WIRESHARK & SNIFFING ATTACKS

11th Sep 2022

OVERVIEW

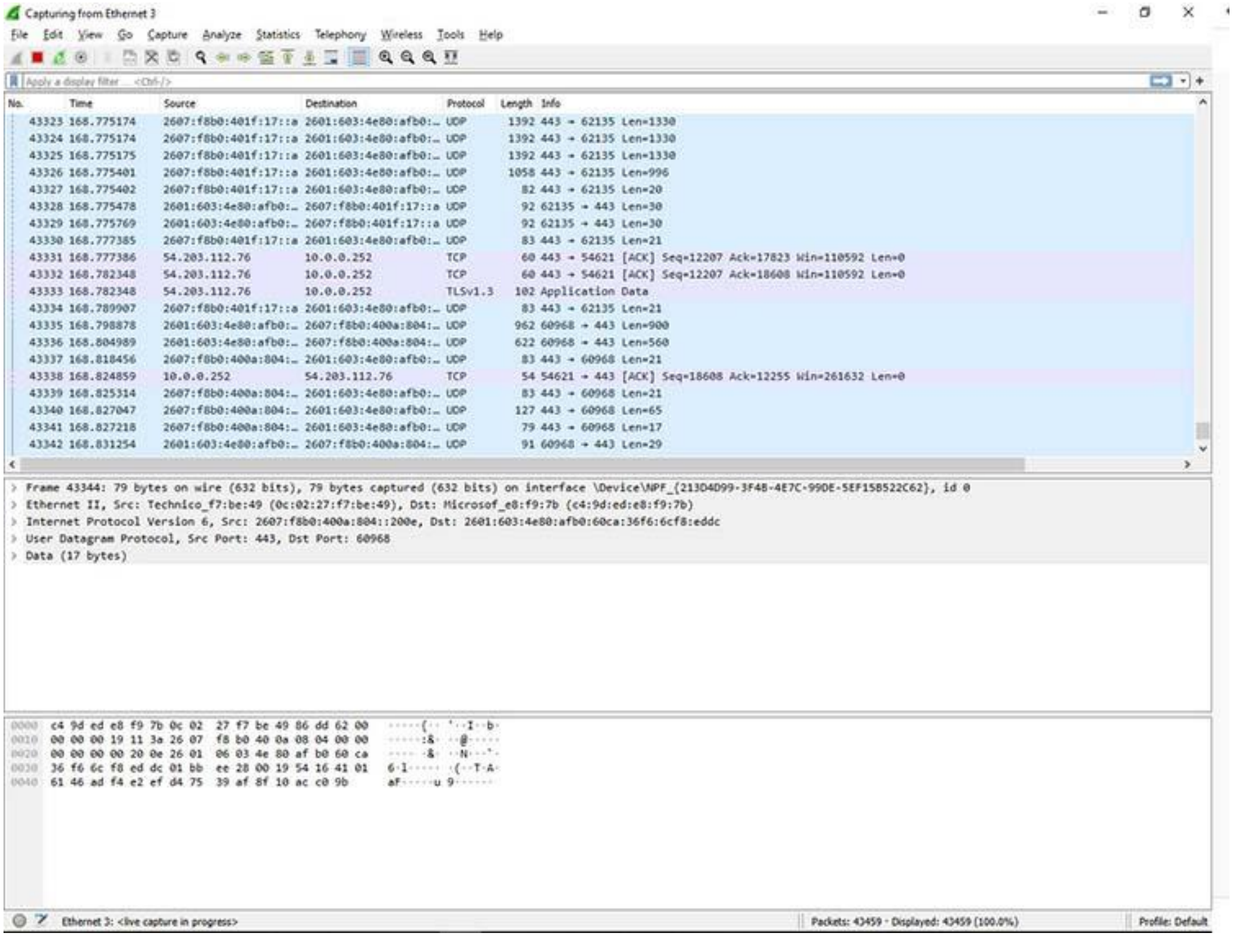
Wireshark is a network protocol analyser or an application that captures packets from a network connection, such as from your computer to your home office or the internet. A packet is a name given to a discrete unit of data in a typical Ethernet network.

Wireshark is the most often-used packet sniffer in the world. Like any other packet sniffer, Wireshark does three things:

1. **Packet Capture:** Wireshark listens to a network connection in real-time and then grabs entire streams of traffic – quite possibly tens of thousands of packets at a time.

2. **Filtering:** Wireshark is capable of slicing and dicing all of this random live data using filters. By applying a filter, you can obtain just the information you need to see.

3. **Visualization:** Wireshark, like any good packet sniffer, allows you to dive right into the very middle of a network packet. It also allows you to visualize entire conversations and network streams.


