Linux
i obsługa sieci dla profesjonalistów

Konfiguracja i stosowanie bezpiecznych usług sieciowych

Rob VandenBrink

Helion

Tytuł oryginału: Linux for Networking Professionals: Securely configure and operate Linux network services for the enterprise

Tłumaczenie: Grzegorz Werner

ISBN: 978-83-283-9710-1

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/linobs
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

ROZDZIAŁ 2

Podstawowa konfiguracja i obsługa sieci w Linuksie — praca z interfejsami lokalnymi

W tym rozdziale wyjaśnimy, jak wyświetlać oraz konfigurować lokalne interfejsy i trasy w linuksowym hoście. Omówimy zarówno nowe, jak i tradycyjne polecenia służące do wykonywania takich operacji jak wyświetlanie i modyfikowanie danych adresowych IP, lokalnych tras oraz innych parametrów interfejsu. Opiszemy też, jak konstruować adresy IP i adresy podsieci w formacie binarnym.

Niniejszy rozdział powinien dać Ci solidne podstawy do zrozumienia zagadnień omawianych dalej w tej książce, takich jak diagnozowanie problemów z siecią, „utwardzanie" hosta oraz instalowanie zabezpieczonych usług.

W tym rozdziale omówimy następujące tematy:

- praca z ustawieniami sieci — dwa zbiory poleceń,
- wyświetlanie informacji o interfejsie IP,
- adresy IPv4 i maski podsieci,
- przypisywanie adresu IP do interfejsu.

Wymagania techniczne

W tym i wszystkich pozostałych rozdziałach, kiedy będziemy omawiać różne polecenia, możesz wypróbowywać je na własnym komputerze. Polecenia w tej książce zilustrowano w systemie Ubuntu Linux 20 (wersja ze wsparciem długoterminowym — LTS), ale w większości powinny działać identycznie lub bardzo podobnie w dowolnej dystrybucji Linuksa.

Praca z ustawieniami sieci — dwa zbiory poleceń

Przez wiele lat `ifconfig` (konfiguracja interfejsu) i powiązane polecenia były jednym z filarów Linuksa, do tego stopnia, że choć obecnie są w większości dystrybucji uznawane za przestarzałe, to nadal pozostają zakodowane w pamięci mięśniowej wielu administratorów systemu i sieci.

Dlaczego zastąpiono te stare polecenia sieciowe? Z paru powodów. Nie obsługują dobrze części nowego sprzętu (zwłaszcza kart sieciowych InfiniBand), a ponadto z biegiem lat, w miarę jak jądro Linuksa się zmieniało, ich działanie było coraz bardziej niespójne. Jednak presja na zachowanie wstecznej kompatybilności utrudniała rozwiązanie tych problemów.

Stare polecenia znajdują się w pakiecie oprogramowania `net-tools`, a nowe w pakiecie `iproute2`. Nowi administratorzy powinni skupić się na nowych poleceniach, ale znajomość starszych nadal bywa przydatna. Często spotyka się wiekowe komputery działające pod kontrolą Linuksa, maszyny, które prawdopodobnie nigdy nie zostaną zaktualizowane, a nadal używają starych poleceń. Dlatego opiszemy oba zestawy narzędzi.

Lekcja, jaką można z tego wyciągnąć, jest taka, że w świecie Linuksa jedyną stałą rzeczą jest zmiana. Stare polecenia pozostają dostępne, ale nie są domyślnie instalowane.

Aby zainstalować tradycyjne polecenia, wydaj następującą instrukcję:

```
robv@ubuntu:~$ sudo apt install net-tools
 [sudo] password for robv:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libfprint-2-tod1
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  net-tools
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/196 kB of archives.
After this operation, 864 kB of additional disk space will be used.
Selecting previously unselected package net-tools.
(Reading database ... 183312 files and directories currently installed.)
```

```
Preparing to unpack .../net-tools_1.60+git20180626.aebd88e-1ubuntu1_amd64.deb ...
Unpacking net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Setting up net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Processing triggers for man-db (2.9.1-1) ...
```

Warto w tym poleceniu instalacyjnym i jego wynikach zwrócić uwagę na kilka rzeczy:

- `sudo`: użyto polecenia `sudo` — **sudo** zasadniczo oznacza „zrób jako superużytkownik" — więc powyższa instrukcja została wykonana z przywilejami użytkownika *root* (administratora). Wymaga to podania hasła użytkownika, który wydaje polecenie. Ponadto użytkownik musi być poprawnie wprowadzony do pliku konfiguracyjnego */etc/sudoers*. W większości dystrybucji identyfikator użytkownika określony podczas instalacji systemu operacyjnego jest automatycznie dołączany do tego pliku. Innych użytkowników lub grupy można dodać poleceniem `visudo`.

  Dlaczego użyto `sudo`? Instalowanie oprogramowania, zmienianie parametrów sieci i inne operacje systemowe wymagają podniesionych uprawnień — byłoby niedobrze, gdyby w systemie korporacyjnym z wieloma użytkownikami osoby niebędące administratorami mogły dokonywać takich zmian.

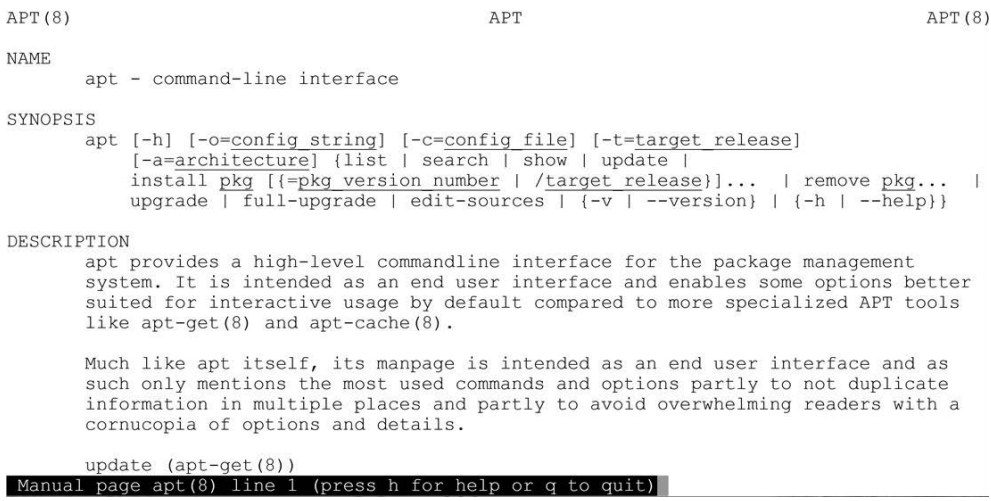
  Skoro więc `sudo` jest takie świetne, dlaczego nie uruchamiać wszystkich programów jako *root*? Oczywiście, wszystko będzie działać poprawnie, jeśli masz przywileje superużytkownika, ale ewentualne pomyłki lub literówki mogą mieć katastrofalne konsekwencje. Ponadto, jeśli pracujesz z pewnymi przywilejami i zdarzy Ci się uruchomić jakieś złośliwe oprogramowanie, będzie ono działać z tymi samymi przywilejami, co z pewnością nie jest pożądane! Jeśli ktoś zapyta, owszem, złośliwe oprogramowanie do Linuksa definitywnie istnieje i niestety towarzyszy systemowi niemal od pierwszych dni.

