Adresy IPv4
Ćwiczenie 3

Piotr Rosa
Grupa 1, poniedziałek 14:15

12 maja 2019

1 Skrypt do instalacji VirtualBoxa

Aby móc korzystać z VirtualBoxa na stacji w labrotatorium, każdorazowo trzeba go instalować. Aby uprościć ten proces, należy stworzyć skrypt, który zainstaluje VirtualBoxa razem z pakietem rozszerzeń. Poprawinoy skrypt, ktry naprawdę nie wymaga interakcji z uytkownikiem, powinien wyglądać tak:

```
#!/bin/sh
sudo apt-get update
sudo DEBIAN_FRONTEND=noninteractive apt-get install -yq virtualbox virtualbox-ext-pack
```

Dodanie **DEBIAN_FRONTEND=noninteractive** ustawia zmienna środowiskową, co pozwala na zainstalowanie programu, bez konieczności ingerencji użytkownika. Bez tej opcji, podczas instalacji użytkownik zostałby zapytany, czy chce kontynować oraz wyświetliłaby się licencja programu.

2 Koordynator ćwiczenia 2

Celem ćwiczenia 2 było napisanie skryptu, który stworzy maszynę wirtualną, znajdzie jej adres IP, prześle na nią skrypt tworzacy zadaną konfigurację sieci, wykona owy skrypt oraz sprawdzi jego poprawność. Skrypt powinien wyglądać następująco:

```
#!/bin/sh
c2-skrypt > c2.log
date >> c2.log
echo "Wynik skryptu zostal zapisany do pliku c2.log"
```

Polecenie **script** wewnątrz skryptu powłowki *sh* nie działa poprawnie, dlatego w powyższym skrypcie umieściłem "ręczną wersję" tego polecenia. Skrypt ten należy uruchomić podając nazwę maszyny jako parametr. Skrypt c2-skrypt, wykonany wzorując się na skrypcie dostarczonym przez prowadzącego, który jest uruchamiany, wygląda następująco:

```
#!/bin/sh
# c2-skrypt
# Całkowicie automatyczna generacja maszyny wirtualnej i siec w jej wnętrzu.
# rosap 2019

VM=FreeBSD                              # Nazwa maszyny

ustaw_automatycznie_IP () {             # Wariant automatyczny
    MAC=$(vb inf $VM | grep MAC: | cut -d, -f1 | sed 's/ *//g')
    IP=$(ssh ldap grep "DHCPACK.\*$MAC" /log/dhcpd | tail -1 | cut -w -f8)
```

```
}

# START

vb-install                              # Instalacja vbox-a

make-vm $VM                             # Generacja maszyny wirtualnej:

VBoxManage startvm $VM --type headless  # Włączenie maszyny

sleep 200                               # Czekaj aż maszyna wstanie

ustaw_automatycznie_IP $VM              # Uzyskanie adresu IP maszyny wirtualnej

scp -p ~/bin/create-int root@$IP        # Skopiowanie generatora sieci do MV:

ssh root@$IP sh -x ./create-int         # Uruchomienie generatora w MV

delete-vm $VM                           # Skasowanie maszyny

VBoxManage controlvm $VM poweroff

exit
```

Funkcja **ustaw_automatycznie_IP** pobiera adres MAC stworzonej maszyny i na podstawie tego adresu, odszukuje w pliku /**log**/**dhcpd** linię, w której został jej nadany adres IP. Następnie wybieram fragment tej linii, będący jej adresem IP.
Skrypt o nazwie **vb-install** instaluje *virtualboxa*.
Skrypt **make-vm**, którego parametrem jest nazwa maszyny, tworzy maszynę o tej nazwie.
Polecenie **sleep** zawiesza dalsze instrukcje na liczbę sekund podaną jako argument. Jest to konieczne, ponieważ maszyna wirtualna musi mieć czas, aby się uruchomić.
Skrypt **create-int** tworzy konfigurację zadaną konfigurację sieci na maszynie.
Skrypt **delete-vm** odpowiedzialny jest za usunięcie stworzonej wcześniej maszyny i wygląda tak:

```
#!/bin/sh
sleep 2
VBoxManage controlvm $1 poweroff
sleep 2
VBoxManage unregistervm $1 --delete
```

Pozostałe skrypty zostały opisane w poprzednim sprawozdaniu.

