Instituto Superior Técnico

Advanced Network Security and Architectures

Laboratory Report:

Module 1

Group 9

Félix Saraiva – 98752

Miguel Ribeiro – 87553

Rafael Martins – 90629

2022

Table of Contents

Table of Figures:

1. Lab No. 1.1 - Network Attacks and Countermeasures

This laboratory aims to study network attacks and countermeasures. The following attacks will be addressed:

- **(i)** CAM table overflow
- **(ii)** DHCP spoofing
- **(iii)** ARP poisoning (MitM)
- **(iv)** Root Bridge spoofing (MitM)
- **(v)** DNS spoofing
- **(vi)** RIP poisoning
- **(vii)** DNS spoofing using DHCP spoofing

1.1. CAM Table Overflow

A content addressable memory (CAM) table contains the MAC addresses mapped to the port numbers on a switch, associated with their respective VLAN Parameters. The switch dynamically learns these MAC addresses from the source address of Ethernet frames that travel through its ports, such as ARP response packets. (Valadas, 2022)

A CAM table overflow is a malicious attack based on the fact that a switch can only memorise a limited number of MAC addresses. If a switch runs out of space, at some point, it drops old addresses for new ones, losing track of actual non-malicious devices on the network. At this point, frames destined for a particular device need to be flooded through all active ports in that same VLAN to reach the correct device eventually (HUB behaviour). That means that an attacker can see every frame the switch forwards.

Network Architecture

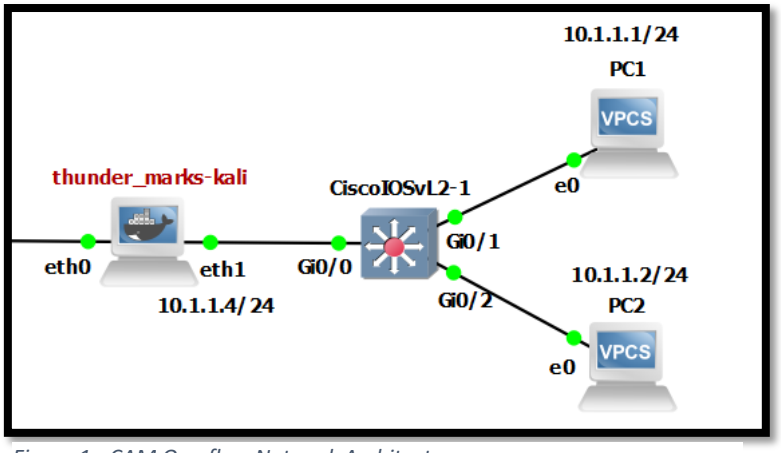


Figure 1 - CAM Overflow Network Architecture

In this network, we use a Cisco IOSvL2 to connect two victim PCs, *PC1* and *PC2*, and the *thunder_marks-kali* machine representing the attacker. To configure this architecture and prepare for the attack, it was required to configure the IPs of all three machine interfaces, as illustrated in figure 1.

Proof of Concept

This attack aims to saturate the MAC address table of an IOSvL2 switch; this will force the switch to work as a HUB and allow us to listen to any frames between its ports.

1. To start the attack, the malicious user starts eavesdropping using Wireshark. At this stage, it can only receive frames destined for itself and the broadcast address.

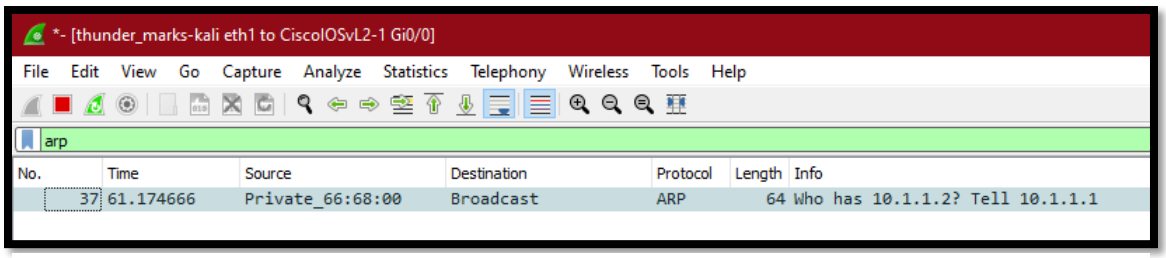


Figure 2 - PC1 tries to communicate with PC2 for the first time

2. At first, the switch does not know any MAC addresses resulting in an empty CAM table.

```
Switch#show mac address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
Switch#
```

Figure 3 - Empty Cam table

When PC1 communicates with PC2, PC1 sends an ARP Request to discover PC2's MAC address. As a result, the switch will learn the origin MAC address of PC1 and flood the request through the other ports. When PC2 responds to the request, the switch learns its MAC address resulting in the cam table in the figure below.

```
Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1    0050.7966.6800    DYNAMIC     Gi0/1
   1    0050.7966.6801    DYNAMIC     Gi0/2
Total Mac Addresses for this criterion: 2
Switch#
```

Figure 4 - CAM table with PC1 & PC2

3. After this stage, if PC1 pings PC2 or the reverse, the attacker does not capture such packets.

4. Attacker executes the script **camov.py** provided and starts overflowing the CAM table with random MAC addresses.

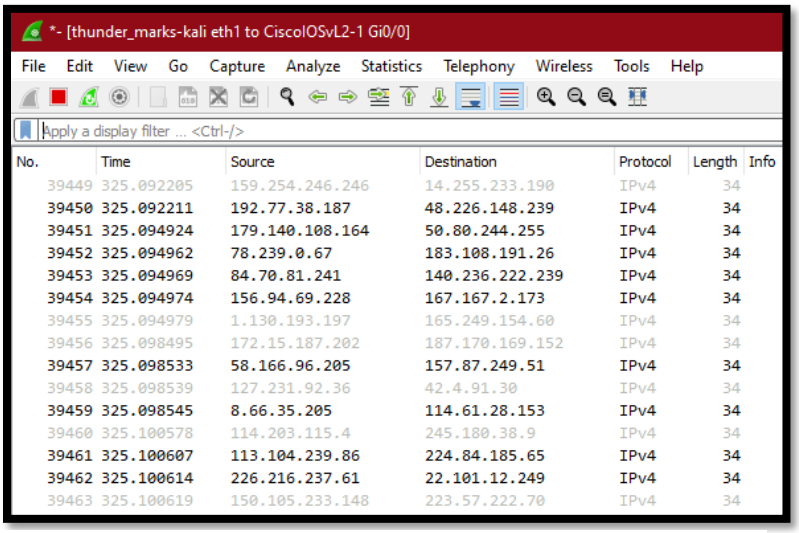


Figure 5 - Attack from Attacker's perspective

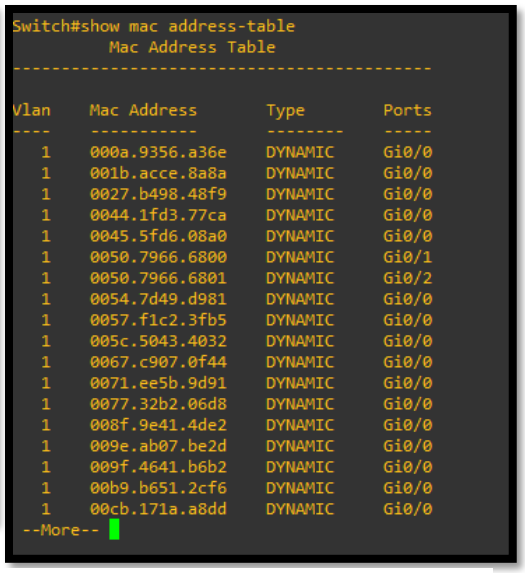


Figure 6 - Attack from the Switch perspective

5. After the CAM table is full, the switch starts behaving as a HUB, flooding all packets through all ports allowing the attacker to eavesdrop on any communications between PC1 and PC2. However, this result is not observed due to the possible size of the MAC address table of the IOSvL2 switch. In a more extensive network, there would be the possibility of infecting other machines and using them to attack the switch simultaneously. Still, it is challenging to do it in this scenario.

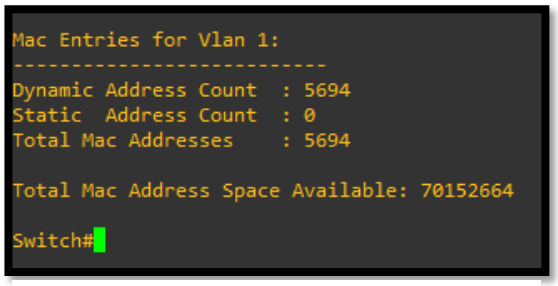


Figure 7 - MAC address table count for IOSvL2 switch

Mitigations

Even though the IOSvL2 switch allows a high number of registered MAC addresses in the CAM table, in a more complex network, attackers might have access to other machines and use them to attack them simultaneously. Port security must be configured to limit the number of MAC addresses injected into the switch. Port Security must be configured in all switch ports. The configuration was the following:

```
1.  Switch(config)# interface <Int>
2.  Switch(config-if)# switchport port-security
3.  Switch(config-if)# switchport port-security maximum <Number of MAC addresses allowed to>
4.  Switch(config-if)# switchport port-security violation shutdown
5.  Switch(config-if)#end
```

Feasibility

This attack can be complex if the switches have large memory addressed for the CAM tables with a low ageing timer. This makes it very difficult to overflow a table before the timer ends.

Although hard to achieve, it is easy to understand the attack, and the countermeasures are also simple to implement and very effective.

1.2. DHCP Spoofing

The DHCP Protocol is based on a connectionless service model through the User Datagram Protocol (UDP), used to assign IP addresses dynamically. It comprises four stages; in the first stage, called the **DISCOVER** phase, the Client broadcasts a request to the DHCP server requesting IP parameters. Then, the DHCP server replies with a DHCP **OFFER**, directly via UDP unicast, with an available IP address. In the third **REQUEST** stage, the Client sends a broadcast message requesting the previously offered address parameters to the server. Lastly, in the **ACKNOWLEDGE** stage, the DHCP server receives the DHCP REQUEST and sends a DHCP ACK packet to the Client, containing the lease duration and the IP address, completing the configuration process.

A DHCP Spoofing attack occurs when a malicious actor pretends to be a DHCP server to respond to legit DHCP requests. There are several possible consequences for the user that issues the request; the attacker can become a man-in-the-middle by changing the default gateway IP on the victim to its own IP and forward the following packets to the real gateway, and therefore remain hidden. It can provide a false default gateway IP address causing a Denial-of-Service, or even change the DNS server provided by the genuine DHCP server and redirect the victim to false and malicious websites. (Cisco, 2022)

Network Architecture

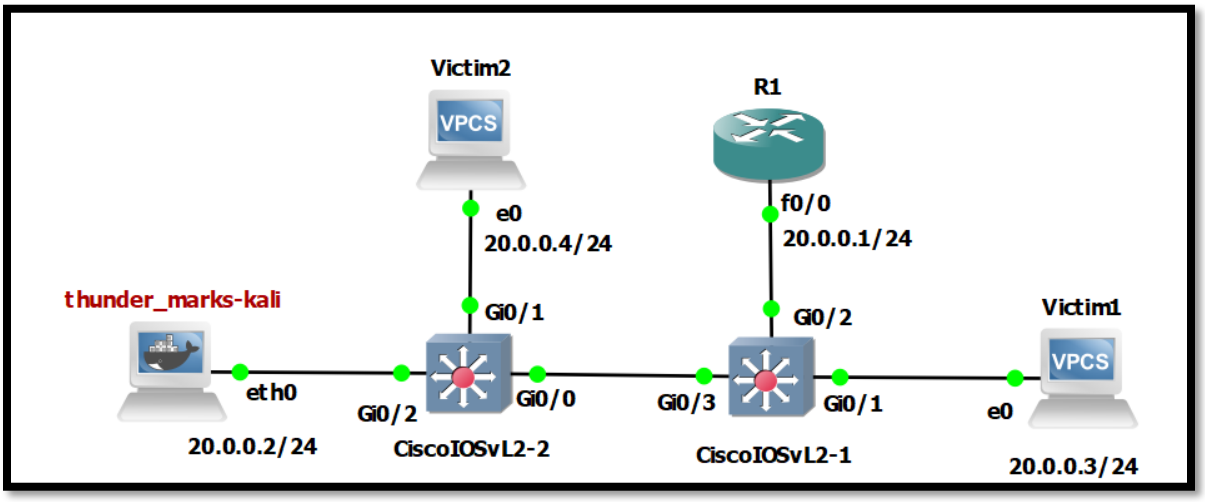


Figure 8 - DHCP Spoofing Network Architecture

In this network, we have two VPCS as victim PCs, connected to two Cisco IOSvL2 switches, a Cisco 3725 router as the legitimate DHCP server and a Docker attacker as the rogue DHCP server. (Valadas, 2022)

Proof of Concept

In this attack, we have the objective of changing the victim's gateway IP address to the attacker's IP address. Since the success of this attack depends on a racing condition between the attacker and the DHCP server (R1), we have designed this network with two victims, one with high chances of being compromised (Victim 2) and another with low chances (Victim 1).

1. The first step is to configure the Cisco router as a DHCP server. The commands for such a configuration can be found in more detail in Annex A.

```
R1#show ip dhcp pool

Pool 0 :
 Utilization mark (high/low)    : 100 / 0
 Subnet size (first/next)       : 0 / 0
 Total addresses                : 254
 Leased addresses               : 1
 Pending event                  : none
 1 subnet is currently in the pool :
 Current index        IP address range                    Leased addresses
 20.0.0.3             20.0.0.1         - 20.0.0.254        1
R1#
```

Figure 9 - R1 DHCP pool after configuration

2. Then activate DHCP IP in both victim PCs.

```
Victim1> ip dhcp
DDORA
Victim1> show ip IP 20.0.0.3/24

NAME        : Victim1[1]
IP/MASK     : 20.0.0.3/24
GATEWAY     : 0.0.0.0
DNS         : 1.1.1.1
DHCP SERVER : 20.0.0.1
DHCP LEASE  : 86397, 86400/43200/75600
MAC         : 00:50:79:66:68:00
LPORT       : 20044
RHOST:PORT  : 127.0.0.1:20045
MTU         : 1500
```

Figure 11 - Victim 1 getting legit IP

```
Victim2> ip dhcp
DDORA
Victim2> show ip IP 20.0.0.4/24

NAME        : Victim2[1]
IP/MASK     : 20.0.0.4/24
GATEWAY     : 0.0.0.0
DNS         : 1.1.1.1
DHCP SERVER : 20.0.0.1
DHCP LEASE  : 86397, 86400/43200/75600
MAC         : 00:50:79:66:68:01
LPORT       : 20046
RHOST:PORT  : 127.0.0.1:20047
MTU         : 1500
```

Figure 10 - Victim 2 getting legit IP

```
R1#show ip dhcp binding
Bindings from all pools not associated with VRF:
IP address          Client-ID/              Lease expiration        Type
                    Hardware address/
                    User name
20.0.0.2            01b2.c6d4.750f.17       Mar 02 2002 12:33 AM    Automatic
20.0.0.3            0100.5079.6668.00       Mar 02 2002 12:56 AM    Automatic
20.0.0.4            0100.5079.6668.01       Mar 02 2002 12:52 AM    Automatic
R1#
```

Figure 12 - DHCP binding of the 3 Machines

3. After all PCs in the network have their IP addresses, we set up the attack on the Docker attacker using Ettercap. **Command:** *ettercap -T -M dhcp:20.0.0.0/255.255.255.0/1.1.1.1* While Ettercap is running, we issue a new DHCP request from victim 2, where the attacker wins the race, due to the proximity in the network, against the router and become a malicious DHCP server.

```
521 412.876090    0.0.0.0          255.255.255.255      DHCP      406 DHCP Discover - Transaction ID 0xe1fcba66
522 412.879998    20.0.0.2         255.255.255.255      DHCP      582 DHCP Offer    - Transaction ID 0xe1fcba66
524 413.870283    0.0.0.0          255.255.255.255      DHCP      406 DHCP Request  - Transaction ID 0xe1fcba66
525 413.872065    20.0.0.2         255.255.255.255      DHCP      582 DHCP ACK      - Transaction ID 0xe1fcba66
```

Figure 13 - Wireshark capture of DHCP between victim 2 and attacker

```
Sat May 14 13:24:42 2022 [642454]
UDP  0.0.0.0:68 --> 255.255.255.255:67 |  (364)
......^.....................Pyfh.....................................................................
.....................................................................................c.Sc5....Vic
tim2=...Pyfh.....................................................................................DHCP: [0
0:50:79:66:68:01] DISCOVER
DHCP spoofing: fake OFFER [00:50:79:66:68:01] offering 20.0.0.0


Sat May 14 13:24:42 2022 [650308]
UDP  20.0.0.2:67 --> 255.255.255.255:68 |  (540)
......^.....................Pyfh.....................................................................
.....................................................................................c.Sc5..6....
.3.....................................................................................................
.....................................................................................................
.....................................DHCP: [20.0.0.2] OFFER : 20.0.0.0 255.255.255.0 GW 20.0.0.2 DNS 1.1.1.1


Sat May 14 13:24:43 2022 [635531]
UDP  0.0.0.0:68 --> 255.255.255.255:67 |  (364)
......^.....................Pyfh.....................................................................
.....................................................................................c.Sc5..6....
.2.....=...Pyfh...Victim27.....................................................................DHCP: [0
0:50:79:66:68:01] REQUEST 20.0.0.0
DHCP spoofing: fake ACK [00:50:79:66:68:01] assigned to 20.0.0.0
SEND L3 ERROR: 392 byte packet (0800:11) destined to 255.255.255.255 was not forwarded (libnet_write_raw_ipv4(): -1 bytes wr
itten (Network is unreachable)
)


Sat May 14 13:24:43 2022 [642299]
UDP  20.0.0.2:67 --> 255.255.255.255:68 |  (540)
......^.....................Pyfh.....................................................................
.....................................................................................c.Sc5..6....
.3.....................................................................................................
.....................................................................................................
.....................................DHCP: [20.0.0.2] ACK : 20.0.0.0 255.255.255.0 GW 20.0.0.2 DNS 1.1.1.1


Sat May 14 13:25:54 2022 [427124]
```

Figure 14 - Ettercap attack

```
Victim2> show ip

NAME        : Victim2[1]
IP/MASK     : 20.0.0.4/24
GATEWAY     : 20.0.0.2
DNS         : 1.1.1.1
DHCP SERVER : 20.0.0.1
DHCP LEASE  : 86394, 86400/43200/75600
MAC         : 00:50:79:66:68:01
LPORT       : 20046
RHOST:PORT  : 127.0.0.1:20047
MTU         : 1500
```

Figure 15 - Victim 2 Gateway changed

4. On the other hand, the same attack on victim 1 is unsuccessful. Because the DHCP server is closer to Victim 1 than the attacker, the legitimate server wins the race and is able to respond faster to the DHCP Request.

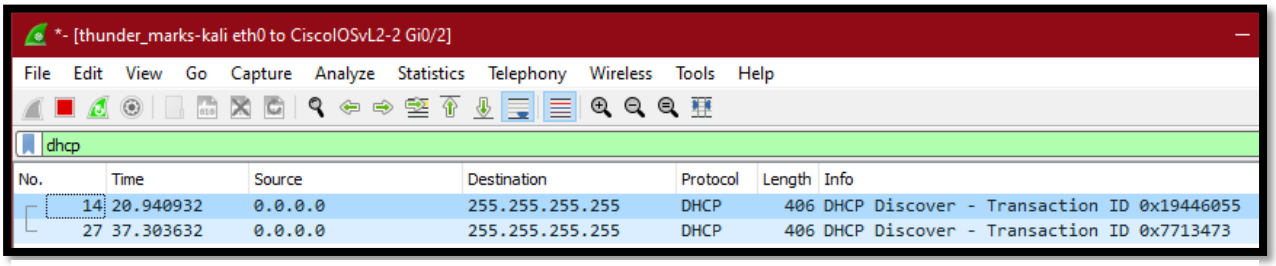


Figure 16 - Wireshark capture of the request from victim 1

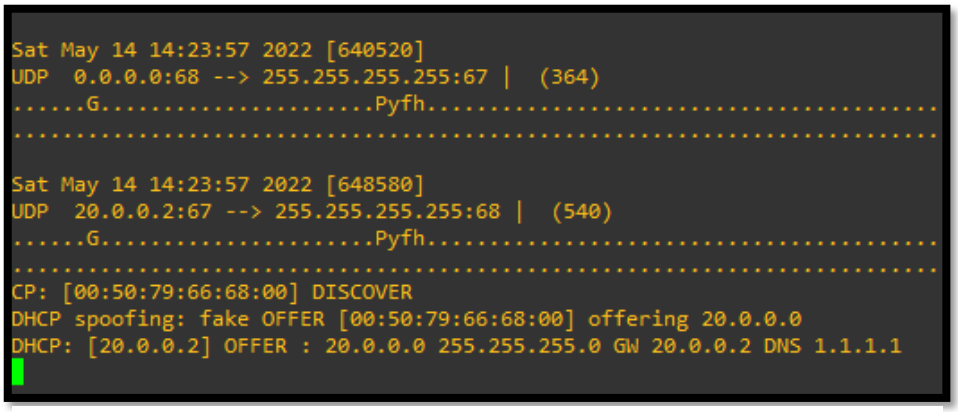


Figure 17 - Failed ettercap attack

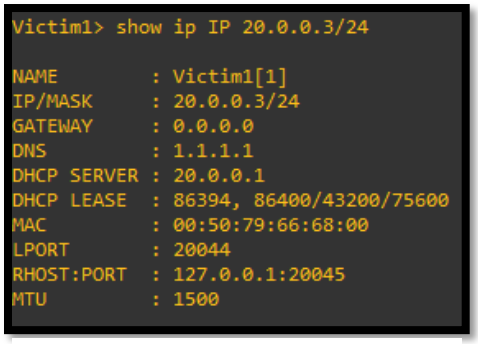


Figure 18 - Victim 1 intact gateway address

Mitigations

DHCP Spoofing attacks can be mitigated using **DHCP snooping** on trusted ports. When DHCP snooping is enabled on an interface, the switch compares the source packet information against the DHCP snooping binding table, denying packets containing:

- Unauthorized DHCP server messages from an untrusted port.
- Unauthorized DHCP client messages not adhering to the snooping binding table or rate limits.
- DHCP relay-agent packets that include option-82 information on an untrusted port.

In our topology, all ports connected to end-user devices should be considered untrusted ports.

1. First, we enable DHCP Snooping using the **ip dhcp snooping** command on both Cisco IOSvL2 switches with the command **no ip dhcp snooping information option** to disable the insertion of option 82 in the DHCP messages so that normal clients can still get IP addresses.
2. On the trusted ports Gi0/2 and Gi0/3 of switch 1 and Gi0/0 of switch 2 we use the command **ip dhcp snooping trust.**
3. On untrusted ports we also use the command **ip dhcp snooping limit rate** to limit the rate of received DHCP discovery messages per second.

```
Switch#show ip dhcp snooping
Switch DHCP snooping is enabled
Switch DHCP gleaning is disabled
DHCP snooping is configured on following VLANs:
1
DHCP snooping is operational on following VLANs:
1
DHCP snooping is configured on the following L3 Interfaces:

Insertion of option 82 is disabled
   circuit-id default format: vlan-mod-port
   remote-id: 0ce8.9c5f.0000 (MAC)
Option 82 on untrusted port is not allowed
Verification of hwaddr field is enabled
Verification of giaddr field is enabled
DHCP snooping trust/rate is configured on the following Interfaces:

Interface                  Trusted    Allow option    Rate limit (pps)
-----------------------    -------    ------------    ----------------
GigabitEthernet0/1         no         no              6
  Custom circuit-ids:
GigabitEthernet0/2         yes        yes             unlimited
  Custom circuit-ids:
GigabitEthernet0/3         yes        yes             unlimited
Interface                  Trusted    Allow option    Rate limit (pps)
-----------------------    -------    ------------    ----------------
```

Figure 19 - Switch 1 DHCP snooping configuration

```
Switch#show ip dhcp snooping
Switch DHCP snooping is enabled
Switch DHCP gleaning is disabled
DHCP snooping is configured on following VLANs:
1
DHCP snooping is operational on following VLANs:
1
DHCP snooping is configured on the following L3 Interfaces:

Insertion of option 82 is disabled
   circuit-id default format: vlan-mod-port
   remote-id: 0c52.ad74.0000 (MAC)
Option 82 on untrusted port is not allowed
Verification of hwaddr field is enabled
Verification of giaddr field is enabled
DHCP snooping trust/rate is configured on the following Interfaces:

Interface                  Trusted    Allow option    Rate limit (pps)
-----------------------    -------    ------------    ----------------
GigabitEthernet0/0         yes        yes             unlimited
  Custom circuit-ids:
GigabitEthernet0/1         no         no              6
  Custom circuit-ids:
GigabitEthernet0/2         no         no              6
Interface                  Trusted    Allow option    Rate limit (pps)
-----------------------    -------    ------------    ----------------
```

Figure 20 - Switch 2 DHCP Snooping configuration

```
Victim2> ip dhcp
DDORA
Victim2> show ip IP 20.0.0.4/24

NAME        : Victim2[1]
IP/MASK     : 20.0.0.4/24
GATEWAY     : 0.0.0.0
DNS         : 1.1.1.1
DHCP SERVER : 20.0.0.1
DHCP LEASE  : 86397, 86400/43200/75600
MAC         : 00:50:79:66:68:01
LPORT       : 20046
RHOST:PORT  : 127.0.0.1:20047
MTU         : 1500
```

Figure 21 - Victim 2 IP from legit DHCP server with Ettercap running

By activating DHCP snooping the attacker can no longer see or respond to the DHCP requests from the victims and therefore the attack is prevented since the victims can only receive DHCP parameters from the legitimate trusted DHCP Server. Such configurations are present in the Figures above.

The *Cisco Nexus 9000 Series* is an example of devices that support DHCP snooping on the other hand the much cheaper *HP J9779A* does not support such a feature.

Feasibility

The attack is quite simple to execute and easy to understand if the concepts behind the DHCP protocol are well known. For a successful attack, the attacker is only required to gain access to the network and, specifically, to a port without DHCP Spoofing. Of course, the attacker's position in the network is also important to win the race against the legitimate DHCP server. The attack becomes problematic if the DHCP server is closer to the victims than the attacker.

1.3. ARP Poisoning (MitM)

The Address Resolution Protocol (ARP) is used to map IP addresses against the respective MAC addresses. When computer A wants to communicate with computer B, it checks its ARP table for the respective MAC address against the IP address of computer B. If B's MAC address is already in the table, A sends a frame containing the IP Packet to B. If not, A has to send a broadcast **ARP REQUEST** requesting an answer for the MAC addresses of B. B then responds to the message with an **ARP REPLY** containing its MAC and IP address. If not there already, B can add A's addresses to its own ARP table. Now that both have each other's MAC and IP addresses they can exchange packets.

ARP poisoning is an attack to the address resolution protocol, based on the fact that the previously explained protocol allows clients to send unsolicited ARP Replies called "Gratuitous ARP". When a client sends a gratuitous ARP other hosts on the same subnet store the MAC and IP addresses contained in it, in their ARP tables. (Cisco, 2022)

In this case the attacker sends unsolicited ARP replies to other hosts with the MAC address of the attacker and the IP of the default gateway and sends ARP replies to the default gateway with the MAC address of the attacker and the IP of the victim. By doing this the attacker can now redirect traffic between the default gateway and the victim becoming a man-in-the-middle. This is merely one of the many possibilities of an ARP poisoning attack.

Network Architecture

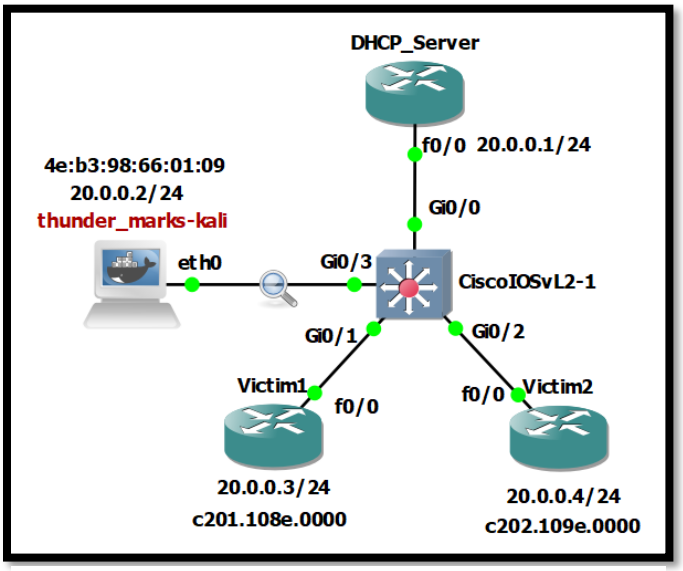


Figure 22 - Network Architecture ARP Poisoning

In this attack, we used two C3725 routers configured as two host victims (Victims 1 and 2). One IOSvL2 that connects all devices on the subnet, a C3725 router as a DHCP server that provides all the respective IP addresses and a Docker attacker.

Proof of Concept

This attack has the objective of using ARP poisoning to establish a MitM attack between the Docker attacker and the two victims. We used the *arpmitm.py* script provided in the lectures to perform the attack. The respective configuration commands are located in Annex A.

1. First, we ping Victim2 from Victim 1 to populate both ARP tables.

```
Victim1#ping 20.0.0.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.0.0.4, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 16/31/48 ms
Victim1#sh arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  20.0.0.4                0   c202.109e.0000  ARPA   FastEthernet0/0
Internet  20.0.0.3                -   c201.108e.0000  ARPA   FastEthernet0/0
Victim1#
```

Figure 23 - Victim 1 ARP table

```
Victim2#sh arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  20.0.0.4                -   c202.109e.0000  ARPA   FastEthernet0/0
Internet  20.0.0.3                0   c201.108e.0000  ARPA   FastEthernet0/0
Victim2#
```

Figure 24 - Victim 2 ARP table

2. The attacker starts the attack, using the *arpmitm.py* script, and we captured the packets using Wireshark in order to better understand the execution of this attack.