Capturing from Ethernet 3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time       Source                Destination           Protocol Length Info
43323 168.775174 2607:f8b0:401f:17::a  2601:603:4e80:afb0:… UDP      1392 443 → 62135 Len=1330
43324 168.775174 2607:f8b0:401f:17::a  2601:603:4e80:afb0:… UDP      1392 443 → 62135 Len=1330
43325 168.775175 2607:f8b0:401f:17::a  2601:603:4e80:afb0:… UDP      1392 443 → 62135 Len=1330
43326 168.775401 2607:f8b0:401f:17::a  2601:603:4e80:afb0:… UDP      1058 443 → 62135 Len=996
43327 168.775402 2607:f8b0:401f:17::a  2601:603:4e80:afb0:… UDP        82 443 → 62135 Len=20
43328 168.775478 2601:603:4e80:afb0:… 2607:f8b0:401f:17::a  UDP        92 62135 → 443 Len=30
43329 168.775769 2601:603:4e80:afb0:… 2607:f8b0:401f:17::a  UDP        92 62135 → 443 Len=30
43330 168.777385 2607:f8b0:401f:17::a  2601:603:4e80:afb0:… UDP        83 443 → 62135 Len=21
43331 168.777386 54.203.112.76         10.0.0.252            TCP        60 443 → 54621 [ACK] Seq=12207 Ack=17823 Win=110592 Len=0
43332 168.782348 54.203.112.76         10.0.0.252            TCP        60 443 → 54621 [ACK] Seq=12207 Ack=18608 Win=110592 Len=0
43333 168.782348 54.203.112.76         10.0.0.252            TLSv1.3   102 Application Data
43334 168.789907 2607:f8b0:401f:17::a  2601:603:4e80:afb0:… UDP        83 443 → 62135 Len=21
43335 168.798878 2601:603:4e80:afb0:… 2607:f8b0:400a:804:… UDP       962 60968 → 443 Len=900
43336 168.804989 2601:603:4e80:afb0:… 2607:f8b0:400a:804:… UDP       622 60968 → 443 Len=560
43337 168.818456 2607:f8b0:400a:804:… 2601:603:4e80:afb0:… UDP        83 443 → 60968 Len=21
43338 168.824859 10.0.0.252            54.203.112.76         TCP        54 54621 → 443 [ACK] Seq=18608 Ack=12255 Win=261632 Len=0
43339 168.825314 2607:f8b0:400a:804:… 2601:603:4e80:afb0:… UDP        83 443 → 60968 Len=21
43340 168.827047 2607:f8b0:400a:804:… 2601:603:4e80:afb0:… UDP       127 443 → 60968 Len=65
43341 168.827218 2607:f8b0:400a:804:… 2601:603:4e80:afb0:… UDP        79 443 → 60968 Len=17
43342 168.831254 2601:603:4e80:afb0:… 2607:f8b0:400a:804:… UDP        91 60968 → 443 Len=29
```

```
> Frame 43344: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface \Device\NPF_{213D4D99-3F48-4E7C-99DE-5EF15B522C62}, id 0
> Ethernet II, Src: Technico_f7:be:49 (0c:02:27:f7:be:49), Dst: Microsof_e8:f9:7b (c4:9d:ed:e8:f9:7b)
> Internet Protocol Version 6, Src: 2607:f8b0:400a:804::200e, Dst: 2601:603:4e80:afb0:60ca:36f6:6cf8:eddc
> User Datagram Protocol, Src Port: 443, Dst Port: 60968
> Data (17 bytes)
```