3 Ręczne nakładanie adresów

3.1 FreeBSD

Nakładanie adresów na systemie **FreeBSD** odbywa się przy użyciu polecenia **ifconfig**. Oto skrypt nakładający określony adres, sprawdzający stan interfejsów oraz usuwający nadane wcześniej adresy:

```
#!/bin/sh
ifconfig em0
ifconfig em0 inet 10.10.0.1/24 add
ifconfig em0
ifconfig em0 inet 10.10.0.1/24 delete
ifconfig em0
```

Log z konsoli po wykonaniu tego skryptu:

```
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=81009b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,VLAN_HWFILTER>
    ether 08:00:27:18:56:0f
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=81009b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,VLAN_HWFILTER>
    ether 08:00:27:18:56:0f
    inet 10.10.0.1 netmask 0xffffff00 broadcast 10.10.0.255
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=81009b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,VLAN_HWFILTER>
    ether 08:00:27:18:56:0f
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

Jak widać skrypt spełnił swoje zadanie.

3.2 Linux

Nałożenia adresów dokonam w systemie **Puppy Linux**. Poleceniem które wykorzystam jest **ip**. Skrypt:

```
#!/bin/sh
ip address show eth1
ip address add 10.10.0.2/24 dev eth1
ip address show eth1
ip address del 10.10.0.2/24 dev eth1
ip address show eth1
```

Wynik skryptu:

```
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 08:00:27:cd:42:3b brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 08:00:27:cd:42:3b brd ff:ff:ff:ff:ff:ff
    inet 10.10.0.2/24 scope global eth1
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 08:00:27:cd:42:3b brd ff:ff:ff:ff:ff:ff
```

3.3 Windows

Nakładanie adresów w systemie **Windows** odbywa się przy użyciu polecenia **netsh interface ipv4**. Zmian będę dokonywał na interfejsie o nazwie "Ethernet". Skrypt realizujacy zadanie:

```
netsh interface ipv4 show config "Ethernet"
netsh interface ipv4 set address "Ethernet" static 10.10.0.1 255.255.255.0
sleep 5
netsh interface ipv4 show config "Ethernet"
netsh interface ipv4 delete address "Ethernet" 10.10.0.1 gateway=all
netsh interface ipv4 show config "Ethernet"
```

Oraz jego wynik:

```
x:\windows\system32>netsh interface ipv4 show config "Ethernet"

Configuration for interface "Ethernet"
    DHCP enabled:                         No
    InterfaceMetric:                      10
    Statically Configured DNS Servers:    None
    Register with which suffix:           Primary only
    Statically Configured WINS Servers:   None

x:\windows\system32>netsh interface ipv4 set address "Ethernet" static 10.10.0.1
255.255.255.0

x:\windows\system32>sleep 5

x:\windows\system32>netsh interface ipv4 show config "Ethernet"

Configuration for interface "Ethernet"
    DHCP enabled:                         No
    IP Address:                           10.10.0.1
    Subnet Prefix:                        10.10.0.0/24 (mask 255.255.255.0)
    InterfaceMetric:                      10
    Statically Configured DNS Servers:    None
    Register with which suffix:           Primary only
    Statically Configured WINS Servers:   None

x:\windows\system32>netsh interface ipv4 delete address "Ethernet" 10.10.0.1
gateway=all

x:\windows\system32>netsh interface ipv4 show config "Ethernet"

Configuration for interface "Ethernet"
    DHCP enabled:                         No
    InterfaceMetric:                      10
    Statically Configured DNS Servers:    None
    Register with which suffix:           Primary only
    Statically Configured WINS Servers:   None
```

Polecenie **sleep 5** jest konieczne, ze względu na to, że adres nie jest dodawany natychmiast.

4 Dynamiczne nakładanie adresów

4.1 FreeBSD

Aby nadac adres przy użyciu protokołu **DHCP**, nalezy uzyc komendy **dhclient** wraz z nazwa interfejsu, na ktory ma zostac nalozony adres:

```
dhclient em0
```

Wynik polecenia:

```
DHCPREQUEST on em0 to 255.255.255.255 port 67
DHCPREQUEST on em0 to 255.255.255.255 port 67
DHCPDISCOVER on em0 to 255.255.255.255 port 67 interval 7
DHCPOFFER from 192.168.11.254
DHCPREQUEST on em0 to 255.255.255.255 port 67
DHCPACK from 192.168.11.254
bound to 192.168.11.1
-- renewal in 43200 seconds.
```

Wynik polecenia **ifconfig em0**:

```
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=81009b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,VLAN_HWFILTER>
    ether 08:00:27:18:56:0f
    inet 192.168.11.1 netmask 0xffffff00 broadcast 192.168.11.255
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

4.2 Linux

W przypadku systemu **Puppy Linux**, poleceniem jest **dhcpcd**. Wynik polecenia z nazwą interfejsu (**dhcpcd eth1**):

```
all: IPv6 kernel autoconf disabled
DUID 00:01:00:01:24:65:0d:25:08:00:27:e2:5e:d5
eth1: IAID 27:cd:42:3b
eth1: adding address fe80::e561:79d2:4514:bd50
if_addaddress6: Operation not supported
eth1: soliciting a DHCP lease
eth1: probing for an IPv4LL address
eth1: using IPv4LL address 169.254.230.85
eth1: adding route to 169.254.0.0/16
forked to background, child pid 15418
```

Oraz wynik polecenia **ip address show eth1**:

```
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 08:00:27:cd:42:3b brd ff:ff:ff:ff:ff:ff
    inet 169.254.230.85/16 brd 169.254.255.255 scope global eth1
        valid_lft forever preferred_lft forever
```

4.3 Windows

W systemie **Windows**, nadawanie adresu ip przy użyciu protokołu **DHCP** odbywa się przy użyciu tego samego polecenia, co w przypadku dodawania statycznego adresu - **netsh interface ipv4**. Skrypt realizujący zadanie:

```
netsh interface ipv4 show config "Ethernet"
netsh interface ipv4 set address "Ethernet" dhcp
sleep 10
netsh interface ipv4 show config "Ethernet"
```

