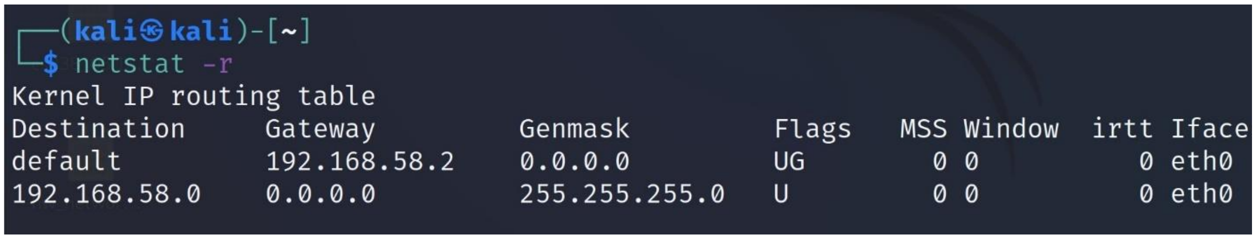
AUTHOR

Kai Johnson, in collaboration with Angela Ellis

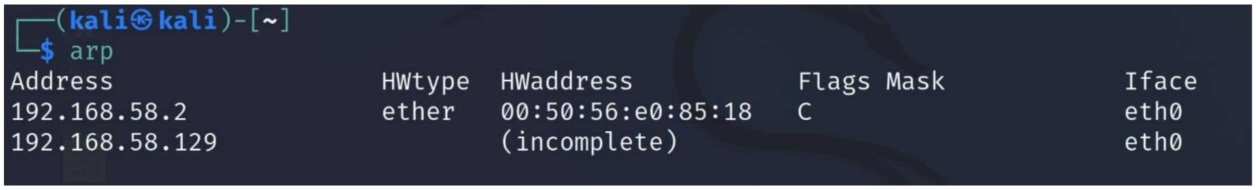
QUESTIONS

A. What is Kali's main interface's MAC address? (The main interface is probably called eth0, but check ifconfig to be sure.)
   a. 00:0c:29:7c:b6:52
B. What is Kali's main interface's IP address?
   a. 192.168.58.128
C. What is Metasploitable's main interface's MAC address?
   a. 00:0c:29:6c:29:3d
D. What is Metasploitable's main interface's IP address?
   a. 192.168.58.129
E. Show Kali's routing table. (Use "netstat -r" to see it with symbolic names, or "netstat -rn" to see it with numerical addresses.)



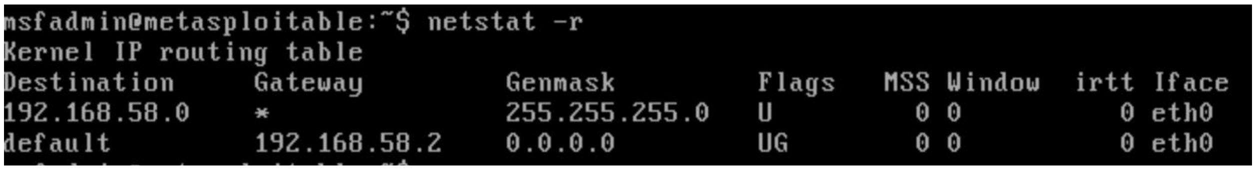
```
┌──(kali㉿kali)-[~]
└─$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         192.168.58.2    0.0.0.0         UG        0 0          0 eth0
192.168.58.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

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



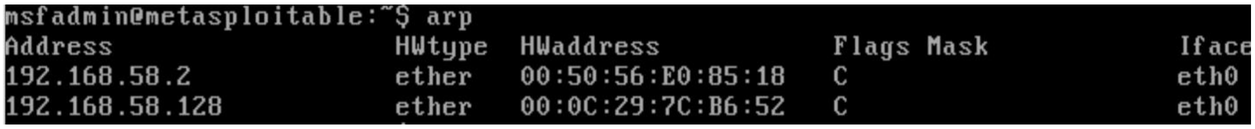
```
┌──(kali㉿kali)-[~]
└─$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.58.2             ether   00:50:56:e0:85:18   C                     eth0
192.168.58.129                   (incomplete)                              eth0
```

G. Show Metasploitable's routing table.