```
0000  c4 9d ed e8 f9 7b 0c 02  27 f7 be 49 86 dd 62 00   ·····{··'··I··b·
0010  00 00 00 19 11 3a 26 07  f8 b0 40 0a 08 04 00 00   ·····:&···@·····
0020  00 00 00 00 20 0e 26 01  06 03 4e 80 af b0 60 ca   ······&···N···`·
0030  36 f6 6c f8 ed dc 01 bb  ee 28 00 19 54 16 41 01   6·l······(··T·A·
0040  61 46 ad f4 e2 ef d4 75  39 af 8f 10 ac c0 9b      aF·····u9······
```

Ethernet 3: <live capture in progress>   Packets: 43459 · Displayed: 43459 (100.0%)   Profile: Default

How to Install Wireshark on Linux

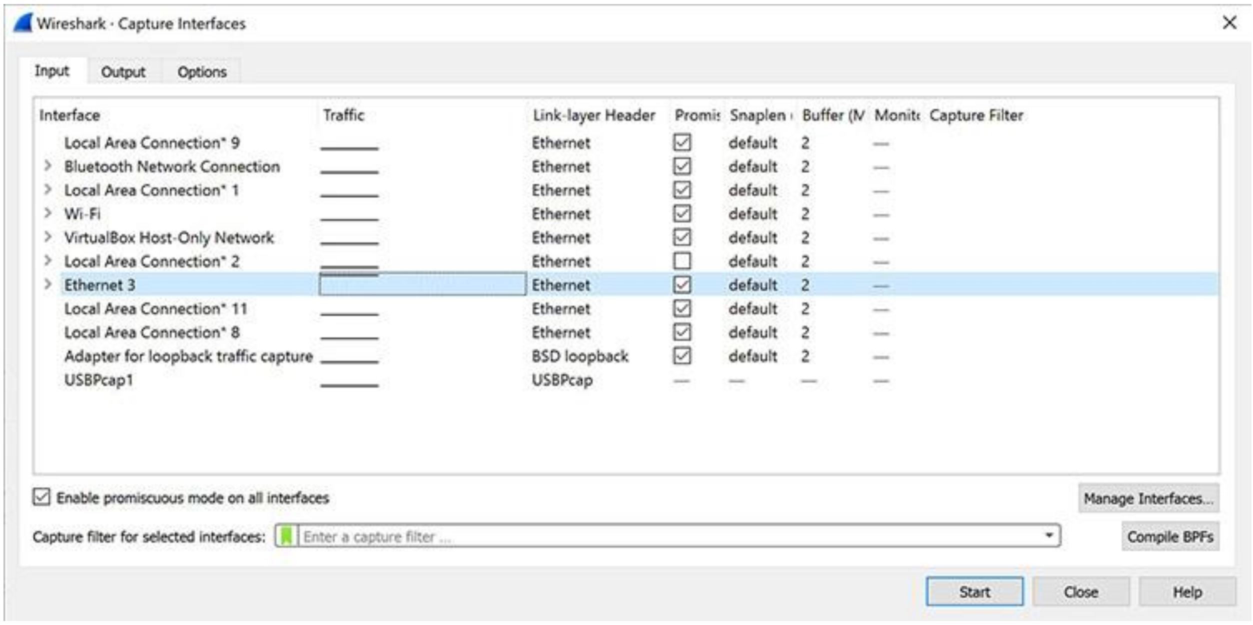
- If you have a Linux system, you'd install Wireshark using the following sequence (notice that you'll need to have root permissions):
- $ sudo apt-get install wireshark
- $ sudo dpkg-reconfigure wireshark-common
- $ sudo usermod -a -G wireshark $USER
- $ newgrp wireshark
- Once you have completed the above steps, you then log out and log back in, and then start Wireshark:
- $ wireshark &

How to Capture Packets Using Wireshark

Once you've installed Wireshark, you can start grabbing network traffic. But remember: To capture any packets, you need to have proper permissions on your computer to put Wireshark into promiscuous mode.

- In a Windows system, this usually means you have administrator access.
- In a Linux system, it usually means that you have root access.

As long as you have the right permissions, you have several options to actually start the capture. Perhaps the best is to select Capture >> Options from the main window. This will bring up the Capture Interfaces window, as shown below.



Interface	Traffic	Link-layer Header	Promis	Snaplen	Buffer (M	Monit	Capture Filter
Local Area Connection* 9		Ethernet	☑	default	2	—	
Bluetooth Network Connection		Ethernet	☑	default	2	—	
Local Area Connection* 1		Ethernet	☑	default	2	—	
Wi-Fi		Ethernet	☑	default	2	—	
VirtualBox Host-Only Network		Ethernet	☑	default	2	—	
Local Area Connection* 2		Ethernet	□	default	2	—	
Ethernet 3		Ethernet	☑	default	2	—	
Local Area Connection* 11		Ethernet	☑	default	2	—	
Local Area Connection* 8		Ethernet	☑	default	2	—	
Adapter for loopback traffic capture		BSD loopback	☑	default	2	—	
USBPcap1		USBPcap	—	—	—	—	

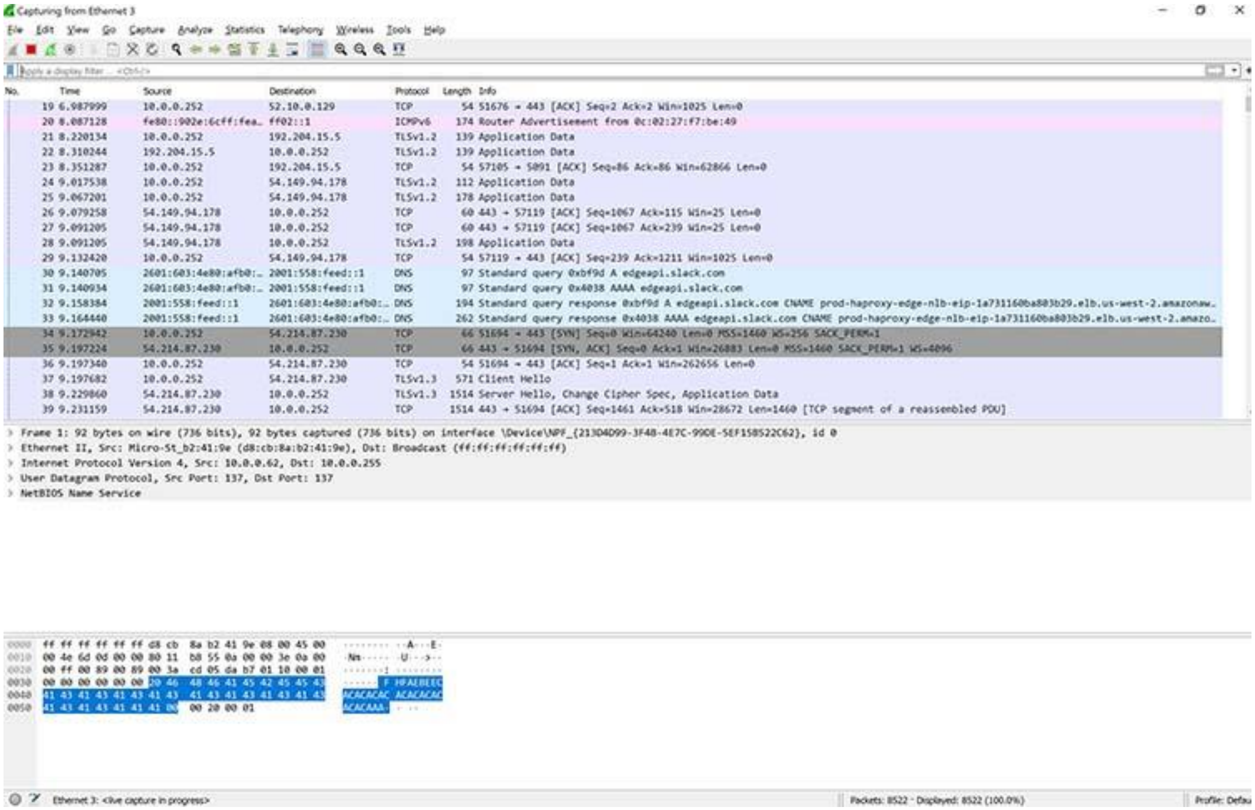
☑ Enable promiscuous mode on all interfaces

Capture filter for selected interfaces: Enter a capture filter ...

This window will list all available interfaces. In this case, Wireshark provides several to choose from.

For this example, we'll select the Ethernet 3 interface, which is the most active interface. Wireshark visualizes the traffic by showing a moving line, which represents the packets on the network.

Once the network interface is selected, you simply click the Start button to begin your capture. As the capture begins, it's possible to view the packets that appear on the screen.


Capturing from Ethernet 3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Ethernet 3: <live capture in progress>
Packets: 8522 · Displayed: 8522 (100.0%)
Profile: Defau

