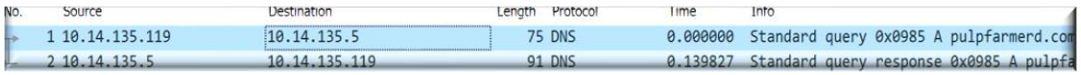
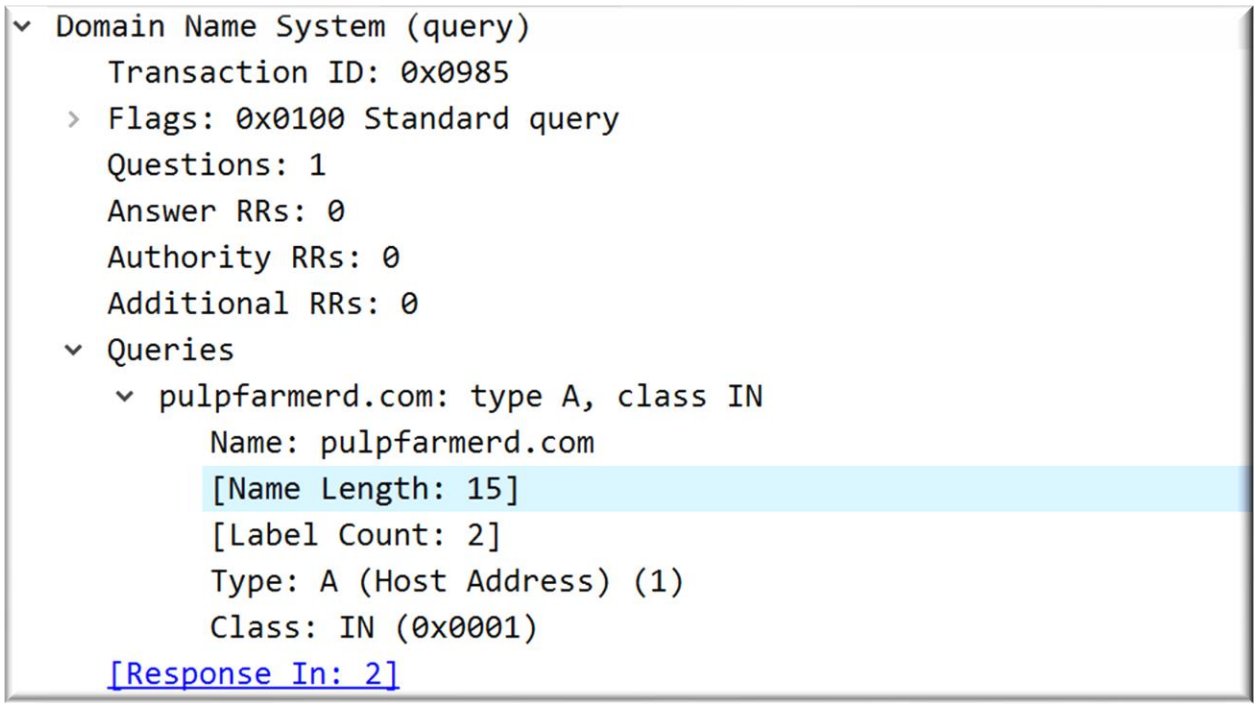
Find 100 things that you have learnt with Wireshark analysis

1. Packets 1 and 2 depicts DNS query and response respectively to DNS server



The client is trying to visit pulpfarmed.com. A request from client located at 10.14.135.119 is sent to the DNS server at 10.14.135.5



2. Flags: 0 means standard query

   Questions: Indicates the number of queries present in the packet. In this case it is 1

   Answers: Indicates the number of answers in response to the query sent.

   Authority RRs: Indicates the number of authority resource records sent as response.

   Additional RRs: Indicates the number of additional resource records sent as response

   Therefore, as per the screenshot above, 1 query was sent in the packet.

3. In response to this query, you will be seeing one more packet with the same transaction ID that denotes the association of a particular query. It is the response packet. Response for our query will usually consist of IPv4 address for the domain we are trying to look for – 45.95.11.201 in our case. This is depicted in the screenshot below.

```
Domain Name System (response)
    Transaction ID: 0x0985
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v pulpfarmerd.com: type A, class IN
        Name: pulpfarmerd.com
        [Name Length: 15]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  v Answers
    > pulpfarmerd.com: type A, class IN, addr 45.95.11.201
    [Request In: 11]
```

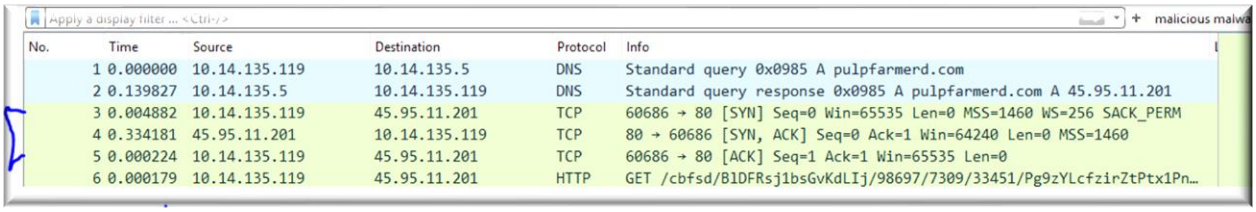
4. At the beginning of every HTTP session, the TCP three-way handshake takes place. It creates a dedicated channel between the communicating hosts followed by HTTP and data packets, which are sent in and received while the session is active. For instance, you are visiting a web server located at http://45.95.11.201 and the client at 10.14.135.119.

5. This is how a request looks. In the first line, there are three things passed on to the server as the arguments, which are HTTP method and requested resource location "/" (root directory) i.e the Host argument which is required by the HTTP/1.1 protocol requests. The value of this field is the webserver's address that you typed in the address bar of the browser.

3	10.14.135.119	45.95.11.201	66	TCP	0.144709	60686 → 80 [SYN] Seq=0 Win=65535 Len=0
4	45.95.11.201	10.14.135.119	58	TCP	0.478890	80 → 60686 [SYN, ACK] Seq=0 Ack=1 Win=6
5	10.14.135.119	45.95.11.201	54	TCP	0.479114	60686 → 80 [ACK] Seq=1 Ack=1 Win=65535
6	10.14.135.119	45.95.11.201	583	HTTP	0.479293	GET /cbfsd/BlDFRsj1bsGvKdLIj/98697/7309
7	45.95.11.201	10.14.135.119	54	TCP	0.479488	80 → 60686 [ACK] Seq=1 Ack=530 Win=6424
8	45.95.11.201	10.14.135.119	1430	TCP	0.801899	80 → 60686 [PSH, ACK] Seq=1 Ack=530 Win
9	10.14.135.119	45.95.11.201	54	TCP	0.802117	60686 → 80 [ACK] Seq=530 Ack=1377 Win=6
10	45.95.11.201	10.14.135.119	1430	TCP	0.804594	80 → 60686 [PSH, ACK] Seq=1377 Ack=530
11	10.14.135.119	45.95.11.201	54	TCP	0.804762	60686 → 80 [ACK] Seq=530 Ack=2753 Win=6
12	45.95.11.201	10.14.135.119	1430	TCP	0.814172	80 → 60686 [PSH, ACK] Seq=2753 Ack=530
13	10.14.135.119	45.95.11.201	54	TCP	0.814378	60686 → 80 [ACK] Seq=530 Ack=4129 Win=6
14	45.95.11.201	10.14.135.119	1514	TCP	0.814509	80 → 60686 [ACK] Seq=4129 Ack=530 Win=6
15	45.95.11.201	10.14.135.119	1346	TCP	0.814514	80 → 60686 [PSH, ACK] Seq=5589 Ack=530
16	45.95.11.201	10.14.135.119	1430	TCP	0.814553	80 → 60686 [PSH, ACK] Seq=6881 Ack=530
17	10.14.135.119	45.95.11.201	54	TCP	0.814711	60686 → 80 [ACK] Seq=530 Ack=8257 Win=6

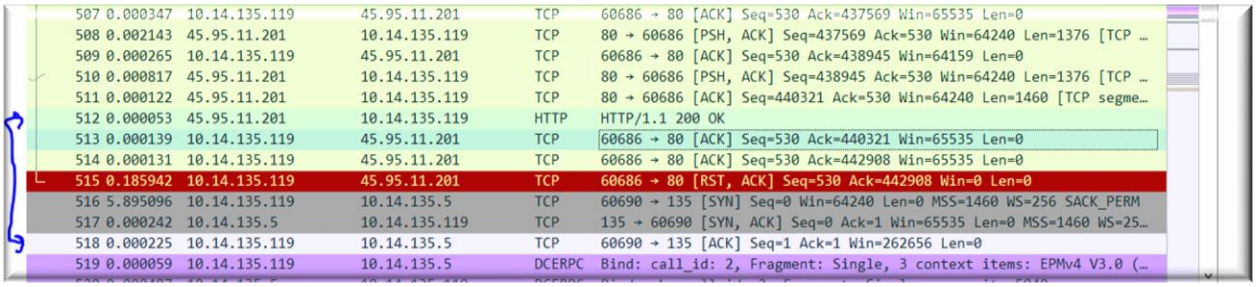
6. In packet number 6, you see that the client performing a GET request as per the screenshot above.

