Kacper Szaruch
Jan Wojciechowski

Politechnika Warszawska

# Sprawozdanie z realizacji laboratorium SLCD nr 1
# Protokół ARP

16 marca 2024, 22:00

## Spis treści

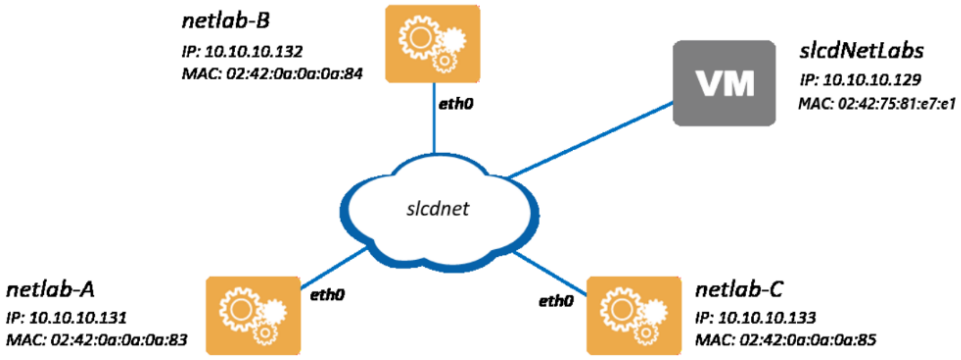
**Wstęp** . . . . . 1
**1. Protokół ARP** . . . . . 2
1.1. Podstawowe obserwacje . . . . . 2
1.2. Obserwacja zmian zawartości cache ARP . . . . . 2
**2. ARP cache poisoning (ARP spoofing)** . . . . . 4

## Wstęp

Niniejszy dokument to sprawozdanie z realizacji laboratorium w ramach przedmiotu SLCD. Oświadczamy, że ta praca, stanowiąca99 podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu SLCD, została wykonana przez nas samodzielnie.

# 1. Protokół ARP

## 1.1. Podstawowe obserwacje




Rys. 1: Architektura sieci wykorzystanej w ćwiczeniu

| Nazwa | IPv4 | MAC |
|---|---|---|
| slcdNetLabs | 10.10.10.129 | 02:42:75:81:e7:e1 |
| netlab-A | 10.10.10.131 | 02:42:0a:0a:0a:83 |
| netlab-B | 10.10.10.132 | 02:42:0a:0a:0a:84 |
| netlab-C | 10.10.10.133 | 02:42:0a:0a:0a:85 |

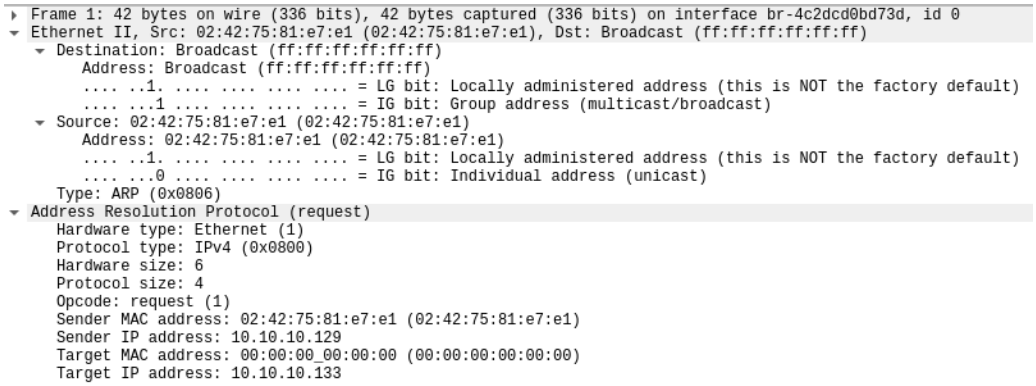
Tabela [6.1.R1]: Adresacja **IP** i **MAC** przypisanym interfejsom **eth0** dla poszczególnych kontenerów i maszyny hosta

