

IPTABLES

by Atom_Zero

Spis treści

- 1. Wstęp**
- 2. Konfiguracja jądra**
- 3. Instalacja**
- 4. Droga pakietu**
- 5. Opis tabel**
- 6. Śledzenie połączeń**
 - a) Połączenia TCP**
 - b) Połączenia UDP**
 - c) Komunikaty ICMP**
 - d) Śledzenie połączeń a usługa FTP**
- 7. Budowanie Reguł**
- 8. Popularne usługi oraz przykładowy skrypt rc.firewall.**
- 9. Bibliografia / sznurki**

1. Wstęp

Dokument ten ma za zadanie przybliżyć, oraz pomóc w zrozumieniu możliwości filtra sieciowego jakim jest iptables.

2. Konfiguracja Jądra

`Config_Netfilter` – Ta opcja jest wymagana jeżeli chcemy by komputer był firewallem lub routerem.

`Config_Packet` – Opcja potrzebna by działały takie aplikacje jak TCPDUMP lub SNORT. Ta opcja nie jest wymagana by nam IPTABLES działało, ale ze względu na użyteczność tego modułu, dodałem go tu.

`Config_IP_NF_Conntrack` – Śledzenie połączeń. Jeżeli chcemy użyć firewalla dla LAN to zaznaczamy tę opcję.

`Config_IP_NF_FTP` – Śledzenie połączeń dla FTP. Bez niego połączenie FTP może nam zniknąć przy FireWall'u, co uniemożliwi nam korzystanie z tej usługi.

`Config_IP_NF_Iptables` – Obsługa Iptables – raczej konieczne.

`Config_IP_NF_Match_Limit` – Pozwala wprowadzić ograniczenie w ilości przesyłanych pakietów na minute – dobre przeciwko atakom DoS i DDoS.

`Config_IP_NF_Match_MAC` – Pozwala nam wprowadzać ograniczenia po adresach MAC kart sieciowych.

`Config_IP_NF_Match_Mark` – Pozwala nam używać w tabeli MANGLE w iptables znaczników, których możemy użyć jako uchwytów w programie Traffic Control (TC).

`Config_IP_NF_Match_Multiport` – Pozwala nam porównywać pakiety poprzez wprowadzenie zakresu portów.

`Config_IP_NF_Match_ToS` – Pozwala nam na porównywanie pakietów na podstawie bitów ToS pozwala także na przerabianie ich w tabeli Mangle – używane w rutingu.

`Config_IP_NF_Match_TCPMSS` – Pozwala porównywać pakiety na podstawie maksymalnego rozmiaru segmentu.

`Config_IP_NF_Match_State` – Pozwala porównywać pakiety na podstawie stanu połączeń.

`Config_IP_NF_Match_Unclean` – Pozwala nam decydować o losie pakietu który w ogóle nie pasuje do protokołów. Ten moduł jest eksperymentalny i nie zawsze dobrze działa.

`Config_IP_NF_Match_Owner` – Pozwala nam decydować o pakiecie na podstawie UID użytkownika (może różnie działać).

`Config_IP_NF_Filter` – Obsługa tabeli Filter (firewall) w iptables – raczej konieczne.

`Config_IP_NF_Mangle` – Obsługa tabeli Mangle w iptables – też dobrze by było to mieć

`Config_IP_NF_NAT` – Obsługa tabeli NAT w iptables – konieczne dla LAN lub DMZ.

`Config_IP_NF_Target_Reject` – Pozwala na wysłanie stosownej wiadomości ICMP zamiast robić sam DROP na pakiecie.

`Config_IP_NF_Target_Mirror` – Pozwala na odbicie pakietów do nadawcy. Np. jeżeli dla łańcucha INPUT oraz dla portu docelowego SSH ustawimy politykę MIRROR to delikwent zaloguje się na swojej własnej maszynie.

`Config_IP_NF_Target_Masquerade` – Potrzebne dla translacji adresów gdzie maszynom przypisywane jest dynamiczne adresy IP (np. mamy serwer DHCP, używamy protokołu PPP lub SLIP)

`Config_IP_NF_Target_Redirect` – Pozwala przekierowywać pakiet najpierw do naszego PROXY a następnie do naszej sieci. Tworzymy w ten sposób Transparent PROXY.

`Config_IP_NF_Target_LOG` – Pozwala nam na logowanie zdarzeń do syslog'a. Raczej potrzebne ze względów na wprowadzanie poprawek do naszego skryptu.

`Config_IP_NF_Target_TCPMSS` – Potrzebne dla ISP, którzy wymagają dzielenia przesyłki na mniejsze segmenty. Poznaje się to po tym, że nie wszystkie strony się nam ładują, lub mniejsze e-maile docierają a większe nie. Ten moduł pozwala nam zmniejszyć rozmiar segmentu.

`Config_IP_NF_Compat_IPCHAINS`, `Config_IP_NF_Compat_IPFWADM` – Wymuszenie kompatybilności starszych firewalli z nowszymi jądrami. Jądra z serii 2.4.x nie powinny mieć z tym problemu, to jest zrobione z myślą jąder 2.6.x. Lepiej sobie tym głowy nie zawracać.

3. Instalacja

Pakiet iptables można pobrać ze strony projektu: www.netfilter.org. Aczkolwiek dystrybucja Debian posiada go na płytach i można go zainstalować narzędziem apt-get:

```
Debian@root:# Apt-get install iptables
```

Ewentualnie możemy wykorzystać program dselect :

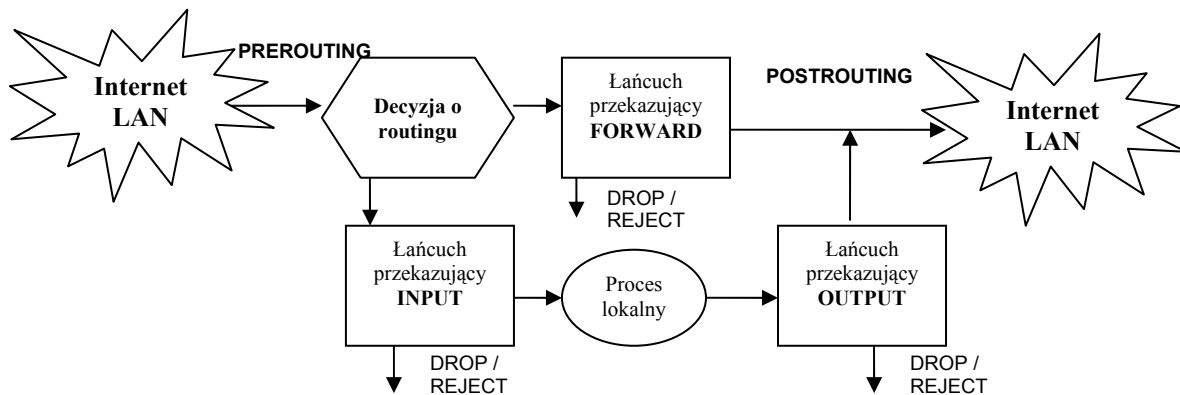
```
dselect - lista pakietów (dostępny, priorytet) wybór: +/-/- szczegóły: v pomoc: ?
EIOM Pri Sekcja Pakiet Zainst. Dostępne Opis
*** Zwy net ipmasqadm 0.4.2-2 0.4.2-2 Utility for configuring e
*** Zwy net iptables 1.2.6a-5 1.2.6a-5 IP packet filter administ
*** Zwy net libdns5 9.2.1-2.wo 9.2.1-2.wo DNS Shared Library used b
*** Zwy net libisc4 9.2.1-2.wo 9.2.1-2.wo ISC Shared Library used b
*** Zwy net liblwres1 9.2.1-2.wo 9.2.1-2.wo Lightweight Resolver Libr
*** Zwy net lpr 2000.05.07- 2000.05.07- BSD lpr/lpd line printer
*** Zwy net mime-support 3.18-1 3.18-1 MIME files 'mime.types' &
*** Zwy net mtr-tiny 0.48-1 0.48-1 Full screen ncurses trace
*** Zwy net nfs-common 1.0-2 1.0-2 NFS support files common
*** Zwy net pidentd 3.0.12-4 3.0.12-4 TCP/IP IDENT protocol ser
iptables zainstalowany ; instalacja (był: instalacja). Zwyczajne
iptables - IP packet filter administration tools for 2.4.4+ kernels

netfilter and iptables are the framework inside the Linux 2.4.x kernel
which enable packet filtering, network address translation (NAT) and other
packet mangling.

netfilter is a set of hooks inside the linux 2.4.x kernel's network stack
which allows kernel modules to register callback functions called every
time a network packet traverses one of those hooks.

opis pakietu iptables -- 47%, wciśnij d - więcej.
```

4. Droga pakietu



Pakiet przychodzi z lewej strony. Po zweryfikowaniu sumy kontrolnej IP pakiet wpada do łańcucha `NF_IP_PRE_ROUTING` (w skrócie `PREROUTING`). Następnie zostaje określona trasa routingu dla pakietu tzn. zapada decyzja, czy pakiet jest skierowany do procesu lokalnego czy może musi być przesłany (rutowany) gdzie indziej. Jeżeli pakiet jest pakietem o adresie docelowym mówiącym, że ma trafić do procesu lokalnego (działającego na maszynie z firewallem) jest on przesyłany do łańcucha `NF_IP_LOCAL_IN` (w skrócie `INPUT`). Jeżeli jednak pakiet ma trafić do innej maszyny musi zostać przesłany poprzez łańcuch `NF_IP_FORWARD` (w skrócie `FORWARD`). Łańcuch `NF_IP_LOCAL_OUT` jest wykorzystywany dla pakietów generowanych lokalnie i wysyłanych na inne maszyny w sieci. Kiedy pakiet wychodzi z firewalla musi przejść przez jeszcze jeden łańcuch zwany `NF_IP_POST_ROUTING` (w skrócie `POSTROUTING`). Pakiety przechodzące przez łańcuchy `FORWARD` i `OUTPUT` przechodzą przez ten ostatni łańcuch.

5. Opis tabel

Pakiet `iptables` pozwala nam zrobić z pakietami 3 rzeczy;

1. Filtrowanie pakietów (tabela filter) – Tabela ta nie modyfikuje pakietów ale tylko je filtruje. Tu możemy wykorzystać 3 łańcuchy: INPUT, OUTPUT, FORWARD. Wydajemy tu decyzje czy dany pakiet wpuszczamy (ACCEPT). Czy też nie wpuszczamy (DROP/REJECT). DROP odrzuca pakiet bez echa, REJECT informuje użytkownika o braku dostępności sieci.
2. Translację adresów (tabela nat) – Tabela pozwala na modyfikowanie adresów źródłowych lub docelowych, które są częścią nagłówka protokołu IP.

Schemat nagłówka IP:

```

      0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|Version|  IHL  |Type of Service|                               Total Length |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Identification |Flags|           Fragment Offset |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Time to Live |           Protocol |                               Header Checksum |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Source Address                               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Destination Address                               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Options                               |           Padding |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Translację wykonujemy tylko w łańcuchach PREROUTING, POSTROUTING i OUTPUT. W łańcuchu PREROUTING używamy DNAT (Destination NAT), z tego względu że, jeżeli udostępniamy usługę, to musimy przed wydaniem decyzji o trasie pakietu na adres lokalny maszyny która tą usługę udostępnia, np.:

```
Debian@root:# iptables -t nat -A PREROUTING -j DNAT
--to-destination 1.2.3.4:8080 -p tcp --dport -i eth1
```

W łańcuchu POSTROUTING używamy SNAT lub MASQUERADE. Używamy to w sytuacji, kiedy chcemy udostępnić Internet dla maszyn klienckich, z tego względu, że aby klient mógł połączyć się z Internetem to jego nagłówek musi mieć zewnętrzny adres IP, więc adres jest zmieniany po decyzji trasy pakietu. Różnica między SNAT a MASQUERADE polega na tym że SNAT stosujemy wtedy kiedy na sztywno mamy przypisane adresy IP dla maszyn znajdujących się w sieci lokalnych; MASQUERADE stosujemy kiedy nie wiemy jakie adresy będą przypisane klientom w sieci (DHCP). Jednak jeżeli chodzi o prędkość to SNAT jest szybszy, gdyż nie potrzebuje dodatkowych obliczeń dla szukania przypisanego już adresu IP.