```
msfadmin@metasploitable:~$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.58.0    *               255.255.255.0   U         0 0          0 eth0
default         192.168.58.2    0.0.0.0         UG        0 0          0 eth0
```

H. Show Metasploitable's ARP cache.



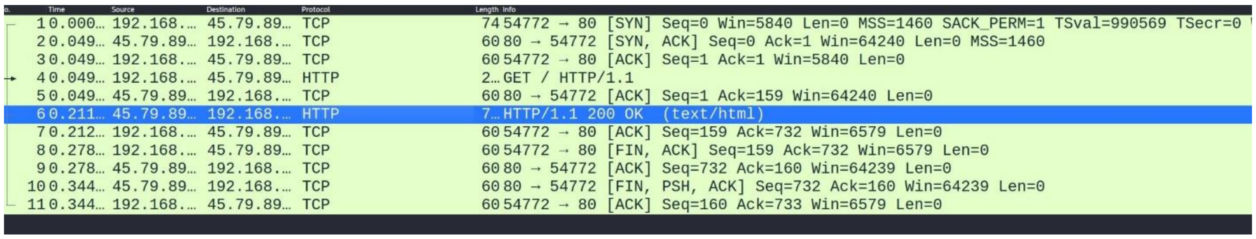
```
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.58.2             ether   00:50:56:E0:85:18   C                     eth0
192.168.58.128           ether   00:0C:29:7C:B6:52   C                     eth0
```

I. 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.

   00:50:56:E0:85:18 -- that's the MAC address for the gateway, which is at 192.168.58.2. We know it's the corresponding MAC address from the arp commands we executed above, and we know that the IP address is for the gateway from using the ip route command. Why does it send it through this MAC address? Because it's essentially the first router for the Metasploitable machine.

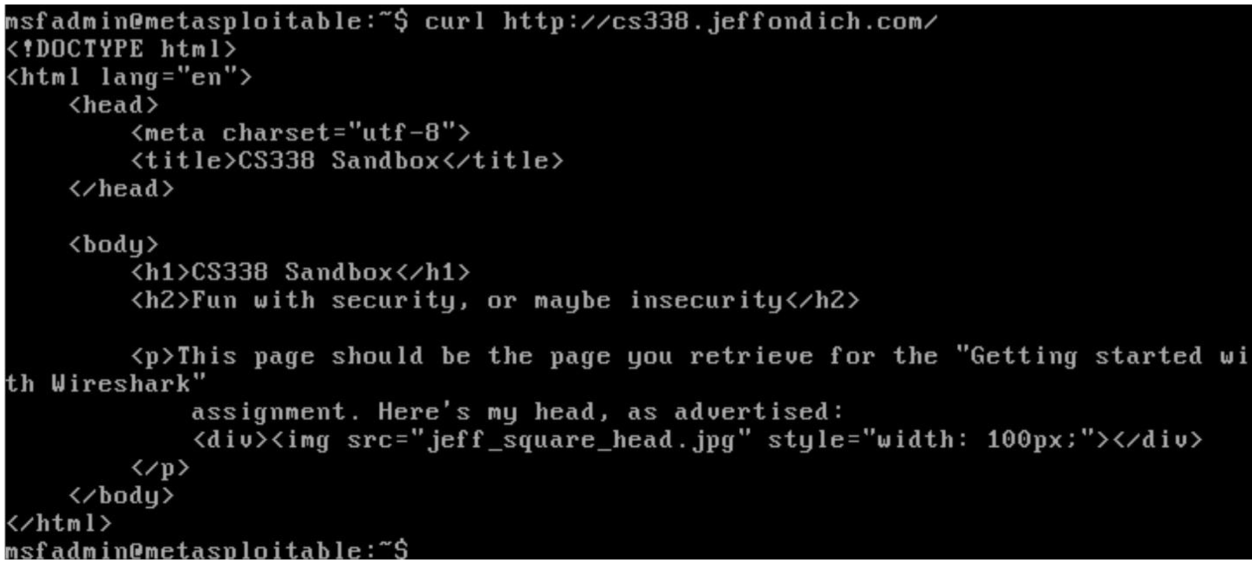
J. Fire up Wireshark on Kali. Start capturing packets for "tcp port http". On Metasploitable, execute "curl http://cs338.jeffondich.com/". On Kali, stop capturing. Do you see an HTTP response on Metasploitable? Do you see any captured packets in Wireshark on Kali?

   Yes! The captured packets (one of the communicating parties being Metasploitable):



