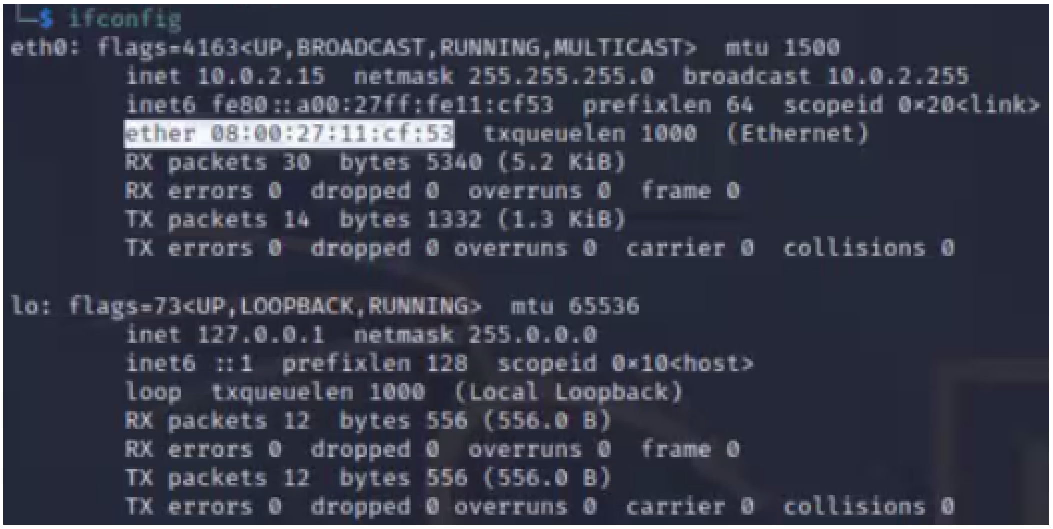
Person-in-the-Middle via ARP spoofing
CS 231
Nathan Hayes-Rich and Peter McCrea

A. What is Kali's main interface's MAC address? (The main interface is probably called eth0, but check ifconfig to be sure.)

eth0 is the main interface.
The MAC address: 08:00:27:11:cf:53



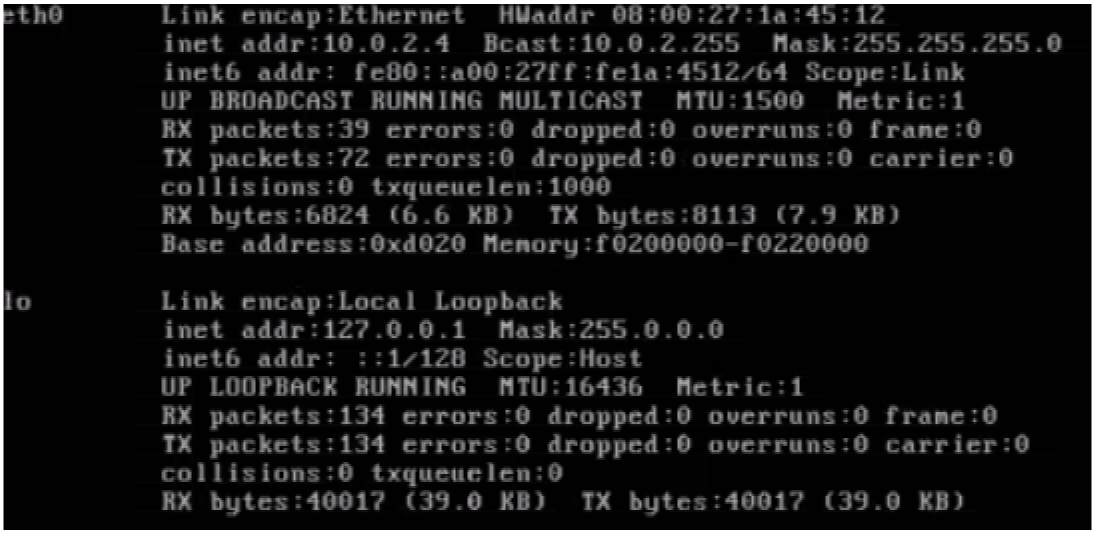
```
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe11:cf53  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:11:cf:53  txqueuelen 1000  (Ethernet)
        RX packets 30  bytes 5340 (5.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14  bytes 1332 (1.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 12  bytes 556 (556.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 556 (556.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

B. What is Kali's main interface's IP address?

Private IP address: 10.0.2.15
Public IP address: 137.22.28.3

C. What is Metasploitable's main interface's MAC address?

eth0 is the main interface.
MAC address: 08:00:27:1a:45:12

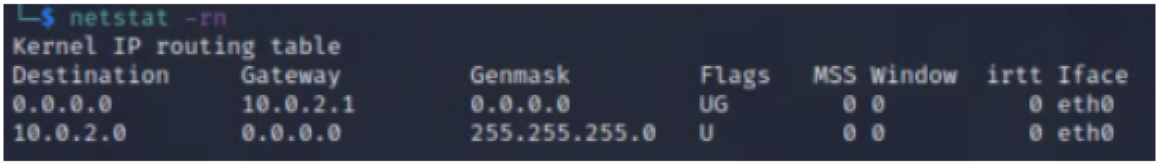