```

Debian@root:# iptables -t nat -A POSTROUTING -j
SNAT --to-source 1.2.3.4

Debian@root:# iptables -t nat -A POSTROUTING -j
MASQUERADE -o ppp0

```

W łańcuchu OUTPUT przerabiamy adres pakietu dla pakietów wychodzących z maszyny na której jest zainstalowany iptables.

3. Zmianę właściwości pakietu (tabela mangle) – Tabela mangle dopuszcza na zmienianie właściwości pakietu we wszystkich pięciu łańcuchach: PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING. Tabela pozwala na zmianę bitów TOS oraz TTL. Dodatkowo pozwala na znakowanie (MARK) pakietów, z tym że nie odbywa to się w nagłówku pakietu lecz w przestrzeni jądra systemowego. Moduł MANGLET wykorzystywany jest w zaawansowanym routingu, dzielenia pasma.

Schemat nagłówka IP:

```

      0              1              2              3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|Version|  IHL | Type of Service |                Total Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                Identification |Flags|                Fragment Offset |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Time to Live | Protocol |                Header Checksum |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                Source Address |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                Destination Address |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                Options |                Padding |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

TTL odpowiedzialny jest za długość życia pakietu. Możemy ustawić np.: że pakiet po przejściu przez firewall, długość życia pakietu będzie wynosiła 1, aby pakiet nie poszedł za daleko. Z kolei bity TOS informują o typie usługi. Typy są następujące:

- Minimize-Delay 16 (0x10),
- Maximize-Throughput 8 (0x08),
- Maximize-Reliability 4 (0x04),
- Minimize-Cost 2 (0x02),
- Normal-Service 0 (0x00).

Minimize-Delay 16 (0x10), informuje, że pakiet wymaga najmniejszych opóźnień, wykorzystuje się to dla interaktywnych usług np.: SSH, TELNET, RLOGIN, FTP-Control. Maximize-Throughput 8 (0x08), informuje że pakiet wymaga maksymalnej przepustowości (FTP-Data, SMB, NFS). Maximize-

Reliability 4 (0x04), skupia się na niezawodności połączenia co wykorzystuje się głównie w protokole UDP (TFTP, BOOTP). Minimize-Cost 2 (0x02), informuje, że pakiet wymaga najkrótszej liczby skoków między węzłami (Radio/Video). Z kolei Normal-Service 0 (0x00), nie znaczy nic poważnego i dotyczy się zwykłych usług internetowych. Tak naprawdę te bity mają małe znaczenie gdyż niewielu adminów zaprzęta sobie tym głowę by przez te bity nadawać pewne priorytety połączeń. Więc nie zawsze warto jest tym zajmować się.

6. Śledzenie połączeń

Pakiet `iptables` posiada także możliwość śledzenia połączeń a co za tym idzie określania jego stanu. I tu rozróżniamy cztery stany połączeń **NEW**, **ESTABLISHED**, **RELATED**, **INVALID**. Dzięki modułowi `state` mamy możliwość akceptowania bądź odrzucania połączeń na podstawie ich stanu. Samo śledzenie połączeń odbywa się w łańcuchach `PREROUTING` i `OUTPUT`. Połączenie nadchodzące uznane jest za nowe jeżeli pakiet z ustawioną flagą `SYN` przejdzie przez łańcuch `PREROUTING`. Natomiast połączenie wygenerowane lokalnie jest uznane za nowe jeżeli pakiet z ustawioną flagą `SYN` przejdzie przez łańcuch `OUTPUT`. Więcej szczegółów zostanie omówione dla każdego z protokołów. Ciekawostką jest plik `/proc/net/ip_conntrack`. W nim mamy podaną listę połączeń śledzonych. Podam przykład:

```
tcp      6 117 SYN_SENT src=192.168.1.6 dst=192.168.1.9 sport=32775 \
dport=22 [UNREPLIED] src=192.168.1.9 dst=192.168.1.6 sport=22 \
dport=32775 use=2
```

Przeanalizujmy co my tutaj mamy podane. Pierwsze „tcp” określa protokół. Następna liczba 6 także określa protokół ale poprzez nr indeksu (`tcp = 6`), potem jest podane jak długo połączenie będzie trwać (w sekundach). Ta wartość regularnie maleje. Gdy osiągnie 0 wtedy sprawdzany jest stan połączenia, a następnie jest resetowane do wartości domyślnej. Następnie mamy stan połączenia. Widzimy że na razie został wysłany pakiet z flagą `SYN`. Następnie mamy adresy źródłowy i docelowy, oraz porty źródłowy i docelowy. Potem mamy słowo kluczowe `[UNREPLIED]` które mówi nam że nie mamy odpowiedzi oraz dalej mamy oczekiwaną odpowiedź w postaci odwróconych adresów i numerów portów. Jeżeli połączenie zostaje nawiązane, to wtedy połączenie zostaje zabezpieczone a na miejscu `[UNREPLIED]` mamy słowo `[ASSURED]`.

a) Połączenia TCP

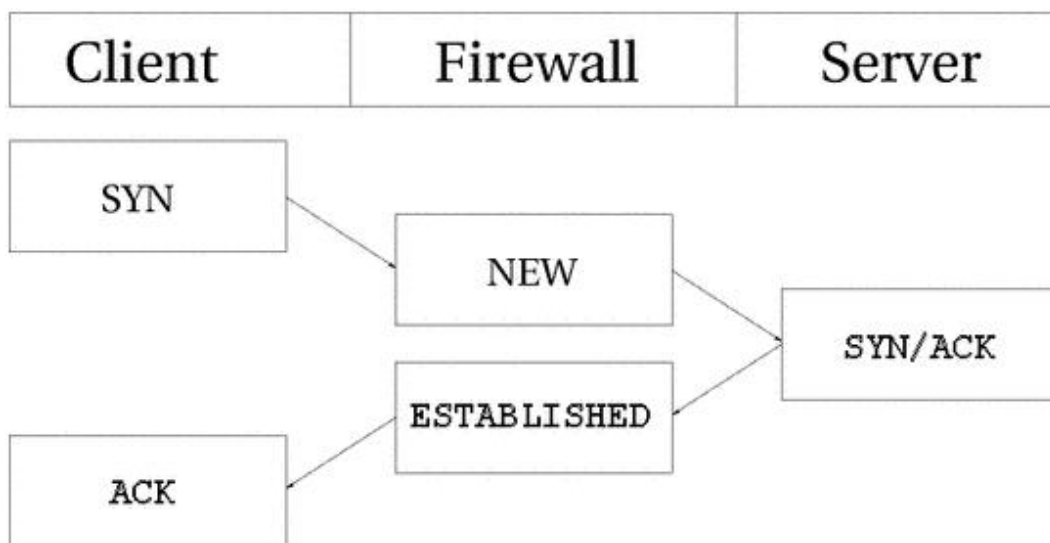
Protokół TCP jest sam w sobie protokołem, który decyduje o stanie połączenia. Oto Nagłówek TCP, w którym widzimy 6 flag `URG`, `ACK`, `PSH`, `RST`, `SYN`, `FIN`:

```
0          1          2          3
0 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Source Port          |          Destination Port          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Sequence Number          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Acknowledgment Number          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Data |          |U|A|P|R|S|F|
| Offset| Reserved |R|C|S|S|Y|I|          Window
|          |          |G|K|H|T|N|N|
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Checksum          |          Urgent Pointer          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Options          |          Padding          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          data          |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Zanim zostanie nawiązane połączenie oparte o protokół TCP, następuje inicjalizacja połączenia odbywająca się w 3 krokach:

- Klient wysyła pakiet z ustawionym bitem SYN
- Serwer odpowiada pakietem z ustawionymi bitami na SYN/ACK
- Klient zatwierdza połączenie wysyłając pakiet z ustawionym bitem ACK.

Po tych trzech krokach następuje przekazywanie danych. Jak to widzi iptables ? Oto rysunek:



Widzimy że dopiero po wysłaniu pakietu z bitem SYN firewall widzi to połączenie jako NEW, a dopiero po odebraniu pakietu SYN/ACK widzi jako połączenie ESTABLISHED. Z widoku Kernela wygląda to tak (`/proc/net/ip_conntrack`):

```
tcp      6 117 SYN_SENT src=192.168.1.5 dst=192.168.1.35 sport=1031 \
dport=23 [UNREPLIED] src=192.168.1.35 dst=192.168.1.5 sport=23 \
dport=1031 use=1
```

Powyżej widzimy już wysłany pakiet SYN. Na razie kernel oczekuje na odpowiedź.

```
tcp      6 57 SYN_RECV src=192.168.1.5 dst=192.168.1.35 sport=1031 \
dport=23 src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 \
use=1
```

Tu Nareszcie kernel dostał odpowiedź.

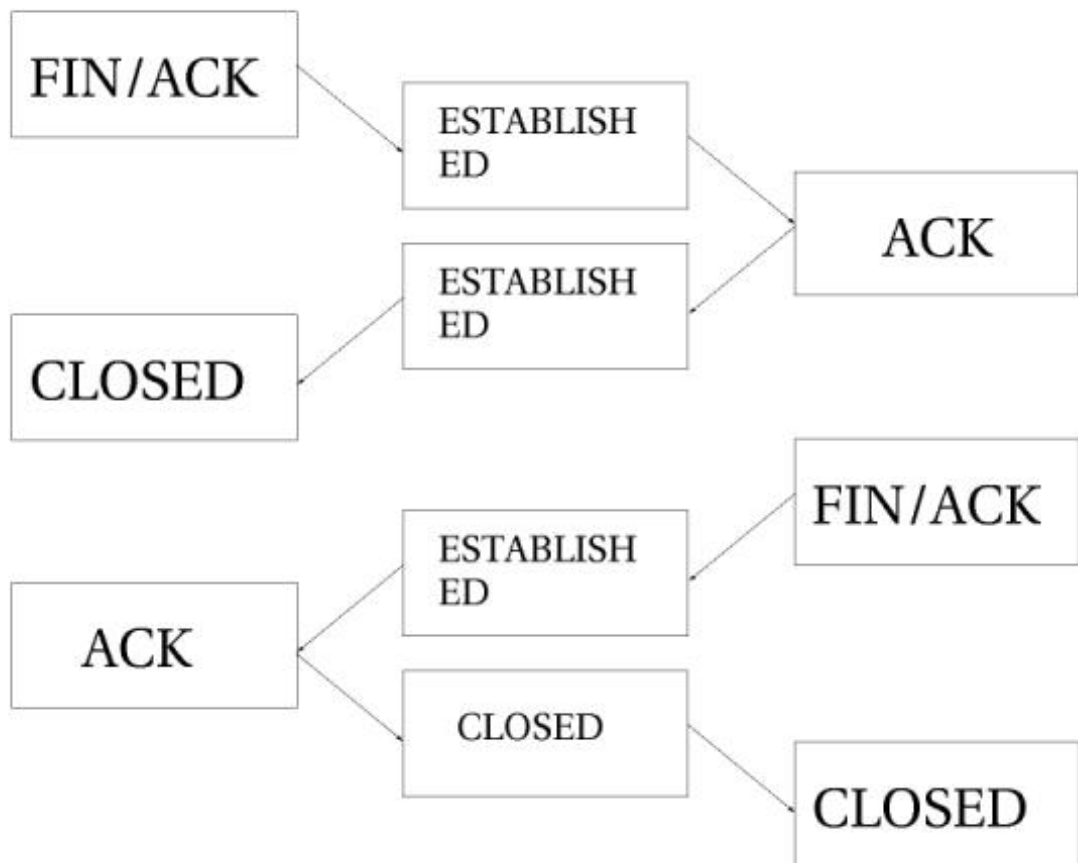
```
tcp      6 431999 ESTABLISHED src=192.168.1.5 dst=192.168.1.35 \  
sport=1031 dport=23 src=192.168.1.35 dst=192.168.1.5 \  
sport=23 dport=1031 use=1
```

W końcu wysłany jest pakiet z bitem ACK. I mamy połączenie ustanowione. Po wysłaniu paru pakietów danych połączenie zostanie zabezpieczone [ASSURED]. Są sytuacje kiedy przychodzi nowy pakiet a nie posiada ustawionego bitu SYN (np. posiadamy drugiego firewalla). Musimy dla sprawdzenia wprowadzić następujące reguły:

```
$IPTABLES -A INPUT -p tcp ! --syn -m state --state NEW -j LOG \  
--log-prefix "Nowy - bez bitu syn:"  
$IPTABLES -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
```

Następnie sprawdzamy w logach pochodzenie tego pakietu. W razie potrzeby zmieniamy DROP na ACCEPT.

A jak wygląda zakończenie połączenia w TCP ? Jak odbiera go iptables ? Oto diagram:



Tutaj widzimy że połączenie nie zostanie zamknięte dopóki nie zostanie wysłany ostatni pakiet ACK. Ten diagram też przedstawia to, że ceremonia obejmuje zamknięcie połączenia najpierw od strony klienta a potem od strony serwera. Jednak to nie jest jedyny sposób na zamknięcie połączenia. Połączenie może być zamknięte przez wysłanie pakietu RST, co spowoduje natychmiastowe zerwanie połączenia.

b) Połączenia UDP

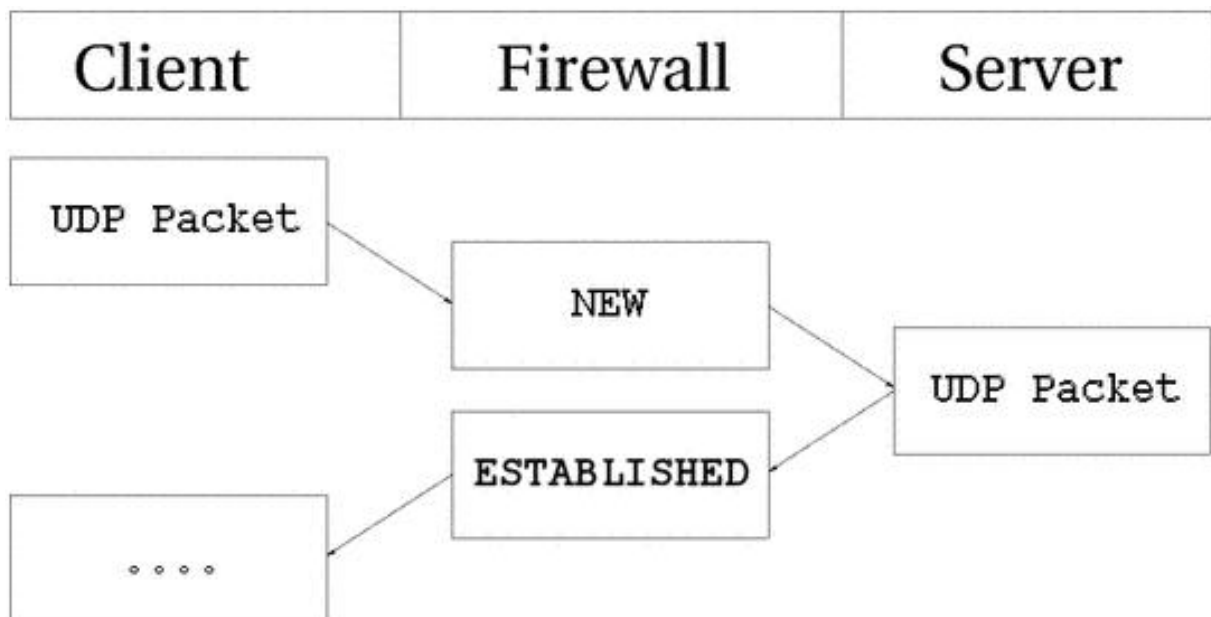
Połączenia oparte o protokół UDP są pozbawione flag informujących o stanie połączenia. Nagłówek wygląda następująco:

```

      0      7 8      15 16      23 24      31
+-----+-----+-----+-----+
|           source address           |
+-----+-----+-----+-----+
|           destination address      |
+-----+-----+-----+-----+
| zero |protocol|  UDP length  |
+-----+-----+-----+-----+

```

Widzimy że nagłówek nie posiada flag odpowiedzialnych za badanie stanu połączenia. Więc jak sobie z tym radzi iptables ? Pokaże nam to następny diagram:



Widzimy, że iptables pomimo różnic w protokołach – działa na tej samej zasadzie. Jak widzi to kernel?

```

udp      17 20 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025 \
[UNREPLIED] src=192.168.1.5 dst=192.168.1.2 sport=1025 \
dport=137 use=1

```

Nic dodać, nic ująć – pierwszy pakiet UDP.

```
udp      17 170 src=192.168.1.2 dst=192.168.1.5 sport=137 \
dport=1025 src=192.168.1.5 dst=192.168.1.2 sport=1025 \
dport=137 use=1
```

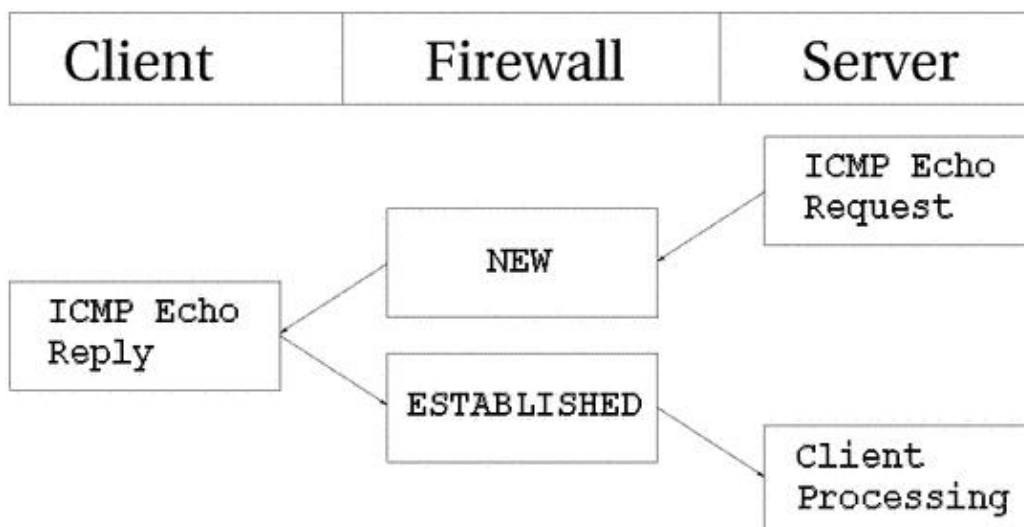
Przybył pakiet ze strony klienta. Czas badania stanu połączenia wydłużył się na 170 sekund.

```
udp      17 175 src=192.168.1.5 dst=195.22.79.2 sport=1025 \
dport=53 src=195.22.79.2 dst=192.168.1.5 sport=53 \
dport=1025 [ASSURED] use=1
```

Po upływie tego czasu połączenie zostaje zabezpieczone przez jądro. Z tym że w odróżnieniu od protokołu TCP. Kernel musi sprawdzać stan co 180 sekund. Dzieje się tak z tego względu że protokół UDP jest protokołem bezpołączeniowym oraz fakt, że ten protokół nie zawiera w nagłówku informacji o ilości przesyłanych danych.

c) Komunikaty ICMP

Komunikaty ICMP są dalekie od protokołu który ma coś wspólnego ze sprawdzaniem stanu połączenia. Ale i z tym „problemem” iptables sobie radzi:



Widzimy tu prostego PING'a. Firewall uznaje za połączenie ustanowione (!) kiedy jest odpowiedź na pinga.

Od strony jądra mamy:

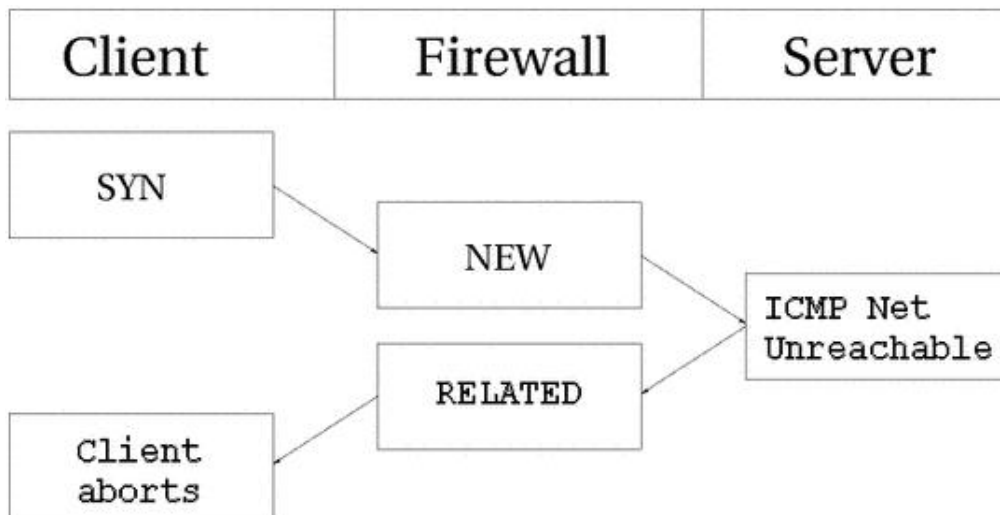
```
icmp      1 25 src=192.168.1.6 dst=192.168.1.10 type=8 code=0 \
id=33029 [UNREPLIED] src=192.168.1.10 dst=192.168.1.6 \
type=0 code=0 id=33029 use=1
```

No i tak naprawdę tu mamy w odróżnieniu od protokołu TCP i UDP 3 dodatkowe pola: `type`, `code`, `id`. Pole `type` i `code` odpowiada za rodzaj komunikatu ICMP. Oto ich pełna lista:

TYPE	CODE	Opis	Query	Error
0	0	Echo Reply	x	
3	0	Network Unreachable		x
3	1	Host Unreachable		x
3	2	Protocol Unreachable		x
3	3	Port Unreachable		x
3	4	Fragmentation needed but no frag. bit set		x
3	5	Source routing failed		x
3	6	Destination network unknown		x
3	7	Destination host unknown		x
3	8	Source host isolated (obsolete)		x
3	9	Destination network administratively prohibited		x
3	10	Destination host administratively prohibited		x
3	11	Network unreachable for TOS		x
3	12	Host unreachable for TOS		x
3	13	Communication administratively prohibited by filtering		x
3	14	Host precedence violation		x
3	15	Precedence cutoff in effect		x
4	0	Source quench		
5	0	Redirect for network		
5	1	Redirect for host		
5	2	Redirect for TOS and network		
5	3	Redirect for TOS and host		
8	0	Echo request	x	
9	0	Router advertisement		
10	0	Route solicitation		
11	0	TTL equals 0 during transit		x
11	1	TTL equals 0 during reassembly		x
12	0	IP header bad (catchall error)		x
12	1	Required options missing		x
13	0	Timestamp request (obsolete)	x	
14		Timestamp reply (obsolete)	x	
15	0	Information request (obsolete)	x	
16	0	Information reply (obsolete)	x	
17	0	Address mask request	x	
18	0	Address mask reply	x	

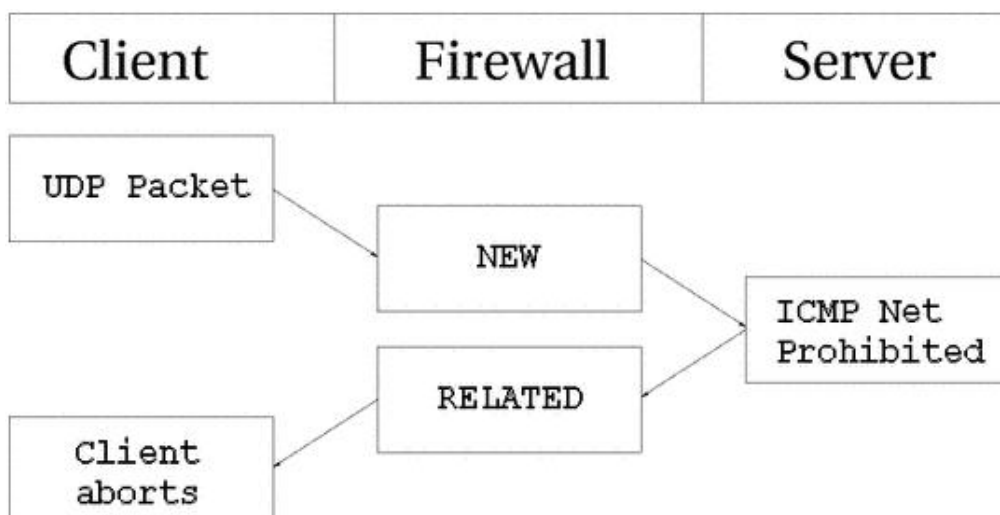
Z kolei `id` odpowiada za identyfikację komunikatu w sieci. Jeżeli host otrzyma pinga o takim numerze to odpowie komunikatem o tym samym `id`. Głównie chodzi o to by się pingi w sieci nie pomieszały.

A jak wygląda sprawa z komunikatami ICMP w przypadku gdy chcemy się połączyć z jakimś hostem w sieci a on nam zwraca komunikat ICMP ? Jak to jest traktowane przez iptables? Kolejny diagram:



Widzimy że tu iptables widzi komunikat ICMP jako połączenie powiązane z połączeniem TCP.

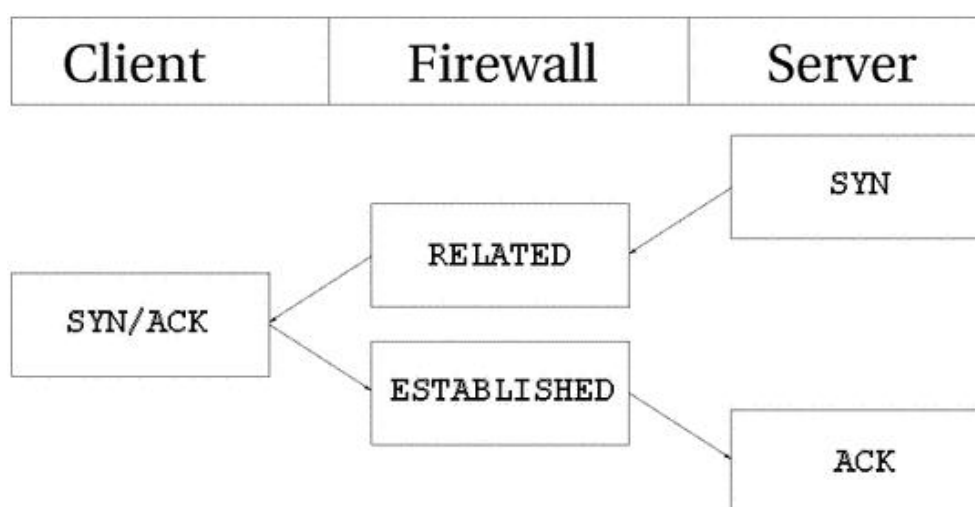
Podobnie z protokołem UDP:



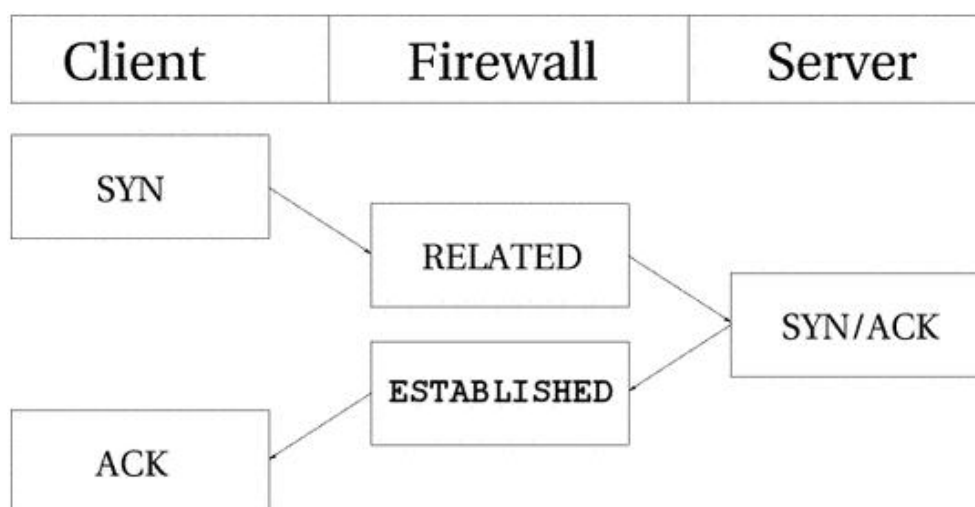
Wniosek z tego jest taki że zamiast zezwalać poszczególnym komunikatom na dotarcie do nas poprzez wpisywanie poszczególnych reguł to możemy sobie pozwolić na jeden wpis który będzie wpuszczał połączenia powiązane.

d) Śledzenie połączeń a usługa FTP

Niektóre usługi są o wiele bardziej złożone niż pozostałe. Do takich należą usługi FTP, IRC oraz ICQ. Każdy z tych protokołów musi przesyłać dane podczas wpisywania rozkazów. Dla przykładu weźmy protokół FTP. Protokół ten najpierw otwiera port odpowiedzialny za sesję. Ale podczas wpisywania i wysyłania komend otwierane są dodatkowe porty, które przesyłają dodatkowe dane na dodatkowo otwartych portach. Połączenia te wykonywane są na dwa sposoby: aktywnie lub pasywnie. W aktywnym połączeniu klient wysyła adres i numer portu do serwera, klient następnie otwiera porty i dane są przesyłane. Problem w tym że iptables nie wie nic o dodatkowych połączeniach i odrzuca je. Wobec tego ważne jest zainstalowanie modułu `IP_NF_FTP`, a następnie dodać regułę by filter przepuszczał połączenia powiązane. Diagram przedstawia widok z perspektywy iptables:



W pasywnym połączeniu klient FTP mówi serwerowi że chce pobrać specyficzne dane więc serwer wysyła klientowi powiadomienie o adresie i portach które będą otwarte. W takiej sytuacji to klient otwiera port 20 i odbiera dane. Oto diagram:



7. Budowanie reguł

Przy budowaniu reguł należy pilnować się następującej składni:

```
iptables [-t tabela] komenda [porównanie (match)] -j [cel/skok (target/jump)]
```

Jeżeli nie podamy tabeli, reguła zostanie domyślnie dopisana do tabeli filter.

Podstawowe komendy to:

-A (--append) reguła – Dodanie regułki do łańcucha. Regułka zostanie dopisana na jego końcu

-D (--delete) łańcuch [reguła | nr reguły] – Usunięcie podanej reguły w podanym łańcuchu. Odnosimy się do łańcucha poprzez jego wpisanie bądź liczbę porządkową.

-R (--replace) łańcuch [reguła] NrReguły – Podmiana podanej reguły podanego łańcucha. J/w z tym że na miejsce starego łańcucha wrzucamy nowy łańcuch.

-I (--insert) łańcuch [reguła] [nr reguły] – Włożenie reguły do podanego łańcucha i jego wiersza. Bez podania numeru wiersza domyślnie reguła zostanie włożona do pierwszego wiersza.

-L (--list) – Wyświetlenie wszystkich reguł w podanej tabeli.

-F (--flush) łańcuch – Wyczyszczenie wpisów w podanym łańcuchu.

-Z (--zero) łańcuch – Wyzerowanie liczników liczących ilość przesłanych pakietów. Ilość przesłanych pakietów możemy zobaczyć wpisując: iptables -Lv łańcuch.

-N (--new-chain) łańcuch– Utworzenie nowego łańcucha użytkownika. To jest sposób by pewien blok reguł tyczących np. danego protokołu umieścić w łańcuchu zdefiniowanym przez użytkownika a następnie umieścić w łańcuchu PREROUTING, INPUT, OUTPUT, FORWARD lub POSTROUTING.

-X (--delete-chain) łańcuch– Usunięcie łańcucha zdefiniowanego przez użytkownika. Usunięcie jest możliwe jedynie wtedy gdy łańcuchy te nie mają żadnych reguł.

-P (--policy) łańcuch – Polityka domyślna łańcucha. Jeżeli pakiet nie pasuje do żadnego z podanych reguł to zostanie przepuszczony bądź odrzucony, zależnie od ustaleń polityki domyślnej.

Podstawowe porównania:

-p protokół – Określa protokół wg którego chcemy porównywać pakiety. Protokoły są trzy TCP, UDP, ICMP. Protokoły można łączyć, można też zastosować słowo kluczowe ALL. Można też zanegować stawiając wykrzyknik przed nazwą protokołu.

-s – określa adres IP źródła

-d – określa adres IP celu

-i – określa interfejs wejściowy, z którego pakiety będą przychodzić

-o – określa interfejs wyjściowy, z którego pakiety będą wychodzić

Porównania oparte o protokoły:

--sport – określa port źródłowy (UDP, TCP)

--dport – określa port docelowy (UDP, TCP)

--icmp-type – określa typ komunikatu icmp. Może być podany poprzez liczbę (patrz tabelka) lub poprzez sam komunikat (ICMP).

--tcp-flags – określa porównania na podstawie ustawień flag: SYN, ACK, RST, PSH, URG, FIN.

--syn – określa porównanie dla pakietów posiadający ustawioną flagę SYN. (Zaleca się stosowanie --tcp-flags, gdyż to zostało zachowane dla kompatybilności z ipchains).

Pozostałe porównania:

limit --limit ile/JednCzasu – ustalamy ilość pakietów przesłanych w określonym czasie jednostkami mogą być /second /minute /hour /day.

mac --mac-source – ustalamy porównanie na podstawie adresów MAC.

multiport --sports port,port,port,... - ustalamy porty źródłowe

multiport --dports port,port,port,... - ustalamy porty docelowe

multiport --ports port,port,port,... - ustalamy porty zarówno źródłowe jak i docelowe

state --state StanPołączenia – ustalamy porównanie poprzez stan połączenia. Stany mogą być następujące: NEW, ESTABLISHED, RELATED, INVALID. Połączenia typu INVALID są to pakiety które nie są powiązane z żadnym z protokołów. Dzieje się tak dlatego że, w pakiecie nastąpiły przekłamania nagłówek lub danych.

tos --tos 0xXX – ustalamy porównanie na podstawie typu usługi. Typy usługi zostały opisane wcześniej.

ttl -ttl XX – ustalamy porównanie oparte o długość życia pakietu.

Skok/Cel (Target/Jump)

Skok wykonujemy poprzez dodanie na końcu reguły -j skok/cel. Cele związane z tabelą **filter**:

ACCEPT – Akceptuje pakiety,

REJECT – Odrzuca pakiety informując o tym klienta komunikatem ICMP. Ew. można zresetować połączenie w protokole TCP wysyłając pakiet z flagą RST.

DROP – Odrzuca pakiety bez komunikatu ICMP,

MIRROR – W sumie użycie tego celu może być niebezpieczne z tego względu że grozi zapętleniem pakietu. Cel powoduje zamianę adresów źródłowych z docelowymi. Czego wynikiem może być sytuacja taka: Łączy się z nami klient pod SSH, ale my mamy MIRROR, więc klient się łączy ze swoim serwerem SSH.

LOG – Loguje pakiet w przestrzeni jądra za pomocą demona **syslogd()**.

ULOG – Loguje pakiet w przestrzeni użytkownika za pomocą demona **ulog** (posiada więcej opcji niż **syslogd()**).

Cele związane z tabelą **nat**:

DNAT – Translacja adresu docelowego na podany adres. Używane przy udostępnianiu usług w Internecie z hostów będących za NAT'em.

SNAT – Translacja adresu źródłowego na podany adres. Używane do udostępnianiu usług z Internetu.

MASQUERADE – Translacja adresu źródłowego. Jednak translacja dokonuje się bez ingerencji użytkownika. Używane przy serwerach DHCP, gdy nie wiemy jaki adres będzie przypisany użytkownikom.

REDIRECT – Wykorzystywane do przekierowywania portów. Używamy np. w sytuacji gdy mamy postawionego **Squida** i nie chcemy by połączenia **http** były wykonywane na porcie 80 ale na 8080. W ten sposób robimy transparentne PROXY. Klienci sieci nie muszą o **Squidzie** nic wiedzieć.

W tabeli **mangle** mamy następujące cele:

TOS – poprzez tos ustawiamy typ usługi

TTL – Ustalamy długość życia pakietów.

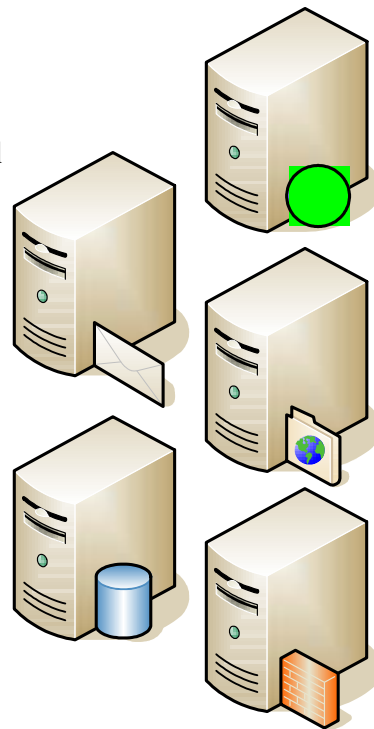
Dodatkowo możemy wykonywać skoki do zdefiniowanych łańcuchów. Z kolei skokiem RETURN robimy skok do Łącucha w którym jest wywołany łańcuch użytkownika, jeżeli robimy RETURN w łańcuchach zdefiniowanych programowo, zostaje wykonana domyślna polityka na pakiecie.

8. Popularne usługi oraz przykładowy skrypt rc.firewall.

Zanim zaczniemy pisać nasz skrypt to, musimy wiedzieć z jakich usług będziemy korzystać, a następnie z jakich one protokołów i portów korzystają. Rozpiszę najważniejsze oraz z jakich protokołów i portów one korzystają. Inne usługi (np.: p2p, komunikatory) mają możliwość zmiany portu więc ich naumyślnie tu nie umieszczę.

Popularne numery portów oraz skojarzone z nimi protokoły:

Port	Protokół	Aplikacja
20	TCP	FTP - Data
21	TCP	FTP - Control
22	TCP	SSH
23	TCP	Telnet
25	TCP	SMTP
53	TCP/UDP	DNS
80	TCP	HTTP
110	TCP	POP3
119	TCP	NNTP
443	TCP	SSL
1080	TCP	SOCKS
6667	TCP	IRC
8080	TCP	PROXY



Przy pisaniu skryptów dobrze skupić się na tym by skrypt był zawsze pisany przejrzysto, oraz był łatwy w dodawaniu poszczególnych reguł łańcucha.

Moją propozycją jest podzielenie skryptu na poszczególne sekcje w których zrobimy następujące czynności:

1. Konfiguracja – Pierwsze co napiszemy są to opcje konfiguracyjne, przez które przypisujemy zmienne adresom, interfejsom, lub programom do których będziemy się odwoływać w skrypcie.
 - 1.1. Internet – Tutaj przypisujemy zmienne dla adresów oraz interfejsów które mają połączenie z Internetem (bramy). Jeżeli nie mamy połączenia z internetem to nie musimy tu nic wprowadzać.
 - 1.1.1. DHCP – Jeżeli skrypt wymaga specjalnych wpisów dla DHCP (np. DHCP jest na innym komputerze niż brama), to dokonujemy tu specjalnych wpisów.
 - 1.1.2. PPPoE – Jeżeli używamy protokołu Point to Point Protocol on Ethernet, wtedy przypisujemy odpowiednie zmienne oraz adres dla tego interfejsu.
 - 1.2. LAN – Jeżeli mamy sieć lokalną to wprowadzamy odpowiednie zmienne tutaj.
 - 1.3. DMZ – Jeżeli mamy w naszej sieci strefę zdemilitaryzowaną to dopisujemy odpowiednie dane
 - 1.4. Localhost – Jeżeli mamy naszą maszynę... ;). Po prostu wprowadzamy zmienne dla jej samej i mało kiedy zachodzi potrzeba by je zmieniać.
 - 1.5. iptables – Tu umieszczamy gdzie znajduje się plik binarny iptables by w skrypcie móc się odnosić przez zmienną a nie ścieżkę dostępu. Mało kiedy jest to konieczne gdyż iptables znajduje się w katalogu z binarkami, więc nie ma problemu z odniesieniem się do niego.
 - 1.6. Pozostałe – Jeżeli chcemy odnieść się do innych plików w naszym skrypcie lub skrypt wymaga innych zmiennych które nie pasują do powyższych sekcji to wprowadzamy je tutaj.
2. Ładowanie modułów – W tej sekcji powinna być zawarta lista modułów do załadowania.
 - 2.1. Moduły wymagane – tutaj podajemy jakie skrypt powinien załadować moduły by nasz iptables mogło w ogóle działać.
 - 2.2. Moduły nie wymagane – tutaj wstawiamy moduły z których chcemy skorzystać przy stawianiu bramy bądź, chcemy się posłużyć specyficznymi regułami filtrowania.

3. Konfiguracja /proc – tu przygotowujemy jądro do współpracy z iptables np.: ustawienie forwardowania pakietów, dynamicznego przydzielania nazw, itp.
- 3.1. Wymagana konfiguracja /proc – tu ustawiamy jądro w taki sposób by nam skrypt działał
- 3.2. Nie wymagana konfiguracja /proc – tu możemy ustawić dodatkowe parametry jądra.
4. Ustawianie reguł – Teraz główna część programu. To tutaj ustalamy politykę bezpieczeństwa oraz ustawienia bramy.
- 4.1. Tabela Filter – Wpierw przejdziemy przez tabelę filter. To tu ustalimy domyślną politykę bezpieczeństwa oraz politykę dla poszczególnych łańcuchów.
 - 4.1.1. Ustawienie polityki domyślnej – Jak sama nazwa wskazuje... generalnie ustawia się politykę DROP a ACCEPT dla portów które są mile widziane.
 - 4.1.2. Deklaracje łańcuchów użytkownika – tu tworzymy łańcuchy (deklarujemy), które potem użyjemy w łańcuchach systemowych. Bez tego nie użyjemy naszych łańcuchów w iptables.
 - 4.1.3. Definicje łańcuchów użytkownika – tu dodajemy reguły (definicje) dla naszych łańcuchów.
 - 4.1.4. Łańcuch INPUT – Wrzucamy do łańcucha systemowego INPUT nasze reguły.
 - 4.1.5. Łańcuch OUTPUT – J/W, z tym że tym razem do łańcucha OUTPUT.
 - 4.1.6. Łańcuch FORWARD – J/W, ale dla łańcucha FORWARD.
- 4.2. Tabela NAT – Po przefiltrowaniu pakietów zabieramy się za translację adresów. To że ta tabela jest konfigurowana po tabeli filter ma swoją przyczynę. Po pierwsze nie chcemy by wszystkie pakiety z danego połączenia ulegały translacji adresowej, czy w wypadku nie przefiltrowywania pakietów, wszystkie zostały przepuszczone przez firewall. Pod drugie to dobrze by było by tabela filter była rdzeniem iptables, czyli jej pierwszą warstwą, a drugą była tabela NAT, trzecią natomiast MANGLE. W paru konfiguracjach mogą być wyjątki od tej zasady, ale generalnie jest to najbardziej zoptymalizowany układ.
 - 4.2.1. Ustawienie polityki domyślnej – W tej tabeli ustawiamy ACCEPT, gdyż nie używamy Tabeli NAT do filtrowania (co zresztą byłoby bardzo niezdrowe ze względu że polityka tej tabeli działa dla całych połączeń a nie dla jej poszczególnych pakietów). Zresztą nie ma innego powodu by to zmieniać.

- 4.2.2. Deklaracje łańcuchów użytkownika – deklarujemy użycie naszych łańcuchów.
- 4.2.3. Definicje łańcuchów użytkownika – definiujemy nasze łańcuchy by były gotowe do użycia w łańcuchach systemowych.
- 4.2.4. Łańcuch PREROUTING – ustalamy reguły dla pakietów zaczynających połączenia z usługami naszej sieci lokalnej (**DNAT**). W warunkach domowych nie używany). Tu ustawiamy usługi do Internetu z naszej sieci lokalnej.
- 4.2.5. Łańcuch POSTROUTING – ustalamy reguły dla pakietów zaczynających połączenia z usługami w Internecie. Głównie używamy tu **SNAT** dla stałych adresów oraz **MASQUERADE** dla hostów będących pod „opieką” serwera **DHCP**.
- 4.2.6. Łańcuch OUTPUT – Ten łańcuch jest bardzo rzadko stosowany, gdyż generalnie używa się firewalla do filtracji pakietów a nie łączenia się z usługami. Sam nie widzę sensu by łączyć się z usługami z firewalla za pomocą zewnętrznego adresu, jeżeli te hosty lokalne są dostępne od wewnątrz. Właściwie to w ogóle nie ma sensu łączyć się z jakimikolwiek usługami przez firewall. Jeżeli ktoś znajdzie powód, to można mnie powiadomić a ja to dodam do tej dokumentacji.
- 4.3. Tabela mangle – Ostatnia tabela używana w celu modyfikacji pakietów. Używana tylko dla specyficznych potrzeb, np.: jeżeli chcemy by nasz router nie był widzialny dla traceroute zwiększając TTL pakietu o jeden, lub zmieniając ToS pakietu. W większości przypadków zostawia się to puste z paroma małymi wyjątkami. Parę tych wyjątków pokarzę.
 - 4.3.1. Ustawienie polityki domyślnej – patrz tabela NAT.
 - 4.3.2. Deklaracja łańcuchów użytkownika – Już sam fakt rzadkiego używania tej tabeli nie daje potrzeby ustawiania łańcuchów użytkownika.
 - 4.3.3. Definicja łańcuchów użytkownika – Jeżeli już zadeklarowaliśmy to teraz definiujemy.
 - 4.3.4. Łańcuch PREROUTING – Ustawiamy reguły modyfikacji dla łańcucha PREROUTING. Szczerze nie znalazłem nic na temat różnic modyfikacji pakietu w różnych łańcuchach i jakie to ma znaczenie. Jeżeli ktoś będzie wiedział proszę o kontakt.
 - 4.3.5. Łańcuch INPUT – Ustawiamy reguły modyfikacji dla łańcucha INPUT.
 - 4.3.6. Łańcuch FORWARD - Ustawiamy reguły modyfikacji dla łańcucha FORWARD.

4.3.7. Łącuch OUTPUT - Ustawiamy reguły modyfikacji dla łańcucha OUTPUT.

4.3.8. Łącuch POSTROUTING - Ustawiamy reguły modyfikacji dla łańcucha POSTROUTING.

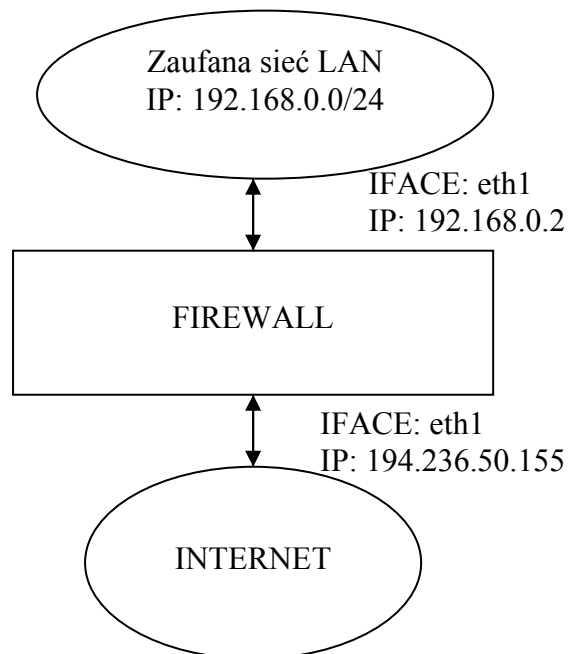
Uff... Doszliśmy do końca.

Następnie przedstawię przykładowe sytuacje oraz przykłady skryptów. Chcę zaznaczyć że te przykłady są bardzo ogólnikowe i nie radzę tego kopiować i wklejać ale jeszcze go trochę zmodyfikować do swoich potrzeb.

1. Skrypt rc.firewall – baza do budowy skryptów

W tym skrypcie zakładam że:

- adresy IP przydzielane są statycznie
- nie udostępniamy żadnych usług do Internetu
- LAN jest siecią zaufaną.



