniu protokołu ('-p tcp' lub '-p udp') ponieważ dopiero one ładują rozszerzenia odpowiednio TCP i UDP
- Opcję TCP '-y' zmieniono na '--syn', i należy ją zapisać po '-p tcp'.
- Cel DENY w końcu nazywa się DROP (WYRZUCIĆ).
- Działa zerowanie pojedynczych łańcuchów z jednoczesnym wylistowaniem ich zawartości.
- Działa zerowanie wbudowanych łańcuchów łącznie z kasowaniem liczników wywołania polityki dla danego łańcucha
- Listowanie łańcuchów dostarcza ci 'atomowego' zdjęcia (ang. *snapshot*) liczników.
- REJECT i LOG są teraz celami rozszerzonymi, co oznacza że są osobnymi modułami kernela.
- Nazwy łańcuchów mogą mieć do 31 znaków.
- Opcja MASQ nazywa się w końcu MASQUERADE i używa innej składni. REDIRECT zachował nazwę, ale również zmieniono składnię. Sprawdź dokument NAT-HOWTO by uzyskać więcej informacji jak użyć tych opcji.
- Opcja '-o' nie jest już używana by kierować pakiety do urządzenia z przestrzeni użytkownika (sprawdź opcję '-i' powyżej). Wysła się je tam poprzez skierowanie do celu QUEUE.
- Prawdopodobnie cała masa rzeczy o których zapomniałem.

## 11. Porady w projektowaniu filtra pakietów

W dziedzinie bezpieczeństwa komputerowego zwykle za powszechną mądrość uważa się blokowanie wszystkiego a dopiero potem otwieraniu odpowiednich portów w miarę jak stają się potrzebne. Mówi się o tym zwykle 'to co nie jest wyraźnie dozwolone, jest zabronione'. Zalecam to podejście, jeśli bezpieczeństwo jest twoim największym priorytetem.

Nie uruchamiaj żadnych usług których nie musisz mieć, nawet jeśli wydaje ci się że zablokowałeś do nich dostęp.

Jeśli budujesz dedykowaną ścianę ogniową, rozpocznij od zera z blokowaniem wszystkich pakietów, potem dodawaj usługi i reguły które pozwolą im działać.

Zalecam również bezpieczeństwo 'warstwowe': połącz tcp-wrappers (dla połączeń do filtra pakietów), proxy (dla połączeń przechodzących przez filtr pakietów), weryfikację drogi (ang. *route verification*) i filtrowanie pakietów. Weryfikacja drogi ma miejsce wtedy, gdy pakiet dociera z niewłaściwego interfejsu i jest odrzucany: na przykład, twoja sieć wewnętrzna ma adresy 10.1.1.0/24, a pakiet z takim adresem dociera do filtra pakietów przez interfejs zewnętrzny - powinien zostać odrzucony. Może to zostać włączone dla jednego interfejsu (ppp0) tak jak niżej:

```
# echo 1 > /proc/sys/net/ipv4/conf/ppp0/rp_filter
#
```

Lub dla wszystkich istniejących i przyszłych interfejsów tak jak niżej:

```
# for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
#     echo 1 > $f
# done
#
```

Debian robi to domyślnie jeśli jest to możliwe. Jeśli masz routing asymetryczny (tzn. spodziewasz się pakietów nadchodzących z dziwnych kierunków), będziesz prawdopodobnie musiał wyłączyć takie filtrowanie na tych interfejsach.

Logowanie jest użyteczne w trakcie konfigurowania ściany ogniowej, gdy coś nie działa, ale na działającej ścianie ogniowej zawsze połącz logowanie z opcją 'limit', by zapobiec zapełnieniu twoich logów.

Zalecam również śledzenie połączeń dla systemów w których bezpieczeństwo jest sprawą priorytetową: wprowadza pewne opóźnienia, ponieważ śledzone są wszystkie połączenia, ale jest to bardzo przydatne w kontrolowaniu dostępu do twoich sieci. Być może będziesz musiał załadować moduł 'ip\_conntrack.o' jeśli twój kernel nie ładuje modułów automatycznie albo jeśli nie jest on już wbudowany w kernel. Jeśli chcesz śledzić dokładnie skomplikowane protokoły, musisz załadować odpowiedni moduł wspomagający (np. 'ip\_conntrack\_ftp.o').

```
# iptables -N no-conns-from-ppp0
```

```
# iptables -A no-conns-from-ppp0 -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A no-conns-from-ppp0 -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A no-conns-from-ppp0 -i ppp0 -m limit -j LOG --log-prefix "Bad packet f:
# iptables -A no-conns-from-ppp0 -i ! ppp0 -m limit -j LOG --log-prefix "Bad packet
# iptables -A no-conns-from-ppp0 -j DROP

# iptables -A INPUT -j no-conns-from-ppp0
# iptables -A FORWARD -j no-conns-from-ppp0
```

Budowanie dobrej ściany ogniowej jest poza tematem tego HOWTO, ale moją radą jest 'bądź minimalistą'. Zajrzyj do Security HOWTO po więcej informacji o testowaniu i sprawdzaniu twojej maszyny.

## 12. Uwagi od tłumacza

Chciałbym podziękować następującym osobom za uwagi co do tłumaczenia i zasugerowanie poprawek:

**Adam Ryba** [aryba \(at\) gate.pl](mailto:aryba@gate.pl)

- literówki i apostrofy ;)