```
o.   Time      Source        Destination   Protocol   Length Info
   1 0.000…  192.168.…   45.79.89…   TCP        74 54772 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=990569 TSecr=0
   2 0.049…  45.79.89…   192.168.…   TCP        60 80 → 54772 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
   3 0.049…  192.168.…   45.79.89…   TCP        60 54772 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
   4 0.049…  192.168.…   45.79.89…   HTTP       2… GET / HTTP/1.1
   5 0.049…  45.79.89…   192.168.…   TCP        60 80 → 54772 [ACK] Seq=1 Ack=159 Win=64240 Len=0
   6 0.211…  45.79.89…   192.168.…   HTTP       7… HTTP/1.1 200 OK  (text/html)
   7 0.212…  192.168.…   45.79.89…   TCP        60 54772 → 80 [ACK] Seq=159 Ack=732 Win=6579 Len=0
   8 0.278…  192.168.…   45.79.89…   TCP        60 54772 → 80 [FIN, ACK] Seq=159 Ack=732 Win=6579 Len=0
   9 0.278…  45.79.89…   192.168.…   TCP        60 80 → 54772 [ACK] Seq=732 Ack=160 Win=64239 Len=0
  10 0.344…  45.79.89…   192.168.…   TCP        60 80 → 54772 [FIN, PSH, ACK] Seq=732 Ack=160 Win=64239 Len=0
  11 0.344…  192.168.…   45.79.89…   TCP        60 54772 → 80 [ACK] Seq=160 Ack=733 Win=6579 Len=0
```

The HTML Metasploitable received:



```
msfadmin@metasploitable:~$ curl http://cs338.jeffondich.com/
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>CS338 Sandbox</title>
    </head>

    <body>
        <h1>CS338 Sandbox</h1>
        <h2>Fun with security, or maybe insecurity</h2>

