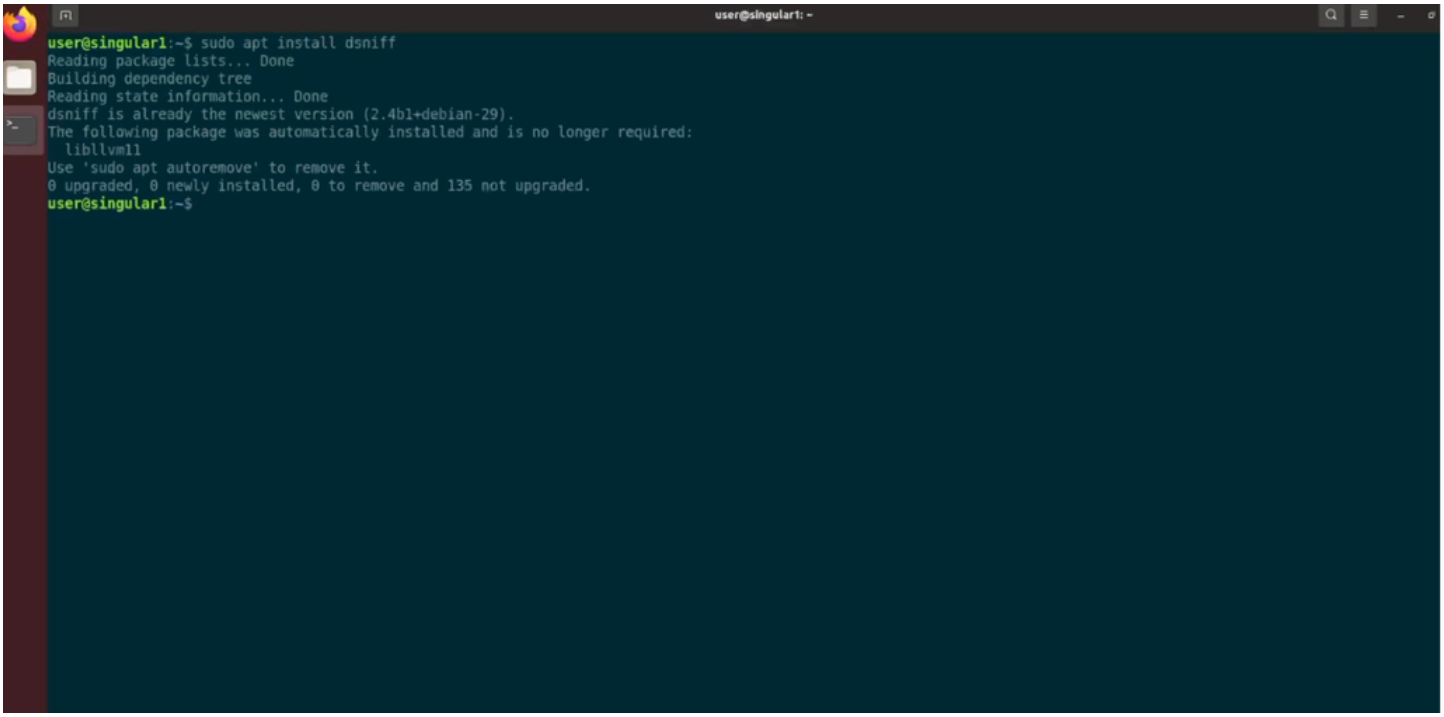
Empezamos con ARP Spoofing, cómo mitigamos dichos ataques?

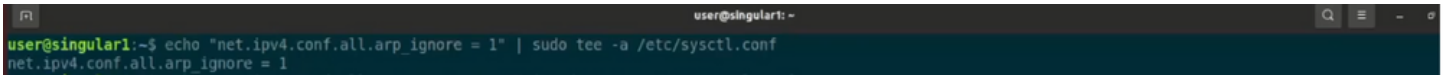
Hay muchísimas formas de protegerse del arp spoofing, pero nos vamos a centrarnos en una solución común y fácil.

Instalación de las herramientas de seguridad. DSNIFF para detectar los intentos de arp spoofing en la red.

