Samuel Steinberg

November 16th, 2019

CS 366

Assignment 4

Task 1:

Question 1:

IP and MAC addresses:

- Windows (A)
  - IP Address: 192.168.1.1
  - MAC Address: 00:0c:29:ca:5c:d6
- Middle (B)
  - IP Address: 192.168.2
  - MAC Address: 00:0c:29:04:85:d3
- Client (C)
  - IP Address: 192.168.1.3
  - MAC Address: 00:0c:29:e1:7a:29

These addresses were confirmed with the **arp -a** command for Windows and the **arp -an** command for Linux. See the screenshots below:

Windows (A):

```
C:\Users\root>arp -a

Interface: 192.168.1.1 --- 0xb
  Internet Address      Physical Address      Type
  192.168.1.2           00-0c-29-04-85-d3     dynamic
  192.168.1.3           00-0c-29-e1-7a-29     dynamic
```

Middle (B):

```
root@kali:~# arp -an
? (192.168.1.1) at 00:0c:29:ca:5c:d6 [ether] on eth0
? (192.168.1.3) at 00:0c:29:e1:7a:29 [ether] on eth0
root@kali:~#
```

Client (C):

```
root@kali:~# arp -an
? (192.168.1.1) at 00:0c:29:ca:5c:d6 [ether] on eth0
? (192.168.1.2) at 00:0c:29:04:85:d3 [ether] on eth0
root@kali:~#
```

Question 2:

The **-M** option for Ettercap activates the man in the middle (MITM) attack. The aim of the attack is to hijack packets and redirect them to Ettercap. The **arp** option implements the arp poisoning man in the middle attack, where arp requests and replies are sent to the victims to poison their ARP cache. After the cache has been poisoned the victims will now send packets to the attacker who can modify and forward them to the real destination. The targets of the operation are the Windows machine (A) and the Client machine (C). Targets are put inserted at the end of the command in the form of: '/ip address here/'. The targets filter traffic coming from one to the other and vice-versa because the connection is bi-directional.

Question 3:

After the MITM attack, here are the resulting ARP caches from each system:

Windows (A):

```
C:\Users\root>arp -a

Interface: 192.168.1.1 --- 0xb
  Internet Address      Physical Address      Type
  192.168.1.2           00-0c-29-04-85-d3     dynamic
  192.168.1.3           00-0c-29-04-85-d3     dynamic
```

Middle (B):

```
root@kali:~# arp -an
? (192.168.1.1) at 00:0c:29:ca:5c:d6 [ether] on eth0
? (192.168.1.3) at 00:0c:29:e1:7a:29 [ether] on eth0
```

Client (C):

```
root@kali:~# arp -an
? (192.168.1.1) at 00:0c:29:ca:5c:d6 [ether] on eth0
? (192.168.1.2) at 00:0c:29:04:85:d3 [ether] on eth0
```

The main change after the attack was that the Windows machine, which acts as the server, will now direct traffic to different IP addresses, but the same physical address which is that of the man in the middle (Machine B). The other machines B and C still show their traffic going to different connections on the network. Hence, this was a successful attack. The Middle attacker will now receive traffic that goes from A to C.

Question 4:

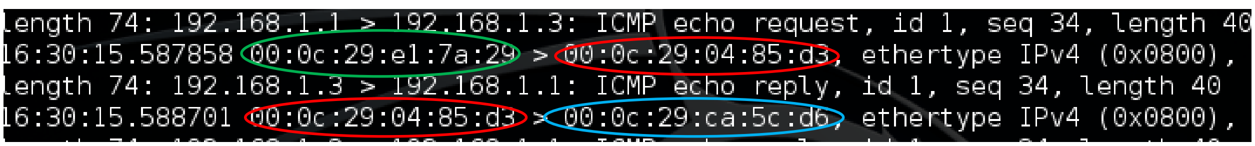
The following screenshot is a portion of the result of the tcpdump from the Middle (B) during/after A and C were pinging each other with Ettercap running:

```
length 74: 192.168.1.1 > 192.168.1.3: ICMP echo request, id 1, seq 34, length 40
16:30:15.585851 00:0c:29:04:85:d3 > 00:0c:29:e1:7a:29, ethertype IPv4 (0x0800),
length 74: 192.168.1.1 > 192.168.1.3: ICMP echo request, id 1, seq 34, length 40
16:30:15.587858 00:0c:29:e1:7a:29 > 00:0c:29:04:85:d3, ethertype IPv4 (0x0800),
length 74: 192.168.1.3 > 192.168.1.1: ICMP echo reply, id 1, seq 34, length 40
16:30:15.588701 00:0c:29:04:85:d3 > 00:0c:29:ca:5c:d6, ethertype IPv4 (0x0800),
length 74: 192.168.1.3 > 192.168.1.1: ICMP echo reply, id 1, seq 34, length 40
16:30:15.912395 00:0c:29:ca:5c:d6 > 33:33:00:01:00:02, ethertype IPv6 (0x86dd),
length 150: fe80::b163:c88a:897d:86a0.546 > ff02::1:2.547: dhcp6 solicit
16:30:16.582984 00:0c:29:ca:5c:d6 > 00:0c:29:04:85:d3, ethertype IPv4 (0x0800),
length 74: 192.168.1.1 > 192.168.1.3: ICMP echo request, id 1, seq 35, length 40
16:30:16.584076 00:0c:29:04:85:d3 > 00:0c:29:e1:7a:29, ethertype IPv4 (0x0800),
length 74: 192.168.1.1 > 192.168.1.3: ICMP echo request, id 1, seq 35, length 40
16:30:16.585804 00:0c:29:e1:7a:29 > 00:0c:29:04:85:d3, ethertype IPv4 (0x0800),
length 74: 192.168.1.3 > 192.168.1.1: ICMP echo reply, id 1, seq 35, length 40
16:30:16.588533 00:0c:29:04:85:d3 > 00:0c:29:ca:5c:d6, ethertype IPv4 (0x0800),
length 74: 192.168.1.3 > 192.168.1.1: ICMP echo reply, id 1, seq 35, length 40
16:30:26.629765 00:50:56:c0:00:01 > ff:ff:ff:ff:ff:ff, ethertype IPv4 (0x0800),
length 60: 192.168.50.1.57261 > 255.255.255.255.8610: UDP, length 16
16:30:26.629814 00:50:56:c0:00:01 > ff:ff:ff:ff:ff:ff, ethertype IPv4 (0x0800),
length 60: 192.168.50.1.57261 > 255.255.255.255.8610: UDP, length 16
16:30:31.917549 00:0c:29:ca:5c:d6 > 33:33:00:01:00:02, ethertype IPv6 (0x86dd),
length 150: fe80::b163:c88a:897d:86a0.546 > ff02::1:2.547: dhcp6 solicit
```

The different behavior when Ettercap is running stems from many more packet movements in the dump due to the traffic being diverted to the physical address of the middleman B. When the server A tries to send packets to the client C or vice-versa, they are being diverted to the physical address of B and then sent to their destination from B.

Question 5:

As shown in the tcpdump, Middle B is receiving the packets intended from A to C and vice-versa. These packets have the potential to be modified before being sent on. This signifies a successful man in the middle attack. See the screenshot below for a visual:



```
length 74: 192.168.1.1 > 192.168.1.3: ICMP echo request, id 1, seq 34, length 40
16:30:15.587858 00:0c:29:e1:7a:29 > 00:0c:29:04:85:d3, ethertype IPv4 (0x0800),
length 74: 192.168.1.3 > 192.168.1.1: ICMP echo reply, id 1, seq 34, length 40
16:30:15.588701 00:0c:29:04:85:d3 > 00:0c:29:ca:5c:d6, ethertype IPv4 (0x0800),
```

In the above image, the MAC address outlined in green signifies the source address while the blue represents the intended address. Clearly, the MAC outlined in red is acting as an unintended middleman; receiving the packets and then sending them on to their intended destination.

Question 6:

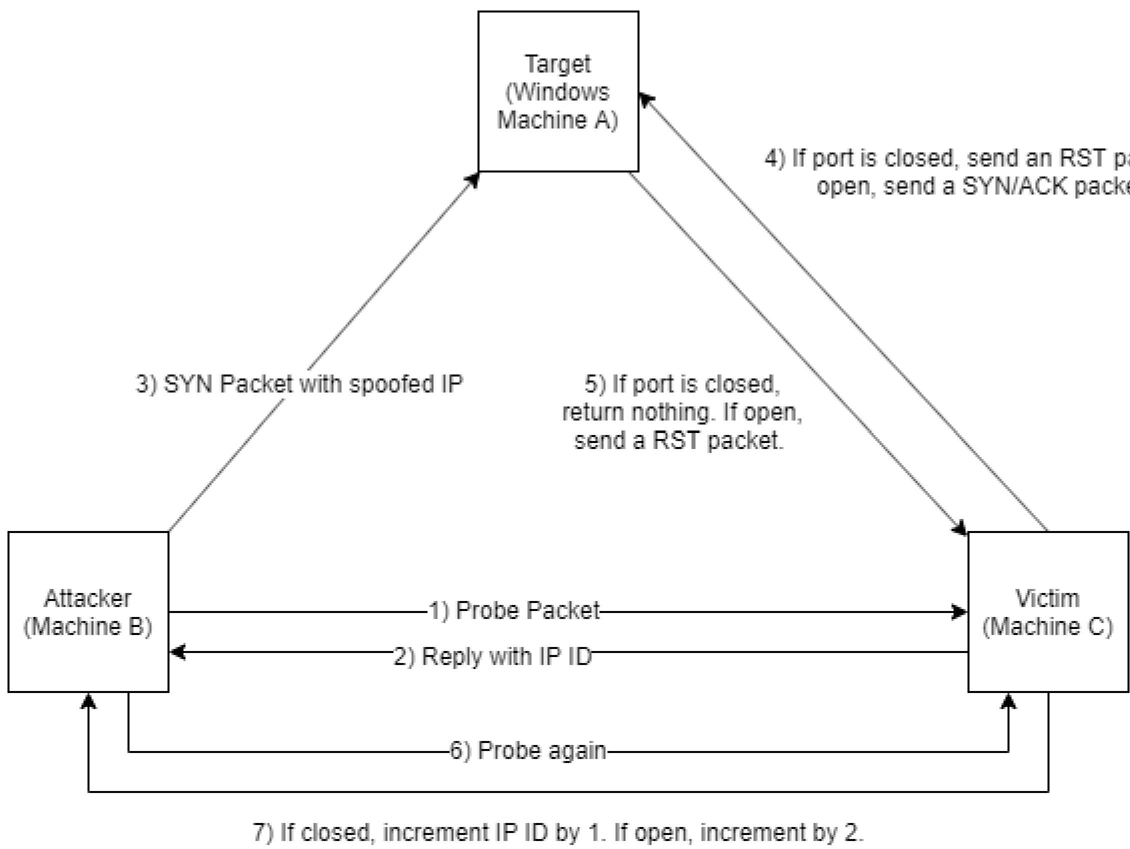
Ettercap contains several plugins to find forensic evidence of an attack having occurred. These include arp_cop, which reports suspicious ARP activity by passively monitoring ARP requests and replies, and also reports ARP poisoning attempts. Evidence such as IP-changes or IP-conflicts are reported and noted. Another way to gather evidence is using scan_poisoner, provided by Ettercap. This plugin allows to check for middleman attacks by checking if two hosts have the same MAC address, since this is a piece of evidence left behind after a session hijack. This is performed by sending ICMP packets to each host in the list and checking if the source MAC address differs from the address stored in a list for that IP.

Task 2

Question 7:

The first step in making this attack work would be to shut down Windows (A) firewall, and this machine will act as the idle host (zombie). This was completed by running the command prompt as an administrator and running the command: **netsh advfirewall set allprofiles state off**. This command performs a shutdown of the firewall using advanced privileges. This command-line utility is only available for Microsoft Windows OS. Next, the Middle (B) needs run a port scan using nmap to discover hosts and services on a network. This is performed by sending probing packets and analyzing their responses. The nmap utility can run on either the Linux or Windows operating systems. The command run here to find open ports and perform an idle scan between networks was the following: **nmap -sI 192.168.1.3 -p 80 192.168.1.1** with '-p 80' specifying the port. This machine will act as the scanner's host. With powerful capabilities and tools, the Middle (B) will not have many obstacles getting to Client (C), which is the victim, especially when Windows (A) firewall is down and can even be used as the target.

To summarize how the attack would work, the attacker's goal is to both port scan the target (A) and spoof the victims (C) IP in order to hide their identity. To be able to do this, the victim needs to run a Windows machine with a predictable IP ID increment and is a quiet host (not busy). Additionally, the attacker and victim need to be on the LAN to increase the chance of success. See the following diagram:



Question 8:

- Initial probing of victim from attacker's machine:

```
root@kali:~# hping3 -r 192.168.1.3
HPING 192.168.1.3 (eth0 192.168.1.3): NO FLAGS are set, 40 headers + 0 data byte
s
len=46 ip=192.168.1.3 ttl=64 DF id=30633 sport=0 flags=RA seq=0 win=0 rtt=0.6 ms
len=46 ip=192.168.1.3 ttl=64 DF id=+180 sport=0 flags=RA seq=1 win=0 rtt=2.2 ms
len=46 ip=192.168.1.3 ttl=64 DF id=+33 sport=0 flags=RA seq=2 win=0 rtt=2.4 ms
len=46 ip=192.168.1.3 ttl=64 DF id=+98 sport=0 flags=RA seq=3 win=0 rtt=2.2 ms
len=46 ip=192.168.1.3 ttl=64 DF id=+14 sport=0 flags=RA seq=4 win=0 rtt=2.3 ms
len=46 ip=192.168.1.3 ttl=64 DF id=+157 sport=0 flags=RA seq=5 win=0 rtt=2.0 ms
len=46 ip=192.168.1.3 ttl=64 DF id=+77 sport=0 flags=RA seq=6 win=0 rtt=2.8 ms
len=46 ip=192.168.1.3 ttl=64 DF id=+8 sport=0 flags=RA seq=7 win=0 rtt=2.3 ms
len=46 ip=192.168.1.3 ttl=64 DF id=+92 sport=0 flags=RA seq=8 win=0 rtt=2.2 ms
```

- Probing an open port on the target (TCP/80)
  - Attacker Machine

```
root@kali:~# hping3 -r 192.168.1.1
HPING 192.168.1.1 (eth0 192.168.1.1): NO FLAGS are set, 40 headers + 0 data byte
s
len=46 ip=192.168.1.1 ttl=128 DF id=782 sport=0 flags=RA seq=0 win=0 rtt=33.9 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=1 win=0 rtt=4.6 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=2 win=0 rtt=1.6 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=3 win=0 rtt=1.0 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=4 win=0 rtt=1.2 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=5 win=0 rtt=1.0 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=6 win=0 rtt=1.0 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=7 win=0 rtt=1.4 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=8 win=0 rtt=0.7 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=9 win=0 rtt=1.1 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=10 win=0 rtt=0.6 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=11 win=0 rtt=1.2 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=12 win=0 rtt=0.6 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+2 sport=0 flags=RA seq=13 win=0 rtt=1.9 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+2 sport=0 flags=RA seq=14 win=0 rtt=1.0 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+2 sport=0 flags=RA seq=15 win=0 rtt=1.1 ms
```

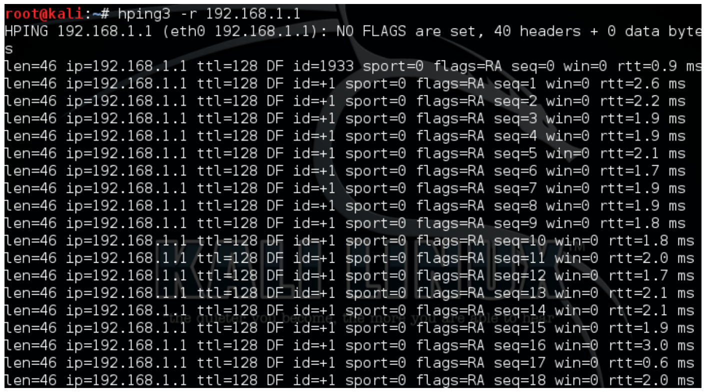
Note that the scan on the open port began at the ID increment change.

  - Target Machine (tcpdump)

```
root@kali:~# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
12:59:34.215617 ARP, Request who-has 192.168.1.1 tell 192.168.1.3, length 28
12:59:34.216819 ARP, Reply 192.168.1.1 is-at 00:0c:29:dd:4f:2c (oui Unknown), le
ngth 46
12:59:34.226011 IP 192.168.1.1.2355 > 192.168.1.3.http: Flags [S], seq 238801063
, win 512, length 0
12:59:34.226094 IP 192.168.1.3.http > 192.168.1.1.2355: Flags [S.], seq 80313860
1, ack 238801064, win 29200, options [mss 1460], length 0
12:59:34.227122 IP 192.168.1.1.2355 > 192.168.1.3.http: Flags [R], seq 238801064
, win 0, length 0
12:59:34.294410 IP 192.168.1.2.2380 > 192.168.1.1.0: Flags [R], win 512, length
0
12:59:34.294437 IP 192.168.1.1.0 > 192.168.1.2.2380: Flags [R.], seq 0, ack 1756
666101, win 0, length 0
12:59:35.228124 IP 192.168.1.1.2356 > 192.168.1.3.http: Flags [S], seq 898383985
, win 512, length 0
12:59:35.228205 IP 192.168.1.3.http > 192.168.1.1.2356: Flags [S.], seq 33564852
05, ack 898383986, win 29200, options [mss 1460], length 0
12:59:35.230091 IP 192.168.1.1.2356 > 192.168.1.3.http: Flags [R], seq 898383986
```

Note that there is an exchange of SYN/ACK packets, noting that there is an open port.