- `apt`: użyto polecenia `apt` — **apt** to skrót od **Advanced Package Tool**. Narzędzie to instaluje nie tylko wybrany pakiet, ale także inne pakiety, biblioteki lub zależności wymagane do działania tego pakietu. W dodatku domyślnie pobiera wszystkie te komponenty z internetowych repozytoriów. Jest to rozwiązanie zdecydowanie szybkie i wygodne w porównaniu ze starym procesem, który wymagał zgromadzenia wszystkich zależności (w odpowiednich wersjach), a następnie zainstalowania ich we właściwej kolejności w celu uruchomienia jakiejkolwiek nowej funkcji.

  Program `apt` jest domyślnym instalatorem w Ubuntu, Debianie i powiązanych dystrybucjach, ale aplikacje do zarządzania pakietami różnią się w zależności od dystrybucji. Oprócz `apt` i odpowiedników obsługiwane jest również instalowanie z pobranych plików. Debian, Ubuntu i powiązane dystrybucje używają plików *deb*, podczas gdy wiele innych dystrybucji używa plików *rpm*. Podsumowano to w poniższej tabeli:

System operacyjny	Format pliku	Narzędzia instalacyjne
Debian	*.deb*	`apt`, `apt-cache`, `apt-get`, `dpkg`
Ubuntu	*.deb*	`apt`, `apt-cache`, `apt-get`, `dpkg`
Red Hat/CentOS	*.rpm*	`yum`, `rpm`
SUSE	*.rpm*	`zypper`, `rpm`

Zatem teraz, kiedy mamy mnóstwo nowych poleceń, którym chcielibyśmy przyjrzeć się bliżej, jak możemy uzyskać więcej informacji na ich temat? Polecenie `man` (od *manual* — podręcznik) wyświetla dokumentację większości programów i operacji w Linuksie. Stronę `man` programu `apt` można na przykład wyświetlić poleceniem `man apt`; wynik wygląda tak jak na rysunku 2.1.



Rysunek 2.1. Strona man programu apt

Kiedy będziemy przedstawiać nowe polecenia, poświęć kilka chwil na sprawdzenie ich za pomocą polecenia `man` — książka ta ma służyć Ci jako przewodnik, nie jako zamiennik rzeczywistej dokumentacji systemu operacyjnego.

Skoro wspomnieliśmy już o współczesnych i tradycyjnych narzędziach, a następnie zainstalowaliśmy starszy pakiet `net-tools`, wyjaśnijmy, czym są te polecenia i do czego służą.

Wyświetlanie informacji o interfejsie IP

Wyświetlanie informacji o interfejsie to zadanie, które często wykonuje się w linuksowej stacji roboczej, zwłaszcza jeśli jej karta sieciowa jest konfigurowana automatycznie, na przykład z wykorzystaniem protokołu **DHCP** (ang. *Dynamic Host Configuration Protocol*) lub autokonfiguracji IPv6.

Jak już wspomniano, istnieją dwa przeznaczone do tego zbiory poleceń. Polecenie `ip` pozwala wyświetlić lub skonfigurować parametry sieci w nowych systemach operacyjnych. W starszych wersjach służy do tego polecenie `ifconfig`.

Polecenie `ip` umożliwia wyświetlanie lub aktualizowanie adresów IP, tras oraz innych informacji sieciowych. Na przykład w celu wyświetlenia informacji o bieżącym adresie IP należy wydać polecenie:

```
ip address
```

Program `ip` obsługuje **uzupełnianie argumentów**, więc te same wyniki da polecenie `ip addr`, a nawet `ip a`:

```
robv@ubuntu:~$ ip ad
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
↪qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
↪default qlen 1000
    link/ether 00:0c:29:33:2d:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.182/24 brd 192.168.122.255 scope global dynamic noprefixroute
    ↪ens33
       valid_lft 6594sec preferred_lft 6594sec
    inet6 fe80::1ed6:5b7f:5106:1509/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Jak widać, nawet najprostsze polecenie czasem zwraca znacznie więcej informacji, niżbyś chciał. Na przykład wyświetlane są informacje zarówno o protokole **IP w wersji 4.** (**IPv4**), jak i IPv6 — możemy ograniczyć je tylko do wersji 4. lub 6. przez dodanie opcji `-4` albo `-6`:

```
robv@ubuntu:~$ ip -4 ad
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
↪qlen 1000
     inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
↪default qlen 1000
     inet 192.168.122.182/24 brd 192.168.122.255 scope global dynamic noprefixroute
     ↪ens33
        valid_lft 6386sec preferred_lft 6386sec
```

W wynikach tych widać, że interfejs `lo` (logiczny, wewnętrzny interfejs pętli zwrotnej) ma adres IP `127.0.0.1`, a interfejs ethernetowy `ens33` ma adres IP `192.168.122.182`.

Teraz jest doskonały moment, żeby wpisać `man ip` i przejrzeć różne operacje, które można wykonać za pomocą tego polecenia (patrz rysunek 2.2).

```
robv@ubuntu: /sbin                                                    - □ ×
IP(8)                                Linux                            IP(8)

NAME
       ip - show / manipulate routing, network devices, interfaces and tunnels

SYNOPSIS
       ip [ OPTIONS ] OBJECT { COMMAND | help }

       ip [ -force ] -batch filename

       OBJECT := { link | address | addrlabel | route | rule | neigh | ntable
                 | tunnel | tuntap | maddress | mroute | mrule | monitor | xfrm
                 | netns | l2tp | tcp_metrics | token | macsec }

       OPTIONS := { -V[ersion] | -h[uman-readable] | -s[tatistics] |
                 -d[etails] | -r[esolve] | -iec | -f[amily] { inet | inet6 |
                 link } | -4 | -6 | -I | -D | -B | -0 | -l[oops] { maximum-addr-
                 flush-attempts } | -o[neline] | -rc[vbuf] [size] | -t[imestamp]
                 | -ts[hort] | -n[etns] name | -N[umeric] | -a[ll] | -c[olor] |
                 -br[ief] | -j[son] | -p[retty] }

