TESTING OF CYBERSECURITY TECHNOLOGIES WITH PARROT OS

Harrison Carranza

Miguel Bustamante

Hossein Rahemi

ECC 2022 14th Annual Enterprise Computing Community National Conference

Marist College, June 12 - 14, 2022

Agenda

INTRODUCTION

PROCEDURAL STEPS FOR:

- *FIREWALL*
- *MALWARE PROTECTION*
- *DNS FILTERING*

CONCLUSIONS

Introduction

Our project focuses on testing the Firewall application, Malware Protection, and DNS Filtering to verify if it accurately keeps the computer system safe from cyberattacks on ParrotOS

Using Nmap and Wireshark we study if Firewall is a reliable cybersecurity technology to have to protect a computer system

We discuss about Malware protection using Metasploit, Wireshark, and Nikto to test if Malware protection is a reliable cybersecurity technology to have to protect a computer system

We also discuss about DNS filtering where the tools VPN and AnonSurf are used to test/bypass it in order to study if DNS filtering is a reliable cybersecurity technology to have to protect a computer system

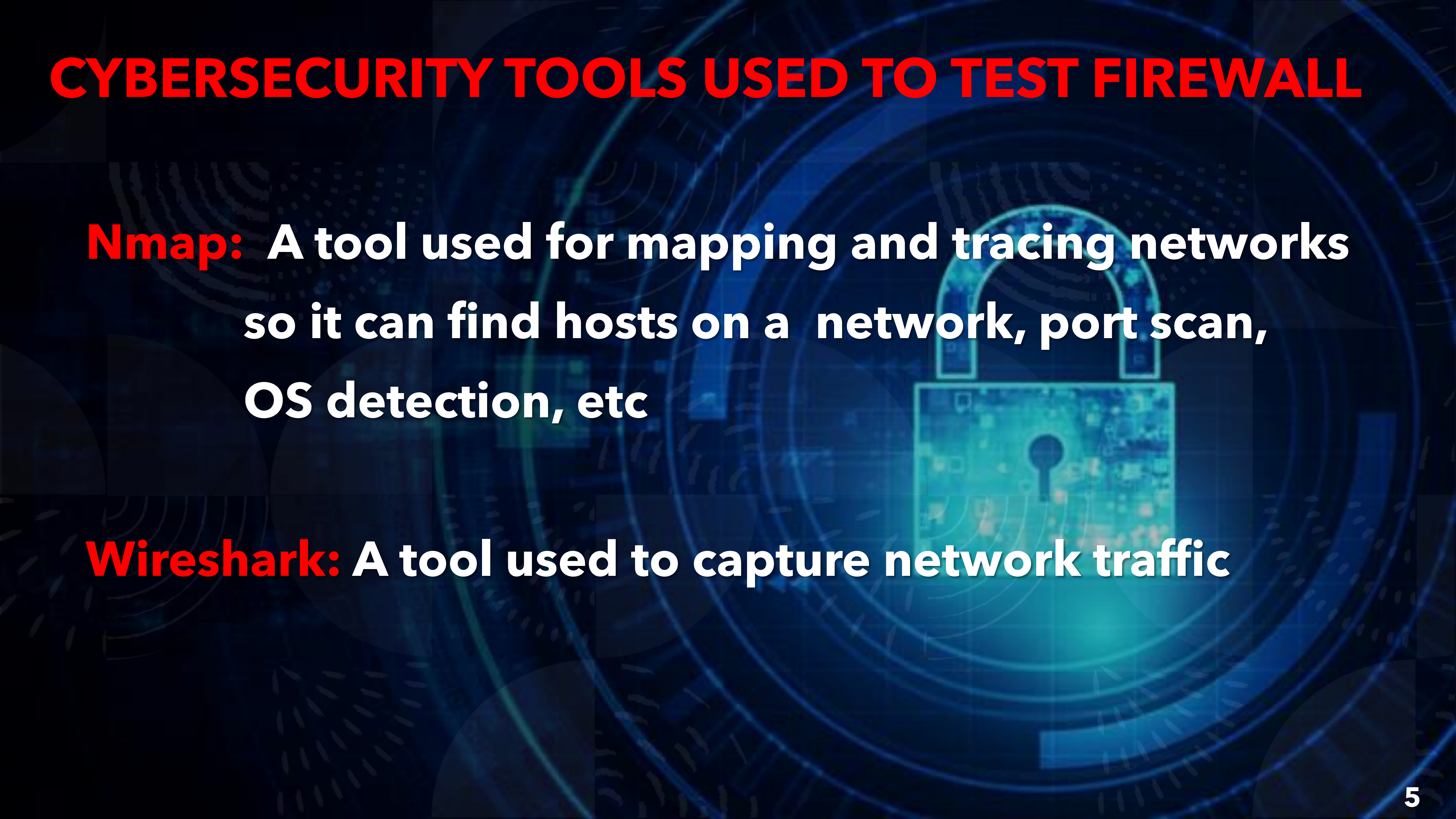
PROCEDURAL STEPS FOR: *FIREWALL*

A Firewall monitors network traffic and decides what should be blocked based on the security guidelines that comes with the firewall

It can be used to block data and ports while still being able to access safe data that is needed for the task

It can avoid sending responses to suspicious behaviors sent by hackers and detect any suspicious activity going on in the network which will notify the user

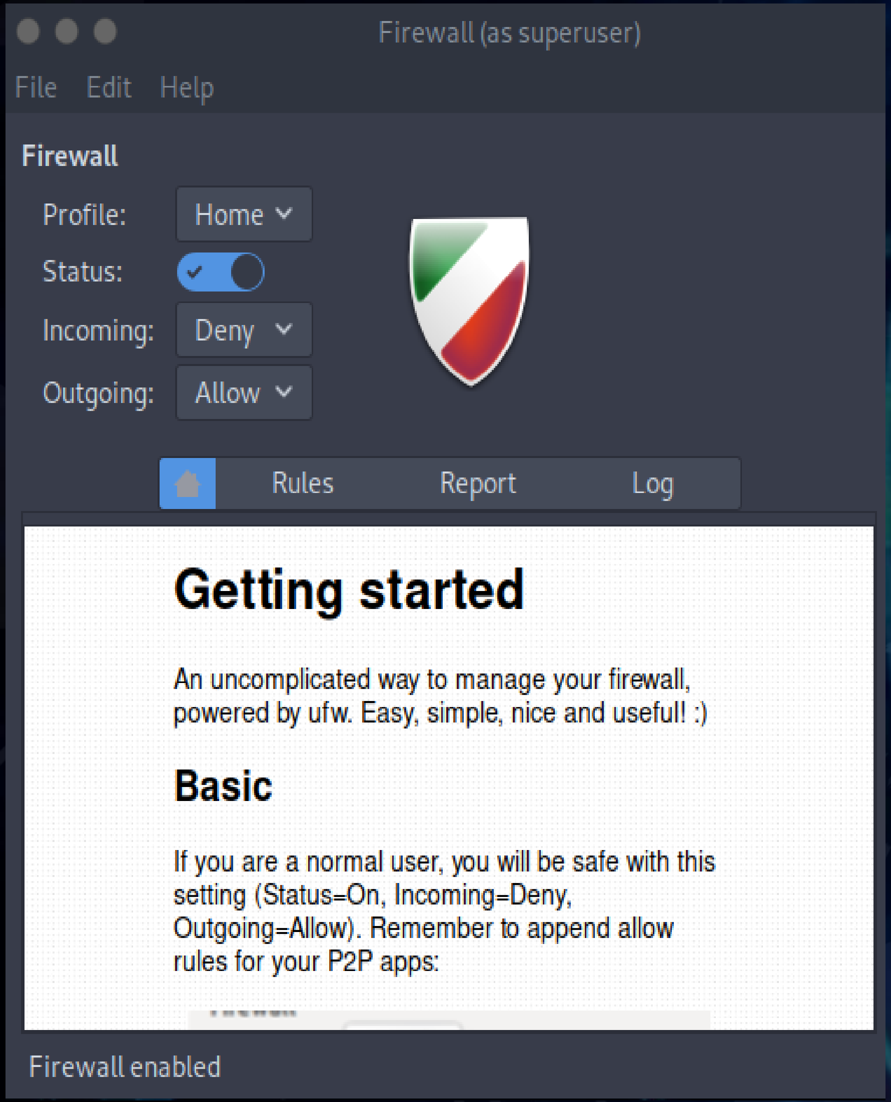
CYBERSECURITY TOOLS USED TO TEST FIREWALL



Nmap: A tool used for mapping and tracing networks so it can find hosts on a network, port scan, OS detection, etc

Wireshark: A tool used to capture network traffic

FIREWALL INSTALLATION



Firewall

Profile: Home

Status: ✓

Incoming: Deny

Outgoing: Allow

Getting started

An uncomplicated way to manage your firewall, powered by ufw. Easy, simple, nice and useful! :)

Basic

If you are a normal user, you will be safe with this setting (Status=On, Incoming=Deny, Outgoing=Allow). Remember to append allow rules for your P2P apps:

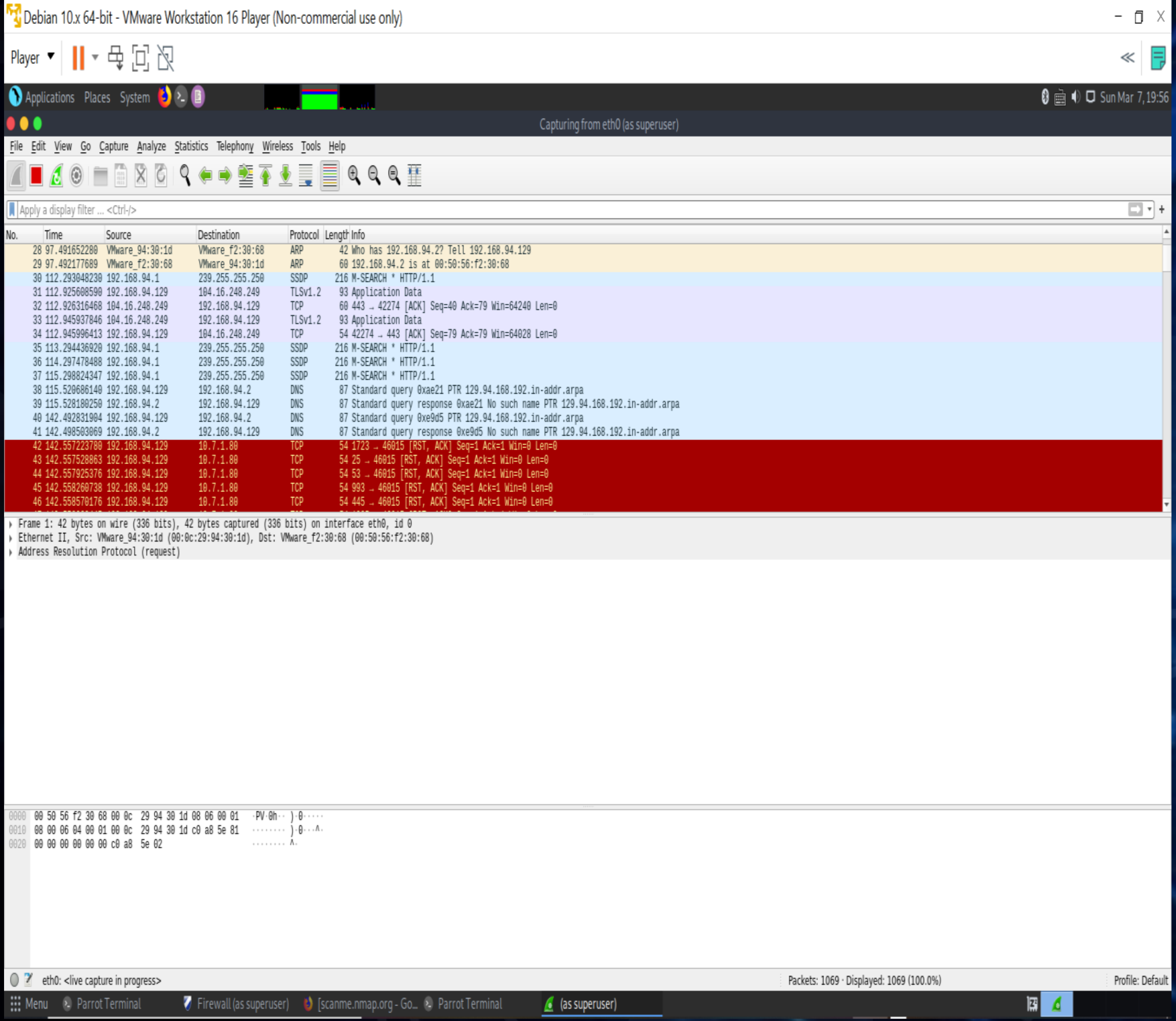
Firewall enabled

At the terminal as root, install the Firewall application by entering the command:

```
apt install gufw
```