7. There are 11866 TCP packets in the pcap file.



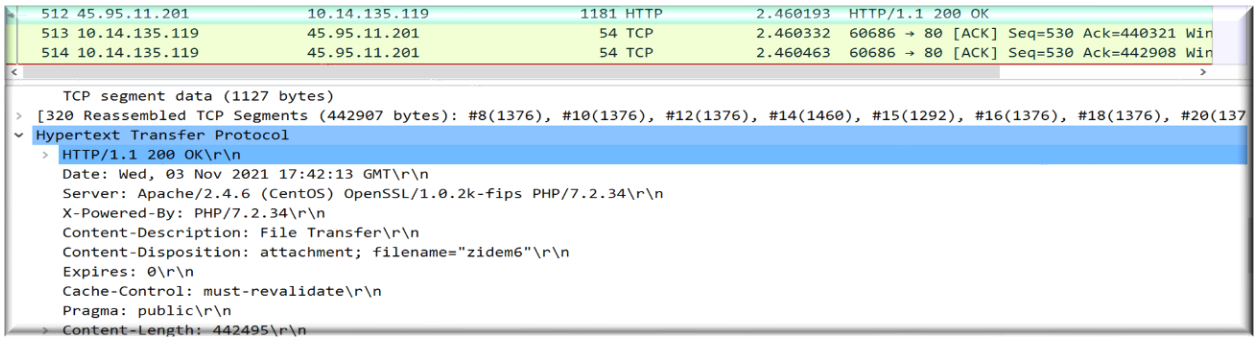
In packets 3, 4, and 5, it is clearly visible that the client and server are trying to create a dedicated channel.

8. The client initiated the creation by sending a SYN packet in the 3 packet with the SEQ set to 0. Since the server was open for communication, the server responded with a SYN/ACK packet with ACK set to 1 and SEQ set to 0.
9. This is followed by a confirmation sent from the client side that is seen in the packet number 5 with SEQ=1 and ACK=1. This is what is known as a three-way handshake process.
10. This can be seen before any real data transfer that happens that follows the TCP approach. After the successful completion of channel creation, the client sends a GET request to access the contents of the web-root directory.

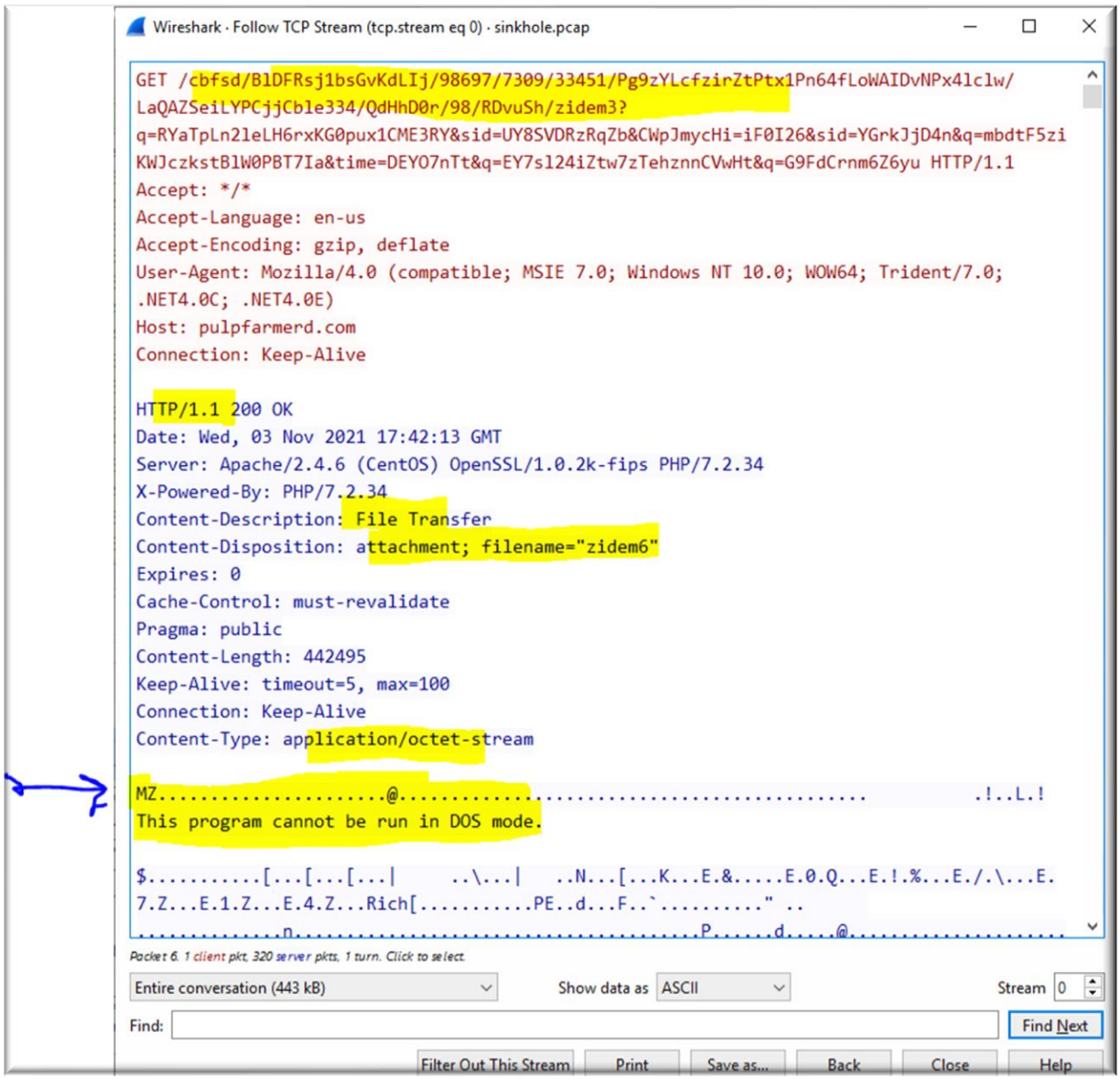


11. The server acknowledged this the packet number 512 but sent the requested content to the client's machine with the 200 OK status message after many packets. Acknowledgement should have happened immediately in the next packets. This looks suspicious.

