Systemy programowych zapór sieciowych (iptables)

1. Wprowadzenie

Programowe zapory sieciowe (*ang. firewall*) wykorzystywane mogą być do przeróżnych celów w zależności od złożoności takiej zapory programowej. Wyróżniamy zapory osobiste (*ang. personal firewall*) oraz zapory ogólnego przeznaczenia.

Zapora sieciowa ma za zadanie filtrować ruch przychodzący i wychodzący z danego urządzenia sieciowego (na przykład komputera). Dzięki filtrowaniu ruchu możemy zapewnić bezpieczeństwo sieciowe urządzenia, jednak należy pamiętać, iż nigdy nie zapewnimy 100% bezpieczeństwa sieciowego, chyba że odłączymy komputer od sieci.

Celem ćwiczenia jest zapoznania się z programową zaporą sieciową systemu Linux IPTABLES oraz wykorzystanie jej możliwości do rozwiązania postawionych problemów.

2. Informacje o zaporach sieciowych

Zapora sieciowa zawiera filtr pakietów, którego zadaniem jest sprawdzanie nagłówków pakietów przepływających przez stos protokołów. Decyduje o ich losie, **akceptując** (ang. *accept*) lub **odrzucając** (ang. *drop*) poszczególne pakiety.

Zapory sieciowe dzielimy na osobiste oraz ogólnego przeznaczenia, różnice są znaczne, pierwsze z wymienionych nadają się wyłącznie do ochrony komputera używanego jako jednostki końcowej do pracy (lub zabawy), najczęściej zapory tego typu pytają się użytkownika czy ma przepuścić czy nie dany ruch sieciowy. Natomiast zapory ogólnego przeznaczenia charakteryzują się bardziej różnorodnym zastosowaniem, jednakże wymagają większej wiedzy, aby poprawnie je skonfigurować.

Iptables jest zaporą ogólnego przeznaczenia, dlatego skupimy się nad jego możliwościami od najbardziej podstawowych po bardziej złożone.

Ogólne założenie zapory sieciowej z systemie operacyjnym Linux

W systemie Linux filtrowanie pakietów IP jest wbudowane w jądro systemu operacyjnego. Od wersji 2.3.15 jądra odpowiedzialny jest za to moduł o nazwie netfilter, oferujący niezależną filtrację pakietów przychodzących, wychodzących oraz routowanych. Moduł ten pracuje jako swoisty rdzeń umożliwiający dołączanie różnych modułów konfiguracji i administracji filtracji pakietów, np. modułu iptables. Z kolei narzędzie iptables służy do dynamicznego konstruowania reguł filtracji. Reguły przechowywane są w pamięci jądra, a ich stałą dostępność można zapewnić przechowując wybrane konfiguracje w plikach (wykorzystując skrypty iptables-save i iptables-restore).

Konfiguracja iptables

Konfigurowanie filtracji polega na definiowaniu tzw. reguł *(ang. rule*). Pojedyncza reguła składa się z wzorca i akcji. Gdy wiadomość poddawana jest analizie, najpierw dopasowywana jest do wzorca; jeśli do niego pasuje, o dalszym losie wiadomości decyduje akcja reguły. Jeśli zaś dane

w wiadomości nie pasują do wzorca danej reguły, pod uwagę brana jest kolejna reguła. Reguły przeglądane są w takim porządku, w jakim zostały wprowadzone. Dla danej wiadomości przeszukiwanie reguł trwa do momentu pierwszego dopasowania do wzorca lub (w przypadku braku dopasowania) do wyczerpania reguł.

Akcja reguły zależy od jej typu. Wyróżnia się trzy typy reguł:

- reguły filtrujące,
- reguły translacji NAT,
- reguły do manipulacji zaawansowanymi opcjami protokołu IP.

Dla przykładu, w regułach filtrujących akcją jest najczęściej odrzucenie (**DROP**) lub akceptacja wiadomości (**ACCEPT**). Wszystkie reguły tego samego typu pamiętane są w strukturze zwanej tablicą (*ang. table)*, od której pakiet bierze swą nazwę. Istnieją zatem trzy wbudowane tablice:

- filter (dla reguł filtrujących),
- nat (dla reguł translacji NAT),
- mangle (dla reguł manipulujących opcjami IP).