Next, ran the *gufw* command, the Firewall Graphical Interface pops up where it is enabled

METHODOLOGY AND RESULTS


Debian 10.x 64-bit - VMware Workstation 16 Player (Non-commercial use only)
Player
Applications Places System
Sun Mar 7, 19:56
Capturing from eth0 (as superuser)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Menu
Parrot Terminal
Firewall (as superuser)
[scanme.nmap.org - Go...
Parrot Terminal
(as superuser)
Profile: Default

```
No.   Time          Source            Destination      Protocol Length Info
   28 97.491652280  VMware_94:30:1d   VMware_f2:30:68  ARP       42 Who has 192.168.94.2? Tell 192.168.94.129
   29 97.492177689  VMware_f2:30:68   VMware_94:30:1d  ARP       60 192.168.94.2 is at 00:50:56:f2:30:68
   30 112.293048230 192.168.94.1      239.255.255.250  SSDP     216 M-SEARCH * HTTP/1.1
   31 112.925608590 192.168.94.129    104.16.248.249   TLSv1.2   93 Application Data
   32 112.926316468 104.16.248.249    192.168.94.129   TCP       60 443 → 42274 [ACK] Seq=40 Ack=79 Win=64240 Len=0
   33 112.945937846 104.16.248.249    192.168.94.129   TLSv1.2   93 Application Data
   34 112.945996413 192.168.94.129    104.16.248.249   TCP       54 42274 → 443 [ACK] Seq=79 Ack=79 Win=64028 Len=0
   35 113.294436920 192.168.94.1      239.255.255.250  SSDP     216 M-SEARCH * HTTP/1.1
   36 114.297478488 192.168.94.1      239.255.255.250  SSDP     216 M-SEARCH * HTTP/1.1
   37 115.298824347 192.168.94.1      239.255.255.250  SSDP     216 M-SEARCH * HTTP/1.1
   38 115.520686140 192.168.94.129    192.168.94.2     DNS       87 Standard query 0xae21 PTR 129.94.168.192.in-addr.arpa
   39 115.528180250 192.168.94.2      192.168.94.129   DNS       87 Standard query response 0xae21 No such name PTR 129.94.168.192.in-addr.arpa
   40 142.492831904 192.168.94.129    192.168.94.2     DNS       87 Standard query 0xe9d5 PTR 129.94.168.192.in-addr.arpa
   41 142.498503069 192.168.94.2      192.168.94.129   DNS       87 Standard query response 0xe9d5 No such name PTR 129.94.168.192.in-addr.arpa
   42 142.557223780 192.168.94.129    10.7.1.80        TCP       54 1723 → 46015 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   43 142.557528863 192.168.94.129    10.7.1.80        TCP       54 25 → 46015 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   44 142.557925376 192.168.94.129    10.7.1.80        TCP       54 53 → 46015 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   45 142.558260738 192.168.94.129    10.7.1.80        TCP       54 993 → 46015 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   46 142.558570176 192.168.94.129    10.7.1.80        TCP       54 445 → 46015 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0
▸ Ethernet II, Src: VMware_94:30:1d (00:0c:29:94:30:1d), Dst: VMware_f2:30:68 (00:50:56:f2:30:68)
▸ Address Resolution Protocol (request)

0000  00 50 56 f2 30 68 00 0c  29 94 30 1d 08 06 00 01   ·PV·0h·· )·0·····
0010  08 00 06 04 00 01 00 0c  29 94 30 1d c0 a8 5e 81   ········ )·0···^·
0020  00 00 00 00 00 00 c0 a8  5e 02                     ········ ^·

eth0: <live capture in progress>        Packets: 1069 · Displayed: 1069 (100.0%)
```

First Method of Testing:

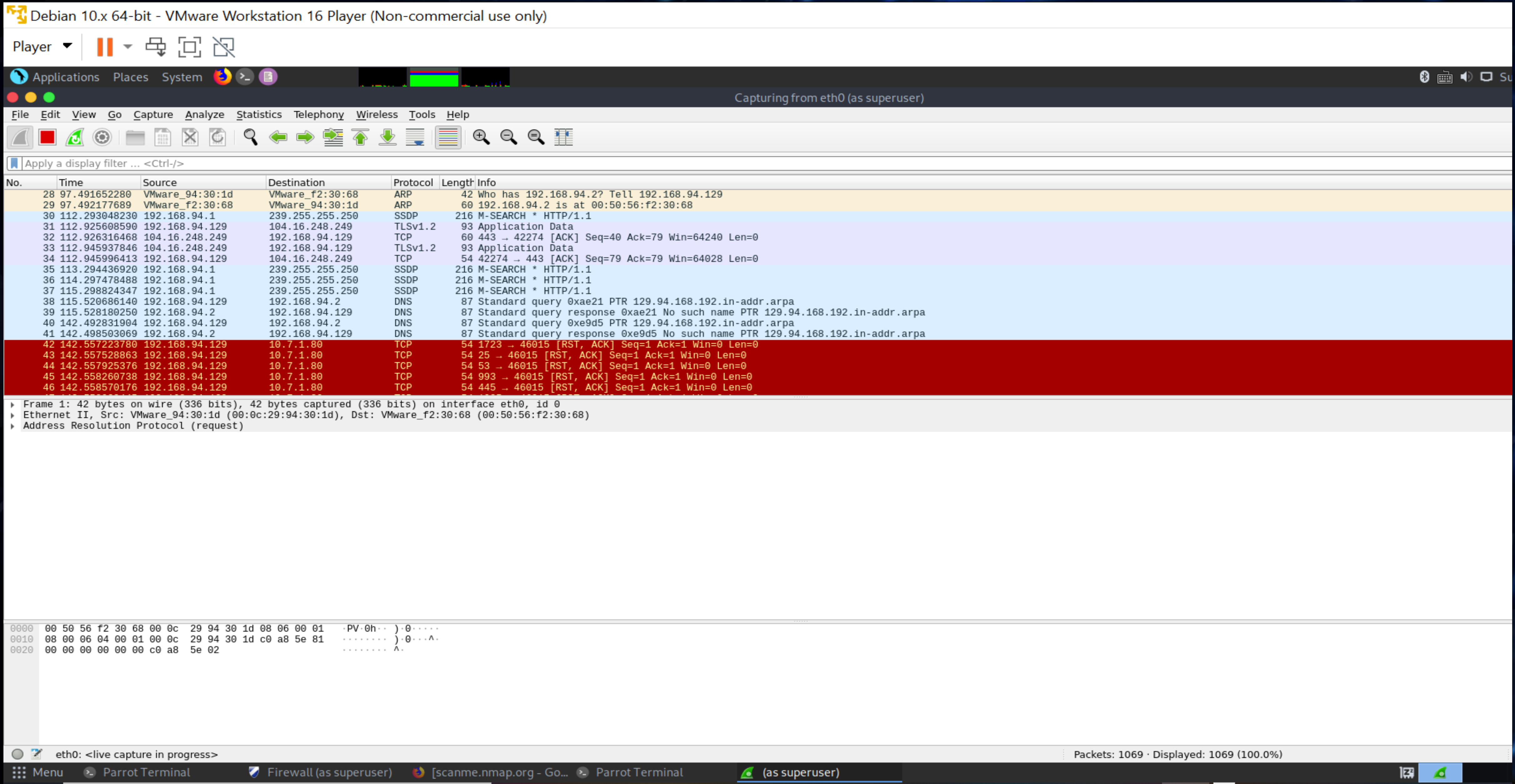
A SYN scan in stealth mode with a decoy IP address and a target IP address

(*nmap -sS -D 10.7.1.80 192.168.94.129*)

Wireshark demonstrated Firewall blocking the communication of the decoy IP address by sending an RST shown in red (*next slide*)

Firewall can detect and block false IP addresses

METHODOLOGY AND RESULTS



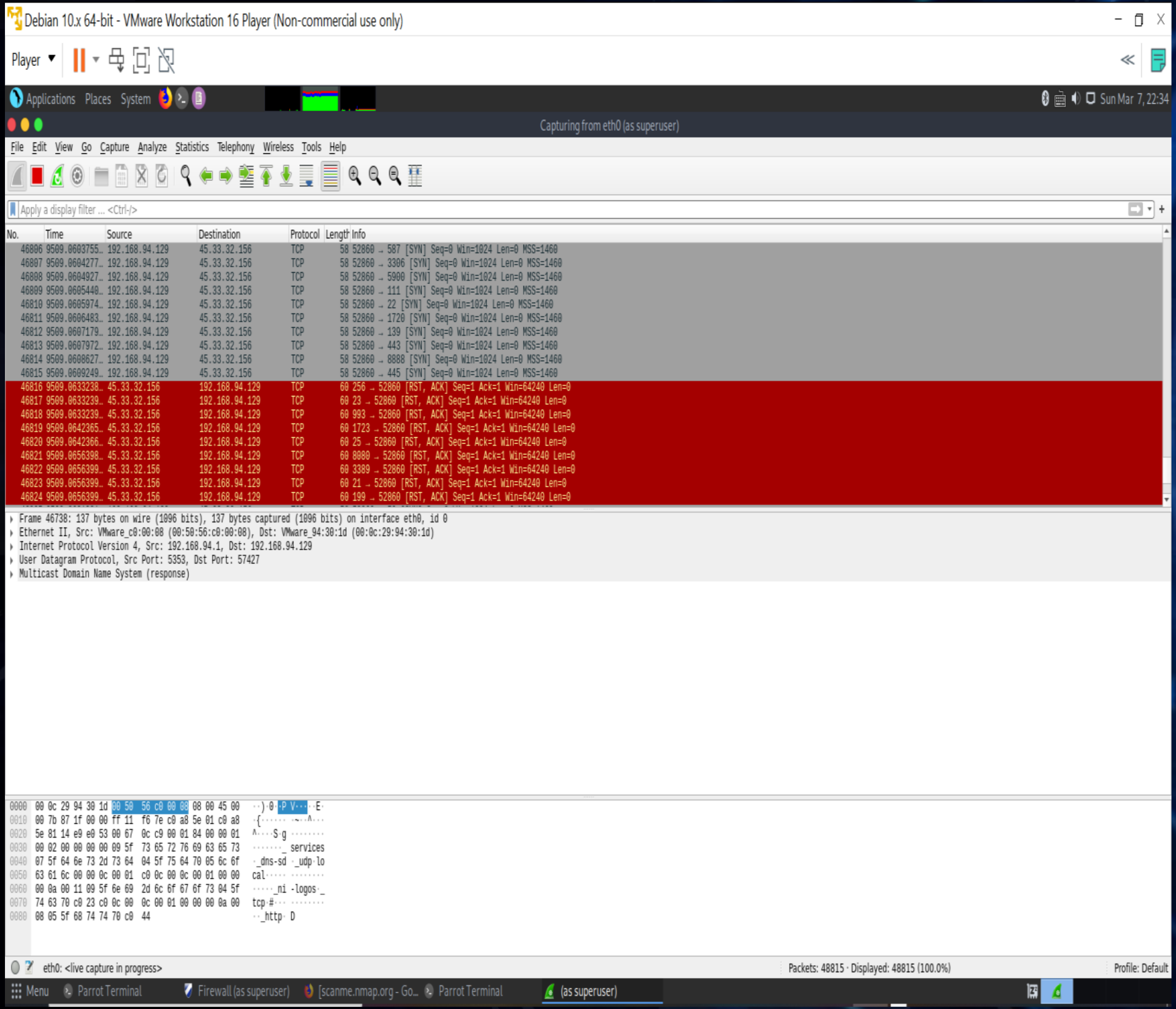
```
No.   Time            Source             Destination        Protocol Length Info
   28 97.491652280   VMware_94:30:1d    VMware_f2:30:68    ARP        42 Who has 192.168.94.2? Tell 192.168.94.129
   29 97.492177689   VMware_f2:30:68    VMware_94:30:1d    ARP        60 192.168.94.2 is at 00:50:56:f2:30:68
   30 112.293048230  192.168.94.1       239.255.255.250    SSDP      216 M-SEARCH * HTTP/1.1
   31 112.925608590  192.168.94.129     104.16.248.249     TLSv1.2    93 Application Data
   32 112.926316468  104.16.248.249     192.168.94.129     TCP        60 443 → 42274 [ACK] Seq=40 Ack=79 Win=64240 Len=0
   33 112.945937846  104.16.248.249     192.168.94.129     TLSv1.2    93 Application Data
   34 112.945996413  192.168.94.129     104.16.248.249     TCP        54 42274 → 443 [ACK] Seq=79 Ack=79 Win=64028 Len=0
   35 113.294436920  192.168.94.1       239.255.255.250    SSDP      216 M-SEARCH * HTTP/1.1
   36 114.297478488  192.168.94.1       239.255.255.250    SSDP      216 M-SEARCH * HTTP/1.1
   37 115.298824347  192.168.94.1       239.255.255.250    SSDP      216 M-SEARCH * HTTP/1.1
   38 115.520686140  192.168.94.129     192.168.94.2       DNS        87 Standard query 0xae21 PTR 129.94.168.192.in-addr.arpa
   39 115.528180250  192.168.94.2       192.168.94.129     DNS        87 Standard query response 0xae21 No such name PTR 129.94.168.192.in-addr.arpa
   40 142.492831904  192.168.94.129     192.168.94.2       DNS        87 Standard query 0xe9d5 PTR 129.94.168.192.in-addr.arpa
   41 142.498503069  192.168.94.2       192.168.94.129     DNS        87 Standard query response 0xe9d5 No such name PTR 129.94.168.192.in-addr.arpa
   42 142.557223780  192.168.94.129     10.7.1.80          TCP        54 1723 → 46015 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   43 142.557528863  192.168.94.129     10.7.1.80          TCP        54 25 → 46015 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   44 142.557925376  192.168.94.129     10.7.1.80          TCP        54 53 → 46015 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   45 142.558260738  192.168.94.129     10.7.1.80          TCP        54 993 → 46015 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   46 142.558570176  192.168.94.129     10.7.1.80          TCP        54 445 → 46015 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