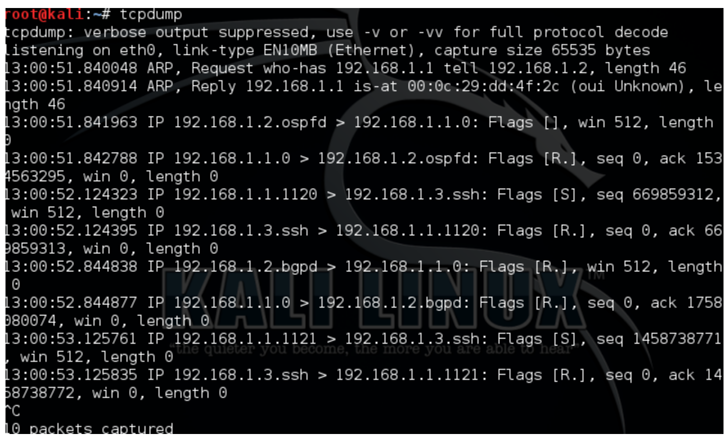
- Probing a closed port on the target (TCP/22)
    - Attacker Machine



```
root@kali:~# hping3 -r 192.168.1.1
HPING 192.168.1.1 (eth0 192.168.1.1): NO FLAGS are set, 40 headers + 0 data byte
s
len=46 ip=192.168.1.1 ttl=128 DF id=1933 sport=0 flags=RA seq=0 win=0 rtt=0.9 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=1 win=0 rtt=2.6 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=2 win=0 rtt=2.2 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=3 win=0 rtt=1.9 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=4 win=0 rtt=1.9 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=5 win=0 rtt=2.1 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=6 win=0 rtt=1.7 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=7 win=0 rtt=1.9 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=8 win=0 rtt=1.9 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=9 win=0 rtt=1.8 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=10 win=0 rtt=1.8 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=11 win=0 rtt=2.0 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=12 win=0 rtt=1.7 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=13 win=0 rtt=2.1 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=14 win=0 rtt=2.1 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=15 win=0 rtt=1.9 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=16 win=0 rtt=3.0 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=17 win=0 rtt=0.6 ms
len=46 ip=192.168.1.1 ttl=128 DF id=+1 sport=0 flags=RA seq=18 win=0 rtt=2.0 ms
```

There is no ID increment change on a closed port.

- Target Machine (tcpdump)



```
root@kali:~# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
13:00:51.840048 ARP, Request who-has 192.168.1.1 tell 192.168.1.2, length 46
13:00:51.840914 ARP, Reply 192.168.1.1 is-at 00:0c:29:dd:4f:2c (oui Unknown), le
ngth 46
13:00:51.841963 IP 192.168.1.2.ospfd > 192.168.1.1.0: Flags [], win 512, length
0
13:00:51.842788 IP 192.168.1.1.0 > 192.168.1.2.ospfd: Flags [R.], seq 0, ack 153
4563295, win 0, length 0
13:00:52.124323 IP 192.168.1.1.1120 > 192.168.1.3.ssh: Flags [S], seq 669859312,
 win 512, length 0
13:00:52.124395 IP 192.168.1.3.ssh > 192.168.1.1.1120: Flags [R.], seq 0, ack 66
9859313, win 0, length 0
13:00:52.844838 IP 192.168.1.2.bgpd > 192.168.1.1.0: Flags [R.], win 512, length
 0
13:00:52.844877 IP 192.168.1.1.0 > 192.168.1.2.bgpd: Flags [R.], seq 0, ack 1758
980074, win 0, length 0
13:00:53.125761 IP 192.168.1.1.1121 > 192.168.1.3.ssh: Flags [S], seq 1458738771
, win 512, length 0
13:00:53.125835 IP 192.168.1.3.ssh > 192.168.1.1.1121: Flags [R.], seq 0, ack 14
58738772, win 0, length 0
^C
10 packets captured
```

There is an exchange of RST packets, showing that the port is closed.

Question 9:

- In the initial probing, the victims IP ID incremented by a seemingly random value. It is usually by a two- or three-digit number.
- When probing an open port, the IP ID changed its increment though it was still a constant value. It went from incrementing by one to incrementing by two.
- In the closed port probing, the IP ID does not change increments. This is because the port is not being serviced.
- In the point of view of the target, the offending host was the victim Machine C. hping3 probing did not leave any trace of the real attacker (B) and only communication and packet exchanges between A and C.