Datagramy IP również dzielą się na kilka rodzajów. Rodzaj datagramu zależy od kombinacji adresów źródłowego i docelowego w nim zawartych. Wyróżnia się zatem:

- datagramy, dla których dana stacja jest celem,
- datagramy, których dana stacja jest źródłem,
- kombinację rodzajów 1 i 2,
- datagramy rutowane.

W zależności od rodzaju datagramu, jego przetwarzanie przebiega w kilku etapach. Na przykład dla datagramów rutowanych etapy te w uproszczeniu wyglądają następująco:

1 – pakiet dociera do interfejsu wejściowego,

2 – zostaje wybrany dla niego właściwy interfejs wyjściowy,

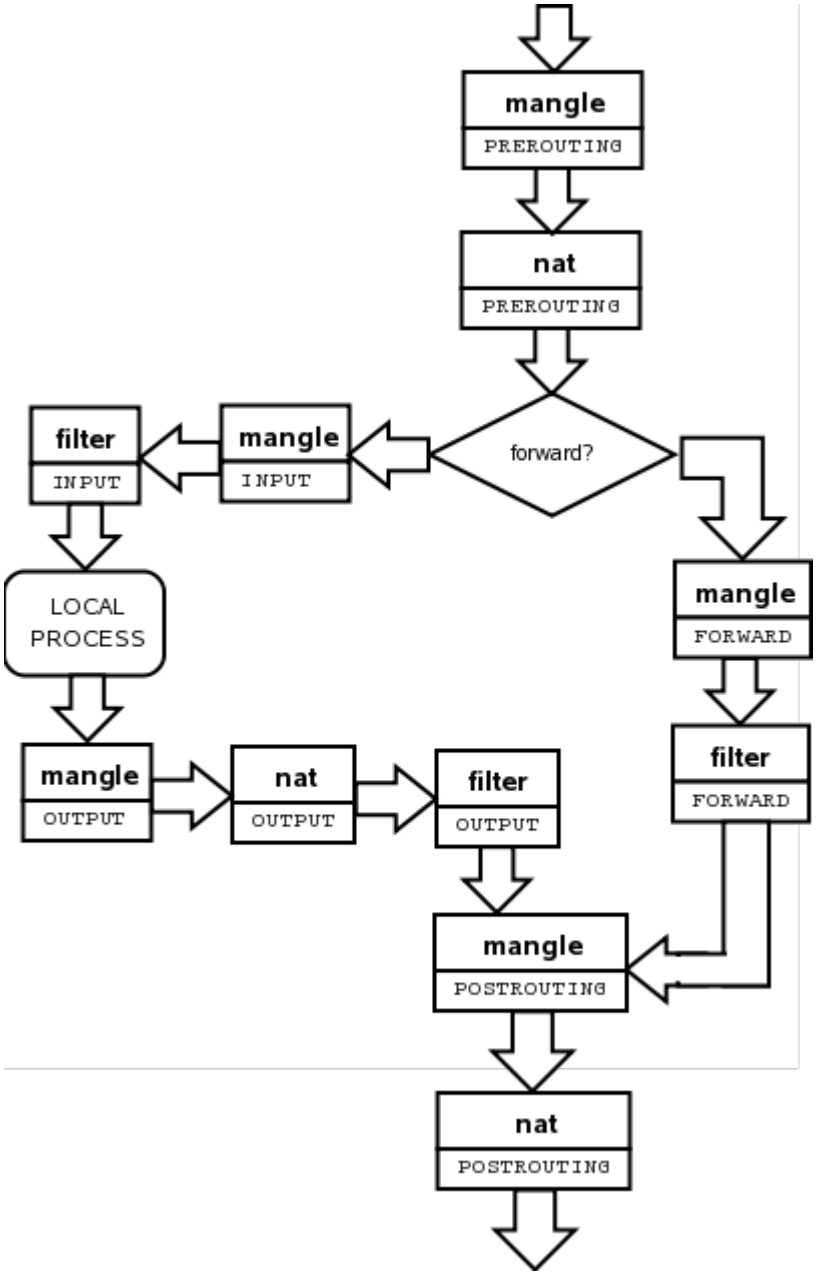
3 – pakiet opuszcza stację przez interfejs wyjściowy.