```
▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0
▸ Ethernet II, Src: VMware_94:30:1d (00:0c:29:94:30:1d), Dst: VMware_f2:30:68 (00:50:56:f2:30:68)
▸ Address Resolution Protocol (request)
```

```
0000  00 50 56 f2 30 68 00 0c  29 94 30 1d 08 06 00 01   ·PV·0h·· )·0·····
0010  08 00 06 04 00 01 00 0c  29 94 30 1d c0 a8 5e 81   ········ )·0···^·
0020  00 00 00 00 00 00 c0 a8  5e 02                     ········ ^·
```

eth0: <live capture in progress>   Packets: 1069 · Displayed: 1069 (100.0%)

METHODOLOGY AND RESULTS


Debian 10.x 64-bit - VMware Workstation 16 Player (Non-commercial use only)
Player
Applications Places System
Sun Mar 7, 22:34
Capturing from eth0 (as superuser)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Menu
Parrot Terminal
Firewall (as superuser)
[scanme.nmap.org - Go…
Parrot Terminal
(as superuser)

```
No.   Time          Source          Destination     Protocol Length Info
46806 9509.0603755… 192.168.94.129  45.33.32.156    TCP      58 52860 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46807 9509.0604277… 192.168.94.129  45.33.32.156    TCP      58 52860 → 3306 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46808 9509.0604927… 192.168.94.129  45.33.32.156    TCP      58 52860 → 5900 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46809 9509.0605440… 192.168.94.129  45.33.32.156    TCP      58 52860 → 111 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46810 9509.0605974… 192.168.94.129  45.33.32.156    TCP      58 52860 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46811 9509.0606483… 192.168.94.129  45.33.32.156    TCP      58 52860 → 1720 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46812 9509.0607179… 192.168.94.129  45.33.32.156    TCP      58 52860 → 139 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46813 9509.0607972… 192.168.94.129  45.33.32.156    TCP      58 52860 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46814 9509.0608627… 192.168.94.129  45.33.32.156    TCP      58 52860 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46815 9509.0609249… 192.168.94.129  45.33.32.156    TCP      58 52860 → 445 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46816 9509.0633238… 45.33.32.156    192.168.94.129  TCP      60 256 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46817 9509.0633239… 45.33.32.156    192.168.94.129  TCP      60 23 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46818 9509.0633239… 45.33.32.156    192.168.94.129  TCP      60 993 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46819 9509.0642365… 45.33.32.156    192.168.94.129  TCP      60 1723 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46820 9509.0642366… 45.33.32.156    192.168.94.129  TCP      60 25 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46821 9509.0656398… 45.33.32.156    192.168.94.129  TCP      60 8080 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46822 9509.0656399… 45.33.32.156    192.168.94.129  TCP      60 3389 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46823 9509.0656399… 45.33.32.156    192.168.94.129  TCP      60 21 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46824 9509.0656399… 45.33.32.156    192.168.94.129  TCP      60 199 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
```

```
› Frame 46738: 137 bytes on wire (1096 bits), 137 bytes captured (1096 bits) on interface eth0, id 0
› Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: VMware_94:30:1d (00:0c:29:94:30:1d)
› Internet Protocol Version 4, Src: 192.168.94.1, Dst: 192.168.94.129
› User Datagram Protocol, Src Port: 5353, Dst Port: 57427
› Multicast Domain Name System (response)
```

```
0000  00 0c 29 94 30 1d 00 50  56 c0 00 08 08 00 45 00   ··)·0··P V·····E·
0010  00 7b 87 1f 00 00 ff 11  f6 7e c0 a8 5e 01 c0 a8   ·{······ ·~··^···
0020  5e 81 14 e9 e0 53 00 67  0c c9 00 01 84 00 00 01   ^····S·g ········
0030  00 02 00 00 00 00 09 5f  73 65 72 76 69 63 65 73   ·······_ services
0040  07 5f 64 6e 73 2d 73 64  04 5f 75 64 70 05 6c 6f   ·_dns-sd ·_udp·lo
0050  63 61 6c 00 00 0c 00 01  c0 0c 00 0c 00 01 00 00   cal····· ········
0060  00 0a 00 11 09 5f 6e 69  2d 6c 6f 67 6f 73 04 5f   ·····_ni -logos·_
0070  74 63 70 c0 23 c0 0c 00  0c 00 01 00 00 00 0a 00   tcp·#··· ········
0080  08 05 5f 68 74 74 70 c0  44                        ··_http· D
```

eth0: <live capture in progress>   Packets: 48815 · Displayed: 48815 (100.0%)   Profile: Default

Second Method of Testing:

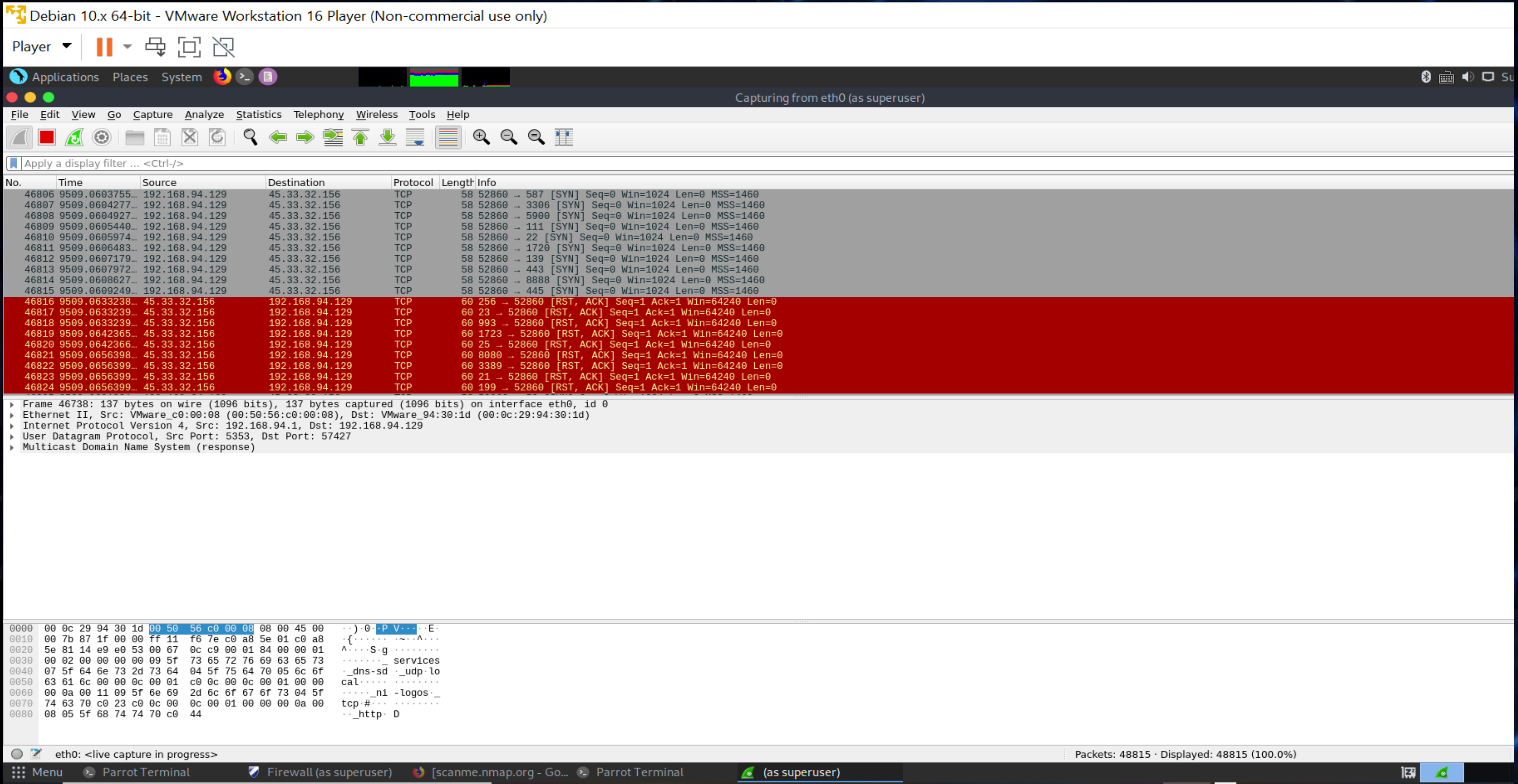
DoS vulnerability test to the target

(*nmap --script dos –Pn scanme.nmap.org*)

DoS (Denial of Service) Attack: An attack to slow down or close off the user's system by sending a ton of network traffic to crash their server

As a result caused some of the ports to be filtered, whereas, other ports remained open, which means the Firewall was able to stop the DoS attack to flood through most of the ports

METHODOLOGY AND RESULTS



