Konfiguracja sieci w systemach Unix/Linux

Witold Paluszyński
Katedra Cybernetyki i Robotyki
Politechnika Wrocławska
http://www.kcir.pwr.edu.pl/~witold/

Tworzenie sieci

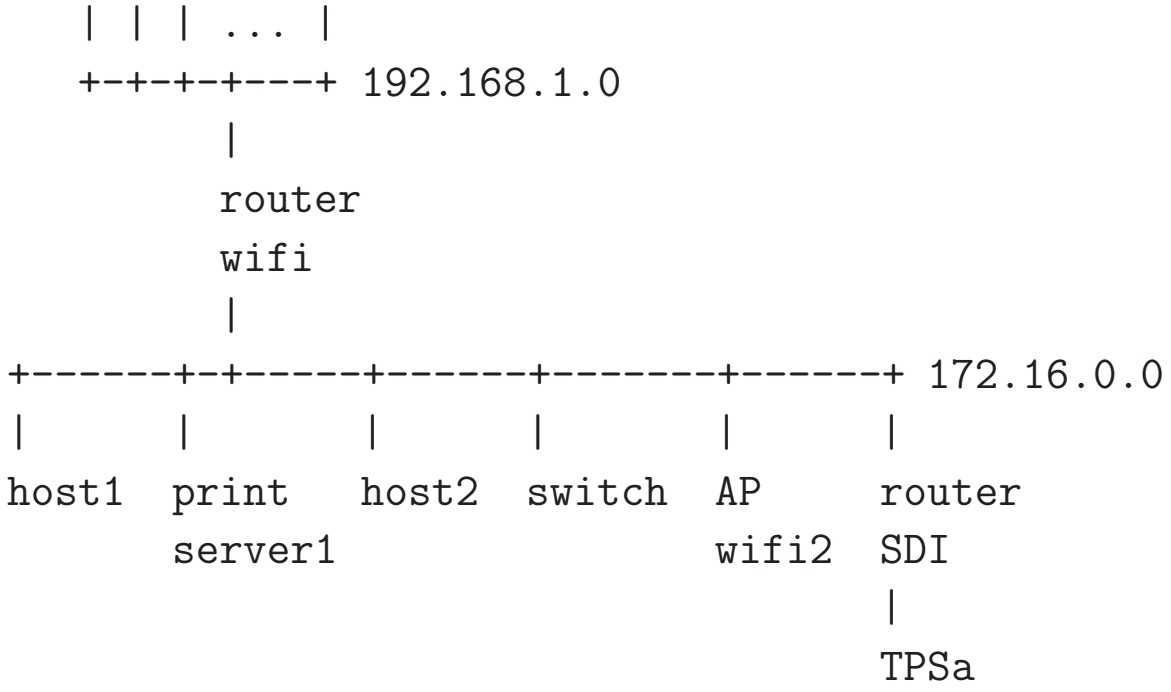
- zaplanowanie struktury fizycznej i logicznej sieci
- przydzielenie adresów IP
- uruchomienie fizyczne sieci:
  - interfejsy sieciowe komputerów
  - urządzenia sieciowe: transceivery, repeatery, huby, switche, routery
  - połączenia

- konfigurowanie interfejsów sieciowych komputerów
- konfigurowanie odwzorowania adresów fizycznych do adresów IP (arp)
- konfigurowanie warstwy filtrowania pakietów (*firewall*-a)
- konfigurowanie ścieżek statycznych i oprogramowania routera/ów
- konfigurowanie rezolwera adresów symbolicznych i DNS-ów

- konfigurowanie usług sieciowych

Planowanie struktury fizycznej i logicznej sieci

Podstawowym i jednym z najprzydatniejszych elementów planowania struktury sieci jest rysunek. W prostych przypadkach może być odręczny i służyć tylko do pierwotnej konfiguracji i uruchamiania sieci. W bardziej rozbudowanych sieciach przydatne jest sporządzenie diagramu komputerowego, który może być aktualizowany przy dokonywaniu wszelkich zmian, a następnie wykorzystywany w raportach dla szefostwa firmy, w trakcie szkolenia nowych użytkowników i administratorów, a także udostępniany w Internecie, itp.



Konfiguracja interfejsu sieciowego

1. nazwa interfejsu (zwykle: nazwa modułu jądra obsługującego kartę sieciową, plus numer karty, np.: eth0, lan1, itp.)
2. adres IP
3. maska sieciowa
4. adres broadcastu

```
sequoia> ifconfig hme0 156.17.9.3 up netmask 255.255.255.128 \
                                      broadcast 156.17.9.127
```

```
sequoia> ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
        inet 127.0.0.1 netmask ff000000
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 156.17.9.3 netmask ffffff80 broadcast 156.17.9.127
```

Konfiguracja interfejsu (Sun Solaris)

```
diablo> ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 156.17.9.14 netmask ffffff80 broadcast 156.17.9.127
hme1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.128.1 netmask ffffff00 broadcast 192.168.128.255
```

Startowa konfiguracja interfejsu:

```
diablo> cat /etc/defaultrouter
156.17.9.6
diablo> cat /etc/hostname.hme0 /etc/hostname.hme1
diablo
diablo-sunray
diablo> cat /etc/netmasks
156.17.9.0      255.255.255.128
192.168.128.0   255.255.255.0 # SUNRAY ADD - DO NOT MODIFY
diablo> cat /etc/networks
loopback        127
arpanet         10              arpa    # Historical
SunRay-hme1     192.168.128.0   SunRay  # SUNRAY ADD - DO NOT MODIFY
```

Konfiguracja interfejsu (Linux)

```
amargosa> ifconfig -a
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU:3584  Metric:1
          RX packets:811140 errors:0 dropped:0 overruns:0 frame:0
          TX packets:811140 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0

eth0      Link encap:Ethernet  HWaddr 02:60:8C:7A:99:1C
          inet addr:156.17.30.22  Bcast:156.17.30.31  Mask:255.255.255.224
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10309963 errors:0 dropped:0 overruns:0 frame:5736
          TX packets:7436831 errors:0 dropped:0 overruns:0 carrier:0
          collisions:13844
          Interrupt:5 Base address:0x300 Memory:c8000-ca000

eth1      Link encap:Ethernet  HWaddr 02:60:8C:7A:96:D5
          inet addr:156.17.9.6  Bcast:156.17.9.127  Mask:255.255.255.128
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14298997 errors:39 dropped:0 overruns:0 frame:199369
          TX packets:8890668 errors:70 dropped:0 overruns:0 carrier:151
          collisions:444116
          Interrupt:9 Base address:0x310 Memory:cc000-ce000
```

Startowa konfiguracja interfejsu (Linux)

W większości współczesnych dystrybucji Linuksa startowa konfiguracja interfejsów komputera czytana jest z pliku `/etc/network/interfaces`. Przykład komputera ze statyczną konfiguracją sieciową:

```
panamint-238> cat /etc/network/interfaces
# The loopback network interface
auto lo eth0 eth0:1
iface lo inet loopback

# The primary network interface
iface eth0 inet static
        address 156.17.9.7
        netmask 255.255.255.128
        network 156.17.9.0
        broadcast 156.17.9.127
        gateway 156.17.9.6
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 156.17.9.3
        dns-search ict.pwr.wroc.pl
```

Konfiguracja DHCP

Przedstawiony dotychczas proces podstawowej konfiguracji sieci komputerowej dotyczy przypadku statycznego, czyli stałego umiejscowienia naszego komputera w sieci, z na stałe przypisanymi parametrami: adresem IP i parametrami sieciowymi, ustaloną konfiguracją routingu (który może wykorzystywać dynamicznie zmienną tablicę ścieżek) i translacji nazw symbolicznych, itd.