## 1.2. Obserwacja zmian zawartości cache ARP

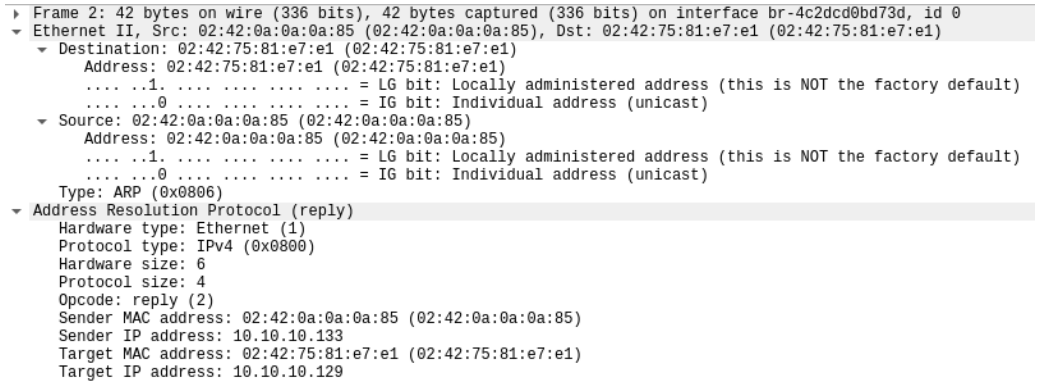
```
root@slcdNetLabs:/home/student# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether   52:54:00:12:35:02   C                     enp0s3
10.10.10.133             ether   02:42:0a:0a:0a:85   C                     br-4c2dcd0bd73d
192.168.56.100           ether   08:00:27:42:d7:7c   C                     enp0s8
root@slcdNetLabs:/home/student#
```

Rys. [6.3.R2]: Zmieniona zawartość **cache ARP** po wykonaniu polecenia *ping*

Po wykonaniu polecenia *ping* z docelowym adresem IP 10.10.10.133 do **cache ARP** został dodany nowy wpis zawierający adres MAC powiązany z adresem IP wykorzystanym w poleceniu *ping*.




Rys. [6.3.R3]: Ramka Ethernet przenosząca ARP_request




Rys. 4: Ramka Ethernet przenosząca ARP_reply

**Ramka Ethernet składa się z poniższych pól:**

- Destination MAC address - adres MAC interfejsu docelowego
- Source MAC address - adres MAC interfejsu źródłowego
- Type - identyfikator typu danych przenoszonych w polu payload w tym przypadku 0x0806 oznaczający protokół ARP
- Payload - przenoszone dane użytkowe, w tym przypadku pakiet ARP
- Cyclic Redundancy Code - suma kontrolna wyliczana dla zawartości całej ramki, niestety zaobserwowanie tego pola w programie Wireshark jest niemożliwe, ponieważ większość NIC (Network Interface Card) pozbywa się tej informacji przed przekazaniem pakietu do sytemu

**Pakiet ARP składa się z poniższych pól:**

- Hardware type - określa typ sieci (dla siecie Ethernet wartość wynosi 1)
- Protocol type - określa rodzaj użytego adresu (dla adresu IPv4 wartość wynosi 0x0800)
- Hardware size - określa długość adresu sprzętowego (W tym przypadku wartość wynosi 6, która wskazuje na adres MAC)
- Protocol size - określa długość użytego adresu (W tym przypadku wartość wynosi 4, która wskazuje na adres IPv4)
- Opcode - określa typ pakietu ARP (dla ARP request wynosi 1, natomiast dla ARP reply wynosi 2)
- Sender MAC address - adres MAC nadawcy w przypadku ARP request, w przypadku ARP reply żądany adres MAC
- Sender IP address - adres IP nadawcy w przypadku ARP request, w przypadku ARP reply żądany adres IP
- Target MAC address - w przypadku ARP request to pole jest puste, ponieważ nadawca nie zna żądanego adresu MAC, w przypadku ARP reply adres MAC urządzenia, które wysłało pierwotnie ARP request
- Target IP address - w przypadku ARP request adres IP urządzenia, którego nadawca chce poznać adres MAC, w przypadku ARP reply adres IP urządzenia, które wysłało pierwotnie ARP request
- PAD - to pole nie jest wyświetlane w programie Wireshark, ponieważ nie niesie ono żadnej informacji, służy tylko do spełnienia wymogu minimalnego rozmiaru pola **Payload** w ramce Ethernet

## 2. ARP cache poisoning (ARP spoofing)

```
netlab-B # arp
netlab-C-container.slcdnet (10.10.10.133) at 02:42:0a:0a:0a:85 [ether]  on eth0
? (10.10.10.129) at 02:42:75:81:e7:e1 [ether]  on eth0
netlab-B #
```