```
No.   Time        Source                  Destination             Protocol  Length Info
   19 6.987999    10.0.0.252              52.10.0.129             TCP         54 51676 → 443 [ACK] Seq=2 Ack=2 Win=1025 Len=0
   20 8.087128    fe80::902e:6cff:fea…    ff02::1                 ICMPv6     174 Router Advertisement from 0c:02:27:f7:be:49
   21 8.220134    10.0.0.252              192.204.15.5            TLSv1.2    139 Application Data
   22 8.310244    192.204.15.5            10.0.0.252              TLSv1.2    139 Application Data
   23 8.351287    10.0.0.252              192.204.15.5            TCP         54 57105 → 5091 [ACK] Seq=86 Ack=86 Win=62866 Len=0
   24 9.017538    10.0.0.252              54.149.94.178           TLSv1.2    112 Application Data
   25 9.067201    10.0.0.252              54.149.94.178           TLSv1.2    178 Application Data
   26 9.079258    54.149.94.178           10.0.0.252              TCP         60 443 → 57119 [ACK] Seq=1067 Ack=115 Win=25 Len=0
   27 9.091205    54.149.94.178           10.0.0.252              TCP         60 443 → 57119 [ACK] Seq=1067 Ack=239 Win=25 Len=0
   28 9.091205    54.149.94.178           10.0.0.252              TLSv1.2    198 Application Data
   29 9.132420    10.0.0.252              54.149.94.178           TCP         54 57119 → 443 [ACK] Seq=239 Ack=1211 Win=1025 Len=0
   30 9.140705    2601:603:4e80:afb0:…    2001:558:feed::1        DNS         97 Standard query 0xbf9d A edgeapi.slack.com
   31 9.140934    2601:603:4e80:afb0:…    2001:558:feed::1        DNS         97 Standard query 0x4038 AAAA edgeapi.slack.com
   32 9.158384    2001:558:feed::1        2601:603:4e80:afb0:…    DNS        194 Standard query response 0xbf9d A edgeapi.slack.com CNAME prod-haproxy-edge-nlb-eip-1a731160ba803b29.elb.us-west-2.amazonaw…
   33 9.164440    2001:558:feed::1        2601:603:4e80:afb0:…    DNS        262 Standard query response 0x4038 AAAA edgeapi.slack.com CNAME prod-haproxy-edge-nlb-eip-1a731160ba803b29.elb.us-west-2.amazo…
   34 9.172942    10.0.0.252              54.214.87.230           TCP         66 51694 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   35 9.197224    54.214.87.230           10.0.0.252              TCP         66 443 → 51694 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=4096
   36 9.197340    10.0.0.252              54.214.87.230           TCP         54 51694 → 443 [ACK] Seq=1 Ack=1 Win=262656 Len=0
   37 9.197682    10.0.0.252              54.214.87.230           TLSv1.3    571 Client Hello
   38 9.229860    54.214.87.230           10.0.0.252              TLSv1.3   1514 Server Hello, Change Cipher Spec, Application Data
   39 9.231159    54.214.87.230           10.0.0.252              TCP       1514 443 → 51694 [ACK] Seq=1461 Ack=518 Win=28672 Len=1460 [TCP segment of a reassembled PDU]

> Frame 1: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface \Device\NPF_{213D4D99-3F48-4E7C-99DE-5EF158522C62}, id 0
> Ethernet II, Src: Micro-St_b2:41:9e (d8:cb:8a:b2:41:9e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 10.0.0.62, Dst: 10.0.0.255
> User Datagram Protocol, Src Port: 137, Dst Port: 137
> NetBIOS Name Service