```
eth0      Link encap:Ethernet  HWaddr 08:00:27:1a:45:12
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe1a:4512/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:39 errors:0 dropped:0 overruns:0 frame:0
          TX packets:72 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6824 (6.6 KB)  TX bytes:8113 (7.9 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:134 errors:0 dropped:0 overruns:0 frame:0
          TX packets:134 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:40017 (39.0 KB)  TX bytes:40017 (39.0 KB)
```

D. What is Metasploitable's main interface's IP address?

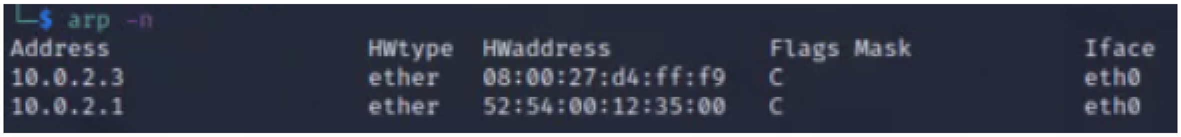
Private IP address: 10.0.2.4
Public IP address: 137.22.28.3

E. Show Kali's routing table. (Use "netstat -r" to see it with symbolic names, or "netstat -rn" to see it with numerical addresses.)



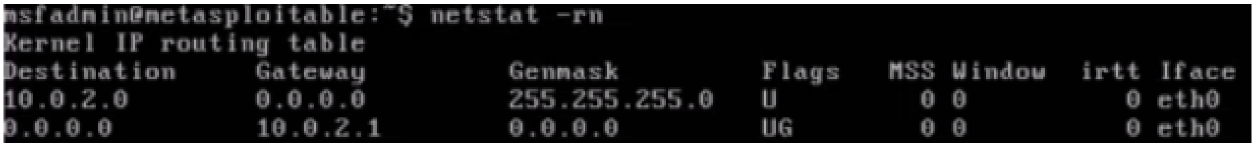
```
└─$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG        0 0          0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

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



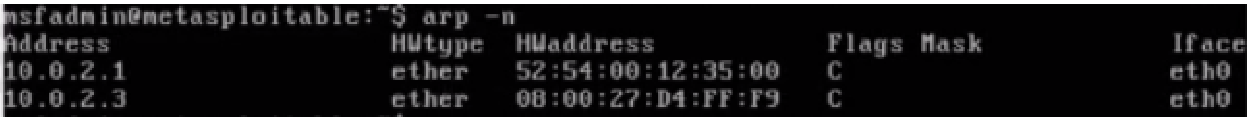
```
└─$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.3                 ether   08:00:27:d4:ff:f9   C                     eth0
10.0.2.1                 ether   52:54:00:12:35:00   C                     eth0
```

G. Show Metasploitable's routing table.



```
msfadmin@metasploitable:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
0.0.0.0         10.0.2.1        0.0.0.0         UG        0 0          0 eth0
```

H. Show Metasploitable's ARP cache.



```
msfadmin@metasploitable:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   52:54:00:12:35:00   C                     eth0
10.0.2.3                 ether   08:00:27:D4:FF:F9   C                     eth0
```

I. Suppose the user of Metasploitable wants to get the CS231 sandbox page via the command "curl http://cs231.jeffondich.com/". To which MAC address should Metasploitable send the TCP SYN packet to get the whole HTTP query started? Explain why.