Alternatywnie, konfiguracja sieciowa komputera może być dynamicznie wynegocjowana z serwerem systemu DHCP (Dynamic Host Configuration Protocol). Interfejs sieciowy możemy skonfigurować jako `dhcp` powodując uruchomienie komunikacji z serwerem DHCP (jeśli jest taki w sieci lokalnej) i pobranie od niego wszystkich parametrów (lub tylko niektórych, zgodnie z konfiguracją klienta DHCP).

Przykład konfiguracji DHCP obu interfejsów sieciowych (typu Debiana):

```
shuksan-206> cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet dhcp
# IIUWr
       wireless-essid stud-WiFi
```

Opcje w konfiguracji DHCP

```
shuksan-533> cat /etc/dhclient.conf
# timing params, slightly aggressive at first, but then infrequent
backoff-cutoff 2;
initial-interval 1;
select-timeout 0;
timeout 30;
reboot 0;
retry 60;

request subnet-mask, broadcast-address, time-offset, routers,
        domain-name, domain-name-servers, host-name,
        netbios-name-servers, netbios-scope;
require subnet-mask, domain-name-servers;

# my individual hostname
#send host-name "shuksan";
supersede host-name "shuksan";
# watch for the trailing space here
prepend domain-name "palnet ict.pwr.wroc.pl up.wroc.pl stud.ii ";
# in case we have a name server we want to use it
#prepend domain-name-servers 127.0.0.1;
```

Ciąg dalszy opcji dla klienta dhcp (/etc/dhclient.conf):

```
# my fallback static "lease"
lease {
  interface "eth0";
  fixed-address 156.17.9.95;
  option subnet-mask 255.255.255.128;
  option routers 156.17.9.6;
  option domain-name-servers 156.17.9.3,156.17.9.6,156.17.18.10;
  option host-name "shuksan";
  option domain-name "palnet ict.pwr.wroc.pl up.wroc.pl stud.ii ";
  renew 0 2038/1/17 02:14:07;
  rebind 0 2038/1/17 03:14:07;
  expire 0 2038/1/17 04:14:07;
}
```

Wirtualne interfejsy sieciowe

Wiele współczesnych systemów uniksowych pozwala na tworzenie **wirtualnych interfejsów** sieciowych. Wirtualny interfejs jest drugim, całkowicie oddzielnym interfejsem, korzystającym z tego samego interfejsu rzeczywistego (rzeczywistej karty sieciowej), co pierwotny interfejs skonfigurowany dla tej karty. Pozwala to np. komputerowi posiadającemu tylko jedną kartę sieciową komunikować się z więcej niż jedną siecią komputerową, albo posiadać więcej niż jeden adres sieciowy, i np. związany z tym adresem serwer.

Jeśli system obsługuje wirtualne interfejsy sieciowe, to aby ich użyć, wystarczy je skonfigurować. Interfejsy wirtualne posiadają zwykle nazwy postaci `eth0:1` gdzie `eth0` jest nazwą podstawowego interfejsu związanego z danym interfejsem fizycznym (kartą sieciową). Na przykład, konfiguracja na Linuksie:

```
ifconfig eth0:1 192.168.128.1 netmask 255.255.255.0
```

Konfiguracja interfejsu na Solarisie:

```
ifconfig eri0:1 plumb
ifconfig eri0:1 192.168.128.1 netmask 255.255.255.0
ifconfig eri0:1 up
```

```
panamint-238> cat /etc/network/interfaces
# The loopback network interface
auto lo eth0 eth0:1
iface lo inet loopback

# The primary network interface
iface eth0 inet static
        address 156.17.9.7
        netmask 255.255.255.128
        network 156.17.9.0
        broadcast 156.17.9.127
        gateway 156.17.9.6
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 156.17.9.3
        dns-search ict.pwr.wroc.pl

# Additional virtual interface
iface eth0:1 inet static
        address 156.17.9.60
        netmask 255.255.255.128
        network 156.17.9.0
        broadcast 156.17.9.127
```

Protokół ARP

```
shasta-749> arp -a
Net to Media Table: IPv4
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- --------------- ----- ---------------
eri0   printer.palnet       255.255.255.255       00:01:e6:23:36:a1
eri0   router.palnet        255.255.255.255       00:03:0a:00:90:ee
eri0   wifi.palnet          255.255.255.255       00:12:a9:55:5d:b7
eri0   sierra.palnet        255.255.255.255       08:00:20:7d:f1:3a
eri0   shasta.palnet        255.255.255.255 SP    00:03:ba:08:47:7b
eri0   BASE-ADDRESS.MCAST.NET 240.0.0.0       SM    01:00:5e:00:00:00
```

W wielu sieciach można wykorzystywać protokół ARP do samodzielnego odkrywania powiązań adresów fizycznych MAC urządzeń sieciowych z ich (deklarowanymi) adresami IP. W pozostałych przypadkach (albo w odniesieniu do niektórych tylko komputerów w sieci) można stosować statyczne powiązania, które nie mogą zostać usunięte przez ARP:

```
arp -s 156.17.9.3 0:1:2:3:4:5
arp -a 156.17.9.3
...
```