Rys. 5: Zawartość ARP cache na kontenerze netlab-B przed uruchomieniem skryptu *arpPoisoner.py*

```
netlab-C # arp
? (10.10.10.129) at 02:42:75:81:e7:e1 [ether]  on eth0
netlab-B-container.slcdnet (10.10.10.132) at 02:42:0a:0a:0a:84 [ether]  on eth0
netlab-C #
```

Rys. 6: Zawartość ARP cache na kontenerze netlab-C przed uruchomieniem skryptu *arpPoisoner.py*

```
root@slcdNetLabs:/home/student# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.10.132             ether   02:42:0a:0a:0a:84   C                     br-4c2dcd0bd73d
192.168.56.100           ether   08:00:27:42:d7:7c   C                     enp0s8
10.10.10.133             ether   02:42:0a:0a:0a:85   C                     br-4c2dcd0bd73d
_gateway                 ether   52:54:00:12:35:02   C                     enp0s3
root@slcdNetLabs:/home/student#
```

Rys. 7: Zawartość ARP cache na host-cie przed uruchomieniem skryptu *arpPoisoner.py*

W celu przeprowadzeniu tego ćwiczenia na wstępie zostały zweryfikowane początkowe wartości ARP cache dla kontenerów netlab-B i netlab-C oraz maszyny hosta. Następnie uruchomiono przygotowany przez prowadzącego ćwiczenie skrypt arpPoisoner.py na kontenerze netlab-A w celu wykonania ataku typu ARP spoofing.

```
netlab-B # arp
netlab-A-container.slcdnet (10.10.10.131) at 02:42:0a:0a:0a:83 [ether]  on eth0
netlab-C-container.slcdnet (10.10.10.133) at 02:42:0a:0a:0a:83 [ether]  on eth0
? (10.10.10.129) at 02:42:75:81:e7:e1 [ether]  on eth0
netlab-B #
```

Rys. [7.2.R1]: Zawartość ARP cache na kontenerze netlab-B po uruchomieniu skryptu *arpPoisoner.py*

```
netlab-C # arp
netlab-A-container.slcdnet (10.10.10.131) at 02:42:0a:0a:0a:83 [ether]  on eth0
? (10.10.10.129) at 02:42:75:81:e7:e1 [ether]  on eth0
netlab-B-container.slcdnet (10.10.10.132) at 02:42:0a:0a:0a:83 [ether]  on eth0
netlab-C #
```

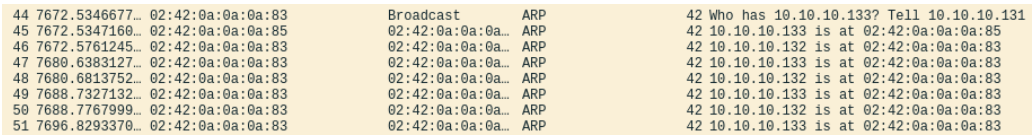
Rys. [7.2.R1]: Zawartość ARP cache na kontenerze netlab-C po uruchomieniu skryptu *arpPoisoner.py*

```
root@slcdNetLabs:/home/student# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.10.132             ether   02:42:0a:0a:0a:84   C                     br-4c2dcd0bd73d
192.168.56.100           ether   08:00:27:42:d7:7c   C                     enp0s8
10.10.10.133             ether   02:42:0a:0a:0a:85   C                     br-4c2dcd0bd73d
_gateway                 ether   52:54:00:12:35:02   C                     enp0s3
root@slcdNetLabs:/home/student#
```

Rys. [7.2.R1]: Zawartość ARP cache na host-cie po uruchomieniu skryptu *arpPoisoner.py*

Jak można zauważyć na powyższych zdjęciach stany ARP cache na kontenerach netlab-B i netlab-C uległy zmianom. Wpisy dotyczące kontenerów zostały zmodyfikowane w taki sposób, że w przypadku próby przesłania danych między nimi w rzeczywistości zostałyby one wysłane do atakującego.