```
No.   Time            Source          Destination     Protocol Length Info
46806 9509.0603755… 192.168.94.129   45.33.32.156     TCP      58 52860 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46807 9509.0604277… 192.168.94.129   45.33.32.156     TCP      58 52860 → 3306 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46808 9509.0604927… 192.168.94.129   45.33.32.156     TCP      58 52860 → 5900 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46809 9509.0605440… 192.168.94.129   45.33.32.156     TCP      58 52860 → 111 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46810 9509.0605974… 192.168.94.129   45.33.32.156     TCP      58 52860 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46811 9509.0606483… 192.168.94.129   45.33.32.156     TCP      58 52860 → 1720 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46812 9509.0607179… 192.168.94.129   45.33.32.156     TCP      58 52860 → 139 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46813 9509.0607972… 192.168.94.129   45.33.32.156     TCP      58 52860 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46814 9509.0608627… 192.168.94.129   45.33.32.156     TCP      58 52860 → 8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46815 9509.0609249… 192.168.94.129   45.33.32.156     TCP      58 52860 → 445 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46816 9509.0633238… 45.33.32.156     192.168.94.129   TCP      60 256 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46817 9509.0633239… 45.33.32.156     192.168.94.129   TCP      60 23 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46818 9509.0633239… 45.33.32.156     192.168.94.129   TCP      60 993 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46819 9509.0642365… 45.33.32.156     192.168.94.129   TCP      60 1723 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46820 9509.0642366… 45.33.32.156     192.168.94.129   TCP      60 25 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46821 9509.0656398… 45.33.32.156     192.168.94.129   TCP      60 8080 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46822 9509.0656399… 45.33.32.156     192.168.94.129   TCP      60 3389 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46823 9509.0656399… 45.33.32.156     192.168.94.129   TCP      60 21 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
46824 9509.0656399… 45.33.32.156     192.168.94.129   TCP      60 199 → 52860 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
```

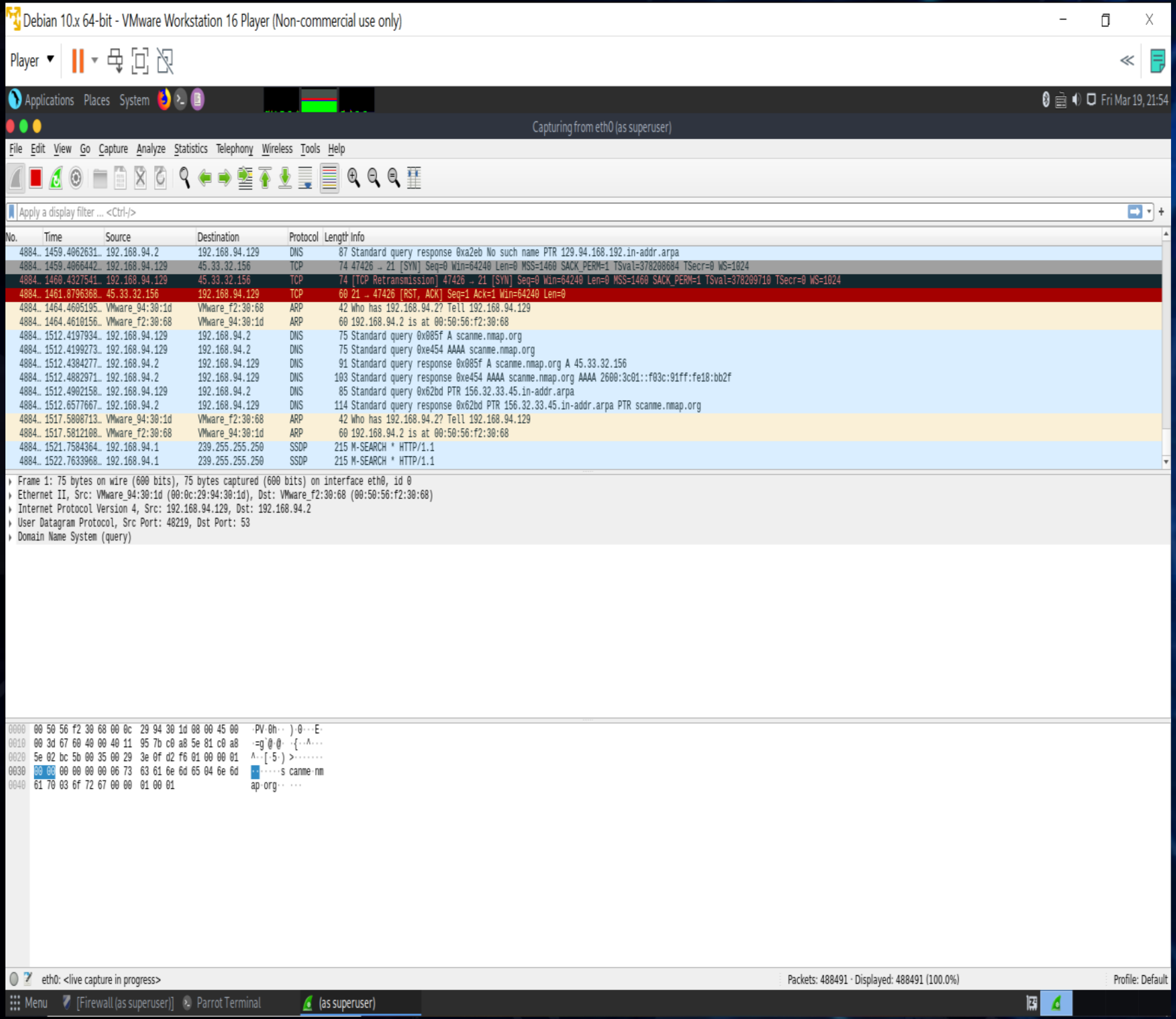
```
▸ Frame 46738: 137 bytes on wire (1096 bits), 137 bytes captured (1096 bits) on interface eth0, id 0
▸ Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: VMware_94:30:1d (00:0c:29:94:30:1d)
▸ Internet Protocol Version 4, Src: 192.168.94.1, Dst: 192.168.94.129
▸ User Datagram Protocol, Src Port: 5353, Dst Port: 57427
▸ Multicast Domain Name System (response)
```

```
0000  00 0c 29 94 30 1d 00 50  56 c0 00 08 08 00 45 00   ··)·0··P V·····E·
0010  00 7b 87 1f 00 00 ff 11  f6 7e c0 a8 5e 01 c0 a8   ·{······ ·~··^···
0020  5e 81 14 e9 e0 53 00 67  0c c9 00 01 84 00 00 01   ^····S·g ········
0030  00 02 00 00 00 00 09 5f  73 65 72 76 69 63 65 73   ·······_ services
0040  07 5f 64 6e 73 2d 73 64  04 5f 75 64 70 05 6c 6f   ·_dns-sd ·_udp·lo
0050  63 61 6c 00 00 0c 00 01  c0 0c 00 0c 00 01 00 00   cal····· ········
0060  00 0a 00 11 09 5f 6e 69  2d 6c 6f 67 6f 73 04 5f   ·····_ni -logos·_
0070  74 63 70 c0 23 c0 0c 00  0c 00 01 00 00 00 0a 00   tcp·#··· ········
0080  08 05 5f 68 74 74 70 c0  44                        ··_http· D
```

eth0: <live capture in progress>

Packets: 48815 · Displayed: 48815 (100.0%)

METHODOLOGY AND RESULTS


Debian 10.x 64-bit - VMware Workstation 16 Player (Non-commercial use only)
Player
Applications Places System
Fri Mar 19, 21:54
Capturing from eth0 (as superuser)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Menu
[Firewall (as superuser)]
Parrot Terminal
(as superuser)

```
No.    Time            Source             Destination        Protocol Length Info
4884…  1459.4062631…   192.168.94.2       192.168.94.129     DNS       87 Standard query response 0xa2eb No such name PTR 129.94.168.192.in-addr.arpa
4884…  1459.4066442…   192.168.94.129     45.33.32.156       TCP       74 47426 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=378208684 TSecr=0 WS=1024
4884…  1460.4327541…   192.168.94.129     45.33.32.156       TCP       74 [TCP Retransmission] 47426 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=378209710 TSecr=0 WS=1024
4884…  1461.8796368…   45.33.32.156       192.168.94.129     TCP       60 21 → 47426 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
4884…  1464.4605195…   VMware_94:30:1d    VMware_f2:30:68    ARP       42 Who has 192.168.94.2? Tell 192.168.94.129
4884…  1464.4610156…   VMware_f2:30:68    VMware_94:30:1d    ARP       60 192.168.94.2 is at 00:50:56:f2:30:68
4884…  1512.4197934…   192.168.94.129     192.168.94.2       DNS       75 Standard query 0x085f A scanme.nmap.org
4884…  1512.4199273…   192.168.94.129     192.168.94.2       DNS       75 Standard query 0xe454 AAAA scanme.nmap.org
4884…  1512.4384277…   192.168.94.2       192.168.94.129     DNS       91 Standard query response 0x085f A scanme.nmap.org A 45.33.32.156
4884…  1512.4882971…   192.168.94.2       192.168.94.129     DNS      103 Standard query response 0xe454 AAAA scanme.nmap.org AAAA 2600:3c01::f03c:91ff:fe18:bb2f
4884…  1512.4902158…   192.168.94.129     192.168.94.2       DNS       85 Standard query 0x62bd PTR 156.32.33.45.in-addr.arpa
4884…  1512.6577667…   192.168.94.2       192.168.94.129     DNS      114 Standard query response 0x62bd PTR 156.32.33.45.in-addr.arpa PTR scanme.nmap.org
4884…  1517.5808713…   VMware_94:30:1d    VMware_f2:30:68    ARP       42 Who has 192.168.94.2? Tell 192.168.94.129
4884…  1517.5812108…   VMware_f2:30:68    VMware_94:30:1d    ARP       60 192.168.94.2 is at 00:50:56:f2:30:68
4884…  1521.7584364…   192.168.94.1       239.255.255.250    SSDP     215 M-SEARCH * HTTP/1.1
4884…  1522.7633968…   192.168.94.1       239.255.255.250    SSDP     215 M-SEARCH * HTTP/1.1
```

```
▸ Frame 1: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface eth0, id 0
▸ Ethernet II, Src: VMware_94:30:1d (00:0c:29:94:30:1d), Dst: VMware_f2:30:68 (00:50:56:f2:30:68)
▸ Internet Protocol Version 4, Src: 192.168.94.129, Dst: 192.168.94.2
▸ User Datagram Protocol, Src Port: 48219, Dst Port: 53
▸ Domain Name System (query)
```

```
0000  00 50 56 f2 30 68 00 0c  29 94 30 1d 08 00 45 00   ·PV·0h·· )·0···E·
0010  00 3d 67 60 40 00 40 11  95 7b c0 a8 5e 81 c0 a8   ·=g`@·@· ·{··^···
0020  5e 02 bc 5b 00 35 00 29  3e 0f d2 f6 01 00 00 01   ^··[·5·) >·······
0030  00 00 00 00 00 00 06 73  63 61 6e 6d 65 04 6e 6d   ·······s canme·nm
0040  61 70 03 6f 72 67 00 00  01 00 01                  ap·org·· ···
```

```
eth0: <live capture in progress>          Packets: 488491 · Displayed: 488491 (100.0%)          Profile: Default
```

Final Method of Testing:

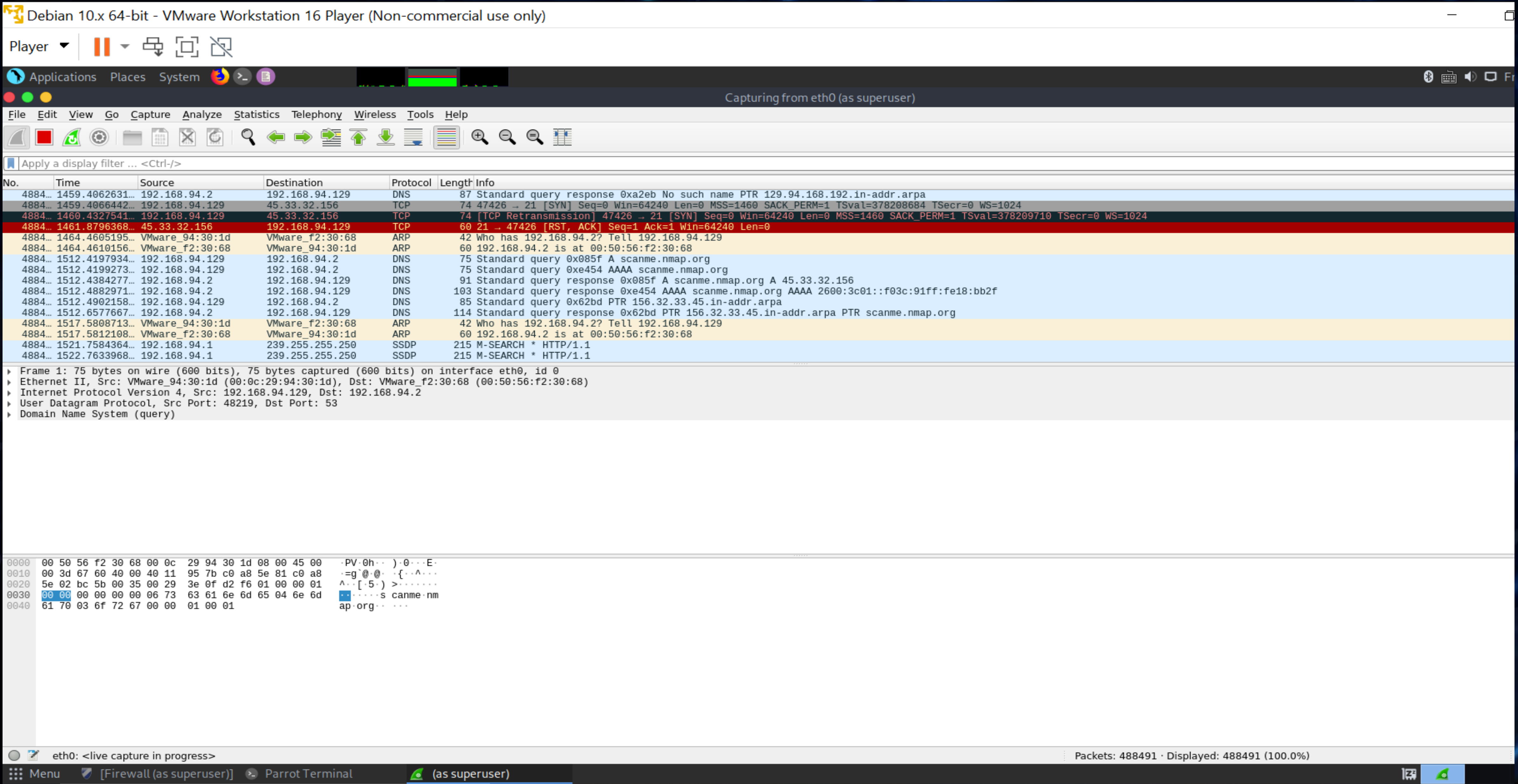
An FTP bounce scan to bypass Firewall

(*nmap -p 22, 25, 135 -Pn -v - b 192.168.94.129 scanme.nmap.org*)

FTP Bounce Scan: Allows a user to connect one server to a third party server to sent files to

As a result Wireshark shows that the Firewall was able to detect the bounce scan attack in black and block the communication to the target IP address (45.33.32.156) with an RST shown in red

METHODOLOGY AND RESULTS


Debian 10.x 64-bit - VMware Workstation 16 Player (Non-commercial use only)
Player
Applications
Places
System
Capturing from eth0 (as superuser)
File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Wireless
Tools
Help
Apply a display filter ... <Ctrl-/>
Menu
[Firewall (as superuser)]
Parrot Terminal
(as superuser)