0000  ff ff ff ff ff ff d8 cb  8a b2 41 9e 08 00 45 00   ··········A···E·
0010  00 4e 6d 0d 00 00 80 11  b8 55 0a 00 00 3e 0a 00   ·Nm······U···>··
0020  00 ff 00 89 00 89 00 3a  cd 05 da b7 01 10 00 01   ·······:········
0030  00 00 00 00 00 00 20 46  48 46 41 45 42 45 45 43   ······ FHFAEBEEC
0040  41 43 41 43 41 43 41 43  41 43 41 43 41 43 41 43   ACACACAC ACACACAC
0050  41 43 41 43 41 41 41 00  00 20 00 01               ACACAAA· ····
```

Once you have captured all the packets that you want, simply click the red, square button at the top. Now you have a static packet capture to investigate.

How Can We Capture Credentials And Other INFO?

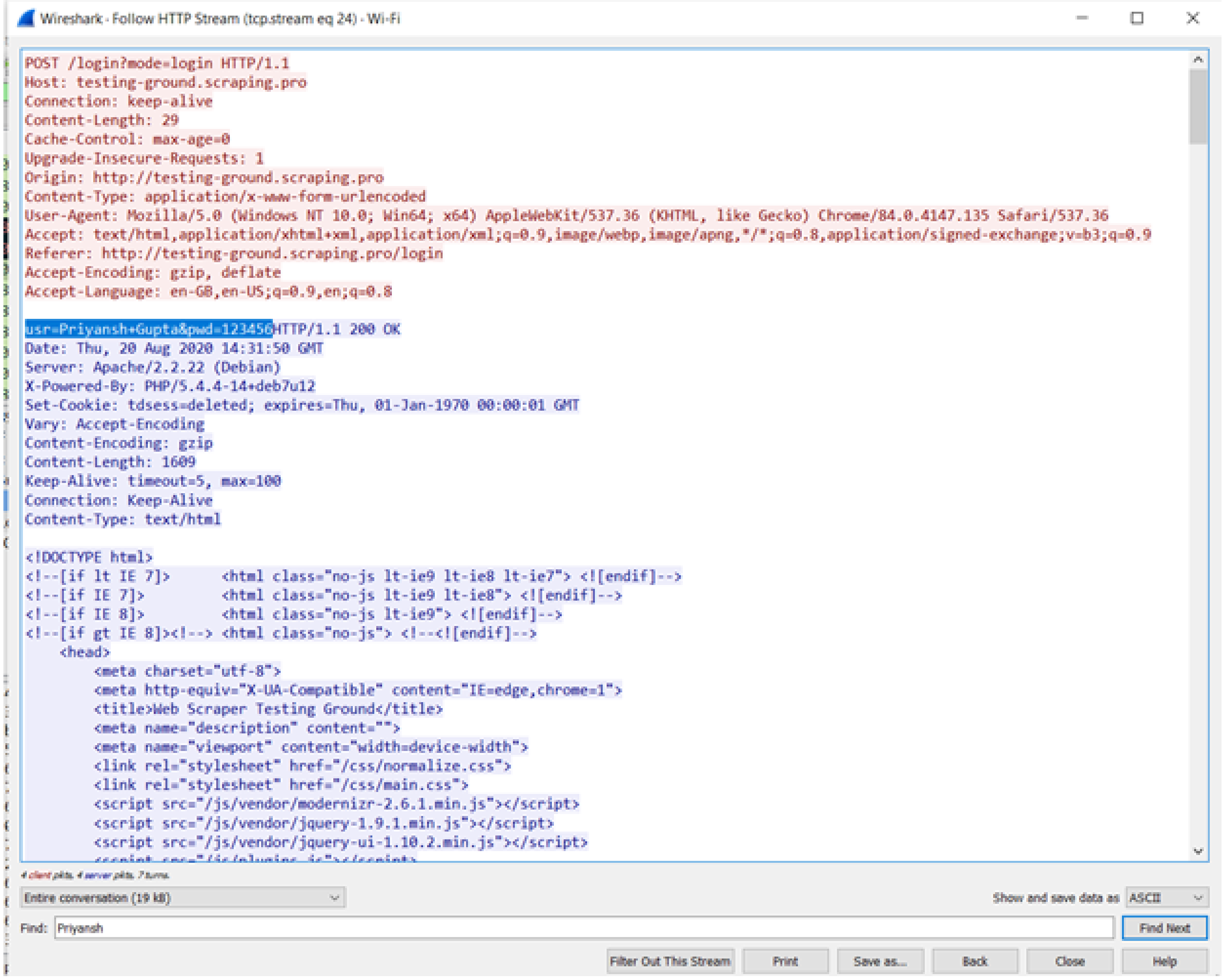
Generally, In A Vulnerable Web App, The Protocol Use Is A Simple HTTP (Not HTTPS) Or An HTTPS GET Method. These Protocols Send Data From One End To The Other Without Applying Any Type Of Encryption Or Other Methods To Hide Data In A Simple Plain Text. Now All Of This Data Is Bound To Exit From A Choke Point In A Network. When An Attacker Monitors Traffic Of This Choke Point, And When He Sniffs The Packets, He Can Easily See All The Relevant Info Via A Network Monitoring Tool Like Wireshark.

We Will Now Try To Capture Credentials Using Wireshark.

1) Capturing Passwords in Insecure Web Apps Using Wireshark:

- Start Wireshark, Select Appropriate Network Adapter (Wi-Fi) In Our Case
- Now Click On The Small Shark Icon At Top Left Of The Screen To Start Capturing The Packets.
- Now Open The Web Browser And Visit Any Insecure Site.
- Enter Credential Info To Login.
- Now Stop Capturing The Packets.
- Now In Wireshark , Go To Edit->Find Packet

- Select Packet Type To Packet Details And Type To String.
- Search For The Phrase ‘Pwd’ Or ‘Pass’ Or ‘Password’.
- Right Click On The Found Packet And Click Follow Ipv4 Stream.
- You Can Now Analyze And See The Username And Password You Entered.



Wireshark · Follow HTTP Stream (tcp.stream eq 24) · Wi-Fi