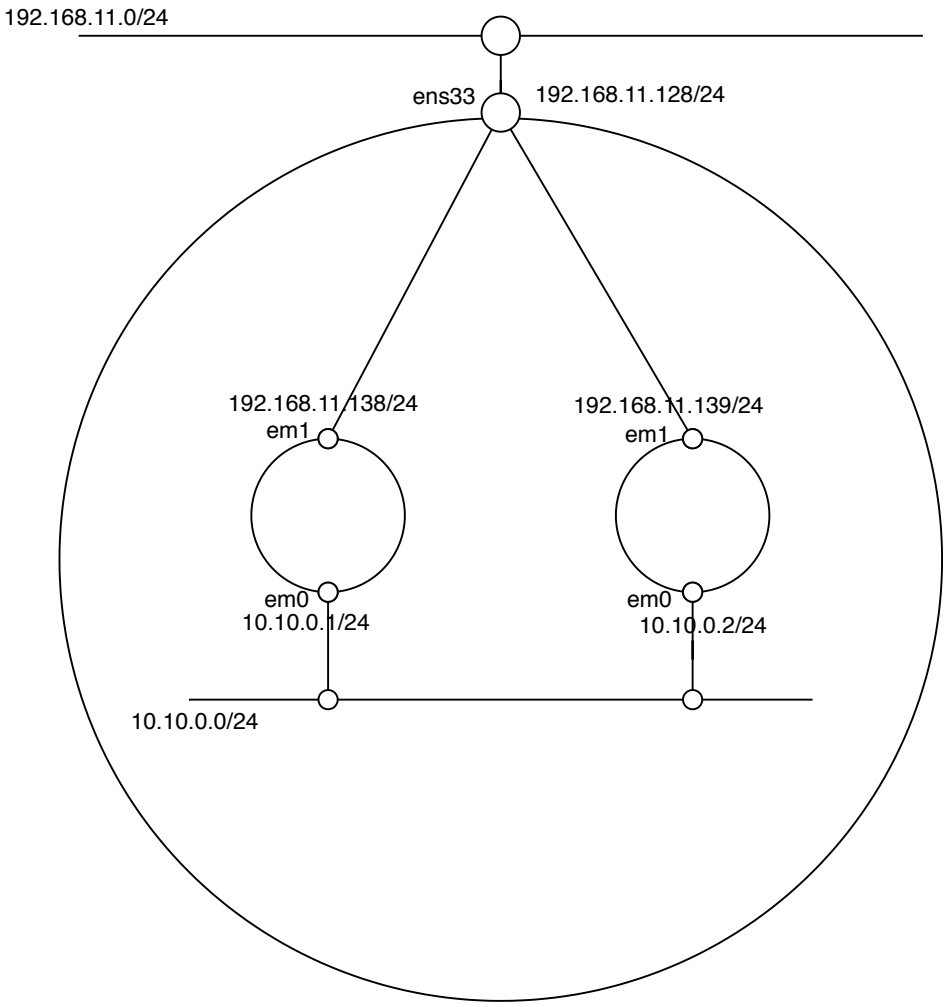
Wynik skryptu:

```
x:\windows\system32>netsh interface ipv4 show config "Ethernet"

Configuration for interface "Ethernet"
    DHCP enabled:                         No
    InterfaceMetric:                      10
    Statically Configured DNS Servers:    None
    Register with which suffix:           Primary only
    Statically Configured WINS Servers:   None

x:\windows\system32>netsh interface ipv4 set address "Ethernet" dhcp

x:\windows\system32>sleep 10

x:\windows\system32>netsh interface ipv4 show config "Ethernet"

Configuration for interface "Ethernet"
    DHCP enabled:                         Yes
    IP Address:                           169.254.43.68
    Subnet Prefix:                        169.254.0.0/16 (mask 255.255.0.0)
    InterfaceMetric:                      10
    DNS servers configured through DHCP:  None
    Register with which suffix:           Primary only
    WINS servers configured through DHCP: None
```

Tutaj również potrzebne jest polecenie **sleep**, jednak tym razem czekamy nawet dłużej, ponieważ nakładanie adresu przy użyciu protokołu **DHCP** trwa więcej czasu.

5 Schemat sieci - połączenie maszyn

Schemat sieci, w której będą komunikowały się stworzone maszyny wirtualne wygląda następująco:


192.168.11.0/24
ens33
192.168.11.128/24
192.168.11.138/24
em1
192.168.11.139/24
em1
em0
10.10.0.1/24
em0
10.10.0.2/24
10.10.0.0/24

6 Komunikacja pomiędzy maszynami

Obie maszyny wirtualne mają systemy operacyjne FreeBSD. Na odpowiednie interfejsy nakładam adresy zgodnie ze schematem. Używam do tego polecenia **ifconfig**. Na maszynie pierwszej:

```
ifconfig em0 inet 10.10.0.1/24 add
```

Na maszynie drugiej:

```
ifconfig em0 inet 10.10.0.2/24 add
```

Aby sprawdzić, czy maszyny mogą się ze sobą komunikować, na pierwszej maszynie używam polecenia **ping**, które wyśle pakiety do drugiej maszyny, tym samym sprawdzając łączność pomiędzy maszynami:

```
ping -c 3 10.10.0.2
```

Wynik tego polecenia:

```
PING 10.10.0.2 (10.10.0.2): 56 data bytes
64 bytes from 10.10.0.2: icmp_seq=0 ttl=64 time=0.813 ms
64 bytes from 10.10.0.2: icmp_seq=1 ttl=64 time=2.392 ms
64 bytes from 10.10.0.2: icmp_seq=2 ttl=64 time=1.544 ms

--- 10.10.0.2 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.813/1.583/2.392/0.645 ms
```

Jak widać, wszystkie pakiety wysłane przez pierwszą maszynę zostały dostarczone do maszyny drugiej. Oznacza to, że maszyny mogą się ze sobą komunikować.

