Thea Traw

Person-in-the-middle via ARP spoofing

EXECUTION

1. What is Kali's main interface's MAC address? (The main interface is probably called eth0, but check ifconfig to be sure.)

```
ether 4e:27:a6:45:1c:d5
```

2. What is Kali's main interface's IP address?

```
inet 192.168.64.2
```

3. What is Metasploitable's main interface's MAC address?

```
HWaddr 76:0f:90:a3:73:e0
```

4. What is Metasploitable's main interface's IP address?

```
inet addr:192.168.64.3
```

5. Show Kali's routing table. (Use "`netstat -r`" to see it with symbolic names, or "`netstat -rn`" to see it with numerical addresses.)

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         192.168.64.1    0.0.0.0         UG        0 0          0 eth0
192.168.64.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

6. Show Kali's ARP cache. (Use "`arp`" or "`arp -n`".)

```
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.64.1             ether   3e:06:30:13:14:64   C                     eth0
192.168.64.3             ether   76:0f:90:a3:73:e0   C                     eth0
```

7. Show Metasploitable's routing table.

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.64.0    *               255.255.255.0   U         0 0          0 eth0
default         192.168.64.1    0.0.0.0         UG        0 0          0 eth0
```

8. Show Metasploitable's ARP cache.

```
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.64.1             ether   3E:06:30:13:14:64   C                     eth0
```

9. Suppose the user of Metasploitable wants to get the CS338 sandbox page via the command "`curl http://cs338.jeffondich.com/`". To which MAC address should Metasploitable send the TCP SYN packet to get the whole HTTP query started? Explain why.

We know that the IP address of http://cs338.jeffondich.com/ is 45.79.89.123. So we turn to the routing table to find how to get there. There are only two options in Metasploitable's routing table: 0.0.0.0 (default) or 192.168.64.0. Given the Genmask for the latter, only IP addresses that differ in the last octet–that is, of the form 129.168.64.*--would apply. So, that means that we fall to the default. (With its Genmask of 0.0.0.0, all other addresses that are not of the prior form must be dealt with here, including this one.) The gateway the packets are next sent to is then 192.168.64.1. When we look in Metaploitable's ARP cache, we see that the MAC address for that IP address is 3E:06:30:13:14:64. Thus, Metasploitable should send the TCP SYN packet there.

10. Fire up Wireshark on Kali. Start capturing packets for "tcp port http". On Metasploitable, execute "`curl http://cs338.jeffondich.com/`". On Kali, stop capturing. Do you see an HTTP response on Metasploitable? Do you see any captured packets in Wireshark on Kali?

No packets were captured by Wireshark on Kali. Metasploitable receives the html payload from the website (following the HTTP request for it) and displays it for the user.

11. Now, it's time to be Mal (who will, today, merely eavesdrop). Use Ettercap to do ARP spoofing (also known as ARP Cache Poisoning) with Metasploitable as your target. There are many online tutorials on how to do this (here's one). Find one you like, and start spoofing your target. NOTE: most of these tutorials are showing an old user interface for Ettercap, which may make them confusing. The steps you're trying to take within Ettercap are:
    - Start sniffing (*not* bridged sniffing) on eth0
    - Scan for Hosts
    - View the Hosts list
    - Select your Metasploit VM from the Host List
    - Add that host as Target 1
    - Start ARP Poisoning (including Sniff Remote Connections)
    - Do your stuff with wireshark and Metasploitable
    - Stop ARP Poisoning
12. I'll post some screenshots on Slack of how I got Ettercap to do these things. Honestly, I don't know who redesigned this user interface to make it so much harder to do things, but they did. (Common enough in the Linux UI world.)
    So, to wrap up this step: start the ARP poisoning. You will keep the ARP poisoning attack active until you are done with your AITM attack. (Realistically, you will probably start and stop ARP poisoning several times as you gradually figure out what's going on while doing the steps below.)
13. Show Metasploitable's ARP cache. How has it changed?