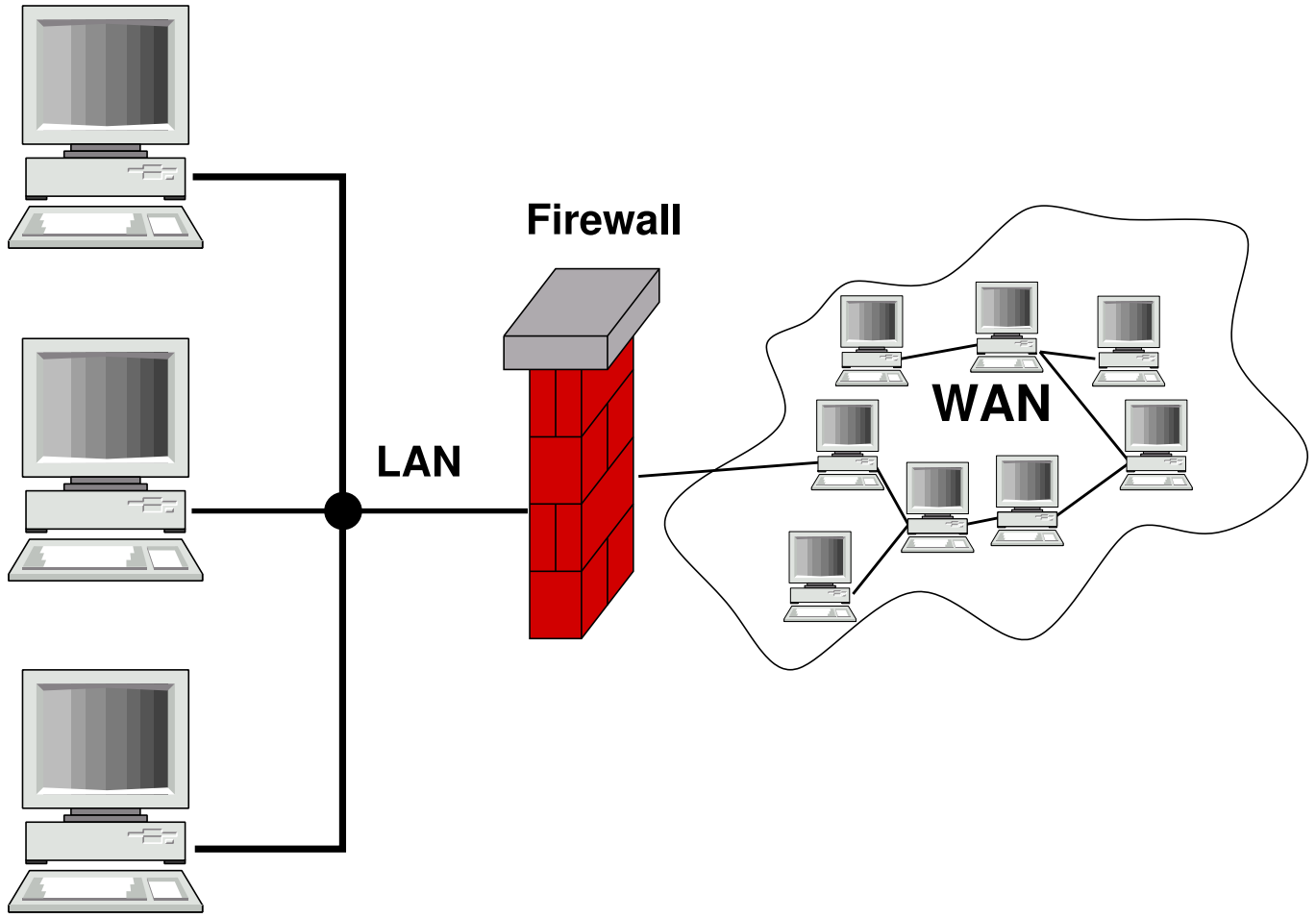
Protokół ARP (Linux)

Funkcje ARP w Linuksie realizuje jądro systemu. Można konfigurować te funkcje ustawiając odpowiednie zmienne jądra (???).

Alternatywnie można uruchomić zewnętrzny program `arpd` realizujący część tych funkcji zamiast jądra, z możliwością bardziej elastycznej ich konfiguracji.

Firewall jądra Linuksa

Jądro Linuksa posiada pewne funkcje o charakterze zapory ogniowej (*firewall*). Zasadniczo zapora ogniowa służy do izolacji i obrony systemu wewnętrznego względem świata zewnętrznego. Funkcje *firewall*a Linuksa pozwalają kontrolować pakiety sieciowe przenikające do systemu przez interfejs sieciowy.



Reguły filtrowania pakietów

Jądro Linuksa posiada wbudowaną, konfigurowalną warstwę filtrowania pakietów sieciowych. Dzięki temu jest możliwa realizacja takich funkcji jak:

- wybiórcze blokowanie pakietów wpływających do systemu z interfejsów sieciowych,
- blokowanie pakietów wysyłanych do sieci przez oprogramowanie systemu,
- konwersja pakietów zgodnie z określonymi regułami.

W starszych implementacjach jądra Linuksa konfigurację warstwy filtrowania pakietów realizowały programy `ipfwadm` i `ipchains`. W nowych jądrach (≥ 2.4) za konfigurację odpowiedzialny jest program `iptables`.

Struktura iptables

System iptables zawiera **łańcuchy** reguł, które zebrane są w **tabele**. Można tworzyć nowe reguły, i dodawać je do konkretnych łańcuchów konkretnych tabel. Jak również modyfikować istniejące w nich reguły. Podstawowe tabele iptables:

filter

To jest domyślna tabela, zawiera łańcuchy: INPUT, FORWARD, OUTPUT.

nat

Trafiają do niej pakiety nawiązujące nowe połączenia, zawiera łańcuchy: PREROUTING, OUTPUT, POSTROUTING.

mangle

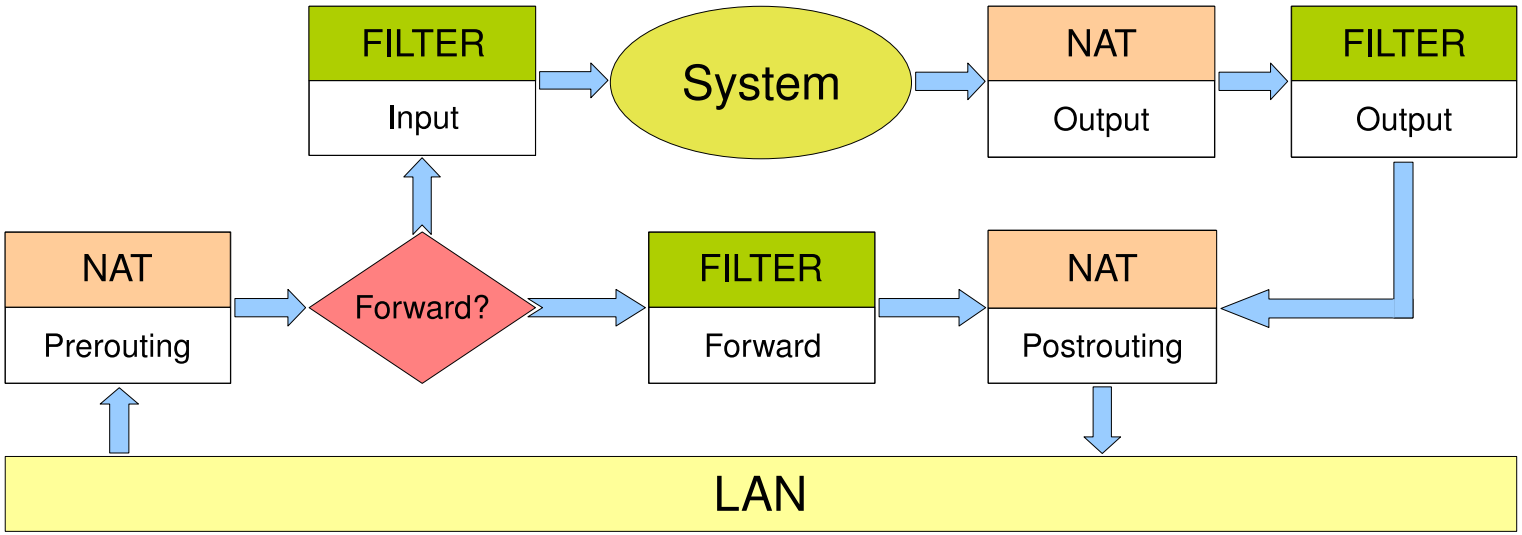
Tabela dla wyspecjalizowanych konwersji pakietów. Zawiera łańcuchy: PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING.