12. After a few packets were exchanged between the hosts in the later frames, the clients received a 200 OK status message, suggesting successful transfer of the malware at packet number 512. Refer below screenshot:

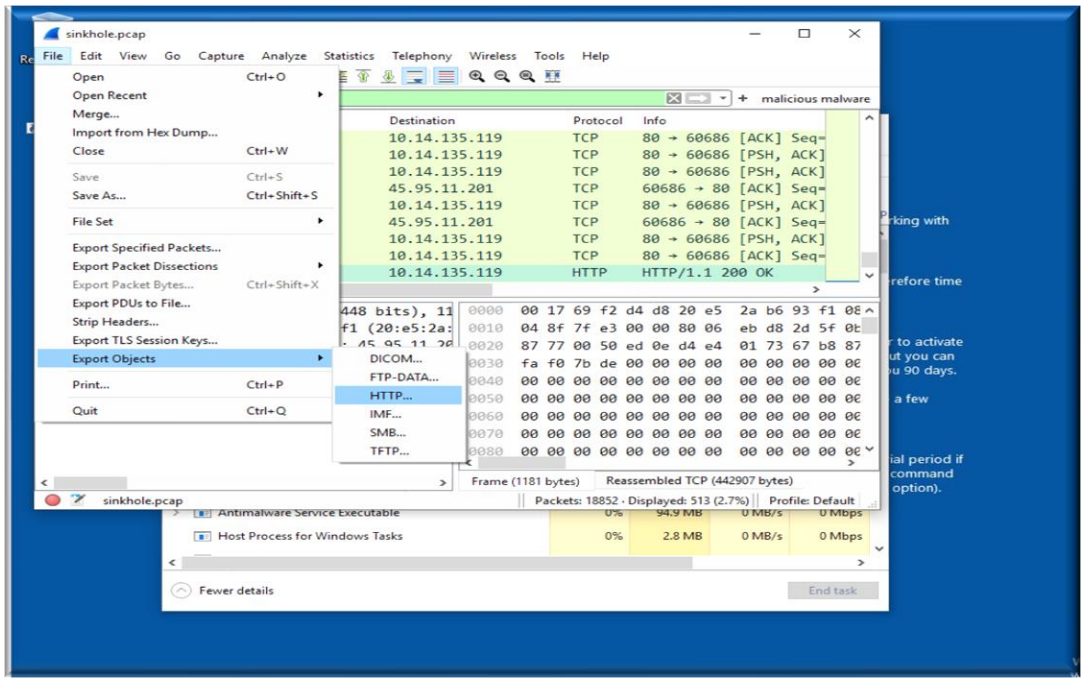


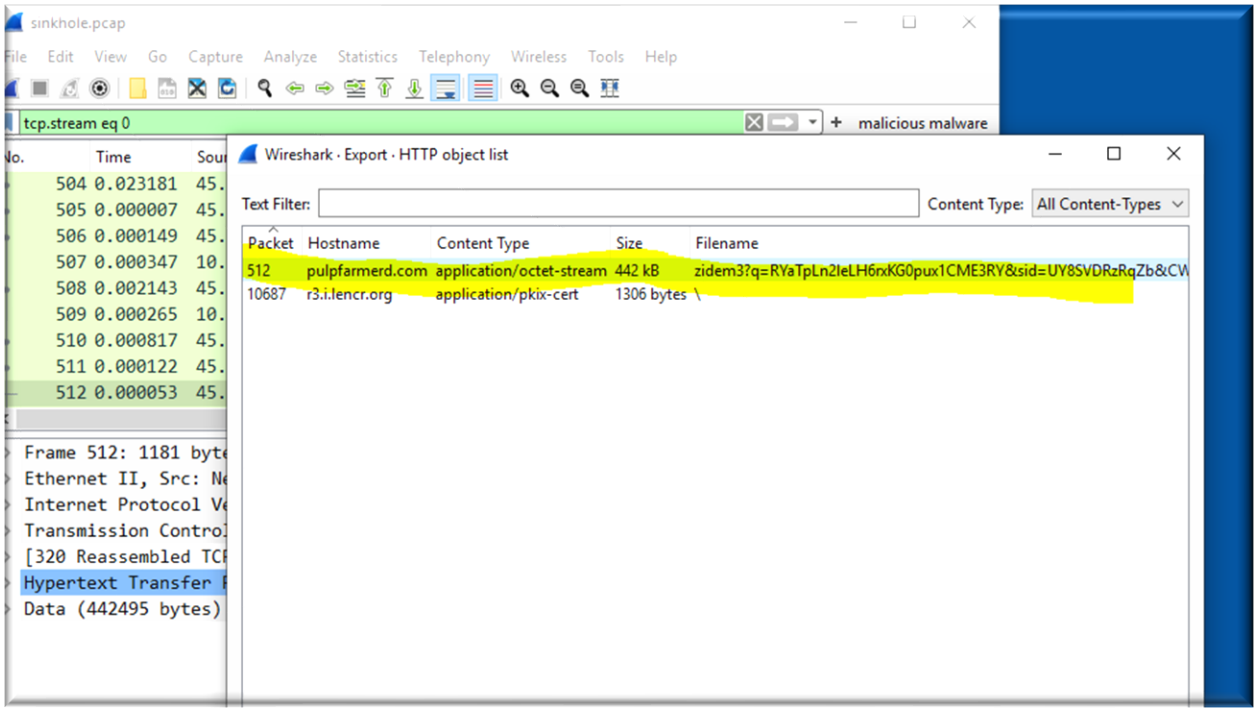
13. Follow the TCP stream for this particular packet and you can see the below details.



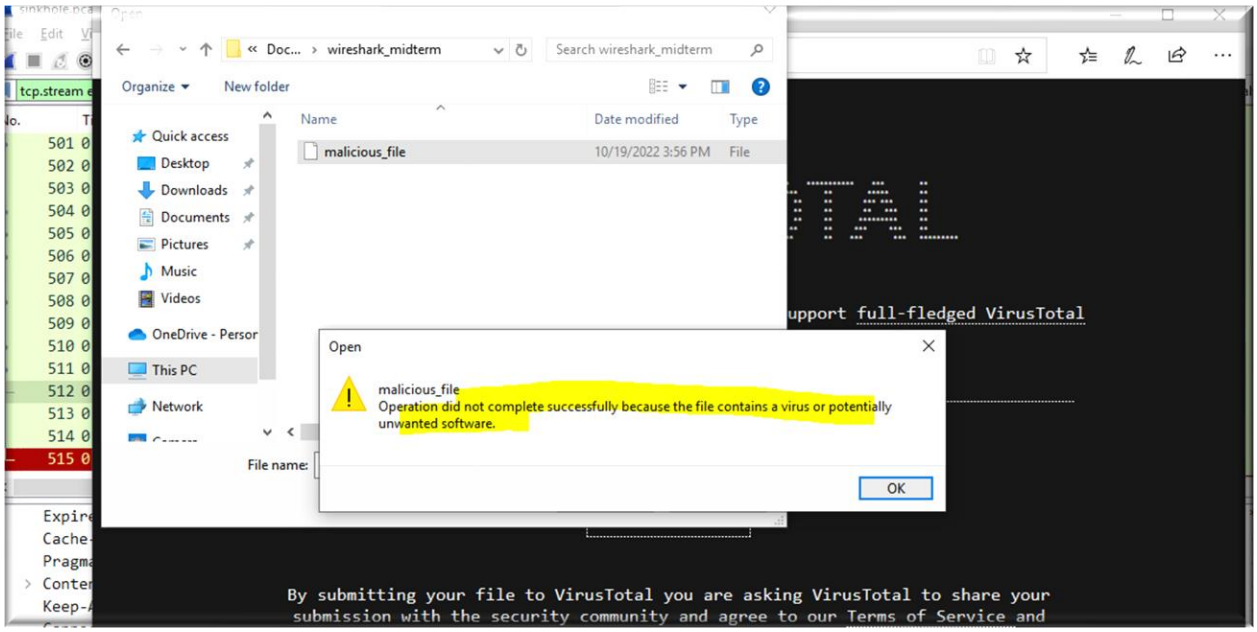
14. The file downloaded as an attachment and name is “zidem6”
    We can observe that this is malicious file by witnessing the malware signature in the screenshot above which starts with “MZ….”