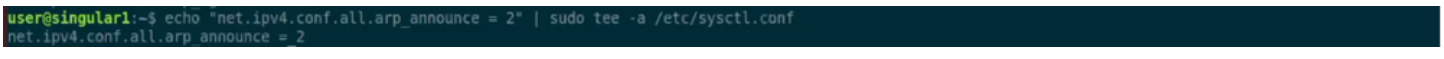


Configuración directa al kernel para mayor resistencia, y ajustar cómo se anuncia el arp en la red. Le diremos al kernel que no haga caso a no ser que tenga una cierta configuración, y ahora hacemos los comandos con dicha configuración.

Le decimos al sistema operativo que ignore los paquetes arp que previenen de una interfaz en la cual la dirección ip de destino no coincide con la dirección ip de ninguna interfaz local. Lo que ayuda a prevenir ataques donde un atacante envia paquetes arp falsos para asociar su propia dirección mac a direcciones ip que no están asociadas con la interfaz de red del sistema:



Le decimos al sistema que anuncie su propia dirección ip cómo la fuente de todas las respuestas arp que envia, para mitigar los ataques si un agente malicioso intenta envenenar las tablas arp de los dispositivos de la red, ya que los dispositivos legitimos puedes recibir respuestas arp genuinas del sistema configurado con ésta opción:



Reiniciamos el servicio con las nuevas configuraciones:

Después de solucionar el ataque hay que monitorizar y hacer un seguimiento, para hacerlo usaremos ARP WATCH para monitorizar los cambios de las asignaciones de ip a direcciones mac lo que nos permite detectar actividades sospechosas en la red. Instalamos arpwatch:

```
user@singular1:~$ sudo apt install arpwatch
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm11
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  gawk ieee-data libsigsegv2
Suggested packages:
  gawk-doc
The following NEW packages will be installed:
  arpwatch gawk ieee-data libsigsegv2
0 upgraded, 4 newly installed, 0 to remove and 135 not upgraded.
Need to get 2,026 kB of archives.
After this operation, 12.2 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Lanzamos arpwatch para monitorizar los cambios en la dirección mac:

```
user@singular1:~$ sudo arpwatch -i eth0
user@singular1:~$
```

Para ver los cambios que van ocurriendo hacemos el siguiente comando. Si vemos un mensaje cómo éste de abajo “no such file or directory” para el arp.dat, es importante crearlo porque en este fichero se lleva el registro de las direcciones ip y mac que son conocidas por al red, sin éste fichero puede que no funciona arpwatch:

```
user@singular1:~$ sudo grep arpwatch /var/log/syslog
Mar  1 11:27:49 ubuntu systemd[1]: Starting arpwatch service...
Mar  1 11:27:49 ubuntu systemd[1]: Finished arpwatch service.
Mar  1 11:28:06 ubuntu arpwatch: listening on eth0
Mar  1 11:28:06 ubuntu arpwatch: fopen(arp.dat): No such file or directory
Mar  1 11:32:00 ubuntu arpwatch: listening on eth0
Mar  1 11:32:00 ubuntu arpwatch: fopen(arp.dat): No such file or directory
user@singular1:~$
```

vamos a crearlo nosotros mismos:

```
user@singular1:~$ sudo touch /var/lib/arpwatch/arp.dat
user@singular1:~$
```

le ponemos bien los permisos:

```
user@singular1:~$ sudo chown arpwatch:arpwatch /var/lib/arpwatch/arp.dat
user@singular1:~$
```

reiniciamos arpwatch para aplicar los cambios:

```
user@singular1:~$ sudo systemctl restart arpwatch
user@singular1:~$
```

Volvemos a lanzar el comando para ver los cambios usando arpwatch, ya no me dice el error de no such file or directory:

```
user@singular1:~$ sudo grep arpwatch /var/log/syslog
Mar  1 11:27:49 ubuntu systemd[1]: Starting arpwatch service...
Mar  1 11:27:49 ubuntu systemd[1]: Finished arpwatch service.
Mar  1 11:28:06 ubuntu arpwatch: listening on eth0
Mar  1 11:28:06 ubuntu arpwatch: fopen(arp.dat): No such file or directory
Mar  1 11:32:00 ubuntu arpwatch: listening on eth0
Mar  1 11:32:00 ubuntu arpwatch: fopen(arp.dat): No such file or directory
Mar  1 11:35:51 ubuntu systemd[1]: arpwatch.service: Succeeded.
Mar  1 11:35:51 ubuntu systemd[1]: Stopped arpwatch service.
Mar  1 11:35:51 ubuntu systemd[1]: Stopping arpwatch service...
Mar  1 11:35:51 ubuntu systemd[1]: Starting arpwatch service...
Mar  1 11:35:51 ubuntu systemd[1]: Finished arpwatch service.
user@singular1:~$
```

Volvemos a lanzar arpwatch en nuestra interfaz de red adecuada con el fichero arp.dat creado:

```
user@singular1:~$ sudo arpwatch -i eth0
user@singular1:~$
```

Para ver cómo se reflejan los cambios vamos a la otra máquina y hacemos cambios a nuestra dirección mac con los siguientes comandos para ver cómo se reflejan en arpwatch y sus logs:

```
user@singular2:~$ sudo ip link set dev eth0 down
[sudo] password for user:
user@singular2:~$ sudo ip link set dev eth0 address 02:01:02:03:04:08
user@singular2:~$ sudo ip link set dev eth0 up
user@singular2:~$ sudo ip link set dev eth0 down
user@singular2:~$ sudo ip link set dev eth0 address 02:01:02:03:04:FF
user@singular2:~$ sudo ip link set dev eth0 up
user@singular2:~$ sudo ip link set dev eth0 down
user@singular2:~$ sudo ip link set dev eth0 address 02:01:02:03:04:04
user@singular2:~$ sudo ip link set dev eth0 down
user@singular2:~$ sudo ip link set dev eth0 address 02:01:02:03:04:04
user@singular2:~$ sudo ip link set dev eth0 up
user@singular2:~$ exit
```

Aqui estan los logs tras hacer los cambios de mac, arpwatch nos va diciendo que hay nuevas mac, y se puede concatenar para que realice avisos a los analistas de que hay un ataque de arp spoofing con cambios de mac:

```
user@singular1:~$ sudo grep arpwatch /var/log/syslog
Mar  1 11:27:49 ubuntu systemd[1]: Starting arpwatch service...
Mar  1 11:27:49 ubuntu systemd[1]: Finished arpwatch service.
Mar  1 11:28:06 ubuntu arpwatch: listening on eth0
Mar  1 11:28:06 ubuntu arpwatch: fopen(arp.dat): No such file or directory
Mar  1 11:32:00 ubuntu arpwatch: listening on eth0
Mar  1 11:32:00 ubuntu arpwatch: fopen(arp.dat): No such file or directory
Mar  1 11:35:51 ubuntu systemd[1]: arpwatch.service: Succeeded.
Mar  1 11:35:51 ubuntu systemd[1]: Stopped arpwatch service.
Mar  1 11:35:51 ubuntu systemd[1]: Stopping arpwatch service...
Mar  1 11:35:51 ubuntu systemd[1]: Starting arpwatch service...
Mar  1 11:35:51 ubuntu systemd[1]: Finished arpwatch service.
Mar  1 11:36:14 ubuntu arpwatch: listening on eth0
Mar  1 11:36:47 ubuntu arpwatch: new station 10.211.55.17 02:01:02:03:04:04 eth0
Mar  1 11:36:47 ubuntu arpwatch: execl: /usr/lib/sendmail: No such file or directory
Mar  1 11:36:47 ubuntu arpwatch: reaper: pid 129661, exit status 1
Mar  1 11:36:53 ubuntu arpwatch: new station 10.211.55.5 00:1c:42:d0:02:5f eth0
Mar  1 11:36:53 ubuntu arpwatch: execl: /usr/lib/sendmail: No such file or directory
Mar  1 11:36:53 ubuntu arpwatch: reaper: pid 129680, exit status 1
Mar  1 11:36:53 ubuntu arpwatch: new station 10.211.55.1 00:1c:42:00:00:18 eth0
Mar  1 11:36:53 ubuntu arpwatch: execl: /usr/lib/sendmail: No such file or directory
Mar  1 11:36:53 ubuntu arpwatch: reaper: pid 129681, exit status 1
user@singular1:~$
```

Ahora empezamos con el ataque ping flood, cómo mitigarlo?

Con iptables, hacemos un par de reglas para mitigar el ataque y ponerle filtros.

Regla que se agrega al input y dice que acepta el paquete si cumple las siguientes condiciones: para mensajes ICMP del tipo echorequest (ping), establecemos un límite de velocidad en 1 paquete por segundo para paquetes ICMP:

```
user@singular1:~$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
```

Siguiente regla que se agrega al input también: Si encuentra un paquete icmp del typo echo request directamente se descarta sin responder:

```
user@singular1:~$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
user@singular1:~$
```

Continuamos con iptables y establecemos el syn proxy con tres reglas distintas:

```
user@singular1:~$ sudo iptables -t raw -A PREROUTING -p tcp -m tcp --syn -j CT --notrack
user@singular1:~$ sudo iptables -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -j SYNPROXY --sack-perm --timestamp --wscale 7 --mss 1460
user@singular1:~$ sudo iptables -A INPUT -m state --state INVALID -j DROP
user@singular1:~$
```

En el transcript hay toda la explicación de éstas reglas de iptables.

Cómo evitar o mitigar el ataque DHCP Starvation?

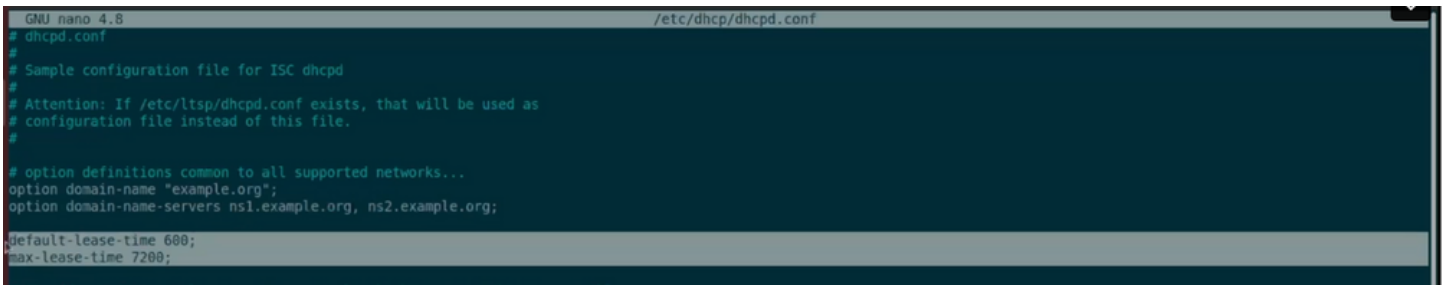
Si tuviesemos hardware de Cisco podríamos activar DHCP Snooping en la terminal de Cisco IOS usando el siguiente comando (no es válido en la terminal de ubuntu), nos faltaría simplemente especificar los parámetros extra (vlan, etc).

IMPORTANTE: para mayor seguridad deberíamos implementar las soluciones y protecciones contra ataques asociados a DHCP desde la red, no los servidores, pero aqui veremos unas pautas a seguir directamente desde el servidor.

Editar el fichero.conf:

Reserva IP: Para equipos críticos y muy importantes podríamos añadir información en el fichero.conf sobre éste host conocido y reservar ésa dirección ip para ése dispositivo. Ponemos la dirección mac del dispositivo y le reservamos una ip para dicho dispositivo, así siempre tendrá un ip reservada desde que se levanta el servidor.

Otra manera, sería volviendo a editar el fichero.conf, podríamos limitar las peticiones dhcp, el lease-time, que poe defecto están arriba:



También se puedes escoger distintas limitaciones de peticiones dhcp según la subnet que queramos, cómo vemos abajo en ésta subnet hemos puesto un lease-time que podríamos poner distinto:

Monitorizar los eventos relacionados con el dhcp:

```
user@ubuntu:~$ sudo tail -f /var/log/syslog | grep dhcp
```

Ahora iría registrando en tiempo real cualquier entrada que estuviese relacionada con dhcp, a medida que entre algo de dhcp aparecerá abajo. Hemos lanzado cómo ejemplo algunas peticiones dhcp y vemos que sólo se han asignado algunas direcciones ip y luego ya no se asigna ninguna:

```
user@ubuntu:~$ sudo tail -f /var/log/syslog | grep dhcp
Mar  2 12:31:02 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:63:7b:99 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:02 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:31:c2:94 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:02 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:6d:36:1e via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:02 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:5b:65:97 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:02 ubuntu dhcpd[122888]: DHCPOFFER on 10.211.55.104 to 00:16:3e:3e:11:2e via eth0
Mar  2 12:31:02 ubuntu dhcpd[122888]: DHCPOFFER on 10.211.55.103 to 00:16:3e:13:b0:8e via eth0
Mar  2 12:31:02 ubuntu dhcpd[122888]: DHCPOFFER on 10.211.55.102 to 00:16:3e:6c:06:62 via eth0
Mar  2 12:31:02 ubuntu dhcpd[122888]: DHCPOFFER on 10.211.55.101 to 00:16:3e:44:7f:f3 via eth0
Mar  2 12:31:02 ubuntu dhcpd[122888]: DHCPOFFER on 10.211.55.100 to 00:16:3e:4d:ee:1b via eth0
Mar  2 12:31:02 ubuntu dhcpd[122888]: DHCPOFFER on 10.211.55.105 to 00:16:3e:79:bf:63 via eth0
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:35:fb:d5 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:2e:10:20 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:3d:86:aa via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:3b:e3:59 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:20:59:0b via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:0d:48:dc via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:55:85:2e via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:39:89:18 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:56:92:27 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:2f:72:ac via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:4c:cd:83 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:24:5e:70 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:22:6d:41 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:5c:5e:06 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:01:62:31 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:28:7b:dd via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:05:e5:a9 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:09:37:51 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:06:d6:24 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:07:f3:4e via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:2a:b4:48 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:55:10:ed via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:1c:bc:4c via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:17:d7:1f via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:26 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:6a:aa:0e via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:0e:d4:bb via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:22:89:54 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:09:d3:69 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:5a:ac:2d via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:50:bd:07 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:3c:fe:ad via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:64:11:26 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:05:10:24 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:41:32:3f via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:2b:46:c3 via eth0: network 10.211.55.0/24: no free leases
Mar  2 12:31:27 ubuntu dhcpd[122888]: DHCPDISCOVER from 00:16:3e:37:63:a2 via eth0: network 10.211.55.0/24: no free leases
```

Es una buena opción tener en una terminal aparte con los logs que van apareciendo para monitorizar las diferentes evoluciones asociadas al dhcp en tiempo real.

Cómo mitigar el dns spoofing?

Vamos a configurar dnssec para validar las respuestas dns y asegurarnos que la información que recibimos es autentica.

El dns security extension (dns sec) nos ayuda a proteger contra el dns spoofing validando las respuestas dns, con lo cual vamos a instalar el bind9 (paquete con la aplicación dns sec):

```
user@singular2:~$ sudo apt install bind9
[sudo] password for user:
```

Luego tenemos que usar éstos comandos para poder activarlo:

```
user@singular2:~$ sudo rndc-confgen -a
wrote key file "/etc/bind/rndc.key"
user@singular2:~$ sudo named-checkconf
user@singular2:~$ sudo systemctl restart bind9
user@singular2:~$
```

También utilizamos resolvers dns que soporten dns sec para agregar una capa adicional de seguridad en la resolución de nombres de dominio.

```
user@singular2:~$ echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf
```

Conclusions

Understanding how to mitigate attacks such as ARP Spoofing, Ping Flood, DHCP Starvation, and DNS Spoofing from a practical standpoint is crucial for maintaining the integrity, availability, and confidentiality of networked systems. These skills empower individuals and organizations to defend against evolving cyber threats, ensuring the security and trustworthiness of their digital environments.