7 Nakładanie drugiej warstwy sieciowej

Aby ręcznie nałożyć drugą warstwę adresową, wystarczy użyć tego samego polecenia nakładającego adres, ale na interfejsie który posiada już nałożony jeden adres. Przykladowy skrypt realizujący to zadanie:

```
#!/bin/sh
ifconfig em0
ifconfig em0 inet 172.11.11.12/24 add
ifconfig em0
ifconfig em0 inet 172.11.11.12/24 delete
ifconfig em0
```

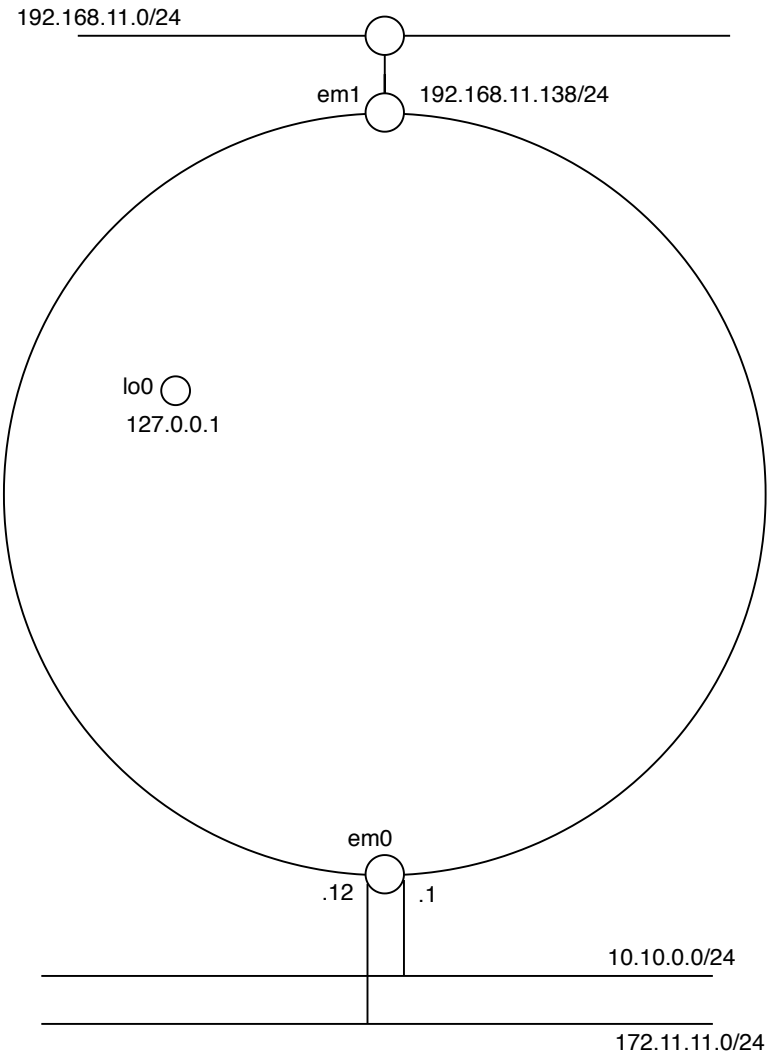
Poniżej wynik skryptu:

```
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=81009b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,VLAN_HWFILTER>
        ether 08:00:27:18:56:0f
        inet 10.10.0.1 netmask 0xffffff00 broadcast 10.10.0.255
        media: Ethernet autoselect (1000baseT <full-duplex>)
```

```
        status: active
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=81009b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,VLAN_HWFILTER>
        ether 08:00:27:18:56:0f
        inet 10.10.0.1 netmask 0xffffff00 broadcast 10.10.0.255
        inet 172.11.11.12 netmask 0xffffff00 broadcast 172.11.11.255
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=81009b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,VLAN_HWFILTER>
        ether 08:00:27:18:56:0f
        inet 10.10.0.1 netmask 0xffffff00 broadcast 10.10.0.255
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

Jak widać skrypt sukcesywnie nakłada i zdejmuje drugą warstwę sieciową.

Poniżej schemat sieci po dodaniu drugiej warstwy sieciowej:


192.168.11.0/24
em1
192.168.11.138/24
lo0
127.0.0.1
em0
.12
.1
10.10.0.0/24
172.11.11.0/24

8 Analiza ruchu sieciowego

W celu zbadania ruchu sieciowego uzywam polecenia **tcpdump**. Polecenie to wyswietla pakiety wyslane i pobrane na danym interfejsie.
Najpierw zbadam ruch sieciowy na pierwszej maszynie, podczas gdy uzyje polecenia **ping** na drugiej maszynie. A wiec - polecenie na drugiej maszynie:

```
ping 192.168.11.138
```

Oraz polecenie na pierwszej maszynie:

```
tcpdump -l -c 10 -i em1 -w ping.pcap
```

Nastepnie aby odczytac zawartosc pliku, ktory zostal przed chwila utworzony:

```
tcpdump -r ping.pcap
```

Wynik polecenia:

```
reading from file ping.pcap, link-type EN10MB (Ethernet)
17:22:37.238622 IP 192.168.11.138.ssh > 192.168.11.128.50154: Flags [P.], seq 245788927:
245789043, ack 196596805, win 1026, options [nop,nop,TS val 2879306469 ecr 499057640],
length 116
17:22:37.264115 IP 192.168.11.138.ssh > 192.168.11.128.50154: Flags [P.], seq 116:232, ack 1,
win 1026, options [nop,nop,TS val 2879306496 ecr 499057640], length 116
17:22:37.264540 IP 192.168.11.128.50154 > 192.168.11.138.ssh: Flags [.], ack 232, win 287,
options [nop,nop,TS val 499057935 ecr 2879306469], length 0
17:22:37.522802 IP 192.168.11.139 > 192.168.11.138: ICMP echo request, id 19459, seq 102,
length 64
17:22:37.522856 IP 192.168.11.138 > 192.168.11.139: ICMP echo reply, id 19459, seq 102,
length 64
17:22:38.268769 IP 192.168.11.138.63586 > 192.168.11.2.domain: 16570+ PTR?
138.11.168.192.in-addr.arpa. (45)
17:22:38.270651 IP 192.168.11.2.domain > 192.168.11.138.63586: 16570 NXDomain 0/0/0 (45)
17:22:38.271057 IP 192.168.11.138.21429 > 192.168.11.2.domain: 46783+ PTR?
128.11.168.192.in-addr.arpa. (45)
17:22:38.273062 IP 192.168.11.2.domain > 192.168.11.138.21429: 46783 NXDomain 0/0/0 (45)
17:22:38.297141 IP 192.168.11.138.ssh > 192.168.11.128.50154: Flags [P.], seq 232:452, ack 1,
win 1026, options [nop,nop,TS val 2879307530 ecr 499057935], length 220
```

Nastepnie zbadam ruch zwiazany z protokolem **ARP**. Nasluchiwanie interfejsu **ens33** na pakietach protokolu **ARP** moze zostac zrealizowane za pomoca komendy:

```
sudo tcpdump -l -c 5 -i ens33 -w arp.pcap arp
```

Aby odczytac utworzony plik uzywam polecenia:

```
tcpdump -r arp.pcap
```

Jego wynik:

```
reading from file arp.pcap, link-type EN10MB (Ethernet)
15:51:09.613378 ARP, Request who-has _gateway tell 192.168.11.1, length 46
15:51:10.572720 ARP, Request who-has _gateway tell 192.168.11.1, length 46
15:51:11.572679 ARP, Request who-has _gateway tell 192.168.11.1, length 46
15:51:12.614615 ARP, Request who-has _gateway tell 192.168.11.1, length 46
15:51:13.572968 ARP, Request who-has _gateway tell 192.168.11.1, length 46
```