Na każdym etapie można stosować do datagramu różne typy reguł. Dlatego wewnątrz każdej tablicy reguły są dodatkowo grupowane ze względu na etap przetwarzania pakietu. Wobec tego różne tablice mogą zawierać grupy reguł związanych z tym samym etapem. Sekwencja reguł powstała przez połączenie tych grup nazywa się **łańcuchem** (*ang. chain*). Pakiet iptables zawiera pięć wbudowanych łańcuchów:

- INPUT (dla datagramów pierwszego rodzaju),
- OUTPUT (dla pakietów drugiego rodzaju),
- FORWARD (dla pakietów rutowanych)
- PREROUTING,
- POSTROUTING.

Dwa ostatnie łańcuchy zawarte są jedynie w tablicach **nat** i **mangle**.

Diagram przepływu pakietów przez zaporę sieciową IPTABLES



Składnia polecenia iptables (składnia reguły)

Ogólna postać reguły:

iptables [-t tablica] komenda [wzorzec] [akcja]

Komendy:

- -P *łańcuch polityka* - ustawienie domyślnej polityki dla łańcucha. Domyślna polityka stosowana jest dopiero wówczas, gdy pakiet nie pasuje do żadnej reguły łańcucha,
- -A *łańcuch* - dodanie reguły do określonego łańcucha,
- -I *łańcuch* [*nr reguły*] – wstawienie reguły do określonego łańcucha, jeśli zostanie podany nr reguły wtedy reguła zostanie wpisane w to miejsce zmieniając kolejność pozostałych,
- -L [*łańcuch*] – wyświetlenie wszystkich reguł łańcucha (lub wszystkich łańcuchów w danej tablicy); często używane z opcjami -n i -v,
- -F [*łańcuch*] – usunięcie wszystkich reguł łańcucha (lub ze wszystkich łańcuchów danej tablicy),
- -D *łańcuch* [*nr reguły*] – usunięcie konkretnej reguły w określonym łańcuchu, jeśli zostanie podany nr reguły wtedy zostanie usunięta reguła o tym numerze,
- -R *łańcuch nr_reguły* – zastąpienie reguły numer w określonym łańcuchu.

Opcje wzorca:

- -s [!] *ip*[*/netmask*] – adres IP źródłowy (może być uogólniony do adresu sieci),
- -d [!] *ip*[*/netmask*] – adres IP docelowy (może być uogólniony do adresu sieci),
- -p [!] *protokół* – wybór protokołu: tcp, udp, icmp lub all (wszystkie protokoły stosu TCP/IP),
- -i [!] *interfejs wejściowy*
- -o [!] *interfejs wyjściowy*
- --sport [!] *port*:[*port*] – port źródłowy,
- --dport [!] *port*:[*port*] – port docelowy,
- -m *moduł* – załadowanie modułu rozszerzającego, dzięki temu można wykorzystać kolejne opcje danego modułu.

Najważniejsze akcje:

- -j ACCEPT – przepuszczenie dopasowanych pakietów,
- -j DROP – usunięcie dopasowanych pakietów,
- -j REJECT – odrzucenie dopasowanych pakietów,
- -j LOG – logowanie dopasowanych pakietów, bez usunięcia ich z łańcucha,
- -j RETURN – usunięcie pakietu z łańcucha, pozwalając jednocześnie, aby dalej był pakiet sprawdzany w kolejnych łańcuchach,
- -j SNAT – translacja adresów źródłowych,
- -j DNAT – translacja adresów docelowych,
- -j SAME – translacja adresów źródłowych i docelowych,

- -j REDIRECT – przekierowanie pakietów do lokalnego systemu,
- -j MASQUERADE – translacja adresów źródłowych na adres dynamicznie przyznawany na interfejsie.

Translacja adresów

Technologia translacji adresów sieciowych (ang. *Network Address Translation, w skrócie NAT*) została po raz pierwszy zaproponowana przez firmę Cisco i w roku 1993 opisana w dokumencie RFC1631. Od końca lat 80-tych liczba komputerów dołączanych do sieci Internet wzrasta wykładniczo. To spowodowało, że przestrzeń adresów IP już w latach 90-tych się wyczerpywała. Zaobserwowano wówczas, że jeśli tempo zużycia przestrzeni IP się utrzyma, to w niedalekiej przyszłości (przewidywanej na lata 1994-95) zupełnie zabraknie publicznych adresów IP dla nowobudowanych sieci. Technologia NAT i leżące u jej podstaw nowe podejście do projektowania sieci IP, pozwoliły przedłużyć tę „niedaleką przyszłość" co najmniej do dziś. Poniżej opisana została idea technologii NAT.