```
No.    Time            Source              Destination       Protocol Length Info
 4884… 1459.4062631… 192.168.94.2       192.168.94.129     DNS        87 Standard query response 0xa2eb No such name PTR 129.94.168.192.in-addr.arpa
 4884… 1459.4066442… 192.168.94.129     45.33.32.156       TCP        74 47426 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=378208684 TSecr=0 WS=1024
 4884… 1460.4327541… 192.168.94.129     45.33.32.156       TCP        74 [TCP Retransmission] 47426 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=378209710 TSecr=0 WS=1024
 4884… 1461.8796368… 45.33.32.156       192.168.94.129     TCP        60 21 → 47426 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
 4884… 1464.4605195… VMware_94:30:1d    VMware_f2:30:68    ARP        42 Who has 192.168.94.2? Tell 192.168.94.129
 4884… 1464.4610156… VMware_f2:30:68    VMware_94:30:1d    ARP        60 192.168.94.2 is at 00:50:56:f2:30:68
 4884… 1512.4197934… 192.168.94.129     192.168.94.2       DNS        75 Standard query 0x085f A scanme.nmap.org
 4884… 1512.4199273… 192.168.94.129     192.168.94.2       DNS        75 Standard query 0xe454 AAAA scanme.nmap.org
 4884… 1512.4384277… 192.168.94.2       192.168.94.129     DNS        91 Standard query response 0x085f A scanme.nmap.org A 45.33.32.156
 4884… 1512.4882971… 192.168.94.2       192.168.94.129     DNS       103 Standard query response 0xe454 AAAA scanme.nmap.org AAAA 2600:3c01::f03c:91ff:fe18:bb2f
 4884… 1512.4902158… 192.168.94.129     192.168.94.2       DNS        85 Standard query 0x62bd PTR 156.32.33.45.in-addr.arpa
 4884… 1512.6577667… 192.168.94.2       192.168.94.129     DNS       114 Standard query response 0x62bd PTR 156.32.33.45.in-addr.arpa PTR scanme.nmap.org
 4884… 1517.5808713… VMware_94:30:1d    VMware_f2:30:68    ARP        42 Who has 192.168.94.2? Tell 192.168.94.129
 4884… 1517.5812108… VMware_f2:30:68    VMware_94:30:1d    ARP        60 192.168.94.2 is at 00:50:56:f2:30:68
 4884… 1521.7584364… 192.168.94.1       239.255.255.250    SSDP      215 M-SEARCH * HTTP/1.1
 4884… 1522.7633968… 192.168.94.1       239.255.255.250    SSDP      215 M-SEARCH * HTTP/1.1
```

```
▸ Frame 1: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface eth0, id 0
▸ Ethernet II, Src: VMware_94:30:1d (00:0c:29:94:30:1d), Dst: VMware_f2:30:68 (00:50:56:f2:30:68)
▸ Internet Protocol Version 4, Src: 192.168.94.129, Dst: 192.168.94.2
▸ User Datagram Protocol, Src Port: 48219, Dst Port: 53
▸ Domain Name System (query)
```

```
0000  00 50 56 f2 30 68 00 0c  29 94 30 1d 08 00 45 00   ·PV·0h·· )·0···E·
0010  00 3d 67 60 40 00 40 11  95 7b c0 a8 5e 81 c0 a8   ·=g`@·@· ·{··^···
0020  5e 02 bc 5b 00 35 00 29  3e 0f d2 f6 01 00 00 01   ^··[·5·) >·······
0030  00 00 00 00 00 00 06 73  63 61 6e 6d 65 04 6e 6d   ·······s canme·nm
0040  61 70 03 6f 72 67 00 00  01 00 01                  ap·org·· ···
```

```
eth0: <live capture in progress>                Packets: 488491 · Displayed: 488491 (100.0%)
```

PROCEDURAL STEPS FOR: *Malware Protection*

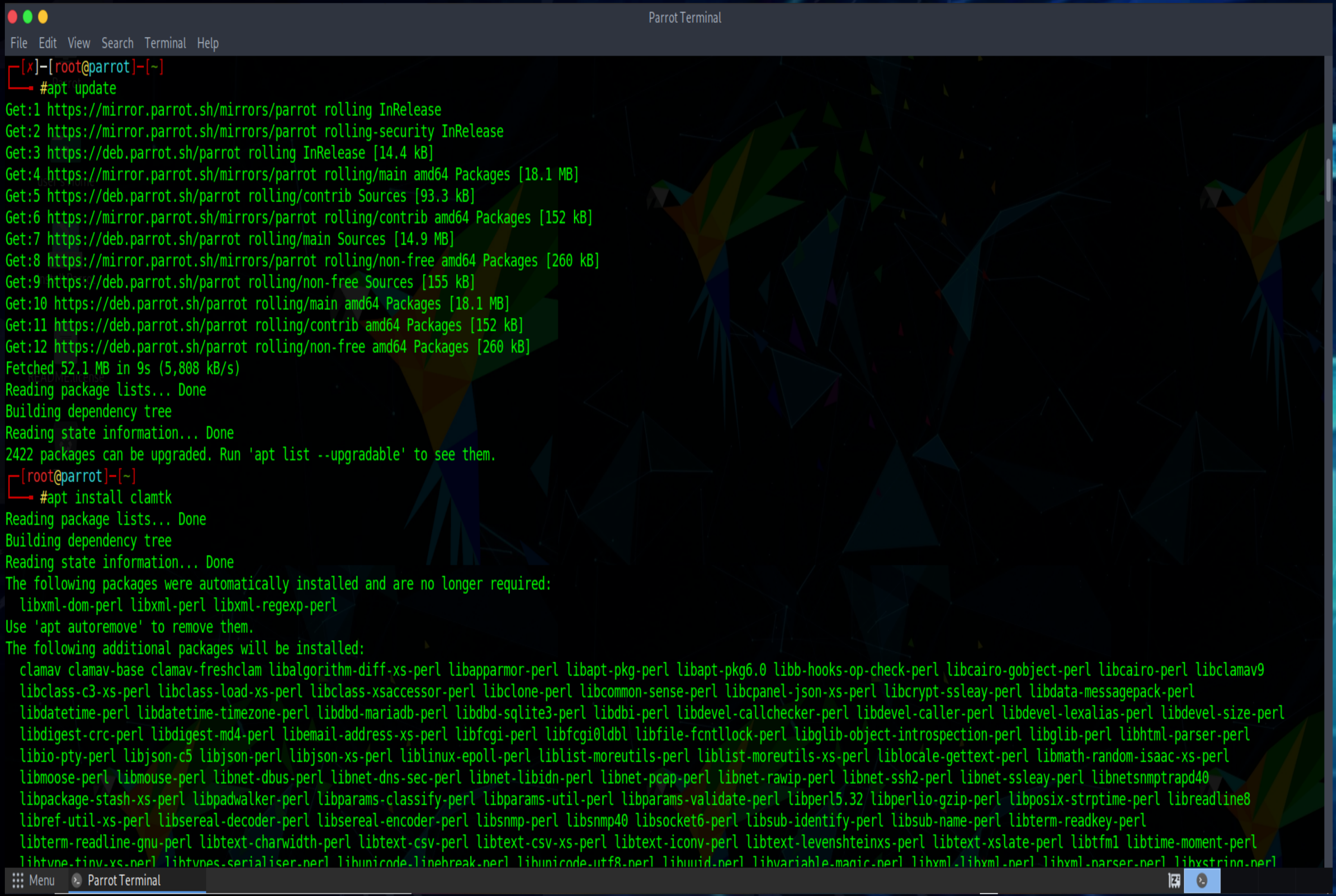
Background of Malware Protection

- Malware protection - Stops all kinds of malware threats where it uploads the suspicious programs to the cloud for scanning so that your system can run smoothly
- Two examples of Malware protection are signature-based detection and behavioral analysis

CYBERSECURITY TOOLS USED TO TEST MALWARE PROTECTION

- **Metasploit** is an Open-Source platform intended to make hacking a simple and important tool for Pentesting, which automates and gathers all the information, detection evasion, and access
- **Wireshark** does three things: packet capture, filtering, and visualization
- **Nikto** is a test web server for multiple items such as program files, checking for outdated version for specific problems on server, and server configuration items

MALWARE PROTECTION INSTALLATION



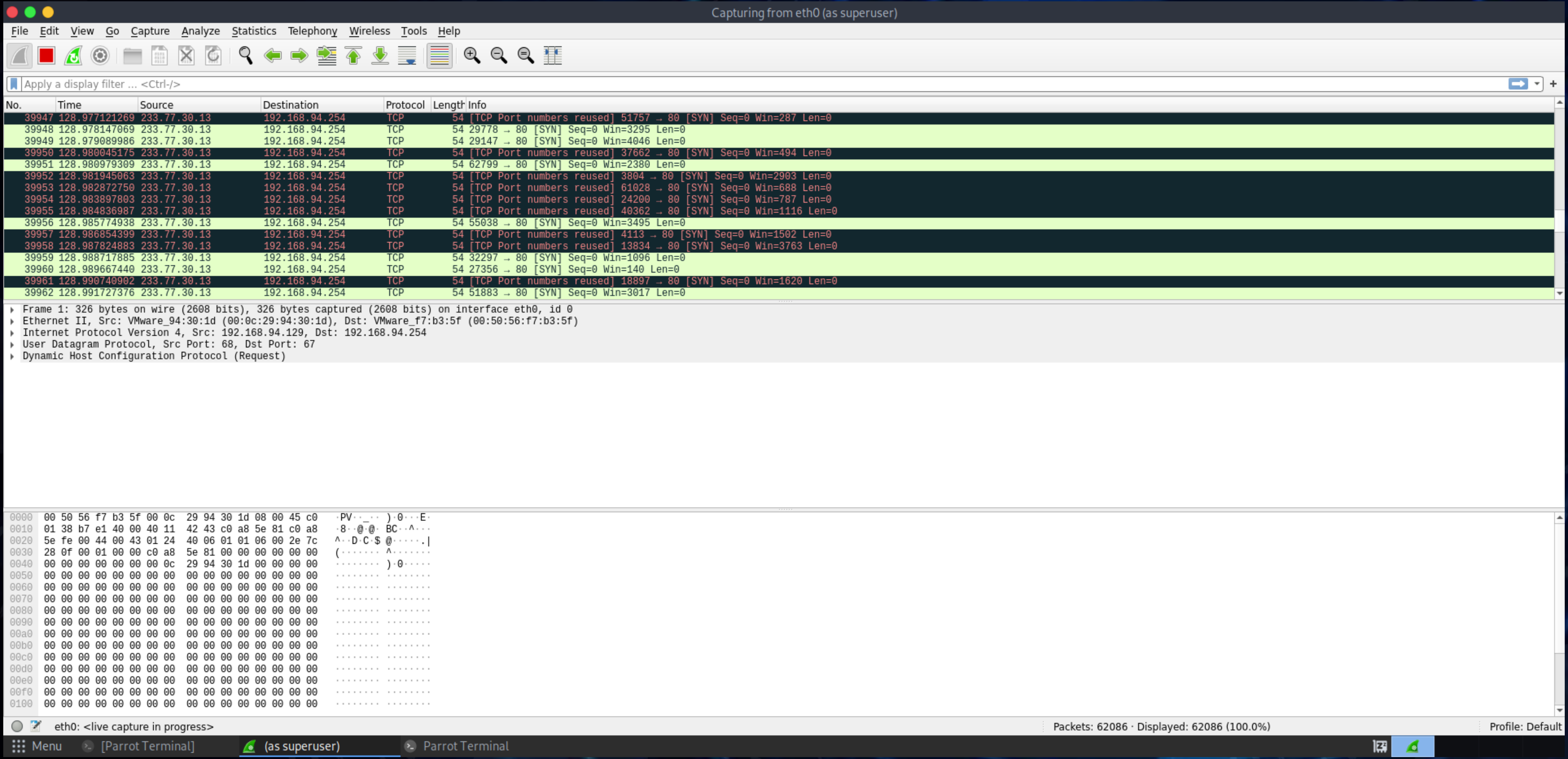
```
┌─[✗]─[root@parrot]─[~]
└──╼ #apt update
Get:1 https://mirror.parrot.sh/mirrors/parrot rolling InRelease
Get:2 https://mirror.parrot.sh/mirrors/parrot rolling-security InRelease
Get:3 https://deb.parrot.sh/parrot rolling InRelease [14.4 kB]
Get:4 https://mirror.parrot.sh/mirrors/parrot rolling/main amd64 Packages [18.1 MB]
Get:5 https://deb.parrot.sh/parrot rolling/contrib Sources [93.3 kB]
Get:6 https://mirror.parrot.sh/mirrors/parrot rolling/contrib amd64 Packages [152 kB]
Get:7 https://deb.parrot.sh/parrot rolling/main Sources [14.9 MB]
Get:8 https://mirror.parrot.sh/mirrors/parrot rolling/non-free amd64 Packages [260 kB]
Get:9 https://deb.parrot.sh/parrot rolling/non-free Sources [155 kB]
Get:10 https://deb.parrot.sh/parrot rolling/main amd64 Packages [18.1 MB]
Get:11 https://deb.parrot.sh/parrot rolling/contrib amd64 Packages [152 kB]
Get:12 https://deb.parrot.sh/parrot rolling/non-free amd64 Packages [260 kB]
Fetched 52.1 MB in 9s (5,808 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
2422 packages can be upgraded. Run 'apt list --upgradable' to see them.
┌─[root@parrot]─[~]
└──╼ #apt install clamtk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libxml-dom-perl libxml-perl libxml-regexp-perl
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  clamav clamav-base clamav-freshclam libalgorithm-diff-xs-perl libapparmor-perl libapt-pkg-perl libapt-pkg6.0 libb-hooks-op-check-perl libcairo-gobject-perl libcairo-perl libclamav9
  libclass-c3-xs-perl libclass-load-xs-perl libclass-xsaccessor-perl libclone-perl libcommon-sense-perl libcpanel-json-xs-perl libcrypt-ssleay-perl libdata-messagepack-perl
  libdatetime-perl libdatetime-timezone-perl libdbd-mariadb-perl libdbd-sqlite3-perl libdbi-perl libdevel-callchecker-perl libdevel-caller-perl libdevel-lexalias-perl libdevel-size-perl
  libdigest-crc-perl libdigest-md4-perl libemail-address-xs-perl libfcgi-perl libfcgi0ldbl libfile-fcntllock-perl libglib-object-introspection-perl libglib-perl libhtml-parser-perl
  libio-pty-perl libjson-c5 libjson-perl libjson-xs-perl liblinux-epoll-perl liblist-moreutils-perl liblist-moreutils-xs-perl liblocale-gettext-perl libmath-random-isaac-xs-perl
  libmoose-perl libmouse-perl libnet-dbus-perl libnet-dns-sec-perl libnet-libidn-perl libnet-pcap-perl libnet-rawip-perl libnet-ssh2-perl libnet-ssleay-perl libnetsnmptrapd40
  libpackage-stash-xs-perl libpadwalker-perl libparams-classify-perl libparams-util-perl libparams-validate-perl libperl5.32 libperlio-gzip-perl libposix-strptime-perl libreadline8
  libref-util-xs-perl libsereal-decoder-perl libsereal-encoder-perl libsnmp-perl libsnmp40 libsocket6-perl libsub-identify-perl libsub-name-perl libterm-readkey-perl
  libterm-readline-gnu-perl libtext-charwidth-perl libtext-csv-perl libtext-csv-xs-perl libtext-iconv-perl libtext-levenshteinxs-perl libtext-xslate-perl libtfm1 libtime-moment-perl
  libtype-tiny-xs-perl libtypes-serialiser-perl libunicode-linebreak-perl libunicode-utf8-perl libuuid-perl libvariable-magic-perl libxml-libxml-perl libxml-parser-perl libxstring-perl
```

To install malware protection ClamAV, one must go to the terminal and enter the command:

apt install clamtk

With the above command, it will successfully be installed onto the device as shown

METHODOLOGY AND RESULTS


Capturing from eth0 (as superuser)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Menu
[Parrot Terminal]
(as superuser)
Parrot Terminal
Profile: Default