15. Moving on my investigation regarding the malicious file, I exported the file using Wireshark by going to File -> Export Objects -> HTTP. You will see a dialog similar to the one shown below:



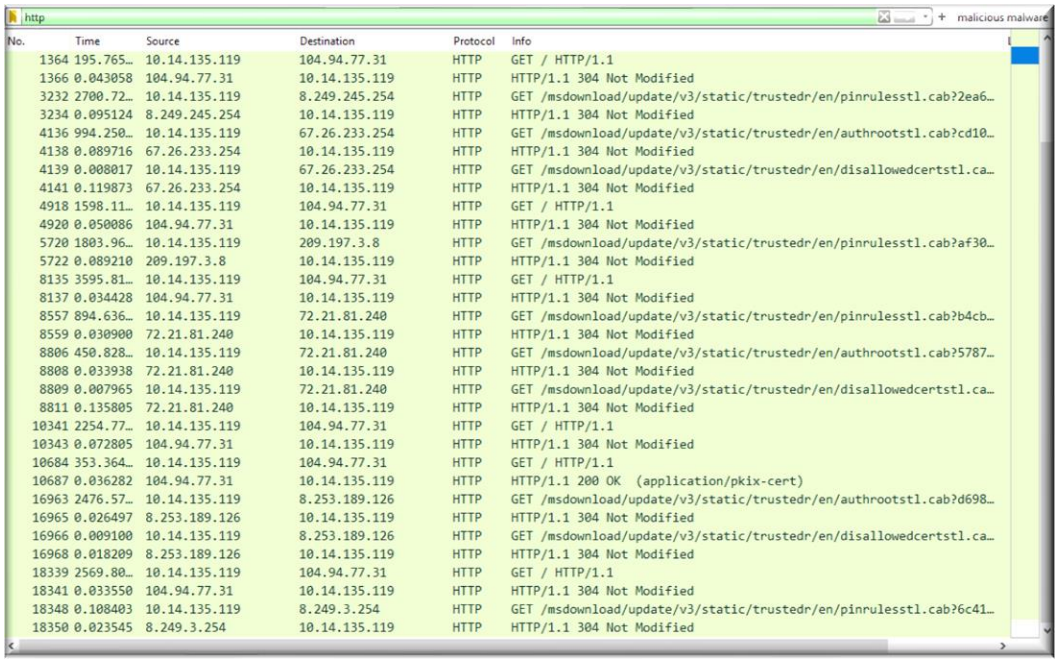


16. I saved it and named it as a malicious file. I tried to open this malicious file in http://www.virustotal.com. It popped up with the below error message.



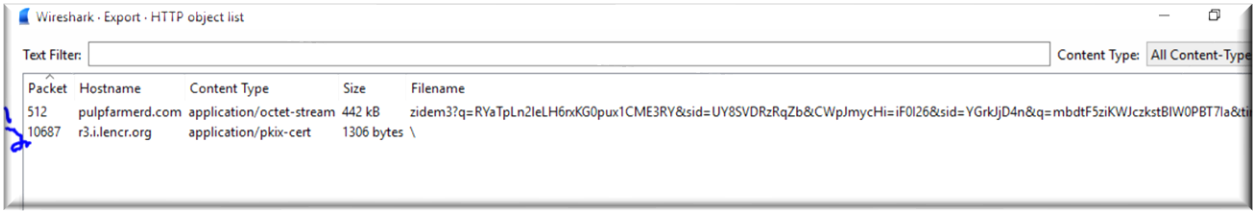
17. The file was deleted in sometime by the antivirus installed on my VM.
18. By this I conclude that, the octet stream content in the form of attachment was a virus file infecting the client machine located at 10.14.135.119.

19. We have 34 packets in total with HTTP protocol. Refer screenshot below:



No.	Time	Source	Destination	Protocol	Info
1364	195.765…	10.14.135.119	104.94.77.31	HTTP	GET / HTTP/1.1
1366	0.043058	104.94.77.31	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
3232	2700.72…	10.14.135.119	8.249.245.254	HTTP	GET /msdownload/update/v3/static/trustedr/en/pinrulesstl.cab?2ea6…
3234	0.095124	8.249.245.254	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
4136	994.250…	10.14.135.119	67.26.233.254	HTTP	GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?cd10…
4138	0.089716	67.26.233.254	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
4139	0.008017	10.14.135.119	67.26.233.254	HTTP	GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.ca…
4141	0.119873	67.26.233.254	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
4918	1598.11…	10.14.135.119	104.94.77.31	HTTP	GET / HTTP/1.1
4920	0.050086	104.94.77.31	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
5720	1803.96…	10.14.135.119	209.197.3.8	HTTP	GET /msdownload/update/v3/static/trustedr/en/pinrulesstl.cab?af30…
5722	0.089210	209.197.3.8	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
8135	3595.81…	10.14.135.119	104.94.77.31	HTTP	GET / HTTP/1.1
8137	0.034428	104.94.77.31	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
8557	894.636…	10.14.135.119	72.21.81.240	HTTP	GET /msdownload/update/v3/static/trustedr/en/pinrulesstl.cab?b4cb…
8559	0.030900	72.21.81.240	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
8806	450.828…	10.14.135.119	72.21.81.240	HTTP	GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?5787…
8808	0.033938	72.21.81.240	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
8809	0.007965	10.14.135.119	72.21.81.240	HTTP	GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.ca…
8811	0.135805	72.21.81.240	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
10341	2254.77…	10.14.135.119	104.94.77.31	HTTP	GET / HTTP/1.1
10343	0.072805	104.94.77.31	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
10684	353.364…	10.14.135.119	104.94.77.31	HTTP	GET / HTTP/1.1
10687	0.036282	104.94.77.31	10.14.135.119	HTTP	HTTP/1.1 200 OK (application/pkix-cert)
16963	2476.57…	10.14.135.119	8.253.189.126	HTTP	GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?d698…
16965	0.026497	8.253.189.126	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
16966	0.009100	10.14.135.119	8.253.189.126	HTTP	GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.ca…
16968	0.018209	8.253.189.126	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
18339	2569.80…	10.14.135.119	104.94.77.31	HTTP	GET / HTTP/1.1
18341	0.033550	104.94.77.31	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified
18348	0.108403	10.14.135.119	8.249.3.254	HTTP	GET /msdownload/update/v3/static/trustedr/en/pinrulesstl.cab?6c41…
18350	0.023545	8.249.3.254	10.14.135.119	HTTP	HTTP/1.1 304 Not Modified

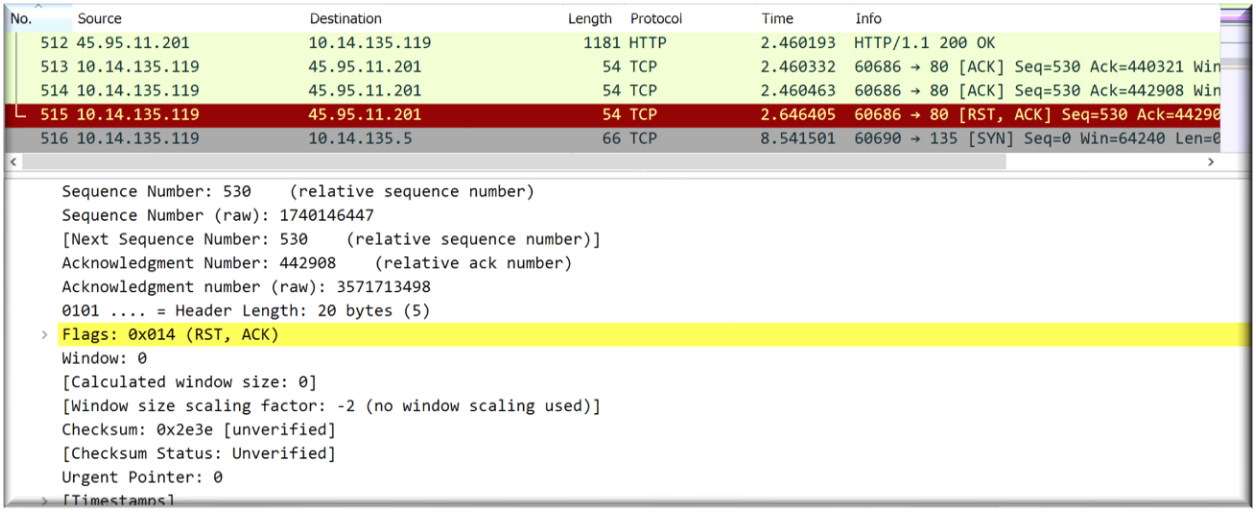
20. I tried to follow TCP stream for every HTTP GET request and found 2 HTTP objects as shown in the screenshot below:



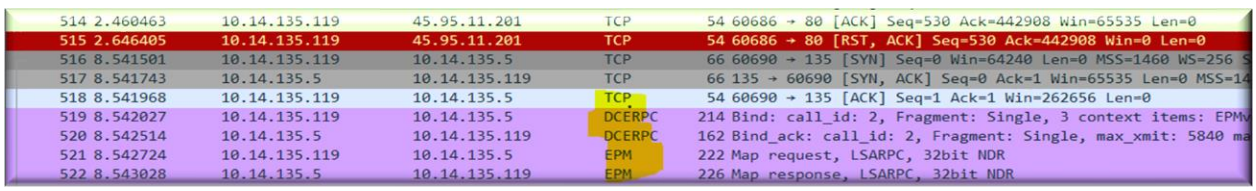
1 is the malicious file and 2nd is the pkix certificate which seems to be clean.