raw

Tabela stosowana przed wszystkimi innymi, zawiera łańcuchy: PREROUTING i OUTPUT.

Przepływ pakietów przez iptables

Poniższy diagram ilustruje przepływu pakietów przez łańcuchy iptables. Dla uproszczenia pominięte w nim zostały tabele raw i mangle.



Każdy pakiet sieciowy jest przetwarzany po kolei przez wszystkie reguły każdego łańcucha, do momentu, w którym reguła „pasuje" do pakietu. Wtedy wykonywana jest akcja określona przez daną regułę. Gdy żadna reguła łańcucha nie pasuje do pakietu, wtedy wykonywana jest domyślna akcja łańcucha.

Reguły iptables

Polecenie iptables może tworzyć nowe reguły filtrowania i dodawać je, lub usuwać, w określonym miejscu sekwencji reguł. Każda reguła określa przeznaczenie pakietu jako: ACCEPT, DROP, QUEUE, RETURN, lub przetwarzanie przez inny, indywidualnie zdefiniowany łańcuch reguł.

- ACCEPT — oznacza akceptację pakietu i wyjście z danego łańcucha
- DROP — skasowanie pakietu i zakończenie jego przetwarzania
- QUEUE — przetwarzanie pakietu w przestrzeni użytkownika
- RETURN — porzucenie bieżącego łańcucha i powrót do przetwarzania pakietu przez następną regułę łańcucha wywołującego; wykonanie akcji domyślnej jeśli jest to ostatnia reguła lub łańcuch wbudowany

Dopasowanie reguły do pakietu może być określone przez parametry pakietu: protokół i rodzaj usługi sieciowej (nr portu lub kod pakietu), adres IP nadawcy lub odbiorcy, i nazwę interfejsu sieciowego przez który pakiet wszedł do systemu, lub przez który ma być wysłany. Ponadto, mogą być stosowane dodatkowe moduły dopasowania pakietów, pozwalające na tworzenie bardziej wyspecjalizowanych reguł.

iptables: przykładowe konfiguracje (1)

Domyślnym celem (przeznaczeniem) jądra Linuksa jest ACCEPT dla wszystkich pakietów w tabeli filter w łańcuchach FORWARD, INPUT i OUTPUT.

Rozważmy sytuację, w której chcemy zezwolić na otwieranie dowolnych połączeń wychodzących (łańcuch OUTPUT), ale zabronić wszelkich pakietów przychodzących, jak również przepływu obcych pakietów przez naszą maszynę.

```
iptables -P FORWARD DROP
iptables -P INPUT DROP
```

Mogłoby się wydawać, że to jest minimalna sensowna konfiguracja systemu. W rzeczywistości jednak nie jest ona przydatna prawie do niczego, ponieważ nie wpuszcza pakietów odpowiedzi na połączenia wychodzące!!

Skonfigurowany powyższymi poleceniami system może tylko wysyłać polecenia, ale nie przyjmie żadnej odpowiedzi.

iptables: przykładowe konfiguracje (2)

Załóżmy, że jako minimalną konfigurację sieciową chcemy mieć zdolność browsowania Internetu, czyli otwierania połączeń HTTP do zdalnych serwerów na porcie 80. Aby wpuścić do systemu pakiety odpowiedzi dotyczące tych połączeń, można dodać reguły:

```
iptables -A INPUT --protocol tcp --source-port 80 -j ACCEPT
iptables -A INPUT --protocol udp --source-port 53 -j ACCEPT
```

Druga reguła umożliwia uzyskiwanie odpowiedzi z DNS w przypadku użycia adresów symbolicznych, które masowo pojawiają się w dokumentach HTTP (np. obrazki). Klient może komunikować się z serwerem DNS protokołem TCP lub UDP. Powyższa reguła umożliwia tylko komunikację UDP.

iptables: przykładowe konfiguracje (3)

Załóżmy teraz, że do podstawowego zestawu wykonywanych połączeń chcemy dodać jeszcze ssh (port 22) regułą:

```
iptables -A INPUT --protocol tcp --source-port 22 -j ACCEPT
```

Gdybyśmy chcieli dalej uruchomić serwer sshd i wpuścić ruch ssh inicjowany na zewnątrz, konieczna byłaby podobna ale dualna reguła:

```
iptables -A INPUT --protocol tcp --destination-port 22 -j ACCEPT
```

Podobna reguła będzie konieczna, gdybyśmy chcieli przyjmować na lokalnym serwerze X Window klientów łączących się z naszym serwerem bezpośrednio (a nie np. przez tunel ssh). Dla serwera :0 reguła ma postać:

```
iptables -A INPUT --protocol tcp --destination-port 6000 -j ACCEPT
```

iptables: przykładowe konfiguracje (4)

Powyższe przykłady pokazują sposób tworzenia zezwoleń na wpuszczanie ruchu pakietów sieciowych w oparciu o ich numery portów wychodzących lub docelowych. Dotyczy to komunikacji sieciowej opartej o połączenia TCP.
W podobny sposób można konfigurować ruch pakietów bezpołączeniowych UDP. Jednak nie obejmuje to np. pingów, wykorzystujących protokół ICMP warstwy sieciowej (IP).