```

#!/bin/sh
#
# rc.firewall
#
#####
#
# 1. Konfiguracja.
#
#
# 1.1 Internet.
#
INET_IP="194.236.50.155"
INET_IFACE="eth0"
INET_BROADCAST="194.236.50.255"
#
# 1.1.1 DHCP
#
#
# 1.1.2 PPPoE
#
#
# 1.2 LAN
#
# Tu określamy naszego Localhosta oraz ilość hostów,
# które mogą być podpięte do LAN
#
LAN_IP="192.168.0.2"
LAN_IP_RANGE="192.168.0.0/16"
LAN_IFACE="eth1"
#
# 1.3 DMZ
#
#
# 1.4 Localhost
#
LO_IFACE="lo"
LO_IP="127.0.0.1"
#
# 1.5 IPTables
#
IPTABLES="/usr/sbin/iptables"
#
# 1.6 Pozostałe
#
#####
#
# 2. Ładowanie modułów.
#
#
# potrzebne by zainicjalizować ładowanie modułów.
#
/sbin/depmod -a
#
# 2.1 Moduły wymagane
#
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_state
#
# 2.2 Moduły nie wymagane - jeżeli któryś będzie potrzebny to odhashujemy
#
#/sbin/modprobe ipt_owner

```

```

#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ipt_MASQUERADE
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
#/sbin/modprobe ip_nat_ftp
#/sbin/modprobe ip_nat_irc
#####
#
# 3. Konfiguracja /proc
#
#
# 3.1 Wymagana konfiguracja /proc
#
echo "1" > /proc/sys/net/ipv4/ip_forward
#
# 3.2 Nie wymagana konfiguracja proc
#
#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr
#####
#
# 4. Ustawianie reguł.
#
#####
# 4.1 Tabela filter
#
#
# 4.1.1 Polityka domyślna
#
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
#
# 4.1.2 Deklaracja łańcuchów użytkownika
#
#
# Tworzymy łańcuch na błędne pakiety TCP
#
$IPTABLES -N bledne_pakiety_TCP
#
# Tworzymy oddzielne łańcuchy dla ICMP, TCP and UDP dla dalszego trawersowania
#
$IPTABLES -N dozwolone
$IPTABLES -N tcp_pakiety
$IPTABLES -N udp_pakiety
$IPTABLES -N icmp_pakiety
#
# 4.1.3 Definicja łańcuchów użytkownika.
#
#
# łańcuch błędne pakiety TCP
#
$IPTABLES -A bledne_pakiety_TCP -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
$IPTABLES -A bledne_pakiety_TCP -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "Nowy bez flagi syn:"
$IPTABLES -A bledne_pakiety_TCP -p tcp ! --syn -m state --state NEW -j DROP
#
# łańcuch dozwolone
#
$IPTABLES -A dozwolone -p TCP --syn -j ACCEPT
$IPTABLES -A dozwolone -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A dozwolone -p TCP -j DROP
#
# Porty TCP
#
$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 21 -j dozwolone

```