21. TCP reset abrupt disconnection RST, ACK error in the packet line number 515.
because the server daemon is not running or because of any reason the server is not accepting our requests, then we could have seen RST ACK packets.
This error states that the server is out of service or perhaps the server is not supposed to respond to our requests.

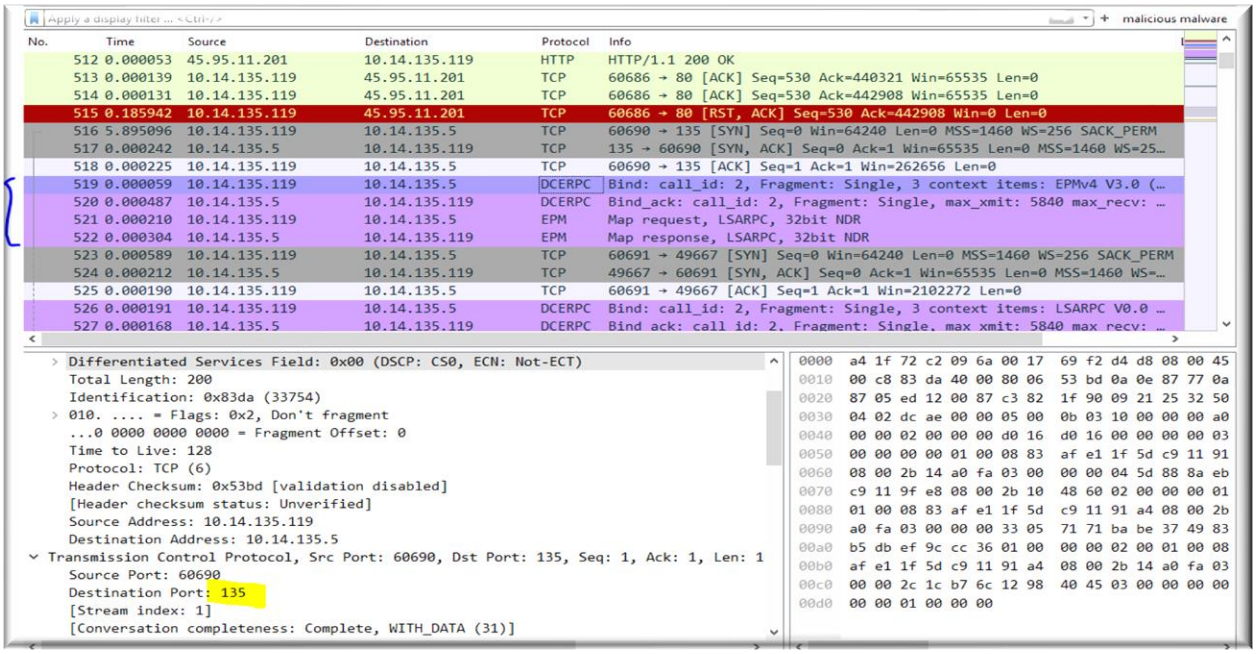
Refer screenshot below:



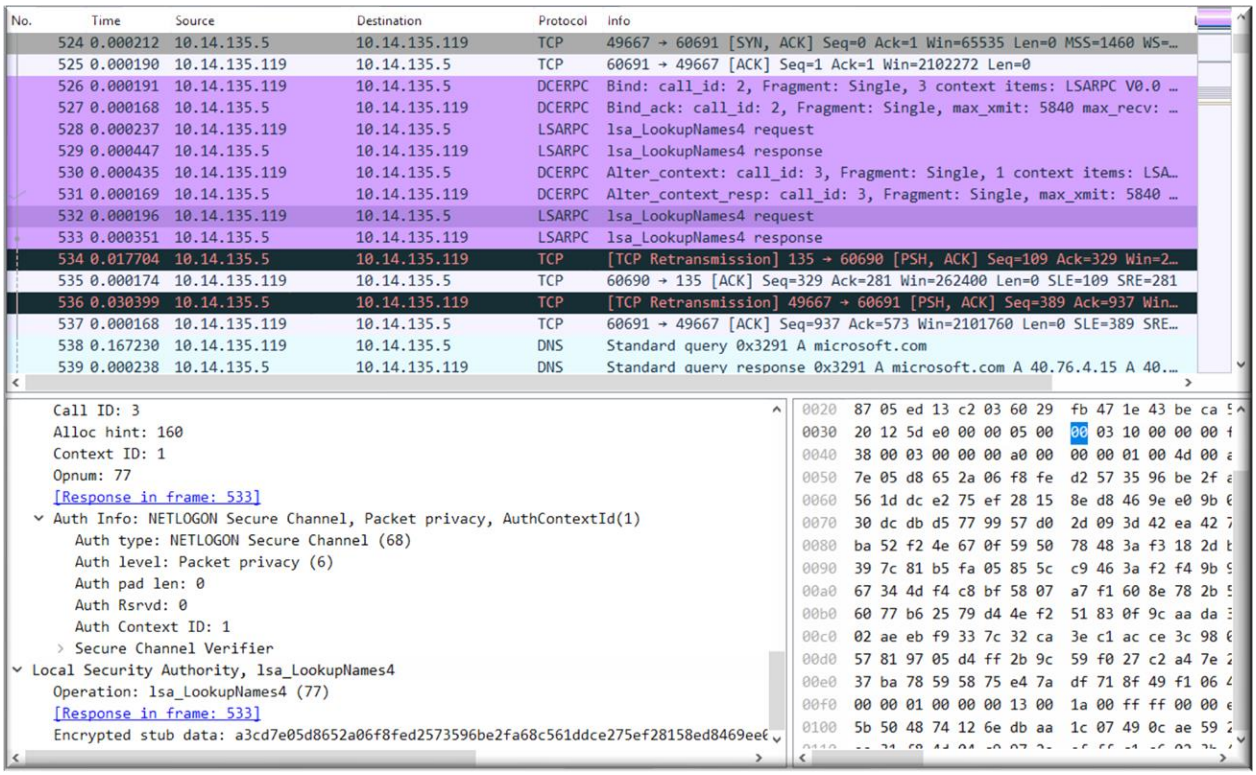
22. After drop, again we see 3 way TCP communication and the client sends an ACK to the server in packet 518.