Wiele dzisiejszych sieci IP nie opiera się już na publicznych adresach sieci, bo takich już brakuje. Zamiast tego stosowane są dla tych sieci adresy prywatne. Należą do nich:

• dla klasy A – adresy z zakresu 10.0.0.0 - 10.255.255.255

• dla klasy B – adresy z zakresu 172.16.0.0 - 172.31.255.255

• dla klasy C – adresy z zakresu 192.168.0.0 - 192.168.255.255

Ponieważ powyższe adresy nie są rejestrowane, może istnieć wiele sieci wykorzystujących tę samą pulę adresów prywatnych, a to czyni sieć Internet skalowalną. Ale z drugiej strony fakt, że adresy prywatne nie są unikalne, stwarza pewne ograniczenie. Otóż pakiety z takimi adresami nie mogą być przesyłane w Internecie, gdyż nie można określić dla nich trasy. I właśnie ten problem rozwiązuje technologia NAT. Dzięki niej do uzyskania łączności sieci prywatnej z Internetem wystarcza jeden adres publiczny IP – adres zewnętrznego interfejsu rutera. Od strony Internetu cała sieć prywatna widziana jest pod tym jednym (lub wieloma) adresem. Dalej opisano dwie główne odmiany NAT: translację adresów źródłowych SNAT (ang. *Source NAT*) i adresów docelowych DNAT (ang. *Destination NAT*).

1. SNAT

Translacja SNAT umożliwia komputerom w sieci prywatnej dostęp do Internetu, a dokładniej – do usług oferowanych przez różne serwery. Oznacza to, że pakiety, których źródłem są komputery sieci prywatnej, powinny docierać do tych serwerów, a odpowiedzi – wracać z powrotem. Proces translacji SNAT przebiega w dwóch krokach:

- gdy pakiet opuszcza sieć źródłową, ruter zamienia prywatny adres IP źródła na adres swojego publicznego interfejsu oraz numer portu źródłowego na inny, nie zajęty numer portu. Zapamiętuje tę zmianę w tablicy translacji w postaci odwzorowania oryginalny_adres_IP:oryginalny_port→nowy_adres_IP:nowy_port. Dzięki zamianie adresu źródłowego na publiczny można dla pakietu z odpowiedzią jednoznacznie określić trasę. Nowy numer portu zaś pozwala dodatkowo rozróżniać poszczególne adresy prywatne, dzięki czemu z usług Internetu może jednocześnie korzystać wiele komputerów sieci prywatnej.
- gdy ruter otrzymuje pakiet-odpowiedź, publiczny adres docelowy (będący adresem źródłowym w poprzednim pakiecie) wraz z numerem portu docelowego są zamieniane z

powrotem na oryginalne wartości zgodnie z zapamiętanym wcześniej odwzorowaniem. Dzięki temu odpowiedź dociera do właściwego procesu na właściwym komputerze.

2. DNAT

Translacja DNAT umożliwia komputerom z sieci publicznej dostęp do usług oferowanych przez serwery znajdujące się w sieci prywatnej. Oznacza to, że pakiety, których źródłem są komputery sieci publicznej, powinny docierać do tych serwerów, a odpowiedzi – wracać z powrotem. Usługi sieci prywatnej dostępne są dla sieci zewnętrznej pod publicznym adresem interfejsu rutera. Translacja DNAT przebiega w dwóch krokach:

- gdy pakiet dociera do sieci prywatnej, jego adres IP docelowy jest ustawiony na publiczny adres zewnętrznego interfejsu rutera; pod tym adresem bowiem usługa jest dostępna. Ruter zamienia ten adres na właściwy adres serwera; w najprostszym przypadku numer portu docelowego nie ulega zmianie.
- gdy ruter otrzymuje pakiet-odpowiedź, z powrotem zamienia prywatny adres źródłowy na adres publiczny swojego zewnętrznego interfejsu.