        <p>This page should be the page you retrieve for the "Getting started wi
th Wireshark"
            assignment. Here's my head, as advertised:
            <div><img src="jeff_square_head.jpg" style="width: 100px;"></div>
        </p>
    </body>
</html>
msfadmin@metasploitable:~$
```

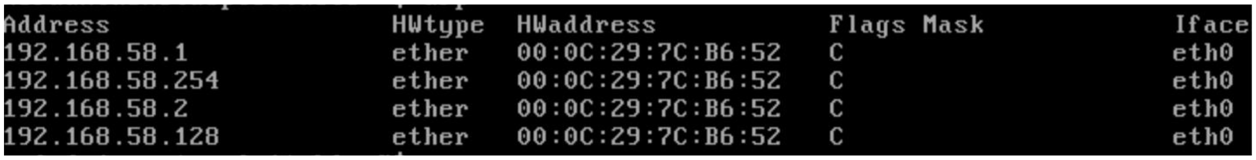
K. Now, it's time to be Mal (who will, today, merely eavesdrop). Use Ettercap to do ARP spoofing (also known as ARP Cache Poisoning) with Metasploitable as your target. There are many online tutorials on how to do this (here's one). Find one you like, and start spoofing your target. NOTE: most of these tutorials are showing an old user interface for Ettercap, which may make them confusing. The steps you're trying to take within Ettercap are:
   a. Start sniffing (*not* bridged sniffing) on eth0
   b. Scan for Hosts
   c. View the Hosts list
   d. Select your Metasploit VM from the Host List

e. Add that host as Target 1
f. Start ARP Poisoning (including Sniff Remote Connections)
g. Do your stuff with wireshark and Metasploitable
h. Stop ARP Poisoning

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 PITM 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.)

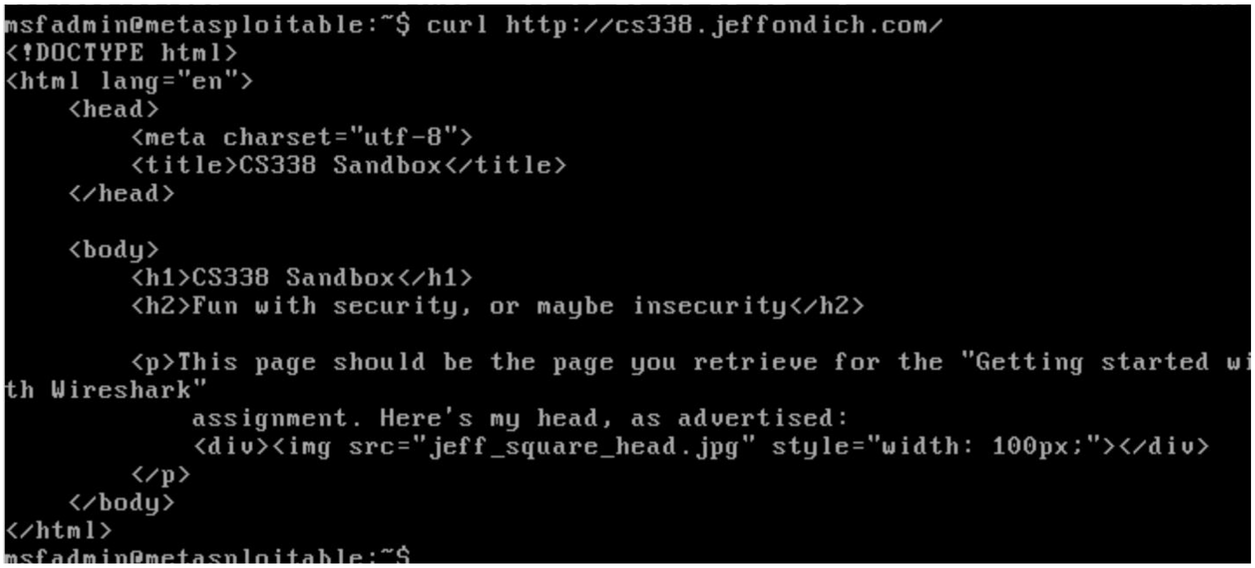
L. Show Metasploitable's ARP cache. How has it changed?
It tracks 2 additional addresses: 192.168.58.1 and 192.168.58.254. Additionally, all the MAC addresses now look to be the same (presumably, it's the MAC address of Ettercap?).