```
No.   Time          Source        Destination     Protocol Length Info
39947 128.977121269 233.77.30.13  192.168.94.254  TCP      54 [TCP Port numbers reused] 51757 → 80 [SYN] Seq=0 Win=287 Len=0
39948 128.978147069 233.77.30.13  192.168.94.254  TCP      54 29778 → 80 [SYN] Seq=0 Win=3295 Len=0
39949 128.979089986 233.77.30.13  192.168.94.254  TCP      54 29147 → 80 [SYN] Seq=0 Win=4046 Len=0
39950 128.980045175 233.77.30.13  192.168.94.254  TCP      54 [TCP Port numbers reused] 37662 → 80 [SYN] Seq=0 Win=494 Len=0
39951 128.980979309 233.77.30.13  192.168.94.254  TCP      54 62799 → 80 [SYN] Seq=0 Win=2380 Len=0
39952 128.981945063 233.77.30.13  192.168.94.254  TCP      54 [TCP Port numbers reused] 3804 → 80 [SYN] Seq=0 Win=2903 Len=0
39953 128.982872750 233.77.30.13  192.168.94.254  TCP      54 [TCP Port numbers reused] 61028 → 80 [SYN] Seq=0 Win=688 Len=0
39954 128.983897803 233.77.30.13  192.168.94.254  TCP      54 [TCP Port numbers reused] 24200 → 80 [SYN] Seq=0 Win=787 Len=0
39955 128.984836987 233.77.30.13  192.168.94.254  TCP      54 [TCP Port numbers reused] 40362 → 80 [SYN] Seq=0 Win=1116 Len=0
39956 128.985774938 233.77.30.13  192.168.94.254  TCP      54 55038 → 80 [SYN] Seq=0 Win=3495 Len=0
39957 128.986854399 233.77.30.13  192.168.94.254  TCP      54 [TCP Port numbers reused] 4113 → 80 [SYN] Seq=0 Win=1502 Len=0
39958 128.987824883 233.77.30.13  192.168.94.254  TCP      54 [TCP Port numbers reused] 13834 → 80 [SYN] Seq=0 Win=3763 Len=0
39959 128.988717885 233.77.30.13  192.168.94.254  TCP      54 32297 → 80 [SYN] Seq=0 Win=1096 Len=0
39960 128.989667440 233.77.30.13  192.168.94.254  TCP      54 27356 → 80 [SYN] Seq=0 Win=140 Len=0
39961 128.990740902 233.77.30.13  192.168.94.254  TCP      54 [TCP Port numbers reused] 18897 → 80 [SYN] Seq=0 Win=1620 Len=0
39962 128.991727376 233.77.30.13  192.168.94.254  TCP      54 51883 → 80 [SYN] Seq=0 Win=3017 Len=0
```

```
Frame 1: 326 bytes on wire (2608 bits), 326 bytes captured (2608 bits) on interface eth0, id 0
Ethernet II, Src: VMware_94:30:1d (00:0c:29:94:30:1d), Dst: VMware_f7:b3:5f (00:50:56:f7:b3:5f)
Internet Protocol Version 4, Src: 192.168.94.129, Dst: 192.168.94.254
User Datagram Protocol, Src Port: 68, Dst Port: 67
Dynamic Host Configuration Protocol (Request)
```

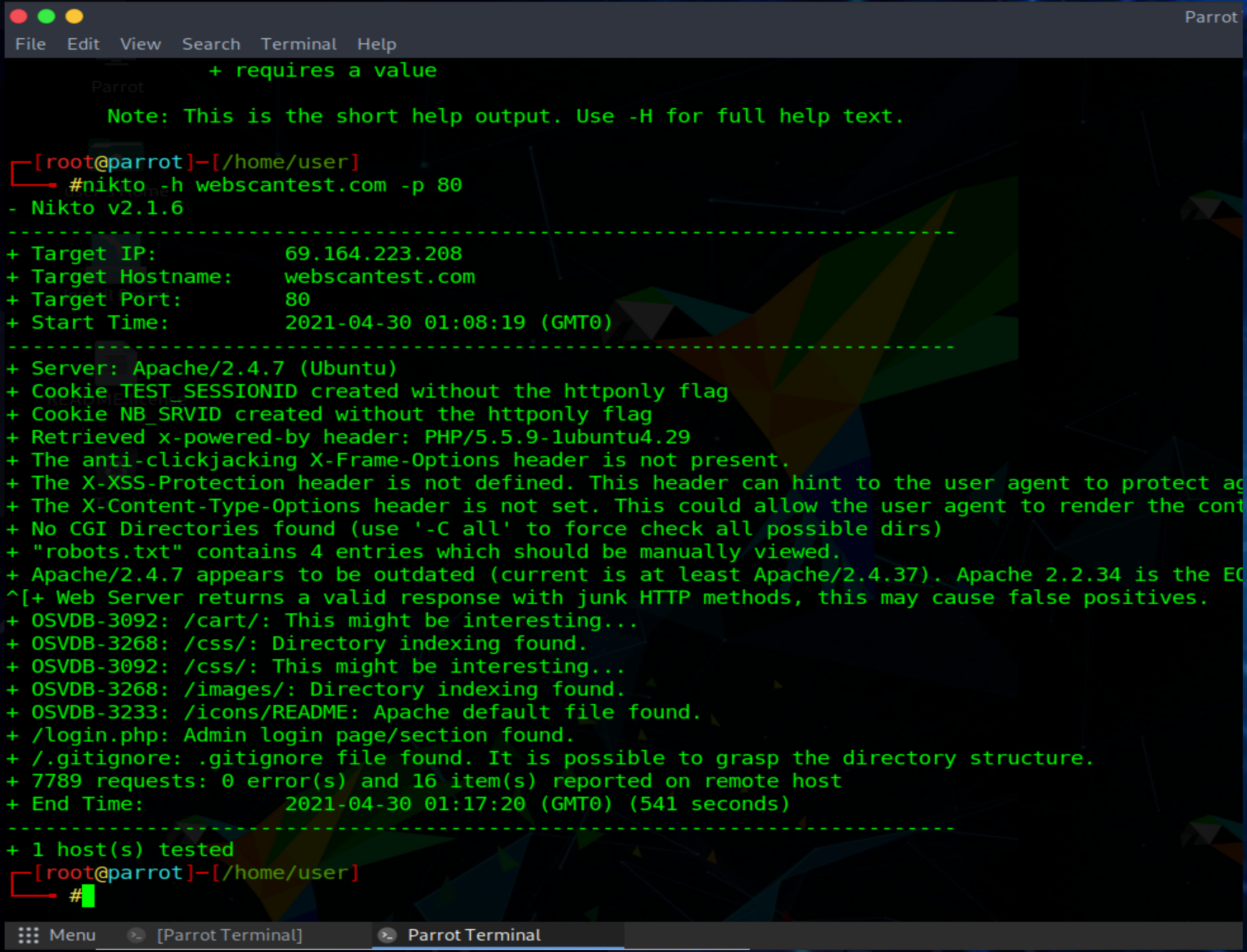
```
0000  00 50 56 f7 b3 5f 00 0c  29 94 30 1d 08 00 45 c0   ·PV··_·· )·0···E·
0010  01 38 b7 e1 40 00 40 11  42 43 c0 a8 5e 81 c0 a8   ·8··@·@· BC··^···
0020  5e fe 00 44 00 43 01 24  40 06 01 01 06 00 2e 7c   ^··D·C·$ @······.|
0030  28 0f 00 01 00 00 c0 a8  5e 81 00 00 00 00 00 00   (······· ^·······
0040  00 00 00 00 00 00 00 0c  29 94 30 1d 00 00 00 00   ········ )·0·····
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   ········ ········
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
```

```
eth0: <live capture in progress>          Packets: 62086 · Displayed: 62086 (100.0%)
```

Wireshark Detecting and Blocking the DoS Attack

METHODOLOGY AND RESULTS:

Detecting the Web Server Using Nikto



```
                    + requires a value

        Note: This is the short help output. Use -H for full help text.