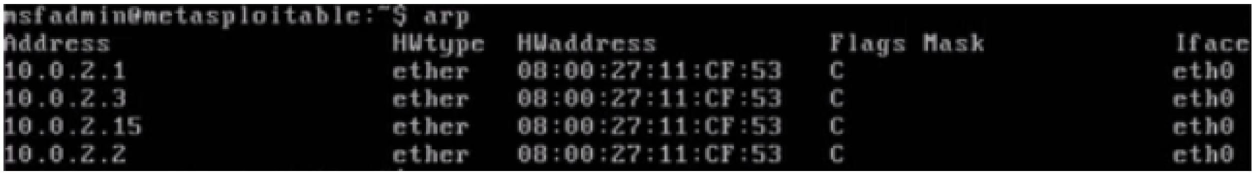
Metasploitable should send the TCP SYN packet to 52:54:00:12:35:00, the hardware address of the virtual network switch. Additionally, it looks like this hardware address is the same as the actual machine on which the two virtual machines are running (which in this case is one of the Olin remote labs).

J. Fire up Wireshark on Kali. Start capturing packets for "tcp port http". On Metasploitable, execute "curl http://cs231.jeffondich.com/". On Kali, stop capturing. Do you see an HTTP response on Metasploitable? Do you see any captured packets in Wireshark on Kali?

We see an HTTP response on Metasploitable. We don't see any captured packets on Kali.

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.

L. Show Metasploitable's ARP cache. How has it changed?



```
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   08:00:27:11:CF:53   C                     eth0
10.0.2.3                 ether   08:00:27:11:CF:53   C                     eth0
10.0.2.15                ether   08:00:27:11:CF:53   C                     eth0
10.0.2.2                 ether   08:00:27:11:CF:53   C                     eth0
```

We changed the ARP cache so that now the Metasploitable user will be first sending all of its network packets to the Kali machine.

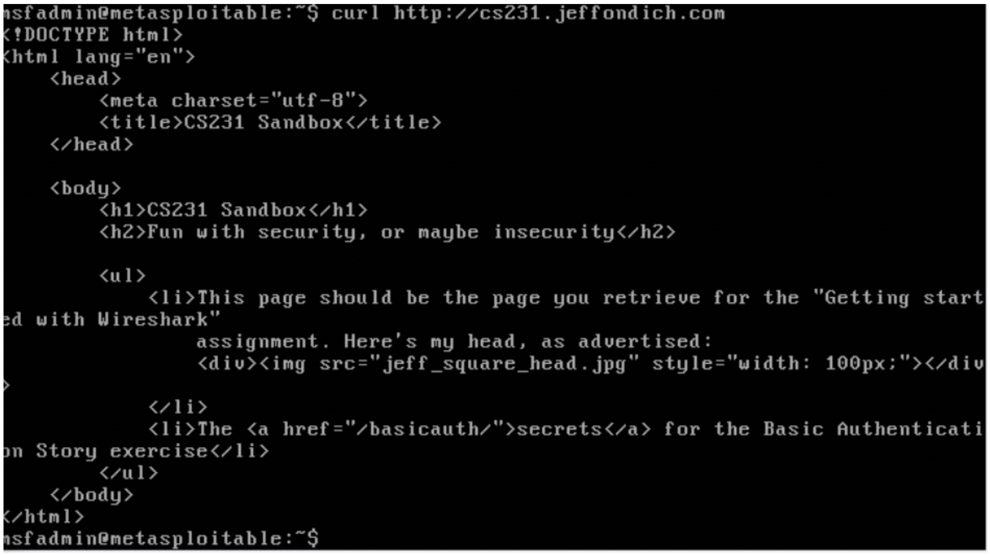
M. If you execute "curl http://cs231.jeffondich.com/" on Metasploitable now, to what MAC address will Metasploitable send the TCP SYN packet? Explain why.