OPTIONS
       -V, -Version
```

Rysunek 2.2. Strona man programu ip

Polecenie `ifconfig` ma funkcje bardzo podobne do polecenia `ip`, ale, jak wspomniano, zwykle spotyka się je w starszych wersjach Linuksa. Tradycyjne polecenia rosły organicznie, a nowe funkcje „doczepiano" do nich w miarę potrzeb. Doprowadziło to do sytuacji, w której im bardziej skomplikowane rzeczy wyświetla się lub konfiguruje, tym mniej spójna staje się składnia. Nowocześniejsze polecenia zaprojektowano od podstaw pod kątem spójności.

Powtórzmy naszą operację, korzystając z tradycyjnego polecenia; aby wyświetlić adres IP interfejsu, po prostu wpisz `ifconfig`:

```
robv@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1400
        inet 192.168.122.22 netmask 255.255.255.0 broadcast 192.168.122.255
        inet6 fe80::1ed6:5b7f:5106:1509 prefixlen 64 scopeid 0x20<link>
        ether 00:0c:29:33:2d:05 txqueuelen 1000 (Ethernet)
        RX packets 161665 bytes 30697457 (30.6 MB)
        RX errors 0 dropped 910 overruns 0 frame 0
        TX packets 5807 bytes 596427 (596.4 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0
collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 1030 bytes 91657 (91.6 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1030 bytes 91657 (91.6 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0
collisions 0
```

Jak widać, wyświetlane są mniej więcej te same informacje w nieco innym formacie. Jeśli przejrzysz strony `man` obu poleceń, zauważysz, że opcje `ip` są bardziej spójne, a `ifconfig` w ograniczonym stopniu obsługuje IPv6 — na przykład nie pozwala wyświetlić tylko informacji o IPv4 lub IPv6.

Wyświetlanie informacji o trasach

W zbiorze współczesnych poleceń sieciowych do wyświetlania informacji o trasach służy ten sam program `ip`. Jak można się było spodziewać, odpowiednie polecenie to `ip route`, które można skrócić nawet do `ip r`:

```
robv@ubuntu:~$ ip route
default via 192.168.122.1 dev ens33 proto dhcp metric 100
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.122.0/24 dev ens33 proto kernel scope link src 192.168.122.156  metric 100

robv@ubuntu:~$ ip r
default via 192.168.122.1 dev ens33 proto dhcp metric 100
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.122.0/24 dev ens33 proto kernel scope link src 192.168.122.156  metric 100
```

Wyniki te pokazują, że mamy *trasę domyślną* wskazującą adres `192.168.122.1`. Trasa domyślna działa dokładnie tak, jak sugeruje jej nazwa — jeśli pakiet jest zaadresowany do miejsca przeznaczenia, którego nie ma w tabeli tras, host wyśle go do swojej bramy domyślnej. Tabela tras zawsze preferuje trasę „najbardziej konkretną" — taką która najlepiej pasuje do docelowego adresu IP. Jeśli nie da się dopasować adresu, najbardziej konkretna jest trasa `0.0.0.0 0.0.0.0`, prowadząca do bramy domyślnej (innymi słowy, jest to trasa oznaczająca „jeśli pakiet nie pasuje do niczego innego"). Host zakłada, że adres IP bramy domyślnej należy do routera, który będzie wiedział (oby!), gdzie dalej przesłać ten pakiet.

Widzimy też trasę do adresu `169.254.0.0/16`. Według definicji w dokumencie RFC 3927 jest to tak zwany **adres lokalny dla łącza** (**LLA**, ang. *Link-Local Address*). **RFC** (skrót od ang. *Request for Comment* — prośba o komentarze) to dokumenty używane w nieformalnym procesie recenzowania rozwijanych standardów internetowych. Lista opublikowanych dokumentów RFC jest dostępna w witrynie **IETF** (ang. *Internet Engineering Task Force*) pod adresem *https://www.ietf.org/standards/rfcs/*.

Adresy LLA działają tylko w bieżącej podsieci — jeśli host nie ma statycznie skonfigurowanego adresu IP, a protokół DHCP nie przydzieli mu adresu, host użyje dwóch pierwszych oktetów zdefiniowanych w dokumencie RFC (`169.254`), a następnie w sposób pseudolosowy wybierze dwa ostatnie oktety. Po teście ping/ARP (protokół ARP omówimy w rozdziale 3., „Używanie Linuksa i linuksowych narzędzi do diagnostyki sieci"), który sprawdza, czy ten obliczony adres jest rzeczywiście dostępny, host jest gotowy do komunikacji. Adres ten jest przeznaczony tylko do komunikacji z innymi adresami LLA w tym samym segmencie sieci, zwykle z wykorzystaniem protokołów rozgłoszeniowych i multiemisyjnych, takich jak ARP i Alljoyn, w celu „odnalezienia" siebie nawzajem. Dla jasności: z adresów tych niemal nigdy nie korzysta się w rzeczywistych sieciach; są one używane tylko wtedy, gdy nie ma absolutnie

żadnej alternatywy. A żeby jeszcze bardziej namieszać nam w głowach, Microsoft określa je innym terminem — **APIPA** (ang. *Automatic Private Internet Protocol Addressing*).

Wreszcie widzimy trasę do lokalnej podsieci, w tym przypadku `192.168.122.0/24`. Jest ona nazywana **trasą połączoną** (ponieważ jest połączona z tym interfejsem). Z perspektywy hosta oznacza to, że nie potrzebuje on żadnego routingu do komunikowania się z innymi hostami w jego własnej podsieci.

Taki zestaw tras jest bardzo często spotykany w prostych sieciach — brama domyślna, segment lokalny, i to wszystko. W wielu systemach operacyjnych nie zobaczysz podsieci `169.254.0.0`, chyba że host rzeczywiście używa adresu LLA.

Jeśli chodzi o polecenia tradycyjne, jest wiele sposobów wyświetlania bieżącego zestawu tras. Najczęściej używa się polecenia `netstat -rn` (*network status* — stan sieci; `-r` — pokaż trasy, `-n` — wyniki w postaci liczbowej). Istnieje jednak również oddzielne polecenie `route` (dlaczego — dowiesz się dalej w tym rozdziale):

