Mary Blanchard and Kaitlyn Peterson

a. Kali MAC address: 00:0c:29:bd:2a:28

b. Kali IP address: 192.168.12.129

c. Metasploitable MAC address: 00:0c:29:e9:25:04

d. Metasploitable IP address: 192.168.12.128

e. Kali routing table:

```
┌──(kali㉿kali)-[~]
└─$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.12.2    0.0.0.0         UG        0 0          0 eth0
192.168.12.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

f. Kali ARP cache:

```
┌──(kali㉿kali)-[~]
└─$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.12.2             ether   00:50:56:f3:c4:72   C                     eth0
```

g. Metasploitable routing table:

```
msfadmin@metasploitable:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.12.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
0.0.0.0         192.168.12.2    0.0.0.0         UG        0 0          0 eth0
```

h. Metasploitable ARP cache:

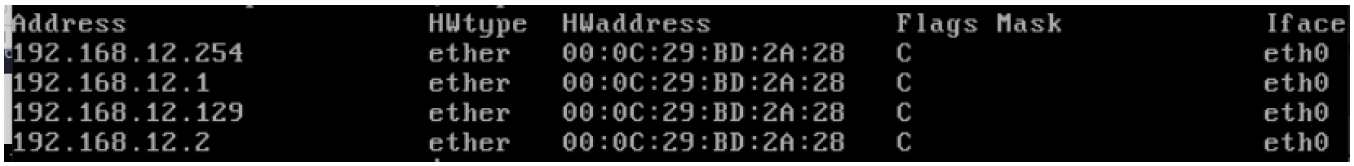
```
msfadmin@metasploitable:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.12.2             ether   00:50:56:F3:C4:72   C                     eth0
```

i. Metasploitable should send the first packet to MAC address 00:50:56LF3:C4:72 because this is the gateway for this machine to reach all IP addresses outside of the local network. We can tell because in the routing table, the gateway IP address is 192.168.12.2. In Metasploitable's ARP table, we found the corresponding MAC address for this IP address.

j. On Metasploitable, we saw the raw text of the HTML for the webpage containing jeff_square_head.jpg. Wireshark did not capture any packets. This follows because the webpage is hosted on the local network, so no packets had to be sent through the gateway. The request was sent directly to the server.

k. [time to poison 👹]

l. New Metasploitable ARP table.The new table includes a few more IP addresses, but all of the associated MAC addresses are the same, and they belong to our Kali machine.



Address	HWtype	HWaddress	Flags Mask	Iface
192.168.12.254	ether	00:0C:29:BD:2A:28	C	eth0
192.168.12.1	ether	00:0C:29:BD:2A:28	C	eth0
192.168.12.129	ether	00:0C:29:BD:2A:28	C	eth0
192.168.12.2	ether	00:0C:29:BD:2A:28	C	eth0

m. Now, when we execute the curl command, we predict that we will see all of the HTTP packets because they will be routed through the Kali machine before being sent along to their destination. This is because the metasploitable machine now thinks that the gateway IP is associated with Kali's MAC address. The very first packet will be sent to Kali's MAC address: 00:0c:29:bd:2a:28.

n. [start wireshark, execute curl command]

o. There is an HTTP response on Metasploitable, the same HTML text came through, but this time it took a lot longer. We can also see captured packets on Wireshark, because the packets are routed through Kali. We can see the HTTP GET Request and the HTTP Response, as well as several TCP packets, including the handshake and several acknowledgements. These packets were repeated twice, as expected with the PITM attack we are running; each packet is sent to on, and then we send it on to its destination.

p. In general, the ARP Protocol is not as secure as other protocols as it accepts updates at any time. This allows us to flood the target machine with ARP updates, which associate IP addresses with our own MAC address. With the ARP Protocol, we send "[IP address] is at [our MAC address]". We do this for each relevant host to whom the target may try to send packets. Thus, in the ARP cache on the target machine, all IP relevant addresses will be routed to our MAC address. When the target machine checks its routing table, it will find an IP address to send its packets. Then it will look up this IP address in its ARP cache, which has been updated by our poisoning attack, and find our MAC address. Then, it will send its packets to us rather than its intended location.

q. When designing an ARP spoofing detector, there are several patterns that the software could detect. Firstly, ARP poisoning often involves gratuitous ARPs: updates to MAC address IP address pairs that are not requested. So, the detector could flag updates that were not requested. Similarly, ARP poisoning often floods these updates to many different local IP addresses, so the detector could also flag high ARP traffic. Finally, the detector could also flag repeated MAC addresses (over different IP addresses). This would be effective as MAC addresses are unique identifiers of the hardware, so if they repeat there is a high probability of ARP poisoning. False positives would be possible with all of these techniques; as it is possible that a server makes a gratuitous ARP update with good intentions, or that there are many requests being made and thus high ARP traffic in general. However, the combination of multiple of these flags are a good indicator of ARP poisoning.