```
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.64.1             ether   4E:27:A6:45:1C:D5   C                     eth0
```

The MAC address for 192.168.64.1 is different! (Specifically, it is now Kali's main interface's MAC address.)

14. Without actually doing it yet, predict what will happen if you execute "`curl http://cs338.jeffondich.com/`" on Metasploitable now. Specifically, to what MAC address will Metasploitable send the TCP SYN packet? Explain why.

Now, Metasploitable's ARP cache has been poisoned. So after the routing table is checked, and 45.79.89.123 is directed to the gateway 192.168.64.1 (following the default), there is a perfectly normal looking MAC address ready to be used in the ARP cache. So Metasploitable will think 4E:27:A6:45:1C:D5 is the MAC address for 192.168.64.1 and will send the TCP SYN packet there. But alas, it was Mal (Kali) all along.

15. Start Wireshark capturing "tcp port http" again.
16. Execute "`curl http://cs338.jeffondich.com/`" on Metasploitable. On Kali, stop capturing. Do you see an HTTP response on Metasploitable? Do you see captured packets in Wireshark? Can you tell from Kali what messages went back and forth between Metasploitable and cs338.jeffondich.com?

Yes, we get the HTML response on Metasploitable, just as we did the first time (when there was no ARP poisoning). There are also now captured packets collected by Wireshark on Kali.

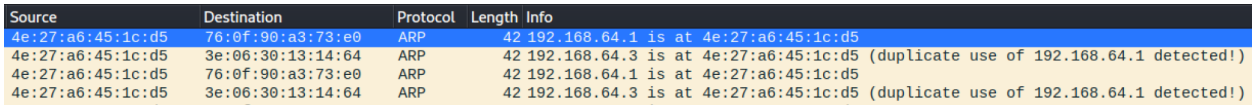
	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.64.3	45.79.89.123	TCP	74	45275 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM TS
2	0.007684630	192.168.64.3	45.79.89.123	TCP	74	[TCP Retransmission] 45275 → 80 [SYN] Seq=0 Win=5840 Len=0
3	0.059356601	45.79.89.123	192.168.64.3	TCP	66	80 → 45275 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1382
4	0.062742608	45.79.89.123	192.168.64.3	TCP	66	[TCP Retransmission] 80 → 45275 [SYN, ACK] Seq=0 Ack=1 Win=
5	0.063417643	192.168.64.3	45.79.89.123	TCP	54	45275 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0
6	0.063417684	192.168.64.3	45.79.89.123	HTTP	212	GET / HTTP/1.1
7	0.071769099	192.168.64.3	45.79.89.123	TCP	54	45275 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0
8	0.071807932	192.168.64.3	45.79.89.123	TCP	212	[TCP Retransmission] 45275 → 80 [PSH, ACK] Seq=1 Ack=1 Win=
9	0.123842983	45.79.89.123	192.168.64.3	TCP	54	80 → 45275 [ACK] Seq=1 Ack=159 Win=64128 Len=0
10	0.125788838	45.79.89.123	192.168.64.3	HTTP	785	HTTP/1.1 200 OK (text/html)
11	0.126122460	45.79.89.123	192.168.64.3	TCP	54	80 → 45275 [ACK] Seq=1 Ack=159 Win=64128 Len=0
12	0.126141501	45.79.89.123	192.168.64.3	TCP	785	[TCP Retransmission] 80 → 45275 [PSH, ACK] Seq=1 Ack=159 Wi
13	0.126754703	192.168.64.3	45.79.89.123	TCP	54	45275 → 80 [ACK] Seq=159 Ack=732 Win=7360 Len=0
14	0.127459113	192.168.64.3	45.79.89.123	TCP	54	45275 → 80 [FIN, ACK] Seq=159 Ack=732 Win=7360 Len=0
15	0.135927693	192.168.64.3	45.79.89.123	TCP	54	[TCP Keep-Alive] 45275 → 80 [ACK] Seq=159 Ack=732 Win=7360
16	0.135946609	192.168.64.3	45.79.89.123	TCP	54	[TCP Retransmission] 45275 → 80 [FIN, ACK] Seq=159 Ack=732
17	0.186377302	45.79.89.123	192.168.64.3	TCP	54	80 → 45275 [FIN, ACK] Seq=732 Ack=160 Win=64128 Len=0
18	0.186570508	45.79.89.123	192.168.64.3	TCP	54	[TCP Retransmission] 80 → 45275 [FIN, ACK] Seq=732 Ack=160
19	0.187132211	192.168.64.3	45.79.89.123	TCP	54	45275 → 80 [ACK] Seq=160 Ack=733 Win=7360 Len=0
20	0.194911423	192.168.64.3	45.79.89.123	TCP	54	[TCP Dup ACK 19#1] 45275 → 80 [ACK] Seq=160 Ack=733 Win=736

We can see the messages exchanged by Metasploitable and cs338.jeffondich.com. A TCP three-way handshake occurs first. (With some suspicious looking “retransmission” packets identified as well…) Then Metasploitable asks for the home page “/” with a GET request. cs338.jeffondich.com obliges with a 200 OK and sends the html payload. Then Metasploitable terminates the connection with a [FIN, ACK] and cs338.jeffondich.com agrees to do so.

17. Explain in detail what happened. How did Kali change Metasploitable's ARP cache? (If you want to watch the attack in action, try stopping the AITM attack by selecting "Stop mitm attack(s)" from Ettercap's Mitm menu, starting a Wireshark capture for "arp", and restarting the ARP poisoning attack in Ettercap.)

