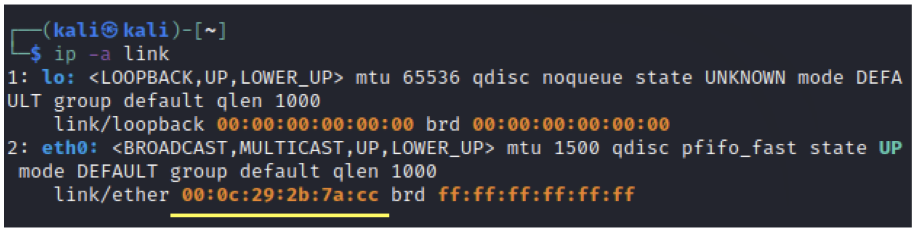
Arp-spoofing

Author: Lily Li
Conferred with: Jeff Ondich

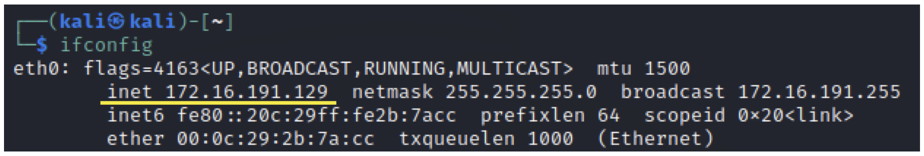
Sources: https://www.javatpoint.com/tcp-retransmission
https://www.imperva.com/learn/application-security/arp-spoofing/
https://www.reddit.com/r/networking/comments/1umrp7/multiple_ip_address_on_one_mac_address/

A: Kali main interface Mac address: **00:0c:29:2b:7a:cc**



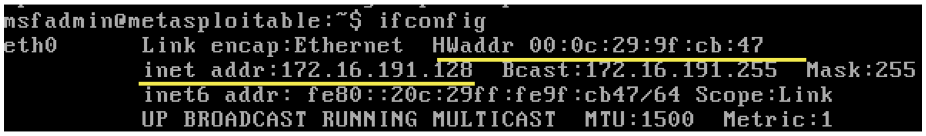
```
┌──(kali㉿kali)-[~]
└─$ ip -a link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFA
ULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
 mode DEFAULT group default qlen 1000
    link/ether 00:0c:29:2b:7a:cc brd ff:ff:ff:ff:ff:ff
```

B: Kali's main interface's IP address: **172.16.191.129**



```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.191.129  netmask 255.255.255.0  broadcast 172.16.191.255
        inet6 fe80::20c:29ff:fe2b:7acc  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:2b:7a:cc  txqueuelen 1000  (Ethernet)
```

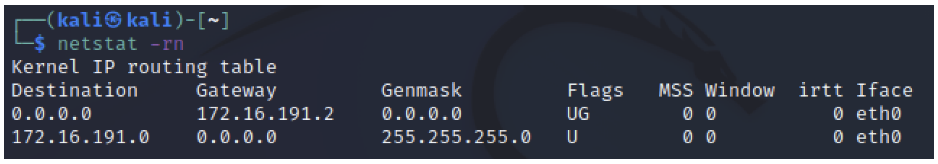
C: Metasploitable main interface Mac address: **00:0c:29:9f:cb:47**



```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:9f:cb:47
          inet addr:172.16.191.128  Bcast:172.16.191.255  Mask:255
          inet6 addr: fe80::20c:29ff:fe9f:cb47/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

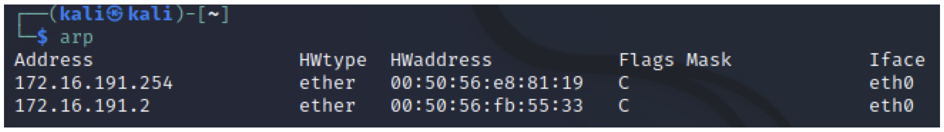
D: Metasploitable main interface IP address: **172.16.191.128**

E: Kali routing table



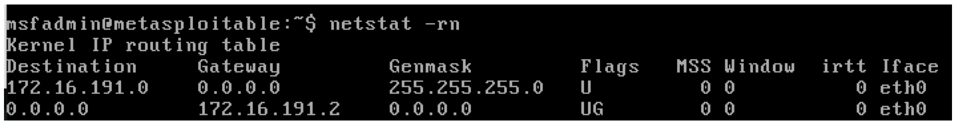
```
┌──(kali㉿kali)-[~]
└─$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         172.16.191.2    0.0.0.0         UG        0 0          0 eth0
172.16.191.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