```

$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 22 -j dozwolone
$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 80 -j dozwolone
$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 113 -j dozwolone
#
# Porty UDP
#
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 53 -j ACCEPT
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 123 -j ACCEPT
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 2074 -j ACCEPT
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 4000 -j ACCEPT
#
# W sieciach Microsoftu aż roi się od broadcastów. Te linie zapobiegają
# pojawieniu się ich w logach
#
#$IPTABLES -A udp_pakiety -p UDP -i $INET_IFACE -d $INET_BROADCAST \
--destination-port 135:139 -j DROP
#
# Jeżeli nasz host będzie dostawał prośby DHCP to także
# Logi zostaną tym zawałone. Poniższa reguła blokuje to zjawisko.
#
#$IPTABLES -A udp_pakiety -p UDP -i $INET_IFACE -d 255.255.255.255 \
--destination-port 67:68 -j DROP
#
# Reguły ICMP
#
$IPTABLES -A icmp_pakiety -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_pakiety -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT
#
# 4.1.4 Łańcuch INPUT
#
#
# Pakiety tcp których nie chcemy.
#
$IPTABLES -A INPUT -p tcp -j bledne_pakiety_tcp
#
# Reguły dla sieci nie będącej w internecie
#
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -s $LAN_IP_RANGE -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT
#
# Reguła dla żądań DHCP z LAN, w innym przypadku nasz serwer DHCP nie będzie
# działał właściwie
#
$IPTABLES -A INPUT -p UDP -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT
#
# Reguły dla pakietów przychodzących z internetu.
#
$IPTABLES -A INPUT -p ALL -d $INET_IP -m state --state ESTABLISHED,RELATED \
-j ACCEPT
$IPTABLES -A INPUT -p TCP -i $INET_IFACE -j tcp_pakiety
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -j udp_pakiety
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_pakiety
#
# Jeżeli mamy sieć Microsoftu na zewnątrz naszego firewalla, możemy
# mieć logi zatopione przez jego multicasty. Jeżeli chcemy - odrzucamy te pakiety.
#
#
#$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP
#
# Logujemy dziwaczne pakiety nie pasujące do żadnych powyższych.
#
$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT pakiet poległ: "
#
# 4.1.5 Łańcuch FORWARD
#

```

```
#
# Błędne pakiety TCP których nie chcemy.
#
$IPTABLES -A FORWARD -p tcp -j bledne_pakiety_tcp
#
# Akceptujemy pakiety, które chcemy forwardować.
#
$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
#
# Logujemy dziwaczne pakiety, które nie pasują do reszty reguł.
#
$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet polegl: "
#
# 4.1.6 Łącuch OUTPUT
#
#
# Błędne pakiety TCP których nie chcemy.
#
$IPTABLES -A OUTPUT -p tcp -j bledne_pakiety_tcp
#
# Specjalne reguły OUTPUT decydujące, którym adresom IP zezwalamy na połączenie.
#
$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT
#
# Dziwaczne pakiety logujemy
#
$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet polegl: "
#####
# 4.2 tabela NAT
#
#
# 4.2.1 Ustawienie polityki domyślnej
#
#
# 4.2.2 Deklaracja łańcuchów użytkownika
#
#
# 4.2.3 Definicja łańcuchów użytkownika
#
#
# 4.2.4 Łącuch PREROUTING
#
#
# 4.2.5 Łącuch POSTROUTING
#
#
# Zezwalamy na forwardowanie pakietów do naszej sieci LAN
#
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP
#
# 4.2.6 Łącuch OUTPUT
#
#####
# 4.3 Tabela mangle
#
#
# 4.3.1 Ustawienie polityki domyślnej
#
#
# 4.3.2 Deklaracja łańcuchów użytkownika
#
#
# 4.3.3 Definicja łańcuchów użytkownika
```

```

#
#
# 4.3.4 Łańcuch PREROUTING
#
#
# 4.3.5 Łańcuch INPUT
#
#
# 4.3.6 Łańcuch FORWARD
#
#
# 4.3.7 Łańcuch OUTPUT
#
#
# 4.3.8 Łańcuch POSTROUTING
#

```

2. Skrypt rc.DMZ.firewall – dorzucamy strefę zdemilitaryzowaną.

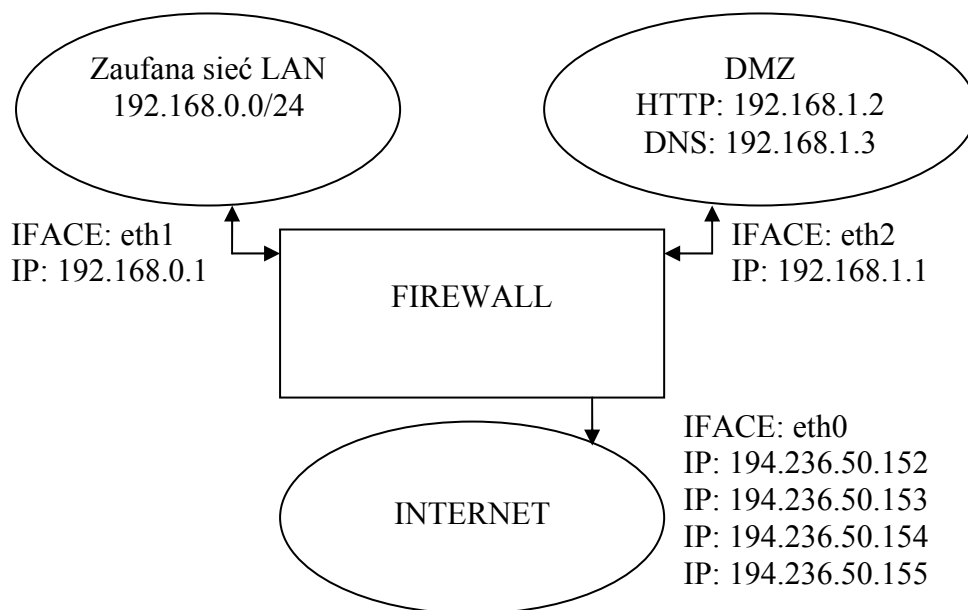
Skrypt jest podobny do powyższego z tym, że:

- Mamy strefę zdemilitaryzowaną a w niej usługi WWW i DNS.

Strefę zdemilitaryzowaną można zrobić na dwa sposoby:

- Dodajemy go do naszej sieci LAN tracąc przy tym dwa numery IP z puli adresów.
- Tworzymy kolejną podsieć, w której będą znajdować się nasze usługi.

Skrypt będzie zrobiony na sposób drugi, jak zresztą przedstawia ten diagram:



```

#!/bin/sh
#
# rc.firewall
#
#####
#
# 1. Konfiguracja.
#
#
# 1.1 Internet.
#
INET_IP="194.236.50.152"
HTTP_IP="194.236.50.153"
DNS_IP="194.236.50.154"
INET_IFACE="eth0"
INET_BROADCAST="194.236.50.255"
#
# 1.1.1 DHCP
#
#
# 1.1.2 PPPoE
#
#
# 1.2 LAN
#
# Tu określamy naszego Localhosta oraz ilość hostów,
# które mogą być podpięte do LAN
#
LAN_IP="192.168.0.2"
LAN_IP_RANGE="192.168.0.0/16"
LAN_IFACE="eth1"
#
# 1.3 DMZ
#
DMZ_HTTP_IP="192.168.1.2"
DMZ_DNS_IP="192.168.1.3"
DMZ_IP="192.168.1.1"
DMZ_IFACE="eth2"
#
# 1.4 Localhost
#
LO_IFACE="lo"
LO_IP="127.0.0.1"
#
# 1.5 IPTables
#
IPTABLES="/usr/sbin/iptables"
#
# 1.6 Pozostałe
#
#####
#
# 2. Ładowanie modułów.
#
#
# potrzebne by zainicjalizować ładowanie modułów.
#
/sbin/depmod -a
#
# 2.1 Moduły wymagane
#
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit

```

```

/sbin/modprobe ipt_state
#
# 2.2 Moduły nie wymagane - jeżeli któryś będzie potrzebny to odhashowujemy
#
/sbin/modprobe ipt_owner
/sbin/modprobe ipt_REJECT
/sbin/modprobe ipt_MASQUERADE
/sbin/modprobe ip_conntrack_ftp
/sbin/modprobe ip_conntrack_irc
/sbin/modprobe ip_nat_ftp
/sbin/modprobe ip_nat_irc
#####
#
# 3. Konfiguracja /proc
#
#
# 3.1 Wymagana konfiguracja /proc
#
echo "1" > /proc/sys/net/ipv4/ip_forward
#
# 3.2 Nie wymagana konfiguracja proc
#
#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr
#####
#
# 4. Ustawianie reguł.
#
#####
# 4.1 Tabela filter
#
#
# 4.1.1 Polityka domyślna
#
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
#
# 4.1.2 Deklaracja łańcuchów użytkownika
#
#
# Tworzymy łańcuch na błędne pakiety TCP
#
$IPTABLES -N bledne_pakiety_TCP
#
# Tworzymy oddzielne łańcuchy dla ICMP, TCP and UDP dla dalszego trawersowania
#
$IPTABLES -N dozwolone
$IPTABLES -N icmp_pakiety
#
# 4.1.3 Definicja łańcuchów użytkownika.
#
#
# łańcuch błędne pakiety TCP
#
$IPTABLES -A bledne_pakiety_TCP -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
$IPTABLES -A bledne_pakiety_TCP -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "Nowy bez flagi syn:"
$IPTABLES -A bledne_pakiety_TCP -p tcp ! --syn -m state --state NEW -j DROP
#
# łańcuch dozwolone
#
$IPTABLES -A dozwolone -p TCP --syn -j ACCEPT
$IPTABLES -A dozwolone -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A dozwolone -p TCP -j DROP

```

```

#
# Reguły ICMP
#
$IPTABLES -A icmp_pakiety -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_pakiety -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT
#
# 4.1.4 Łańcuch INPUT
#
#
# Pakiety tcp których nie chcemy.
#
$IPTABLES -A INPUT -p tcp -j bledne_pakiety_tcp
#
# Pakiety z Internetu do naszej maszyny
#
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_pakiety
#
# Pakiety z LAN, DMZ lub Localhosta
#
#
# Od DMZ Interfejsu do DMZ firewall IP
#
$IPTABLES -A INPUT -p ALL -i $DMZ_IFACE -d $DMZ_IP -j ACCEPT
#
# Od LAN Interfejsu do LAN firewall IP
#
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_IP -j ACCEPT
#
# Od interfejsu localhosta do interfejsu IP.
#
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT
#
# Reguła dla żądań DHCP z LAN, w innym przypadku nasz serwer DHCP nie będzie
# działał właściwie
#
$IPTABLES -A INPUT -p UDP -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT
#
# Reguły dla połączeń ustanowionych i powiązanych z Internetu do firewalla.
#
$IPTABLES -A INPUT -p ALL -d $INET_IP -m state --state ESTABLISHED,RELATED \
-j ACCEPT
#
# Jeżeli mamy sieć Microsoftu na zewnątrz naszego firewalla, możemy
# mieć logi zatopione przez jego multicasty. Jeżeli chcemy - odrzucamy te pakiety.
#
#$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP
#
# Jeżeli nasz host będzie dostawał prośby DHCP to także
# Logi zostaną tym zawalone. Poniższa reguła blokuje to zjawisko.
#
#$IPTABLES -A udp_pakiety -p UDP -i $INET_IFACE -d 255.255.255.255 \
--destination-port 67:68 -j DROP
#
# Logujemy dziwaczne pakiety nie pasujące do żadnych powyższych.
#
$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT pakiet poległ: "
#
# 4.1.5 Łańcuch FORWARD
#
#
# Błędne pakiety TCP których nie chcemy.
#
$IPTABLES -A FORWARD -p tcp -j bledne_pakiety_tcp

```