```
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.58.1             ether   00:0C:29:7C:B6:52   C                     eth0
192.168.58.254           ether   00:0C:29:7C:B6:52   C                     eth0
192.168.58.2             ether   00:0C:29:7C:B6:52   C                     eth0
192.168.58.128           ether   00:0C:29:7C:B6:52   C                     eth0
```

M. 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.
It will probably proceed normally, but the MAC address will be 00:0C:29:7C:B6:52. Kali is sitting in the middle, getting all of Metasploitable's traffic, and sending it on.
N. Start Wireshark capturing "tcp port http" again.
O. 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?

I still get the HTML page on Metasploitable just fine:

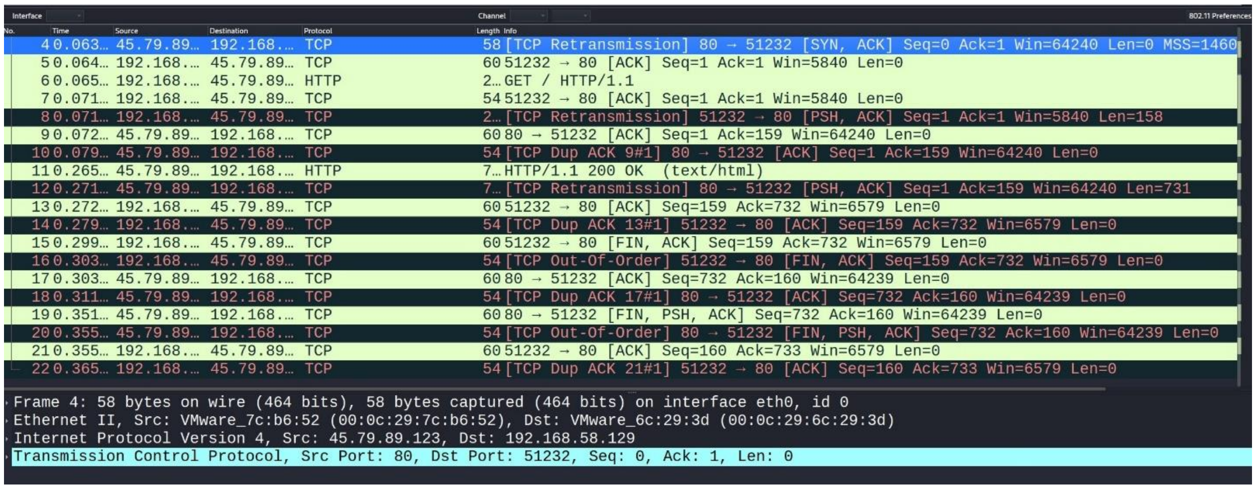


```
msfadmin@metasploitable:~$ curl http://cs338.jeffondich.com/
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>CS338 Sandbox</title>
    </head>

    <body>
        <h1>CS338 Sandbox</h1>
        <h2>Fun with security, or maybe insecurity</h2>

        <p>This page should be the page you retrieve for the "Getting started wi
th Wireshark"
            assignment. Here's my head, as advertised:
            <div><img src="jeff_square_head.jpg" style="width: 100px;"></div>
        </p>
    </body>
</html>
msfadmin@metasploitable:~$
```

But, the captured Wireshark packets look a little different:

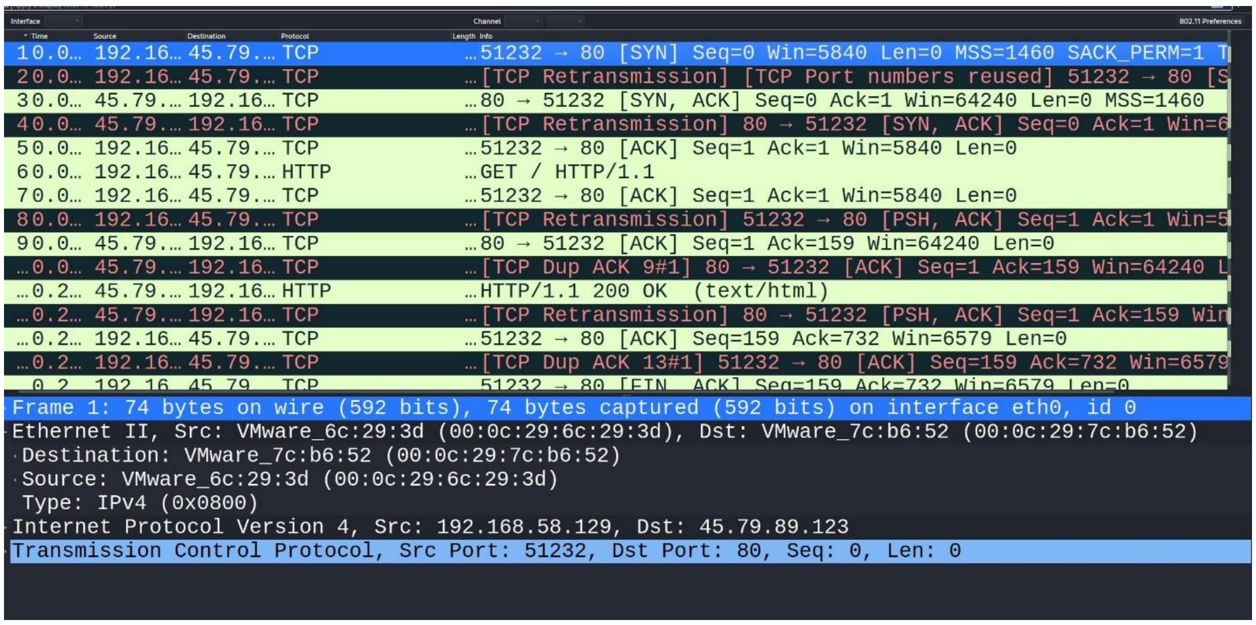