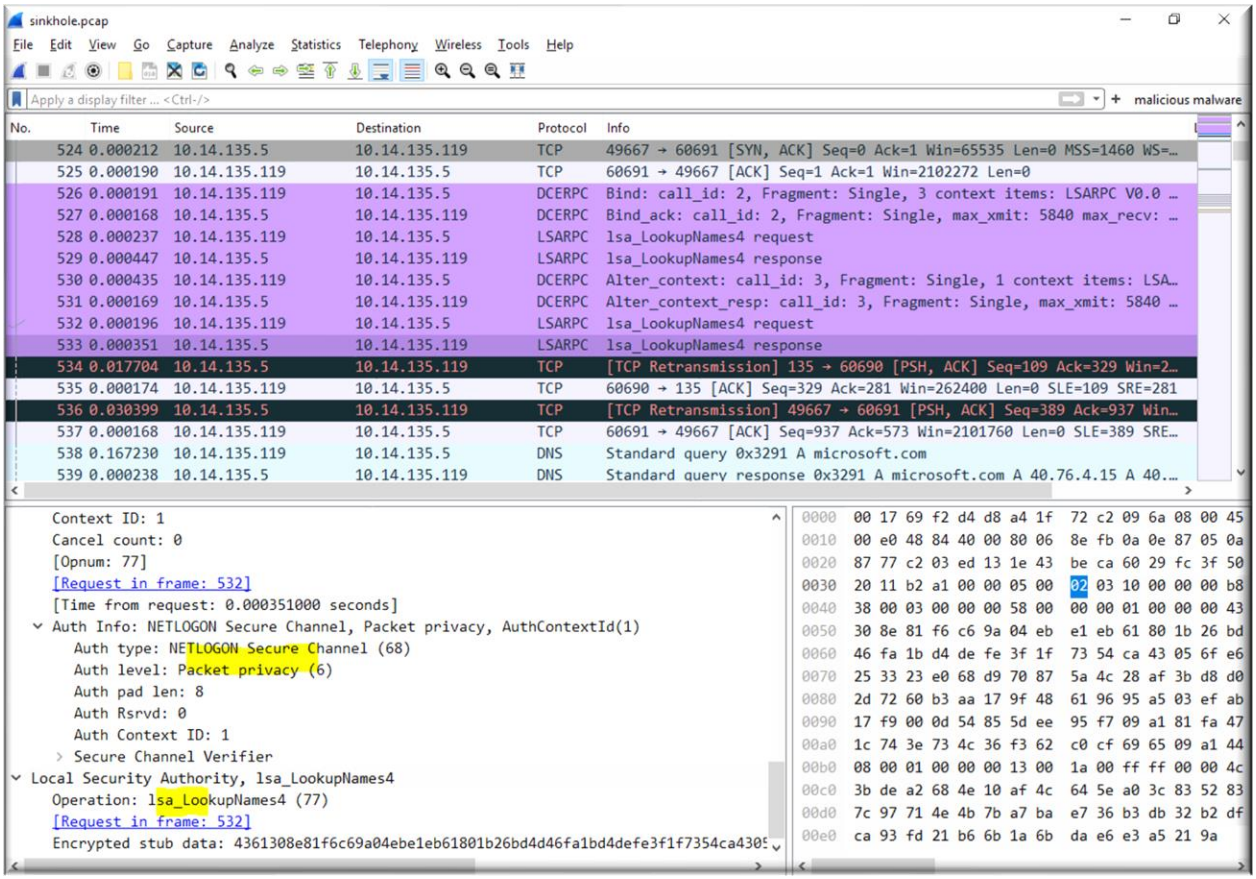


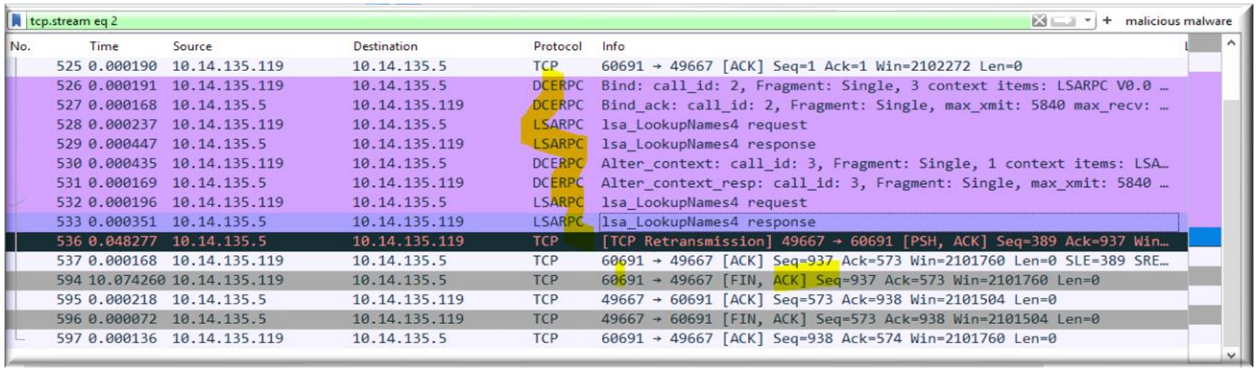
23. Post acknowledgement we see DCERPC protocol running on top of TCP using default TCP port of 135.



24. Connection oriented DCE/RPC uses TCP as its transport protocol.
    DCE/RPC is a specification for a remote procedure call mechanism that defines both APIs and an over-the-network protocol.
25. A DCE/RPC server's endpoint mapper (EPMAP) will listen for incoming calls. A client will call this endpoint mapper and ask for a specific interface, which will be accessed on a different connection. After that, the client can request calls to the server.
    The well known TCP port for DCE/RPC EPMAP is 135. This transport is called ncacn_ip_tcp


No. Time Source Destination Protocol Info
524 0.000212 10.14.135.5 10.14.135.119 TCP 49667 → 60691 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=…
525 0.000190 10.14.135.119 10.14.135.5 TCP 60691 → 49667 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
526 0.000191 10.14.135.119 10.14.135.5 DCERPC Bind: call_id: 2, Fragment: Single, 3 context items: LSARPC V0.0 …
527 0.000168 10.14.135.5 10.14.135.119 DCERPC Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_recv: …
528 0.000237 10.14.135.119 10.14.135.5 LSARPC lsa_LookupNames4 request
529 0.000447 10.14.135.5 10.14.135.119 LSARPC lsa_LookupNames4 response
530 0.000435 10.14.135.119 10.14.135.5 DCERPC Alter_context: call_id: 3, Fragment: Single, 1 context items: LSA…
531 0.000169 10.14.135.5 10.14.135.119 DCERPC Alter_context_resp: call_id: 3, Fragment: Single, max_xmit: 5840 …
532 0.000196 10.14.135.119 10.14.135.5 LSARPC lsa_LookupNames4 request
533 0.000351 10.14.135.5 10.14.135.119 LSARPC lsa_LookupNames4 response
534 0.017704 10.14.135.5 10.14.135.119 TCP [TCP Retransmission] 135 → 60690 [PSH, ACK] Seq=109 Ack=329 Win=2…
535 0.000174 10.14.135.119 10.14.135.5 TCP 60690 → 135 [ACK] Seq=329 Ack=281 Win=262400 Len=0 SLE=109 SRE=281
536 0.030399 10.14.135.5 10.14.135.119 TCP [TCP Retransmission] 49667 → 60691 [PSH, ACK] Seq=389 Ack=937 Win…
537 0.000168 10.14.135.119 10.14.135.5 TCP 60691 → 49667 [ACK] Seq=937 Ack=573 Win=2101760 Len=0 SLE=389 SRE…
538 0.167230 10.14.135.119 10.14.135.5 DNS Standard query 0x3291 A microsoft.com
539 0.000238 10.14.135.5 10.14.135.119 DNS Standard query response 0x3291 A microsoft.com A 40.76.4.15 A 40.…
Call ID: 3
Alloc hint: 160
Context ID: 1
Opnum: 77
[Response in frame: 533]
Auth Info: NETLOGON Secure Channel, Packet privacy, AuthContextId(1)
Auth type: NETLOGON Secure Channel (68)
Auth level: Packet privacy (6)
Auth pad len: 0
Auth Rsrvd: 0
Auth Context ID: 1
Secure Channel Verifier
Local Security Authority, lsa_LookupNames4
Operation: lsa_LookupNames4 (77)
[Response in frame: 533]
Encrypted stub data: a3cd7e05d8652a06f8fed2573596be2fa68c561ddce275ef28158ed8469ee6
0020 87 05 ed 13 c2 03 60 29 fb 47 1e 43 be ca
0030 20 12 5d e0 00 00 05 00 00 03 10 00 00 00
0040 38 00 03 00 00 00 a0 00 00 00 01 00 4d 00
0050 7e 05 d8 65 2a 06 f8 fe d2 57 35 96 be 2f
0060 56 1d dc e2 75 ef 28 15 8e d8 46 9e e0 9b
0070 30 dc db d5 77 99 57 d0 2d 09 3d 42 ea 42
0080 ba 52 f2 4e 67 0f 59 50 78 48 3a f3 18 2d
0090 39 7c 81 b5 fa 05 85 5c c9 46 3a f2 f4 9b
00a0 67 34 4d f4 c8 bf 58 07 a7 f1 60 8e 78 2b
00b0 60 77 b6 25 79 d4 4e f2 51 83 0f 9c aa da
00c0 02 ae eb f9 33 7c 32 ca 3e c1 ac ce 3c 98
00d0 57 81 97 05 d4 ff 2b 9c 59 f0 27 c2 a4 7e
00e0 37 ba 78 59 58 75 e4 7a df 71 8f 49 f1 06
00f0 00 00 01 00 00 00 13 00 1a 00 ff ff 00 00
0100 5b 50 48 74 12 6e db aa 1c 07 49 0c ae 59