F: Kali ARP cache

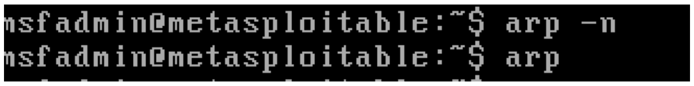


```
┌──(kali㉿kali)-[~]
└─$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
172.16.191.254           ether   00:50:56:e8:81:19   C                     eth0
172.16.191.2             ether   00:50:56:fb:55:33   C                     eth0
```

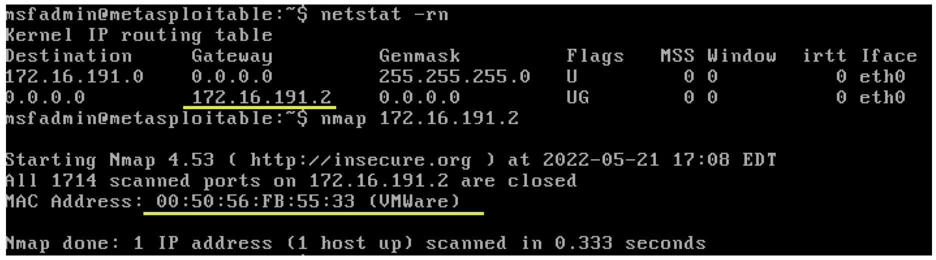
G: Metasploitable routing table

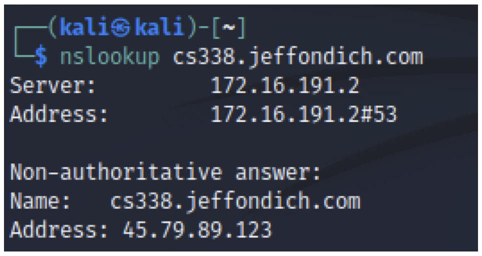

```
msfadmin@metasploitable:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
172.16.191.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
0.0.0.0         172.16.191.2    0.0.0.0         UG        0 0          0 eth0
```

H: Metasploitable ARP cache (currently no ARP cache)

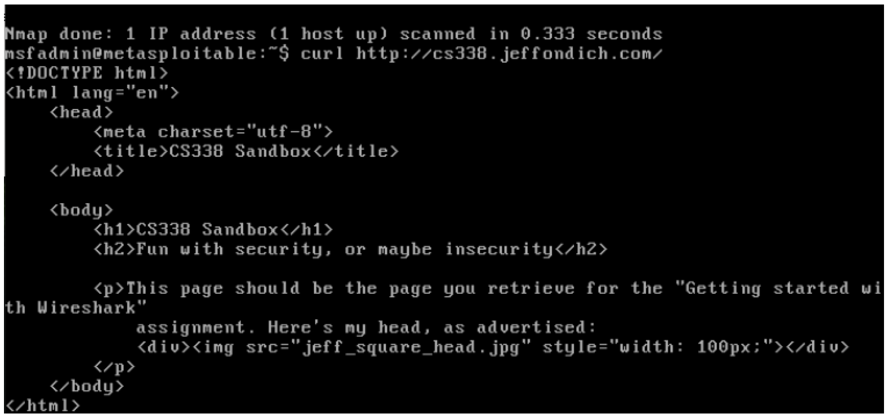

```
msfadmin@metasploitable:~$ arp -n
msfadmin@metasploitable:~$ arp
```