┌─[root@parrot]─[/home/user]
└──╼ #nikto -h webscantest.com -p 80
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          69.164.223.208
+ Target Hostname:    webscantest.com
+ Target Port:        80
+ Start Time:         2021-04-30 01:08:19 (GMT0)
---------------------------------------------------------------------------
+ Server: Apache/2.4.7 (Ubuntu)
+ Cookie TEST_SESSIONID created without the httponly flag
+ Cookie NB_SRVID created without the httponly flag
+ Retrieved x-powered-by header: PHP/5.5.9-1ubuntu4.29
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect ag
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the cont
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ "robots.txt" contains 4 entries which should be manually viewed.
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EO
^[+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-3092: /cart/: This might be interesting...
+ OSVDB-3268: /css/: Directory indexing found.
+ OSVDB-3092: /css/: This might be interesting...
+ OSVDB-3268: /images/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /login.php: Admin login page/section found.
+ /.gitignore: .gitignore file found. It is possible to grasp the directory structure.
+ 7789 requests: 0 error(s) and 16 item(s) reported on remote host
+ End Time:           2021-04-30 01:17:20 (GMT0) (541 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
┌─[root@parrot]─[/home/user]
└──╼ #
```

The Figure shows the command:

nikto -h webscantest.com

This means that we are detecting and scanning the website webscantest.com for any vulnerabilities

It also shows the result of all other vulnerabilities

PROCEDURAL STEPS FOR: *DNS FILTERING*

Background of DNS Filtering

It is a strategy that protects the user by blocking access to certain websites and IP addresses that is considered a threat to you and your computer

This method ensures the protection of data, keeps it secure, and allows companies to have control over what their employees can access on a company managed networks

Not only companies use this but public schools have DNS filtering, their sole purpose is to protect underage personnel from browsing the Internet

CYBERSECURITY TOOLS USED TO BYPASS DNS FILTERING

VPN

- is a tool that can protect a user from hackers by protecting the network traffic
- It Gives online privacy and creates a private network from a public Internet connection

Anonsurf

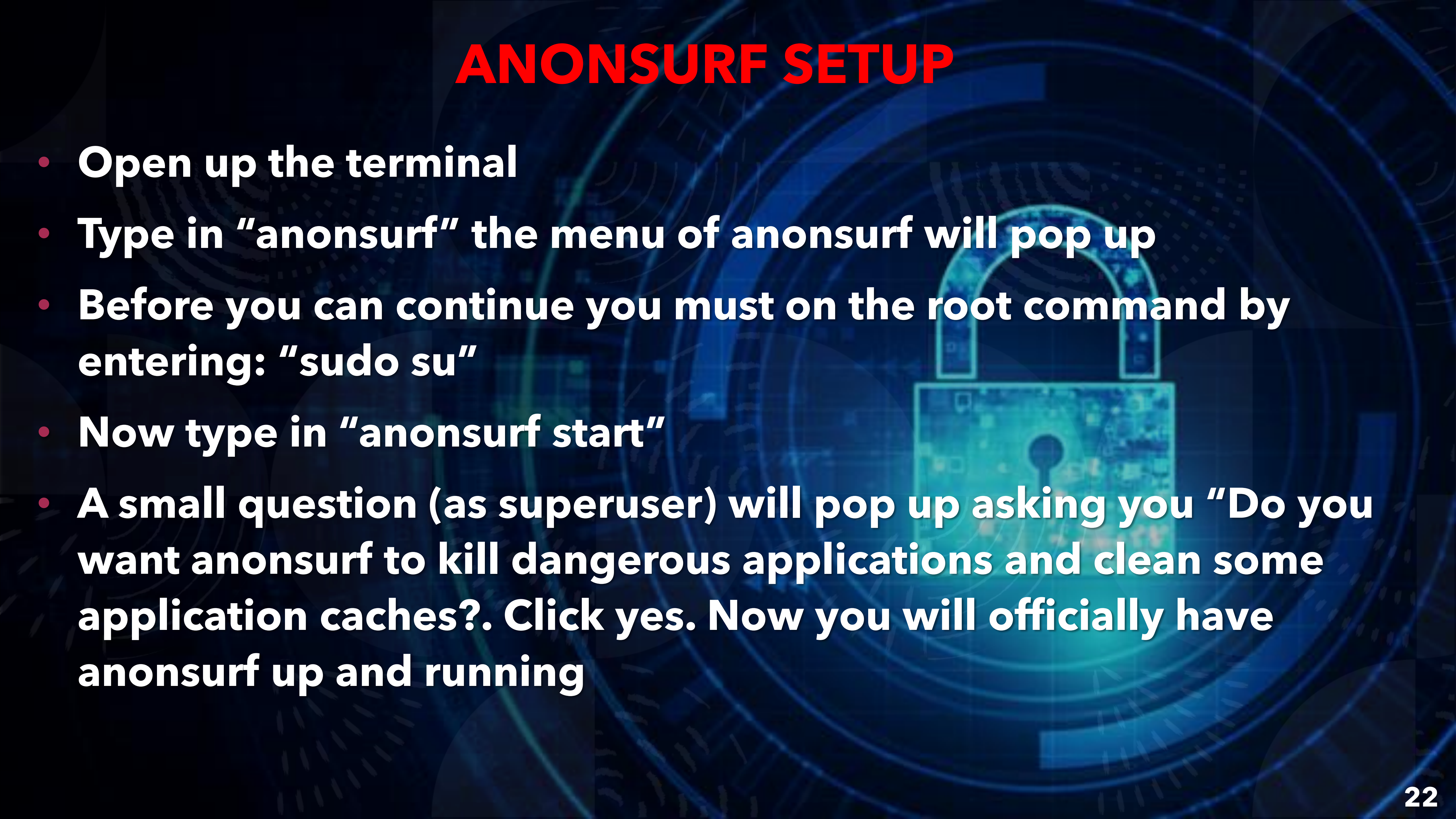
- A tool to navigate through the Internet and being protected and hidden at the same time
- By routing each and every packet through the TOR relay which change/mask your IP address

VPN INSTALLATION

- Search on the Internet for “https://protonvpn.com” and once in the website create an account
- After the account selection is over on the left side bar select download option and there you will be able to select any VPN for free in Japan, U.S., and Netherlands
- Before downloading, make sure you select the proper platform and protocol
- After this step you are ready to install it in your computer ur
- On your computer open up your VPN connections and select to configure VPN and create a new one

VPN INSTALLATION (Continued)

- For connection type click on "Import and save VPN configuration" and the window will pop up, navigate to your download folder and click open the VPN file you downloaded
- You will be brought to an editing window for the VPN. Here you will need to input username and password for this VPN. You will find this information in the website you were earlier
- Go back to it and on the left side bar click "account" and then click the selection that says "Open VPN/IKEv2 username". There copy the username and password and put in the info unto the authentication section. Once completed the info click save and you are finished

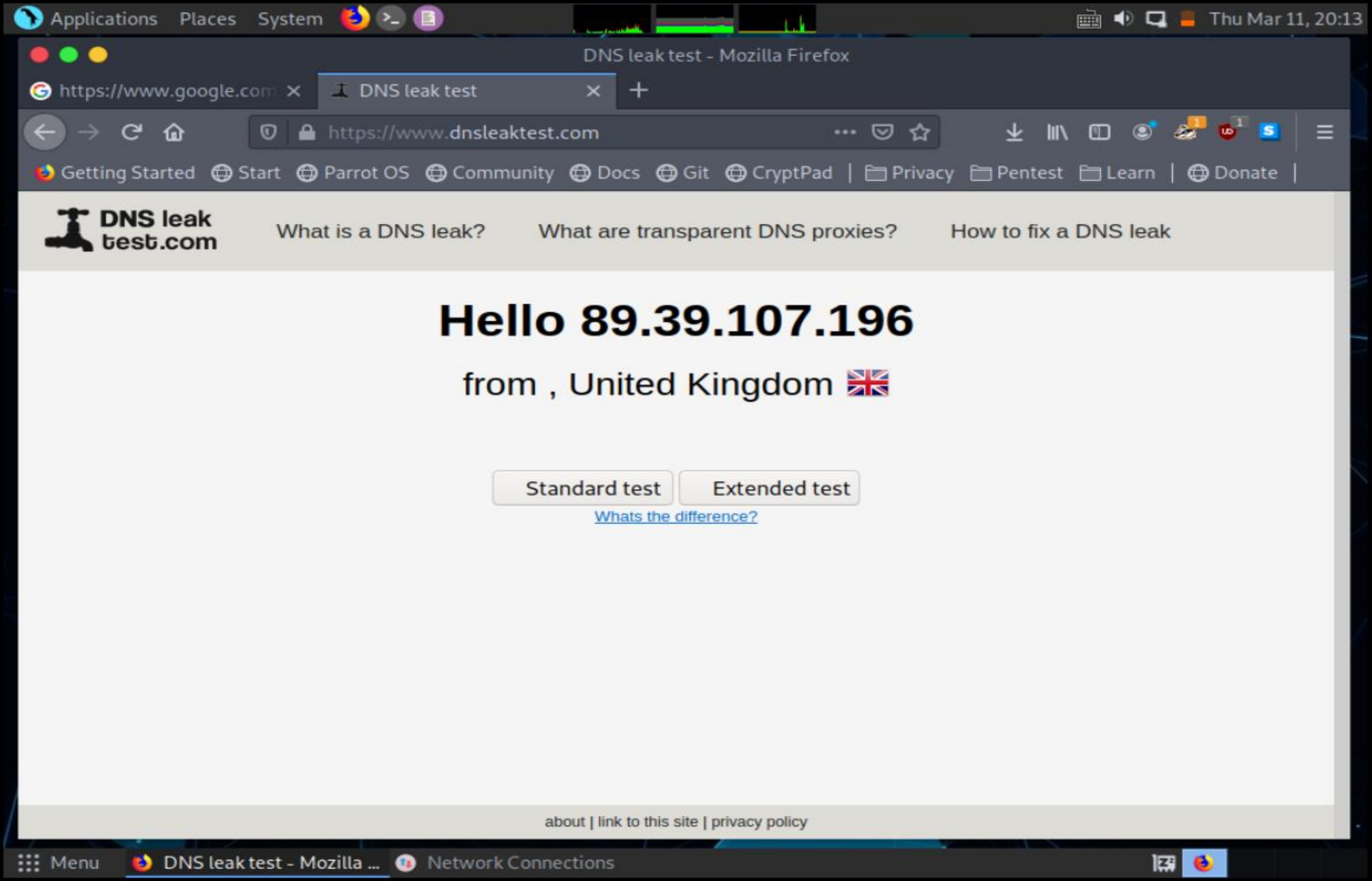


ANONSURF SETUP

- Open up the terminal
- Type in "anonsurf" the menu of anonsurf will pop up
- Before you can continue you must on the root command by entering: "sudo su"
- Now type in "anonsurf start"
- A small question (as superuser) will pop up asking you "Do you want anonsurf to kill dangerous applications and clean some application caches?. Click yes. Now you will officially have anonsurf up and running

METHODOLOGY AND RESULTS: *First Method of Testing for VPN*

Using the DNS leak test website to see current IP address having the VPN on

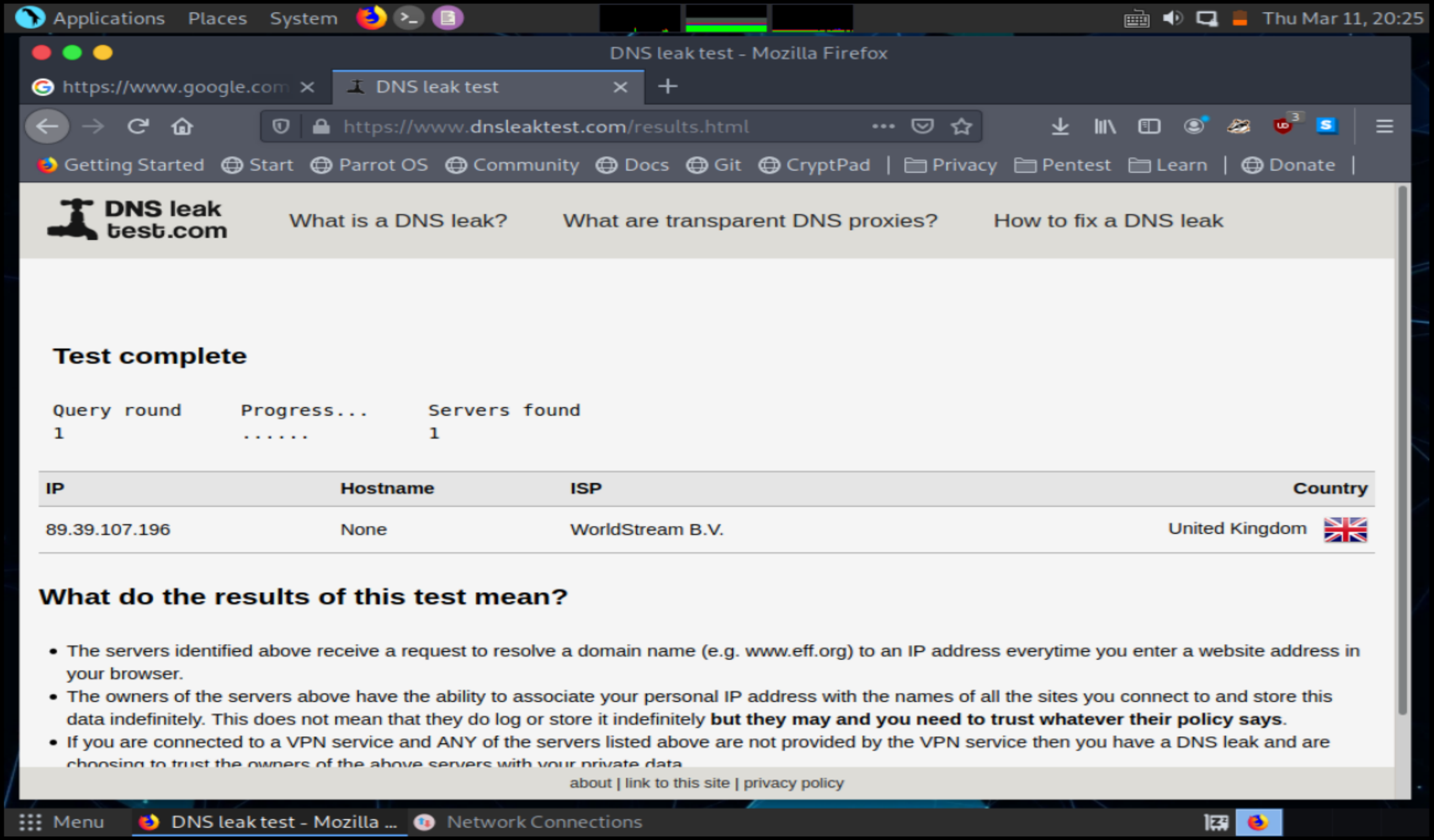


Hello 89.39.107.196

from , United Kingdom

METHODOLOGY AND RESULTS: *Second Method of Testing for VPN*

Using the DNS leak test website to run a standard or extended test, to see if there are no leaks in the VPN connections



Test complete