```

#
# Sekcja DMZ
#
# Reguły generalne
#
$IPTABLES -A FORWARD -i $DMZ_IFACE -o $INET_IFACE -j ACCEPT
$IPTABLES -A FORWARD -i $INET_IFACE -o $DMZ_IFACE -m state \
--state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $LAN_IFACE -o $DMZ_IFACE -j ACCEPT
$IPTABLES -A FORWARD -i $DMZ_IFACE -o $LAN_IFACE -m state \
--state ESTABLISHED,RELATED -j ACCEPT
#
# Serwer HTTP
#
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_HTTP_IP \
--dport 80 -j dozwolone
$IPTABLES -A FORWARD -p ICMP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_HTTP_IP \
-j icmp_pakiety
#
# Serwer DNS
#
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP \
--dport 53 -j dozwolone
$IPTABLES -A FORWARD -p UDP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP \
--dport 53 -j ACCEPT
$IPTABLES -A FORWARD -p ICMP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP \
-j icmp_pakiety
#
# Sekcja LAN
#
$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
#
# Logujemy dziwaczne pakiety, które nie pasują do reszty reguł.
#
$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet polegl: "
#
# 4.1.6 Łańcuch OUTPUT
#
# Błędne pakiety TCP których nie chcemy.
#
$IPTABLES -A OUTPUT -p tcp -j bledne_pakiety_tcp
#
# Specjalne reguły OUTPUT decydujące, którym adresom IP zezwalamy na połączenie.
#
$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT
#
# Dziwaczne pakiety logujemy
#
$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet polegl: "
#####
# 4.2 tabela NAT
#
#
# 4.2.1 Ustawienie polityki domyślnej
#
#
# 4.2.2 Deklaracja łańcuchów użytkownika
#
#
# 4.2.3 Definicja łańcuchów użytkownika
#
#
#

```

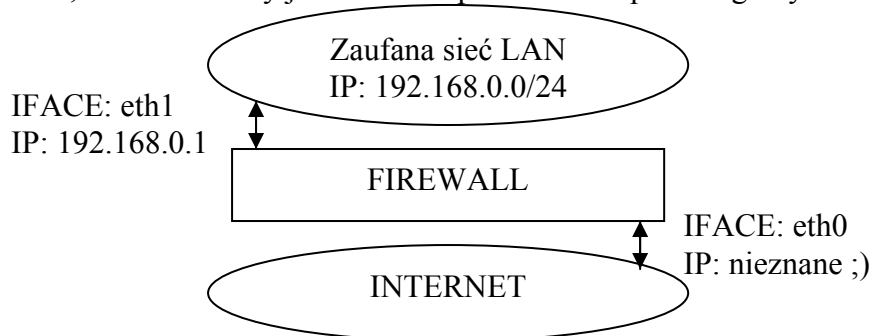
```

# 4.2.4 Łańcuch PREROUTING
#
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $HTTP_IP --dport 80 \
-j DNAT --to-destination $DMZ_HTTP_IP
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $DNS_IP --dport 53 \
-j DNAT --to-destination $DMZ_DNS_IP
$IPTABLES -t nat -A PREROUTING -p UDP -i $INET_IFACE -d $DNS_IP --dport 53 \
-j DNAT --to-destination $DMZ_DNS_IP
#
# 4.2.5 Łańcuch POSTROUTING
#
#
# Zezwalamy na forwardowanie pakietów do naszej sieci LAN
#
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP
#
# 4.2.6 Łańcuch OUTPUT
#
#####
# 4.3 Tabela mangle
#
#
# 4.3.1 Ustawienie polityki domyślnej
#
#
# 4.3.2 Deklaracja łańcuchów użytkownika
#
#
# 4.3.3 Definicja łańcuchów użytkownika
#
#
# 4.3.4 Łańcuch PREROUTING
#
#
# 4.3.5 Łańcuch INPUT
#
#
# 4.3.6 Łańcuch FORWARD
#
#
# 4.3.7 Łańcuch OUTPUT
#
#
# 4.3.8 Łańcuch POSTROUTING
#

```

3. Skrypt rc.DHCP.firewall – Mamy serwer DHCP

Ten przykład różni się od pierwszego tym że mamy przydzielane adresy IP dynamicznie. Przedstawi to poniższy diagram. W skrypcie zamieniłem `-d $STATIC_IP` na `-i $INET_IFACE`. Inna sprawa to to, że nie możemy już filtrować pakietów dla poszczególnych hostów w LAN.



```

#!/bin/sh
#
# rc.DHCP.firewall
#
#####
#
# 1. Konfiguracja.
#
#
# 1.1 Internet.
#
INET_IFACE="eth0"
#
# 1.1.1 DHCP
#
# Ustawiamy zmienną dla DHCP, jeżeli nie otrzymujemy od niego IP. Jeżeli adres
# Jest przydzielany przez DHCP z Internetu to ustawiamy zmienną na yes i ustwiamy
# właściwy adres IP.
#
DHCP="no"
DHCP_SERVER="195.22.90.65"
#
# 1.1.2 PPPoE
#
# Jeżeli mamy problemy z PPPoE, np. nie dochodzą nam większe e-maile, a mniejsze
# przechodzą to ustawiamy tą zmienną na yes. Powoduje ona zmniejszenie rozmiaru
# przesyłanych pakietów. Zmienną wykorzystujemy w tabeli mangle w łańcuchu
# prerouting.
#
PPPOE_PMTU="no"
#
# 1.2 LAN
#
# Tu określamy naszego Localhosta oraz ilość hostów,
# które mogą być podpięte do LAN.
#
LAN_IP="192.168.0.2"
LAN_IP_RANGE="192.168.0.0/16"
LAN_IFACE="eth1"
#
# 1.3 DMZ
#
#
# 1.4 Localhost
#
LO_IFACE="lo"
LO_IP="127.0.0.1"
#
# 1.5 IPTables
#
IPTABLES="/usr/sbin/iptables"
#
# 1.6 Pozostałe
#
#####
#
# 2. Ładowanie modułów.
#
#
# potrzebne by zainicjalizować ładowanie modułów.
#
/sbin/depmod -a
#
# 2.1 Moduły wymagane
#
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter

```

```

/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_state
/sbin/modprobe ipt_MASQUERADE
#
# 2.2 Moduły nie wymagane - jeżeli któryś będzie potrzebny to odhashowujemy
#
#/sbin/modprobe ipt_owner
#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
#/sbin/modprobe ip_nat_ftp
#/sbin/modprobe ip_nat_irc
#####
#
# 3. Konfiguracja /proc
#
#
# 3.1 Wymagana konfiguracja /proc
#
echo "1" > /proc/sys/net/ipv4/ip_forward
#
# 3.2 Nie wymagana konfiguracja proc
#
#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr
#####
#
# 4. Ustawianie reguł.
#
#####
# 4.1 Tabela filter
#
#
# 4.1.1 Polityka domyślna
#
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
#
# 4.1.2 Deklaracja łańcuchów użytkownika
#
#
# Tworzymy łańcuch na błędne pakiety TCP
#
$IPTABLES -N bledne_pakiety_TCP
#
# Tworzymy oddzielne łańcuchy dla ICMP, TCP and UDP dla dalszego trawersowania
#
$IPTABLES -N dozwolone
$IPTABLES -N tcp_pakiety
$IPTABLES -N udp_pakiety
$IPTABLES -N icmp_pakiety
#
# 4.1.3 Definicja łańcuchów użytkownika.
#
#
# łańcuch błędne pakiety TCP
#
$IPTABLES -A bledne_pakiety_TCP -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
$IPTABLES -A bledne_pakiety_TCP -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "Nowy bez flagi syn:"
$IPTABLES -A bledne_pakiety_TCP -p tcp ! --syn -m state --state NEW -j DROP
#

```

```

# łańcuch dozwolone
#
$IPTABLES -A dozwolone -p TCP --syn -j ACCEPT
$IPTABLES -A dozwolone -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A dozwolone -p TCP -j DROP
#
# Porty TCP
#
$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 21 -j dozwolone
$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 22 -j dozwolone
$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 80 -j dozwolone
$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 113 -j dozwolone
#
# Porty UDP
#
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --source-port 53 -j ACCEPT
if [ $DHCP == "yes" ] ; then
$IPTABLES -A udp_pakiety -p UDP -s $DHCP_SERVER --sport 67 \
--dport 68 -j ACCEPT
fi
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 53 -j ACCEPT
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 123 -j ACCEPT
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 2074 -j ACCEPT
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 4000 -j ACCEPT
#
# W sieciach Microsoftu aż roi się od broadcastów. Te linie zapobiegają
# pojawieniu się ich w logach
#
#$IPTABLES -A udp_pakiety -p UDP -i $INET_IFACE -d $INET_BROADCAST \
--destination-port 135:139 -j DROP
#
# Jeżeli nasz host będzie dostawał prośby DHCP to także
# logi zostaną tym zawałone. Poniższa reguła blokuje to zjawisko.
#
#$IPTABLES -A udp_pakiety -p UDP -i $INET_IFACE -d 255.255.255.255 \
--destination-port 67:68 -j DROP
#
# Reguły ICMP
#
$IPTABLES -A icmp_pakiety -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_pakiety -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT
#
# 4.1.4 Łańcuch INPUT
#
#
# Pakiety tcp których nie chcemy.
#
$IPTABLES -A INPUT -p tcp -j bledne_pakiety_tcp
#
# Reguły dla sieci nie będącej w internecie
#
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -s $LAN_IP_RANGE -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -j ACCEPT
#
# Reguła dla żądań DHCP z LAN, w innym przypadku nasz serwer DHCP nie będzie
# działał właściwie
#
$IPTABLES -A INPUT -p UDP -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT
#
# Reguły dla pakietów przychodzących z internetu.
#
$IPTABLES -A INPUT -p ALL -d $INET_IFACE -m state --state ESTABLISHED,RELATED \
-j ACCEPT
$IPTABLES -A INPUT -p TCP -i $INET_IFACE -j tcp_pakiety
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -j udp_pakiety
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_pakiety
#
# Jeżeli mamy sieć Microsoftu na zewnątrz naszego firewalla, możemy

```

```

# mieć logi zatopione przez jego multicasty. Jeżeli chcemy - odrzucamy te pakiety.
#
#
#IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP
#
# Logujemy dziwaczne pakiety nie pasujące do żadnych powyższych.
#
$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT pakiet poległ: "
#
# 4.1.5 Łańcuch FORWARD
#
# Błędne pakiety TCP których nie chcemy.
#
$IPTABLES -A FORWARD -p tcp -j bledne_pakiety_tcp
#
# Akceptujemy pakiety, które chcemy forwardować.
#
$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
#
# Logujemy dziwaczne pakiety, które nie pasują do reszty reguł.
#
$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet poległ: "
#
# 4.1.6 Łańcuch OUTPUT
#
# Błędne pakiety TCP których nie chcemy.
#
$IPTABLES -A OUTPUT -p tcp -j bledne_pakiety_tcp
#
# Specjalne reguły OUTPUT decydujące, którym adresom IP zezwalamy na połączenie.
#
$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IFACE -j ACCEPT
#
# Dziwaczne pakiety logujemy
#
$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet poległ: "
#####
# 4.2 tabela NAT
#
#
# 4.2.1 Ustawienie polityki domyślnej
#
#
# 4.2.2 Deklaracja łańcuchów użytkownika
#
#
# 4.2.3 Definicja łańcuchów użytkownika
#
#
# 4.2.4 Łańcuch PREROUTING
#
#
# 4.2.5 Łańcuch POSTROUTING
#
#
# Zezwalamy na forwardowanie pakietów do naszej sieci LAN
#
if [ $PPPOE_PMTU == "yes" ] ; then
$IPTABLES -t nat -A POSTROUTING -p tcp --tcp-flags SYN,RST SYN \
-j TCPMSS --clamp-mss-to-pmtu

```

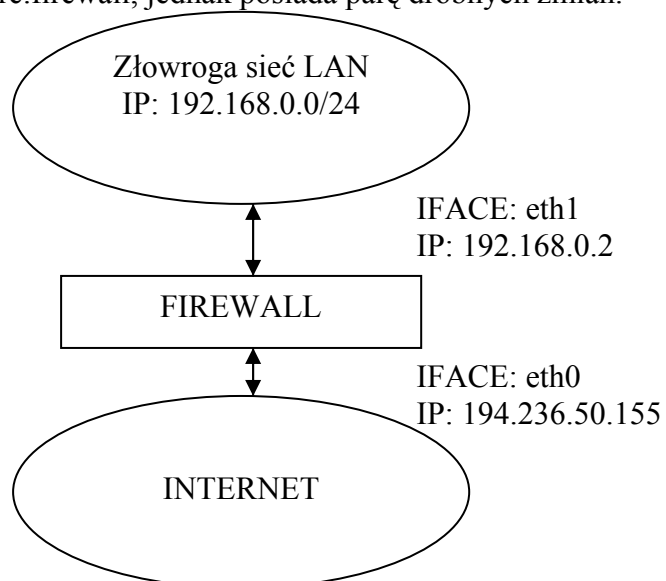
```

fi
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j MASQUERADE
#
# 4.2.6 Łańcuch OUTPUT
#
#####
# 4.3 Tabela mangle
#
#
# 4.3.1 Ustawienie polityki domyślnej
#
#
# 4.3.2 Deklaracja łańcuchów użytkownika
#
#
# 4.3.3 Definicja łańcuchów użytkownika
#
#
# 4.3.4 Łańcuch PREROUTING
#
#
# 4.3.5 Łańcuch INPUT
#
#
# 4.3.6 Łańcuch FORWARD
#
#
# 4.3.7 Łańcuch OUTPUT
#
#
# 4.3.8 Łańcuch POSTROUTING
#

```

4. Skrypt rc.UITN.firewall – sieć lokalna której nie ufamy.

W tej sytuacji użytkowników sieci LAN obdarowujemy takim samym zaufaniem jak użytkownikom Internetu. Jedynie otwarte będą usługi WWW, FTP oraz POP3. Skrypt pokazuje złotą zasadę by nie ufać nikomu, nawet pracownikom naszej firmy. Jest to smutny fakt, ale coraz więcej jest włamań do systemu z sieci wewnętrznej. Skrypt nie różni się za bardzo od skryptu rc.firewall, jednak posiada parę drobnych zmian.