```
Interface                                              Channel                                         802.11 Preferences
No.  Time     Source     Destination  Protocol          Length Info
  4 0.063… 45.79.89… 192.168.… TCP     58 [TCP Retransmission] 80 → 51232 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  5 0.064… 192.168.… 45.79.89… TCP     60 51232 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
  6 0.065… 192.168.… 45.79.89… HTTP    2… GET / HTTP/1.1
  7 0.071… 192.168.… 45.79.89… TCP     54 51232 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
  8 0.071… 192.168.… 45.79.89… TCP     2… [TCP Retransmission] 51232 → 80 [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=158
  9 0.072… 45.79.89… 192.168.… TCP     60 80 → 51232 [ACK] Seq=1 Ack=159 Win=64240 Len=0
 10 0.079… 45.79.89… 192.168.… TCP     54 [TCP Dup ACK 9#1] 80 → 51232 [ACK] Seq=1 Ack=159 Win=64240 Len=0
 11 0.265… 45.79.89… 192.168.… HTTP    7… HTTP/1.1 200 OK  (text/html)
 12 0.271… 45.79.89… 192.168.… TCP     7… [TCP Retransmission] 80 → 51232 [PSH, ACK] Seq=1 Ack=159 Win=64240 Len=731
 13 0.272… 192.168.… 45.79.89… TCP     60 51232 → 80 [ACK] Seq=159 Ack=732 Win=6579 Len=0
 14 0.279… 192.168.… 45.79.89… TCP     54 [TCP Dup ACK 13#1] 51232 → 80 [ACK] Seq=159 Ack=732 Win=6579 Len=0
 15 0.299… 192.168.… 45.79.89… TCP     60 51232 → 80 [FIN, ACK] Seq=159 Ack=732 Win=6579 Len=0
 16 0.303… 192.168.… 45.79.89… TCP     54 [TCP Out-Of-Order] 51232 → 80 [FIN, ACK] Seq=159 Ack=732 Win=6579 Len=0
 17 0.303… 45.79.89… 192.168.… TCP     60 80 → 51232 [ACK] Seq=732 Ack=160 Win=64239 Len=0
 18 0.311… 45.79.89… 192.168.… TCP     54 [TCP Dup ACK 17#1] 80 → 51232 [ACK] Seq=732 Ack=160 Win=64239 Len=0
 19 0.351… 45.79.89… 192.168.… TCP     60 80 → 51232 [FIN, PSH, ACK] Seq=732 Ack=160 Win=64239 Len=0
 20 0.355… 45.79.89… 192.168.… TCP     54 [TCP Out-Of-Order] 80 → 51232 [FIN, PSH, ACK] Seq=732 Ack=160 Win=64239 Len=0
 21 0.355… 192.168.… 45.79.89… TCP     60 51232 → 80 [ACK] Seq=160 Ack=733 Win=6579 Len=0
 22 0.365… 192.168.… 45.79.89… TCP     54 [TCP Dup ACK 21#1] 51232 → 80 [ACK] Seq=160 Ack=733 Win=6579 Len=0