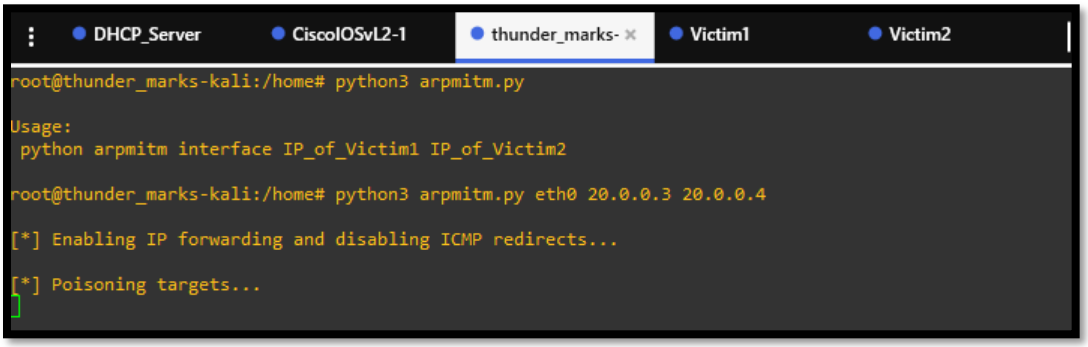


Figure 25 - Execution of the attack

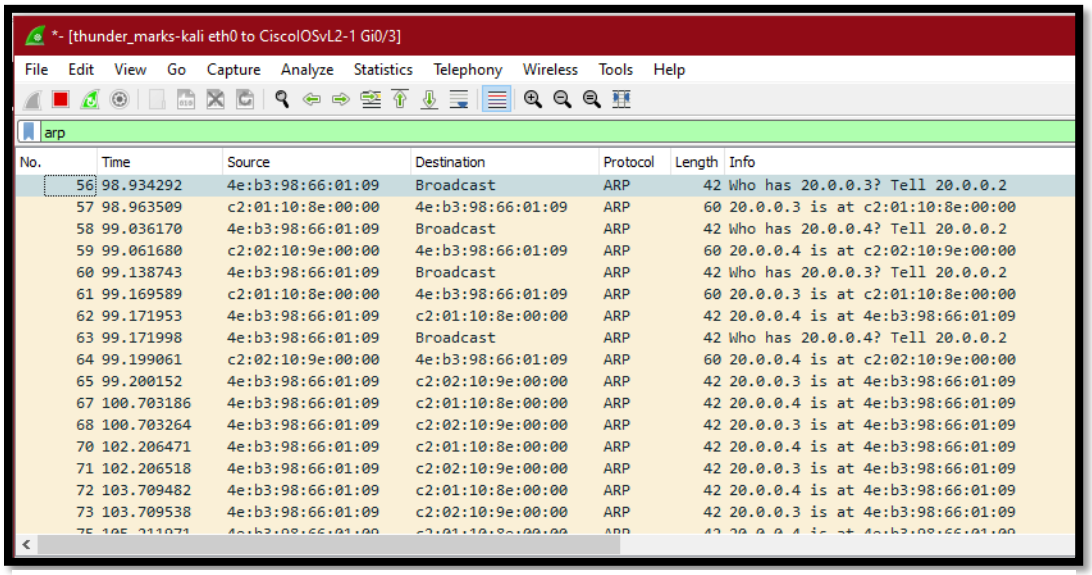


Figure 26 - Gratuitous ARP messages in the attack

```
Victim1#sh arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  20.0.0.4               0   4eb3.9866.0109  ARPA   FastEthernet0/0
Internet  20.0.0.3               -   c201.108e.0000  ARPA   FastEthernet0/0
Victim1#
```

Figure 27 - Victim 1 ARP table with the MAC address of the attacker on the IP of Victim 2

```
Victim2#sh arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  20.0.0.4               -   c202.109e.0000  ARPA   FastEthernet0/0
Internet  20.0.0.3               0   4eb3.9866.0109  ARPA   FastEthernet0/0
Victim2#
```

Figure 28 - Victim 2 ARP table with the MAC address of the attacker on the IP of Victim 1

3. Then Victim 1 Pings Victim 2, and the attacker can intercept and redirect all packets, becoming man-in-the-middle and resulting in a successful attack.

```
Victim1#ping 20.0.0.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.0.0.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 36/53/80 ms
Victim1#
```

Figure 29 - Victim 1 pings Victim 2 after being attacked

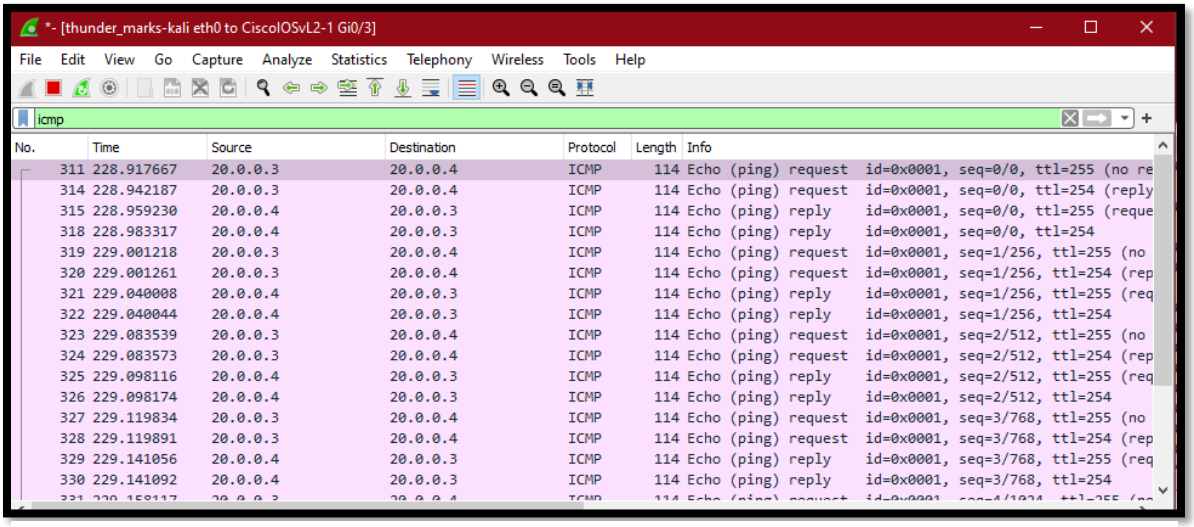


Figure 30 - Attacker captures and redirects the ICMP packets from Victim 1 to Victim 2

Mitigations

To prevent this type of attack, the switch connecting all the devices should not accept ARP replies from an attacker and therefore must ensure that only valid ARP requests and replies are relayed.

The solution is to activate **Dynamic ARP Inspection** (DAI), which inspects all ARP Requests and replies on the untrusted ports. Each interception is verified for a valid IP to MAC address mapping. Resulting in ARP replies from invalid devices being either dropped or logged by the switch for auditing. (Cisco, 2022)

DAI determines the validity of an ARP packet based on a valid MAC/IP address pair stored on a database built by DHCP Snooping, resulting in this last feature being a requirement for DAI.

1. We first apply the required countermeasures by activating both DHCP snooping and DAI on the switch, considering only the DHCP Server port as a trusted port.

```
Switch(config)#no cdp advertise-v2
Switch(config)#ip dhcp snooping
Switch(config)#no ip dhcp snooping information option
Switch(config)#ip dhcp snooping vlan 1
Switch(config)#ip arp inspection vlan 1
Switch(config)#int gi0/0
Switch(config-if)#ip dhcp snooping trust
Switch(config-if)#ip arp inspection trust
Switch(config-if)#int range gi0/1 - 3
Switch(config-if-range)#ip dhcp snooping limit rate 6
Switch(config-if-range)#exit
Switch(config)#ip arp inspection validate src-mac dst-mac ip
Switch(config)#
```

Figure 31 - Applying the countermeasures on the switch

2. Then we execute the attack again and capture the packets on Wireshark.

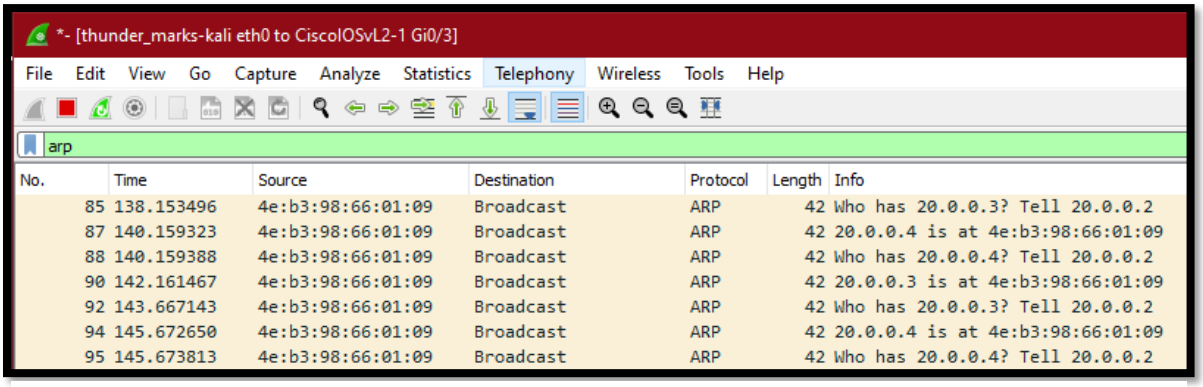


Figure 32 - Capture of ARP messages during the attack

3. As we can see, gratuitous ARP messages do not go through the switch, and the ARP table on the victims is not altered. Also, in the switch, we receive log messages about the attack attempt and the invalid ARP requests being denied.

```
Victim1#sh arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  20.0.0.4               73   c202.109e.0000  ARPA   FastEthernet0/0
Internet  20.0.0.3                -   c201.108e.0000  ARPA   FastEthernet0/0
Victim1#
```

Figure 33 - Victim 1 ARP table with the correct IP-MAC addresses

```
0.0000/20.0.0.4/15:08:02 UTC Sun May 15 2022])
*May 15 15:08:04.746: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/3, vlan 1.([4eb3.9866.0109/20.0.0.3/0000.00
0.0000/20.0.0.4/15:08:04 UTC Sun May 15 2022])
*May 15 15:08:04.746: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Req) on Gi0/3, vlan 1.([4eb3.9866.0109/20.0.0.2/0000.00
0.0000/20.0.0.4/15:08:04 UTC Sun May 15 2022])
*May 15 15:08:07.121: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/3, vlan 1.([4eb3.9866.0109/20.0.0.3/0000.00
0.0000/20.0.0.4/15:08:06 UTC Sun May 15 2022])
*May 15 15:08:07.122: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Req) on Gi0/3, vlan 1.([4eb3.9866.0109/20.0.0.2/0000.00
0.0000/20.0.0.4/15:08:06 UTC Sun May 15 2022])
*May 15 15:08:08.127: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/3, vlan 1.([4eb3.9866.0109/20.0.0.3/0000.00
0.0000/20.0.0.4/15:08:08 UTC Sun May 15 2022])
*May 15 15:08:08.127: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Req) on Gi0/3, vlan 1.([4eb3.9866.0109/20.0.0.2/0000.00
0.0000/20.0.0.4/15:08:08 UTC Sun May 15 2022])
*May 15 15:08:10.425: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/3, vlan 1.([4eb3.9866.0109/20.0.0.3/0000.00
0.0000/20.0.0.4/15:08:09 UTC Sun May 15 2022])
```

Figure 34 - Log messages on the switch

Feasibility

To be successful at this attack, the malicious actor only requires the victim's IP address, which can become complex to obtain due to the fact that the attacker required access to the same subnetwork as the victim. The knowledge behind the attack is straightforward since it is only dependent on the ARP protocol properties. The countermeasures are also simple to apply if the defender has a good perception of the network's architecture and if well applied, a successful attack is not possible. Since this mitigation requires DHCP Snooping we the example of switches that support DAI are the same presented on the previous DHCP Spoofing attack.

1.4. Root Bridge spoofing (MitM)

Spanning Tree Protocol (STP) is a Layer 2 protocol that runs on bridges and switches. The specification for STP is IEEE 802.1D. The primary purpose of STP is to ensure that loops are not created when redundant paths are present in the network. (Cisco, 2022)

Electing the Root Bridge is one of the steps of the STP. The root bridge is the bridge with the lowest Bridge ID. All the decisions, like which ports are the root ports (the port with the best path to the root bridge), are made from the perspective of the root bridge. The Bridge ID is a combination of the bridge priority and the MAC address of the device. If switches have the same priority, the one with the lowest MAC address will be elected Root Bridge.

Root Bridge Spoofing attack consists of sending fake BPDU packets with a lower priority or lower MAC address to force STP recalculation so the attacker can become Root Bridge. By becoming the Root Bridge, the attacker can see some information exchanged between network elements. The attacker might not see all the traffic between the devices in the network since not all traffic goes through the Root Bridge.

In addition, the attacker can cause network instability through denial-of-service (DoS) by causing an interruption each time the root bridge changes.

Network Architecture

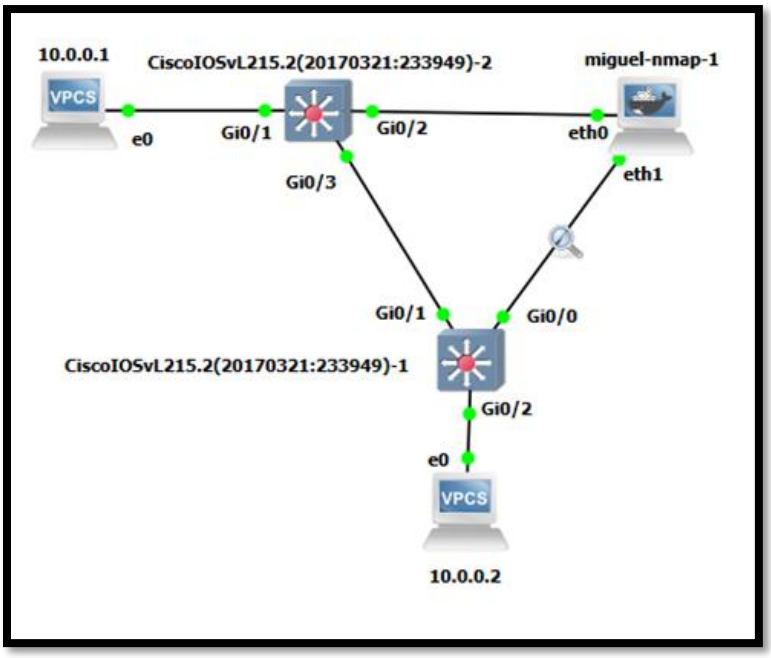


Figure 35 - Root Bridge Spoofing Network Architecture

In order to demonstrate the attack in the simplest possible network topology the network presented above is structured with two IOSvL2 switches one Docker attacker and two VPCS as the hosts.

Proof of Concept

1. As we can see from the previous figures, Switch 1 is the Root Bridge with priority 32769.

```
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    32769
             Address     0c31.3e46.0000
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0c31.3e46.0000
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  15  sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi0/0               Desg FWD 4         128.1    P2p
Gi0/1               Desg FWD 4         128.2    P2p
Gi0/2               Desg FWD 4         128.3    P2p
Gi0/3               Desg FWD 4         128.4    P2p
Gi1/0               Desg FWD 4         128.5    P2p
Gi1/1               Desg FWD 4         128.6    P2p
Gi1/2               Desg FWD 4         128.7    P2p
Gi1/3               Desg FWD 4         128.8    P2p

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------

Gi2/0               Desg FWD 4         128.9    P2p
Gi2/1               Desg FWD 4         128.10   P2p
Gi2/2               Desg FWD 4         128.11   P2p
Gi2/3               Desg FWD 4         128.12   P2p
Gi3/0               Desg FWD 4         128.13   P2p
Gi3/1               Desg FWD 4         128.14   P2p
Gi3/2               Desg FWD 4         128.15   P2p
Gi3/3               Desg FWD 4         128.16   P2p
```

Figure 37 - Switch 1 STP Information

```
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    32769
             Address     0c31.3e46.0000
             Cost        4
             Port        4 (GigabitEthernet0/3)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0cfc.db74.0000
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  15  sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi0/0               Desg FWD 4         128.1    P2p
Gi0/1               Desg FWD 4         128.2    P2p
Gi0/2               Desg FWD 4         128.3    P2p
Gi0/3               Root LRN 4         128.4    P2p
Gi1/0               Desg FWD 4         128.5    P2p
Gi1/1               Desg FWD 4         128.6    P2p
Gi1/2               Desg FWD 4         128.7    P2p
 --More--
```

Figure 36 - Switch 2 STP Information

2. To simulate the attack, we used the following command in the attacker ***python3 rbmitm.py eth0 eth1***. We set a **Wireshark** capture represented in the network topology in order to better understand how the attacker becomes Root Bridge.

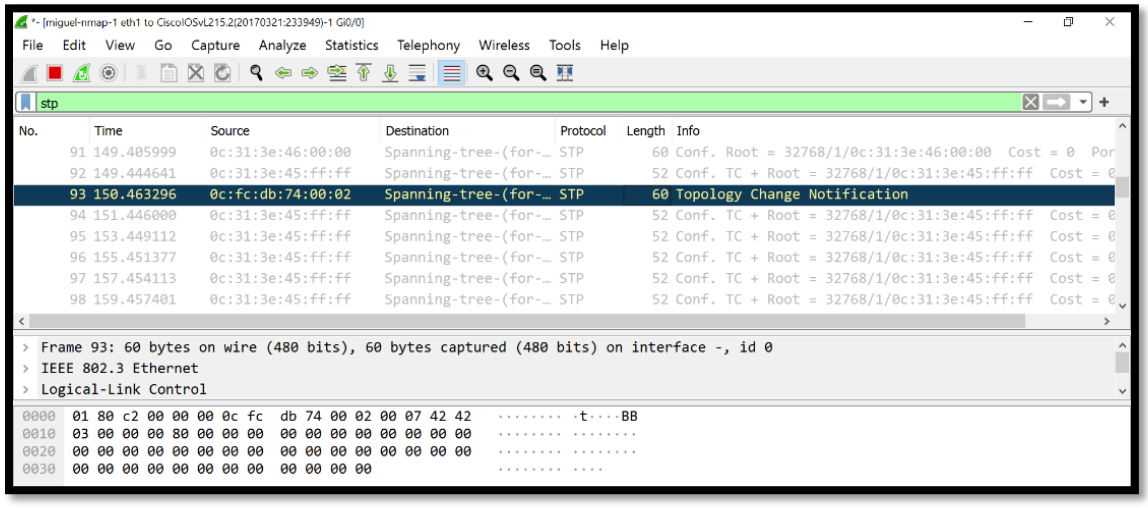


Figure 38 - Wireshark Capture

3. We can see an **STP configuration packet** sent by the attacker (highlighted in the previous screenshot) by analysing the capture. This packet indicates the same priority but a lower MAC address (0c31.3e45.ffff).
   When selecting the Root Bridge, if priorities are the same for all switches, the switch with lower MAC will be elected. In this case, by sending an STP packet with a lower MAC than the actual Root Bridge, the attacker will be elected Root Bridge.

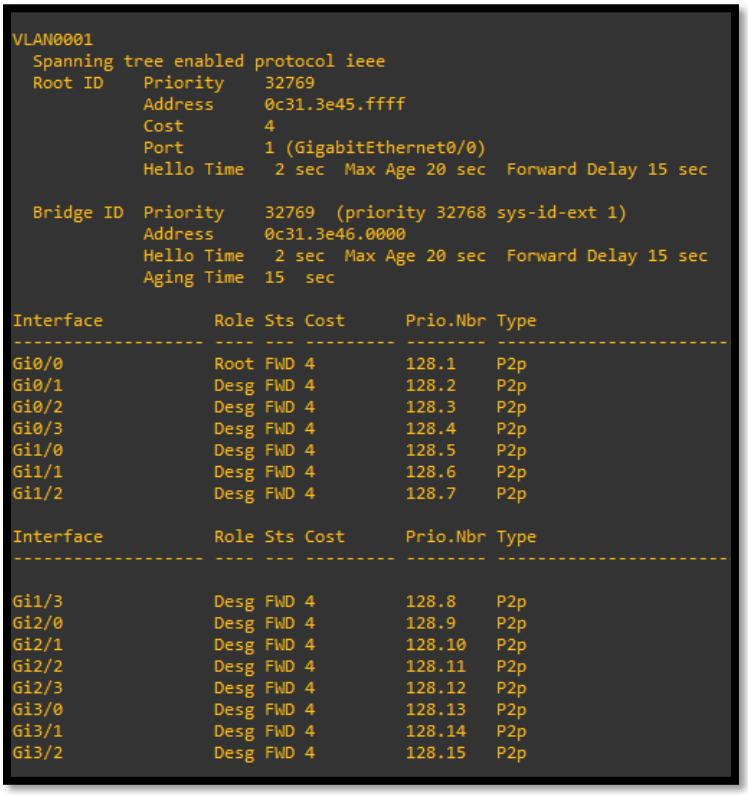


Figure 39 - Switch 1 new STP Information

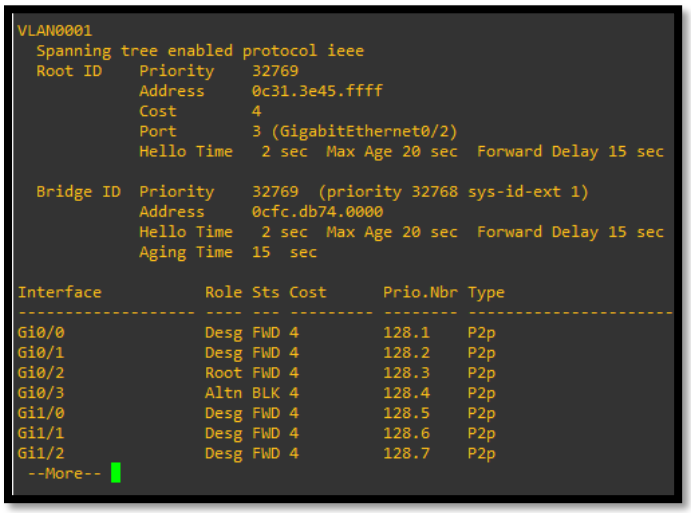


Figure 40 - Switch 2 new STP Information

4. Furthermore, we can see that the interfaces connected with the attacker **(SW1 – Gi0/0 and SW2 – Gi0/2)** are classified as Root ports. To confirm the attack was successful, we pinged PC2 from PC1, and we can see in the following screenshot that the attacker was able to see the information exchanged between PC1 and PC2. We can conclude that the attacker is now the Root Bridge.

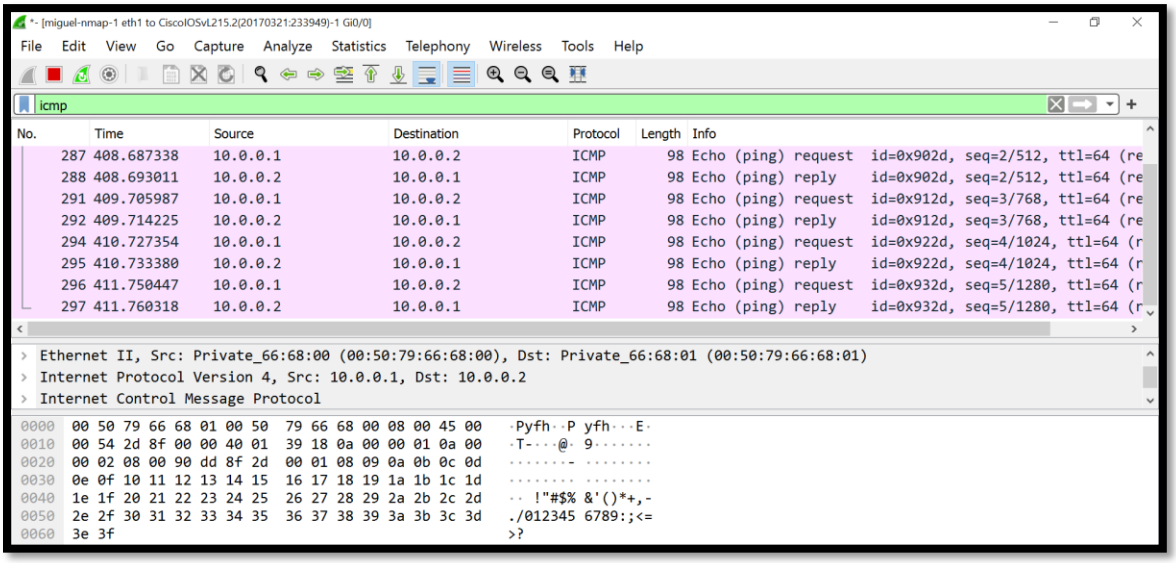


Figure 41 - ICMP Packages on Wireshark

Mitigations

To stop this attack, we should configure **BPDU Guard** on the switches' interfaces connected to end-users.

1. In this case, we will configure **BPDU Guard** on the switches ports that are connected to the attacker: **SW1 – Gi0/0 and SW2 – Gi0/2**. The configuration applied was the following:

```
Switch#show run int Gi0/0
Building configuration...

Current configuration : 103 bytes
!
interface GigabitEthernet0/0
 media-type rj45
 negotiation auto
 spanning-tree bpduguard enable
end
```

Figure 42 - Switch 1 after BPDU Guard configuration

2. We then tried to run the attack after the **BPDU Guard** configuration and obtained the following errors in both SW1 and SW2:

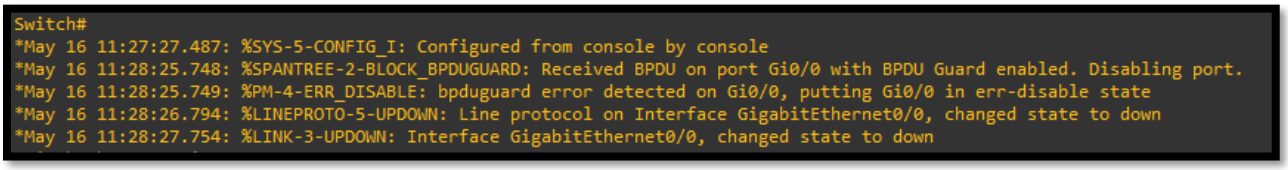


Figure 43 - Switch 1 response to attack after BPDU Guard configuration

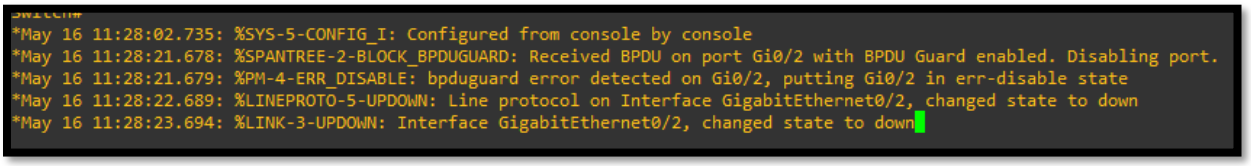


Figure 44 - Switch 2 response to attack after BPDU Guard configuration

Looking at the messages in previous figures, we can see that there were reported **BDPU packets** in the switch's interfaces connected to the attacker, so those interfaces will be disabled. This way the attacker is no longer able to interfere in the STP. From our research switches like Catalyst 4500 Series support BPDU guard features. On the other hand, the switch HP 1420-16G is an example of a model that does not support this feature.

Feasibility

In order to perform the attack, the attacker must be connected to the victim's local network at two switches. This is not always a simple task; the attack is relatively simple after being connected. However, the countermeasures are also simple to implement and effective if the network administrator can detect the switch's ports connected to the attacker.

1.5. DNS spoofing

DNS Spoofing attack consists in tricking the client into thinking that an attacker's website is an actual legitimate website. The attacker can do this through ARP poisoning techniques.

Network Architecture

To simulate this attack, we applied the following topology:

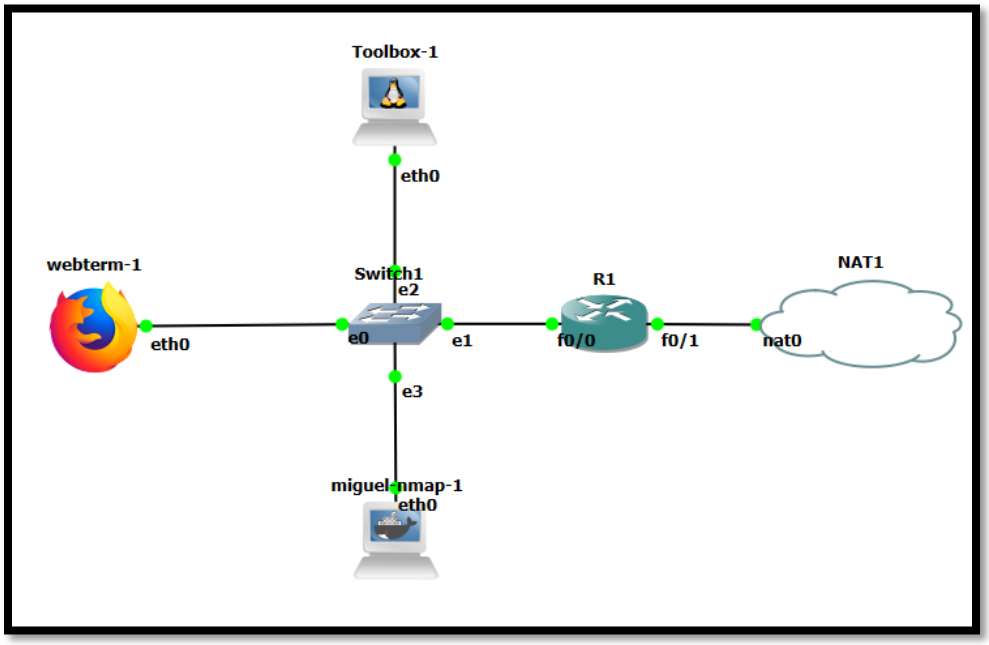


Figure 45 - DNS Spoofing Network Topology

Proof of Concept

After the initial network configuration, we started the fake web server. To perform the attack, we executed the script dnsspoof.py by running the command ***python3 dnsspoof.py eth0 biding.txt 20.0.0.0/24*** in the attacker, where **binding.txt** is the file containing the binding between the IP address of the fake Web server and the name of the Web site to be spoofed, in our case ***linkedin.com***.

A Wireshark capture was set up in the link between the Web Server and the SW1 so we can better understand how the attack takes place.

1. Firstly, the script will poison the subnet's ARP cache of the interfaces, allowing the attacker to become a "man-in-the-middle". We can see in the following image that the attacker was sending ARP replies to all interfaces.