Now, Metasploitable will send the TCP SYN packet to the Kali virtual machine because we poisoned the ARP cache in the Metasploitable machine to show that the Kali machine's MAC address was the correct hardware address for all of the IP addresses in the Metasploitable computer's ARP cache.

N. Start Wireshark capturing "tcp port http" again.

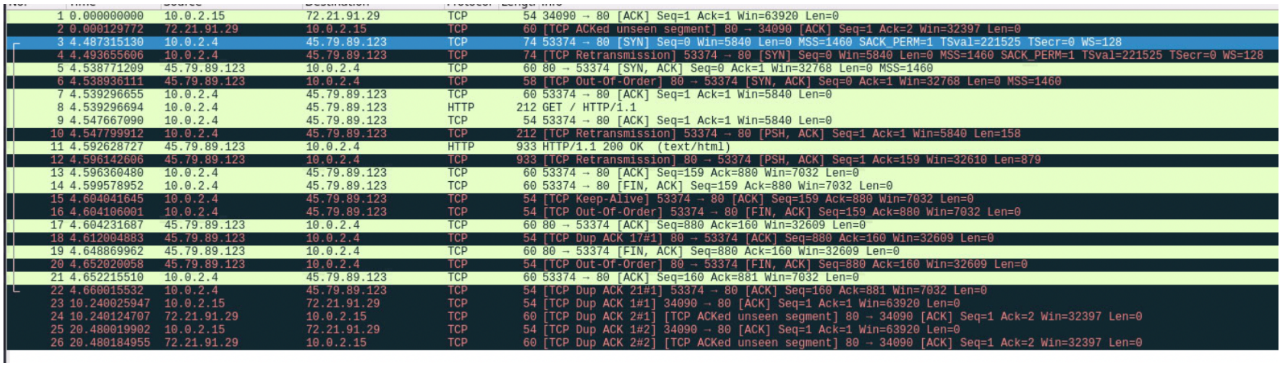
O. Execute "curl http://cs231.jeffondich.com/" on Metasploitable. On Kali, stop capturing. Do you see an HTTP response on Metasploitable? Do you see captured packets in Wireshark? Can you tell from Kali what messages went back and forth between Metasploitable and cs231.jeffondich.com?

We see an HTTP response on Metasploitable. It looks identical to the HTTP response that we received before our ARP cache was poisoned.



```
sfadmin@metasploitable:~$ curl http://cs231.jeffondich.com
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>CS231 Sandbox</title>
    </head>

    <body>
        <h1>CS231 Sandbox</h1>
        <h2>Fun with security, or maybe insecurity</h2>

        <ul>
            <li>This page should be the page you retrieve for the "Getting start
ed with Wireshark"
                assignment. Here's my head, as advertised:
                <div><img src="jeff_square_head.jpg" style="width: 100px;"></div
>
            </li>
            <li>The <a href="/basicauth/">secrets</a> for the Basic Authenticati
on Story exercise</li>
        </ul>
    </body>
</html>
sfadmin@metasploitable:~$
```

