———— Execution ————

a. Kali's eth0 MAC address: `00:0c:29:f4:78:16`

b. Kali's eth0 IP address: `192.168.240.128`

c. Metasploitable's eth0 MAC address: `00:0c:29:45:b4:15`

d. Metasploitable's eth0 IP address: `192.168.240.129`

e. Kali's routing table:

```
┌──(kali㉿kali)-[~]
└─$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         192.168.240.2   0.0.0.0         UG        0 0          0 eth0
192.168.240.0   0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

f. Kali's ARP cache:

```
┌──(kali㉿kali)-[~]
└─$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.240.2            ether   00:50:56:ea:c5:10   C                     eth0
192.168.240.254          ether   00:50:56:e1:76:c0   C                     eth0
```

g. Metasploitable's routing table:

```
msfadmin@metasploitable:~$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.240.0   *               255.255.255.0   U         0 0          0 eth0
default         192.168.240.2   0.0.0.0         UG        0 0          0 eth0
```

h. Metasploitable's ARP cache:

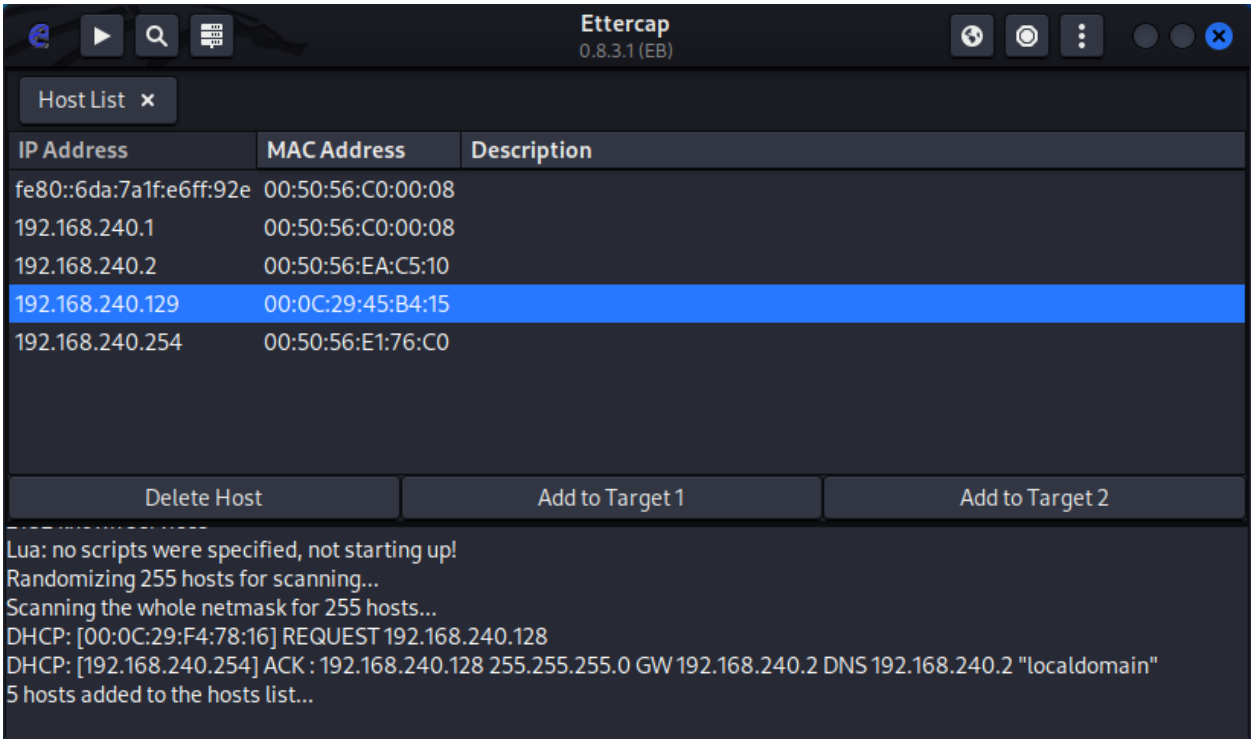
```
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.240.2            ether   00:50:56:EA:C5:10   C                     eth0
```

i. Metasploitable should send the TCP SYN packet to MAC address `00:50:56:ea:c5:10`. The destination IP masked with `255.255.255.0` is not `192.168.240.0`, so the packet needs to be sent to the gateway `192.168.240.2`, which resolves to the MAC address `00:50:56:ea:c5:10` in the ARP cache.

j. After running the curl command on Metasploitable, I was able to see an HTTP response. There were also packets captured by Kali, showcasing a full TCP interaction for the HTTP request (11 packets captured in total):

No.	Source	Destination	Protocol	Info
1	192.168.240.129	45.79.89.123	TCP	51297 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460
2	45.79.89.123	192.168.240.129	TCP	80 → 51297 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len
3	192.168.240.129	45.79.89.123	TCP	51297 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
4	192.168.240.129	45.79.89.123	HTTP	GET / HTTP/1.1
5	45.79.89.123	192.168.240.129	TCP	80 → 51297 [ACK] Seq=1 Ack=159 Win=64240 Len=0
6	45.79.89.123	192.168.240.129	HTTP	HTTP/1.1 200 OK (text/html)
7	192.168.240.129	45.79.89.123	TCP	51297 → 80 [ACK] Seq=159 Ack=732 Win=6579 Len=0
8	192.168.240.129	45.79.89.123	TCP	51297 → 80 [FIN, ACK] Seq=159 Ack=732 Win=6579
9	45.79.89.123	192.168.240.129	TCP	80 → 51297 [ACK] Seq=732 Ack=160 Win=64239 Len=
10	45.79.89.123	192.168.240.129	TCP	80 → 51297 [FIN, PSH, ACK] Seq=732 Ack=160 Win=
11	192.168.240.129	45.79.89.123	TCP	51297 → 80 [ACK] Seq=160 Ack=733 Win=6579 Len=0

k. List of hosts in Ettercap (Metasploitable IP/MAC highlighted):



l. Below is Metasploitable's new ARP cache (after poisoning). It now has listings for three unique IP addresses, and says that each IP address is associated with Kali's MAC address.