```
POST /login?mode=login HTTP/1.1
Host: testing-ground.scraping.pro
Connection: keep-alive
Content-Length: 29
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://testing-ground.scraping.pro
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.135 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://testing-ground.scraping.pro/login
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8

usr=Priyansh+Gupta&pwd=123456HTTP/1.1 200 OK
Date: Thu, 20 Aug 2020 14:31:50 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.4-14+deb7u12
Set-Cookie: tdsess=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1609
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <title>Web Scraper Testing Ground</title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width">
        <link rel="stylesheet" href="/css/normalize.css">
        <link rel="stylesheet" href="/css/main.css">
        <script src="/js/vendor/modernizr-2.6.1.min.js"></script>
        <script src="/js/vendor/jquery-1.9.1.min.js"></script>
        <script src="/js/vendor/jquery-ui-1.10.2.min.js"></script>
        <script src="/js/plugins.js"></script>
```

4 client pkts, 4 server pkts, 7 turns.

Entire conversation (19 kB)

Show and save data as: ASCII

Find: Priyansh

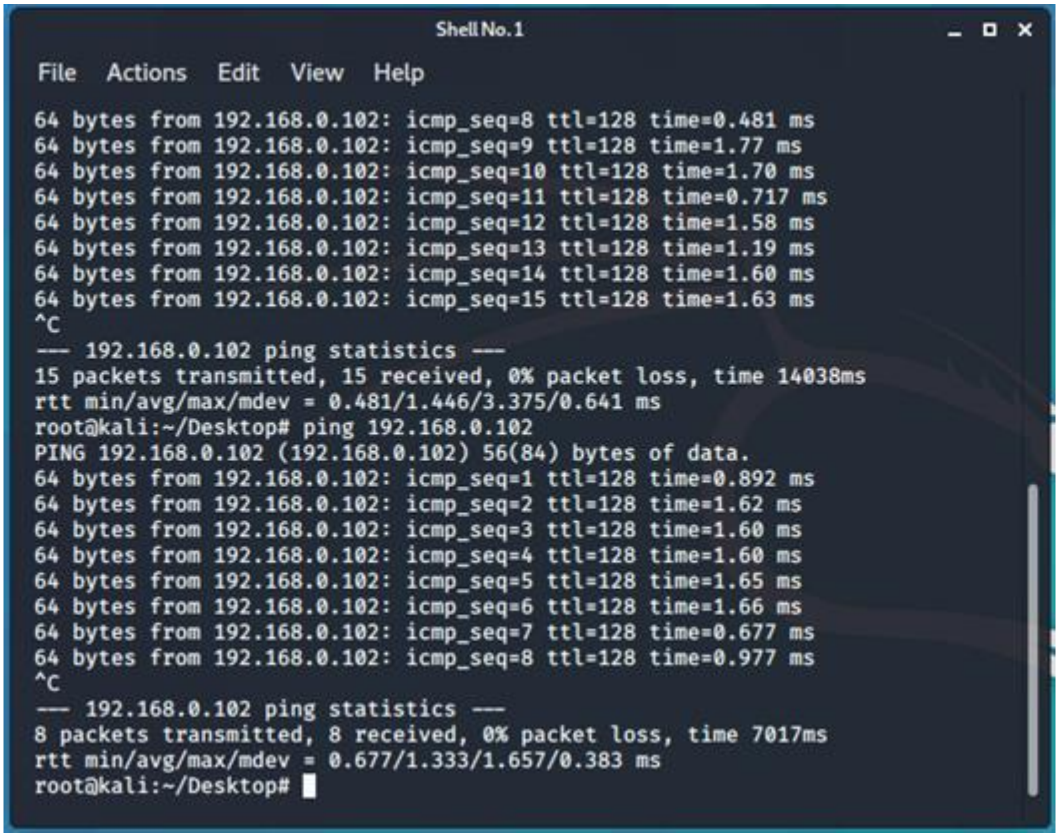
As Seen In The Above Screenshot , The First Line Mentions The Request Method And The Protocol Used. It States That A POST Method With An HTTP Protocol Was Make To The Server , Which Sent The Credentials In A Plain Text Instead In An Encrypt Format.

Capturing OS Info And Packet Details While Pinging :

We Can Also Use Wireshark To Capture Different Packets Having Different Protocols Other Than HTTP Or HTTPS. For Instance , We Will Now Capture Traffic Generated Via A “PING” Command, While Pinging PC A To PC B. The Ping Command Uses The ICMP Protocol.

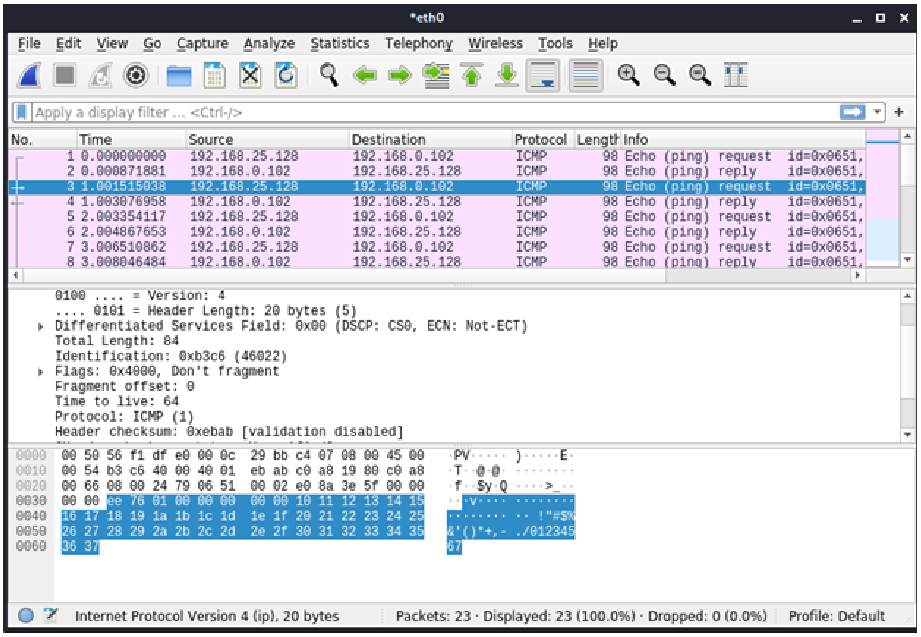
- Start Kali Linux Vm.
- Then, Start Wireshark In Kali Linux.
- Start Capturing The Packets By Selecting Appropriate Network Adapter And By Clicking The Shark Icon At The Top Left Of The Screen.

- Open Terminal And Type “Ping Ipaddress_of_windows_machine” To Ping The Windows Machine From Kali Linux.