We also see the captured packets in Wireshark. We can definitely see the messages that went back and forth between Metasploitable and cs231.jeffondich.com.



```
 1 0.000000000  10.0.2.15      72.21.91.29    TCP    54 34090 → 80 [ACK] Seq=1 Ack=1 Win=63920 Len=0
 2 0.000129772  72.21.91.29    10.0.2.15      TCP    60 [TCP ACKed unseen segment] 80 → 34090 [ACK] Seq=1 Ack=2 Win=32397 Len=0
 3 4.487315130  10.0.2.4       45.79.89.123   TCP    74 53374 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=221525 TSecr=0 WS=128
 4 4.493655606  10.0.2.4       45.79.89.123   TCP    74 [TCP Retransmission] 53374 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=221525 TSecr=0 WS=128
 5 4.538771209  45.79.89.123   10.0.2.4       TCP    60 80 → 53374 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
 6 4.538936111  45.79.89.123   10.0.2.4       TCP    58 [TCP Out-Of-Order] 80 → 53374 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
 7 4.539296655  10.0.2.4       45.79.89.123   TCP    60 53374 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
 8 4.539296694  10.0.2.4       45.79.89.123   HTTP  212 GET / HTTP/1.1
 9 4.547667090  10.0.2.4       45.79.89.123   TCP    54 53374 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0
10 4.547799912  10.0.2.4       45.79.89.123   TCP   212 [TCP Retransmission] 53374 → 80 [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=158
11 4.592628727  45.79.89.123   10.0.2.4       HTTP  933 HTTP/1.1 200 OK  (text/html)
12 4.596142606  45.79.89.123   10.0.2.4       TCP   933 [TCP Retransmission] 80 → 53374 [PSH, ACK] Seq=1 Ack=159 Win=32610 Len=879
13 4.596360480  10.0.2.4       45.79.89.123   TCP    60 53374 → 80 [ACK] Seq=159 Ack=880 Win=7032 Len=0
14 4.599578952  10.0.2.4       45.79.89.123   TCP    60 53374 → 80 [FIN, ACK] Seq=159 Ack=880 Win=7032 Len=0
15 4.604041645  10.0.2.4       45.79.89.123   TCP    54 [TCP Keep-Alive] 53374 → 80 [ACK] Seq=159 Ack=880 Win=7032 Len=0
16 4.604106001  10.0.2.4       45.79.89.123   TCP    54 [TCP Out-Of-Order] 53374 → 80 [FIN, ACK] Seq=159 Ack=880 Win=7032 Len=0
17 4.604231687  45.79.89.123   10.0.2.4       TCP    60 80 → 53374 [ACK] Seq=880 Ack=160 Win=32609 Len=0
18 4.612004883  45.79.89.123   10.0.2.4       TCP    54 [TCP Dup ACK 17#1] 80 → 53374 [ACK] Seq=880 Ack=160 Win=32609 Len=0
19 4.648869962  45.79.89.123   10.0.2.4       TCP    60 80 → 53374 [FIN, ACK] Seq=880 Ack=160 Win=32609 Len=0
20 4.652020058  45.79.89.123   10.0.2.4       TCP    54 [TCP Out-Of-Order] 80 → 53374 [FIN, ACK] Seq=880 Ack=160 Win=32609 Len=0
21 4.652215510  10.0.2.4       45.79.89.123   TCP    60 53374 → 80 [ACK] Seq=160 Ack=881 Win=7032 Len=0
22 4.660015532  10.0.2.4       45.79.89.123   TCP    54 [TCP Dup ACK 21#1] 53374 → 80 [ACK] Seq=160 Ack=881 Win=7032 Len=0
23 10.240025947 10.0.2.15      72.21.91.29    TCP    54 [TCP Dup ACK 1#1] 34090 → 80 [ACK] Seq=1 Ack=1 Win=63920 Len=0
24 10.240124707 72.21.91.29    10.0.2.15      TCP    60 [TCP Dup ACK 2#1] [TCP ACKed unseen segment] 80 → 34090 [ACK] Seq=1 Ack=2 Win=32397 Len=0
25 20.480019902 10.0.2.15      72.21.91.29    TCP    54 [TCP Dup ACK 1#2] 34090 → 80 [ACK] Seq=1 Ack=1 Win=63920 Len=0
26 20.480184955 72.21.91.29    10.0.2.15      TCP    60 [TCP Dup ACK 2#2] [TCP ACKed unseen segment] 80 → 34090 [ACK] Seq=1 Ack=2 Win=32397 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.)

From this article by the GRC, (https://www.grc.com/nat/arp.htm), we determined that when the Metasploitable machine begins its HTTP request, it first sends out and ARP request for the IP associated with http://cs231.jeffondich.com to figure out where to direct its packet. Kali maliciously responded to these ARP requests and changed Metasploitable's ARP cache by responding with a “fake” ARP reply. This fake ARP reply contained the false information that the Kali machine’s hardware address was associated with the IP address that the ARP request was sent for. Thus, instead of directing its HTTP packet to the virtual network switch (or the router in a more typical scenario), the Metasploitable machine directed its packet first to the Kali machine.

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.)

First, we could keep a table of historical MAC addresses and what IP addresses they are associated with. If we ever observe more than one IP address associated with a MAC address, this might key us into the fact that something is wrong. A false positive could be a situation in which a load balancing server has multiple redundant internet connections and/or a datacenter uses different connections (and hence different IP addresses) for different services on a computer.