```
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.240.1            ether   00:0C:29:F4:78:16   C                     eth0
192.168.240.254          ether   00:0C:29:F4:78:16   C                     eth0
192.168.240.128          ether   00:0C:29:F4:78:16   C                     eth0
192.168.240.2            ether   00:0C:29:F4:78:16   C                     eth0
```

m. When I execute the `curl` command on Metasploitable, I predict that it will send the TCP SYN packet to the Kali VM. Metasploitable will try to send the packet to the IP of Jeff's server, which will resolve to `192.168.240.2` via the routing table. Then, Metasploitable will look at the poisoned ARP cache and see that the IP `192.168.240.2` is associated with the MAC address `00:0C:29:F4:78:16` so it will send the packet to that MAC address on the local network. But this MAC address is actually Kali's, so Kali will receive the packet. Also, since Kali doesn't have the info Metasploitable wants, the curl command will be unsuccessful.

n. Wireshark started :)

o. When I execute the `curl` command on the poisoned Metasploitable, I still get an HTTP response. This time, Wireshark captured 22 packets and labels almost half of them as either [TCP Retransmission] or [TCP Dup ACK]. Looking at each of the packets reveals that Kali is acting as a true entity in the middle of this interaction. Every packet in the unimpeded TCP interaction from part (j) above has two counterparts here: one between Kali and Metasploitable and one between Kali and cs338.jeffondich.com.

No.	Source	Destination	Protocol	Info
1	192.168.240.129	45.79.89.123	TCP	33128 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460
2	192.168.240.129	45.79.89.123	TCP	[TCP Retransmission] [TCP Port numbers reused]
3	45.79.89.123	192.168.240.129	TCP	80 → 33128 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len
4	45.79.89.123	192.168.240.129	TCP	[TCP Retransmission] 80 → 33128 [SYN, ACK] Seq=
5	192.168.240.129	45.79.89.123	TCP	33128 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
6	192.168.240.129	45.79.89.123	HTTP	GET / HTTP/1.1
7	192.168.240.129	45.79.89.123	TCP	33128 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
8	192.168.240.129	45.79.89.123	TCP	[TCP Retransmission] 33128 → 80 [PSH, ACK] Seq=
9	45.79.89.123	192.168.240.129	TCP	80 → 33128 [ACK] Seq=1 Ack=159 Win=64240 Len=0
10	45.79.89.123	192.168.240.129	TCP	[TCP Dup ACK 9#1] 80 → 33128 [ACK] Seq=1 Ack=15
11	45.79.89.123	192.168.240.129	HTTP	HTTP/1.1 200 OK (text/html)
12	45.79.89.123	192.168.240.129	TCP	[TCP Retransmission] 80 → 33128 [PSH, ACK] Seq=
13	192.168.240.129	45.79.89.123	TCP	33128 → 80 [ACK] Seq=159 Ack=732 Win=6579 Len=0
14	192.168.240.129	45.79.89.123	TCP	[TCP Dup ACK 13#1] 33128 → 80 [ACK] Seq=159 Ack
15	192.168.240.129	45.79.89.123	TCP	33128 → 80 [FIN, ACK] Seq=159 Ack=732 Win=6579
16	192.168.240.129	45.79.89.123	TCP	[TCP Retransmission] 33128 → 80 [FIN, ACK] Seq=
17	45.79.89.123	192.168.240.129	TCP	80 → 33128 [ACK] Seq=732 Ack=160 Win=64239 Len=
18	45.79.89.123	192.168.240.129	TCP	[TCP Dup ACK 17#1] 80 → 33128 [ACK] Seq=732 Ack
19	45.79.89.123	192.168.240.129	TCP	80 → 33128 [FIN, PSH, ACK] Seq=732 Ack=160 Win=
20	45.79.89.123	192.168.240.129	TCP	[TCP Retransmission] 80 → 33128 [FIN, PSH, ACK]
21	192.168.240.129	45.79.89.123	TCP	33128 → 80 [ACK] Seq=160 Ack=733 Win=6579 Len=0
22	192.168.240.129	45.79.89.123	TCP	[TCP Dup ACK 21#1] 33128 → 80 [ACK] Seq=160 Ack