Protokoły warstwy sieciowej nie mają połączeń ani numerów portów. Mają tylko kody pakietów, i iptables umożliwia tworzenie reguł według kodów pakietów. Chcąc mieć możliwość „ping"-owania innych systemów, musimy umożliwić przyjmowanie pakietów ICMP typu ECHO-REPLY, które mają kod 0.

```
iptables -A INPUT --protocol icmp --icmp-type 0 -j ACCEPT
```

Dla odmiany, chcąc umożliwić innym systemom skutecznego pingowania nas, musimy umożliwić pakietom ICMP typu ECHO (kod 8) wchodzenia do naszego systemu:

```
iptables -A INPUT --protocol icmp --icmp-type 8 -j ACCEPT
```

iptables: podsumowanie — konfiguracja minimalna

Powyższe reguły stanowią niezłą minimalną konfigurację komputera osobistego:

```
# ustawienie polityki restrykcyjnej
iptables -P FORWARD DROP
iptables -P INPUT DROP

# umozliwienie translacji adresow
iptables -A INPUT --protocol udp --source-port 53 -j ACCEPT

# umozliwienie pracy z ssh i HTTP
iptables -A INPUT --protocol tcp --source-port 80 -j ACCEPT
iptables -A INPUT --protocol tcp --source-port 22 -j ACCEPT

# przyjmowanie przychodzacych polaczen ssh
iptables -A INPUT --protocol tcp --destination-port 22 -j ACCEPT

# przyjmowanie klientow X Window
iptables -A INPUT --protocol tcp --destination-port 6000 -j ACCEPT

# pingowanie innych
iptables -A INPUT --protocol icmp --icmp-type 0 -j ACCEPT
# mozliwosc pingowania nas - w zaleznosci od preferencji
###iptables -A INPUT --protocol icmp --icmp-type 8 -j ACCEPT
```

iptables: przykładowe konfiguracje (5)

Jako małe uzupełnienie powyższej konfiguracji można uwzględnić następujące reguły, umożliwiające pracę w systemie NFS (na dysku importowanym z sieciowego serwera plików):

```
iptables -A INPUT --protocol tcp --source-port 2049 -j ACCEPT
iptables -A INPUT --protocol tcp --destination-port 2049 -j ACCEPT
```

Druga z powyższych reguł ma zastosowanie w przypadku eksportowania dysku z lokalnego systemu do innych systemów w sieci.

Routing (trasowanie?)

Czynność wyboru ścieżki sieciowej, do której należy wysłać dany pakiet sieciowy. Decyzja jest podejmowana na podstawie docelowego adresu IP pakietu, i jej wynikiem jest wybór komputera w (jednej z) sieci lokalnej(ych), do której(ych) dany komputer jest podłączony. Routing jest czynnością wykonywaną w ramach protokołu IP (warstwy sieciowej, w nomenklaturze ISO).

Routing realizowany jest w sposób niezwykle prosty: jądro Uniksa posiada tablicę ścieżek sieciowych, określająca powiązania docelowych adresów IP komputerów i całych sieci, z bramami, czyli adresami IP komputerów w sieci lokalnej, czyli takich, do których przesłanie jest bezpośrednie.

Może istnieć wiele ścieżek w tej tablicy, i wybierana jest zawsze najlepiej dopasowana, to znaczy najbardziej szczegółowa ścieżka zgodna z danym adresem docelowym. W braku takiej ścieżki używana jest specjalna ścieżka domyślna, a gdy jej nie ma, pakietu nie da się wysłać do miejsca przeznaczenia, i routing kończy się niepowodzeniem. Pakiet zostaje zwyczajnie skasowany, natomiast do nadawcy może zostać wysłany komunikat informujący go o błędzie w jego tablicy ścieżek.

Routing — tablica ścieżek sieciowych

zwykły komputer

```
sequoia> netstat -rn

Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
156.17.9.0           156.17.9.3            U         1  27617  le0
224.0.0.0            156.17.9.3            U         1      0  le0
default              156.17.9.16           UG        1  87628
127.0.0.1            127.0.0.1             UH        1     58  lo0
```

komputer służący jako router kilku sieci lokalnych

```
amargosa-201> netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
156.17.9.160    156.17.9.13     255.255.255.224 UG        0 0          0 eth1
156.17.9.128    156.17.9.22     255.255.255.224 UG        0 0          0 eth1
156.17.30.0     0.0.0.0         255.255.255.224 U         0 0          0 eth0
156.17.9.0      0.0.0.0         255.255.255.128 U         0 0          0 eth1
127.0.0.0       0.0.0.0         255.0.0.0       U         0 0          0 lo
0.0.0.0         156.17.9.16     0.0.0.0         UG        0 0          0 eth1
```

```
tahoe-201> netstat -rn

Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
156.17.19.158        156.17.30.126         UGH      1      0
156.17.19.190        156.17.30.126         UGH      1      0
156.17.19.191        156.17.30.126         UGH      1      0
156.17.19.188        156.17.30.126         UGH      1      0
156.17.19.156        156.17.30.126         UGH      1      0
156.17.19.189        156.17.30.126         UGH      1      0
156.17.236.2         156.17.30.126         UGH      1      0
156.17.19.154        156.17.30.126         UGH      1      0
156.17.19.186        156.17.30.126         UGH      1      0
156.17.19.155        156.17.30.126         UGH      1      0
156.17.19.187        156.17.30.126         UGH      1      0
156.17.19.184        156.17.30.126         UGH      1      0
156.17.19.153        156.17.30.126         UGH      1      0
156.17.19.182        156.17.30.126         UGH      1      0
156.17.63.250        156.17.30.126         UGH      1      0
156.17.19.180        156.17.30.126         UGH      1      0
156.17.249.158       156.17.30.126         UGH      1      0
156.17.19.149        156.17.30.126         UGH      1      0
156.17.19.146        156.17.30.126         UGH      1      0
156.17.19.144        156.17.30.126         UGH      1      0
156.17.19.176        156.17.30.126         UGH      1      0
156.17.19.145        156.17.30.126         UGH      1      0
156.17.19.177        156.17.30.126         UGH      1      0
156.17.19.174        156.17.30.126         UGH      1      0
156.17.19.143        156.17.30.126         UGH      1      0
156.17.19.175        156.17.30.126         UGH      1      0
156.17.229.217       156.17.30.126         UGH      1      0
156.17.19.140        156.17.30.126         UGH      1      0
... dalszych 617 sciezek
```

Routing statyczny

Utrzymywanie tablicy ścieżek w stanie aktualnym nie jest łatwym zadaniem. Ścieżki sieciowe mogą być wprowadzane ręcznie przez administratora (są to tzw. ścieżki statyczne), jak również automatycznie, przez programy odpowiedzialne za aktualizację informacji o połączeniach między sieciami (ścieżki dynamiczne).

