

Execution

- a. 00:0c:29:7e:8d:ac
- b. 192.168.225.128
- c. 00:0c:29:8a:f0:f9
- d. 192.168.225.129

e.

```
(kali㉿kali)-[~]  
$ netstat -r  
Kernel IP routing table  
Destination      Gateway          Genmask          Flags      MSS Window  irtt Iface  
e  
default          192.168.225.2   0.0.0.0          UG         0 0        0 eth0  
192.168.225.0    0.0.0.0         255.255.255.0    U         0 0        0 eth0
```

f.

```
(kali㉿kali)-[~]  
$ arp  
Address           HWtype  HWaddress         Flags Mask          Iface  
192.168.225.254   ether   00:50:56:e6:64:a9 C                eth0  
192.168.225.2     ether   00:50:56:f1:36:31 C                eth0
```

g.

```
msfadmin@metasploitable:~$ netstat -r  
Kernel IP routing table  
Destination      Gateway          Genmask          Flags      MSS Window  irtt Iface  
192.168.225.0    *              255.255.255.0    U         0 0        0 eth0  
default          192.168.225.2   0.0.0.0          UG         0 0        0 eth0  
msfadmin@metasploitable:~$ _
```

h.

```
msfadmin@metasploitable:~$ arp  
Address           HWtype  HWaddress         Flags Mask          Iface  
192.168.225.2     ether   00:50:56:f1:36:31 C                eth0  
msfadmin@metasploitable:~$
```

- i. Metasploitable would send the packet to 00:50:56:F1:36:31, this is because it is the only MAC address in Metasploitable's arp cache so it would by default send it to this MAC address.
- j. On metasploitable I do see the HTML file that is from the CS338 website and on Kali there are 11 captured packets of the communication between metasploitable and cs338
- k. Done, more details below (I may have jumped the gun and did the curl command execution early)

l.

```
msfadmin@metasploitable:~$ arp  
Address           HWtype  HWaddress         Flags Mask          Iface  
192.168.225.2     ether   00:50:56:f1:36:31 C                eth0  
192.168.225.128   ether   00:0C:29:7E:8D:AC C                eth0  
192.168.225.1     ether   00:50:56:C0:00:08 C                eth0  
192.168.225.254   ether   00:50:56:E6:64:A9 C                eth0  
msfadmin@metasploitable:~$ _
```

- m. With ettercap spoofing the ARP cache, I imagine when we execute "curl <http://cs338.jeffondich.com/>" on metasploitable it will send it to the MAC address of Kali. that is 00:0c:29:7e:8d:ac.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|-----------------|-----------------|----------|--------|--|
| 1 | 0.000000000 | 192.168.225.129 | 45.79.89.123 | TCP | 74 | 42434 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM TSval=1456083 TSecr=0 WS=32 |
| 2 | 0.007702809 | 192.168.225.129 | 45.79.89.123 | TCP | 74 | [TCP Retransmission] 42434 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM TSval= |
| 5 | 0.063872759 | 192.168.225.129 | 45.79.89.123 | TCP | 60 | 42434 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0 |
| 6 | 0.063916658 | 192.168.225.129 | 45.79.89.123 | HTTP | 212 | GET / HTTP/1.1 |
| 7 | 0.071728555 | 192.168.225.129 | 45.79.89.123 | TCP | 54 | 42434 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0 |
| 8 | 0.071783124 | 192.168.225.129 | 45.79.89.123 | TCP | 212 | [TCP Retransmission] 42434 → 80 [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=158 |
| 13 | 0.127964339 | 192.168.225.129 | 45.79.89.123 | TCP | 60 | 42434 → 80 [ACK] Seq=159 Ack=732 Win=6579 Len=0 |
| 14 | 0.128783091 | 192.168.225.129 | 45.79.89.123 | TCP | 60 | 42434 → 80 [FIN, ACK] Seq=159 Ack=732 Win=6579 Len=0 |
| 15 | 0.135848045 | 192.168.225.129 | 45.79.89.123 | TCP | 54 | [TCP Keep-Alive] 42434 → 80 [ACK] Seq=159 Ack=732 Win=6579 Len=0 |
| 16 | 0.135908044 | 192.168.225.129 | 45.79.89.123 | TCP | 54 | [TCP Retransmission] 42434 → 80 [FIN, ACK] Seq=159 Ack=732 Win=6579 Len=0 |
| 21 | 0.187964761 | 192.168.225.129 | 45.79.89.123 | TCP | 60 | 42434 → 80 [ACK] Seq=160 Ack=733 Win=6579 Len=0 |
| 22 | 0.199870576 | 192.168.225.129 | 45.79.89.123 | TCP | 54 | [TCP Dup ACK 21#1] 42434 → 80 [ACK] Seq=160 Ack=733 Win=6579 Len=0 |
| 3 | 0.056506782 | 45.79.89.123 | 192.168.225.129 | TCP | 60 | 80 → 42434 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 |
| 4 | 0.063738317 | 45.79.89.123 | 192.168.225.129 | TCP | 58 | [TCP Retransmission] 80 → 42434 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 |
| 9 | 0.071860632 | 45.79.89.123 | 192.168.225.129 | TCP | 60 | 80 → 42434 [ACK] Seq=1 Ack=159 Win=64240 Len=0 |
| 10 | 0.079760708 | 45.79.89.123 | 192.168.225.129 | TCP | 54 | [TCP Dup ACK 9#1] 80 → 42434 [ACK] Seq=1 Ack=159 Win=64240 Len=0 |
| 11 | 0.122563505 | 45.79.89.123 | 192.168.225.129 | HTTP | 785 | HTTP/1.1 200 OK (text/html) |
| 12 | 0.12784731 | 45.79.89.123 | 192.168.225.129 | TCP | 785 | [TCP Retransmission] 80 → 42434 [PSH, ACK] Seq=1 Ack=159 Win=64240 Len=731 |
| 17 | 0.130021201 | 45.79.89.123 | 192.168.225.129 | TCP | 60 | 80 → 42434 [ACK] Seq=732 Ack=160 Win=64239 Len=0 |
| 18 | 0.143886007 | 45.79.89.123 | 192.168.225.129 | TCP | 54 | [TCP Dup ACK 17#1] 80 → 42434 [ACK] Seq=732 Ack=160 Win=64239 Len=0 |
| 19 | 0.184565239 | 45.79.89.123 | 192.168.225.129 | TCP | 60 | 80 → 42434 [FIN, PSH, ACK] Seq=732 Ack=160 Win=64239 Len=0 |
| 20 | 0.187837894 | 45.79.89.123 | 192.168.225.129 | TCP | 54 | [TCP Retransmission] 80 → 42434 [FIN, PSH, ACK] Seq=732 Ack=160 Win=64239 Len=0 |