```
64 bytes from 192.168.0.102: icmp_seq=8 ttl=128 time=0.481 ms
64 bytes from 192.168.0.102: icmp_seq=9 ttl=128 time=1.77 ms
64 bytes from 192.168.0.102: icmp_seq=10 ttl=128 time=1.70 ms
64 bytes from 192.168.0.102: icmp_seq=11 ttl=128 time=0.717 ms
64 bytes from 192.168.0.102: icmp_seq=12 ttl=128 time=1.58 ms
64 bytes from 192.168.0.102: icmp_seq=13 ttl=128 time=1.19 ms
64 bytes from 192.168.0.102: icmp_seq=14 ttl=128 time=1.60 ms
64 bytes from 192.168.0.102: icmp_seq=15 ttl=128 time=1.63 ms
^C
--- 192.168.0.102 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14038ms
rtt min/avg/max/mdev = 0.481/1.446/3.375/0.641 ms
root@kali:~/Desktop# ping 192.168.0.102
PING 192.168.0.102 (192.168.0.102) 56(84) bytes of data.
64 bytes from 192.168.0.102: icmp_seq=1 ttl=128 time=0.892 ms
64 bytes from 192.168.0.102: icmp_seq=2 ttl=128 time=1.62 ms
64 bytes from 192.168.0.102: icmp_seq=3 ttl=128 time=1.60 ms
64 bytes from 192.168.0.102: icmp_seq=4 ttl=128 time=1.60 ms
64 bytes from 192.168.0.102: icmp_seq=5 ttl=128 time=1.65 ms
64 bytes from 192.168.0.102: icmp_seq=6 ttl=128 time=1.66 ms
64 bytes from 192.168.0.102: icmp_seq=7 ttl=128 time=0.677 ms
64 bytes from 192.168.0.102: icmp_seq=8 ttl=128 time=0.977 ms
^C
--- 192.168.0.102 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7017ms
rtt min/avg/max/mdev = 0.677/1.333/1.657/0.383 ms
root@kali:~/Desktop# 
```

- Now Stop Capturing The Packets.
- Open A Packet Having Protocol As ICMP(Request). This Is The Request Packet Sent From Kali Linux To Windows. Analyze The Packet And Lookout For Header Details And Time To Live (TTL). It Should Be 64 To Ensure A Linux Machine.



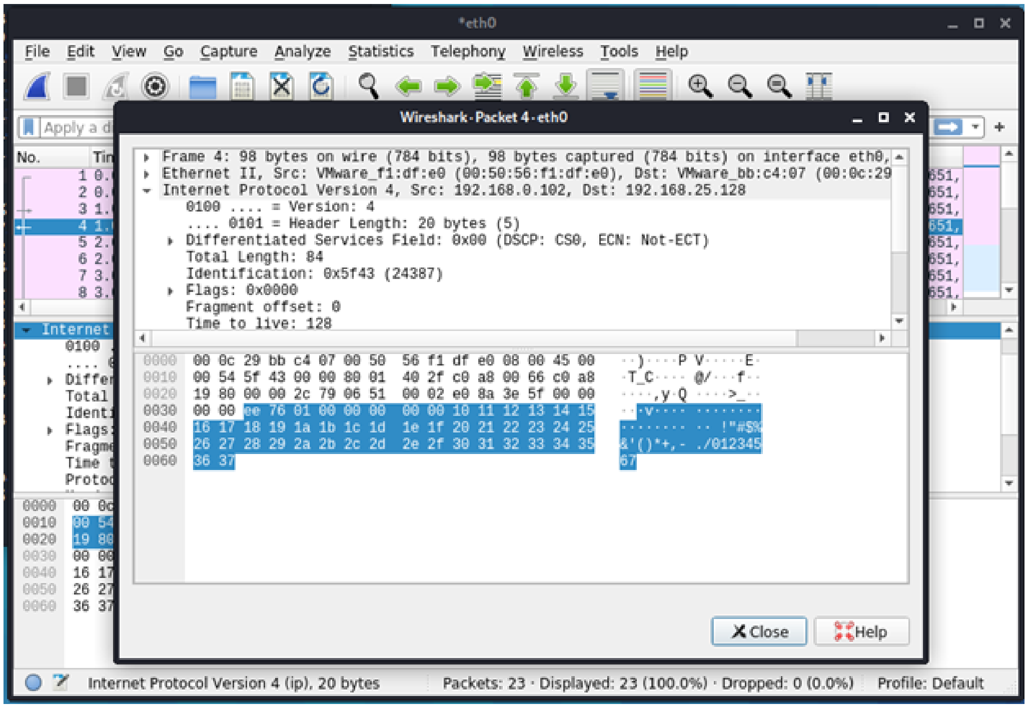
```
No.   Time         Source          Destination     Protocol Length Info
    1 0.000000000  192.168.25.128  192.168.0.102   ICMP       98 Echo (ping) request  id=0x0651,
    2 0.000871881  192.168.0.102   192.168.25.128  ICMP       98 Echo (ping) reply    id=0x0651,
    3 1.001515038  192.168.25.128  192.168.0.102   ICMP       98 Echo (ping) request  id=0x0651,
    4 1.003076958  192.168.0.102   192.168.25.128  ICMP       98 Echo (ping) reply    id=0x0651,
    5 2.003354117  192.168.25.128  192.168.0.102   ICMP       98 Echo (ping) request  id=0x0651,
    6 2.004867653  192.168.0.102   192.168.25.128  ICMP       98 Echo (ping) reply    id=0x0651,
    7 3.006510862  192.168.25.128  192.168.0.102   ICMP       98 Echo (ping) request  id=0x0651,
    8 3.008046484  192.168.0.102   192.168.25.128  ICMP       98 Echo (ping) reply    id=0x0651,
```