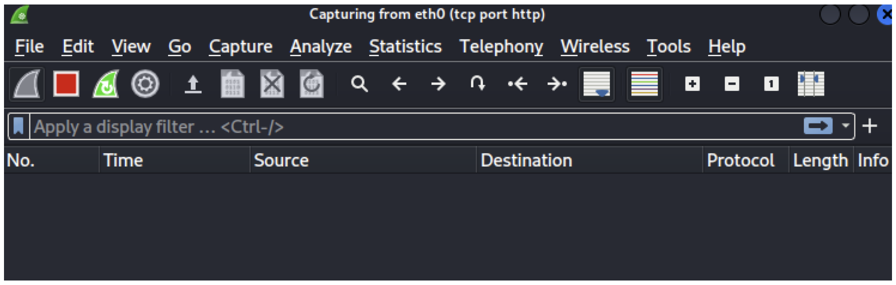
I: It should send to MAC address **00:50:56:FB:55:33**
Looking at the routing table, we can see that the destination *cs338.jeffondich.com* (45.79.89.123) is not on the destination. Therefore the packet will go through the default gateway 172.16.191.2 which has the MAC address **00:50:56:FB:55:33**


```
msfadmin@metasploitable:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
172.16.191.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
0.0.0.0         172.16.191.2    0.0.0.0         UG        0 0          0 eth0
msfadmin@metasploitable:~$ nmap 172.16.191.2

Starting Nmap 4.53 ( http://insecure.org ) at 2022-05-21 17:08 EDT
All 1714 scanned ports on 172.16.191.2 are closed
MAC Address: 00:50:56:FB:55:33 (VMWare)

Nmap done: 1 IP address (1 host up) scanned in 0.333 seconds
```

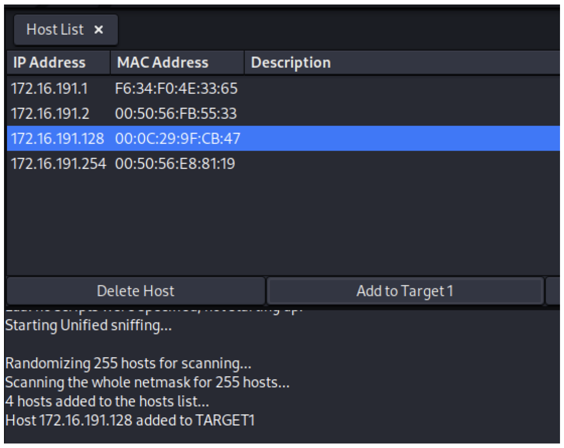

```
┌──(kali㉿kali)-[~]
└─$ nslookup cs338.jeffondich.com
Server:         172.16.191.2
Address:        172.16.191.2#53

Non-authoritative answer:
Name:   cs338.jeffondich.com
Address: 45.79.89.123
```

j: HTTP response is seen on Metasploitable, but there are no captured packets on Wireshark.


```
Nmap done: 1 IP address (1 host up) scanned in 0.333 seconds
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>
```

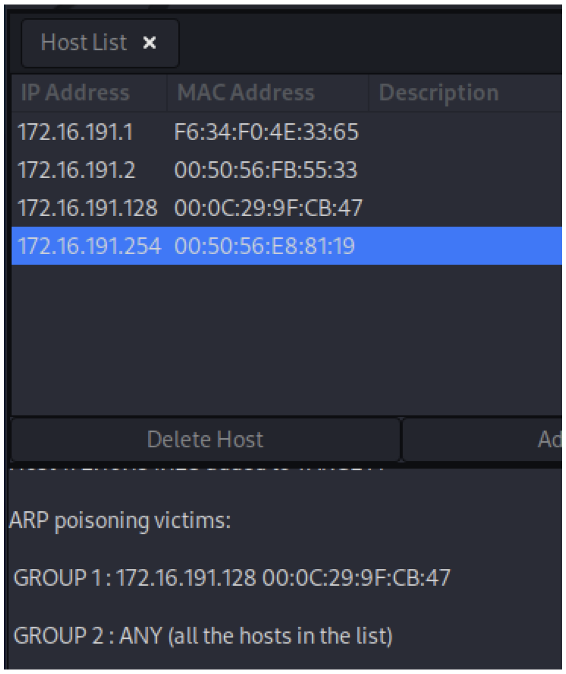

Capturing from eth0 (tcp port http)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
No. Time Source Destination Protocol Length Info

K: Select host on Ettercap and begin ARP poisoning



IP Address	MAC Address	Description
172.16.191.1	F6:34:F0:4E:33:65	
172.16.191.2	00:50:56:FB:55:33	
172.16.191.128	00:0C:29:9F:CB:47	
172.16.191.254	00:50:56:E8:81:19	

```
Starting Unified sniffing...

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
4 hosts added to the hosts list...
Host 172.16.191.128 added to TARGET1
```