p. When the attack started, Kali sent out 6 ARP reply packets. 3 of these replies were sent to the Metasploitable VM, telling it that the other three IPs on the VMWare network were all located at MAC address `00:0c:29:f4:78:16` (Kali's MAC address). Each of the other 3 packets were sent out to the other three IPs on the VMWare network, telling those machines that Metasploitable's IP was located at MAC address `00:0c:29:f4:78:16` (Kali's MAC address). Kali repeated this process 5 times, sending bundles of ARP reply packets out every 1 second. After this, it sent out 3 ARP request packets, looking to see which machines were at each of the non-Metasploitable IPs on the network. It got replies from each of the other machines, presumably updated its ARP cache, and then sent out two more reply bundles before I stopped capturing packets. In this way, the Kali VM has set itself up so that all of the traffic that would go to or from Metasploitable would now travel to Kali first (before being forwarded on correctly, since Kali's ARP cache remained unpoisoned).

q. When the ARP poisoning attack started, Kali sent out a bunch of ARP reply packets telling the other machines on the network about the change in MAC address. Most other machines on the network don't appear to send ARP announcements out frequently, so an ARP spoofing detector might look for one MAC address generating a lot of packets in a relatively short amount of time. However, since there doesn't seem to be a complete standard amount of time that announcement packets should be sent out, it's possible that false positives will be thrown on devices that happen to be more talkative than most others, even if no ARP spoofing is involved. False positives could also be thrown on devices connecting to the network for the first time, which need to send out requests to establish their ARP cache.

———— Synthesis ————

a. In this attack, Mal's strategy is to intercept all of the traffic going between Alice and Bob. Mal does this by manipulating the ARP caches of the other machines on the network. An ARP cache is a table stored on a machine which tells it which other machine on the network a specific packet should be sent to, given the destination IP address of the packet. If Mal's attack is successful, the ARP caches of the other machines on the network should now be configured such that any traffic looking to reach Alice's IP is first directed to Mal, and any traffic coming from Alice's machine is also directed to Mal. Since Mal intercepts all of the packets going to and from Alice's machine, she can control exactly what information Alice receives.

b. Alice could possibly detect that an ARP poisoning attack is happening if she recognizes that all of her ARP cache listings now point to one MAC address. But this wouldn't necessarily mean that she's the victim of an attack; it could just be that there is only one other machine on the network. In general, if Alice was able to detect that a listing in her ARP cache was wrong, there wouldn't be much of a point to her looking in her ARP cache in the first place. Alice could set up an ARP spoofing detector to warn her of the attack, but that would require extra setup.

c. If Mal uses this attack to block traffic to and from Alice's machine, unless Bob is expecting a connection, he simply won't know that Alice is trying to send packets to him, making this attack pretty undetectable to Bob. Even if Bob is expecting a connection, he still might not be able to tell if ARP poisoning is the exact attack at play, or if an interceptor is simply blocking all traffic at some point in between him and Alice in a different way. And for successful connections, Bob shouldn't have any knowledge of Alice's MAC address in the first place.

d. While the poisoning of the ARP cache can't be prevented with HTTPS, the fact that the connection is initiated with a TLS handshake and all of the data is encrypted with public-key cryptography means that Mal can really only act as an eavesdropper. At worst, she could drop all of the packets going out from (or coming to) Mal's device, but she wouldn't be able to read the data since only Alice and Bob would have the encryption keys.