```

#!/bin/sh
#
# rc.firewall
#
#####
#
# 1. Konfiguracja.
#
#
# 1.1 Internet.
#
INET_IP="194.236.50.155"
INET_IFACE="eth0"
INET_BROADCAST="194.236.50.255"
#
# 1.1.1 DHCP
#
#
# 1.1.2 PPPoE
#
#
# 1.2 LAN
#
# Tu określamy naszego Localhosta oraz ilość hostów,
# które mogą być podpięte do LAN
#
LAN_IP="192.168.0.2"
LAN_IP_RANGE="192.168.0.0/16"
LAN_IFACE="eth1"
#
# 1.3 DMZ
#
#
# 1.4 Localhost
#
LO_IFACE="lo"
LO_IP="127.0.0.1"
#
# 1.5 IPTables
#
IPTABLES="/usr/sbin/iptables"
#
# 1.6 Pozostałe
#
#####
#
# 2. Ładowanie modułów.
#
#
# potrzebne by zainicjalizować ładowanie modułów.
#
/sbin/depmod -a
#
# 2.1 Moduły wymagane
#
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_state
#
# 2.2 Moduły nie wymagane - jeżeli któryś będzie potrzebny to odhashowujemy
#
#/sbin/modprobe ipt_owner

```

```

#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ipt_MASQUERADE
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
#/sbin/modprobe ip_nat_ftp
#/sbin/modprobe ip_nat_irc
#####
#
# 3. Konfiguracja /proc
#
#
# 3.1 Wymagana konfiguracja /proc
#
echo "1" > /proc/sys/net/ipv4/ip_forward
#
# 3.2 Nie wymagana konfiguracja proc
#
#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr
#####
#
# 4. Ustawianie reguł.
#
#####
# 4.1 Tabela filter
#
#
# 4.1.1 Polityka domyślna
#
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
#
# 4.1.2 Deklaracja łańcuchów użytkownika
#
#
# Tworzymy łańcuch na błędne pakiety TCP
#
$IPTABLES -N bledne_pakiety_TCP
#
# Tworzymy oddzielne łańcuchy dla ICMP, TCP and UDP dla dalszego trawersowania
#
$IPTABLES -N dozwolone
$IPTABLES -N tcp_pakiety
$IPTABLES -N udp_pakiety
$IPTABLES -N icmp_pakiety
#
# 4.1.3 Definicja łańcuchów użytkownika.
#
#
# łańcuch błędne pakiety TCP
#
$IPTABLES -A bledne_pakiety_TCP -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
$IPTABLES -A bledne_pakiety_TCP -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "Nowy bez flagi syn:"
$IPTABLES -A bledne_pakiety_TCP -p tcp ! --syn -m state --state NEW -j DROP
#
# łańcuch dozwolone
#
$IPTABLES -A dozwolone -p TCP --syn -j ACCEPT
$IPTABLES -A dozwolone -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A dozwolone -p TCP -j DROP
#
# Porty TCP
#
$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 21 -j dozwolone

```

```

$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 22 -j dozwolone
$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 80 -j dozwolone
$IPTABLES -A tcp_pakiety -p TCP -s 0/0 --dport 113 -j dozwolone
#
# Porty UDP
#
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 53 -j ACCEPT
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 123 -j ACCEPT
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 2074 -j ACCEPT
$IPTABLES -A udp_pakiety -p UDP -s 0/0 --destination-port 4000 -j ACCEPT
#
# W sieciach Microsoftu aż roi się od broadcastów. Te linie zapobiegają
# pojawieniu się ich w logach
#
#$IPTABLES -A udp_pakiety -p UDP -i $INET_IFACE -d $INET_BROADCAST \
--destination-port 135:139 -j DROP
#
# Jeżeli nasz host będzie dostawał prośby DHCP to także
# logi zostaną tym zawałone. Poniższa reguła blokuje to zjawisko.
#
#$IPTABLES -A udp_pakiety -p UDP -i $INET_IFACE -d 255.255.255.255 \
--destination-port 67:68 -j DROP
#
# Reguły ICMP
#
$IPTABLES -A icmp_pakiety -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_pakiety -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT
#
# 4.1.4 Łańcuch INPUT
#
#
# Pakiety tcp których nie chcemy.
#
$IPTABLES -A INPUT -p tcp -j bledne_pakiety_tcp
#
# Reguły dla sieci nie będącej w internecie
#
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT
#
# Reguła dla żądań DHCP z LAN, w innym przypadku nasz serwer DHCP nie będzie
# działał właściwie
#
$IPTABLES -A INPUT -p UDP -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT
#
# Reguły dla pakietów przychodzących z internetu.
#
$IPTABLES -A INPUT -p ALL -d $INET_IP -m state --state ESTABLISHED,RELATED \
-j ACCEPT
$IPTABLES -A INPUT -p TCP -i $INET_IFACE -j tcp_pakiety
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -j udp_pakiety
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_pakiety
#
# Jeżeli mamy sieć Microsoftu na zewnątrz naszego firewalla, możemy
# mieć logi zatopione przez jego multicasty. Jeżeli chcemy - odrzucamy te pakiety.
#
#
#$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP
#
# Logujemy dziwaczne pakiety nie pasujące do żadnych powyższych.
#
$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT pakiet poległ: "
#
# 4.1.5 Łańcuch FORWARD
#
#

```

```

# Błędne pakiety TCP których nie chcemy.
#
$IPTABLES -A FORWARD -p tcp -j bledne_pakiety_tcp
#
# Akceptujemy pakiety, które chcemy forwardować.
#
$IPTABLES -A FORWARD -p tcp --dport 21 -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 80 -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 110 -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
#
# Logujemy dziwaczne pakiety, które nie pasują do reszty reguł.
#
$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet polegl: "
#
# 4.1.6 Łącuch OUTPUT
#
# Błędne pakiety TCP których nie chcemy.
#
$IPTABLES -A OUTPUT -p tcp -j bledne_pakiety_tcp
#
# Specjalne reguły OUTPUT decydujące, którym adresom IP zezwalamy na połączenie.
#
$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT
#
# Dziwaczne pakiety logujemy
#
$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet polegl: "
#####
# 4.2 tabela NAT
#
#
# 4.2.1 Ustawienie polityki domyślnej
#
#
# 4.2.2 Deklaracja łańcuchów użytkownika
#
#
# 4.2.3 Definicja łańcuchów użytkownika
#
#
# 4.2.4 Łącuch PREROUTING
#
#
# 4.2.5 Łącuch POSTROUTING
#
#
# Zezwalamy na forwardowanie pakietów do naszej sieci LAN
#
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP
#
# 4.2.6 Łącuch OUTPUT
#
#####
# 4.3 Tabela mangle
#
#
# 4.3.1 Ustawienie polityki domyślnej
#
#
# 4.3.2 Deklaracja łańcuchów użytkownika
#
#

```

```
# 4.3.3 Definicja łańcuchów użytkownika
#
#
# 4.3.4 Łańcuch PREROUTING
#
#
# 4.3.5 Łańcuch INPUT
#
#
# 4.3.6 Łańcuch FORWARD
#
#
# 4.3.7 Łańcuch OUTPUT
#
#
# 4.3.8 Łańcuch POSTROUTING
#
```

5. Skrypt `rc.test-iptables` – tak naprawdę dany jest tak sobie dla jajek. Co prawda wymaga jeszcze podrasowania (Translacja adresów, ustawienie forwardowania w `/proc`), ale daje na wgląd na to jak pakiety są wysyłane do poszczególnych łańcuchów (które, do jakiego). Po odpaleniu tego skryptu możemy zrobić:

```
Debian@root:# ping -c 1 host.w.internecie
```

a następnie odpalamy

```
Debian@root:# tail -n 0 -f /var/log/messages
```

To powinno nam pokazać jak są używane łańcuchy i w jakim porządku to się odbywa.

UWAGA!!! Skrypt napisany tylko w celach testowych. Innymi słowami nie używajmy jego z tego względu, że jeżeli wypełni nasze partycje na których trzymamy logi, staniemy się łatwym celem na ataki DoS.

```
#!/bin/bash
#
# rc.test-iptables - skrypt testujący wszystkie połączenia - loguje dosłownie
#wszystko więc, jeżeli używać to w celach naukowych.
#
# Tabela filter, wszystkie łańcuchy
#
iptables -t filter -A INPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter INPUT:"
iptables -t filter -A INPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter INPUT:"
iptables -t filter -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter OUTPUT:"
iptables -t filter -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter OUTPUT:"
iptables -t filter -A FORWARD -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter FORWARD:"
iptables -t filter -A FORWARD -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter FORWARD:"
#
# Tabela NAT, wszystkie łańcuchy za wyjątkiem OUTPUT który i tak nie będzie
```

```

# działał.
#
iptables -t nat -A PREROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat PREROUTING:"
iptables -t nat -A PREROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat PREROUTING:"
iptables -t nat -A POSTROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat POSTROUTING:"
iptables -t nat -A POSTROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat POSTROUTING:"
iptables -t nat -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat OUTPUT:"
iptables -t nat -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat OUTPUT:"
#
# Tabela mangle, wszystkie łańcuchy
#
iptables -t mangle -A PREROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle PREROUTING:"
iptables -t mangle -A PREROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle PREROUTING:"
iptables -t mangle -I FORWARD 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle FORWARD:"
iptables -t mangle -I FORWARD 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle FORWARD:"
iptables -t mangle -I INPUT 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle INPUT:"
iptables -t mangle -I INPUT 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle INPUT:"
iptables -t mangle -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle OUTPUT:"
iptables -t mangle -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle OUTPUT:"
iptables -t mangle -I POSTROUTING 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle POSTROUTING:"
iptables -t mangle -I POSTROUTING 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle POSTROUTING:"

```

6. Skrypt rc.flush-iptables – tak naprawdę nie sędzę, żeby można było to nazwać skrytem, dlatego że resetuje ustawienia iptables.

```

#!/bin/sh
#
# rc.flush-iptables - Skrypt resetujący ustawienia iptables.
#
#
# Konfiguracja
#
IPTABLES="/usr/sbin/iptables"
#
# reset domyślnej polityki w tabeli filter.
#
$IPTABLES -F INPUT ACCEPT
$IPTABLES -F FORWARD ACCEPT
$IPTABLES -F OUTPUT ACCEPT
#
# reset domyślnej polityki w tabeli NAT.
#
$IPTABLES -t nat -F PREROUTING ACCEPT
$IPTABLES -t nat -F POSTROUTING ACCEPT
$IPTABLES -t nat -F OUTPUT ACCEPT
#
# reset domyślnej polityki w tabeli mangle.
#

```

```
$IPTABLES -t mangle -P PREROUTING ACCEPT
$IPTABLES -t mangle -P OUTPUT ACCEPT
#
# Czyścimy nasze reguły we wszystkich tablicach.
#
$IPTABLES -F
$IPTABLES -t nat -F
$IPTABLES -t mangle -F
#
# usuwamy łańcuchy użytkownika z wszystkich tabel.
#
$IPTABLES -X
$IPTABLES -t nat -X
$IPTABLES -t mangle -X
```

9. Bibliografia / sznurki

Głównie wzorowałem się na tutorialu Oscara Andreassona, który moim zdaniem jest chyba najbardziej bogatym tutorialiem o iptables.

Jego strona: <http://iptables-tutorial.frozentux.net/>

Następnym linkiem o który się oparłem to polski serwis HOWTO

Strona: www.howto.pl

Oraz dokumenty RFC, które można pobrać ze strony:

www.faqs.org

Jeżeli coś napisałem źle, albo mało zrozumiale to proszę o kontakt pod agentsmith0@wp.pl, w miarę możliwości i skromnej wiedzy postaram się odpowiedzieć na pytania dotyczące w/w tematu.