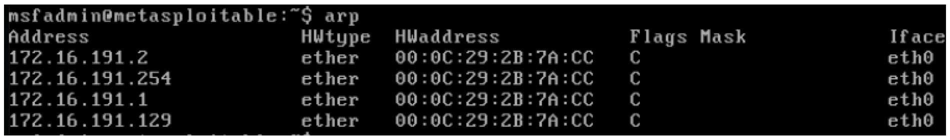


IP Address	MAC Address	Description
172.16.191.1	F6:34:F0:4E:33:65	
172.16.191.2	00:50:56:FB:55:33	
172.16.191.128	00:0C:29:9F:CB:47	
172.16.191.254	00:50:56:E8:81:19	

```
ARP poisoning victims:

 GROUP 1 : 172.16.191.128 00:0C:29:9F:CB:47

 GROUP 2 : ANY (all the hosts in the list)
```

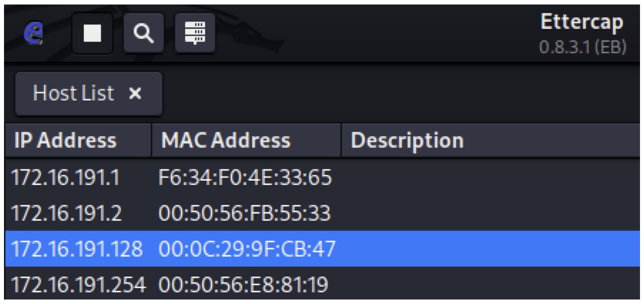
L: Metasploitable ARP cache changes:
All hosts in the same subnet (including Kali) are added to the ARP cache
However, all of their MAC addresses are the same!

00:0c:29:2b:7a:cc (MAC address of Kali)



```
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
172.16.191.2             ether   00:0C:29:2B:7A:CC   C                     eth0
172.16.191.254           ether   00:0C:29:2B:7A:CC   C                     eth0
172.16.191.1             ether   00:0C:29:2B:7A:CC   C                     eth0
172.16.191.129           ether   00:0C:29:2B:7A:CC   C                     eth0
```

In comparison,
we can see the correct Mac addresses of the hosts on Ettercap's host list



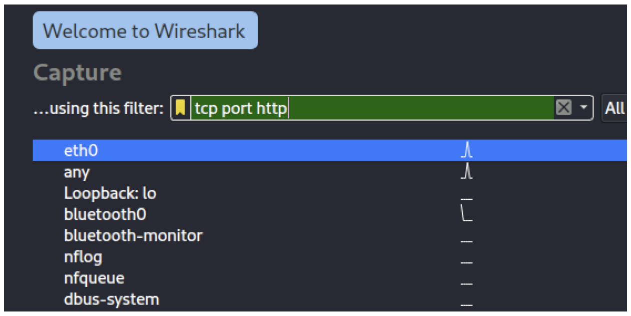
IP Address	MAC Address	Description
172.16.191.1	F6:34:F0:4E:33:65	
172.16.191.2	00:50:56:FB:55:33	
172.16.191.128	00:0C:29:9F:CB:47	
172.16.191.254	00:50:56:E8:81:19	

M:
I believe that when you execute *"curl http://cs338.jeffondich.com/"* right now, we will be able to capture the packet on Wireshark, because the packet will be sent to the Mac address **00:0c:29:2b:7a:cc**.

Just as in the previous question, the routing table does not have the destination for "*http://cs338.jeffondich.com/*" . Therefore, the packet will be sent to the default gateway *172.16.191.2*

Different from I however, now, the ARP cache does contain the MAC address for *172.16.191.2*. Therefore this time, instead of ARP broadcasting, Metasploitable will just send the packet to the address **00:0c:29:2b:7a:cc** according to the ARP cache, without investigating its authenticity. And the packet will end up at Kali.