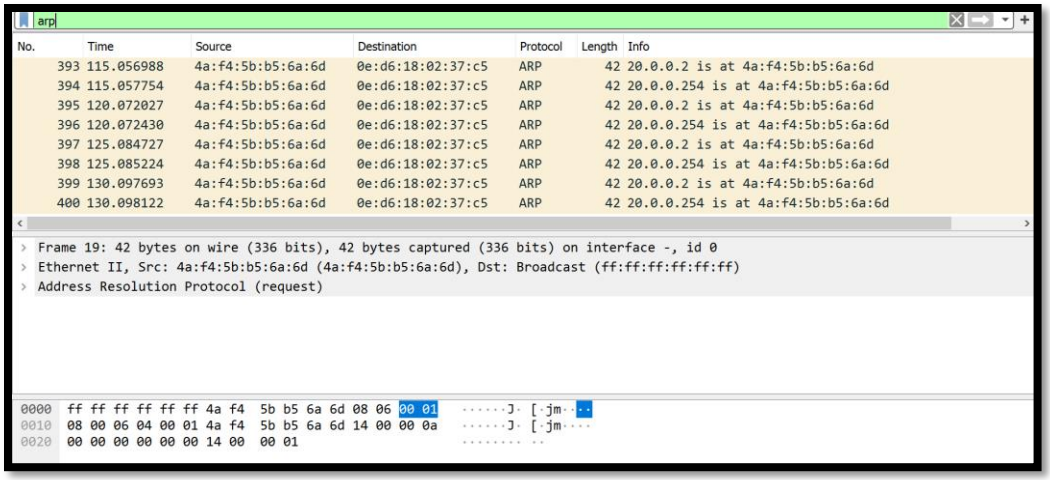


Figure 46 - ARP Replies on Wireshark

2. By performing the command ***arp –a*** on both the Web Server and the fake Web Server we can see the ARP table information:

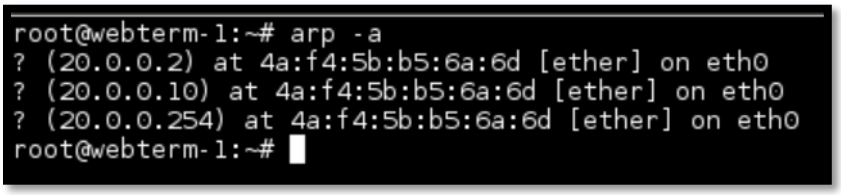


Figure 47 - ARP information on the Web Server

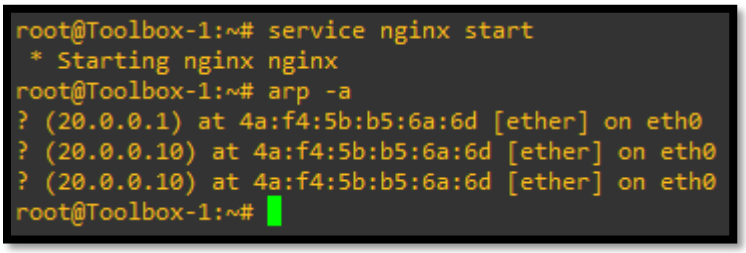


Figure 48 - ARP information on the Fake Web Server

3. We can see that both tables were successfully poisoned. When the client sends a DNS request looking for the IP of the site to be accessed, it will go through the attacker that will indicate the IP of the fake web server. The following images confirm this process:

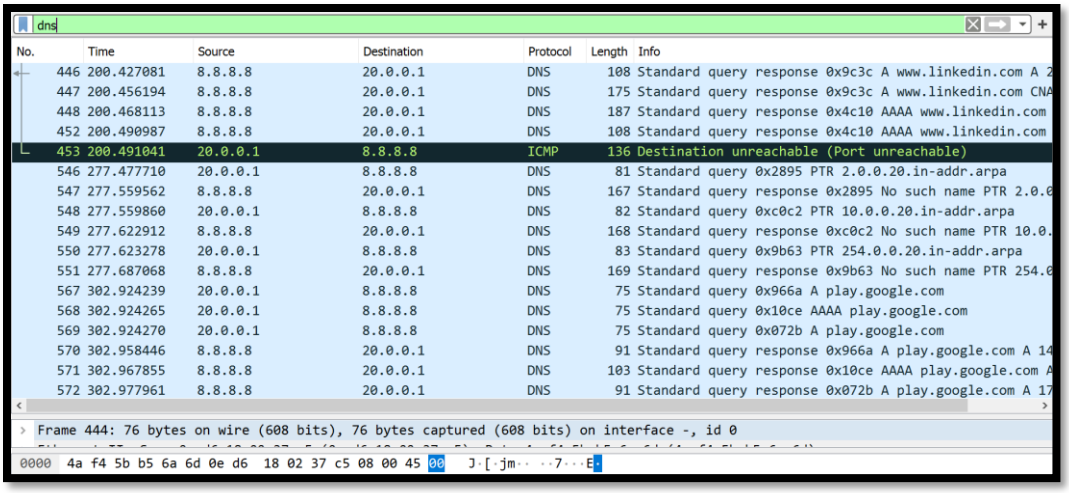


Figure 49 - DNS packets on Wireshark

4. Even though the DNS request seems to be sent to the IP 8.8.8.8 (Google DNS) we can see that the real destination is the attacker. This happens because the client's ARP cache was previously poisoned. In the response to the DNS Request, the attacker will send the IP of the fake Web Server as can be seen in the following figure (in the **Answers** field):

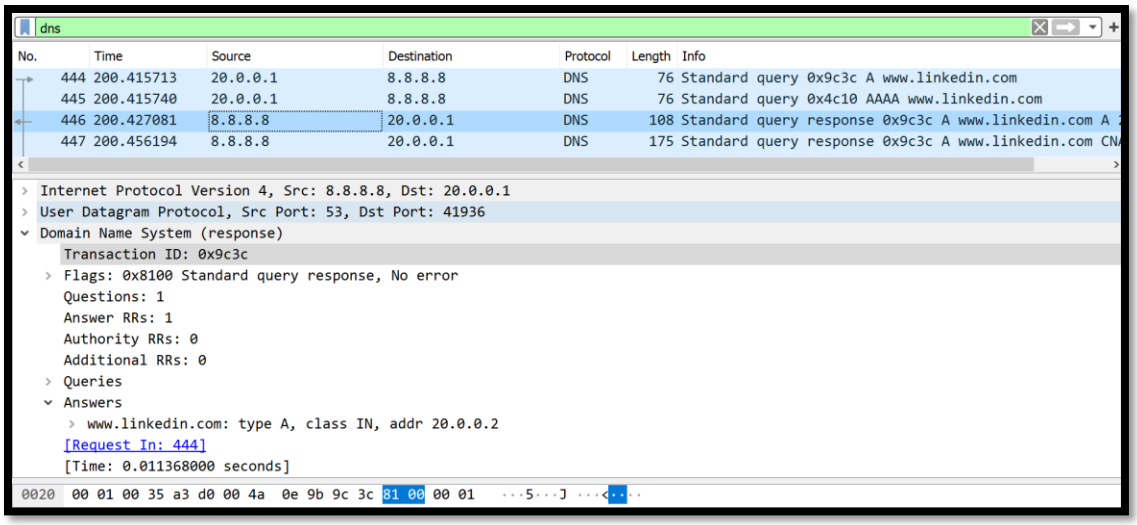


Figure 50 - DNS request details

5. The victim will now access the fake Web Server thinking it is the legitimate website.

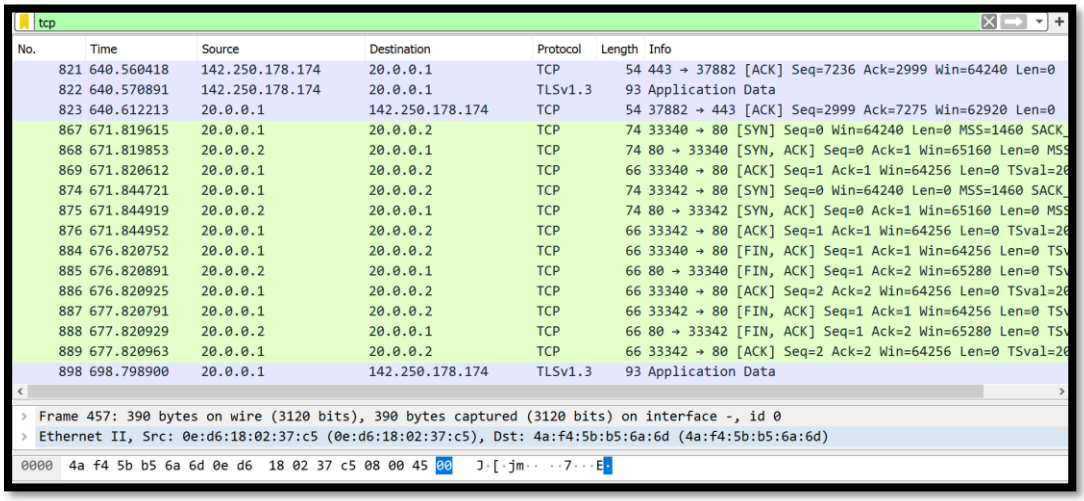


Figure 51 - TCP Packets on Wireshark

6. By analysing the previous figure, we can see the that the client and the fake Web Server exchange packets. We can now confirm that the client was the target to a DNS spoofing attack.

7. We can have additional confirmation by checking the browser's DNS cache:

Figure 52 - DNS Cache of Web Browser

As expected, the IP associated with LinkedIn is the IP of the fake Web Server.

Mitigations

As previously seen, this attack's base is the ARP poisoning of the interfaces in the subnet. Having this in mind, we can use the **Dynamic ARP Inspection** (DAI) explained in module 1.3 ARP Poisoning (MitM) as mitigation. In addition, there are technologies and protocols such as **Domain Name System Security (DNSSEC)** that make it harder for the attacker to be successful. DNSSEC is a DNS protocol that uses signed DNS requests to prevent forgery.

Feasibility

Regarding feasibility, a DNS Spoofing attack can be complex since it requires the attacker to be connected to the same network as the victim. In addition, several measures can be taken to prevent and mitigate these attacks, as seen previously.

1.6. RIP poisoning

Routing Information Protocol (RIP) is a dynamic routing protocol that uses hop count as a routing metric to find the best path between the source and the destination network. It is a distance-vector routing protocol and works at the Network layer of the OSI model. RIP uses port number 520.

RIP poisoning attack intends to redirect the traffic from a legit Web server to a fake Web server by sending fake RIP responses.

In this case, the attacker will poison the RIP table of the routers in the network, announcing a better route to the subnet where the legit Web server is. This way, when the client tries to access the legit Web server, the traffic will be redirected to the fake server. This is possible because RIP selects routes using the **longest prefix match routing rule**.

Network Architecture

In this demonstration we used two different topologies where we situate the attacker and the web server in different points of the network to better demonstrate the attack.

- **Topology 1:**

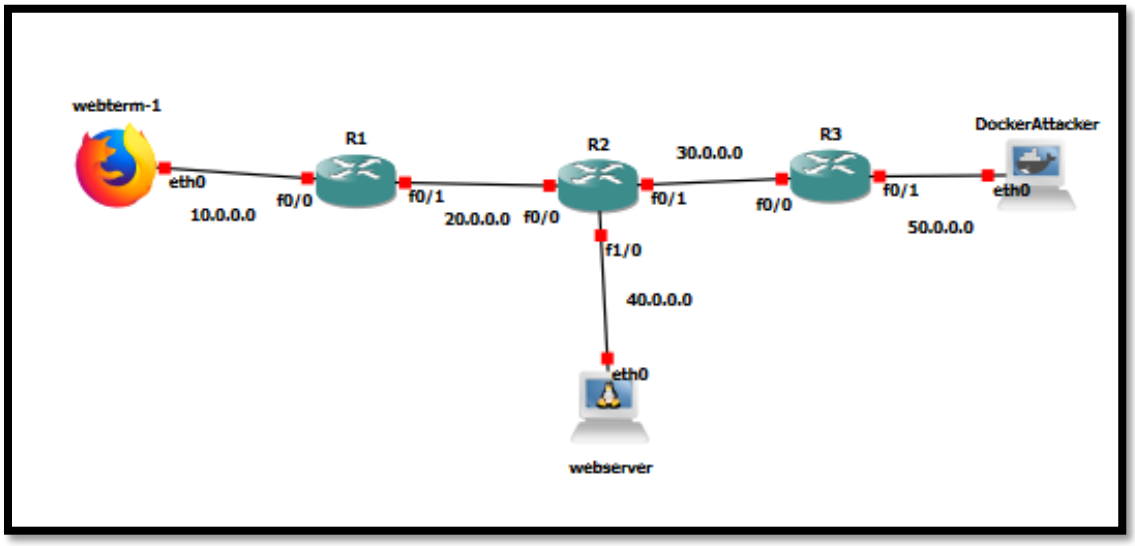


Figure 53 - RIP poisoning Network Topology 1

- **Topology 2:**

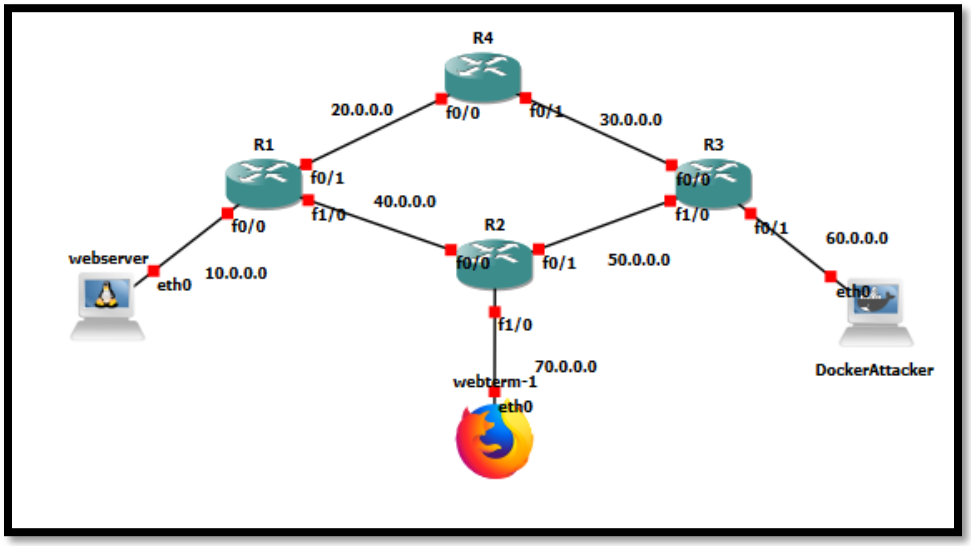


Figure 54 - RIP poisoning Network Topology 2

Proof of Concept

After addresses attribution we configured RIPv2 in the networks in order to establish connection between them. The auto-summary feature was disabled for this first case. We can see in Figure 55 the routing table for R3:

```
     50.0.0.0/24 is subnetted, 1 subnets
C       50.0.0.0 is directly connected, FastEthernet0/1
     20.0.0.0/24 is subnetted, 1 subnets
R       20.0.0.0 [120/1] via 30.0.0.2, 00:00:00, FastEthernet0/0
     40.0.0.0/24 is subnetted, 1 subnets
R       40.0.0.0 [120/1] via 30.0.0.2, 00:00:00, FastEthernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
R       10.0.0.0 [120/2] via 30.0.0.2, 00:00:00, FastEthernet0/0
     30.0.0.0/24 is subnetted, 1 subnets
C       30.0.0.0 is directly connected, FastEthernet0/0
R3#
```

Figure 55 - Routing Table of R3

We tested connectivity with success. We then used the script ***rippoison.py*** to poison the RIP tables of the routers. This script sends a RIPv2 response advertising a new route to the selected subnet. For the client to connect to the attacker when trying to access the legitimate web server, we configured a virtual secondary interface in the attacker (eth0:0) with the IP of the legitimate web server.

It is mandatory when running the script to pick a subnet mask greater than 24 for the routers, due to the longest prefix matching rule, to choose the route advertised by the attacker.

After running the attack by looking at the routing table of R3 in Figure 56, we can see that it was successfully poisoned. When the client tries to connect to the legitimate web server, it will take the path that connects with the attacker.

```
     50.0.0.0/24 is subnetted, 1 subnets
C       50.0.0.0 is directly connected, FastEthernet0/1
     20.0.0.0/24 is subnetted, 1 subnets
R       20.0.0.0 [120/1] via 30.0.0.2, 00:00:26, FastEthernet0/0
     40.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
R       40.0.0.0/24 [120/1] via 30.0.0.2, 00:00:26, FastEthernet0/0
R       40.0.0.1/32 [120/1] via 50.0.0.2, 00:00:00, FastEthernet0/1
     10.0.0.0/24 is subnetted, 1 subnets
R       10.0.0.0 [120/2] via 30.0.0.2, 00:00:26, FastEthernet0/0
     30.0.0.0/24 is subnetted, 1 subnets
C       30.0.0.0 is directly connected, FastEthernet0/0
R3#
```

Figure 56 - Routing Table of R3 after attack

In order to better see this process, we configured a Wireshark capture in the link between R3 and the attacker. In the capture, we can observe packets like the one shown in Figure 57. We can see the RIP Response messages from the attacker announcing a new route to the network where the legitimate web server is with the attacker as the next hop.

No.	Time	Source	Destination	Protocol	Length	Info
310	303.705545	50.0.0.2	255.255.255.255	RIPv2	66	Response
311	304.707685	50.0.0.2	255.255.255.255	RIPv2	66	Response
312	305.709131	50.0.0.2	255.255.255.255	RIPv2	66	Response
313	306.711069	50.0.0.2	255.255.255.255	RIPv2	66	Response
314	307.713257	50.0.0.2	255.255.255.255	RIPv2	66	Response
315	308.714567	50.0.0.2	255.255.255.255	RIPv2	66	Response

```
> Internet Protocol Version 4, Src: 50.0.0.2, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 520, Dst Port: 520
v Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
  v IP Address: 40.0.0.1, Metric: 1
      Address Family: IP (2)
      Route Tag: 0
      IP Address: 40.0.0.1
      Netmask: 255.255.255.255
      Next Hop: 50.0.0.2
      Metric: 1
```

Figure 57 - Wireshark capture of RIP response packet

As expected, when we try to access the web server, we are redirectioned to the fake web server.

Figure 58 - Fake WebServer

As a confirmation, we can see in Figure 58 the exchange of HTTP and TCP packets between R3 and the attacker.

It is now essential to see how the auto-summary feature can affect the attack performance to test, so we build Topology 2. Firstly, it is essential to understand what is the auto-summary feature. Auto-summary is a feature that allows the router to advertise a single summarized route that represents all contiguous routes using the natural mask of class. For this experiment, we will consider the network represented in Topology 2.

After network configurations, we performed the attack again in the same way, noticing it was unsuccessful. In order to understand what routing information is being passed on through the routers, we configured a Wireshark between the R2 and R3. We were able to capture packets like the one in Figure 59.

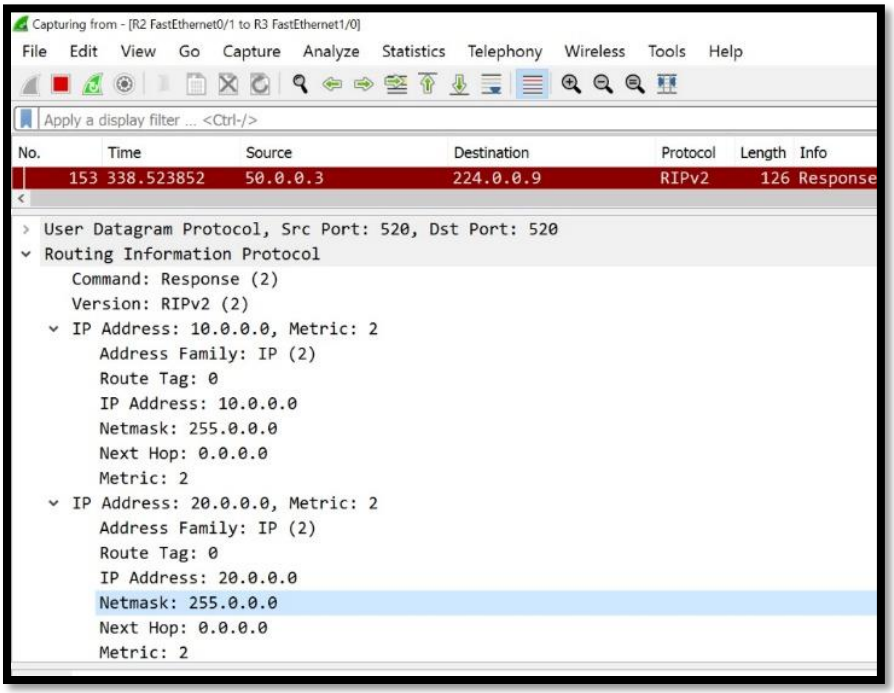


Figure 59 - Wireshark capture

With the auto-summary feature enabled, R3 will propagate a route to the client subnet with a mask 255.255.255.0 (/24). With the other routers in the network propagating paths with the same prefix, the most extended prefix matching rule no longer applies. Routers will then use the metrics to calculate the best path.

If we disable the auto-summary feature for the same topology, the attack will be successful. In this case, R3 will advertise a path with a higher prefix, and the router will again apply the longest matching prefix and prefer that route.

To conclude, we can see that when the auto-summary feature is disabled, the attack will be successful no matter the attacker's position. When the auto-summary feature is enabled, the attack will only be successful when the attacker is positioned closer to the client than the webserver.

Mitigations

In order to mitigate this attack, we must configure RIPv2 Message Authentication. There are two modes we can choose for authentication: plaintext or MD5. To increase security in our configuration, we went for MD5 authentication mode. Firstly, we must configure a key that allows authentication between routers and secondly, we must configure authentication in each interface.

To test the mitigations measure, we reran the attack. In order to see the router's reaction to the RIP packets sent by the attacker, we used the command debug ip rip. We got the following message in R3:

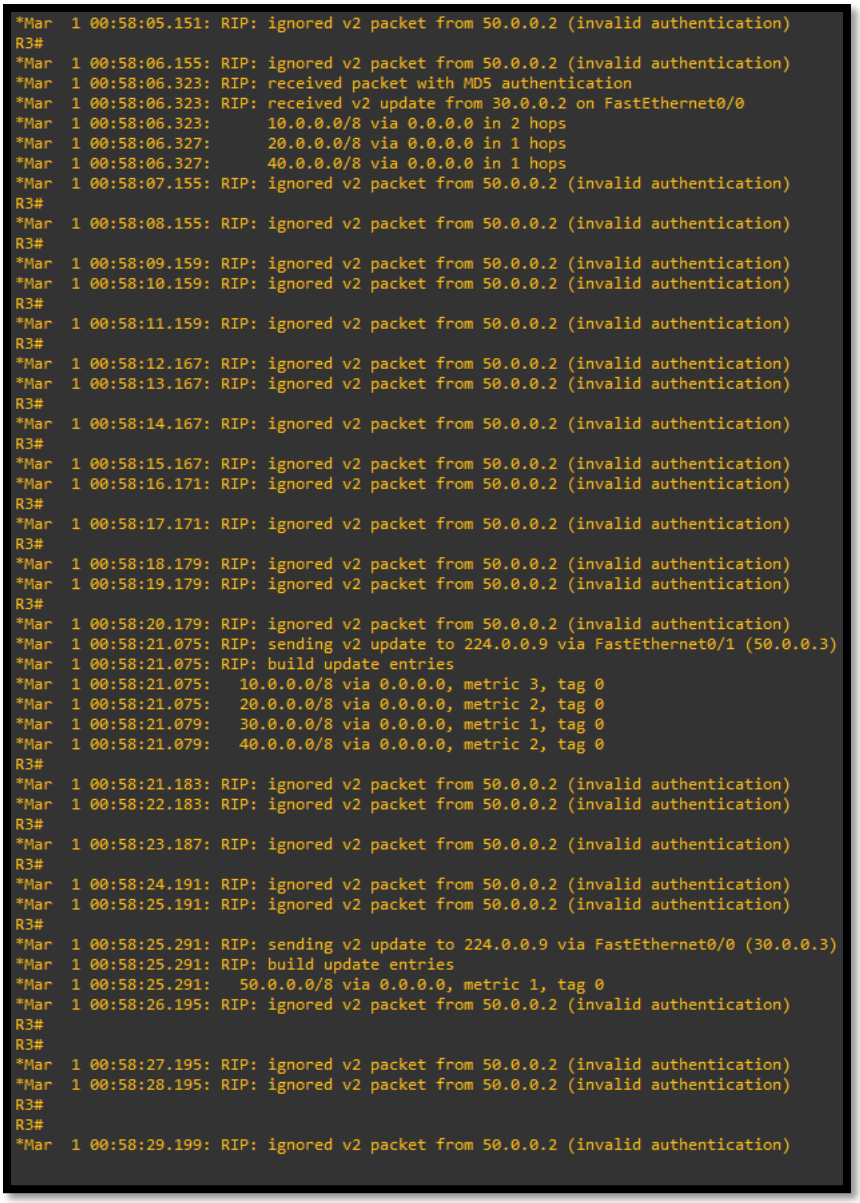


Figure 60 - Router R3 response to attack

Since the packets sent from the attacker fail authentication they will be dropped by the router. This way the attacker is no longer capable of poisoning the RIP table of the routers.

Feasibility

As seen previously this attack might be complex for the attacker. In order for the attack to be successful the attacker must be closer to the client then the Web server or if not it must have the auto-summary feature disabled. Such might not be trivial.

1.7. DNS spoofing using DHCP spoofing

For this exercise we should reproduce the DNS Spoofing attack but where a fake DNS address is provided to the victim via DHCP spoofing.

Network Architecture

We used the following network architecture:

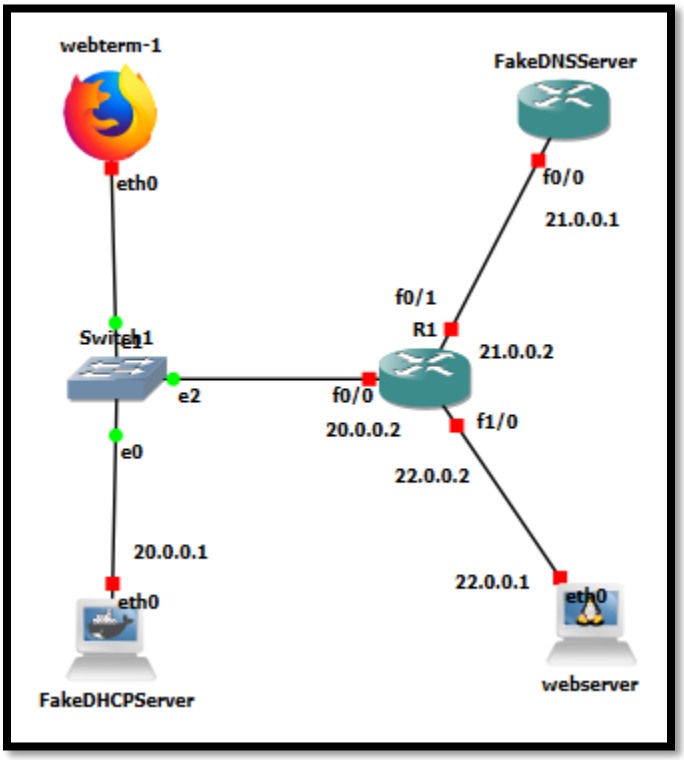


Figure 61 - DNS spoofing using DHCP spoofing Network Architecture

Proof of Concept

1. Firstly, we configured necessary interfaces with the correct IPs as indicated in the network topology.

2. The second step was to configure R2 as the fake DNS server. After enabling DNS service, in order for the attack to work we must specify the binding between the legitimate website in our case linkedin.com and the IP of the fake webserver.

3. In order to make the attacker the fake DHCP server we ran the following command: **python3 dhcpspoof.py eth0 20.0.0.4 20.0.0.10 255.255.255.0 20.0.0.2 21.0.0.1.** The last argument of the function is the address of the fake DNS server to be passed to the client. In order to see this process, we configured a Wireshark capture between the attacker and the switch. In the

capture we can see the DHCP packets sent by the attacker (Figure 62). Is important to highlight that in the DHCP packets there is information of the fake DNS server that is passed on to the client as can be seen in Figure 63.

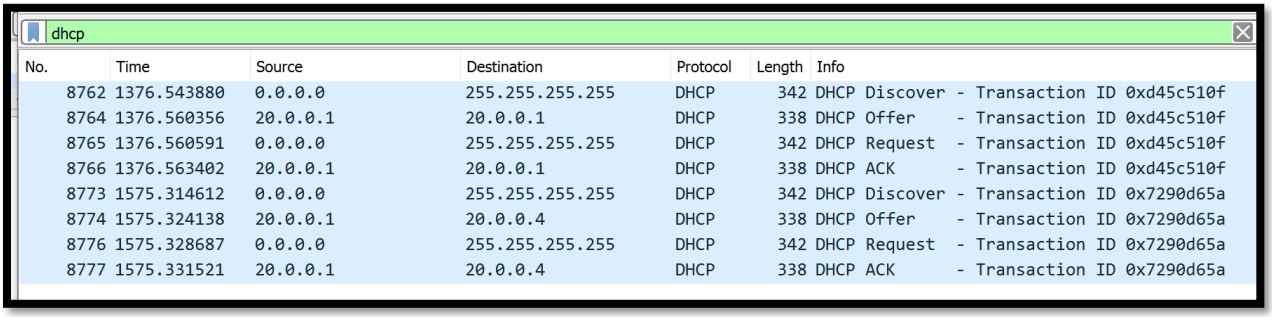


Figure 62 - DHCP Packets

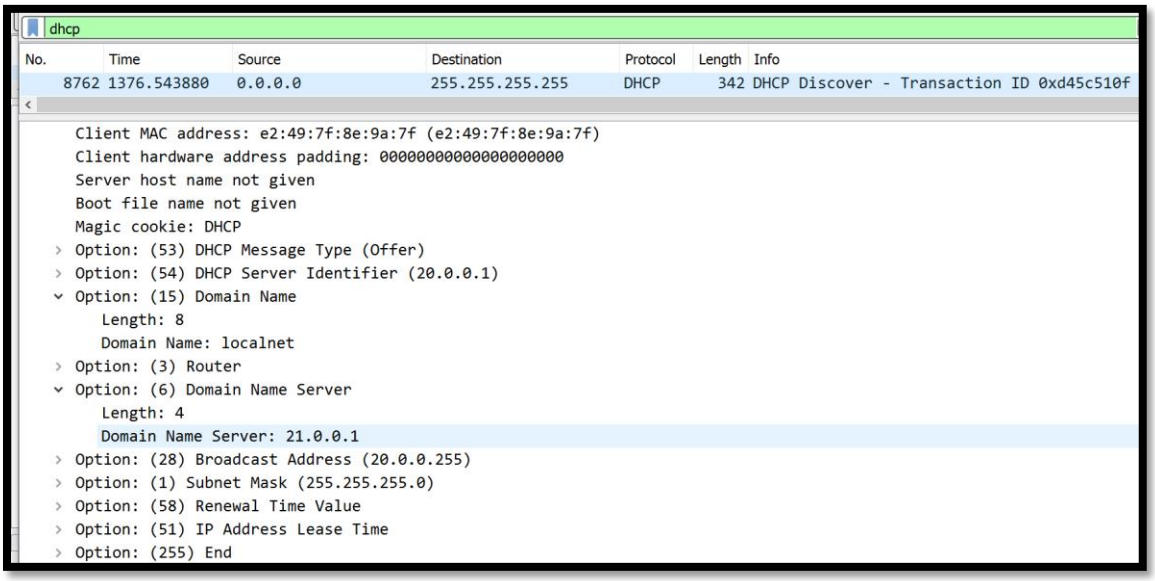


Figure 63 - Information of the fake DNS server that is passed

The IP 20.0.0.4 was attributed to the client. We can now confirm that the attacker was able to perform DHCP spoofing.