Na podstawie poniższego zdjęcia można zaobserwować jak atakujący był w stanie wywołać zmiany w ARP cache na pozostałych kontenerach. Analizując ruch zaobserwowany w trakcie działania skryptu *arpPoisoner.py* wywnioskowano, że atakujący regularnie wysyłał bezpośrednio informację do ofiar o tym, że adres IP kontenerów netlab-B i netlab-C (zależnie od tego, gdzie był wysyłany pakiet) są powiązane z adresem MAC **02:42:0a:0a:0a:83**, który w rzeczywistości jest przypisany do kontenera atakującego.




Rys. 11: Ruch zaobserwowany w trakcie korzystanie z skryptu *arpPoisoner.py*

Obecnie atakujący nic nie robi z przechwyconymi pakietami, dlatego komunikacja pomiędzy netlab-B i netlab-C jest niemożliwa. Potwierdzają to poniższe wykonania polecenia *ping*.

```
netlab-B # ping -c 1 10.10.10.133
PING 10.10.10.133 (10.10.10.133): 56 data bytes
^C
--- 10.10.10.133 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
netlab-B #
```

Rys. [7.2.R2]: Wynik wykonania polecenia *ping* z netlab-B do netlab-C

```
netlab-C # ping -c 1 10.10.10.132
PING 10.10.10.132 (10.10.10.132): 56 data bytes
^C
--- 10.10.10.132 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
netlab-C #
```

Rys. [7.2.R2]: Wynik wykonania polecenia *ping* z netlab-C do netlab-B

W drugiej części eksperymentu na kontenerze atakującego uruchomiono dodatkowy skrypt *ethForwarder.py*, który ponownie umożliwia komunikacje pomiędzy netlab-B i netlab-C. Ponownie potwierdza to poniższe wykonanie polecenia *ping*.

```
netlab-B # ping -c 1 10.10.10.133
PING 10.10.10.133 (10.10.10.133): 56 data bytes
64 bytes from 10.10.10.133: seq=0 ttl=64 time=95.878 ms

--- 10.10.10.133 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 95.878/95.878/95.878 ms
netlab-B #
```

Rys. [7.2.R3]: Wynik wykonania polecenia *ping* z netlab-B do netlab-C po uruchomieniu dodatkowego skryptu

Analizując ruch zaobserwowany po wykonaniu komendy *ping*, można zauważyć, że w rzeczywistości atakujący przekazuje teraz pakiety w obu kierunkach. Przebieg komunikacji można podzielić na trzy etapy:

1. netlab-B chce wysłać pakiet ICMP do netlab-C, ale z powodu ARP poisoning wysyła ten pakiet do atakującego, na który nie otrzymuje od razu odpowiedzi przez co w pierwszym pakiecie widzimy referencje "No response found".
2. Atakujący podszywając się pod netlab-B (zamiana adresu nadawcy) dokonuje "normalnej" wymiany dwóch pakietów z netlab-C.
3. Na koniec atakujący podszywając się pod netlab-C przesyła kopię pakietu reply z powrotem do netlab-B.


Rys. 15: Ruch zaobserwowany w trakcie korzystanie z skryptu *ethForwarder.py*

Adresy źródłowe oraz docelowe pakietów są pełni poprawne i nie wzbudzają żadnych podejrzeń. Jednakże jak przedstawiono powyżej istnieją ślady wskazujące na trwanie ataku ARP spoofing. Najbardziej widoczny z nich to zwiększona ilość pakietów, trzeb jednak pamiętać, że z perspektywy samych kontenerów netlab-B i netlab-C nie jest możliwe dostrzeżenia tego zjawiska. Kolejnym alarmującym elementem wymiany pakietów w trakcie ataku są wystąpienia referencji, które nie są częścią standardowego outputu dla komendy *ping*. Z perspektywy netlab-B jest to referencja "No response found", natomiast dla netlab-C "request in 223" przy odpowiedzi.

| X | Pracownik zdalny | Cloud |
|---|---|---|
| slcdNetLabs | 10.10.10.129 | 02:42:75:81:e7:e1 |
| netlab-A | 10.10.10.131 | 02:42:0a:0a:0a:83 |
| netlab-B | 10.10.10.132 | 02:42:0a:0a:0a:84 |
| netlab-C | 10.10.10.133 | 02:42:0a:0a:0a:85 |

Tabela [6.1.R1]: Adresacja **IP** i **MAC** przypisanym interfejsom **eth0** dla poszczególnych kontenerów i maszyny hosta