Moduły rozszerzające IPTABLES

NetFilter jest zbudowany z modułów, które rozszerzają możliwości zapory sieciowej. Kilka z nich jest używane domyślnie, na przykład tcp, udp. Poniżej krótko omówiono większość z dostępnych modułów pakietu NetFilter. Wiele z wymienionych modułów dostępna jest w dodatku *patch-o-matic-ng*, który nie zawsze jest całkowicie instalowany.

3. Krótkie opisy modułów

1. Dotyczące IPSec:

- ah – dopasowują się do pola “spi” w nagłówku AH pakietu,
- esp – dopasowują się do pola “spi” w nagłówku ESP pakietu,
- policy – dopasowują się do “policy” pakietu.

2. Dopasowujące się do nagłówka pakietu IP:

- dscp – dopasowuje się do 6 bitów DSCP znajdujących się w polu ToS,
- ecn – sprawdza bit ECN w nagłówku pakietu,
- ipv4options (*) – sprawdza różne opcje nagłówka pakietu, np. source routing, record route,
- tcpmss – sprawdza pole MSS (Maximum Segment Size),
- tos – pole ToS (Type of Service),
- ttl – pole TTL (Time To Live).

3. Nadzorujące połączenia (stanowość):

- state – umożliwia zidentyfikowanie istniejących połączeń oraz nowych,
- conntrack – rozszerza możliwości modułu state,
- helper – pozwala określić "pomocnika" do przekazywania połączenia, np. standardowym pomocnikiem jest moduł pozwalający przekazywać połączenia ftp-data w trybie aktywnym.

4. Określające typ pakietu:

- icmp – dopasowuje się do pakietów typu ICMP,
- tcp – dotyczy pakietów typu TCP,
- udp – dotyczy pakietów typu UDP,
- unclean(*) – dotyczy pakietów zniszczonych i niepoprawnych.

5. Określające limity:

- connlimit(*) – umożliwia określenie maksymalnej liczby połączeń do konkretnego adresu IP z jednego adresu IP klienta,
- connrate(*) - umożliwia określenie maksymalnego/minimalnego transferu,
- hashlimit – umożliwia określenie maksymalnego transferu do danej usługi/serwera.

6. Znakowanie pakietów:

- mark – umożliwia znalezienie oznakowanego pakietu,
- connmark – umożliwia znalezienie każdego pakietu związanego z oznakowanym połączeniem.

7. Ułatwiające tworzenie reguł:

- iprange – umożliwia wpisanie zakresu adresów IP,
- mac – umożliwia sprawdzenie adresu sprzętowego karty sieciowej MAC,
- multiport – umożliwia wpisanie zakresu portów,
- owner – dla lokalnych połączeń umożliwia określenie który użytkownik wysłał dany pakiet,
- pkttype – określa typ pakietu (unicast, multicast, broadcast)
- set(*) – wykorzystanie stworzonych zbiorów adresów (polecenie ipset),
- time(*) – określenie daty i/lub czasu,
- comment – umożliwia dopisanie komentarza do każdej reguły.

8. Rozszerzone sprawdzanie pakietów:

- length – sprawdzające wielkość pakietu,
- string(*) – dopasowujące się do zawartości pakietu w polu DATA,

- ipp2p(*) – dopasowujące się do ruchu generowanego przez aplikacje typu P2P.

9. Określające częstość:

- limit – określa jak często reguła będzie dopasowana, np. 3 razy na sekundę – 3/s,
- nth(*) – określa co jaką liczbę pakietów będzie dopasowana reguła, np. co 5 pakiet,
- random(*) – określa dopasowanie reguły do pakietu z zadanym prawdopodobieństwem, np. 50% pakietów.

10. Monitorujące:

- recent – tworzy listy adresów wykorzystających łącze (przechodzących przez tą regułę), wyniki zapisywane są w katalogu /proc/net/ipt_recent/NAZWA,
- account(*) – zlicza ruch do określonych adresów, wyniki w katalogu /proc/net/ipt_account/NAZWA,
- quota(*) – określa quota'ę dla określonych pakietów.

11. Inne:

- condition(*) – pozwala warunkować stosowanie reguły, poprzez plik w katalogu /proc/net/ipt_condition/NAZWA
- osf(*) – odczytuje pasywnie “odcisk palca” (fingerprint) konkretnego adresu IP i zapisuje w pliku /proc/sys/net/ipv4/osf
- psd(*) – pozwala wykryć skanowanie portów TCP i UDP.