```
robv@ubuntu:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS  Window  irtt Iface
0.0.0.0         192.168.122.1   0.0.0.0         UG        0  0          0 ens33
169.254.0.0     0.0.0.0         255.255.0.0     U         0  0          0 ens33
192.168.122.0   0.0.0.0         255.255.255.0   U         0  0          0 ens33

robv@ubuntu:~$ route -n
Destination     Gateway           Genmask         Flags Metric Ref    Use  Iface
0.0.0.0         192.168.122.0     0.0.0.0         UG    100    0        0  ens33
169.254.0.0     0.0.0.0           255.255.0.0     U     1000   0        0  ens33
192.168.122.0   0.0.0.0           255.255.255.0   U     100    0        0  ens33
```

Wyświetlane są te same informacje, ale teraz mamy dwa dodatkowe polecenia — `netstat` i `route`. Zbiór tradycyjnych poleceń zwykle ma oddzielne, unikatowe programy do każdego celu, a w tym przypadku widzimy dwa, których funkcje w dużej mierze się pokrywają. Nauczenie się tych wszystkich poleceń i zapamiętanie ich zróżnicowanej składni może być trudne dla początkujących użytkowników Linuksa. Zbiór poleceń `ip` bardzo ułatwia sprawę!

Bez względu na to, którego zbioru narzędzi ostatecznie będziesz używał, wiesz teraz, jak sprawdzić informacje o adresach IP i trasach, co powinno Ci pomóc w zapewnieniu podstawowej łączności dla swojego hosta.

Adresy IPv4 i maski podsieci

W poprzednim rozdziale krótko wspomnieliśmy o adresach IP, ale porozmawiajmy o nich nieco bardziej szczegółowo. Protokół IPv4 pozwala unikatowo zaadresować każde urządzenie w *podsieci* przez przypisanie mu adresu oraz maski podsieci. W naszym przykładzie adres IPv4 to `192.168.122.182`. Każdy *oktet* adresu IPv4 może mieć wartość od `0` do `255`, a maska

podsieci to /24, co często zapisuje się również jako 255.255.255.0. Wydaje się to skomplikowane, dopóki nie przekształcimy liczb w reprezentację binarną. 255 w formacie binarnym to 11111111 (8 bitów), a w trzech takich grupach są 24 bity. Zatem adres i maska mówią nam, że po zamaskowaniu część reprezentująca sieć to 192.168.122.0, a część reprezentująca hosta (która może przybierać wartości od 1 do 254) to 182.

Rozpiszmy to:

Adres	192	168	122	182
Adres w formacie binarnym	11000000	10101000	01111010	10110110
Maska w formacie binarnym	11111111	11111111	11111111	00000000

A gdybyśmy potrzebowali większej podsieci? Możemy po prostu przesunąć maskę o kilka bitów w lewo. Na przykład w przypadku maski 20-bitowej otrzymujemy:

Adres	192	168	122	182
Adres w formacie binarnym	11000000	10101000	01111010	10110110
Maska w formacie binarnym	11111111	11111111	11110000	00000000

Trzeci oktet maski jest teraz równy 0b11110000 (zauważ prefiks 0b, oznaczający „liczbę binarną"), co jest równoważne liczbie 240 w zapisie dziesiętnym. Zwiększa to zakres adresów dla naszych hostów do 0 - 15 (0 - 0b1111) w trzecim oktecie oraz 0 - 255 (0 - 0b11111111) w czwartym oktecie, co razem daje $15 \cdot 255 - 1 = 3824$ adresy (w następnym podrozdziale wyjaśnimy, skąd to –1).

Jak widzisz, aplikacja kalkulatora, która konwertuje liczby binarne na dziesiętne i odwrotnie, może być bardzo przydatna dla specjalisty ds. sieci! Upewnij się, że obsługuje również liczby szesnastkowe (o podstawie 16); zajmiemy się nimi za kilka minut.

Teraz, kiedy już wiesz, jak pracować z adresami i maskami podsieci w formacie dziesiętnym, a zwłaszcza binarnym, rozwińmy te informacje i zbadajmy, jak wykorzystać je do zilustrowania innych koncepcji adresowania.

Adresy specjalnego przeznaczenia

Istnieje kilka adresów *specjalnego przeznaczenia*, które musimy omówić, abyś zrozumiał, jak działają adresy IP w lokalnej podsieci. Po pierwsze, jeśli wszystkie bity hosta w adresie są ustawione na 1, taki adres nazywamy **adresem rozgłoszeniowym** (ang. *broadcast address*). Jeśli wyślesz dane na adres rozgłoszeniowy, zostaną one odczytane przez wszystkie interfejsy sieciowe w podsieci.

Zatem adres rozgłoszeniowy dla przykładowej sieci /24 wyglądałby tak:

Adres	192	168	122	182
Adres w formacie binarnym	11000000	10101000	01111010	10110110
Maska w formacie binarnym	11111111	11111111	11111111	00000000
Adres rozgłoszeniowy	11000000	10101000	01111010	11111111

Innymi słowy, nasz adres rozgłoszeniowy to 192.168.122.255.

Adres rozgłoszeniowy dla sieci /20 wygląda następująco:

Adres	192	168	122	182
Adres w formacie binarnym	11000000	10101000	01111010	10110110
Maska w formacie binarnym	11111111	11111111	11110000	00000000
Adres rozgłoszeniowy	11000000	10101000	01111111	11111111

Możemy przekształcić go z powrotem w format dziesiętny i otrzymać 192.168.127.255.

Przesuwanie granicy między bitami sieci a bitami hosta w adresie IPv4 przywodzi na myśl koncepcję **klas adresów**. Po przekształceniu w postać binarną kilka pierwszych bajtów definiuje tak zwaną **klasową** maskę podsieci dla adresu. W większości systemów operacyjnych, kiedy ustawiasz adres IP w graficznym interfejsie użytkownika, ta klasowa maska podsieci jest często wprowadzana automatycznie. Relacja między klasami a maskami podsieci przedstawia się następująco:

Klasa	Początkowe bity adresu	Maska podsieci (bity)	Maska podsieci (dziesiętna)	Pierwszy adres w zakresie	Ostatni adres w zakresie