```
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xb3c6 (46022)
  ▸ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)
    Header checksum: 0xebab [validation disabled]
```

```
0000  00 50 56 f1 df e0 00 0c  29 bb c4 07 08 00 45 00   ·PV····· )·····E·
0010  00 54 b3 c6 40 00 40 01  eb ab c0 a8 19 80 c0 a8   ·T··@·@· ········
0020  00 66 08 00 24 79 06 51  00 02 e0 8a 3e 5f 00 00   ·f··$y·Q ····>_··
0030  00 00 ee 76 01 00 00 00  00 00 10 11 12 13 14 15   ···v···· ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

Internet Protocol Version 4 (ip), 20 bytes · Packets: 23 · Displayed: 23 (100.0%) · Dropped: 0 (0.0%) · Profile: Default

- Do The Above Step To The ICMP(Reply) Packet. This Is The Reply Packet From Windows. Lookout And Analyze For Time To Live Value(TTL). It Should Be 128 To Ensure That It Is A Windows Machine.


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a d
No. Tin
Wireshark · Packet 4 · eth0
Close
Help

```
▸ Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0,
▸ Ethernet II, Src: VMware_f1:df:e0 (00:50:56:f1:df:e0), Dst: VMware_bb:c4:07 (00:0c:29
▾ Internet Protocol Version 4, Src: 192.168.0.102, Dst: 192.168.25.128
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x5f43 (24387)
  ▸ Flags: 0x0000
    Fragment offset: 0
    Time to live: 128
```

```
0000  00 0c 29 bb c4 07 00 50  56 f1 df e0 08 00 45 00   ··)····P V·····E·
0010  00 54 5f 43 00 00 80 01  40 2f c0 a8 00 66 c0 a8   ·T_C···· @/···f··
0020  19 80 00 00 2c 79 06 51  00 02 e0 8a 3e 5f 00 00   ····,y·Q ····>_··
0030  00 00 ee 76 01 00 00 00  00 00 10 11 12 13 14 15   ···v···· ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

Internet Protocol Version 4 (ip), 20 bytes   Packets: 23 · Displayed: 23 (100.0%) · Dropped: 0 (0.0%)   Profile: Default