(*) moduły te nie znajdują się w domyślnej instalacji systemu SuSE Linux 10.0 i nie są dostępne w laboratorium ćwiczeniowym.

4. Krótkie opisy celów (TARGETs)

Cele określają co zostanie zrobione z danym pakietem, domyślnymi celami są akceptacja (ACCEPT) i odrzucenie (DROP) pakietu. Jednak tworcy zapory postanowili rozszerzyć listę dopuszczalnych celów, które również mogą być tworzone przez nas samych dla naszych potrzeb.

Lista celów została podzielona na kategorie:

1. Zmiana adresów IP:

- BALANCE(*) – podobne do DNAT, ale równomiernie rozkłada obciążenie na wiele adresów IP,
- DNAT – Destination NAT,
- MASQUERADE – ukrywanie źródłowych adresów IP,
- NETMAP – statyczna zamiana adresów całych podsieci,
- REDIRECT – przekierowanie ruchu na lokalny komputer,

- SAME – podobne do DNAT/SNAT, ale zawsze przydziela te same adresy IP konkretnym klientom,
- SNAT – Source NAT.

2. Znakowanie pakietów:

- CONNMARK – znakuje połączenia,
- IPMARK(*) – znakowanie pakietów na podstawie adresu IP,
- MARK – znakuje pakiety.

3. Zmieniające nagłówek pakietu:

- DSCP – umożliwia zmianę 6 bitów DSCP pola ToS,
- ECN – umożliwia usunięcie bitu ECN,
- IPV4OPTSSTRIP(*) – usunięcie wszystkich opcji IP z nagłówka pakietu,
- TCPMSS – ustawienie pola MSS,
- TOS – ustawienie pola ToS,
- TTL – ustawienie pola TTL.

4. Śledzenie pakietów:

- NOTRACK – wyłącza śledzenie połączenia,
- TARPIT(*) – przechwytuje połączenia i zawiesza je, w celu zabezpieczenie się przez skanowaniem typu Code Red i Nimda,
- TRACE(*) – włącza śledzenie połączenia.

5. Logowanie:

- LOG – logowanie pakietów,
- ULOG – logowanie w przestrzeni użytkownika,

6. Inne:

- REJECT – odrzuca pakiety wysyłając określony kod błędu do nadawcy,
- ROUTE(*) – pozwala nadpisać wpisy tablicy tras,
- SET(*) – dodaje/usuwa adresy z zakresów (polecenie ipset),
- XOR(*) – pozwala zastosować proste zabezpieczenie danych pakietu, poprzez wykonanie funkcji XOR na danych i określonym haśle.

5. Szczegółowe informacje

Wszystkie dodatkowe informacje odnoście każdego z wyżej wymienionych modułów i celów można uzyskać z plików pomocy.

Uzyskanie pomocy o module:

```
iptables -m <moduł> -h
```

Uzyskanie pomocy o celu:

```
iptables -j <cel> -h
```

3. Przykłady

Ustawienie domyślnej polityki łańcucha INPUT na DROP:

```
iptables -P INPUT DROP
```

Akceptacja pakietów o adresie źródłowym 150.254.20.20:

```
iptables -A INPUT -s 150.254.20.20 -j ACCEPT
```

Translacja adresów źródłowych dla podsieci 192.168.1.0/24 na adres 150.254.20.20

```
iptables -t nat -I PREROUTING -s 192.168.1.0/24 -j SNAT \
--to 150.254.20.20
wpisy równoważne
iptables -t nat -I PREROUTING -s 192.168.1.0/24 -j SAME \
--to 150.254.20.20
```

Translacja adresów docelowych z 150.254.20.21 na 192.168.1.2:

```
iptables -t nat -A POSTROUTING -d 150.254.20.21 -j DNAT \
--to 192.168.1.2
wpisy równoważne
iptables -t nat -A POSTROUTING -d 150.254.20.21 -j SAME \
--to 191.168.1.2
```

Translacja adresów źródłowych na zakres adresów:

```
iptables -t nat -A PREROUTING -s 192.168.1.0/24 -j SNAT \
--to 150.254.20.20 - 150.254.20.30
wpisy równoważne
iptables -t nat -A PREROUTING -s 192.168.1.0/24 -j SAME \
--to 150.254.20.20 - 150.254.20.30
```