4. As expected, when trying to access linkedin.com we can see that the client was redirected to the fake web server as we can see in Figure 64. To confirm, we configured a Wireshark capture between the Web Server and the switch. We can see the exchange of TCP and HTTP packets between the client and the fake web server in Figure 65. We can now confirm the attack was successful.

Figure 64 - Fake Webserver

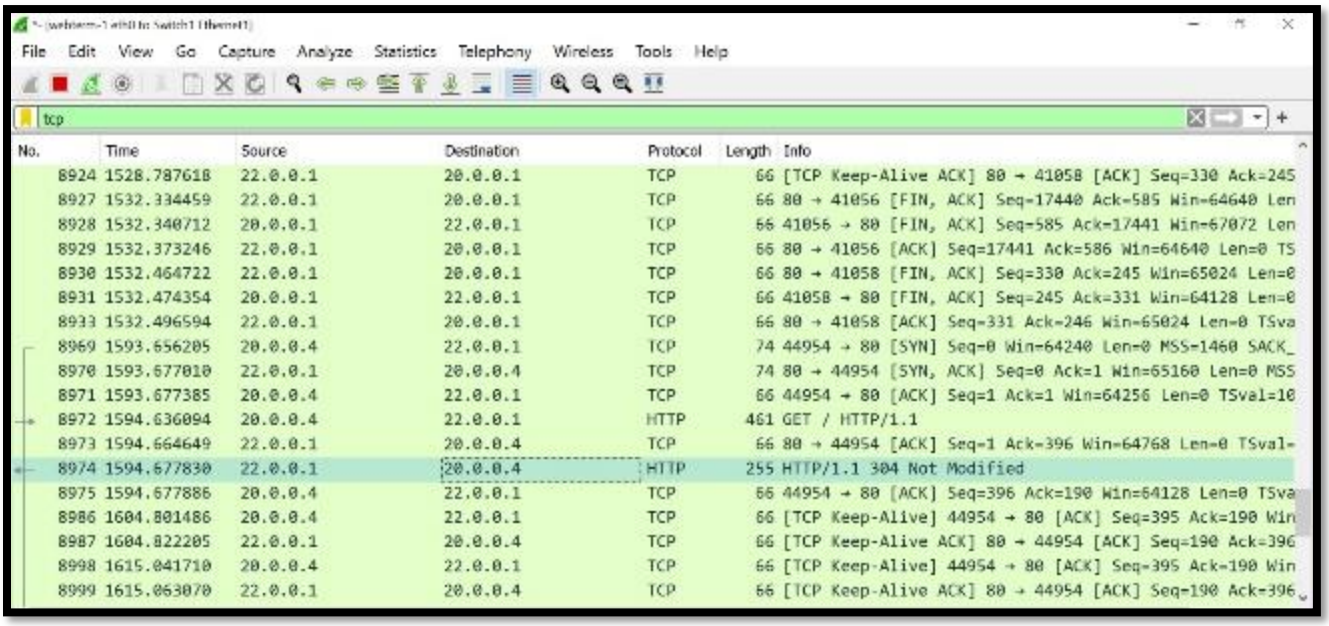


Figure 65 - Exchange of TCP and HTTP packets between the client and the fake web server

2. Lab No. 1.2 - Firewalls

The aim of this laboratory work is to study firewalls. The following attacks will be addressed:

- **(i)** Classical firewalls versus Zone Based Policy Firewalls
- **(ii)** Protecting a campus network using a ZBPF
- **(iii)** Defence against DoS attacks

2.1. Classical firewalls versus Zone-Based Policy Firewalls

The goal of this exercise is to compare classical firewalls with Zone Based Policy Firewalls, using a simple scenario. Classical firewalls are also known as Context-Based Access Control (CBAC) firewalls.

To better understand the capabilities and features of both firewall types Nmap is going to be applied in the attack. Nmap is a network scanner used to discover hosts and services on a network by sending packets and analysing the responses.

Network Architecture

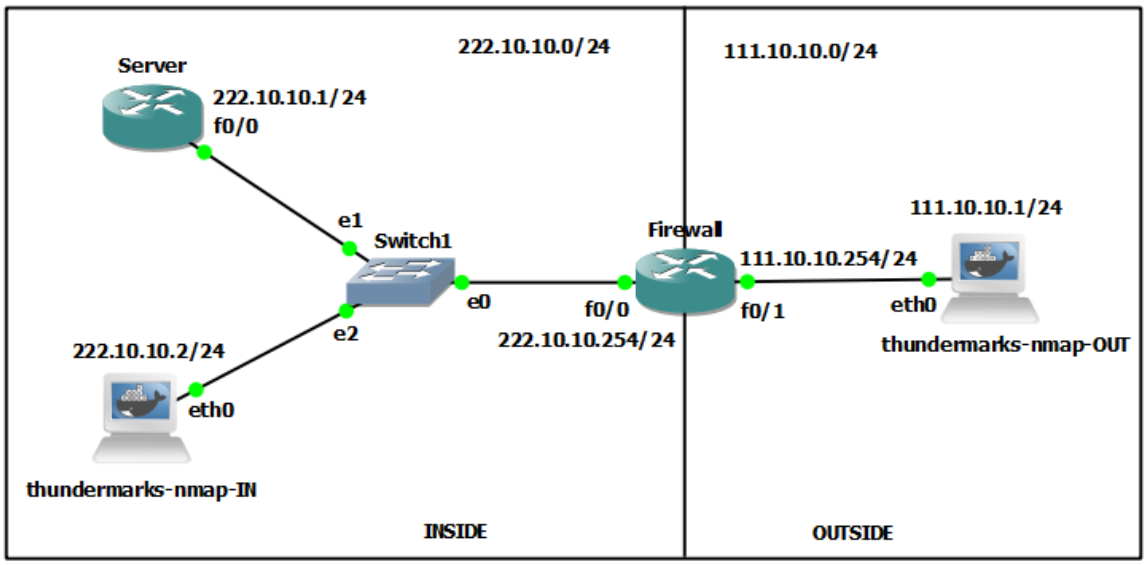


Figure 66 - Classical vs Zone-based policy Firewalls Network Architecture

This laboratory exercise presents a network topology with two zones: inside and outside. These zones are separated by a 7200-router operating as a Firewall. We have a 3725-router working as a Server and a Docker attacker in the inside subnetwork. The outside network only has one Docker attacker.

Proof of Concept

All the configuration commands of this study are presented in Annex A. In this report we choose to present proofs of such configurations to analyse their consequences instead of reporting the actual commands.

1. First, we configured all the IP addresses and tested the connectivity between all devices. All the respective configurations can be found in Annex A.

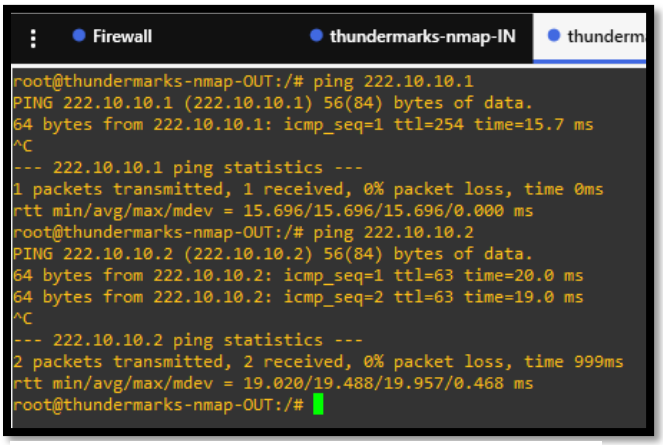


Figure 68 - Attacker-OUT connecting with both Server and Attacker-IN

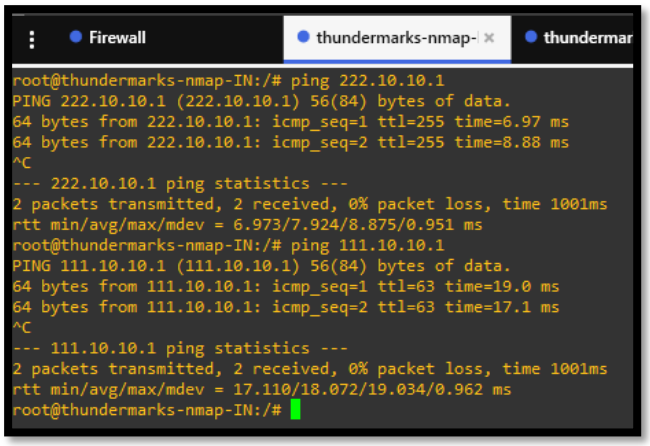


Figure 67 - Attacker-IN connecting with both Server and Attacker-OUT

1. After having the network configured and running, we were able to use ***nmap*** from the OUTSIDE attacker to discover the hosts at the INSIDE subnet.

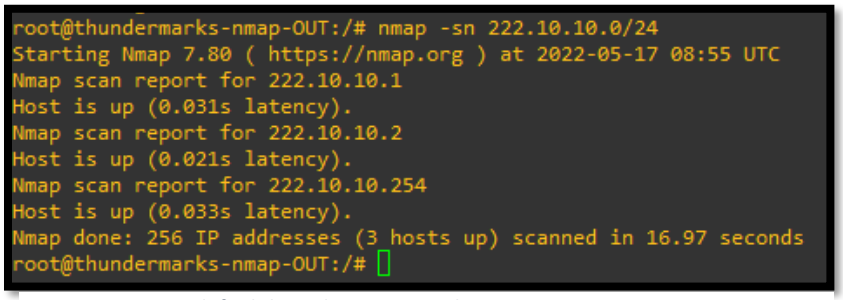


Figure 69 - Nmap default host discovery results

We are using a default Nmap hosts discovery command with the *-sn* option that tells Nmap not to do a port scan after host discovery and only print out the available hosts that responded to the host discovery probes.

Since no host discovery options are given, Nmap sends an ICMP echo request, a TCP SYN packet to port 443, a TCP ACK packet to port 80, and an ICMP timestamp request. Such probes can be seen when analysing the packets using Wireshark. (Nmap, 2022)

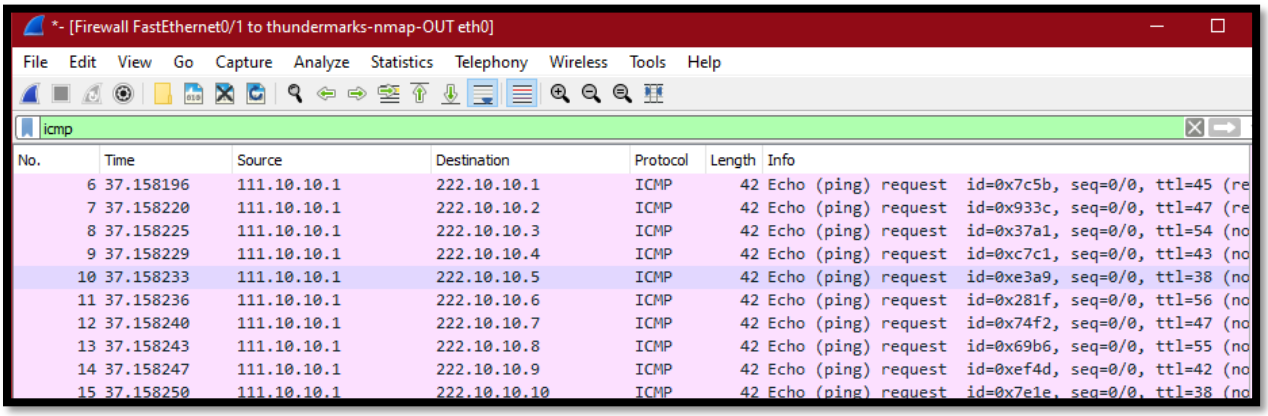


Figure 70 – Hosts discovery ICMP echo Requests

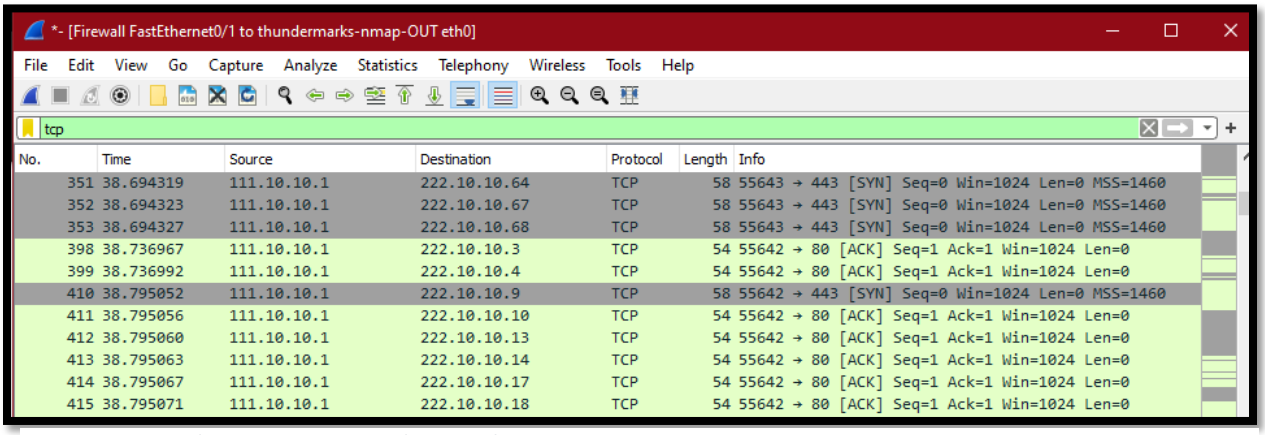


Figure 71 - Hosts discovery TCP SYN and ACK packets

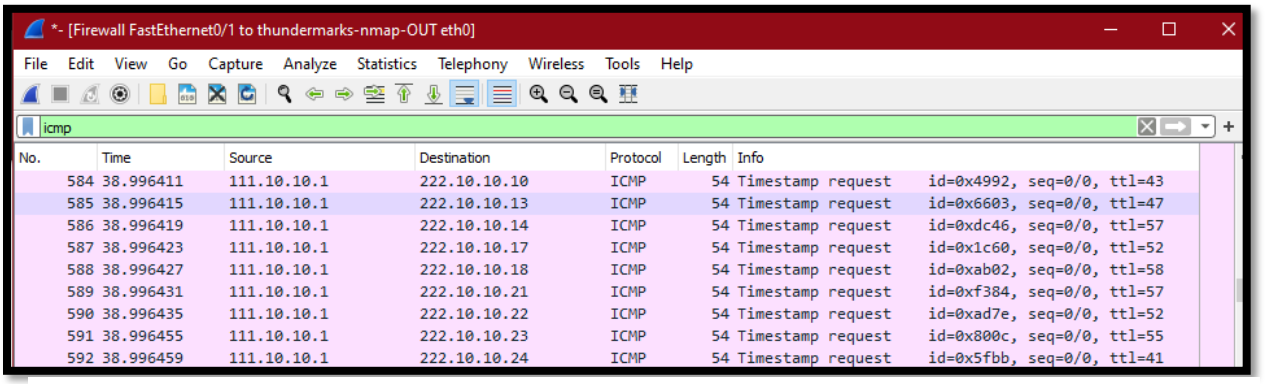


Figure 72 - Hosts discovery ICMP Timestamp requests

2. After this first scan, we install an HTTP Server in the **Server** and open a few more ports for test purposes, in particularly we enable minor TCP services such as *chargen*, *daytime*, *discard*, and *echo*. This time we use Nmap from the OUTSIDE to check which Server ports are open.

```
root@thundermarks-nmap-OUT:/# nmap -np0-1023 222.10.10.1
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-17 09:33 UTC
Nmap scan report for 222.10.10.1
Host is up (0.033s latency).
Not shown: 1018 closed ports
PORT   STATE SERVICE
7/tcp  open  echo
9/tcp  open  discard
13/tcp open  daytime
19/tcp open  chargen
23/tcp open  telnet
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 13.76 seconds
root@thundermarks-nmap-OUT:/#
```

Figure 73 - Nmap port scanning results

This Nmap command uses the *-n* option that skipped a reverse-DNS query that improved the speed and stealthiness of the scan, and the -p0 that asks Nmap to scan every possible TCP port.

This scan starts by pinging the host with a default ICMP echo request packet and a TCP ACK packet to port 80 that determines if it is up and running. It then launched a TCP port scan on all possible TCP ports. (Nmap, 2022)

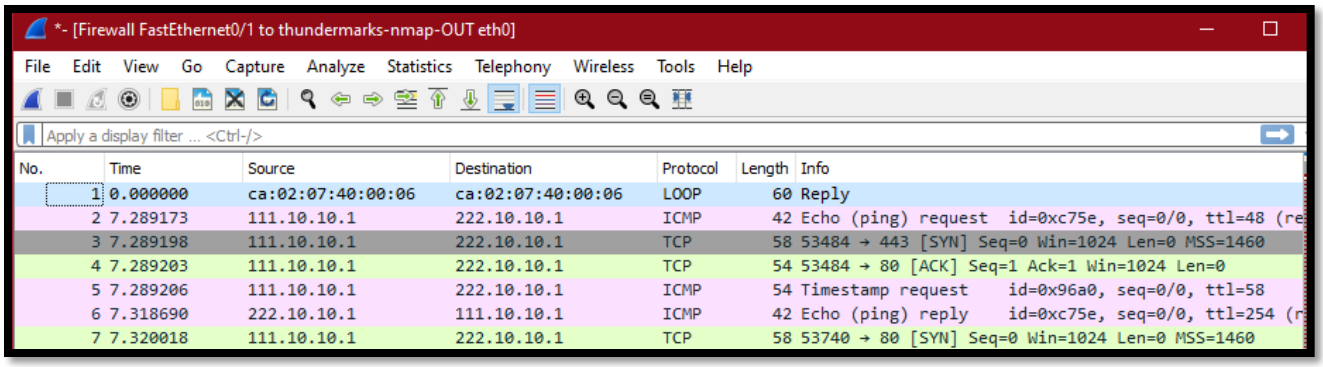


Figure 75 - Port scan initial ICMP echo request and TCP ACK packet

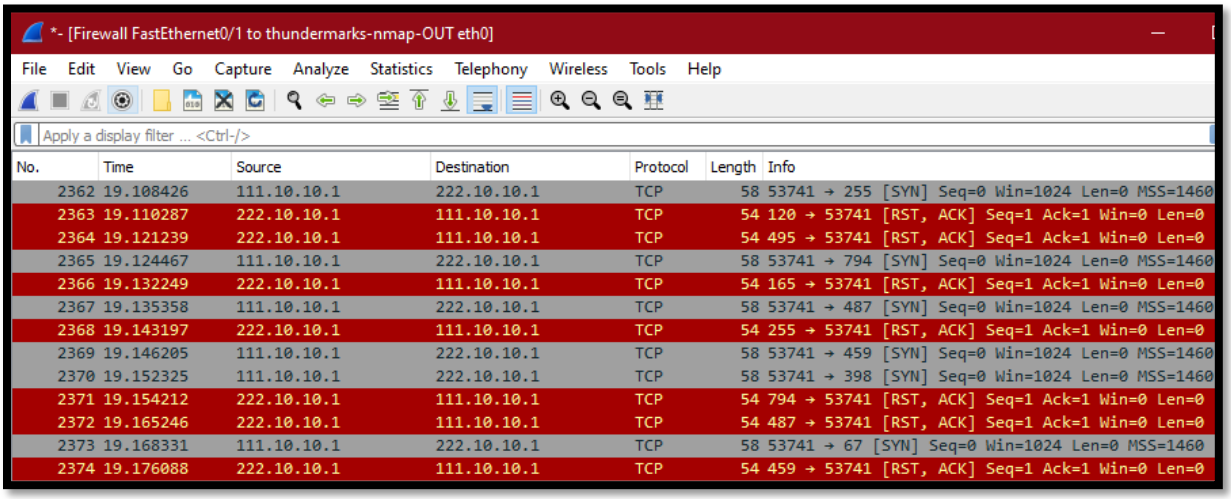


Figure 74 – TCP port scan

3. After doing these scans we are now going to configure the classical firewall. This configuration is based on applying inspection rules on the ICMP and HTTP protocols on the f0/1 interface and defining an Access Control List (ACL) to permit traffic to the Server.

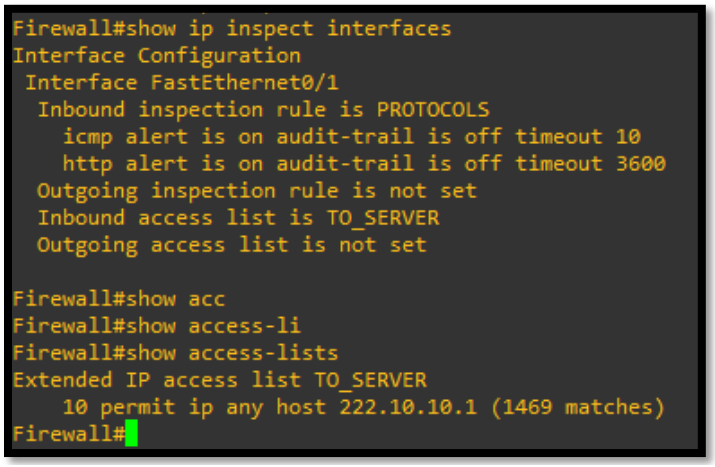


Figure 76 - First Firewall inspect rules and access-list configurations

Although, using Nmap we can see that the above configuration is not enough to prevent access to all open Server Ports because the rule introduced in the ACL *permit ip* means permitting both TCP and UDP including all ports. Because inspect rules do not have an implicit deny at the end, traffic from other protocols still enters the INISIDE network and hits the Server, in this case we need to be more specific in the protocols we wish to permit and also introduce rules to block all traffic from inside to outside in the f0/0 interface.

4. Using the firewall rules presented bellow we were able to prevent the previous situation. We can also see that the Nmap can only identify the http service as open and only the OUTSIDE can ping the INSIDE not the other way around, just like required.

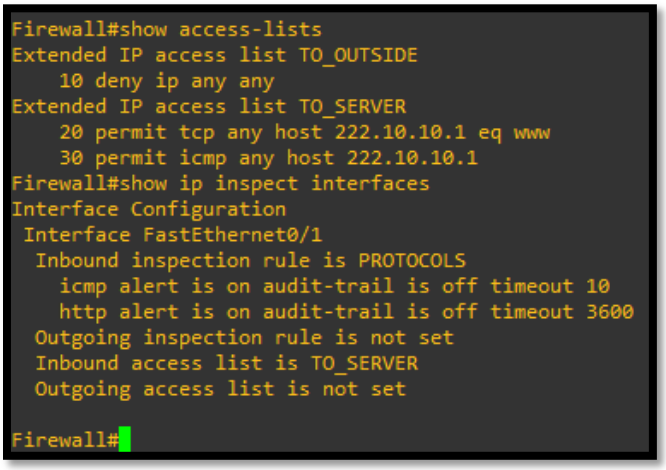


Figure 77 - Second Firewall inspect rules and access-list configurations

```
root@thundermarks-nmap-OUT:/# nmap -np0-1023 222.10.10.1
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-18 14:47 UTC
Nmap scan report for 222.10.10.1
Host is up (0.013s latency).
Not shown: 1023 filtered ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.88 seconds
root@thundermarks-nmap-OUT:/# ping 222.10.10.1
PING 222.10.10.1 (222.10.10.1) 56(84) bytes of data.
64 bytes from 222.10.10.1: icmp_seq=1 ttl=254 time=12.8 ms
64 bytes from 222.10.10.1: icmp_seq=2 ttl=254 time=30.1 ms
^C
--- 222.10.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 12.773/21.436/30.100/8.663 ms
root@thundermarks-nmap-OUT:/#
```

Figure 79 - Nmap port scan and ping from OUTSIDE to INSIDE

```
Server#ping 111.10.10.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 111.10.10.1, timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
Server#
```

Figure 78 - Ping from INSIDE to OUTSIDE

5. We now start to configure the Zone-Based Policy Firewall by assigning security zones to the firewall interfaces:

```
Firewall#show zone security
zone self
  Description: System defined zone


zone INSIDE
  Member Interfaces:
    FastEthernet0/0


zone OUTSIDE
  Member Interfaces:
    FastEthernet0/1
```

Figure 80 - Security Zones

Using Nmap from both the INSIDE and OUTSIDE zones we are able to determine that the outside attacker could scan both the inside and outside interfaces of the firewall, and the same can be reached from the inside attacker.

```
root@thundermarks-nmap-OUT:/# nmap -np0-1023 222.10.10.254
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-18 15:32 UTC
Nmap scan report for 222.10.10.254
Host is up (0.0089s latency).
Not shown: 1023 closed ports
PORT   STATE SERVICE
23/tcp open  telnet

Nmap done: 1 IP address (1 host up) scanned in 8.67 seconds
root@thundermarks-nmap-OUT:/# nmap -np0-1023 111.10.10.254
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-18 15:32 UTC
Nmap scan report for 111.10.10.254
Host is up (0.0077s latency).
Not shown: 1023 closed ports
PORT   STATE SERVICE
23/tcp open  telnet
MAC Address: CA:02:AF:61:00:06 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 8.82 seconds
root@thundermarks-nmap-OUT:/#
```

Figure 82 - Nmap scan on both Firewall interfaces from OUTSIDE attacker

```
root@thundermarks-nmap-IN:/# nmap -np0-1023 222.10.10.254
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-18 15:34 UTC
Nmap scan report for 222.10.10.254
Host is up (0.0089s latency).
Not shown: 1023 closed ports
PORT   STATE SERVICE
23/tcp open  telnet
MAC Address: CA:02:AF:61:00:08 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 8.14 seconds
root@thundermarks-nmap-IN:/# nmap -np0-1023 111.10.10.254
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-18 15:34 UTC
Nmap scan report for 111.10.10.254
Host is up (0.0081s latency).
Not shown: 1023 closed ports
PORT   STATE SERVICE
23/tcp open  telnet

Nmap done: 1 IP address (1 host up) scanned in 8.54 seconds
root@thundermarks-nmap-IN:/#
```

Figure 81 - Nmap scan on both Firewall interfaces from INSIDE attacker

6. We now complete the configuration of the ZBPF firewall and test it again using Nmap. Remembering that the goal is still to allow HTTP and ICMP from the outside to inside and block all traffic from inside to outside with a new condition that access to the firewall interfaces is not allowed.
   To achieve this, we now define the TO_SERVER ACL presented below:

```
Firewall#show access-lists
Extended IP access list TO_SERVER
    10 permit ip any host 222.10.10.1 (2055 matches)
Firewall#
```

Figure 83 - ZBPF ACL

We then identify traffic with a class map. In the image bellow we can see the three class maps configured, CLASSMAP, PROTOCOLS and TO_FW_CLASSMAP:

```
Firewall#show class-map type inspect
 Class Map type inspect match-all CLASSMAP (id 2)
   Match access-group name TO_SERVER
   Match class-map PROTOCOLS

 Class Map type inspect match-any TO_FW_CLASSMAP (id 14)
   Match none

 Class Map type inspect match-any PROTOCOLS (id 1)
   Match protocol icmp
   Match protocol http

Firewall#
```

Figure 84 - Class maps

Then we define the action of *inspect* with the following policy maps:

```
Firewall#show policy-map type inspect
 Policy Map type inspect TO_FW_POLICYMAP
   Class TO_FW_CLASSMAP
     Pass
   Class class-default
     Drop log

 Policy Map type inspect POLICYMAP
   Class CLASSMAP
     Inspect
   Class class-default
     Drop log

Firewall#
```

Figure 85 - Policy Maps

And finally, since we already identified the two security zones and associated them to the respective interfaces, we can now just associate the pair with the policy maps created:

```
Firewall#show zone-pair security
Zone-pair name OUT_TO_IN
    Source-Zone OUTSIDE  Destination-Zone INSIDE
    service-policy POLICYMAP
Zone-pair name IN_TO_SELF
    Source-Zone INSIDE  Destination-Zone self
    service-policy TO_FW_POLICYMAP
Zone-pair name OUT_TO_SELF
    Source-Zone OUTSIDE  Destination-Zone self
    service-policy TO_FW_POLICYMAP

Firewall#
```

Figure 86 - Zone Pair associated with the respective policy maps

7. After the configuration of the ZBPF we now test the firewall using Nmap and concluding that all the respective rules are applied and operational. The following images prove that no traffic is allowed from INSIDE to OUTSIDE. From OUTSIDE to INSIDE, only traffic to the Server machine is allowed, and only for ICMP and HTTP protocols and also access to firewall interfaces from either zone is not allowed.

```
root@thundermarks-nmap-OUT:/# nmap -np0-1023 111.10.10.254
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-18 18:04 UTC
Nmap scan report for 111.10.10.254
Host is up (0.038s latency).
All 1024 scanned ports on 111.10.10.254 are filtered
MAC Address: CA:02:AF:61:00:06 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 40.63 seconds
root@thundermarks-nmap-OUT:/# nmap -np0-1023 222.10.10.254
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-18 18:05 UTC
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.07 seconds
root@thundermarks-nmap-OUT:/#
```

Figure 87 - OUTSIDE Attacker unable to access firewall interfaces

```
root@thundermarks-nmap-IN:/# nmap -np0-1023 111.10.10.254
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-18 18:07 UTC
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.08 seconds
root@thundermarks-nmap-IN:/# nmap -np0-1023 222.10.10.254
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-18 18:07 UTC
Nmap scan report for 222.10.10.254
Host is up (0.0060s latency).
All 1024 scanned ports on 222.10.10.254 are filtered
MAC Address: CA:02:AF:61:00:08 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 21.83 seconds
root@thundermarks-nmap-IN:/#
```

Figure 88 - INSIDE Attacker unable to access firewall interfaces

```
Server#ping 111.10.10.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 111.10.10.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Server#
```

Figure 90 - Traffic not allowed from INSIDE to OUTSIDE

```
root@thundermarks-nmap-OUT:/# nmap -np0-1023 222.10.10.1
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-18 18:10 UTC
Nmap scan report for 222.10.10.1
Host is up (0.034s latency).
Not shown: 1023 filtered ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 6.12 seconds
root@thundermarks-nmap-OUT:/#
```

Figure 91 - HTTP traffic allowed to Server from OUTSIDE

```
root@thundermarks-nmap-OUT:/# ping 222.10.10.2
PING 222.10.10.2 (222.10.10.2) 56(84) bytes of data.
^C
--- 222.10.10.2 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4102ms

root@thundermarks-nmap-OUT:/# ping 222.10.10.1
PING 222.10.10.1 (222.10.10.1) 56(84) bytes of data.
64 bytes from 222.10.10.1: icmp_seq=1 ttl=254 time=26.2 ms
64 bytes from 222.10.10.1: icmp_seq=2 ttl=254 time=31.1 ms
^C
--- 222.10.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 26.220/28.655/31.090/2.435 ms
root@thundermarks-nmap-OUT:/#
```

Figure 89 - Only ICMP packets to Server are allowed from OUTSIDE

Classical Firewalls vs ZBPFs

After the laboratory demonstration of both firewalls' configuration and testing, we can now point out the main differences between the two.

