Emma Roskopf & Lily Haas

A. Kali's MAC address from eth0 is 00:0c:29:7a:77:84
B. Kali's main interface IP address 192.168.60.128
C. Metasploitable's main MAC address is fe80::20c::29ff:feff:ae4d/64
D. Metasploitable's main IP address is 192.168.60.129
E. Kali's routing table

```
┌──(kali㉿kali)-[~]
└─$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Ifac
e
default         192.168.60.2    0.0.0.0         UG        0 0          0 eth0
192.168.60.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0

┌──(kali㉿kali)-[~]
└─$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.60.2    0.0.0.0         UG        0 0          0 eth0
192.168.60.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

F. Kali's ARP cache

```
┌──(kali㉿kali)-[~]
└─$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.60.254           ether   00:50:56:fa:6f:3f   C                     eth0
192.168.60.2             ether   00:50:56:ee:bc:c1   C                     eth0

┌──(kali㉿kali)-[~]
└─$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.60.254           ether   00:50:56:fa:6f:3f   C                     eth0
192.168.60.2             ether   00:50:56:ee:bc:c1   C                     eth0
```

G. Metasploitable's routing table

```
msfadmin@metasploitable:~$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.60.0    *               255.255.255.0   U         0 0          0 eth0
default         192.168.60.2    0.0.0.0         UG        0 0          0 eth0
msfadmin@metasploitable:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.60.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
0.0.0.0         192.168.60.2    0.0.0.0         UG        0 0          0 eth0
msfadmin@metasploitable:~$
```

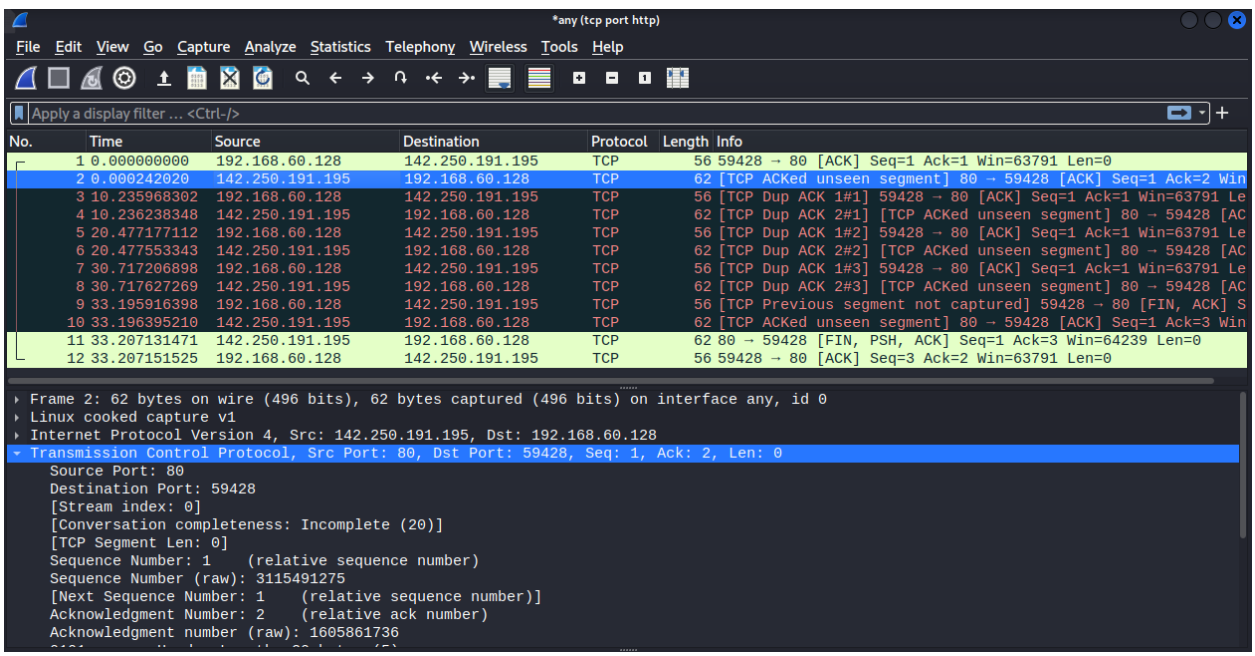
H. Metasploitable's ARP cache