Frame 20: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface
 Ethernet II, Src: VMware_7e:8d:ac (00:0c:29:7e:8d:ac), Dst: VMware_8a:f0:f9 (00:0c:29:7e:8d:ac) 0000 00 0c 29 7e 8d ac 08 00 45 00
 Internet Protocol Version 4, Src: 45.79.89.123, Dst: 192.168.225.129 0010 00 28 be 22 00 00 00 06 53 b9 2d 4f 59 7b c0 a8
 Transmission Control Protocol, Src Port: 80, Dst Port: 42434, Seq: 732, Ack: 160 0020 e1 81 00 50 a5 c2 23 ff 75 b0 8a ce 5f de 50 19
 0030 fa ef 61 78 00 00

- n.
- o. We do see the HTTP response on metasploitable which received the HTML file of the webpage. There are several captured packets in wireshark. We can see in wire shark the [TCP retransmission] packets which is essentially where a packet, intended for cs338.jeffondich was intercepted by Kali and then resent back on its way to cs33.jeffondich.
- p. It appears that ettercap was able to poison metasploitable's ARP cache by adding additional MAC addresses so that instead of everything getting sent by default to 00:50:56:F1:36:31 (which is likely the router) it instead adds some additional checking where in our case, the IP address lined up with instructions (in the ARP cache) to send the packet of data to the MAC address of Kali 00:0c:29:7e:8d:ac. Essentially what happened was additional IP to MAC address mappings were added to the ARP cache which derailed the communication to go through Kali.
- q. If I was going to make an ARP spoofing detector I would probably have my tool keep track of the ARP table over time and whenever there is a change made to the table, check if the MAC address lines up with the proper IP address. Additionally, it would probably want to check if something like one IP address was mapped to two different MAC addresses (alert for possible spoofing) or but also if the number of ARP requests significantly outweighs the number of ARP replies could be an indicator of ARP spoofing. One scenario where my detector might generate false positives would be in something like real life network changes, such as replacing/reconfiguring specific devices. To help combat things like this, we should allow whitelisting by administrators so that these changes don't generate false positives.

Synthesis

- a. Mal's first goal would be to identify who Alice and Bob are. In this case it would be that Alice is metasploitable, Bob is cs33.jeffondich.com and Mal is Kali. Mal would then do

some ARP cache manipulation to Alice's machine, essentially sending forged ARP messages that would associate Mal's MAC address with the IP address of Bob, meaning that data which is intended to be sent to Bob will unknowingly be routed to Mal. While it might make sense that Alice should send all of its data intended for recipients outside its local network to the address of the router, if Alice has a particularly vulnerable ARP cache it is possible for Mal to simply send these ARP messages telling Alice to update her cache with the forged information. Once this is done, all messages intended for Bob will instead be received by Mal who can choose to edit them and send them to Bob or simply eavesdrop on the contents and send it to Bob.

- b. From Alice's perspective she is simply looking at the ARP cache and its associated MAC addresses. If she wasn't able to detect that the ARP cache has been manipulated it would be difficult to detect that there is an issue (so long as Mal sends the proper packets back to Alice so Alice's access to the intended web service isn't denied). Also, ARP spoofing is a relatively silent attack which wouldn't trigger any alerts or change the MAC address of Alice's devices, rather just rerouting their signals. One way that Alice could detect the altering of her ARP cache would be to continuously monitor the ARP cache for changes and inconsistencies.
- c. From Bob's perspective the attack is likely not visible. If Bob is simply acting as a web server which responds to requests then it would not have a way of telling that the person contacting them is impersonating someone else unless it had some anti ARP spoofing technology. It's likely that even if Bob receives a request from an incorrect MAC address they will still send a normal response to the request.
- d. With ARP spoofing Mal would be able to intercept and eavesdrop on the packets being sent between Alice and Bob, but with a HTTPS protocol in place the data itself would be encrypted with a method such that Mal would not have access to the necessary information to decrypt the message. Mal would likely be able to intercept the initial TCP handshake and determine the IP addresses of Alice and Bob but because of the verification procedures used by HTTPS, the meaningful data will be encrypted when it reaches Mal and Mal won't be able to decrypt it. The use of HTTPS doesn't directly assist in the detection of ARP spoofing however, it just increases the security of the data when it is in transmission.