While classical firewalls apply their rules to their interfaces, ZBPF assigns their interfaces to security zones and applies the firewall policies to traffic between those zones. In ZBFP, we can define a self-zone on the firewall that protects the firewall's interfaces.

2.2. Protecting a campus network using a ZBPF

In this laboratory we aim to protect a simulated campus network using a Zone Based Policy Firewall.

Network Architecture

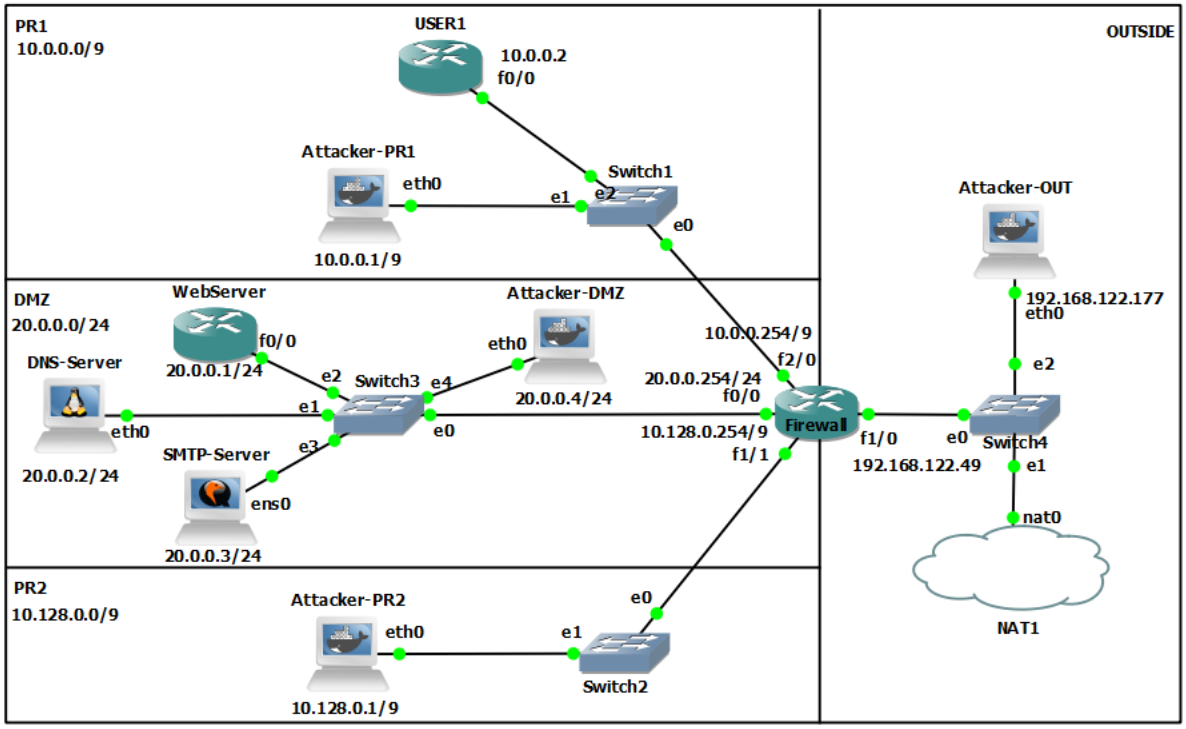


Figure 92 - Campus Network Architecture

This network topology is divided into 4 zones: Private Zone 1 (PR1), Private Zone 2 (PR2), a DMZ which represent the Inside of the network, and the Outside connected to the internet. The Firewall splitting the zones is a 7200 router in the PR1, PR2 and OUTSIDE we have a Docker Attacker; in the DMZ we have a 3725-cisco router as a Web Server, a DNS Server and a SMTP Server.

As we can see in the above image the private zones share an address range of 10.0.0.0/8 divided into two subnets.

Proof of Concept

Due to the size of the configuration rules of this laboratory, we present merely parts of the entire configuration to demonstrate how we achieved the requirements in the lab guide. The complete configuration is present in Annex A.

1. First, we start by configuring all network devices and services without the firewall policies implemented.
2. Then we create the defined zones in the firewall and assign each zone to the respective interface.

```
## ZBPF ZONE CONFIGURATION ##
conf t
zone security PRIVATE-ZONE-1
zone security PRIVATE-ZONE-2
zone security OUTSIDE
zone security DMZ
end

## DEFINITION OF ZONES ##
conf t
int f2/0
zone-member security PRIVATE-ZONE-1
exit
int f1/1
zone-member security PRIVATE-ZONE-2
exit
int f0/0
zone-member security DMZ
exit
int f1/0
zone-member security OUTSIDE
end
```

3. After the assigned zones we create the necessary ACLs and class-maps according with the respective flow of traffic we wish to inspect. The class-map presented below is applied to the Web Server.

```
## Definition of ACLs ##
conf t
ip access-list extended TO-WEB-SERVER
permit ip any host 20.0.0.1
exit
ip access-list extended TO-DNS-SERVER
permit ip any host 20.0.0.2
exit
ip access-list extended TO-SMTP-SERVER
permit ip any host 20.0.0.3
end

## ZBPF CLASS-MAPS PROTOCOLS CONFIGURATION ##
class-map type inspect match-any TO-DMZ-WEB
match protocol icmp
match protocol http
match protocol https
exit
```

```
19.
20. class-map type inspect match-all TO-WEB-SERVER-CLASSMAP
21. match access-group name TO-WEB-SERVER
22. match class-map TO-DMZ-WEB
23. exit
```

4. Now we assign the class-maps to policy-maps related to the flow of traffic from one zone to another. The example bellow shows the policy-map from the DMZ to the OUTSIDE zone:

```
1.  policy-map type inspect DMZ-TO-OUTSIDE-POLICY
2.  class type inspect DMZ-TO-OUTSIDE-CLASSMAP
3.  inspect
4.  class class-default
5.  drop log
6.  exit
7.  exit
```

5. We now identify the zone-pairs and assign them to the respective policy-maps. The following example shows the policy-map presented previously being assigned to the two zones DMZ and OUTSIDE:

```
1.  zone-pair security DMZ-TO-OUT source DMZ destination OUTSIDE
2.  service-policy type inspect DMZ-TO-OUTSIDE-POLICY
3.  exit
```

6. After using this method to configure all zones and achieve the specifications of this campus network we can now test the network against its requirements.

Firewall Specifications and Network Test

Here we demonstrate all the requirements with proofs of the network tests. Tools such as Nmap, SSH and Telnet were applied under the consideration of the respective protocols of each test.

- **Private Zones:**
  1. The private zones PR1 and PR2 cannot communicate with each other. In the figures bellow we can see an Nmap scan from PR1 to PR2 and from PR2 to PR1 where both get blocked by the firewall.

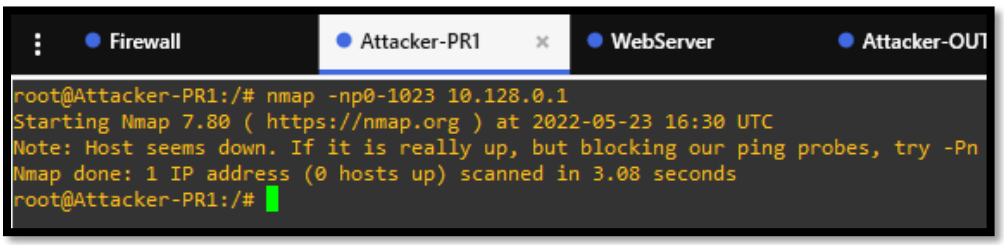


Figure 93 - Nmap scan Attacker-PR1 to Attacker-PR2

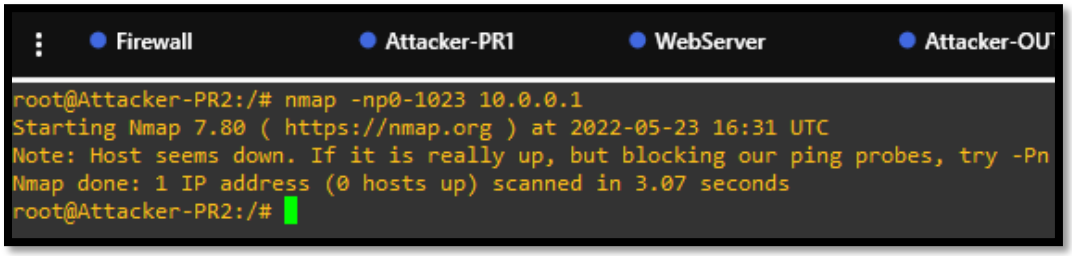


Figure 94 - Nmap scan Attacker-PR2 to Attacker-PR1

2. The private zones PR1 and PR2 cannot be accessed from other zones. In the figures bellow we can see hosts from the DMZ and from the OUTSIDE trying to reach both private zones and in both cases the firewall blocks the packets.

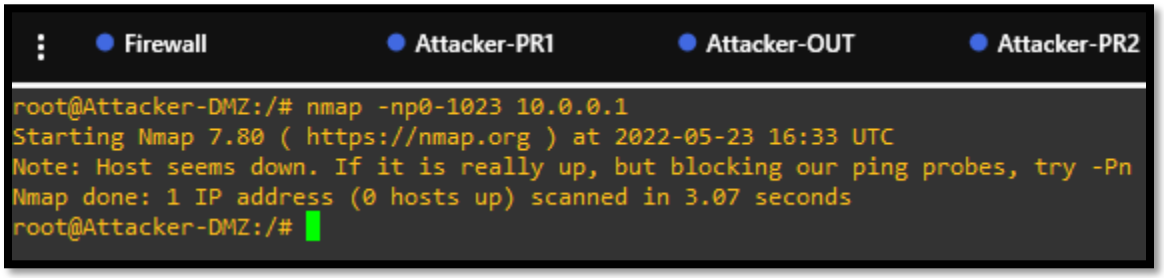


Figure 95 - Nmap scan from Attacker-DMZ to Attacker-PR1

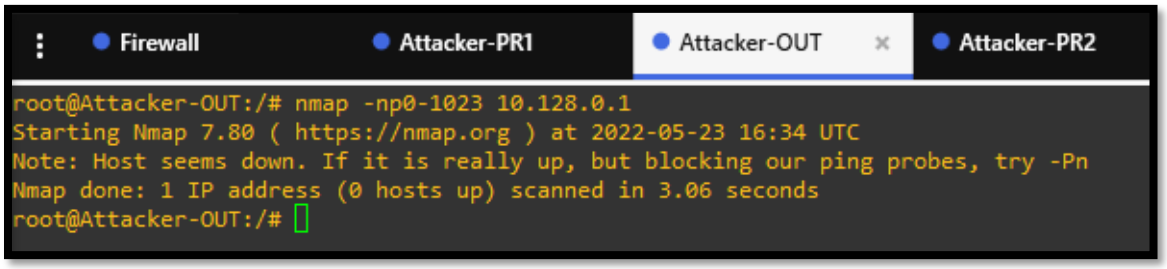


Figure 96 - Nmap scan from Attacker-OUT to Attacker-PR2

3. Communication from the private zones to the OUT zone is only allowed for Web browsing (HTTP and HTTPS), ICMP, and DNS. In the image bellow we can see that all those services are allowed but no other is.

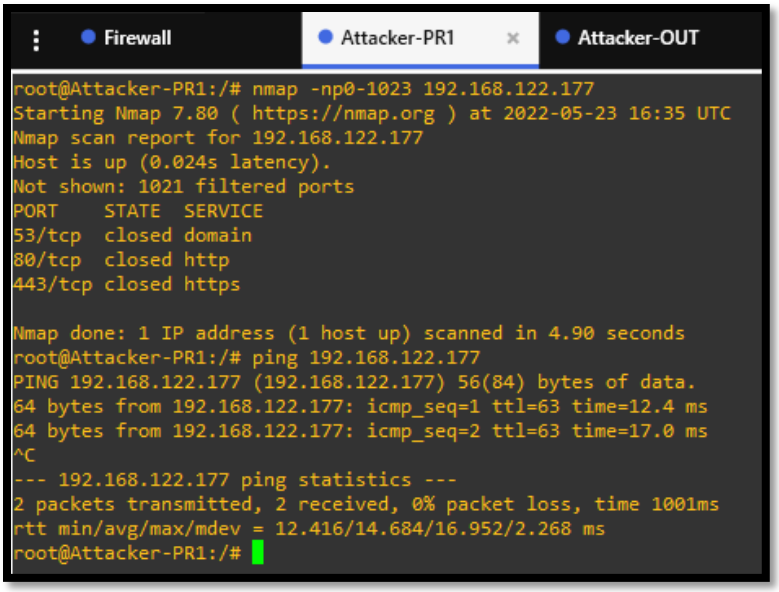


Figure 97 - Ping and Nmap Scan from Attacker-PR1 to Attacker-OUT

4. Communication from the private zones to the DMZ is only allowed for Web browsing (HTTP and HTTPS), ICMP, Email (POP3, IMAP, and SMTP), and DNS, and only for the three DMZ servers. In the following figures we see tests of each of these services from both private zones to the DMZ where we also obtain the desired result.

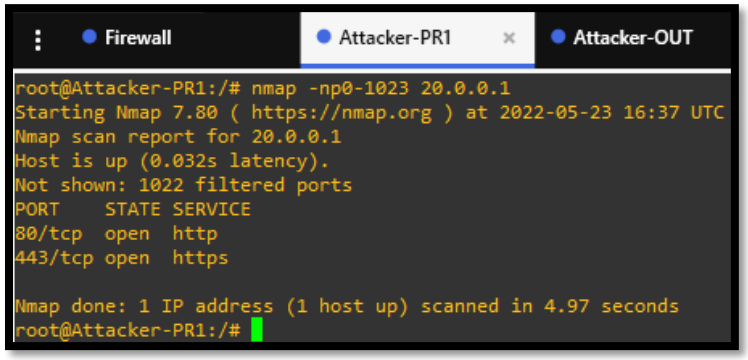


Figure 98 - Nmap Scan from Attacker-PR1 to WebServer on DMZ

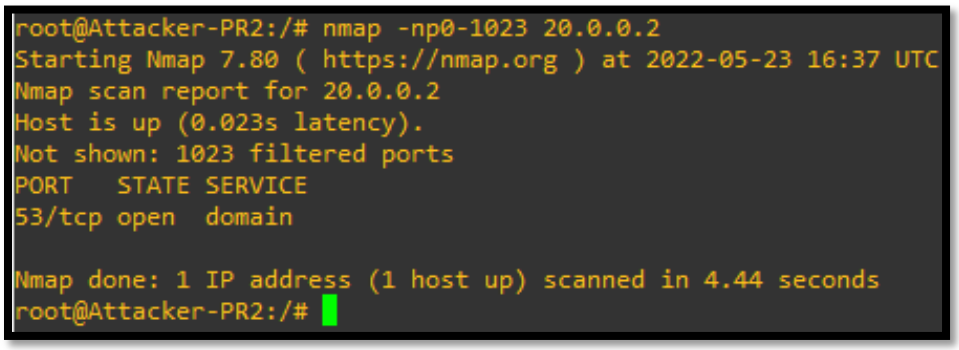


Figure 99 - Nmap scan from Attacker-PR2 to DNS-Server at DMZ

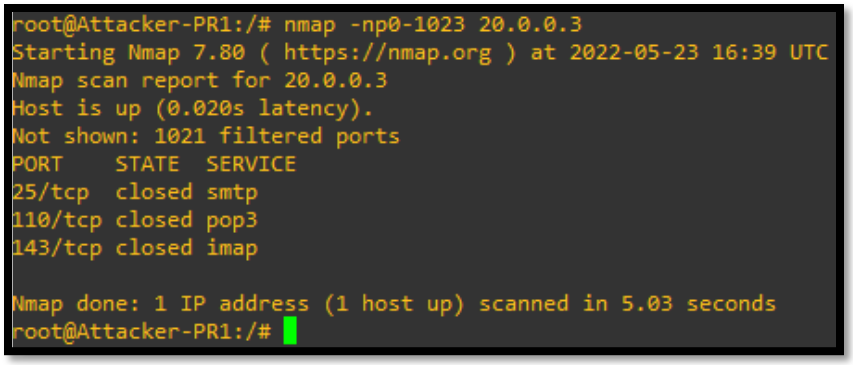


Figure 100 - Nmap scan from Attacker-PR1 to Email-Server on DMZ

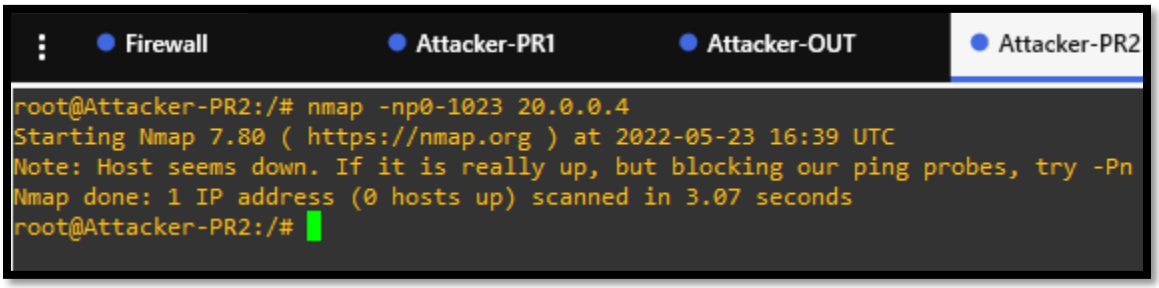


Figure 101 - Nmap scan from Attacker-PR2 to Attacker-DMZ

- **DMZ:**
  1. Access to DMZ servers should be granted from all other zones. In the figure bellow we can see three Nmap scans from OUTSIDE to the DMZ. Regarding zone PR1 and PR2 we have already tested this rule in the previous tests.

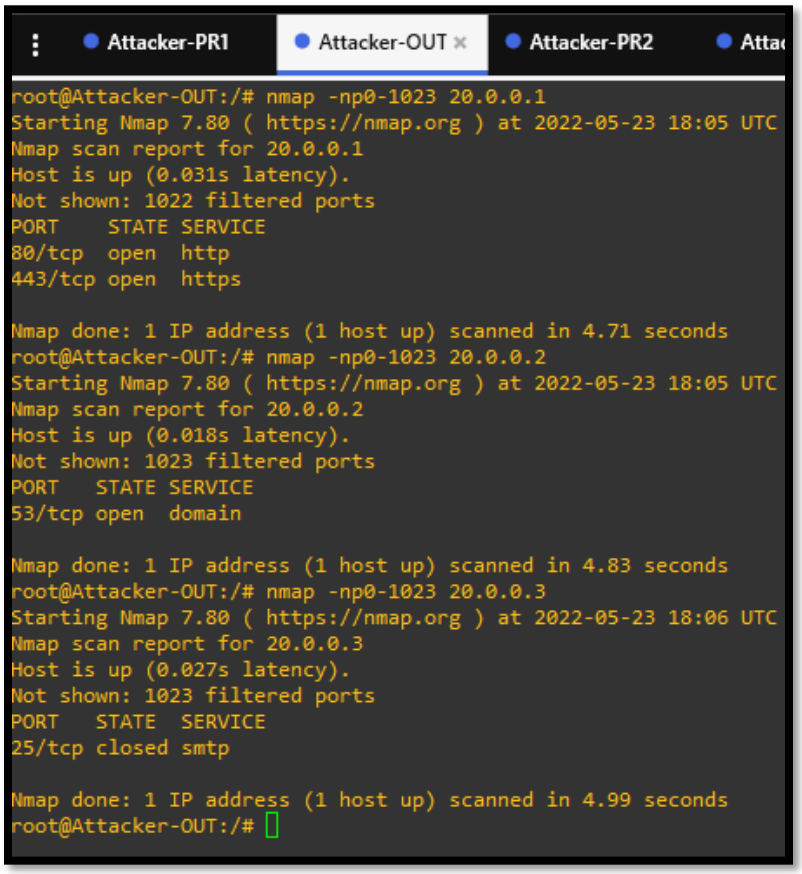


Figure 102 - Nmap scan from Attacker-OUT to WebServer, DNS-Server and Email-Server on the DMZ

2. Communication from the DMZ to the OUT zone is only allowed for ICMP, Email (SMTP), and DNS. In the next figure we can see an Nmap scan from the DMZ to the outside where we demonstrate that only these three services are allowed.

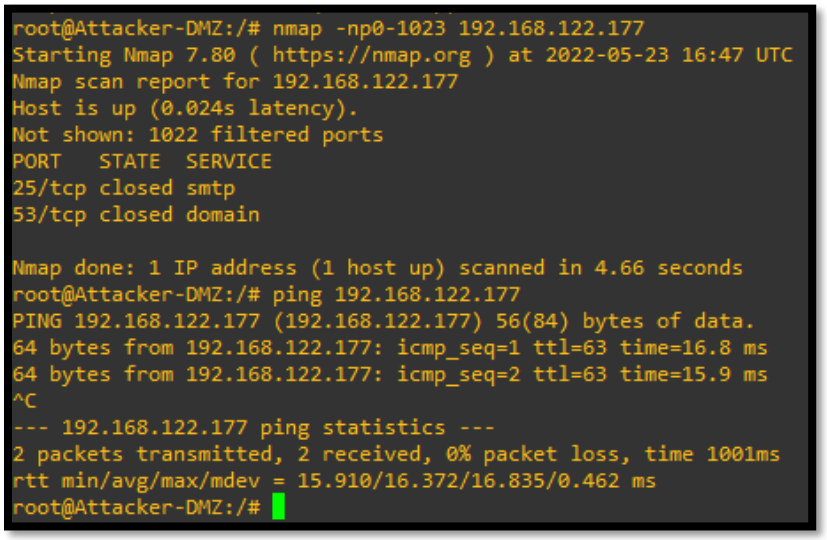


Figure 103 - Ping and Nmap scan from Attacker-DMZ to Attacker-OUT

3. Communication from the OUT zone to the DMZ is only allowed for Web browsing (HTTP and HTTPS), ICMP, Email (SMTP), and DNS, and only for the three DMZ servers. In the next figure we demonstrate that only these services are allowed and only to the specified machines.

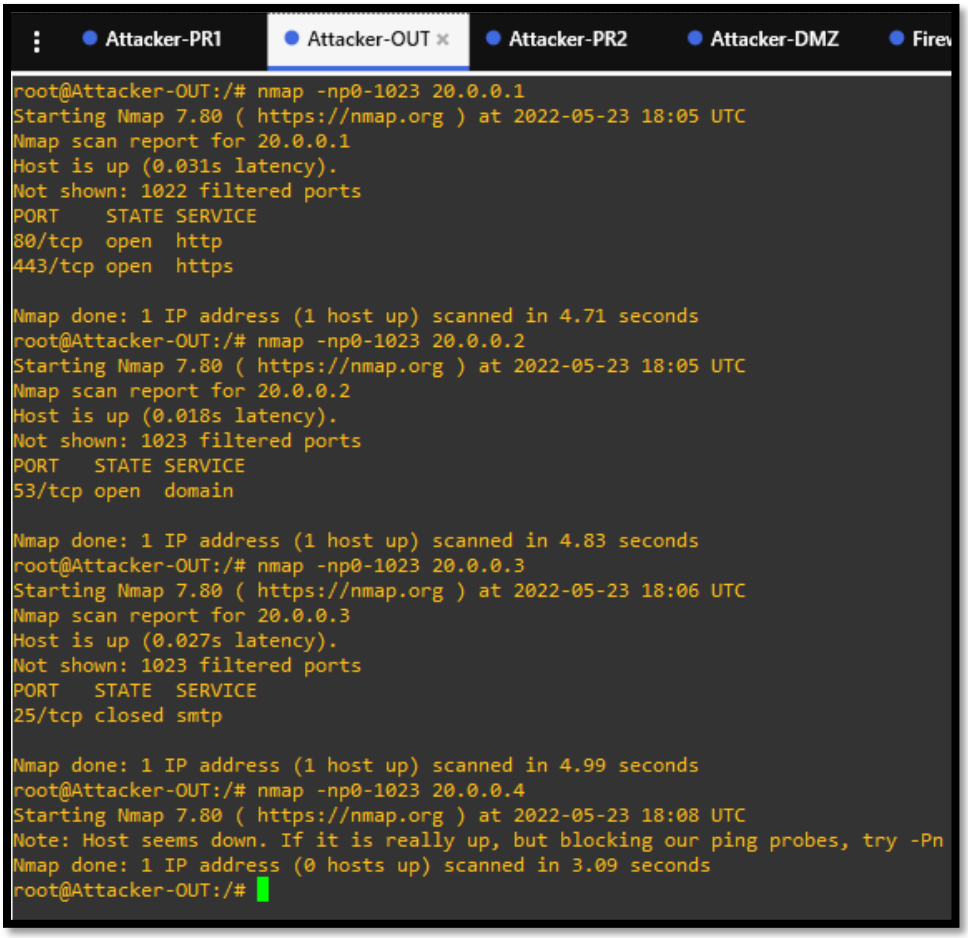


Figure 104 - Nmap scans from Attacker Outside to WebServer, DNS-Server and Email-Server and Ping to Attacker-DMZ

- **Firewall:**

1. The firewall must include NAT for communication with the OUT zone. In Figure 105 we can see the attacker from the DMZ accessing google.com using the DNS server.

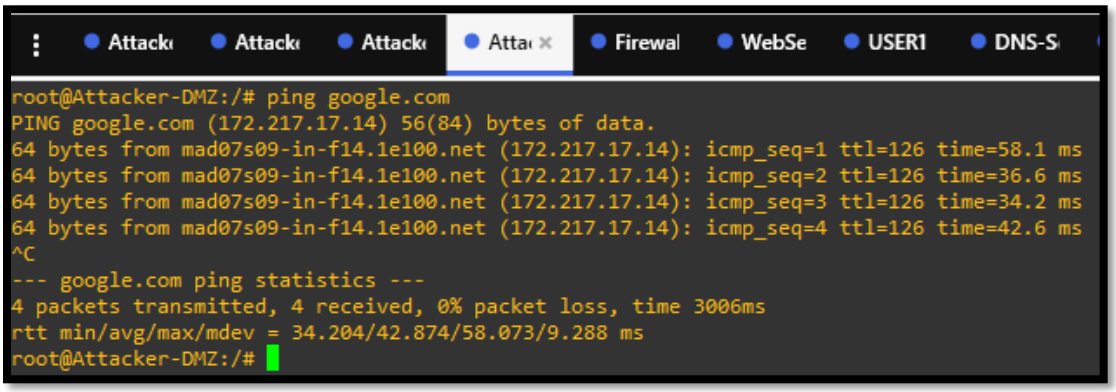


Figure 105 - Ping from Attacker-DMZ to Google.com

2. Access to the firewall must be allowed from all zones, but only using SSH. In the two figures bellow we can see that SSH is allowed from Attacker on Private Zone 1 and telnet is denied from that same zone.

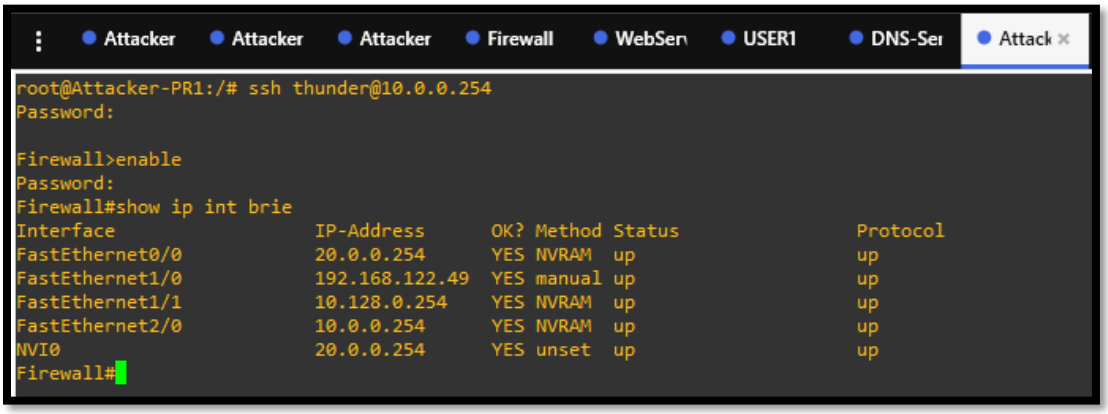


Figure 106 - SSH from Attacker-PR1 to Firewall

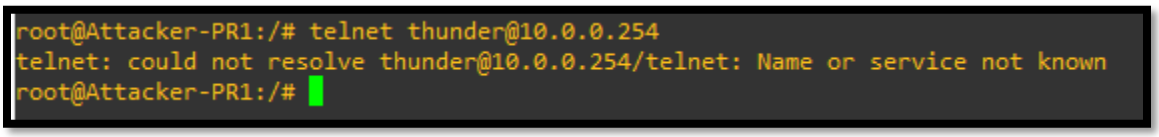


Figure 107 - Failed Telnet connection from Attacker-PR1 to Firewall

In the figure bellow we can see log messages from the Firewall demonstrating some of the blocked connections attempted on the previous tests.

```
Firewall#
*May 23 16:24:14.407: %FW-6-DROP_PKT: Dropping udp session 20.0.0.3:57591 91.189.94.4:123 on zone-pair DMZ-TO-OUT class class-default
 due to  DROP action found in policy-map with ip ident 0
Firewall#
*May 23 16:24:45.079: %FW-6-DROP_PKT: Dropping udp session 20.0.0.3:60676 91.189.89.199:123 on zone-pair DMZ-TO-OUT class class-defau
lt due to  DROP action found in policy-map with ip ident 0
Firewall#
*May 23 16:25:59.715: %FW-6-DROP_PKT: Dropping tcp session 10.0.0.1:62681 20.0.0.2:443 on zone-pair PR1-TO-DMZ class class-default du
e to  DROP action found in policy-map with ip ident 0
Firewall#
*May 23 16:26:33.803: %FW-6-DROP_PKT: Dropping tcp session 192.168.122.177:36764 20.0.0.2:443 on zone-pair OUT-TO-DMZ class class-def
ault due to  DROP action found in policy-map with ip ident 0
Firewall#
*May 23 16:29:58.635: %FW-6-DROP_PKT: Dropping tcp session 20.0.0.3:49892 34.122.121.32:80 on zone-pair DMZ-TO-OUT class class-defaul
t due to  DROP action found in policy-map with ip ident 0
```

Figure 108 - Firewall Log messages

2.3. Defence against DoS attacks

This exercise addresses the use firewalls to defend from two types of DoS attacks: ICMP flood and TCP SYN flood.