Klasa A	0	/8	255.0.0.0	0.0.0.0	127.255.255.255
Klasa B	10	/16	255.255.0.0	128.0.0.0	191.255.255.255
Klasa C	110	/24	255.255.255.0	192.0.0.0	223.255.255.255
Klasa D (adresy multiemisyjne)	1110	nd.	nd.	224.0.0.0	239.255.255.255
Klasa E (zarezerwowana, nieużywana)	1111	nd.	nd.	240.0.0.0	255.255.255.255

Powyższa tabela definiuje domyślne klasowe maski podsieci. Przyjrzymy się temu bliżej w następnych dwóch podrozdziałach.

Z tego wszystkiego łatwo wywnioskować, dlaczego administratorzy zwykle używają **klasowych granic** podsieci w swoich organizacjach. Zdecydowana większość wewnętrznych podsieci ma maskę `255.255.255.0` lub `255.255.0.0`. Każdy inny wybór prowadzi do zamieszania i grozi błędami w konfiguracji serwerów lub stacji roboczych, kiedy do zespołu dołącza nowa osoba. Poza tym większość osób nie ma ochoty wykonywać obliczeń za każdym razem, kiedy trzeba ustawić lub zinterpretować adres sieciowy.

Drugi typ specjalnego adresu, o którym przed chwilą napomknęliśmy, to adres **multiemisyjny** (ang. *multicast address*). Adres ten umożliwia komunikację między kilkoma urządzeniami jednocześnie. Możesz na przykład użyć adresu multiemisyjnego, aby wysłać identyczny strumień wideo do kilku ekranów podłączonych do sieci albo poprowadzić połączenie konferencyjne w aplikacji głosowej lub wideo. Lokalne adresy multiemisyjne mają następującą postać:

Możliwe wartości binarne	11100000	00000000	00000xxx	xxxxxxxx
Możliwe wartości dziesiętne	224	0	Dowolna liczba od 1 do 8	Dowolna liczba od 1 do 255

Ostatnie 11 bitów (3 + 8) zwykle tworzy „dobrze znane adresy" różnych protokołów multiemisji. Niektóre często używane adresy multiemisyjne to:

`224.0.0.1`	Wszystkie hosty w podsieci
`224.0.0.2`	Wszystkie routery w podsieci
`224.0.0.12`	Urządzenia uczestniczące w protokole VRRP
`224.0.0.18`	Serwery DHCP i przekaźniki DHCP
`224.0.0.102`	Urządzenia uczestniczące w protokole **HSRP** (ang. *Hot Standby Router Protocol*)
`224.0.1.1`	Wszystkie serwery **NTP** (ang. *Network Time Protocol*)
`224.0.0.113`	Hosty Alljoyn (używane przez Windows do odkrywania sąsiednich urządzeń)

Pełna lista dobrze znanych, zarejestrowanych adresów multiemisyjnych jest dostępna w witrynie organizacji **IANA** (ang. *Internet Assigned Numbers Authority*) pod adresem *https://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml*. Choć wydaje się ona kompletna, producenci często definiują własne adresy multiemisyjne w tej przestrzeni adresowej.

To tylko krótkie wprowadzenie do adresowania multiemisyjnego — zagadnienie to jest znacznie bardziej skomplikowane, do tego stopnia, że projektowaniu, wdrażaniu i teorii sieci multiemisyjnych poświęcono całe książki. To, co omówiliśmy, powinno jednak dać Ci ogólne pojęcie i wystarczyć na początek.

Po omówieniu adresów rozgłoszeniowych i multiemisyjnych zajmijmy się „rodzinami" adresów IP, które najprawdopodobniej są używane w Twoim środowisku.

Adresy prywatne — RFC 1918

Innym zbiorem adresów specjalnych jest przestrzeń adresowa RFC 1918. Dokument RFC 1918 zawiera listę podsieci IP przydzielonych do wewnętrznego użytku organizacji. Adresy te są ignorowane w publicznym internecie, więc jeśli mają być używane do komunikacji przez internet, konieczne jest „tłumaczenie" ich z wykorzystaniem **translacji adresów sieciowych** (**NAT**, ang. *Network Address Translation*).

Adresy RFC1918 są następujące:

- `10.0.0.0/8` (klasa A),
- od `172.16.0.0` do `172.31.0.0/16` (klasa B) — można je podsumować jako `172.16.0.0/12`,
- `192.168.0.0/16` (klasa C).

Adresy te zapewniają organizacjom dużą przestrzeń adresową IP do użytku wewnętrznego, która nie koliduje z żadnymi adresami w publicznym internecie.

Interesującym ćwiczeniem będzie użycie tych podsieci RFC 1918 do zweryfikowania domyślnej klasy adresów przez przetłumaczenie pierwszego oktetu na postać binarną i porównanie go z tabelą w poprzednim podrozdziale.

Pełną specyfikację RFC 1918 można znaleźć pod adresem *https://tools.ietf.org/html/rfc1918*.

Teraz, kiedy omówiliśmy binarne aspekty adresowania IP i maski podsieci, a także różne specjalne grupy adresów IP, zapewne masz już dość teorii oraz matematyki i chcesz jak najszybciej wrócić do eksperymentowania w wierszu poleceń swojego komputera. Zła wiadomość jest taka, że musimy jeszcze omówić szczegóły adresowania IPv6 (wersji 6. protokołu IP). Dobra zaś taka, że zaczekamy z tym do dodatku, więc będziesz mógł szybciej wrócić do klawiatury!

Skoro umiesz już wyświetlać parametry i rozumiesz, jak działa adresowanie IP, skonfigurujmy interfejs IP do użytku.

Przypisywanie adresu IP do interfejsu

Przypisywanie stałego adresu IPv4 do interfejsu to coś, co będziesz robił w niemal każdym samodzielnie zbudowanym serwerze. Na szczęście jest to dość proste. W nowym zbiorze poleceń użyjemy do tego programu `nmcli` (ang. *Network Manager Command Line* — **wiersz poleceń menedżera sieci**). Ustawimy adres IP, bramę domyślną i serwer DNS. Wreszcie

ustawimy tryb adresowania na `manual` (ręczny). Zacznijmy od wyświetlenia połączeń sieciowych w formacie `nmcli`:

```
robv@ubuntu:~$ sudo nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  02ea4abd-49c9-3291-b028-7dae78b9c968  ethernet  ens33
```

Nasza nazwa połączenia to `Wired connection 1`. Nie musimy jednak wpisywać jej w całości za każdym razem; wystarczy, że napiszemy `Wi` i naciśniemy klawisz *Tab*, aby uzupełnić nazwę. Warto też wiedzieć, że `ncmli` obsługuje skrócone nazwy argumentów, więc możemy pisać `mod` zamiast `modify`, `con` zamiast `connection` itd. Kontynuujmy naszą sekwencję poleceń (zauważ skrócone argumenty w ostatnim poleceniu):

```
$ sudo nmcli connection modify "Wired connection 1" ipv4.addresses 192.168.122.22/24
$
$ sudo nmcli connection modify "Wired connection 1" ipv4.gateway 192.168.122.1
$
$ sudo nmcli connection modify "Wired connection 1" ipv4.dns "8.8.8.8"
$
$ sudo nmcli con mod "Wired connection 1" ipv4.method manual
$
```

Teraz zapiszmy zmiany i uaktywnijmy je:

```
$ sudo nmcli connection up "Wired connection 1"
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/5)
$
```

W tradycyjnym podejściu wszystkie zmiany wprowadza się przez edycję plików. A żeby było ciekawiej, nazwy plików i ich lokalizacje bywają różne w różnych dystrybucjach. Tutaj przedstawimy najbardziej typowe modyfikacje i pliki.

Aby zmienić serwer DNS, otwórz w edytorze plik */etc/resolv.conf* i zmodyfikuj wiersz `nameserver` tak, aby odzwierciedlał adres IP wybranego serwera:

```
nameserver 8.8.8.8
```

Aby zmienić adres IP, maskę podsieci itd., otwórz w edytorze plik */etc/sysconfig/network-scripts/ifcfg-eth0* i zmodyfikuj wartości w następujący sposób:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETMASK=255.255.255.0
IPADDR=10.0.1.27
```

Jeśli ten interfejs jest połączony z bramą domyślną, możesz dodać wiersz:

```
GATEWAY=192.168.122.1
```

Warto jeszcze raz przypomnieć, że w różnych dystrybucjach pliki do edycji mogą być inne, a **podejście to nie jest kompatybilne wstecz**. W większości nowoczesnych dystrybucji zmienianie ustawień sieci przez edycję plików nie działa.

Teraz, kiedy już wiesz, jak przypisać adres IP do interfejsu, nauczysz się modyfikować trasy w swoim hoście.

Dodawanie trasy

Aby dodać tymczasową trasę statyczną, ponownie sięgniemy po polecenie `ip`. W tym przykładzie informujemy hosta, że ma kierować pakiety na adres `192.168.122.10`, aby dostać się do sieci `10.10.10.0/24`:

```
robv@ubuntu:~$ sudo ip route add 10.10.10.0/24 via 192.168.122.10
[sudo] password for robv:
robv@ubuntu:~$ ip route
default via 192.168.122.1 dev ens33 proto dhcp metric 100
10.10.10.0/24 via 192.168.122.10 dev ens33
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.122.0/24 dev ens33 proto kernel scope link src 192.168.122.156 metric 100
```

Możesz też dodać przeznaczony do tego wyjściowy interfejs sieciowy przez dołączenie `dev <nazwa urządzenia>` na końcu tego polecenia `ip route add`.

Jest to jednak tylko trasa tymczasowa, która zniknie, jeśli host albo procesy sieciowe zostaną uruchomione ponownie. Trwałą trasę statyczną można dodać za pomocą polecenia `nmcli`. Najpierw wyświetlmy połączenia sieciowe w formacie `nmcli`:

```
robv@ubuntu:~$ sudo nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  02ea4abd-49c9-3291-b028-7dae78b9c968  ethernet  ens33
```

Teraz za pomocą `nmcli` dodajmy do połączenia `Wired connection 1` trasę, która prowadzi do sieci `10.10.11.0/24` przez adres `192.168.122.11`:

```
robv@ubuntu:~$ sudo nmcli connection modify "Wired connection 1" +ipv4.routes
"10.10.11.0/24 192.168.122.11"
```

Zapiszmy dokonane zmiany poleceniem `nmcli`:

```
$ sudo nmcli connection up "Wired connection 1"
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/5)
$
```

Jeśli teraz przyjrzymy się tabeli tras, zobaczymy obie nasze trasy statyczne:

```
robv@ubuntu:~$ ip route
default via 192.168.122.1 dev ens33 proto dhcp metric 100 10.10.10.0/24 via
192.168.122.10 dev ens33
10.10.11.0/24 via 192.168.122.11 dev ens33 proto static metric 100
```

```
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.122.0/24 dev ens33 proto kernel scope link src 192.168.122.156  metric 100
```

Jeśli jednak ponownie uruchomimy komputer, przekonamy się, że trasa tymczasowa zniknęła, a trasa trwała nadal jest obecna:

```
robv@ubuntu:~$ ip route
default via 192.168.122.1 dev ens33 proto dhcp metric 100
10.10.11.0/24 via 192.168.122.11 dev ens33 proto static metric 100
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.122.0/24 dev ens33 proto kernel scope link src 192.168.122.156  metric 100
```

Znasz już podstawy dodawania tras. Przyjrzyjmy się teraz, jak wykonać to samo zadanie w starszym hoście linuksowym za pomocą tradycyjnych poleceń `route`.

Dodawanie trasy tradycyjnym sposobem

Aby tymczasowo dodać trasę, użyj polecenia:

```
$ sudo route add -net 10.10.12.0 netmask 255.255.255.0 gw 192.168.122.12
```

Trwałe dodawanie trasy jest jednak bardziej skomplikowane — trwałe trasy są przechowywane w plikach, których nazwy i lokalizacje zależą od dystrybucji. Właśnie dlatego spójność poleceń `iproute2/nmcli` tak bardzo ułatwia pracę we współczesnych systemach.

W starszych dystrybucjach Debian/Ubuntu zwykle edytuje się plik */etc/network/interfaces* i dodaje następujący wiersz:

```
up route add -net 10.10.12.0 netmask 255.255.255.0 gw 192.168.122.12
```

W starszych dystrybucjach z rodziny Red Hat edytuje się plik */etc/sysconfig/network- scripts/route-<nazwa urządzenia>* i dodaje następujący wiersz:

```
10.10.12.0/24 via 192.168.122.12
```

Można też po prostu dodać trasy za pomocą poleceń `route` umieszczonych w pliku */etc/rc.local* — ten sposób zadziała właściwie w każdym systemie linuksowym, ale jest uważany za mało elegancki, ponieważ plik ten jest ostatnim miejscem, gdzie następny administrator będzie szukał odpowiedniego ustawienia (bo nie jest to plik, w którym typowo zapisuje się ustawienia sieci). Plik *rc.local* jest uruchamiany podczas rozruchu systemu i wykonuje dowolne polecenia, które są w nim zapisane. W tym przypadku może to być nasze polecenie `route add`:

```
/sbin/route add -net 10.10.12.0 netmask 255.255.255.0 gw 192.168.122.12
```

W tym momencie jesteśmy już bliscy chwili, gdy w pełni skonfigurujemy łączność sieciową w naszym hoście. Ustawiliśmy adres IP, maskę podsieci i trasy. Jednak często, zwłaszcza podczas rozwiązywania problemów albo wstępnej konfiguracji, trzeba włączyć lub wyłączyć interfejs; zajmiemy się tym w następnym punkcie.