```
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.60.2             ether   00:50:56:EE:BC:C1   C                     eth0
192.168.60.254           ether   00:50:56:FA:6F:3F   C                     eth0
msfadmin@metasploitable:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.60.2             ether   00:50:56:EE:BC:C1   C                     eth0
192.168.60.254           ether   00:50:56:FA:6F:3F   C                     eth0
msfadmin@metasploitable:~$
```

I. To which MAC address should Metasploitable send the TCP SYN packet to get the whole HTTP query started? Explain why.

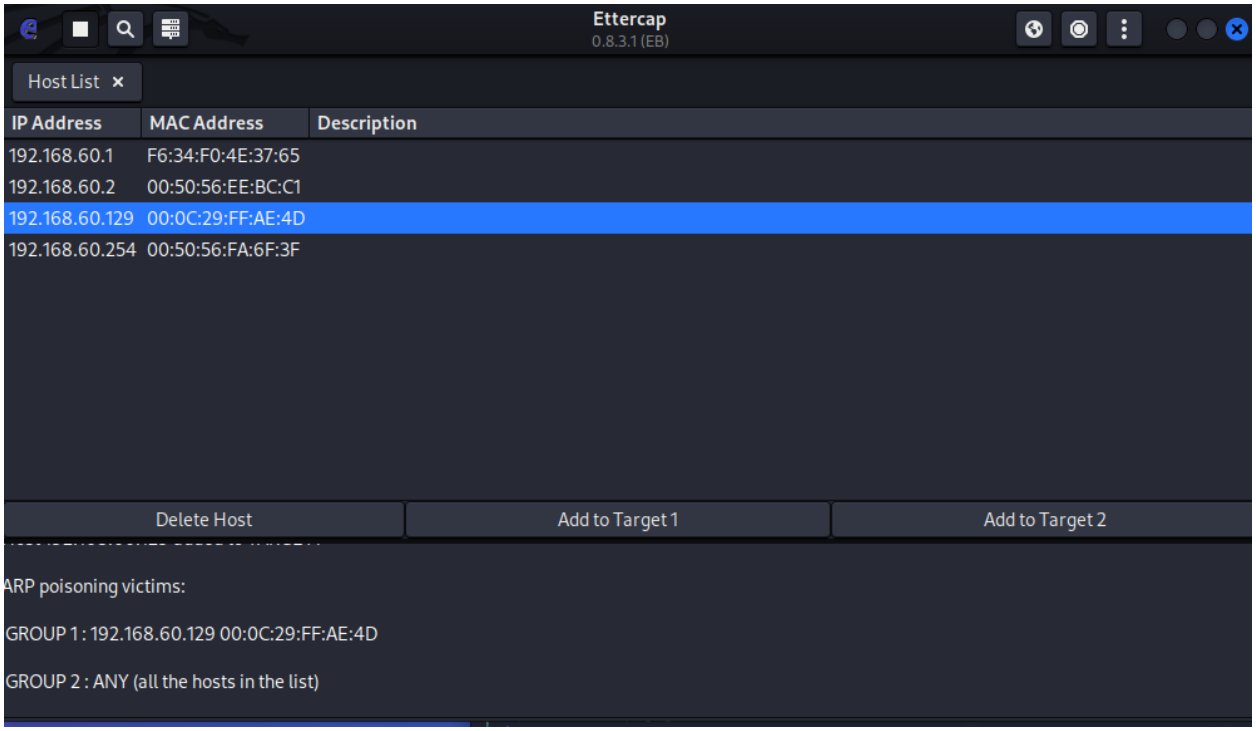
Metasploitable should send the TCP SYN packet to the MAC address of the Gateway, 00:50:56:EE:BC:C1 because this is the MAC address of 192.168.60.2, the Gateway as identified by netstat. The packet will be wrapped with 00:50:56:EE:BC:C1 as the destination hardware address, and the gateway will pass it on until the packet gets where it is supposed to go.

J. Capturing "tcp port http" on wireshark, I get TCP packets going back and forth from 192.168.60.128 to 142.250.191.195 which must be http://cs338.jeffondich.com/.



Yes we do see packets, but only TCP ones

K. We ARP poisoned



L. Metasploitable's ARP cache has changed. The ARP cache of the Gateway's IP, 192.168.60.2, now has Kali's MAC address instead of the actual one. All MAC addresses in the ARP cache are Kali. Metasploitable now thinks Kali is the gateway.