In the first attack, a fast stream of TCP packets with the SYN flag set is sent to the victim. In this case, the victim is forced to open many TCP connections, which remain half-open since the TCP ACK packet that finishes the 3-way handshake is never sent to server.

The second attack consists in sending a fast stream of ICMP Echo Request packets to the victim.

Network Architecture

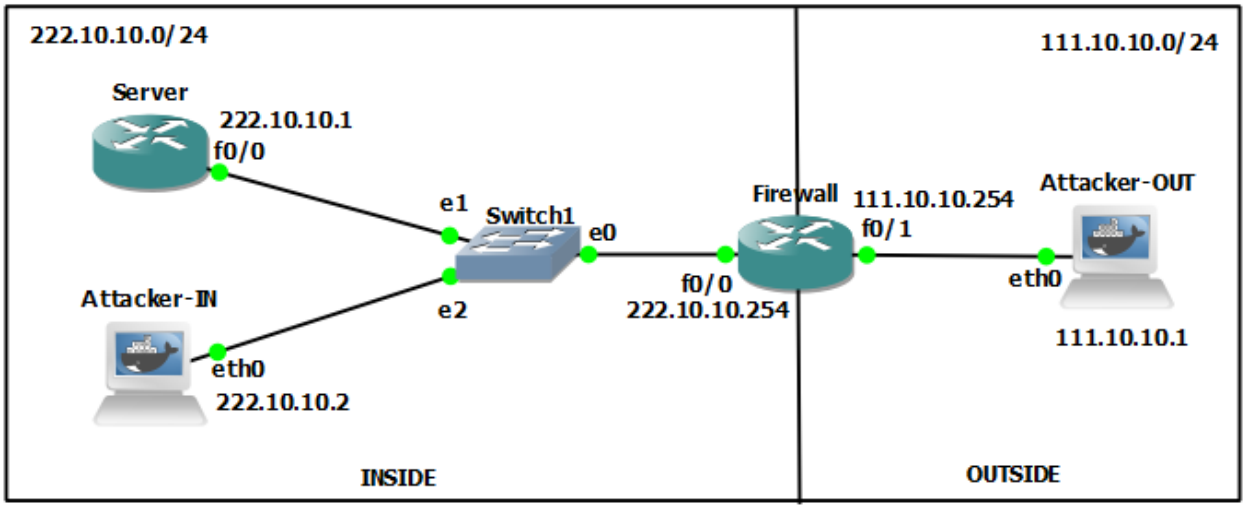
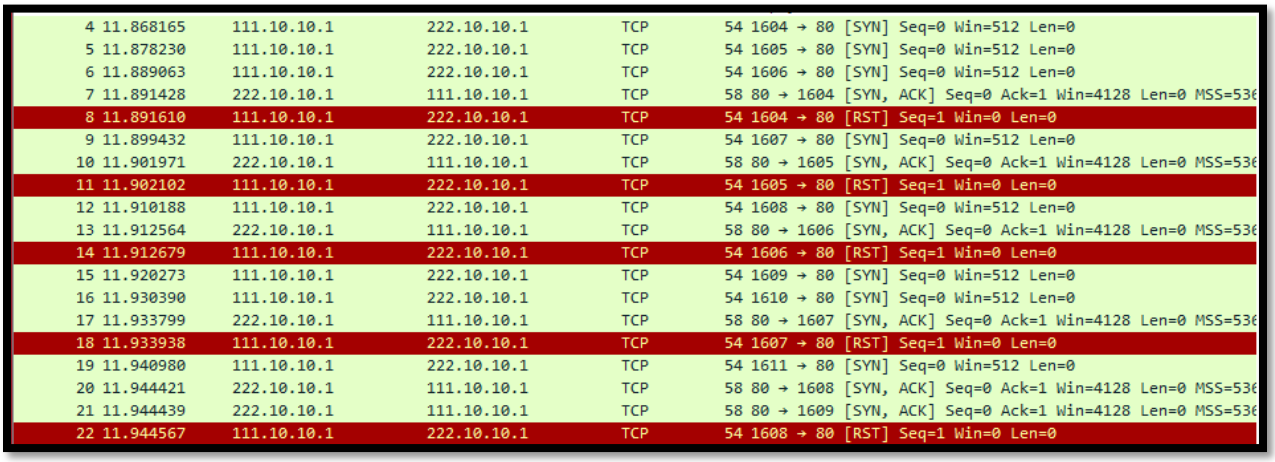


Figure 109 - Defence against DoS Network Architecture

In this network topology we will use the same as in the laboratory exercise 2.1.

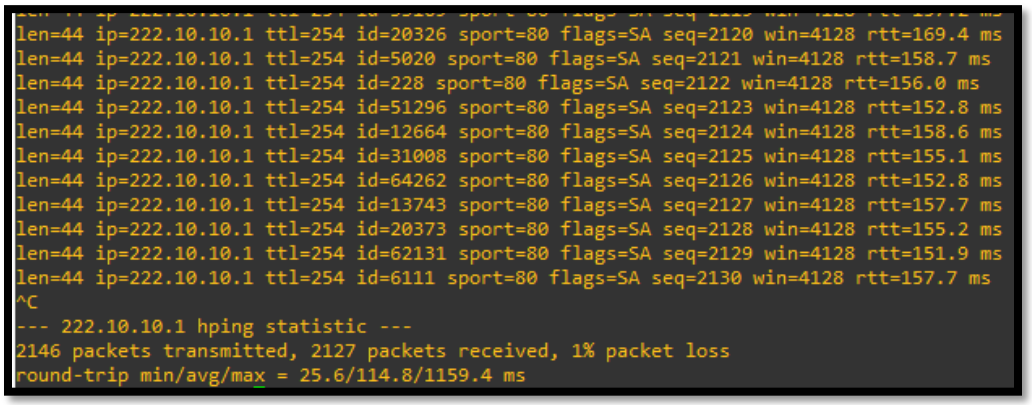
Proof of Concept

1. After configuring the network topology presented before, the Server allows access using both HTTP and ICMP protocols.
   Then the outside attacker uses the command: **hping3 -i u10000 -S -p 80 222.10.10.1** against the server. However, the attack is unsuccessful as shown in the figures bellow, where we can see that the target can still respond to the TCP SYN messages, Figure 110 and Figure 111, and drops most of the connections Figure 112.



4	11.868165	111.10.10.1	222.10.10.1	TCP	54 1604 → 80 [SYN] Seq=0 Win=512 Len=0
5	11.878230	111.10.10.1	222.10.10.1	TCP	54 1605 → 80 [SYN] Seq=0 Win=512 Len=0
6	11.889063	111.10.10.1	222.10.10.1	TCP	54 1606 → 80 [SYN] Seq=0 Win=512 Len=0
7	11.891428	222.10.10.1	111.10.10.1	TCP	58 80 → 1604 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MSS=536
8	11.891610	111.10.10.1	222.10.10.1	TCP	54 1604 → 80 [RST] Seq=1 Win=0 Len=0
9	11.899432	111.10.10.1	222.10.10.1	TCP	54 1607 → 80 [SYN] Seq=0 Win=512 Len=0
10	11.901971	222.10.10.1	111.10.10.1	TCP	58 80 → 1605 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MSS=536
11	11.902102	111.10.10.1	222.10.10.1	TCP	54 1605 → 80 [RST] Seq=1 Win=0 Len=0
12	11.910188	111.10.10.1	222.10.10.1	TCP	54 1608 → 80 [SYN] Seq=0 Win=512 Len=0
13	11.912564	222.10.10.1	111.10.10.1	TCP	58 80 → 1606 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MSS=536
14	11.912679	111.10.10.1	222.10.10.1	TCP	54 1606 → 80 [RST] Seq=1 Win=0 Len=0
15	11.920273	111.10.10.1	222.10.10.1	TCP	54 1609 → 80 [SYN] Seq=0 Win=512 Len=0
16	11.930390	111.10.10.1	222.10.10.1	TCP	54 1610 → 80 [SYN] Seq=0 Win=512 Len=0
17	11.933799	222.10.10.1	111.10.10.1	TCP	58 80 → 1607 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MSS=536
18	11.933938	111.10.10.1	222.10.10.1	TCP	54 1607 → 80 [RST] Seq=1 Win=0 Len=0
19	11.940980	111.10.10.1	222.10.10.1	TCP	54 1611 → 80 [SYN] Seq=0 Win=512 Len=0
20	11.944421	222.10.10.1	111.10.10.1	TCP	58 80 → 1608 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MSS=536
21	11.944439	222.10.10.1	111.10.10.1	TCP	58 80 → 1609 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MSS=536
22	11.944567	111.10.10.1	222.10.10.1	TCP	54 1608 → 80 [RST] Seq=1 Win=0 Len=0

Figure 110 - Wireshark TCP flood attack from one IP



```
len=44 ip=222.10.10.1 ttl=254 id=20326 sport=80 flags=SA seq=2120 win=4128 rtt=169.4 ms
len=44 ip=222.10.10.1 ttl=254 id=5020 sport=80 flags=SA seq=2121 win=4128 rtt=158.7 ms
len=44 ip=222.10.10.1 ttl=254 id=228 sport=80 flags=SA seq=2122 win=4128 rtt=156.0 ms
len=44 ip=222.10.10.1 ttl=254 id=51296 sport=80 flags=SA seq=2123 win=4128 rtt=152.8 ms
len=44 ip=222.10.10.1 ttl=254 id=12664 sport=80 flags=SA seq=2124 win=4128 rtt=158.6 ms
len=44 ip=222.10.10.1 ttl=254 id=31008 sport=80 flags=SA seq=2125 win=4128 rtt=155.1 ms
len=44 ip=222.10.10.1 ttl=254 id=64262 sport=80 flags=SA seq=2126 win=4128 rtt=152.8 ms
len=44 ip=222.10.10.1 ttl=254 id=13743 sport=80 flags=SA seq=2127 win=4128 rtt=157.7 ms
len=44 ip=222.10.10.1 ttl=254 id=20373 sport=80 flags=SA seq=2128 win=4128 rtt=155.2 ms
len=44 ip=222.10.10.1 ttl=254 id=62131 sport=80 flags=SA seq=2129 win=4128 rtt=151.9 ms
len=44 ip=222.10.10.1 ttl=254 id=6111 sport=80 flags=SA seq=2130 win=4128 rtt=157.7 ms
^C
--- 222.10.10.1 hping statistic ---
2146 packets transmitted, 2127 packets received, 1% packet loss
round-trip min/avg/max = 25.6/114.8/1159.4 ms
```

Figure 111 - TCP flood attack

```
Server#show TCP statistics
Rcvd: 7864 Total, 0 no port
      0 checksum error, 0 bad offset, 0 too short
      0 packets (0 bytes) in sequence
      0 dup packets (0 bytes)
      0 partially dup packets (0 bytes)
      0 out-of-order packets (0 bytes)
      0 packets (0 bytes) with data after window
      0 packets after close
      0 window probe packets, 0 window update packets
      0 dup ack packets, 0 ack packets with unsend data
      0 ack packets (0 bytes)
Sent: 3924 Total, 0 urgent packets
      3924 control packets (including 0 retransmitted)
      0 data packets (0 bytes)
      0 data packets (0 bytes) retransmitted
      0 data packets (0 bytes) fastretransmitted
      0 ack only packets (0 delayed)
      0 window probe packets, 0 window update packets
0 Connections initiated, 3924 connections accepted, 0 connections established
3912 Connections closed (including 3912 dropped, 0 embryonic dropped)
0 Total rxmt timeout, 0 connections dropped in rxmt timeout
0 Keepalive timeout, 0 keepalive probe, 0 Connections dropped in keepalive
```

Figure 112 - TCP statistics from Server

2. We now use the command: **hping3 -i u10000 -S -p 80 --rand-source 222.10.10.1**, and we can see that the attack was successful in the figures bellow. This is because in this attack we use random source IP addresses as seen in Figure 114, and the target cannot respond to all these requests, as it can be seen in Figure 113 and in Wireshark. Also using **show TCP statistics** on the server, we can see that 0 connections were dropped.

```
root@Attacker-OUT:/#  hping3 -i u10000 -S -p 80 --rand-source 222.10.10.1
HPING 222.10.10.1 (eth0 222.10.10.1): S set, 40 headers + 0 data bytes
^C
--- 222.10.10.1 hping statistic ---
25030 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@Attacker-OUT:/#
```

Figure 113 - TCP flood random IP attack

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1950	85.849415	30.100.152.39	222.10.10.1	TCP	54	4296 → 80 [SYN] Seq=0 Win=512 Len=0
1951	85.860339	203.176.87.81	222.10.10.1	TCP	54	4297 → 80 [SYN] Seq=0 Win=512 Len=0
1952	85.871430	175.218.116.156	222.10.10.1	TCP	54	4298 → 80 [SYN] Seq=0 Win=512 Len=0
1953	85.882215	163.32.177.42	222.10.10.1	TCP	54	4299 → 80 [SYN] Seq=0 Win=512 Len=0
1954	85.893487	58.4.94.13	222.10.10.1	TCP	54	4300 → 80 [SYN] Seq=0 Win=512 Len=0
1955	85.904307	70.251.58.146	222.10.10.1	TCP	54	4301 → 80 [SYN] Seq=0 Win=512 Len=0
1956	85.915141	140.195.231.42	222.10.10.1	TCP	54	4302 → 80 [SYN] Seq=0 Win=512 Len=0
1957	85.925320	178.16.64.182	222.10.10.1	TCP	54	4303 → 80 [SYN] Seq=0 Win=512 Len=0
1958	85.936106	24.51.183.32	222.10.10.1	TCP	54	4304 → 80 [SYN] Seq=0 Win=512 Len=0
1959	85.946361	130.29.176.136	222.10.10.1	TCP	54	4305 → 80 [SYN] Seq=0 Win=512 Len=0
1960	85.957169	178.64.64.195	222.10.10.1	TCP	54	4306 → 80 [SYN] Seq=0 Win=512 Len=0
1961	85.968046	134.130.213.121	222.10.10.1	TCP	54	4307 → 80 [SYN] Seq=0 Win=512 Len=0
1962	85.978922	101.2.70.169	222.10.10.1	TCP	54	4308 → 80 [SYN] Seq=0 Win=512 Len=0
1963	85.989903	87.25.26.12	222.10.10.1	TCP	54	4309 → 80 [SYN] Seq=0 Win=512 Len=0
1964	86.000927	173.153.201.171	222.10.10.1	TCP	54	4310 → 80 [SYN] Seq=0 Win=512 Len=0
1965	86.011853	116.32.151.0	222.10.10.1	TCP	54	4311 → 80 [SYN] Seq=0 Win=512 Len=0
1966	86.022829	183.89.249.35	222.10.10.1	TCP	54	4312 → 80 [SYN] Seq=0 Win=512 Len=0
1967	86.033872	254.139.42.156	222.10.10.1	TCP	54	4313 → 80 [SYN] Seq=0 Win=512 Len=0
1968	86.044712	166.165.81.188	222.10.10.1	TCP	54	4314 → 80 [SYN] Seq=0 Win=512 Len=0
1969	86.055915	116.9.232.17	222.10.10.1	TCP	54	4315 → 80 [SYN] Seq=0 Win=512 Len=0
1970	86.066767	208.112.72.134	222.10.10.1	TCP	54	4316 → 80 [SYN] Seq=0 Win=512 Len=0
1971	86.077633	219.44.0.26	222.10.10.1	TCP	54	4317 → 80 [SYN] Seq=0 Win=512 Len=0

Figure 114 - Wireshark capture of TCP Flood with random IPs

```
Server#show TCP statistics
Rcvd: 21852 Total, 0 no port
      0 checksum error, 0 bad offset, 0 too short
      0 packets (0 bytes) in sequence
      0 dup packets (0 bytes)
      0 partially dup packets (0 bytes)
      0 out-of-order packets (0 bytes)
      0 packets (0 bytes) with data after window
      0 packets after close
      0 window probe packets, 0 window update packets
      0 dup ack packets, 0 ack packets with unsend data
      0 ack packets (0 bytes)
Sent: 611 Total, 0 urgent packets
      611 control packets (including 67 retransmitted)
      0 data packets (0 bytes)
      0 data packets (0 bytes) retransmitted
      0 data packets (0 bytes) fastretransmitted
      0 ack only packets (0 delayed)
      0 window probe packets, 0 window update packets
0 Connections initiated, 544 connections accepted, 0 connections established
528 Connections closed (including 0 dropped, 0 embryonic dropped)
67 Total rxmt timeout, 0 connections dropped in rxmt timeout
0 Keepalive timeout, 0 keepalive probe, 0 Connections dropped in keepalive
Server#
```

Figure 115 - TCP statistics from server with 0 dropped connections

Countermeasures

1. The first countermeasure will be the configuration of a rate limiting policy. To do so we added the command: **police rate 128000 burst 8000** in each class type. To test this countermeasure, we executed both ICMP and TCP flood attacks.

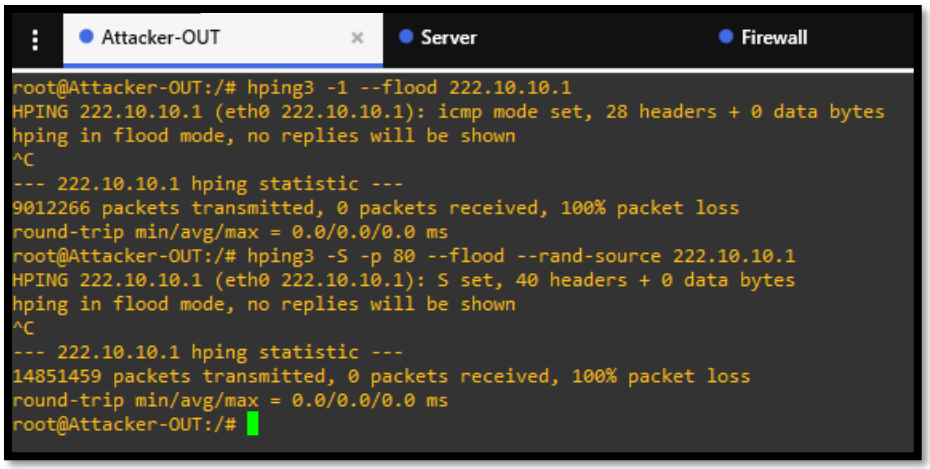


Figure 116 - ICMP and TCP flood attacks

```
 Class-map: WWW_CLASSMAP (match-all)
   Match: access-group name TO_IN_HOST
   Match: protocol http

Inspect
     Packet inspection statistics [process switch:fast switch]
     tcp packets: [78:410]

     Session creations since subsystem startup or last reset 5016
     Current session counts (estab/half-open/terminating) [0:0:0]
     Maxever session counts (estab/half-open/terminating) [0:4808:0]
     Last session created 00:00:34
     Last statistic reset never
     Last session creation rate 1488
     Maxever session creation rate 5016
     Last half-open session total 0
     TCP reassembly statistics
     received 0 packets out-of-order; dropped 0
     peak memory usage 0 KB; current usage: 0 KB
     peak queue length 0

    Police
     rate 8000 bps,1000 limit
     conformed 488 packets, 26352 bytes; actions: transmit
     exceeded 4332 packets, 233928 bytes; actions: drop
     conformed 0 bps, exceed 3000 bps
```

Figure 117 - HTTP classmap Police drops result

```
 Class-map: ICMP_CLASSMAP (match-all)
   Match: access-group name TO_IN_HOST
   Match: protocol icmp

Inspect
     Packet inspection statistics [process switch:fast switch]
     icmp packets: [1:630]

     Session creations since subsystem startup or last reset 1
     Current session counts (estab/half-open/terminating) [0:1:0]
     Maxever session counts (estab/half-open/terminating) [0:1:0]
     Last session created 00:00:25
     Last statistic reset never
     Last session creation rate 1
     Maxever session creation rate 1
     Last half-open session total 1
     TCP reassembly statistics
     received 0 packets out-of-order; dropped 0
     peak memory usage 0 KB; current usage: 0 KB
     peak queue length 0

    Police
     rate 8000 bps,1000 limit
     conformed 632 packets, 26544 bytes; actions: transmit
     exceeded 122961 packets, 5174820 bytes; actions: drop
     conformed 4000 bps, exceed 89000 bps

 Class-map: class-default (match-any)
   Match: any
   Drop
     0 packets, 0 bytes
```

Figure 118 - ICMP classmap police drops results

2. The second countermeasure is to protect against the TCP SYN flood attack, by taking the initiative of removing half-open connections when it detects an attack of this type. This requires configuring a parameter map. A parameter map allows you to specify parameters that control the behaviour of actions and match criteria specified under a policy map and a class map, respectively (CISCO, 2022).

```
1.  parameter-map type inspect PARMAP
2.  tcp synwait-time 3
3.  end
```

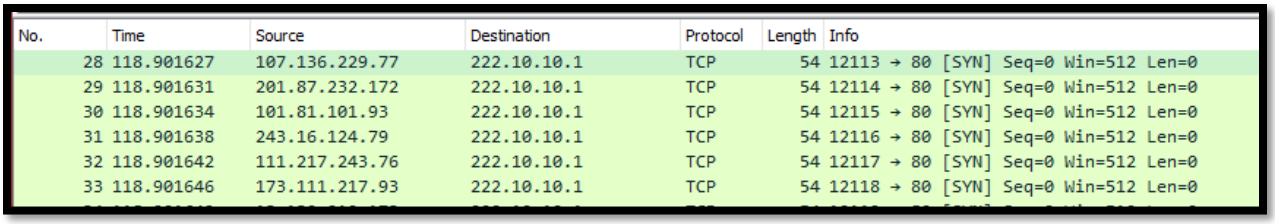
3. To demonstrate this countermeasure, we apply a parameter map that sets a timer of 3 seconds after a SYN TCP message. If that timer expires the ZBPF closes the half-open sessions of the Server.

```
Server#show TCP statistics
Rcvd: 1524 Total, 360 no port
      0 checksum error, 0 bad offset, 0 too short
      0 packets (0 bytes) in sequence
      0 dup packets (0 bytes)
      0 partially dup packets (0 bytes)
      0 out-of-order packets (0 bytes)
      0 packets (0 bytes) with data after window
      0 packets after close
      0 window probe packets, 0 window update packets
      0 dup ack packets, 0 ack packets with unsend data
      0 ack packets (0 bytes)
Sent: 35 Total, 0 urgent packets
      35 control packets (including 3 retransmitted)
      0 data packets (0 bytes)
      0 data packets (0 bytes) retransmitted
      0 data packets (0 bytes) fastretransmitted
      0 ack only packets (0 delayed)
      0 window probe packets, 0 window update packets
0 Connections initiated, 32 connections accepted, 0 connections established
16 Connections closed (including 0 dropped, 0 embryonic dropped)
3 Total rxmt timeout, 0 connections dropped in rxmt timeout
0 Keepalive timeout, 0 keepalive probe, 0 Connections dropped in keepalive
Server#
```

Figure 119 - TCP statistics

```
Service-policy inspect : DOS_POLICYMAP

  Class-map: WWW_CLASSMAP (match-all)
    Match: access-group name TO_IN_HOST
    Match: protocol http

 Inspect
      Packet inspection statistics [process switch:fast switch]
      tcp packets: [153:2684]

      Session creations since subsystem startup or last reset 8137
      Current session counts (estab/half-open/terminating) [0:0:0]
      Maxever session counts (estab/half-open/terminating) [0:4808:0]
      Last session created 00:00:04
      Last statistic reset never
      Last session creation rate 3121
      Maxever session creation rate 5016
      Last half-open session total 0
      TCP reassembly statistics
      received 0 packets out-of-order; dropped 0
      peak memory usage 0 KB; current usage: 0 KB
      peak queue length 0

     Police
      rate 128000 bps,8000 limit
      conformed 2837 packets, 153198 bytes; actions: transmit
      exceeded 4649 packets, 251046 bytes; actions: drop
      conformed 4000 bps, exceed 2000 bps
```

Figure 120 - HTTP Policy map



No.	Time	Source	Destination	Protocol	Length	Info
28	118.901627	107.136.229.77	222.10.10.1	TCP	54	12113 → 80 [SYN] Seq=0 Win=512 Len=0
29	118.901631	201.87.232.172	222.10.10.1	TCP	54	12114 → 80 [SYN] Seq=0 Win=512 Len=0
30	118.901634	101.81.101.93	222.10.10.1	TCP	54	12115 → 80 [SYN] Seq=0 Win=512 Len=0
31	118.901638	243.16.124.79	222.10.10.1	TCP	54	12116 → 80 [SYN] Seq=0 Win=512 Len=0
32	118.901642	111.217.243.76	222.10.10.1	TCP	54	12117 → 80 [SYN] Seq=0 Win=512 Len=0
33	118.901646	173.111.217.93	222.10.10.1	TCP	54	12118 → 80 [SYN] Seq=0 Win=512 Len=0

Figure 121 - Wireshark capture of TCP SYN Flood

3. Lab No. 1.3 – AAA

The aim of this laboratory work is to study AAA. The performed experiments are:

(i) AAA with TACACS+
(ii) 802.1X authentication

3.1. AAA with TACACS+

TACACS+, which stands for Terminal Access Controller Access Control Server, is a protocol that handles authentication, authorization, and accounting (AAA) services. AAA refers to Authentication (to identify), Authorization (to give permission) and Accounting (to log resource usage). TACACS+ is a remote authentication protocol that will allow a remote access server to communicate with an authentication server to validate user access to the networking devices.

Network Topology

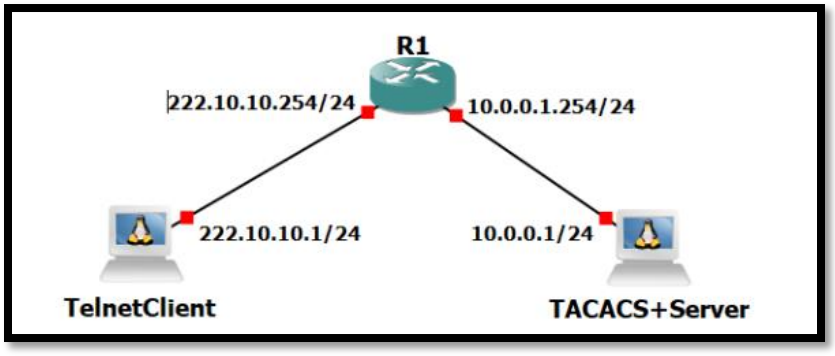


Figure 122 - AAA with TACACS+ Network Topology

After finishing the TACACS configuration as indicated in the Lab guide, we will test it by trying to connect to R1 via telnet. We set up a Wireshark capture in the link between the R1 and the AAA server to see the messages exchanged.

Proof-of-Concept

In order to observe the process, we set up a Wireshark capture between the links that connect the client and R1 and R1 to the AAA server.

Firstly, we can see in Figure 123 an Authentication message from R1 to the AAA server with an authentication request, and in Figure 124, we see the respective response containing the message displayed on the client's terminal: "User Access Verification\n\Username:".

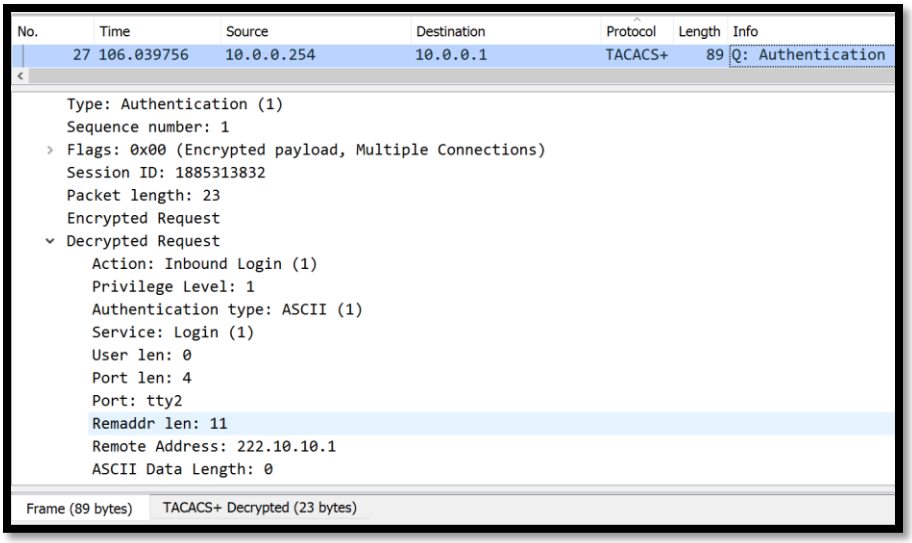


Figure 123 - Authentication Request from router to AAA Server

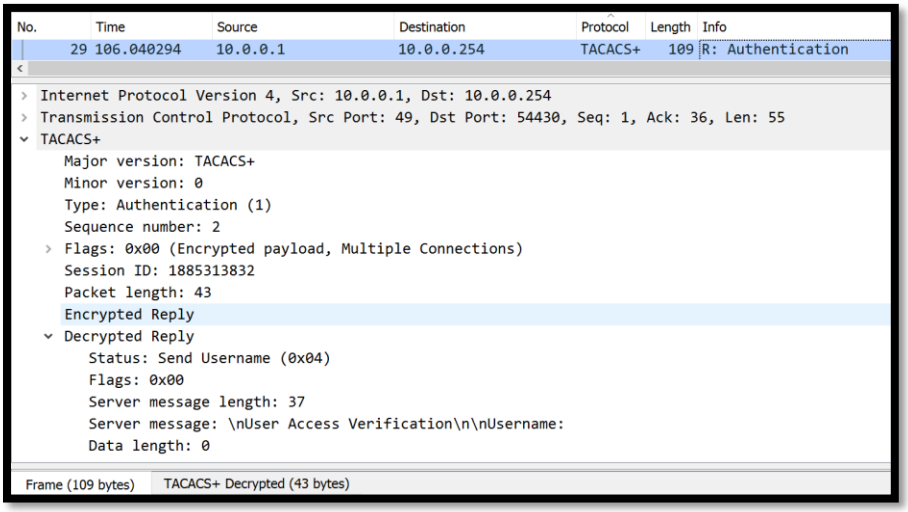


Figure 124 - Response Message of the Authentication Request

After writing the username in the terminal (in our case is “gns3”) we can see in Figure 125 a packet containing the username and in Figure 126 the response, again with the information displayed in the client's terminal asking for the password.