sinkhole.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
malicious malware
No. Time Source Destination Protocol Info
524 0.000212 10.14.135.5 10.14.135.119 TCP 49667 → 60691 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=…
525 0.000190 10.14.135.119 10.14.135.5 TCP 60691 → 49667 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
526 0.000191 10.14.135.119 10.14.135.5 DCERPC Bind: call_id: 2, Fragment: Single, 3 context items: LSARPC V0.0 …
527 0.000168 10.14.135.5 10.14.135.119 DCERPC Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_recv: …
528 0.000237 10.14.135.119 10.14.135.5 LSARPC lsa_LookupNames4 request
529 0.000447 10.14.135.5 10.14.135.119 LSARPC lsa_LookupNames4 response
530 0.000435 10.14.135.119 10.14.135.5 DCERPC Alter_context: call_id: 3, Fragment: Single, 1 context items: LSA…
531 0.000169 10.14.135.5 10.14.135.119 DCERPC Alter_context_resp: call_id: 3, Fragment: Single, max_xmit: 5840 …
532 0.000196 10.14.135.119 10.14.135.5 LSARPC lsa_LookupNames4 request
533 0.000351 10.14.135.5 10.14.135.119 LSARPC lsa_LookupNames4 response
534 0.017704 10.14.135.5 10.14.135.119 TCP [TCP Retransmission] 135 → 60690 [PSH, ACK] Seq=109 Ack=329 Win=2…
535 0.000174 10.14.135.119 10.14.135.5 TCP 60690 → 135 [ACK] Seq=329 Ack=281 Win=262400 Len=0 SLE=109 SRE=281
536 0.030399 10.14.135.5 10.14.135.119 TCP [TCP Retransmission] 49667 → 60691 [PSH, ACK] Seq=389 Ack=937 Win…
537 0.000168 10.14.135.119 10.14.135.5 TCP 60691 → 49667 [ACK] Seq=937 Ack=573 Win=2101760 Len=0 SLE=389 SRE…
538 0.167230 10.14.135.119 10.14.135.5 DNS Standard query 0x3291 A microsoft.com
539 0.000238 10.14.135.5 10.14.135.119 DNS Standard query response 0x3291 A microsoft.com A 40.76.4.15 A 40.…
Context ID: 1
Cancel count: 0
[Opnum: 77]
[Request in frame: 532]
[Time from request: 0.000351000 seconds]
Auth Info: NETLOGON Secure Channel, Packet privacy, AuthContextId(1)
Auth type: NETLOGON Secure Channel (68)
Auth level: Packet privacy (6)
Auth pad len: 8
Auth Rsrvd: 0
Auth Context ID: 1
Secure Channel Verifier
Local Security Authority, lsa_LookupNames4
Operation: lsa_LookupNames4 (77)
[Request in frame: 532]
Encrypted stub data: 4361308e81f6c69a04ebe1eb61801b26bd4d46fa1bd4defe3f1f7354ca430
0000 00 17 69 f2 d4 d8 a4 1f 72 c2 09 6a 08 00 45
0010 00 e0 48 84 40 00 80 06 8e fb 0a 0e 87 05 0a
0020 87 77 c2 03 ed 13 1e 43 be ca 60 29 fc 3f 50
0030 20 11 b2 a1 00 00 05 00 02 03 10 00 00 00 b8
0040 38 00 03 00 00 00 58 00 00 00 01 00 00 00 43
0050 30 8e 81 f6 c6 9a 04 eb e1 eb 61 80 1b 26 bd
0060 46 fa 1b d4 de fe 3f 1f 73 54 ca 43 05 6f e6
0070 25 33 23 e0 68 d9 70 87 5a 4c 28 af 3b d8 d0
0080 2d 72 60 b3 aa 17 9f 48 61 96 95 a5 03 ef ab
0090 17 f9 00 0d 54 85 5d ee 95 f7 09 a1 81 fa 47
00a0 1c 74 3e 73 4c 36 f3 62 c0 cf 69 65 09 a1 44
00b0 08 00 01 00 00 00 13 00 1a 00 ff ff 00 00 4c
00c0 3b de a2 68 4e 10 af 4c 64 5e a0 3c 83 52 83
00d0 7c 97 71 4e 4b 7b a7 ba e7 36 b3 db 32 b2 df
00e0 ca 93 fd 21 b6 6b 1a 6b da e6 e3 a5 21 9a



26. I think, post malware transmission from webserver, domain security policies and account database information are trying to be captured by the intruder since DCERPC /LSARPC have vulnerabilities. These are followed by PSH, ACK and eventually FIN, ACK closing the channel
27. Moving further, I checked total DNS packets and found 352 in total with standard query and response providing IP addresses to below domains:

amazon.com
arc.msn.com
bing.com
client.wns.windows.com
ctldl.windowsupdate.com
dns.msftncsi.com
evoke-windowsservices-tas.msedge.net
google.com
introwebsites.com
login.live.com
login.microsoftonline.com
microsoft.com
myexternalip.com
nexusrules.officeapps.live.com
pulpfarmerd.com
r3.i.lencr.org
ris.api.iris.microsoft.com
settings-win.data.microsoft.com
update.googleapis.com
v10.events.data.microsoft.com
wpad.rusticmoment.com
www.amazon.com
www.bing.com
www.microsoft.com
www.yahoo.com
x1.c.lencr.org
yahoo.com

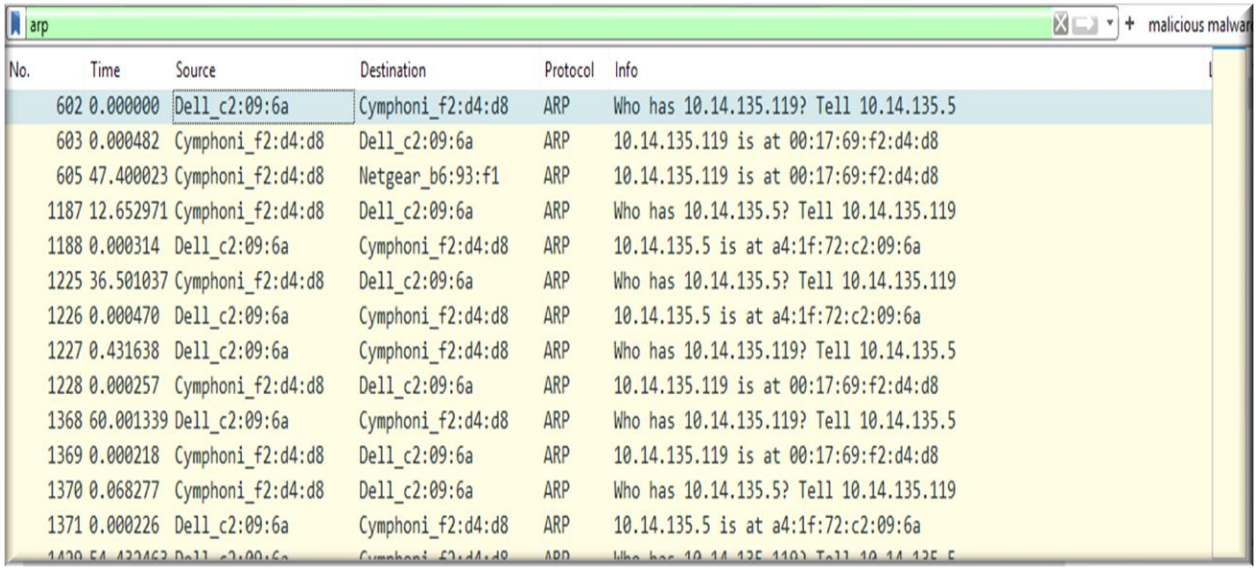
28. I have checked all the above domain in virus total and found

Domain	Virus total scan
amazon.com	Clean
arc.msn.com	Clean
bing.com	Clean
client.wns.windows.com	Clean
ctldl.windowsupdate.com	Clean
dns.msftncsi.com	Clean
evoke-windowsservices-tas.msedge.net	Clean
google.com	Clean
introwebsites.com	Malicious
login.live.com	Clean
login.microsoftonline.com	Clean
microsoft.com	Clean
myexternalip.com	Malicious
nexusrules.officeapps.live.com	Clean
pulpfarmerd.com	Malicious
r3.i.lencr.org	Clean
ris.api.iris.microsoft.com	Clean
settings-win.data.microsoft.com	Clean
update.googleapis.com	Clean
v10.events.data.microsoft.com	Clean
wpad.rusticmoment.com	Clean
www.amazon.com	Clean
www.bing.com	Clean
www.microsoft.com	Clean
www.yahoo.com	Clean
x1.c.lencr.org	Clean
yahoo.com	Clean