Routing statyczny oznacza wyłącznie ręczną aktualizację tablicy ścieżek. Ścieżki są tworzone w czasie startu systemu z plików startowych, i ewentualnie zmieniane ręcznie przez administratora, gdy pojawia się taka potrzeba, na przykład przy przeadresowaniu komputerów i/lub sieci lokalnych.

```
### polecenie w skryptach startowych
# route add default 156.17.9.16 1
# route add -net 156.17.4.0 gw 156.17.9.6

Solaris# cat /etc/defaultrouter
156.17.9.16

Linux# cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=panamint
GATEWAY=156.17.9.16
NISDOMAIN=stud.ict.pwr.
```

Aktualizacja tablicy ścieżek — protokół ICMP

Protokół komunikatów kontrolnych ICMP warstwy sieciowej (IP) posiada opcje odpowiadania na pakiety, co do których router wie, że albo nie mogą być poprawnie doręczone (ICMP Host Unreachable), albo że istnieje dla tych pakietów prostsza ścieżka do przeznaczenia (ICMP Host Redirect). Na podstawie takich informacji komputer wysyłający pakiet może zaktualizować swoje tablice ścieżek.

```
sequoia-379> ping 156.17.9.131
PING 156.17.9.131: 56 data bytes
ICMP Host redirect from gateway tahoe (156.17.9.16)
 to ithaca (156.17.9.22) for 156.17.9.131
ICMP Host Unreachable from gateway ithaca (156.17.9.155)
 for icmp from sequoia (156.17.9.3) to 156.17.9.131
ICMP Host Unreachable from gateway ithaca (156.17.9.155)
 for icmp from sequoia (156.17.9.3) to 156.17.9.131
ICMP Host Unreachable from gateway ithaca (156.17.9.155)
 for icmp from sequoia (156.17.9.3) to 156.17.9.131
^C
----156.17.9.131 PING Statistics----
3 packets transmitted, 0 packets received, 100% packet loss
```

Mechanizm ICMP Host Redirect można wywołać, tworząc niepoprawną ścieżkę sieciową przez komputer, który nie jest routerem (bramą) sieci lokalnej.

```
### host 156.17.193.201 jest poza siecia lokalna
### host 156.17.9.13 jest w sieci lokalnej ale nie jest routerem

# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
156.17.9.0      0.0.0.0         255.255.255.128 U         0 0          0 eth0
0.0.0.0         156.17.9.6      0.0.0.0         UG        0 0          0 eth0

# route add -host 156.17.193.201 gw 156.17.9.13

# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
156.17.193.201  156.17.9.13     255.255.255.255 UGH       0 0          0 eth0
156.17.9.0      0.0.0.0         255.255.255.128 U         0 0          0 eth0
0.0.0.0         156.17.9.6      0.0.0.0         UG        0 0          0 eth0

# ping 156.17.193.201
PING 156.17.193.201 (156.17.193.201) 56(84) bytes of data.
64 bytes from 156.17.193.201: icmp_req=1 ttl=249 time=3.31 ms
From 156.17.9.13: icmp_seq=2 Redirect Host(New nexthop: 156.17.9.6)
64 bytes from 156.17.193.201: icmp_req=2 ttl=249 time=0.918 ms
64 bytes from 156.17.193.201: icmp_req=3 ttl=249 time=0.811 ms
64 bytes from 156.17.193.201: icmp_req=4 ttl=249 time=0.913 ms
```

Zarządzanie routingiem — Linux

Mechanizm ICMP Host Redirect może być przydatny w przypadku błędnej konfiguracji komputera, ale może też być zagrożeniem, przed którym czasami warto się zabezpieczyć.

```
### zablokowanie przyjmowania redirect-ow ICMP
# echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects

### zablokowanie wysylania redirect-ow ICMP
# echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects

### zablokowanie przekazywania "obcych" pakietow sieciowych
# echo "0" > /proc/sys/net/ipv4/ip_forward

### nie wysylamy odpowiedzi na rozglaszane "pingi" ICMP
# echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

### nie wysylamy odpowiedzi na zadne "pingi" ICMP
# echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

Routing dynamiczny

Ścieżki statyczne wpisane „ręcznie" do tablicy ścieżek pozostają tam na stałe. Odróżniają się one od ścieżek dynamicznych, wpisywanych przez tzw. demony routingu, stale aktualizujące tablicę ścieżek przez wpisywanie i usuwanie ścieżek. Te programy posiadają wiedzę aprioryczną o połączeniach w sieci lokalnej, i wymieniają się tą wiedzą z innymi takimi programami, stosując w tym celu jeden z kilku istniejących protokołów komunikacyjnych, tzw. protokołów routingu.

Najstarszym takim protokołem jest RIP (*Routing Information Protocol*), pozwalający na wyrażanie informacji o połączeniach i ich kosztach w formie liczby przeskoków (*hop-count*). Standardowy demon routingu realizujący RIP w systemie Unix to `routed`. Przykładowy plik konfiguracyjny:

```
sequoia-295> cat /etc/gateways
net 0.0.0.0 gateway 156.17.9.16 metric 1 passive
net 156.17.30.0 gateway 156.17.9.6 metric 1 active
```

Protokół RIP, podobnie jak demon `routed`, ma wiele ograniczeń, i niezwykle prostą konstrukcję opierającą się na rozgłaszaniu posiadanej informacji o ścieżkach. Ta prostota jest wielką zaletą w konstruowaniu małych sieci o niewielkich wymaganiach. Jednak w poważniejszych sytuacjach konieczne jest stosowanie innych programów i protokołów.

W systemie Linux dostępny jest bardzo dobry demon routingu `gated`. Obsługuje on wiele protokołów routingu, np. OSPF. Przykład konfiguracji:

```
amargosa> cat /etc/gated.conf
rip yes {
  preference 200 ;
  interface "eth1" noripin ripout ;
  interface "eth0" noripin noripout ;
};
ospf yes {
  area 156.17.40.0 {
    networks {
      156.17.9.0 masklen 25;
      156.17.30.0 masklen 27;
    };
    interface "eth0"  cost 20 {enable ;};
    interface "eth1"  cost 2 {enable ;};
  };
};
```

Badanie ścieżek sieciowych

Użytecznym narzędziem do badania ścieżek sieciowych jest traceroute. Wysyła ono określone pakiety do zadanego adresu docelowego, jednak ustawiając czas życia pakietów (maksymalna liczba kroków sieciowych, jakie pakiet może pokonać) na stopniowo zwiększające się od zera wartości. Każda kolejna brama sieciowa, kasująca pakiet ze względu na wyzerowany czas życia odsyła (nieobowiązkowo) nadawcy informację o tym, którą traceroute wykorzystuje dla odtworzenia ścieżki pokonywanej przez pakiet w drodze do przeznaczenia.

