Linux i Python w Elektronicznej Sieci – ćwiczenia #10: Sieci komputerowe – ethernet, konfiguracja i programowanie usług sieciowych

Projekt „Matematyka dla Ciekawych Świata”,
Robert Ryszard Paciorek
<rrp@opcode.eu.org>

2023-07-03

Uwaga (1): Informacje teoretyczne przydatne do wykonania zadań oraz linki do prezentacji wideo znajdziesz w skrypcie wykładowym. Gorąco zachęcamy do korzystania z tych materiałów w trakcie rozwiązywania zadań.
Uwaga (2): Po skończeniu zadania prosimy pokazać rozwiązanie prowadzącemu - może mieć on jakiś wartościowy komentarz dotyczący tego rozwiązania.

Zadanie 1.0.1

Napisz polecenie lub serię poleceń uruchamiającą interfejs eth0 i ustawiające na nim adres 192.168.101.13/24

Zadanie 1.0.2

Napisz polecenie które doda trasę routingową do sieci 10.111.77.0/28 poprzez adres 10.111.66.7.

Zadanie 1.0.3

Napisz polecenia aktywujące przekazywanie pakietów (routing) pomiędzy tap0 a br0 (i wyłącznie pomiędzy tymi interfejsami).
Wskazówka: Może warto skorzystać z reguł filtracji pakietów?

Zadanie 1.0.4

Zmodyfikuj rozwiązanie zadania 1.0.3 tak aby przekazywane były tylko połączenia nawiązane, oraz połączenia ssh (port 22 tcp) i http (port 80 tcp) z tap0 do br0.

Zadanie 2.0.1

Napisz program będący serwerwem UDP, który odbierze od klienta ścieżkę do pliku i odeśle do niego zawartość tego pliku. W przypadku nie istnienia pliku program powinien odesłać do klienta odpowiedni komunikat. Zakładamy że podawane pliki są plikami tekstowymi i nie ma potrzeby ich dodatkowego kodowania.

Zadanie 2.0.2

Zmodyfikuj rozwiązanie zadania 2.0.1 tak aby używać TCP zamiast UDP.

Zadanie 2.0.3

Zastanów się w jaki sposób klient w zadaniach 2.0.1 i 2.0.2 może odróżnić przypadek otrzymania komunikatu o błędzie od otrzymania treści pliku zawierającego taki komunikat? Zwróć uwagę że protokoły takie jak np. HTTP mają osobną część nagłówka z kodem odpowiedzi oraz następującą po nim treść przesyłanego pliku / komunikatu.

Serwer echo UDP

```
import socket, sys

if len(sys.argv) != 2:
    print("USAGE: " + sys.argv[0] + " listenPort", file=sys.stderr)
    exit(1);

sfd = socket.socket(socket.AF_INET6, socket.SOCK_DGRAM)
sfd.setsockopt(socket.IPPROTO_IPV6, socket.IPV6_V6ONLY, 0)
sfd.bind(('::', int(sys.argv[1])))

while True:
    data, sAddr, = sfd.recvfrom(4096)
    print("odebrano od", sAddr, ":", data.decode());
    sfd.sendto(data, sAddr)
```

Zadanie 2.0.4

Uruchom dwie instancje serwera echo korzystającego z protokołu UDP.
Zastanów się co by się stało jeżeli jeden z tych serwerów dostałby pakiet pochodzący od drugiego z nich?
Korzystając z pakietu *scapy* oraz posiadając prawa root'a możemy przy pomocy Pythona wysyłać dowolnie spreparowane pakiety IP:

```
from scapy.all import IP, IPv6, UDP, send

send(IPv6(src=sIP, dst=dIP) / UDP(sport=sPort, dport=dPort) / "ABC ... XYZ")
# powyższa funkcja utworzy (a następnie wyśle):
#  → pakiet IPv6 od sIP do dIP
#    (adresy podajemy jako napisy),
#  → w którym jest pakiet UDP z portem źródłowym sPort i docelowym dport
#    (porty podajemy jako wartości numeryczne)
#  → w którym są dane "ABC ... XYZ"

# jeżeli zamiast IPv6() użyjemy IP() będziemy używać pakietu IPv4

# możemy też zaimportować inne funkcjonalności z modułu scapy
# (np. ICMP, TCP, ...) i używać ich do budowy naszych pakietów
```

Modyfikując powyższy kod spróbuj wysłać sfałszowany pakiet adresowany do jednego z serwerów, który jako adres nadawcy ma podany drugi z serwerów.
Scapy nie jest elementem biblioteki standardowej pythona – konieczne może być zainstalowanie pakietu ***python3-scapy*** *albo zainstalowanie go poprzez managera modułów pythonowych „pip”:* ***pip3 install scapy-python3****.*

Zadanie 2.0.5

Zobacz co się stanie jeżeli w sfałszowanym pakiecie podasz ten sam serwer jako nadawcę i odbiorcę. Usługa UDP-echo była kiedyś powszechnie stosowaną usługą diagnostyczną umożliwiająca testowanie połączenia sieciowego. Do tej pory ma nawet przyznany standardowy numer portu (7). Jak myślisz dlaczego usługa UDP-echo nie jest już powszechnie dostępną na każdym komputerze podłączonym do Internetu?

Zadania dodatkowe

Zadanie 4.0.1

Napisz polecenia, które:

- utworzą interfejs eth0.999 związany z tagowanym VLANem 999 na interfejsie eth0
- włączą ten interfejs

Zadanie 4.0.2

Napisz polecenia, które:

- utworzą bridge br0
- dodadzą interfejs eth0.999 do tego bridga
- włączą ten bridge
- ustawią na nim adres 10.111.66.13/28

Zadanie 4.0.3 

Napisz polecenia, które:

- utworzą interfejs tunelowy typu tap o nazwie tap0
- ustawią na nim adres 10.212.66.13/28
- włączą ten interfejs

Zadanie 4.0.4

Jakie ustawienie konfiguracyjne musi być wykonane na hostach podłączonych do tap0 i br0 aby mogły się one komunikować dzięki przekazywaniu pakietów skonfigurowanemu w zadaniu 1.0.3.

Zadanie 4.0.5

Napisz serwer UDP lub TCP (określ który wariant wybrałeś/wybrałaś), który na ciąg znaków `ip` wysłany przez klienta odeśle do niego informację o jego adresie IP.

Zadanie 4.0.6

Zapoznaj się z RFC1924 i napisz program konwertujący adresy IPv6 pomiędzy notacją dwukropkową a notacją base-85 zgodną z tą specyfikacją.
Wskazówka: do odczytu adresu w standardowej notacji dwukropkowej możesz użyć narzędzi z modułu `ipaddress`

© Matematyka dla Ciekawych Świata, 2021-2023.
© Robert Ryszard Paciorek <rrp@opcode.eu.org>, 2021-2023.
Kopiowanie, modyfikowanie i redystrybucja dozwolone pod warunkiem zachowania informacji o autorach.