Kali poisoned Metasploitable’s ARP cache by effectively drowning out the legitimate MAC address with a bunch of ARP replies carrying its own MAC address. Before the attack, the actual router had let Metasploitable know that its MAC address is 3E:06:30:13:14:64. (Before that, Metasploitable would have sent out a ARP request packet to the whole network for the IP address 192.168.64.1, because it wouldn’t have yet had it in its cache. Or possibly, the router probably sends out its ARP reply

packet–saying “I am 192.168.64.1, here is my MAC address: 3E:06:30:13:14:64”–on a regular basis, like every minute or so. In any case, whether the ARP packet was given unprompted or not, Metasploitable had the correct MAC address for the IP address.) Now, however, Kali broadcasts a whole bunch of ARP replies to the network, just one after another after another in rapid succession.



Source	Destination	Protocol	Length	Info
4e:27:a6:45:1c:d5	76:0f:90:a3:73:e0	ARP	42	192.168.64.1 is at 4e:27:a6:45:1c:d5
4e:27:a6:45:1c:d5	3e:06:30:13:14:64	ARP	42	192.168.64.3 is at 4e:27:a6:45:1c:d5 (duplicate use of 192.168.64.1 detected!)
4e:27:a6:45:1c:d5	76:0f:90:a3:73:e0	ARP	42	192.168.64.1 is at 4e:27:a6:45:1c:d5
4e:27:a6:45:1c:d5	3e:06:30:13:14:64	ARP	42	192.168.64.3 is at 4e:27:a6:45:1c:d5 (duplicate use of 192.168.64.1 detected!)

Each of these packets contains the same IP address–192.168.64.1–but Kali’s MAC address–4E:27:A6:45:1C:D5. So, when Metasploitable receives this information, it updates its ARP cache and is poisoned. Metasploitable is unable to differentiate between the ARP packets from the actual router and from Kali, and so it just changes its ARP cache as it is informed to do so.

18. If you wanted to design an ARP spoofing detector, what would you have your detector do? (As you think about this, consider under what circumstances your detector might generate false positives.)

The key indicator of an attack would be bombardment of ARP replies, all linking the same IP address and MAC address. Since this attack only works when there is a super high rate of transmission (otherwise the router with its proper MAC address would re-update the cache as well on its standard schedule), this would be a pretty obvious sign. So an ARP spoofing detector should certainly raise a red flag if such an event occurs. And while this could be a false positive (like potentially some misbehaving software on the network, accidentally sending way too many of its standard issue ARP replies, but with no malintent), it is something you would want to be alerted to, either way, and a problem you would want to resolve. The ARP spoofing detector could also potentially keep track of the MAC addresses that have been associated with a specific IP address. Then, if there’s a deviation from the normal linkage, that could be cause for concern. This would require some (non-significant?) amount of memory to store the information, however, which might not be ideal.

SYNTHESIS

1. Explain in detail Mal's strategy for intercepting the traffic between Alice and Bob. Use any of your observations from the Execution section to clarify your explanation. But be careful not to just reiterate all the steps, and not to focus on

specific tools. (For example, I would not expect you to refer to Ettercap in this explanation, since it is merely one of many available tools for generating suitable ARP messages.) Your goal here is to explain to a technical audience (e.g., other CS majors who have not studied security) what Mal is up to, and how ARP cache poisoning works.