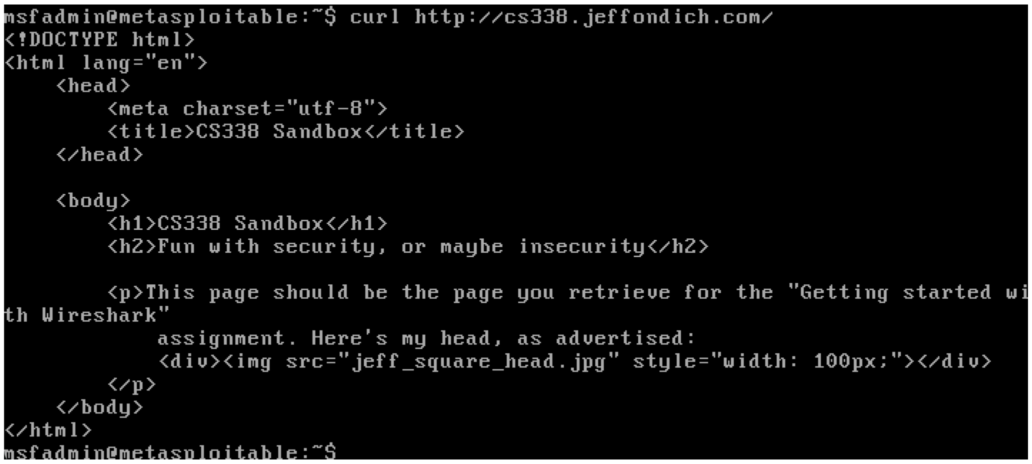
N:



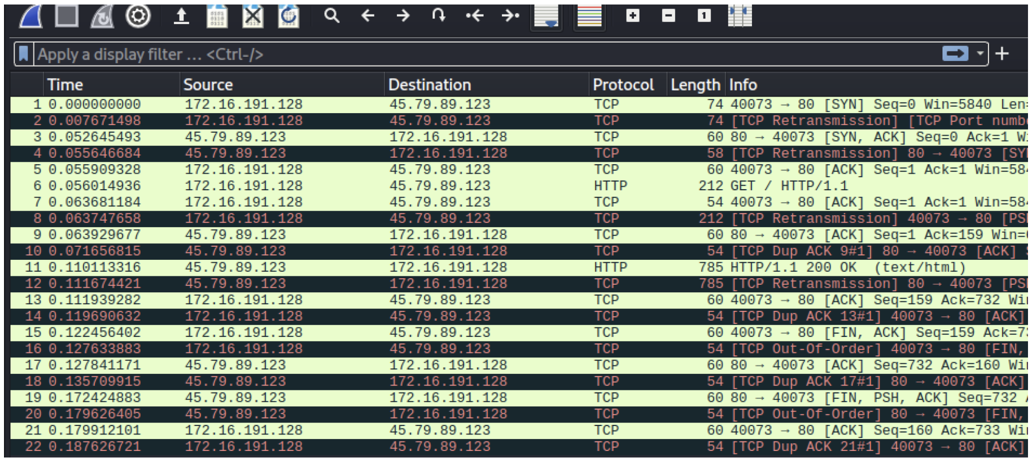
tcp port http

eth0
any
Loopback: lo
bluetooth0
bluetooth-monitor
nflog
nfqueue
dbus-system

O:
Yes, there are HTTP responses on Metasploitable.
Yes, there are packets in Wireshark.



```
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:~$
```



```
   Time         Source            Destination       Protocol  Length Info
 1 0.000000000  172.16.191.128    45.79.89.123      TCP         74 40073 → 80 [SYN] Seq=0 Win=5840 Len
 2 0.007671498  172.16.191.128    45.79.89.123      TCP         74 [TCP Retransmission] [TCP Port numb
 3 0.052645493  45.79.89.123      172.16.191.128    TCP         60 80 → 40073 [SYN, ACK] Seq=0 Ack=1 W
 4 0.055646684  45.79.89.123      172.16.191.128    TCP         58 [TCP Retransmission] 80 → 40073 [SY
 5 0.055909328  172.16.191.128    45.79.89.123      TCP         60 40073 → 80 [ACK] Seq=1 Ack=1 Win=58
 6 0.056014936  172.16.191.128    45.79.89.123      HTTP       212 GET / HTTP/1.1
 7 0.063681184  172.16.191.128    45.79.89.123      TCP         54 40073 → 80 [ACK] Seq=1 Ack=1 Win=58
 8 0.063747658  172.16.191.128    45.79.89.123      TCP        212 [TCP Retransmission] 40073 → 80 [PS
 9 0.063929677  45.79.89.123      172.16.191.128    TCP         60 80 → 40073 [ACK] Seq=1 Ack=159 Win=
10 0.071656815  45.79.89.123      172.16.191.128    TCP         54 [TCP Dup ACK 9#1] 80 → 40073 [ACK]
11 0.110113316  45.79.89.123      172.16.191.128    HTTP       785 HTTP/1.1 200 OK  (text/html)
12 0.111674421  45.79.89.123      172.16.191.128    TCP        785 [TCP Retransmission] 80 → 40073 [PS
13 0.111939282  172.16.191.128    45.79.89.123      TCP         60 40073 → 80 [ACK] Seq=159 Ack=732 Wi
14 0.119690632  172.16.191.128    45.79.89.123      TCP         54 [TCP Dup ACK 13#1] 40073 → 80 [ACK]
15 0.122456402  172.16.191.128    45.79.89.123      TCP         60 40073 → 80 [FIN, ACK] Seq=159 Ack=7
16 0.127633883  172.16.191.128    45.79.89.123      TCP         54 [TCP Out-Of-Order] 40073 → 80 [FIN,
17 0.127841171  45.79.89.123      172.16.191.128    TCP         60 80 → 40073 [ACK] Seq=732 Ack=160 Wi
18 0.135709915  45.79.89.123      172.16.191.128    TCP         54 [TCP Dup ACK 17#1] 80 → 40073 [ACK]
19 0.172424883  45.79.89.123      172.16.191.128    TCP         60 80 → 40073 [FIN, PSH, ACK] Seq=732
20 0.179626405  45.79.89.123      172.16.191.128    TCP         54 [TCP Out-Of-Order] 80 → 40073 [FIN,
21 0.179912101  172.16.191.128    45.79.89.123      TCP         60 40073 → 80 [ACK] Seq=160 Ack=733 Wi
22 0.187626721  172.16.191.128    45.79.89.123      TCP         54 [TCP Dup ACK 21#1] 40073 → 80 [ACK]
```