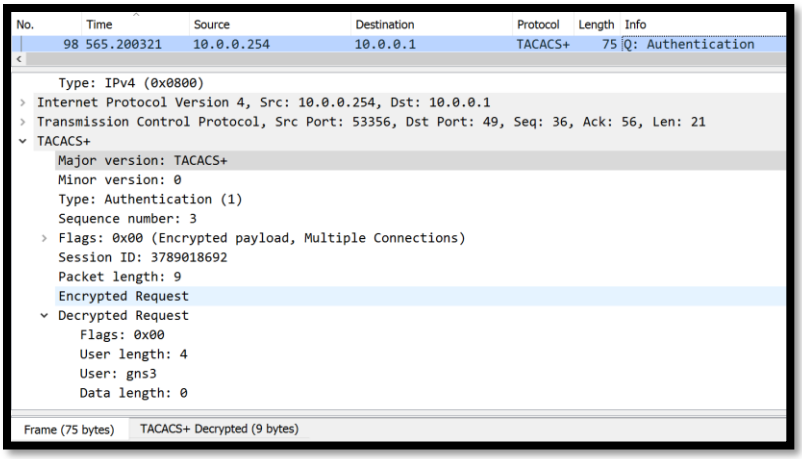


Figure 125 - Packet with username

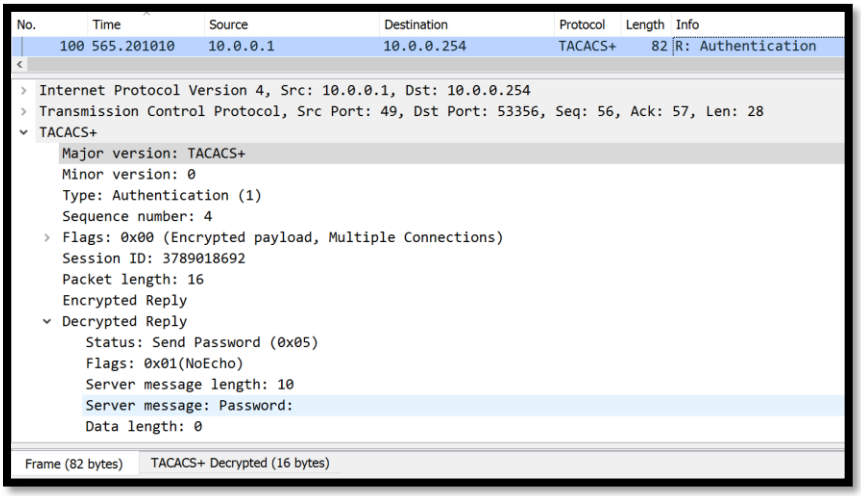


Figure 126 - Packet containing the Password

After writing the password (in our case “admin”) we can see in Figure 127 the packet containing the password that was written in the terminal and in Figure 128 the response indicating that the Authentication process was successful.

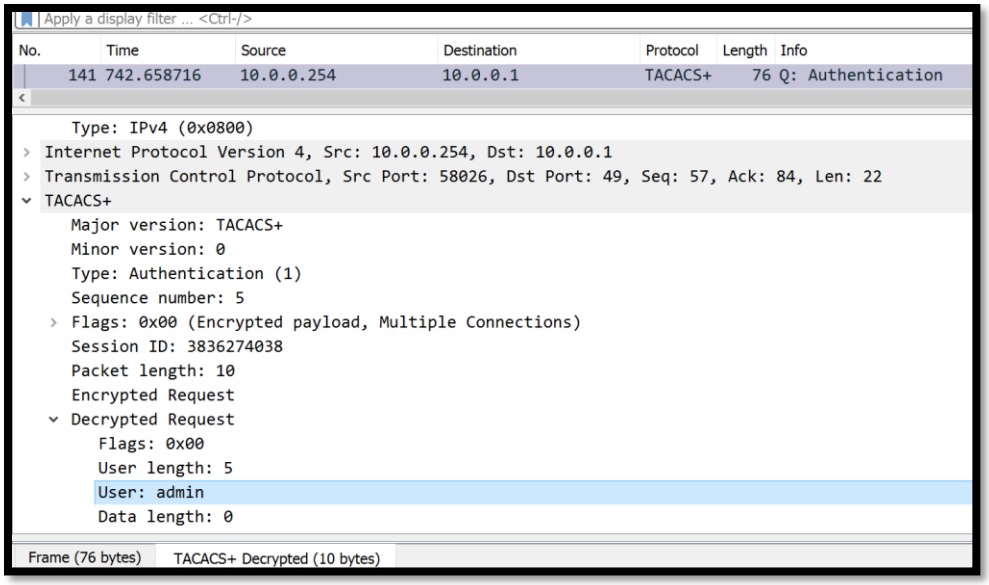


Figure 127 - Packet containing password

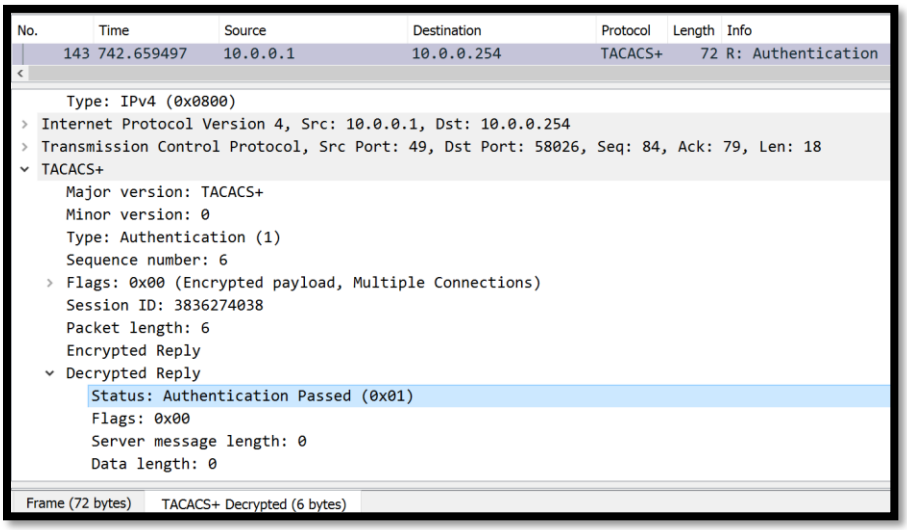


Figure 128 - Response Authentication successful

It is essential to point out that various keys are involved in the TACACS process. There is a key to protecting the communication between R1 and the AAA server. In our case, this key is “gns3”.

There are also keys associated with the users. In order to change the password associated with the user "gns3", we used the tool ***tac_pwd***. The tool inputs the password to be used and returns the cyphered password using DES hash. This process can be seen in Figure 129.

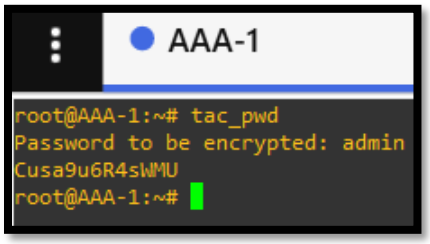


Figure 129 - Cyphered Password

We can now take the cyphered password and associate it with the correspondent user in the TACACS configuration file. This can be seen in the figure bellow.

```
# Much more features are availables, like ACL, more service compatibilities,
# commands authorization, scripting authorization.
# See the man page for those features.

user = gns3 {
    name = "Admin User"
    member = admin
    login = des Cusa9u6R4sWMU
                service = junos-exec {
                        local-user-name = remote-admin
        }
}
```

Figure 130 - TACACS configuration file

We found the following packets (Figure 131 and Figure 132) relevant for the Authorization phase. We can see the user and the request for the shell service in Figure 131, and in the following figure, we can see that the privilege level attributed to the user "gns3" is privilege level 15, meaning the user can perform all operations in the device.

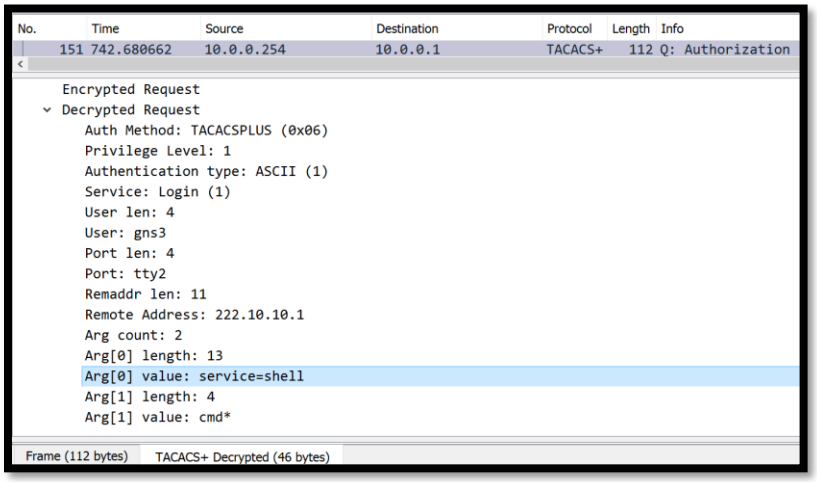


Figure 131 - User and the request for the shell service

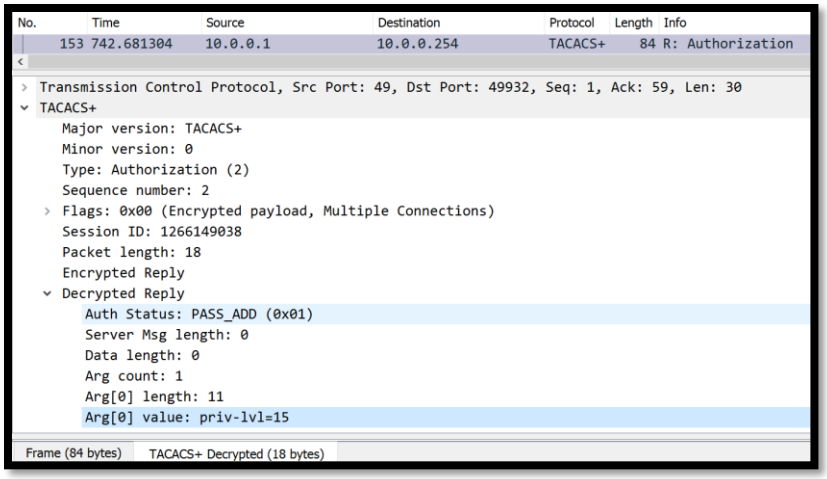


Figure 132 - Privilege level attributed to the user

Regarding Accounting, we can see the following packets containing information regarding the connection's starting and the end.

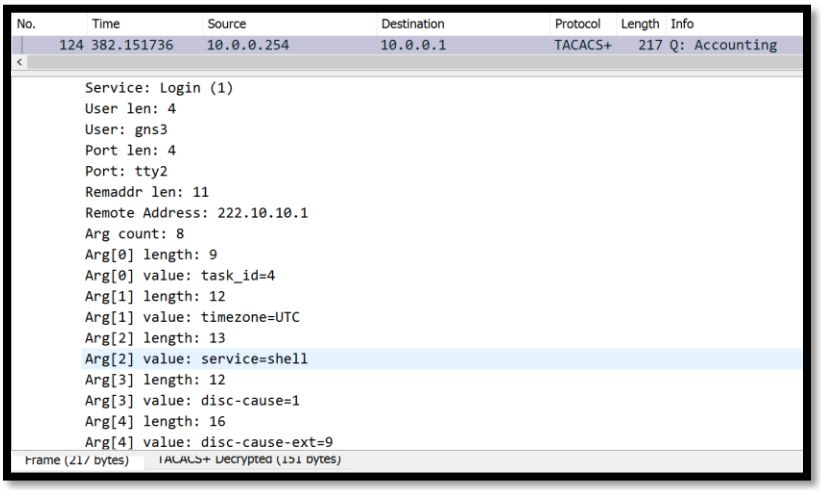


Figure 133 - Accounting start

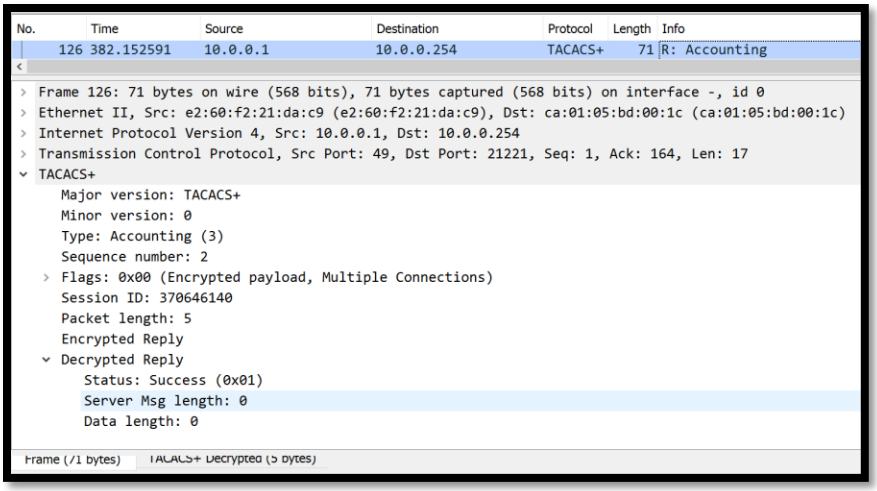


Figure 134 - Accounting response

When it comes to the benefits of server base AAA technology such as TACACS+, it allows using locally configured users or users and groups defined in Active Directory or LDAP to control access to devices in the network. This enables Single Sign-On (SSO), increasing **security, simplifying management and its easier for users**. Server-based AAA allows centralized management of user permissions individually or by applying group policies to a group of users.

3.2. 802.1X Authentication

802.1X Authentication provides port-based network access control. It involves an authentication server called a RADIUS Server. It checks a user's credentials to see if they are an active member of the organization and, depending on the network policies, grants users varying levels of access to the network.

Network Architecture

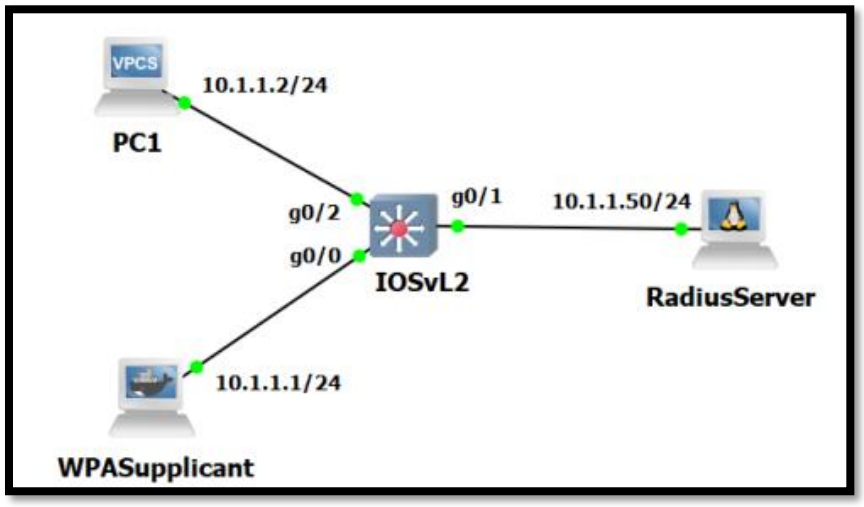


Figure 135 - 802.1X Authentication Network Architecture

After the port/interface configuration as indicated and after creating the **wpa.conf** file, we can confirm that the PC1 cannot access the AAA server, but the Supplicant can. In order to inspect how the 802.1X Authentication process works, we set up a Wireshark connection between the Supplicant and SW1 and between the SW1 and the AAA server.

Proof-of-Concept

First, we can see an EAP request and an EAP response in Figure 137 and Figure 138, respectively. In the request packet, the client will be asked for its username. The client will then respond with the user in the response packet.

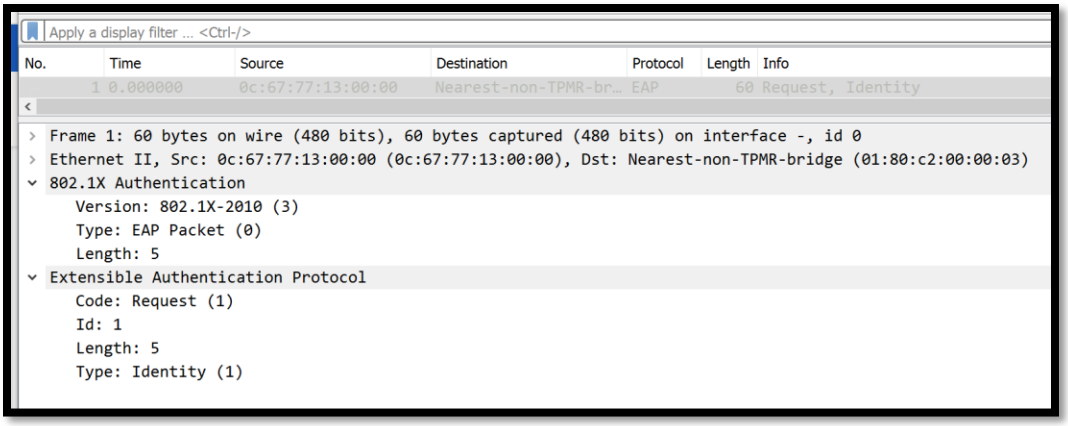


Figure 136 - EAP Request message

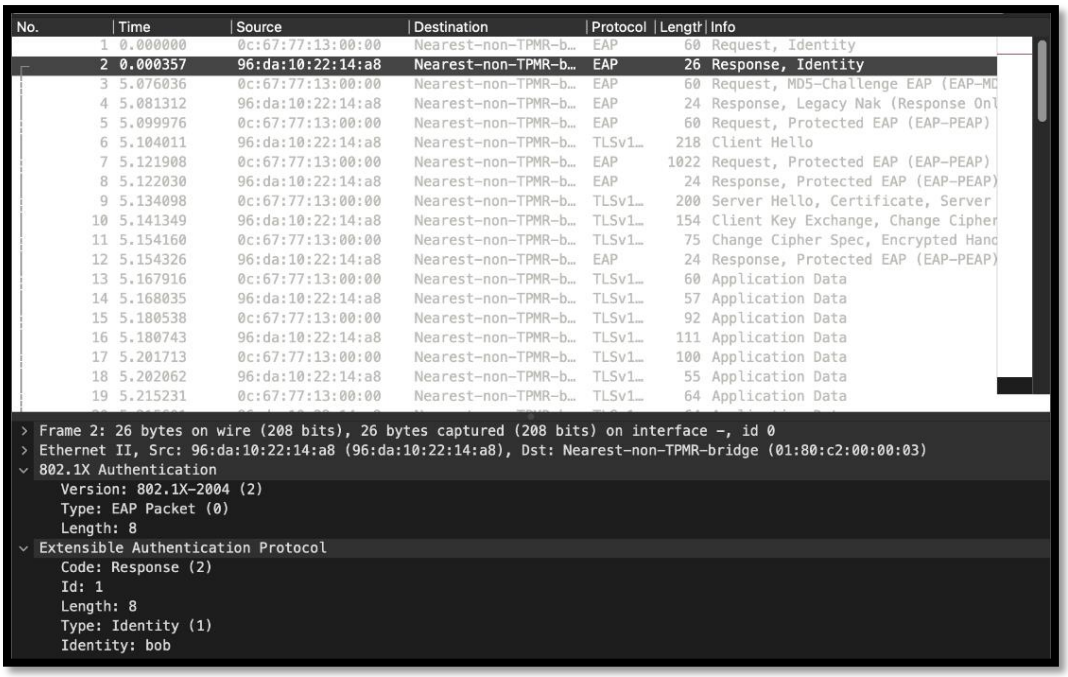


Figure 137 - EAP Response Message

The SW will then send an Access-Request packet to test the user against the RADIUS server as can be seen in Figure 138.

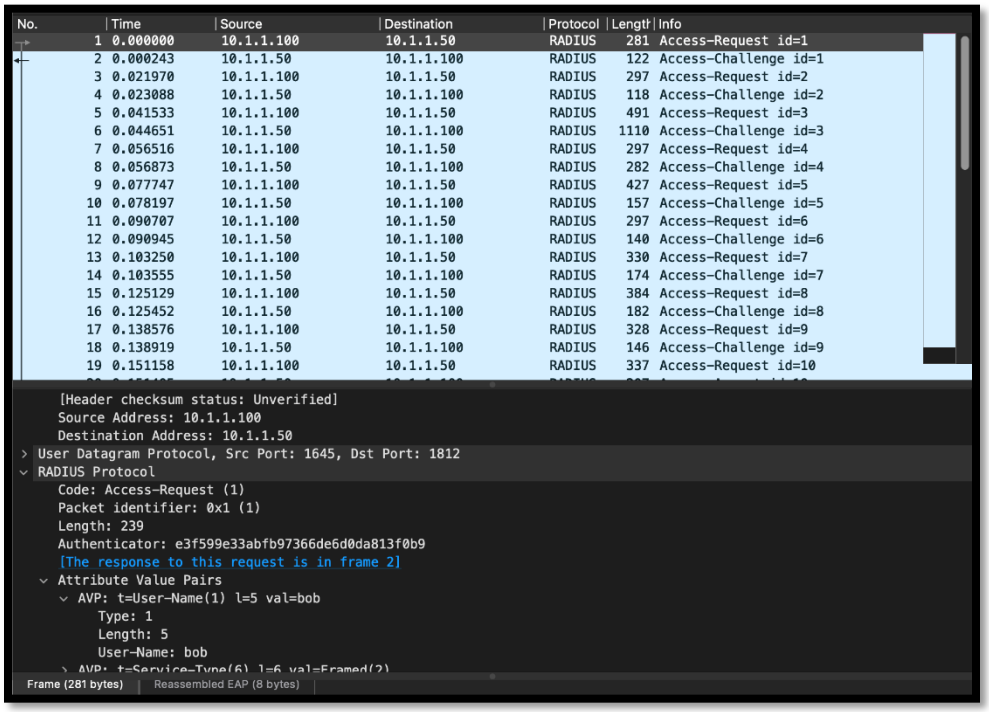


Figure 138 - Access-request with identity Bob

The AAA will send a challenge back to the client as we can see in the Figure 139 and Figure 140.

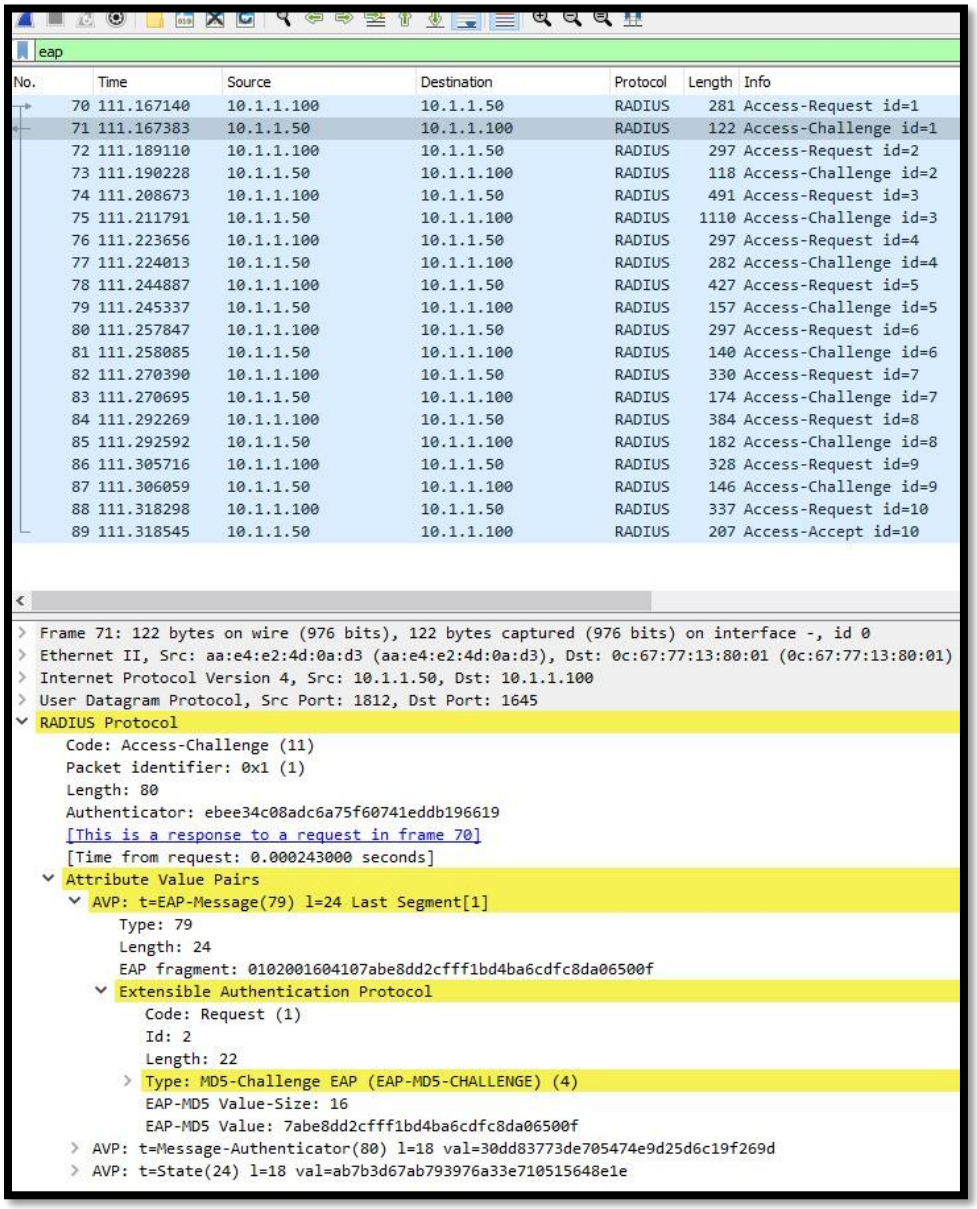


Figure 139 - Access challenge in MD5

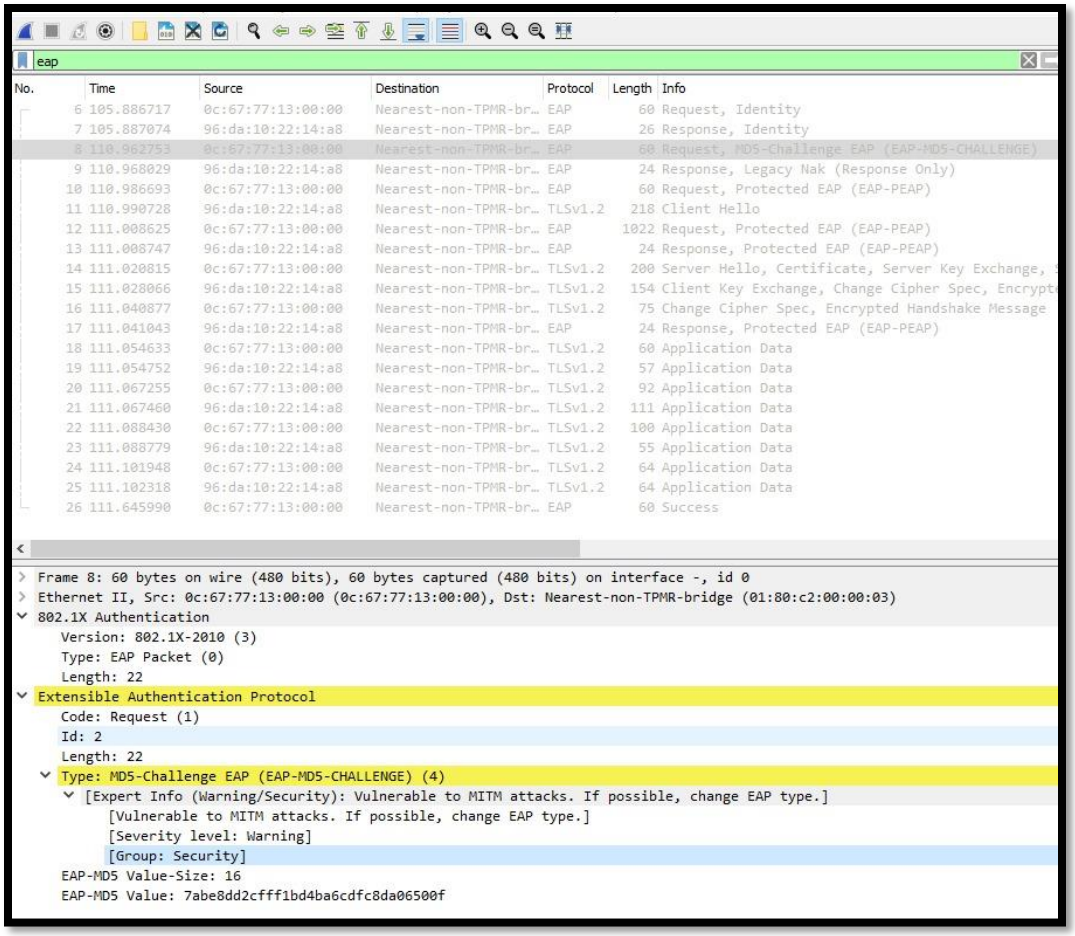


Figure 140 - Challenge that will reach the client

In the next step of the process, the client negotiates the encryption protocol to be used. In this case, it will be used EAP-PEAP protocol. Once this step is completed, a TLS tunnel is established between the client and the AAA server.

The TLS encrypts traffic, so we cannot see the data in the packets captured, as we can see in Figure 141 and Figure 142.