```
Query round     Progress...     Servers found
1               ......          1
```

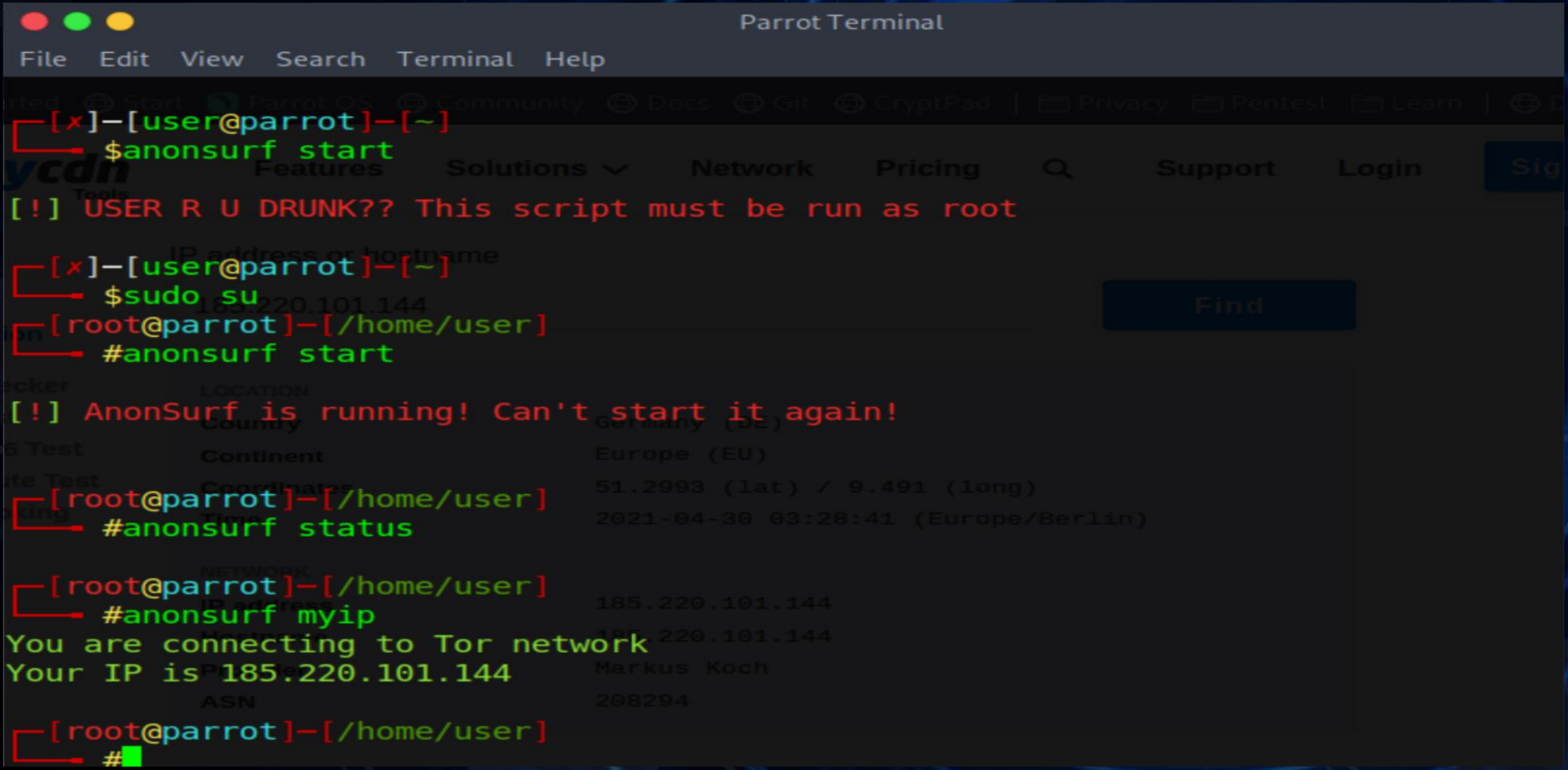
IP	Hostname	ISP	Country
89.39.107.196	None	WorldStream B.V.	United Kingdom

What do the results of this test mean?

- The servers identified above receive a request to resolve a domain name (e.g. www.eff.org) to an IP address everytime you enter a website address in your browser.
- The owners of the servers above have the ability to associate your personal IP address with the names of all the sites you connect to and store this data indefinitely. This does not mean that they do log or store it indefinitely **but they may and you need to trust whatever their policy says**.
- If you are connected to a VPN service and ANY of the servers listed above are not provided by the VPN service then you have a DNS leak and are choosing to trust the owners of the above servers with your private data.

METHODOLOGY AND RESULTS: *First Method of Testing for AnonSurf*

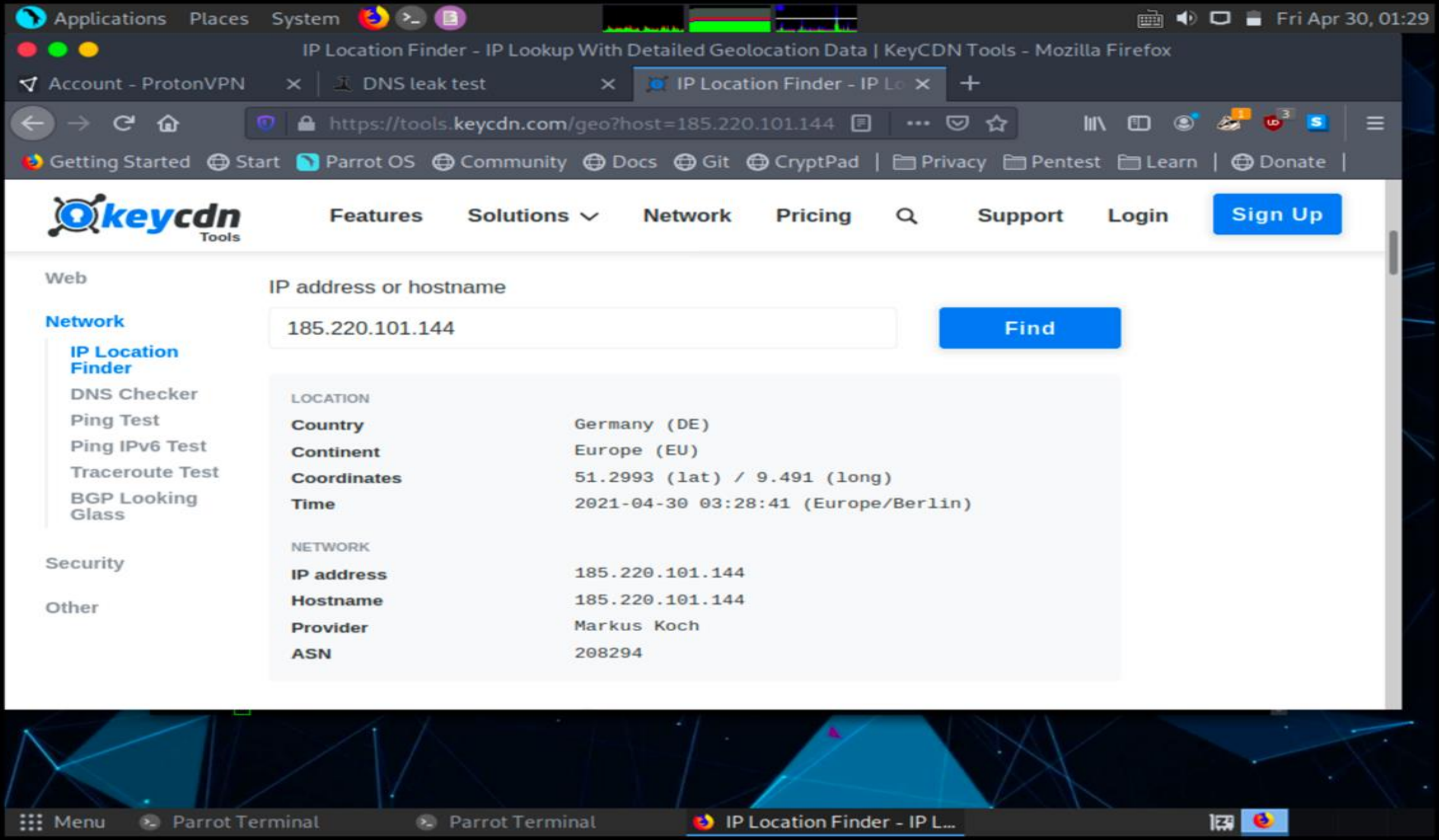
Typing in the terminal "anonsurf myip" will show your masked IP address



```
┌─[x]─[user@parrot]─[~]
└──╼ $anonsurf start

[!] USER R U DRUNK?? This script must be run as root

┌─[x]─[user@parrot]─[~]
└──╼ $sudo su
┌─[root@parrot]─[/home/user]
└──╼ #anonsurf start

[!] AnonSurf is running! Can't start it again!

┌─[root@parrot]─[/home/user]
└──╼ #anonsurf status

┌─[root@parrot]─[/home/user]
└──╼ #anonsurf myip
You are connecting to Tor network
Your IP is 185.220.101.144

┌─[root@parrot]─[/home/user]
└──╼ #
```

METHODOLOGY AND RESULTS: *Second Method of Testing for AnonSurf*

Using the IP Location Finder website to enter the IP address the AnonSurf provided us to discover the IP address details like location, region, hostname, provider, etc.



IP address or hostname

185.220.101.144

LOCATION

Country	Germany (DE)
Continent	Europe (EU)
Coordinates	51.2993 (lat) / 9.491 (long)
Time	2021-04-30 03:28:41 (Europe/Berlin)

NETWORK

IP address	185.220.101.144
Hostname	185.220.101.144
Provider	Markus Koch
ASN	208294

Conclusion

- ***We tested Firewall using Nmap, simulated DoS vulnerability test, and FTP bounce scan where the network activity was observed through Wireshark***
- ***We implemented multiple methods for bypassing malware protection using Metasploit, Wireshark, and Nikto***
- ***We learned that DNS filtering is a strategy that protects the user by preventing threat attacks to a computer system***
- ***We learned and tested both tools VPN and Anonsurf to navigate through the Internet and be protected/hidden at the same time***