Messages in between Metasploitable and *cs338.jeffondich.com*:

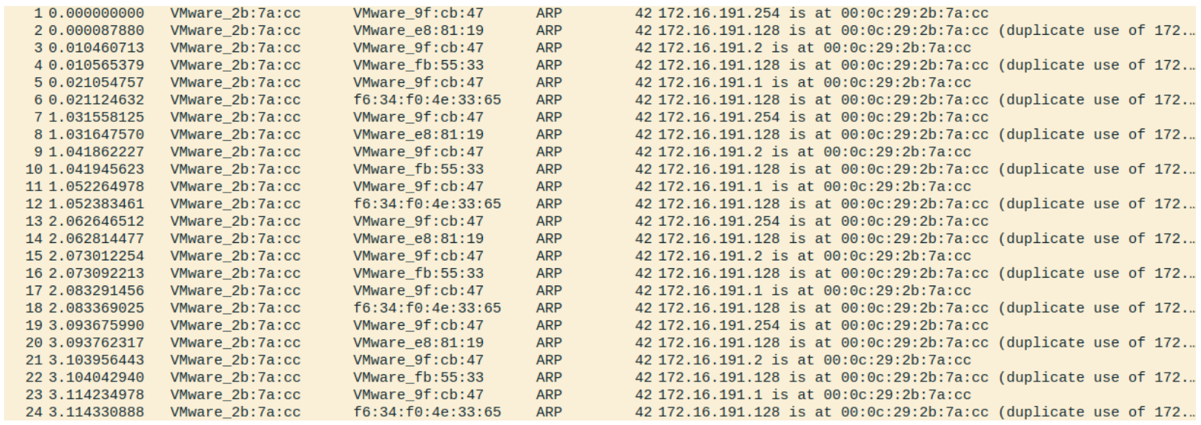
1: Metasploitable initiates a TCP handshake with *cs338.jeffondich.com* for initiating connection, and is successfully established
2: Metasploitable sends a HTTP Get request to *cs338.jeffondich.com*, requiring the webpage HTML.
3: *cs338.jeffondich.com* replies with a HTTP response to Metasploitable with status code 200 (ok), along with the HTML code of the webpage.
4: Metasploitable initiates a TCP process to terminate connection, connection terminated.

Note: It is interesting that there are a lot of *[TCP Retransmission]* and some *[TCP Dup],* which we have not seen in previous assignments when we do similar processes. The

[TCP Retransmission] is triggered when there is a lack of acknowledgement of a packet after a certain time. And *[TCP Dup]* is triggered when packets are received out of order. Therefore we can deduce that the interference of Ettercap slows the packet transmission down and might also be messing up the order of some of these packets.