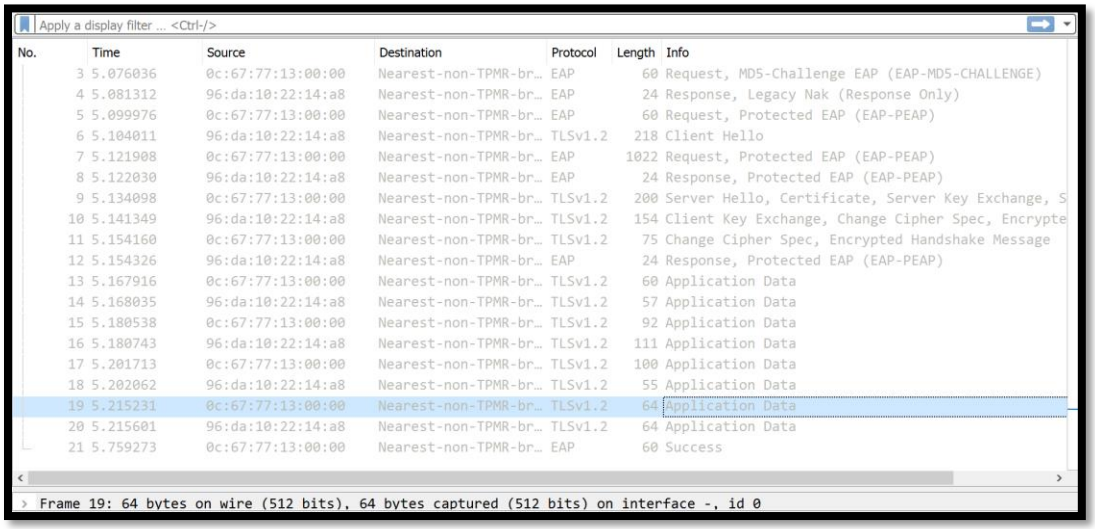


Figure 141 - Auth method negotiation and TLS channel

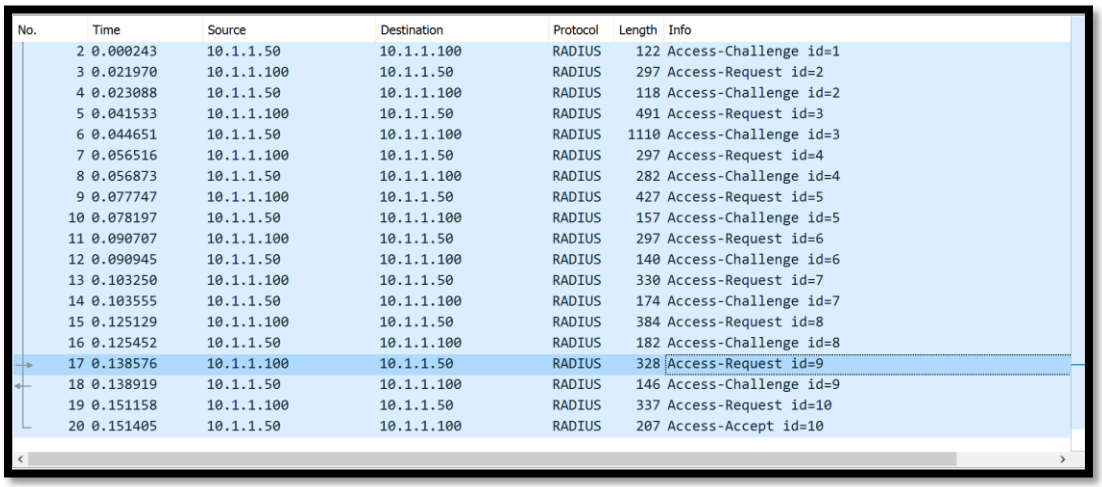


Figure 142 - Authentication process in the RADIUS server

When the Authentication phase is completed, the server sends an Access-accepted, as seen in Figure 143, and the client receives an EAP success message, as seen in Figure 144.

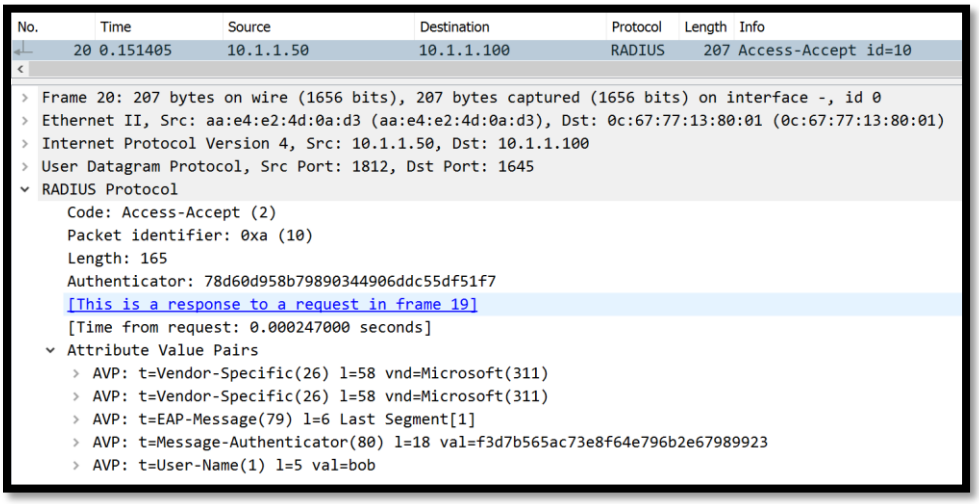


Figure 143 - Access-Accept message

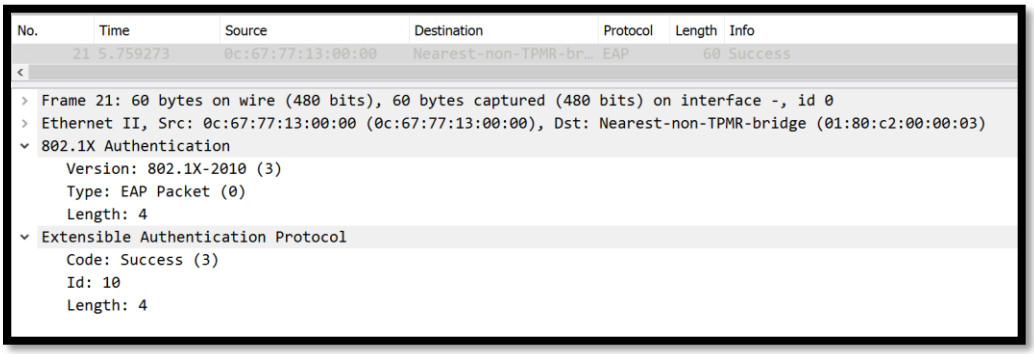


Figure 144 - EAP success message

This authentication method helps prevent man-in-the-middle attacks like ARP poisoning and Root Bridge spoofing present in lab 1. Having an authenticator responsible for patrolling and making those connections secure prevents any man-in-the-middle attack since they cannot make any concrete connections in the network.

References

Cisco. (2022). *CCNA Security: Implementing Network Security 2.01*. Retrieved from Network Academy.

CISCO. (2022). *Security Configuration Guide: Zone-Based Policy Firewall, Cisco IOS Release 15M&T*. Retrieved from Cisco.com: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_data_zbf/configuration/15-mt/sec-data-zbf-15-mt-book/sec-zone-pol-fw.html#GUID-3919A834-5643-4437-BACB-9B5B00E8B776

Nmap. (2022, May 17). *A Quick Port Scanning Tutorial*. Retrieved from Nmap.org: https://nmap.org/book/port-scanning-tutorial.html

Nmap. (2022, May 17). *Host Discovery*. Retrieved from Nmap.org: https://nmap.org/book/man-host-discovery.html

Valadas, R. (2022, May 09). *Lab guide No. 1.1 – Network attacks and countermeasures.* Lisboa: Instituto Superior Tecnico.

Annex A: Configurations

DHCP Spoofing:

Attack:

- **c3725 Router:**

```
conf t
ip dhcp pool 0
network 20.0.0.0 /24
dns-server 1.1.1.1
exit
int f0/0
ip address 20.0.0.1 255.0.0.0
no shut
end
show ip dhcp pool
show ip dhcp binding
```

- **Victim 1 & 2:**

```
# edit config -> Uncomment the line to enable DHCP
ip dhcp
```

- **IOSvL2:**

```
no cdp advertise-v2
```

- **Thunder Marks-Kali:**

```
ettercap -T -M dhcp:20.0.0.0/255.255.255.0/1.1.1.1
```

Countermeasures:

- **Switch 1:**

```
enable
conf t
ip dhcp snooping
no ip dhcp snooping information option
int gi0/2
ip dhcp snooping trust
int gi0/3
ip dhcp snooping trust
int gi0/1
ip dhcp snooping limit rate 6
exit
ip dhcp snooping vlan 1
```

- **Switch 2:**

```
1.  enable
2.  conf t
3.  ip dhcp snooping
4.  no ip dhcp snooping information option
5.  int gi0/0
6.  ip dhcp snooping trust
7.  int gi0/1
8.  ip dhcp snooping limit rate 6
9.  int gi0/2
10. ip dhcp snooping limit rate 6
11. exit
12. ip dhcp snooping vlan 1
```

ARP Poisoning:

Attack:

- **DHCP_Server (c3725 Router):**

```
1.  conf t
2.  ip dhcp pool 0
3.  network 20.0.0.0 /24
4.  dns-server 1.1.1.1
5.  exit
6.  int f0/0
7.  ip address 20.0.0.1 255.0.0.0
8.  no shut
9.  end
10. show ip dhcp pool
11. show ip dhcp binding
```

- **Switch 1 (IOSvL2):**

```
1.  no cdp advertise-v2
```

- **Victim 1 & 2 (c3725 Router):**

```
1.  enable
2.  conf t
3.  no ip routing
4.  int f0/0
5.  ip address dhcp
6.  no shutdown
7.  end
8.  sh arp
```

- **Attacker:**

```
1.  python3 arpmitm.py eth0 20.0.0.3 20.0.0.4
```

Countermeasures:

- **Switch 1:**

```
1.  enable
2.  conf t
3.  ip dhcp snooping
4.  no ip dhcp snooping information option
5.  ip dhcp snooping vlan 1
6.  ip arp inspection vlan 1
7.  int gi0/0
8.  ip dhcp snooping trust
9.  ip arp inspection trust
10. int range gi0/1 - 3
11. ip dhcp snooping limit rate 6
12. exit
13. ip arp inspection validate src-mac dst-mac ip
```

Root Bridge Spoofing:

```
#VPCS1
ip 10.0.0.1 255.255.255.0

#VPCS2
ip 10.0.0.2 255.255.255.0

#Attacker
python3 rbmitm.py eth0 eth1

Countermeasures

#SW1
conf t
int Gi0/0
spanning-tree bpduguard enable
end
wr

#SW2
conf t
int Gi0/2
spanning-tree bpduguard enable
end
wr

```

DNS Spoofing:

```
1.  #Webterm
2.  ifconfig eth0 20.0.0.1
3.  ifcongif add default gw 20.0.0.254
4.
5.  #Fake Web server
6.  ifconfig eth0 20.0.0.2
7.  ifcongif add default gw 20.0.0.254
8.
9.  #Attacker
10. ifconfig eth0 20.0.0.10
11. ifcongif add default gw 20.0.0.254
12. service nginx start
13. python3 dnsspoof.py eth0 binding.txt 20.0.0.0/24
14.
15. #R1
16. conf t
17. int f0/0
18. ip add 20.0.0.254 255.255.255.0
19. no shut
20. exit
21. int f1/0
22. ip dhcp
23. end
24. wr
25.
```

RIP Poison

- **First topology:**

```
1.  R1:
2.
3.  conf t
4.  int f0/0
5.  ip add 10.0.0.1 255.255.255.0
6.  no shut
7.  exit
8.  int f0/1
9.  ip add 20.0.0.1 255.255.255.0
10. no shut
11. exit
12. router rip
13. version 2
14. network 10.0.0.0
15. network 20.0.0.0
16. no auto-summary
17. end
18. wr
19.
20. R2
21. conf t
22. int f0/0
23. ip add 20.0.0.2 255.255.255.0
24. no shut
25. exit
26. int f0/1
27. ip add 30.0.0.2 255.255.255.0
28. no shut
29. exit
30. int f1/0
31. ip add 40.0.0.2 255.255.255.0
32. no shut
33. exit
34. router rip
35. version 2
36. network 20.0.0.0
37. network 30.0.0.0
38. network 40.0.0.0
39. no auto-summary
40. end
41. wr
42.
43. R3:
44. conf t
45. int f0/0
46. ip add 30.0.0.3 255.255.255.0
47. no shut
48. int f0/1
49. ip add 50.0.0.3 255.255.255.0
50. no shut
51. exit
52. router rip
53. version 2
54. network 30.0.0.0
55. network 50.0.0.0
56. no auto-summary
57. end
```

```
wr

Webterm:
ifconfig eth0 10.0.0.2
route add default gw 10.0.0.1 eth0

webserver:
ifconfig eth0 40.0.0.1
route add default gw 40.0.0.2 eth0

DockerAttacker
ifconfig eth0 50.0.0.1
route add default gw 50.0.0.3 eth0
ip addr
ifconfig eth0:0 40.0.0.1 netmask 255.255.255.0
ip addr
pyhton3 rippoison.py 40.0.0.1 255.255.255.255 eth0

```

- **Countermeasures:**

```

R1:
conf t
key chain saar
key 1
key-string lab1
int f0/0
ip rip authentication key-chain saar
ip rip authentication mode md5
exit
int f0/1
ip rip authentication key-chain saar
ip rip authentication mode md5
end
wr

R2:
conf t
key chain saar
key 1
key-string lab1
int f0/0
ip rip authentication key-chain saar
ip rip authentication mode md5
 exit
 int f0/1
 ip rip authentication key-chain saar
 ip rip authentication mode md5
 exit
 int f1/0
 ip rip authentication key-chain saar
 ip rip authentication mode md5
 end
 wr

 R3
 conf t
 key chain saar
 key 1
 key-string lab1
 int f0/0
 ip rip authentication key-chain saar
```

```
ip rip authentication mode md5
exit
inf f0/1
ip rip authentication key-chain saar
ip rip authentication mode md5
end
wr
```

- **Second topology:**

```
R1
conf t
int f0/0
ip add 10.0.0.1 255.255.255.0
no shut
exit
int f0/1
ip add 20.0.0.1 255.255.255.0
no shut
exit
int f1/0
ip add 40.0.0.1 255.255.255.0
no shut
exit
router rip
version 2
network 10.0.0.0
network 20.0.0.0
network 40.0.0.0
end
wr

R2
conf t
int f0/0
ip add 40.0.0.2 255.255.255.0
no shut
exit
int f0/1
ip add 50.0.0.2 255.255.255.0
no shut
exit
int f1/0
ip add 70.0.0.2 255.255.255.0
no shut
exit
router rip
version 2
network 40.0.0.0
network 50.0.0.0
network 70.0.0.0
end
wr

R3:
conf t
int f0/0
ip add 30.0.0.3 255.255.255.0
no shut
exit
```

```
int f0/1
ip add 60.0.0.3 255.255.255.0
no shut
exit
int f1/0
ip add 50.0.0.3 255.255.255.0
no shut
exit
router rip
version 2
network 30.0.0.0
network 60.0.0.0
network 50.0.0.0
end
wr

R4:
conf t
int f0/0
ip add 20.0.0.4 255.255.255.0
no shut
exit
int f0/1
ip add 30.0.0.4 255.255.255.0
no shut
exit
router rip
version 2
network 20.0.0.0
network 30.0.0.0
end
wr

Webserver:
ifconfig eth0 10.0.0.2
route add default gw 10.0.0.1 eth0

webterm:
ifconfig eth0 70.0.0.1
route add default gw 70.0.0.2 eth0

DockerAttacker
ifconfig eth0 60.0.0.1
route add default gw 60.0.0.3 eth0

```

DNS spoofing using DHCP spoofing

```
#R1
conf t
int f0/0
ip add 20.0.0.2 255.255.255.0
no shut
exit
int f1/0
ip add 22.0.0.2 255.255.255.0
no shut
exit
int f0/1
ip add 21.0.0.2 255.255.255.0
no shut
end
wr

#FakeDNSServer
conf t
no ip routing
ip dns server
ip domain-lookup
ip host www.linkedin.com 22.0.0.1
ip host linkedin.com 22.0.0.1

#Attacker
python3 dhcpspoof.py eth0 20.0.0.4 20.0.0.10 255.255.255.0 20.0.0.2 21.0.0.1

#FakeWebServer
service nginx start

```

Classical firewalls versus Zone Based Policy Firewalls:

- **Server:**

```
1.  enable
2.  conf t
3.  no ip routing
4.  int f0/0
5.  ip address 222.10.10.1 255.255.255.0
6.  no shutdown
7.  end
8.  conf t
9.  ip http server
10. service tcp-small-servers
```

- **Firewall Configuration:**

```
1.  enable
2.  conf t
3.  int f0/0
4.  ip address 222.10.10.254 255.255.255.0
5.  no shutdown
6.  int f0/1
7.  ip address 111.10.10.254 255.255.255.0
8.  no shutdown
9.  end
10. show ip interface brief
```

- **CBAC Firewall Configuration:**

```
1.  #Define inspect rules and apply to interface
2.  conf t
3.  ip inspect name PROTOCOLS icmp
4.  ip inspect name PROTOCOLS http
5.  int f0/1
6.  ip inspect PROTOCOLS in
7.  exit
8.
9.  #Define ACL and apply to interface from outside to server (http and icmp only)
10. conf t
11. ip access-list extended TO_SERVER
12. permit tcp any host 222.10.10.1 eq www
13. permit icmp any host 222.10.10.1
14. int f0/1
15. ip access-group TO_SERVER in
16. exit
17.
18. #Delete "all ports" entry
19. conf t
20. ip access-list extended TO_SERVER
21. no permit ip any host 222.10.10.1
22. exit
23.
24. #Define ACL and apply to interface from inside to outside
25. conf t
26. ip access-list extended TO_OUTSIDE
27. deny ip any any
```

```
int f0/0
ip access-group TO_OUTSIDE in
end

show access-lists
show ip inspect interfaces
```

- **ZBPF Firewall Configuration:**

```
#Define security zones
conf t
zone security INSIDE
zone security OUTSIDE
exit

#Assign zones to interfaces
int f0/0
zone-member security INSIDE
int f0/1
zone-member security OUTSIDE
end

#Define ACL
conf t
ip access-list extended TO_SERVER
permit ip any host 222.10.10.1
exit

#Define class maps
conf t
class-map type inspect match-any PROTOCOLS
match protocol icmp
match protocol http
class-map type inspect match-all CLASSMAP
match access-group name TO_SERVER
match class-map PROTOCOLS
exit

#Define policy maps
conf t
policy-map type inspect POLICYMAP
class type inspect CLASSMAP
inspect
class class-default
drop log
end

#Pair the zones
conf t
zone-pair security OUT_TO_IN source OUTSIDE destination INSIDE
service-policy type inspect POLICYMAP
end
```

- **No access to firewall interfaces configuration:**

```
#Define class maps
conf t
class-map type inspect match-any TO_FW_CLASSMAP

#Define policy maps
```

```
conf t
policy-map type inspect TO_FW_POLICYMAP
class type inspect TO_FW_CLASSMAP
pass
class class-default
drop log

#Apply policy to zone pairs
conf t
zone-pair security IN_TO_SELF source INSIDE destination self
service-policy type inspect TO_FW_POLICYMAP
zone-pair security OUT_TO_SELF source OUTSIDE destination self
service-policy type inspect TO_FW_POLICYMAP
end
```

- **Attacker-Out:**

```
nmap -sn 222.10.10.0/24
nmap -np0-1023 222.10.10.1
nmap -np0-1023 222.10.10.254
nmap -np0-1023 111.10.10.254
```

Protecting a campus network using a ZBPF:

- **Firewall Configuration:**

```
### Configuration ###
enable
conf t
int f2/0
ip address 10.0.0.254 255.128.0.0
ip nat inside
no shut

int f0/0
ip address 20.0.0.254 255.255.255.0
ip nat inside
no shut

int f1/1
ip address 10.128.0.254 255.128.0.0
ip nat inside
no shut

int f1/0
ip address dhcp
ip nat outside
no shutdown
exit
ip nat inside source list 1 interface FastEthernet1/0 overload

#To Test connectivity
conf t
access-list 1 permit 10.0.0.0 0.255.255.255
end

show ip interface brief
show ip route
show access-lists


## SSH Configuration ##
conf t
ip domain-name campus.com
crypto key generate rsa general-keys modulus 1024
ip ssh version 2
username thunder secret thunder
line vty 0 4
login local
transport input ssh
enable password thunder
end


######################### FIREWALL CONFIGURATION ############################
## ZBPF ZONE CONFIGURATION ##
conf t
zone security PRIVATE-ZONE-1
zone security PRIVATE-ZONE-2
zone security OUTSIDE
zone security DMZ
end

## DEFINITION OF ZONES ##
```

```
conf t
int f2/0
zone-member security PRIVATE-ZONE-1
exit
int f1/1
zone-member security PRIVATE-ZONE-2
exit
int f0/0
zone-member security DMZ
exit
int f1/0
zone-member security OUTSIDE
end

show zone security


## Definition of ACLs ##
conf t
ip access-list extended TO-WEB-SERVER
permit ip any host 20.0.0.1
exit
ip access-list extended TO-DNS-SERVER
permit ip any host 20.0.0.2
exit
ip access-list extended TO-SMTP-SERVER
permit ip any host 20.0.0.3
end

show access-lists


## ZBPF CLASS-MAPS PROTOCOLS CONFIGURATION ##
conf t
class-map type inspect match-any ACCESS-TO-FW-CLASSMAP
match protocol ssh
exit

class-map type inspect match-any PRIVATE-TO-OUTSIDE-CLASSMAP
match protocol icmp
match protocol http
 match protocol https
 match protocol dns
 exit

 class-map type inspect match-any DMZ-TO-OUTSIDE-CLASSMAP
 match protocol icmp
 match protocol smtp
 match protocol dns
 exitw

 class-map type inspect match-any TO-DMZ-SMTP
 match protocol icmp
 match protocol smtp
 exit

 class-map type inspect match-any TO-DMZ-WEB
 match protocol icmp
 match protocol http
 match protocol https
 exit

 class-map type inspect match-any TO-DMZ-DNS
 match protocol icmp
 match protocol dns
```

```
124.  exit
125.
126.  class-map type inspect match-any TO-DMZ-EMAIL
127.  match protocol icmp
128.  match protocol smtp
129.  match protocol pop3
130.  match protocol imap
131.  exit
132.
133.
134.  ## ZBPF CLASS-MAPS MATCH-ALL SERVERS CONFIGURATION ##
135.  class-map type inspect match-all TO-WEB-SERVER-CLASSMAP
136.  match access-group name TO-WEB-SERVER
137.  match class-map TO-DMZ-WEB
138.  exit
139.
140.  class-map type inspect match-all TO-EMAIL-SERVER-CLASSMAP
141.  match access-group name TO-SMTP-SERVER
142.  match class-map TO-DMZ-EMAIL
143.  exit
144.
145.  class-map type inspect match-all TO-SMTP-SERVER-CLASSMAP
146.  match access-group name TO-SMTP-SERVER
147.  match class-map TO-DMZ-SMTP
148.  exit
149.
150.  class-map type inspect match-all TO-DNS-SERVER-CLASSMAP
151.  match access-group name TO-DNS-SERVER
152.  match class-map TO-DMZ-DNS
153.  exit
154.
155.
156.  ## ZBPF CLASS-MAPS CONFIGURATION ##
157.  class-map type inspect match-any PRIVATE-TO-DMZ-CLASSMAP
158.  match class-map TO-WEB-SERVER-CLASSMAP
159.  match class-map TO-EMAIL-SERVER-CLASSMAP
160.  match class-map TO-DNS-SERVER-CLASSMAP
161.
162.  class-map type inspect match-any OUTSIDE-TO-DMZ-CLASSMAP
163.  match class-map TO-WEB-SERVER-CLASSMAP
164.  match class-map TO-SMTP-SERVER-CLASSMAP
165.  match class-map TO-DNS-SERVER-CLASSMAP
166.  end
167.
168.  show class-map type inspect
169.
170.
171.  ## ZBPF POLICY MAPS CONFIGURATION ##
172.  conf t
173.  policy-map type inspect ACCESS-TO-FW-POLICY
174.  class type inspect ACCESS-TO-FW-CLASSMAP
175.  pass
176.  class class-default
177.  drop log
178.  exit
179.  exit
180.
181.  policy-map type inspect PRIVATE-TO-OUTSIDE-POLICY
182.  class type inspect PRIVATE-TO-OUTSIDE-CLASSMAP
183.  inspect
184.  class class-default
185.  drop log
186.  exit
187.  exit
188.
```

```
189.  policy-map type inspect PRIVATE-TO-DMZ-POLICY
190.  class type inspect PRIVATE-TO-DMZ-CLASSMAP
191.  inspect
192.  class class-default
193.  drop log
194.  exit
195.  exit
196.
197.  policy-map type inspect DMZ-TO-OUTSIDE-POLICY
198.  class type inspect DMZ-TO-OUTSIDE-CLASSMAP
199.  inspect
200.  class class-default
201.  drop log
202.  exit
203.  exit
204.
205.  policy-map type inspect OUTSIDE-TO-DMZ-POLICY
206.  class type inspect OUTSIDE-TO-DMZ-CLASSMAP
207.  inspect
208.  class class-default
209.  drop log
210.  end
211.
212.  show policy-map type inspect
213.
214.
215.
216.  ## ZONE-PAIRS TO POLICY-MAPS ##
217.  conf t
218.  zone-pair security PR1-TO-FW source PRIVATE-ZONE-1 destination self
219.  service-policy type inspect ACCESS-TO-FW-POLICY
220.  exit
221.
222.  zone-pair security PR2-TO-FW source PRIVATE-ZONE-2 destination self
223.  service-policy type inspect ACCESS-TO-FW-POLICY
224.  exit
225.
226.  #provides errors in DHCP
227.  zone-pair security OUT-TO-FW source OUTSIDE destination self
228.  service-policy type inspect ACCESS-TO-FW-POLICY
229.  exit
230.
231.  zone-pair security PR1-TO-OUT source PRIVATE-ZONE-1 destination OUTSIDE
232.  service-policy type inspect PRIVATE-TO-OUTSIDE-POLICY
233.  exit
234.
235.  zone-pair security PR1-TO-DMZ source PRIVATE-ZONE-1 destination DMZ
236.  service-policy type inspect PRIVATE-TO-DMZ-POLICY
237.  exit
238.
239.  zone-pair security PR2-TO-OUT source PRIVATE-ZONE-2 destination OUTSIDE
240.  service-policy type inspect PRIVATE-TO-OUTSIDE-POLICY
241.  exit
242.
243.  zone-pair security PR2-TO-DMZ source PRIVATE-ZONE-2 destination DMZ
244.  service-policy type inspect PRIVATE-TO-DMZ-POLICY
245.  exit
246.
247.  zone-pair security DMZ-TO-OUT source DMZ destination OUTSIDE
248.  service-policy type inspect DMZ-TO-OUTSIDE-POLICY
249.  exit
250.
251.  zone-pair security OUT-TO-DMZ source OUTSIDE destination DMZ
252.  service-policy type inspect OUTSIDE-TO-DMZ-POLICY
253.  end
```

```
254.
255.  show zone-pair security
```

- **Other Machines Configuration:**

```
256.  ##############################################################################
257.  SMTP-Server:
258.  sudo ifconfig ens3 20.0.0.3/24
259.  sudo route add default gw 20.0.0.254
260.
261.  sudo nano /etc/resolv.conf
262.  nameserver 20.0.0.2
263.  ip route | grep default
264.
265.
266.  DNS-Server:
267.  # DNS Configuration ##
268.  nano /etc/hosts
269.  20.0.0.1 webserver.lab web1
270.  20.0.0.3 smtpserver.lab smtp1
271.  service dnsmasq restart
272.
273.
274.  Web-Server:
275.  enable
276.  conf t
277.  int f0/0
278.  ip address 20.0.0.1 255.255.255.0
279.  no shut
280.  exit
281.
282.  no ip routing
283.  ip default-gateway 20.0.0.254
284.  ip http server
285.  ip http secure-server
286.  ip domain lookup
287.  ip name-server 20.0.0.2
288.  end
289.
290.  WebBrowser:
291.  ifconfig eth0 10.0.0.2/24
292.  route add default gw 10.0.0.254
293.
294.  Attacker1:
295.  nmap -sn 20.0.0.0/24
296.  nmap -np0-1023 20.0.0.1
297.
298.  #For ssh
299.  ~/.ssh/config
300.  Host 10.0.0.254
301.  KexAlgorithms +diffie-hellman-group1-sha1
302.  Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc
303.
```

Defence against DoS attacks

- **Server:**

```
enable
conf t
no ip routing
int f0/0
ip address 222.10.10.1 255.255.255.0
no shutdown
end
conf t
ip http server
service tcp-small-servers
show TCP statistics
```

- **Firewall:**

```
enable
conf t
int f0/0
ip address 222.10.10.254 255.255.255.0
no shutdown
int f0/1
ip address 111.10.10.254 255.255.255.0
no shutdown
end
show ip interface brief

              #### ZBPF CONFIGURATION ####
#Define security zones
conf t
zone security INSIDE
zone security OUTSIDE
exit

#Assign zones to interfaces
int f0/0
zone-member security INSIDE
int f0/1
zone-member security OUTSIDE
end

#Access control lists
ip access-list extended TO_IN_HOST
permit ip any host 222.10.10.1

#Class maps
class-map type inspect match-all ICMP_CLASSMAP
match access-group name TO_IN_HOST
match protocol icmp

class-map type inspect match-all WWW_CLASSMAP
match access-group name TO_IN_HOST
match protocol http

#Policy maps
policy-map type inspect DOS_POLICYMAP
class type inspect WWW_CLASSMAP
inspect
```

```
class type inspect ICMP_CLASSMAP
inspect
class class-default
drop log


#Pair the zones
conf t
zone-pair security OUT_TO_IN source OUTSIDE destination INSIDE
service-policy type inspect DOS_POLICYMAP
end

## Policy-Map countermeasures ##

#Policy maps
conf t
policy-map type inspect DOS_POLICYMAP
class type inspect WWW_CLASSMAP
inspect
police rate 8000 burst 1000
class type inspect ICMP_CLASSMAP
inspect
police rate 8000 burst 1000
class class-default
drop log
end

## Parameter Map ##
conf t
parameter-map type inspect PARMAP
tcp synwait-time 3
end

conf t
policy-map type inspect DOS_POLICYMAP
class type inspect WWW_CLASSMAP
inspect PARMAP
police rate 128000 burst 8000
class type inspect ICMP_CLASSMAP
inspect
police rate 128000 burst 8000
class class-default
drop log

show policy-map type inspect zone-pair OUT_TO_IN

```

- **Attacker-OUT:**

```
hping3 -i u10000 -S -p 80 222.10.10.1
hping3 -i u10000 -S -p 80 --rand-source 222.10.10.1

hping3 -1 --flood 222.10.10.1
hping3 -S -p 80 --flood --rand-source 222.10.10.1

hping3 -S -p 80 --flood --rand-source 222.10.10.1
```

TACACAS+

```
#R1
conf t
int f0/0
ip add 222.10.10.254
no shut
exit
int f0/1
ip add 10.0.0.254
no shut
exit
username admin privilege 15 password 0 admin
enable password cisco
aaa new-model
tacacs server Server-T
address ipv4 10.0.0.1
key gns3
exit
aaa authentication login default group tacacs+ local
aaa authorization exec default group tacacs+
aaa authorization commands 15 default group tacacs+
aaa accounting exec default start-stop group tacacs+
end
wr

#Client
ifconfig eth0 222.10.10.1
ifconfig add default gw 222.10.10.254 eth0

#TACACAS+ Server
ifconfig eth0 222.10.10.1
ifconfig add default gw 1.0.0.254 eth0

```

802.IX

```
#PC1
ip 10.1.1.2 255.255.255.0

#Supplicant
ifconfig eth0 10.1.1.1

#AAA Server
ifconfig eth0 10.1.1.50

#SW
aaa new-model
radius server AAAServer
address ipv4 10.1.1.50 auth-port 1812 acct-port 1813
key gns3
exit
aaa authentication dot1x default group radius
dot1x system-auth-control
int gi0/0
switchport mode access
authentication port-control auto
dot1x pae authenticator
int gi0/2
switchport mode access
authentication port-control auto
dot1x pae authenticator
int vlan 1
ip add 10.1.1.100 255.255.255.0
no shutdown
end
wr
```