First, Mal needs to trick Alice into sending the packets to Mal. Then, Mal can intercept all of the traffic, read it, and then pass it along to Bob. In a Mal-less world, Alice will use her IP routing table and ARP cache to determine how to send the packets to Bob. From the routing table, she will find which gateway (a specific IP address) to use to reach Bob (also an IP address). Then, in her ARP cache, Alice will look up what MAC address is associated with that IP address. She can now wrap her packets in the proper headers and send them off. Mal's strategy is to poison Alice's ARP cache. The ARP cache either sends an ARP request (asking for the MAC address for a given IP address, if that IP address is not already present) or receives an ARP reply (which provides a MAC address for a given IP address, and can either be sent prompted or unprompted). The well-behaved software on the network will send an ARP reply every so often, on a regular schedule, like every minute or so. The ARP cache does not verify anything else about the ARP replies it receives; it simply saves and displays the information. So, Mal will take advantage of that, and will send a whole bunch of ARP replies that link the gateway IP to Mal's MAC address. With such a deluge of MAC replies, the actual MAC address will be drowned out. So, now when Alice wants to communicate with Bob, she will look up the gateway's IP address in her ARP cache. But wait! The MAC address is now Mal's, not the gateway's. But Alice does not know that. So, she sends her packets to that MAC address, and Mal can intercept them and read them. Then, Mal sends them to the actual gateway's MAC address and they make their way to Bob. The packets will be unchanged and thus unable to be traced back to Mal. This is basically just a redirection of how the traffic is moving through the network (one additional hop).

2. From Alice's perspective, is this attack detectable? If not, why not? If so, how would Alice's setup need to change to detect the attack?

Alice needs to keep a close eye on her ARP cache. She would do well to implement an ARP spoofing detector to help her know when she is being bombarded by ARP replies. She also could confirm that the MAC address for a given IP address is correct by sending out an ARP request for it, and then checking if it matches what exists in her ARP cache. (Mal's attack wouldn't delete other ARP replies, it just sends a whole bunch of other poisoned ones.) Further, Alice could keep track of whether performance has changed (receiving so many ARP replies is bound to slow things down) or traffic has

dramatically increased (if she doesn't normally receive so many ARP replies, at such a high rate of transmission, then that is a key sign). So yes, Alice should be able to detect the attack if she is vigilant and watches over her ARP cache.

3. From Bob's perspective, is this attack detectable?

No, not really. The packets remain unchanged, and they don't come with information about the hops taken in between (so the detour into Mal's clutches will not be indicated). It will appear on Bob's end as though all is well. However, the latency of the communications could be increased, as the packets will have to travel further and pass through Mal. So potentially, if Bob keeps close track of typical interactions with Alice (or others), and this interaction is much slower, then that could be an indicator of Mal's presence. (This has the potential for many false positives though.) But mostly, Bob will not be able to detect if anything is wrong (as his ARP cache is not affected, nor are the packets he received, as Mal is just eavesdropping).

4. Could Alice or Bob detect and/or prevent this attack if the website in question was using HTTPS instead of HTTP? Explain.

By using HTTPS, Alice and Bob very much diminish the productivity of Mal as an eavesdropper. Listening in on encrypted messages is not going to do Mal much good, so that would prevent the attack from being successful (as well as dissuade motivation for the attack to even happen in the first place!). However, using HTTPS is not going to change the ability for Alice and Bob to detect the presence of Mal. The same strategies apply as they do for HTTP (watching for bombardment of ARP replies, changing MAC addresses, or suddenly slower traffic), but HTTPS will not provide new ways. This is because the attack occurs earlier in the process, before TLS even kicks in. Mal poisoned Alice's ARP cache to contain Mal's MAC address rather than the router's. This is not an HTTP/HTTPS problem (which is in the packet); instead, it has to do with where the packet is sent (the MAC address in the header), and how Mal tricked the ARP cache into displaying poisoned information (Mal's MAC address).