Frame 4: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface eth0, id 0
Ethernet II, Src: VMware_7c:b6:52 (00:0c:29:7c:b6:52), Dst: VMware_6c:29:3d (00:0c:29:6c:29:3d)
Internet Protocol Version 4, Src: 45.79.89.123, Dst: 192.168.58.129
Transmission Control Protocol, Src Port: 80, Dst Port: 51232, Seq: 0, Ack: 1, Len: 0
```

P. 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 PITM/MITM 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.)

The introduction of the man in the middle made added two additional hosts to the cache and set their MAC addresses to all be identical for Metasploitable's ARP cache. As a result, the packets of the request now get sent through Kali, who then retransmits them onwards ("towards" whatever server will respond to the HTTP request). For instance, the first SYN request that wireshark captures is sent from 192.168.58.129 (Metasploitable), from the MAC address 00:0c:29:6c:29:3d (again, Metasploitable). It is sent to the IP address of 45.(etc), presumably that of cs338.jeffondich.com, but the destination mac address in the ethernet header is 00:0c:29:7c:b6:52 (which is kali!). We then observe a TCP retransmission with the same IP & TCP headers (the source & destinations IP addresses are still the same), but a modified ethernet header, this time with kali's MAC address as the source, and 00:50:56:E0:85:18 as the destination—that is the MAC address for the gateway that we identified in part I. All of the TCP packets are intercepted and modified and retransmitted in this manner; however, the HTTP request & response are captured only once (GET is between Kali and Metasploitable, OK between the gateway and Kali).



```
Time    Source     Destination Protocol  Length Info
1 0.0… 192.16… 45.79.… TCP        …51232 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 T
2 0.0… 192.16… 45.79.… TCP        …[TCP Retransmission] [TCP Port numbers reused] 51232 → 80 [S
3 0.0… 45.79.… 192.16… TCP        …80 → 51232 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
4 0.0… 45.79.… 192.16… TCP        …[TCP Retransmission] 80 → 51232 [SYN, ACK] Seq=0 Ack=1 Win=6
5 0.0… 192.16… 45.79.… TCP        …51232 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
6 0.0… 192.16… 45.79.… HTTP       …GET / HTTP/1.1
7 0.0… 192.16… 45.79.… TCP        …51232 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
8 0.0… 192.16… 45.79.… TCP        …[TCP Retransmission] 51232 → 80 [PSH, ACK] Seq=1 Ack=1 Win=5
9 0.0… 45.79.… 192.16… TCP        …80 → 51232 [ACK] Seq=1 Ack=159 Win=64240 Len=0
…0.0… 45.79.… 192.16… TCP        …[TCP Dup ACK 9#1] 80 → 51232 [ACK] Seq=1 Ack=159 Win=64240 L
…0.2… 45.79.… 192.16… HTTP       …HTTP/1.1 200 OK  (text/html)
…0.2… 45.79.… 192.16… TCP        …[TCP Retransmission] 80 → 51232 [PSH, ACK] Seq=1 Ack=159 Win
…0.2… 192.16… 45.79.… TCP        …51232 → 80 [ACK] Seq=159 Ack=732 Win=6579 Len=0
…0.2… 192.16… 45.79.… TCP        …[TCP Dup ACK 13#1] 51232 → 80 [ACK] Seq=159 Ack=732 Win=6579
…0.2  192.16  45.79   TCP         51232 → 80 [FIN, ACK] Seq=159 Ack=732 Win=6579 Len=0

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0, id 0
Ethernet II, Src: VMware_6c:29:3d (00:0c:29:6c:29:3d), Dst: VMware_7c:b6:52 (00:0c:29:7c:b6:52)
 Destination: VMware_7c:b6:52 (00:0c:29:7c:b6:52)
 Source: VMware_6c:29:3d (00:0c:29:6c:29:3d)
 Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.58.129, Dst: 45.79.89.123
Transmission Control Protocol, Src Port: 51232, Dst Port: 80, Seq: 0, Len: 0
```

Q. 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.)

I might count how many IP addresses are associated with a particular MAC address; in the above example, the MAC addresses for each of the IP addresses in Metasploitable's ARP cache were unique pre MITM and identical during MITM. Maybe it's would be possible to count how many IP addresses a MAC address is usually associated with, and then send a warning if that number deviates? That would require an accurate baseline, though. Additionally, it's probable that there are some legitimate cases when multiple IP addresses are linked to a single MAP address, so there would likely be some false positives with this method in those cases.