```
msfadmin@metasploitable:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.60.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
0.0.0.0         192.168.60.2    0.0.0.0         UG        0 0          0 eth0
msfadmin@metasploitable:~$ _
```

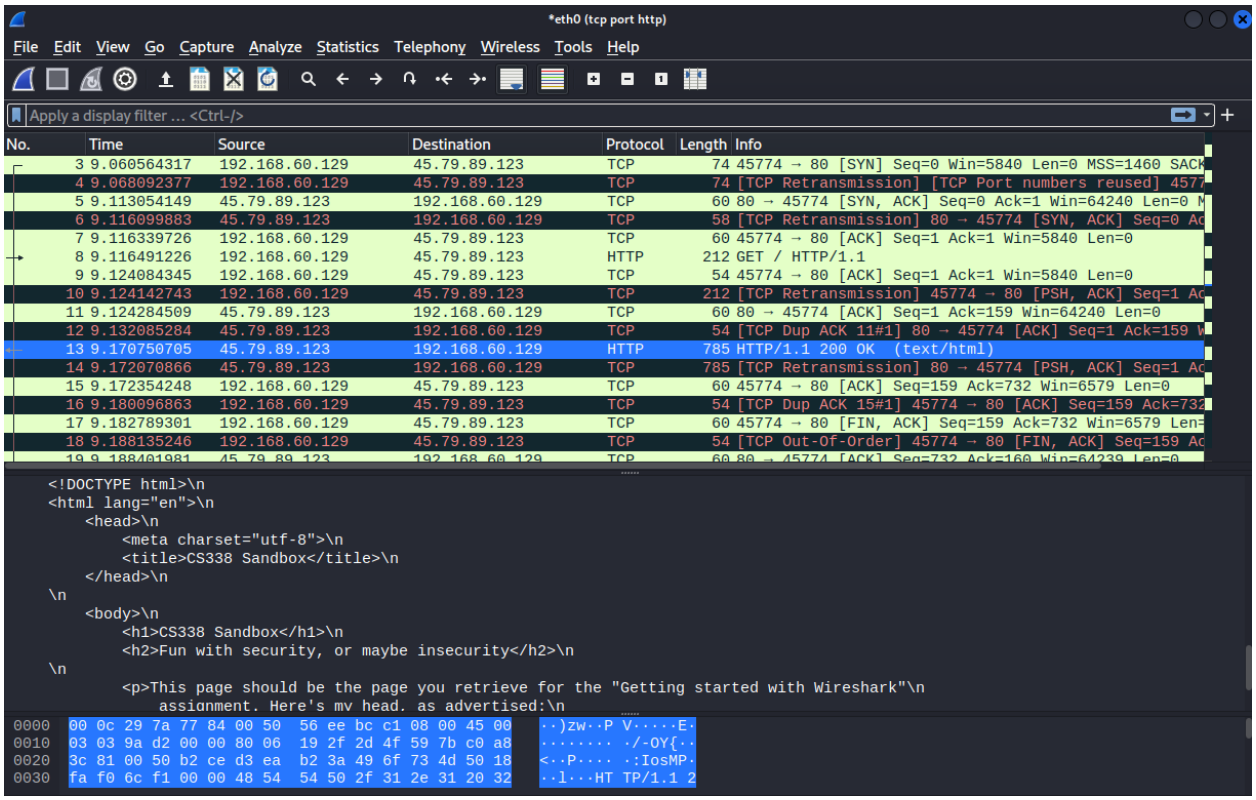
```
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.60.1             ether   00:0C:29:7A:77:84   C                     eth0
192.168.60.254           ether   00:0C:29:7A:77:84   C                     eth0
192.168.60.2             ether   00:0C:29:7A:77:84   C                     eth0
```

M. 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.

Our MAC address is now the Gateway's MAC address because we made all MAC addresses in the ARP cache us. So when Metasploitable sends a packet, it is wrapped with the destination hardware address for Kali. Kali receives the packet, and forwards it to the actual gateway which ends up going to the website, and when the packet comes back it routes through Kali again.

N. Yeah we started the Wireshark capture again

O. Yes we are correct, we can now see the HTTP packets and read them under “Line-based text data.” We have the same TCP things as before, but also we can now pick up the HTTP packets because we are the gateway. For the HTTP packets, we can see the “GET / HTTP/1.1” and then the “HTTP/1.1 200 OK” response from the website, and we can read the corresponding HTML code by looking at this packet.



P. Explain:
When we started ARP poisoning we changed all of the MAC addresses in the ARP cache to our MAC address. This means instead of sending packets to the real gateway to send them off the local network, Metasploitable is sending them to us and we read them before forwarding them to the real gateway. Then, the gateway sends the packets back to us because we are the ones that sent the request through, and so we see everything.

Q. If we made a detector, we could have it make sure that the MAC addresses for each device in the ARP cache are unique. A false positive that could arise from this is if one device somehow had multiple connections to the network. We could prevent this by only checking that the MAC address of the gateway is not duplicated in the network. Also there could be a false positive if the randomly generated MAC address of some program like Kali, that generates in a certain range, could collide with another iteration of Kali. [This question on vmware.com]

asks a similar question if virtual machine generated MAC addresses could end up being the same, and yes they could if the number of virtual machines exceeds the number of addresses the server can assign to them. But checking for duplicates against specifically the gateway could still work to stop PITM attacks. In the case of duplicates due to too many virtual machines, the computer just looking for duplicates might accidentally send a warning about a PITM attack when it’s actually just someone using an absolutely ridiculous number of virtual machines.