```
traceroute
```

```
traceroute -I
```

```
traceroute -I -m 60
```

Istnieje wiele innych narzędzi do badania ścieżek sieciowych. Na przykład, program mtr łączy w sobie funkcjonalność traceroute i pinga, wysyłając pakiety w kółko i wyświetlając dynamicznie zmieniający się stan połączeń.

Translacja nazw symbolicznych — rezolwer

- rezolwer: algorytm translacji adresów symbolicznych na adresy IP stosowany w danym komputerze
- rezolwer ma zwykle postać biblioteki dynamicznej kompilatora C (`libresolv`) implementującej funkcje: `gethostbyname` i `gethostbyaddr`
- z biblioteką tą linkują się zarówno programy systemowe jak i programy użytkowników, co pozwala na jednolitą interpretację danej nazwy

- W starszych systemach rezolwer korzysta z tablicy translacji adresów `/etc/hosts`, i opcjonalnie z serwera lub serwerów DNS zgodnie ze specyfikacją w pliku `/etc/resolv.conf`

```
sierra> cat /etc/hosts

127.0.0.1       localhost
172.16.0.1      sierra.palnet   sierra loghost
172.16.0.3      shasta.palnet   shasta

156.17.9.3      sequoia.ict.pwr.wroc.pl sequoia

156.17.4.4      tryglaw.ii.uni.wroc.pl  tryglaw

156.17.181.99   yoyo.ar.wroc.pl         yoyo
156.17.181.101  geo1.ar.wroc.pl         geo1



sierra> cat /etc/resolv.conf
domain palnet
search palnet ict.pwr.wroc.pl ii.uni.wroc.pl ar.wroc.pl
nameserver 172.16.0.1
```

- W nowszych systemach rezolwer może korzystać z wielu różnych źródeł danych, takich jak: lokalne pliki konfiguracyjne, system DNS, bazy danych NIS lub LDAP. Odwołuje się do tych źródeł zgodnie z precyzyjnym algorytmem określającym w jakiej kolejności z nich korzystać i w jakich sytuacjach odwoływać się do innych źródeł. Algorytm zadany jest plikiem konfiguracyjnym `/etc/nsswitch.conf` natomiast plik `/etc/resolv.conf` określa tylko kolejność zapytań do serwerów DNS.

  Przykłady specyfikacji rezolwera z pliku `/etc/nsswitch.conf`:

  ```
  hosts:      files dns
  ```

  Sprawdzamy najpierw w pliku `/etc/hosts`, a tylko dla nazw, których tam nie ma, odwołujemy się do systemu DNS.

  ```
  hosts:      ldap dns [NOTFOUND=return] files
  ```

  Sprawdzamy kolejno: w bazie danych LDAP i systemie DNS. Gdy system DNS odpowie, że nazwy nie ma, to traktujemy tę odpowiedź jako ostateczną i kończymy zapytanie. Tylko gdyby system DNS nie odpowiadał, sprawdzamy w lokalnym pliku `/etc/hosts`

Translacja nazw symbolicznych — system DNS

- DNS (*Domain Name System*) — hierarchiczny, rozproszony system nazw symbolicznych w Internecie
- oparty na oddelegowaniu administracji domenami różnym instytucjom, korzystającym z własnych serwerów DNS, automatycznie wymieniającym między sobą informacje o administrowanych przez siebie domenach

  domena — poddrzewo hierarchicznego drzewa nazw
- własności: nadmiarowość, replikacja, buforowanie, duża niezawodność i tolerancja błędów, optymalizacja procesu uzyskiwania odpowiedzi w warunkach rzadkich zmian
- serwer DNS — program, którego zadaniem jest podawanie translacji adresu określonego w zapytaniu klienta, i komunikujący się z innymi serwerami DNS, w celu jej znalezienia
- serwery DNS mogą posiadać redundancję — dla danej domeny można wprowadzić oprócz serwera głównego (**primary**), równoważne serwery dodatkowe (**secondary**)

Serwery systemu DNS

- serwer DNS domyślnie jest **rekurencyjny**; w sytuacji gdy nie zna odpowiedzi na otrzymane zapytanie, sam kontaktuje się z innymi serwerami aby ją uzyskać, i udzielić pytającemu klientowi

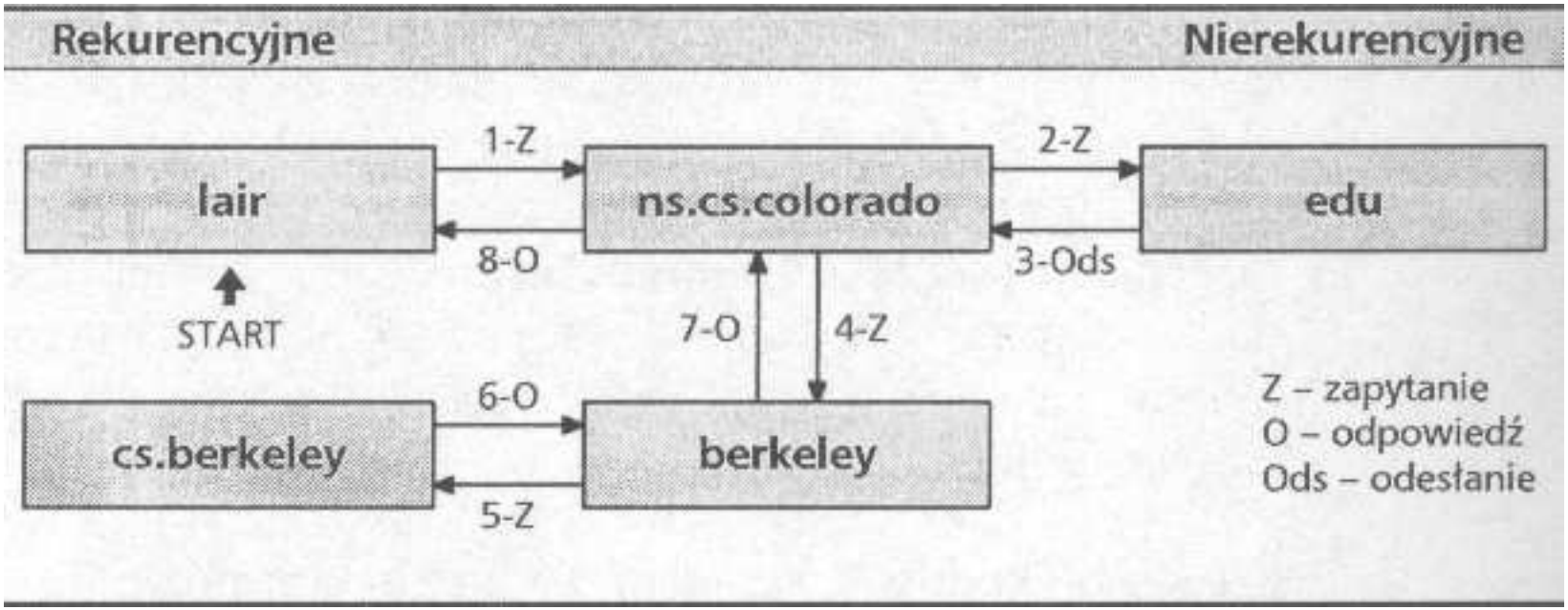
  rekurencyjny serwer jest właściwym rozwiązaniem dla sieci lokalnej, ponieważ pozwala klientom zawsze uzyskiwać odpowiedzi na swoje pytania, a ponadto może przechowywać uzyskane odpowiedzi, i udzielać ich potem kolejnym klientom bez ponownego odpytywania rekurencyjnego

- serwer DNS może być również **nierekurencyjny**; w przypadku nieznajomości odpowiedzi serwer taki nie pyta się innych serwerów, tylko odpowiada tzw. odsyłaczem (ang. *referral*), podającym adres innego, bardziej właściwego dla danej domeny serwera DNS

  serwery DNS wyższego poziomu w hierarchii Internetu (np. serwery główne takich domen jak .com albo .pl) są z zasady nierekurencyjne, więc tym bardziej nie przechowują informacji, które ich nie dotyczą

Dla domen pośrednich pomiędzy siecią lokalną a domeną główną Internetu musimy wybrać pomiędzy pracą rekurencyjną a nierekurencyjną serwera DNS. Jednak nierekurencyjny serwer nie może obsługiwać normalnych klientów, nieprzygotowanych na otrzymanie na swoje zapytanie odpowiedzi w postaci odsyłacza.

Przykład sekwencji odwołań do serwerów DNS dla zapytania o nazwę `mammoth.cs.berkeley.edu` wykonanego na komputerze `lair.cs.colorado.edu`:



Serwery DNS

primary — jest tylko jeden taki serwer dla strefy (ang. *zone*); strefa jest częścią domeny administrowaną przez serwer

secondary — takich może być dla danej strefy wiele, automatycznie aktualizują one swoje dane i ich odpowiedź jest równoważna odpowiedzi serwera *primary*

caching-only — nie jest właściwym źródłem informacji o żadnej strefie, nie posiada własnych informacji tylko realizuje funkcję rekurencyjnego odpytywania innych serwerów i przechowuje informacje przez dozwolony okres; można go uważać za rodzaj aktywnego klienta; jeśli nie chcemy zakładać w danym systemie serwera DNS, ale chcemy zaoszczędzić na ruchu sieciowym do zewnętrznych serwerów DNS, to możemy założyć właśnie serwer *caching-only*

Jeden serwer (uruchomiona instancja programu) może być serwerem primary dla jednej strefy (lub kilku), i serwerem secondary dla grupy innych stref, albo może być czystym serwerem caching-only.

Konfiguracja serwera DNS

Pliki konfiguracyjne serwera DNS:

- /etc/named.conf — określa zestaw stref administrowanych przez serwer i położenie plików definiujących strefy

- pliki definiujące strefy — zbiory rekordów zasobów (*resource records*):
  - rekord SOA — definiuje strefę i jej podstawowe parametry
  - rekord NS — identyfikuje serwery DNS autorytatywne (właściwe) dla danej strefy
  - rekord A — powiązanie pojedynczej nazwy z adresem IP
  - rekord TXT — dodatkowe informacje dla rekordu A
  - rekord HINFO — dodatkowe informacje o typie komputera rekordu A
  - rekord MX — określa obsługę usług pocztowych dla adresu
  - rekord CNAME — określa dodatkową nazwę (alias)
  - rekord PTR — powiązanie odwrotne: adresu IP z nazwą symboliczną

Odpytywanie serwerów DNS

Narzędzia pozwalające wysyłać zapytania do serwerów DNS: nslookup, host, dig.

nslookup jest narzędziem tradycyjnym i wygodnym dla prostych zapytań.