Translacja adresów i portów docelowych, jedynie dla wybranego portów:

```
iptables -t nat -A POSTROUTING -p tcp -d 150.254.20.21 \
--dport 80 -j DNAT -to 192.168.1.3:8080
wpisy równoważne
iptables -t nat -A POSTROUTING -p tcp -d 150.254.20.21 \
--dport 80 -j SAME -to 192.168.1.3:8080
```

Akceptacja pakietów ustanowionych i związanych z innymi połączeniami:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED \
-j ACCEPT
```

Blokada więcej niż 2 połączeń na port 23

```
iptables -A INPUT -p tcp --syn --dport 23 -m connlimit \
--connlimit-above 2 -j REJECT \
--reject-with icmp-host-unreachable
```

Umożliwienie tylko jednego na minutę połączenia z portem 22

```
iptables -A INPUT -p tcp --dport 22 -m hashlimit \
--hashlimit 1/min --hashlimit-mode srcip \
--hashlimit-name ssh -m state --state NEW -j ACCEPT
```

Blokada pakietów większych niż 1000 bajtów

```
iptables -A INPUT -m length --length 1000: -j DROP
```

Akceptacja 3 połączeń na minutę

```
iptables -A INPUT -m limit --limit 3/min -j ACCEPT
```

Stworzenie statystyk połączeń z portem 80

```
iptables -A INPUT -p tcp --dport 80 -m recent --name www \
--set -j ACCEPT
```

Utworzenie zbiorów adresów prywatnych i ich blokowanie

```
ipset --create priv nethash
ipset --add priv 10.0.0.0/8
ipset --add priv 172.16.0.0/12
ipset --add priv 192.168.0.0/16
ipset --add priv 169.254.0.0/16
iptables -A INPUT -m set --set priv src -j DROP
```

4. Zadania

Wykorzystać zaprezentowane powyżej informacje do zbudowania różnego typu reguł oraz przetestowanie ich poprawnego działania:

- Ograniczyć maksymalną wielkość pakietu ICMP-echo do 1kB.
- Ograniczyć do 3 na minutę liczbę pakietów ICMP-echo.
- Utworzyć statystyki odbierania pakietów.
- Ograniczyć żądania HTTP do 2kB wielkości i do 3 na minutę.
- Logować pakiety, które naruszają regułę numer 4, ale ich nie usuwać.
- Zmienić wartość pola TTL dla każdego pakietu na 56 i sprawdzić wykorzystując sniffer.
- Usuwać pakiety większe niż 3 kB zwracając komunikat błędy ICMP-net-unreachable.
- Zmienić wartość pola MSS – sprawdzić wykorzystując sniffer.
- Wykorzystać moduł comment w celu opisania reguł zapory sieciowej.
- Wykorzystać moduł pkttype w celu zablokowania pakietów ICMP-echo wysyłanych na adres rozgłoszeniowy. Wcześniej należy zdjąć blokadę jądra systemu:

```
echo ”0” > /proc/sys/net/ipv4/icmp_echo_ignore_broadcast
```

5. Problemy do dyskusji

- Architektura IPTABLES, wady i zalety.

- Modułowość IPTABLES zaleta czy wada?
- Mnogość łańcuchów i tablic w iptables, czy ma to swoje uzasadnienie?
- Jaki jest związek między "szczelnością" a elastyczności reguł filtracji?
- Technologia NAT, dobre, złe rozwiązanie?
- Inne metody rozwiązania wyczerpujących się się adresów Ipv4?
- SNAT, DNAT dlaczego każdy z nich może być wykonywany tylko w określonych łańcuchach?
- Czy wszystkie z opisanych modułów rozszerzających są warte używania, proszę o podanie przykładów oraz dyskusji dlaczego?

6. Bibliografia

[WWW1] http://netfilter.samba.org

[WWW2] http://www.linuxdoc.org/HOWTO/mini/TransparentProxy.html

[WWW3] http://www.squid-cache.org

[WWW4] http://mr0vka.eu.org/tlumaczenia/index.htm

[WWW5] http://www.netfilter.org

[T] A. S. Tanenbaum, "Computer Network", Prentice Hall, 1996

[KR] J. Kurose, K. Ross, "Computer Networking – A Top-Down Approch Featuring the Internet", Addison Wesley Longman, Inc., 2001