Wyłączanie i włączanie interfejsu

W świecie nowych poleceń do włączania i wyłączania interfejsu służy — zgadłeś — program `ip`. Poniżej „przeładowujemy" interfejs przez jego wyłączenie i ponowne włączenie:

```
robv@ubuntu:~$ sudo ip link set ens33 down
robv@ubuntu:~$ sudo ip link set ens33 up
```

W starym zbiorze poleceń interfejs włącza się lub wyłącza poleceniem `ipconfig`:

```
robv@ubuntu:~$ sudo ifconfig ens33 down
robv@ubuntu:~$ sudo ifconfig ens33 up
```

Podczas wykonywania poleceń związanych z interfejsami zawsze pamiętaj, żeby nie *podcinać gałęzi, na której siedzisz*. Jeśli jesteś zalogowany zdalnie (na przykład za pomocą `ssh`) i zmienisz adresowanie IP lub trasy albo wyłączysz interfejs, możesz utracić połączenie z hostem.

Omówiliśmy już większość zadań, które będziesz musiał wykonywać, aby skonfigurować swój system do pracy w nowoczesnej sieci. Ważną częścią administrowania siecią jest jednak również diagnozowanie jej działania i dobieranie ustawień do specjalnych przypadków, na przykład regulowanie ustawień w celu zoptymalizowania ruchu, kiedy potrzebne są mniejsze lub większe rozmiary pakietów.

Ustawianie jednostki MTU interfejsu

Jedną z operacji, którą coraz częściej wykonuje się w nowoczesnych systemach, jest ustawianie **maksymalnej jednostki transmisji** (**MTU**, ang. *Maximum Transmission Unit*). Jest to rozmiar największej **jednostki danych protokołu** (**PDU**, ang. *Protocol Data Unit* — w większości sieci nazywanej **ramką**), którą interfejs będzie wysyłać lub odbierać. W Ethernecie domyślną jednostką MTU jest 1500 bajtów, co przekłada się na maksymalny rozmiar pakietu równy 1500 bajtom. Maksymalny rozmiar pakietu w danym nośniku jest zwykle nazywany **maksymalnym rozmiarem segmentu** (**MSS**, ang. *Maximum Segment Size*). W przypadku Ethernetu te trzy parametry mają wartości przedstawione w tabeli 2.1.

Tabela 2.1. Relacja między rozmiarem ramki, MTU, rozmiarem pakietu i MSS dla Ethernetu

Maksymalny rozmiar ramki	1518 bajtów
MTU — maksymalny rozmiar danych w ramce	1500 bajtów
Maksymalny rozmiar pakietu (taki sam jak MTU, ponieważ pakiet jest danymi użytkowymi ramki)	1500 bajtów
MSS — maksymalny rozmiar danych użytkowych jednego pakietu	1460 bajtów

Dlaczego miałbyś je zmieniać? 1500 to dobry kompromis, ponieważ pakiet jest na tyle mały, że w razie błędu można szybko wykryć problem, a ilość danych do ponownego przesłania jest względnie mała. Istnieje jednak kilka wyjątków, zwłaszcza w centrach danych.

W przypadku ruchu związanego z pamięcią masową, w szczególności iSCSI, pożądane są większe rozmiary ramki, żeby pakiet mógł pomieścić więcej danych. Wówczas MTU zwykle ustawia się na mniej więcej 9000 (co często określa się mianem **pakietu jumbo**). Sieci te zazwyczaj działają z prędkością 1 Gb/s, 10 Gb/s lub wyższą. Większe pakiety pojawiają się też w ruchu związanym z tworzeniem kopii zapasowych lub migracją maszyn wirtualnych (na przykład VMotion w VMware lub Live Migration w Hyper-V).

Na drugim końcu spektrum często zdarzają się sytuacje, w których potrzebny jest mniejszy rozmiar pakietu. Jest to szczególnie ważne, ponieważ nie wszystkie hosty potrafią to poprawnie wykrywać, a wiele aplikacji ustawia w swoim ruchu bit **DF** (ang. *Don't Fragment* — **nie fragmentuj**). Pakiet o rozmiarze 1500 bajtów z ustawionym bitem DF może wówczas pojawić się w nośniku, który obsługuje tylko pakiety o rozmiarze na przykład 1380 bajtów — w takim przypadku aplikacja po prostu zawiedzie, a komunikaty o błędach nie pomogą rozwiązać problemu. W jakich okolicznościach może tak się zdarzyć? Przykładowo na dowolnym łączu z kapsułkowaniem pakietów, takim jak tunel lub sieć VPN. Rozmiar ramki (i w konsekwencji rozmiar pakietu) jest tu ograniczony przez dodatkowe dane związane z kapsułkowaniem, których rozmiar zwykle dość łatwo obliczyć. Innym typowym przykładem są łącza satelitarne. Mają one często ramki o domyślnym rozmiarze 512 bajtów — w takich przypadkach rozmiary są publikowane przez dostawców usług.

Ustawianie MTU jest dokładnie tak proste, jak można się było spodziewać — ponownie użyjemy do tego polecenia `nmcli`. Zauważ, że w tym przykładzie skracamy argumenty wiersza polecenia `nmcli`, a na koniec zapisujemy zmiany w konfiguracji — jednostka MTU jest zmieniana natychmiast po ostatnim poleceniu. Ustawmy MTU na `9000`, aby zoptymalizować ruch iSCSI:

```
$ sudo nmcli con mod "Wired connection 1" 802-3-ethernet.mtu 9000
$ sudo nmcli connection up "Wired connection 1"
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/5)
$
```

Co jeszcze można zrobić za pomocą `nmcli` po ustawieniu MTU?

Więcej o poleceniu nmcli

Program `nmcli` można również uruchomić w trybie interaktywnym, aby dokonywać zmian w interpreterze czasu rzeczywistego, czyli powłoce. Aby uruchomić tę powłokę dla interfejsu ethernetowego, wydaj polecenie `nmcli connection edit type ethernet`. Po uruchomieniu powłoki możesz wydać polecenie `print`, aby wyświetlić listę wszystkich parametrów, które można zmienić dla tego typu interfejsu. Zauważ, że parametry są podzielone na logiczne grupy — zredagowaliśmy tę (bardzo długą) listę, aby pokazać ustawienia, które prawdopodobnie będziesz musiał regulować, modyfikować lub diagnozować w różnych sytuacjach:

```
nmcli> print
===============================================================================
                    Connection profile details (ethernet)
```

```
===============================================================================
connection.id:                          ethernet
connection.uuid:                        e0b59700-8dcb-4801-9557-9dee5ab7164f
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              --
...
connection.lldp:                        -1 (default)
connection.llmnr:                       -1 (default)
-------------------------------------------------------------------------------
```

Oto często używane opcje połączenia ethernetowego:

```
802-3-ethernet.port:                    --
802-3-ethernet.speed:                   0
802-3-ethernet.duplex:                  --
802-3-ethernet.auto-negotiate:          no
802-3-ethernet.mac-address:             --
802-3-ethernet.mtu:                     auto
...
802-3-ethernet.wake-on-lan:             default
802-3-ethernet.wake-on-lan-password:    --
-------------------------------------------------------------------------------
```