```
### zwykle zapytanie o adres symboliczny
nslookup amargosa

### zapytanie konkretnego serwera o rekord MX dla adresu
nslookup -query=mx stud.ict.pwr.wroc.pl sun2.pwr.wroc.pl
```

dig ma bardzo rozbudowany interfejs i domyślnie wyświetla dużo informacji. Powoduje to, że lepiej nadaje się do zapytań wyspecjalizowanych.

```
### zwykle zapytanie o adres symboliczny
dig www.uw.edu.pl

### zablokowanie zapytan rekurencyjnych
dig +norecurse www.uw.edu.pl

### zapytanie o serwery DNS danego hosta
dig +nssearch www.uw.edu.pl
```

Badanie stanu warstwy sieciowej narzędziem netstat

Pierwszy argument określa podsystem (domyślnie lista otwartych gniazdek):

- -r - tablica routingu jądra
- -g - istniejące grupy multicast
- -i - istniejące interfejsy sieciowe
- -M - połączenia przekierowywane (masquerade)
- -s - statystyki

```
# What Network Services are Running?
netstat -tanup

# or if you just want tcp services
netstat -tanp
netstat -ap|grep LISTEN|less

# Need stats on dropped UDP packets?
netstat -s -u

# or TCP
netstat -s -t

# or summary of everything
netstat -s

# or looking for error rates on the interface?
netstat -i

# Listening interfaces?
netstat -l
```

Badanie sieci narzędziem nmap

nmap — przykłady z manuala:

```
# skanuje wszystkie porty i wyswietla poszerzone informacje
nmap -v target.example.com

# skanuje metoda Stealth SYN caly segment sieci (255 adresow ip)
# z proba odgadniecia systemu operacyjnego
sudo nmap -sS -O target.example.com/24

# skanuje metoda Xmas tree pierwsza polowke segmentu sieci
# tylko wybrane porty: sshd, DNS, pop3d, imapd, i 4564???
nmap -sX -p 22,53,110,143,4564 198.116.*.1-127

# skanuje w standardowy sposob cala domene
host -l company.com | cut  -d  -f 4 | ./nmap -v -iL -

# rozszerzone, zaawansowane testy (-A), z agresywnym ustawieniem
# opoznien (skrocone timeouty oczekiwania na odpowiedzi)
nmap -A -T4 scanme.nmap.org
```

Monitorowanie ruchu sieciowego narzędziem tcpdump

Używanie tcpdump wymaga użycia przywilejów roota, ponieważ przełącza kartę sieciową w tryb „podsłuchiwania" *promiscous*.

Monitorowanie całego ruchu z wyjątkiem bieżącego połączenia ssh:

```
tcpdump -i eth0 -nN -vvv -xX -s 1500 port not 22
```

Odfiltrowanie również portu 123, badanie całych pakietów:

```
tcpdump -i eth0 -nN -vvv -xX -s 0 port not 22  and port not 123
```

Filtrowanie konkretnego hosta:

```
tcpdump -i eth0 -nN -vvv -xX  port not 22 and host 81.169.158.205
```

Wyświetlanie tylko adresu IP i małej porcji danych, przerwij po 20-tu pakietach:

```
tcpdump -i eth0 -nN  -s 1500 port not 22 -c 20
```

Wykrywanie ataków DOS przez wyświetlanie pakietów SYN na wszystkich interfejsach:

```
tcpdump 'tcp[13] & 2 == 2'
```

Materiały źródłowe

http://wiki.debian.org/iptables

http://nmap.org/bennieston-tutorial/

http://www.thegeekstuff.com/2011/06/iptables-rules-examples/