P: What happened in ARP poisoning

To every host in the subnet, Ettercap sends forged ARP responses that say the MAC address of the attacker (Ettercap) links to the IP address of the victim. In Ettercap, it attacked all hosts in the subnet, not just the target victim Metasploitable. As Jeff explained in Discord, through this method, Ettercap will be able to intercept all information coming in and out of the victim.



```
 1 0.000000000   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.254 is at 00:0c:29:2b:7a:cc
 2 0.000087880   VMware_2b:7a:cc    VMware_e8:81:19    ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
 3 0.010460713   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.2 is at 00:0c:29:2b:7a:cc
 4 0.010565379   VMware_2b:7a:cc    VMware_fb:55:33    ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
 5 0.021054757   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.1 is at 00:0c:29:2b:7a:cc
 6 0.021124632   VMware_2b:7a:cc    f6:34:f0:4e:33:65  ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
 7 1.031558125   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.254 is at 00:0c:29:2b:7a:cc
 8 1.031647570   VMware_2b:7a:cc    VMware_e8:81:19    ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
 9 1.041862227   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.2 is at 00:0c:29:2b:7a:cc
10 1.041945623   VMware_2b:7a:cc    VMware_fb:55:33    ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
11 1.052264978   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.1 is at 00:0c:29:2b:7a:cc
12 1.052383461   VMware_2b:7a:cc    f6:34:f0:4e:33:65  ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
13 2.062646512   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.254 is at 00:0c:29:2b:7a:cc
14 2.062814477   VMware_2b:7a:cc    VMware_e8:81:19    ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
15 2.073012254   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.2 is at 00:0c:29:2b:7a:cc
16 2.073092213   VMware_2b:7a:cc    VMware_fb:55:33    ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
17 2.083291456   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.1 is at 00:0c:29:2b:7a:cc
18 2.083369025   VMware_2b:7a:cc    f6:34:f0:4e:33:65  ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
19 3.093675990   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.254 is at 00:0c:29:2b:7a:cc
20 3.093762317   VMware_2b:7a:cc    VMware_e8:81:19    ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
21 3.103956443   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.2 is at 00:0c:29:2b:7a:cc
22 3.104042940   VMware_2b:7a:cc    VMware_fb:55:33    ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
23 3.114234978   VMware_2b:7a:cc    VMware_9f:cb:47    ARP      42 172.16.191.1 is at 00:0c:29:2b:7a:cc
24 3.114330888   VMware_2b:7a:cc    f6:34:f0:4e:33:65  ARP      42 172.16.191.128 is at 00:0c:29:2b:7a:cc (duplicate use of 172…
```

The vulnerability of the ARP protocol is that: the hosts in the network automatically accept and cache the ARP responses they receive, even if they have not sent out an ARP request to start with.

In this case, Ettercap floods the ARP cache of all hosts in the network, and maps all IP addresses on it to the MAC address of the attacker. *Just like what we see in problem L,* where all IP addresses in Metasploitable's ARP cache link to the same Mac address. Thus, directing all the traffic(packets) sent out by the victim to the attacker (Kali Ettercap).

As a result, the attacker(Ettercap) now can intercept, modify or even block communications to the legitimate MAC address to/from the victim.

Q: Detect ARP spoofing

1: Scan the ARP cache periodically, or prior to using information from it. If there are two or more (or even all) IP addresses in the table with the same MAC address, then there is probably ARP poisoning going on.
However, it is possible to have multiple IP addresses assigned to the same MAC address (NIC), for instance: a server which has multiple services, and each role/service has its own IP address. In these cases, false positives may occur.

2: Monitor ARP responses on the local-network. If there is a flood of ARP responses from the same source all of a sudden on the local network, without or with very limited numbers of ARP requests occurring previously, it might be that some source is attempting to do ARP poisoning.

2: Monitor ARP responses that a system receives. If there is ARP response received by the system which links an IP address already in the ARP cache to a new MAC address, then it may be ARP spoofing. But in cases such as at carleton, where people come and go in the local network with their personal devices, use VPN… it is very probable that the same IP address may be connected to different devices over time so false positives may occur.

4:?Detects network traffic in general?: This one may be a little bit ambiguous but as we discovered in question 0, ARP poisoning steers network trafficking, maybe causing packet time delays (*TCP Retransmission*) and packets arriving in the wrong order. So maybe detecting irregular network activity will also help discover ARP poisoning. But I can envision this method creating many false positives as networks can be unstable.