Oto często używane opcje IPv4:

```
ipv4.method:                            auto
ipv4.dns:                               --
ipv4.dns-search:                        --
ipv4.dns-options:                       --
ipv4.dns-priority:                      0
ipv4.addresses:                         --
ipv4.gateway:                           --
ipv4.routes:                            --
ipv4.route-metric:                      -1
ipv4.route-table:                       0 (unspec)
ipv4.routing-rules:                     --
ipv4.ignore-auto-routes:                no
ipv4.ignore-auto-dns:                   no
ipv4.dhcp-client-id:                    --
ipv4.dhcp-iaid:                         --
ipv4.dhcp-timeout:                      0 (default)
ipv4.dhcp-send-hostname:                yes
ipv4.dhcp-hostname:                     --
ipv4.dhcp-fqdn:                         --
ipv4.dhcp-hostname-flags:               0x0 (none)
ipv4.never-default:                     no
ipv4.may-fail:                          yes
ipv4.dad-timeout:                       -1 (default)
-------------------------------------------------------------------------------
```

(Tutaj następowałyby opcje IPv6, ale je usunęliśmy, aby zwiększyć czytelność listingu).
Oto ustawienia proxy:

```
------------------------------------------------------------------------------
proxy.method:                           none
proxy.browser-only:                     no
proxy.pac-url:                          --
proxy.pac-script:                       --
------------------------------------------------------------------------------
```

Jak już wspomniano, powyższy listing jest nieco skrócony. Pokazaliśmy ustawienia, które najczęściej sprawdza się lub zmienia podczas konfigurowania i diagnozowania połączeń sieciowych. Aby zobaczyć pełny listing, uruchom polecenie we własnym komputerze.

Jak pokazaliśmy, polecenie `nmcli` pozwala zmienić kilka parametrów interfejsu sieciowego albo interaktywnie, albo z poziomu wiersza poleceń. Interfejs wiersza poleceń umożliwia określanie ustawień sieciowych w skryptach, co jest bardziej skalowalnym sposobem, pozwalającym na modyfikowanie parametrów dziesiątek, setek, a nawet tysięcy stacji jednocześnie.

Podsumowanie

Po przeczytaniu tego rozdziału powinieneś rozumieć adresowanie IP z perspektywy binarnej. Wiesz, jak działają podsieci i maski, a także adresowanie rozgłoszeniowe i multiemisyjne. Poznałeś też różne klasy adresów IP. Dzięki tej wiedzy będziesz umiał wyświetlić lub ustawić adresy IP i trasy w hoście linuksowym za pomocą wielu różnych poleceń. Łatwo zmienisz też inne parametry interfejsu sieciowego, takie jak jednostka MTU.

Po opanowaniu tych umiejętności jesteś dobrze przygotowany na nasz następny temat: używanie Linuksa i narzędzi linuksowych do diagnozowania sieci.

Pytania

Oto lista pytań, które pozwolą Ci sprawdzić swoją wiedzę na temat materiału omawianego w tym rozdziale. Odpowiedzi znajdziesz w dodatku A, „Sprawdziany wiadomości”:

1. Do czego służy brama domyślna?
2. Jaka jest maska podsieci i adres rozgłoszeniowy sieci 192.168.25.0/24?
3. W jaki sposób w tej samej sieci używany jest adres rozgłoszeniowy?
4. Jakie są w tej samej sieci możliwe adresy hostów?
5. Gdybyś musiał statycznie ustawić szybkość i tryb dupleksowy interfejsu ethernetowego, jakiego polecenia byś użył?

Dalsza lektura

- RFC 1918 — Address Allocation for Private Internets: *https://tools.ietf.org/html/rfc1918*
- RFC 791 — Internet Protocol: *https://tools.ietf.org/html/rfc791*

Skorowidz

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

Ś

T

U

V

W

Y

Z

Ź

Ż

Linux: korzystaj z najwyższych standardów bezpieczeństwa!

Linux zdobył popularność zarówno wśród użytkowników, jak i administratorów sieci i zaskarbił sobie ich uznanie. Stało się tak nie bez powodu, ponieważ pozwala on na uzyskanie imponującej elastyczności usług sieciowych przy relatywnie niewielkich kosztach. Usługi sieciowe Linuksa mogą zapewnić funkcjonalność niedostępną w przypadku innych systemów. Dzięki nim można stworzyć solidnie zabezpieczone, efektywne i doskonale dopasowane do szczególnych potrzeb organizacji środowisko sieciowe. Wystarczy dobrze poznać i zrozumieć działanie poszczególnych usług sieciowych Linuksa.

Ta książka jest przeznaczona dla inżynierów zarządzających infrastrukturą sieciową dowolnego rodzaju. Znajdziesz tu niezbędne informacje, których potrzebujesz do uruchomienia i skonfigurowania różnych użytecznych usług sieciowych. Najpierw poznasz najważniejsze dystrybucje i podstawy konfiguracji sieci w Linuksie. Następnie przejdziesz do diagnozowania sieci, konfigurowania zapory oraz używania Linuksa jako hosta usług sieciowych. W dalszej kolejności uzyskasz informacje o przydatnych usługach i o ich wdrażaniu w środowisku korporacyjnym. Sporo miejsca w książce poświęcono zagadnieniom ochrony przed nieuprawnionym dostępem: omówiono typowe sposoby przeprowadzania ataków oraz techniki skutecznego zabezpieczania usług sieciowych. Ta publikacja dostarczy Ci przydatnych wskazówek, które pozwolą nie tylko skonfigurować potrzebne usługi sieciowe, ale także zbudować centrum danych oparte wyłącznie na Linuksie.

Najciekawsze zagadnienia:

- Linux jako platforma do diagnozowania sieci i rozwiązywania problemów
- konfiguracja zapory Linuksa
- konfiguracja usług sieciowych, w tym DNS i DHCP
- rejestrowanie zdarzeń w celu monitorowania sieci
- wdrażanie i konfiguracja systemów zapobiegania włamaniom (IPS)
- konfiguracja usługi honeypot w celu wykrywania i odpierania ataków

Rob VandenBrink jest uznanym ekspertem w dziedzinie bezpieczeństwa informacji, infrastruktury sieciowej oraz projektowania sieci i centrów danych, a także automatyzacji IT i wirtualizacji. Napisał oprogramowanie ułatwiające bezpieczne korzystanie z VPN i stworzył różnorodne narzędzia sieciowe dla Cisco IOS. Uzyskał liczne certyfikaty SANS/GIAC, VMware i Cisco.

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-9710-1

9 788328 397101

Cena: 119,00 zł

Packt