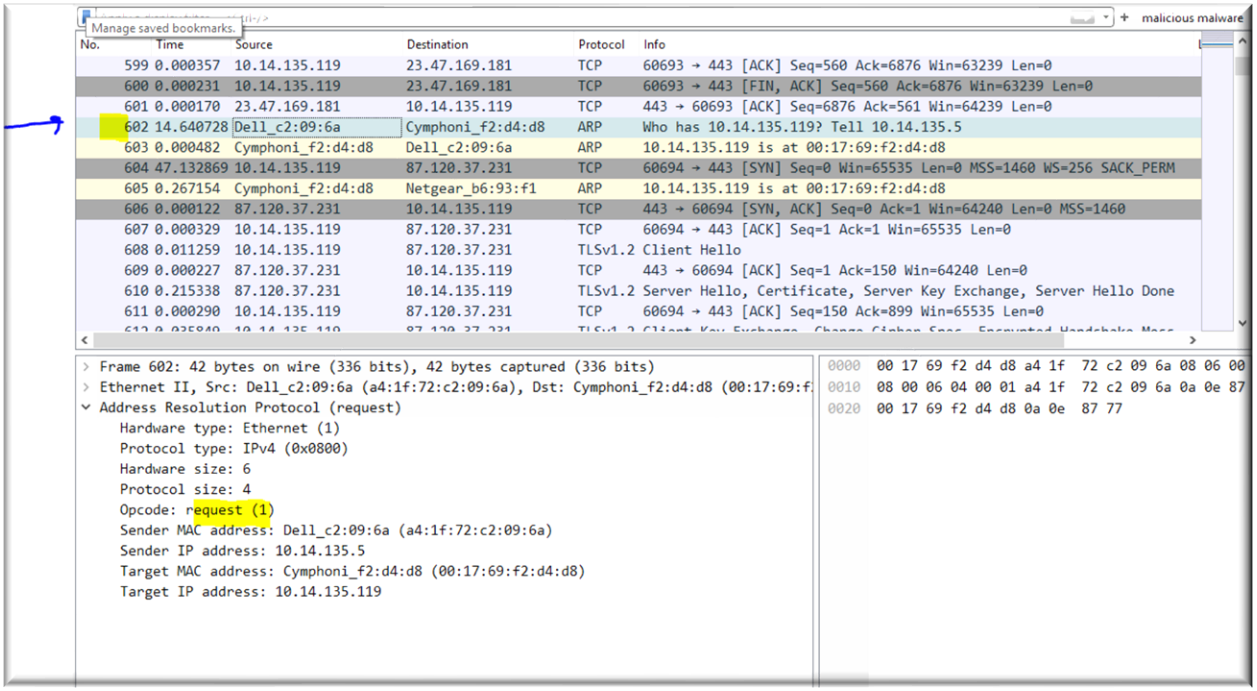
29. Now let's observe link layer protocol i.e. ARP packets.

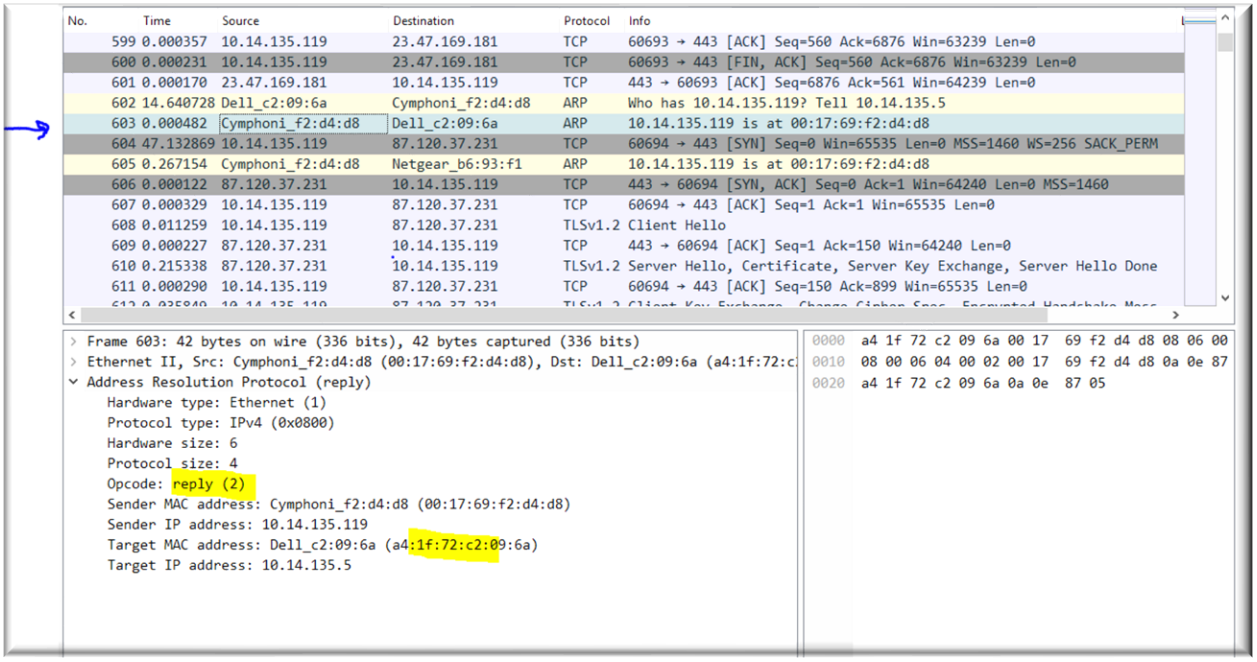
    The Address Resolution Protocol (ARP) is a communication protocol used for discovering the link-layer address, such as a MAC address, associated with a given internet layer address, typically an IPv4 address.

30. There are 730 ARP packets observed in wireshark. Few are shown below:



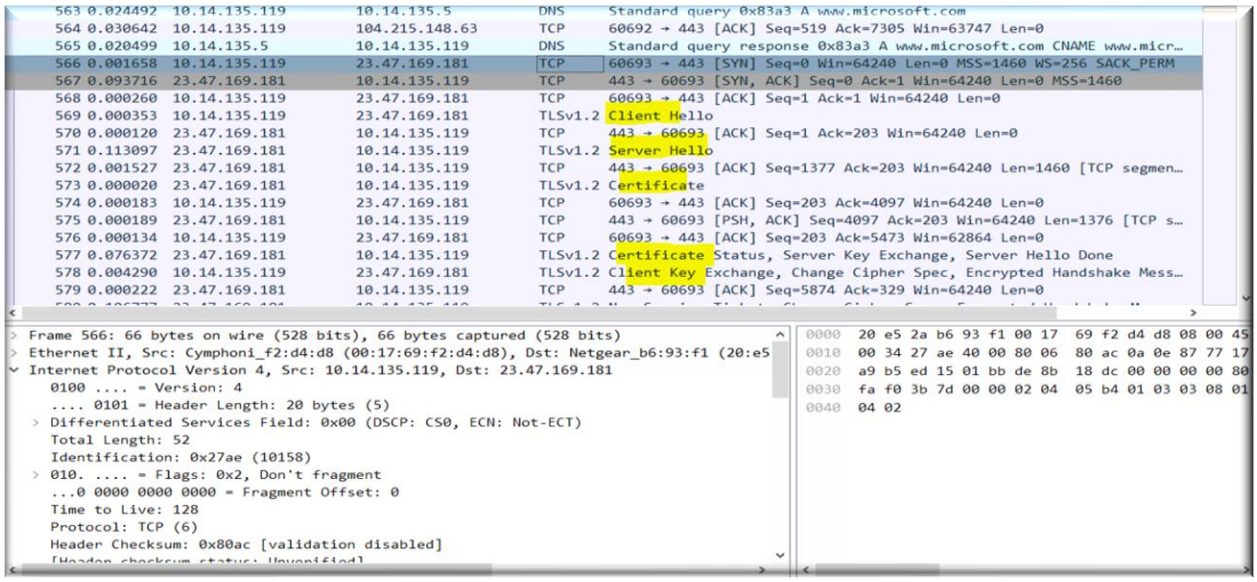
31. Let's look into 602-605 packet numbers.





32. Dell_c2:09:6a and Cymphoni_f2:d4:d8 successfully obtains one another's physical(MAC address) by broadcasting ARP request onto the network.

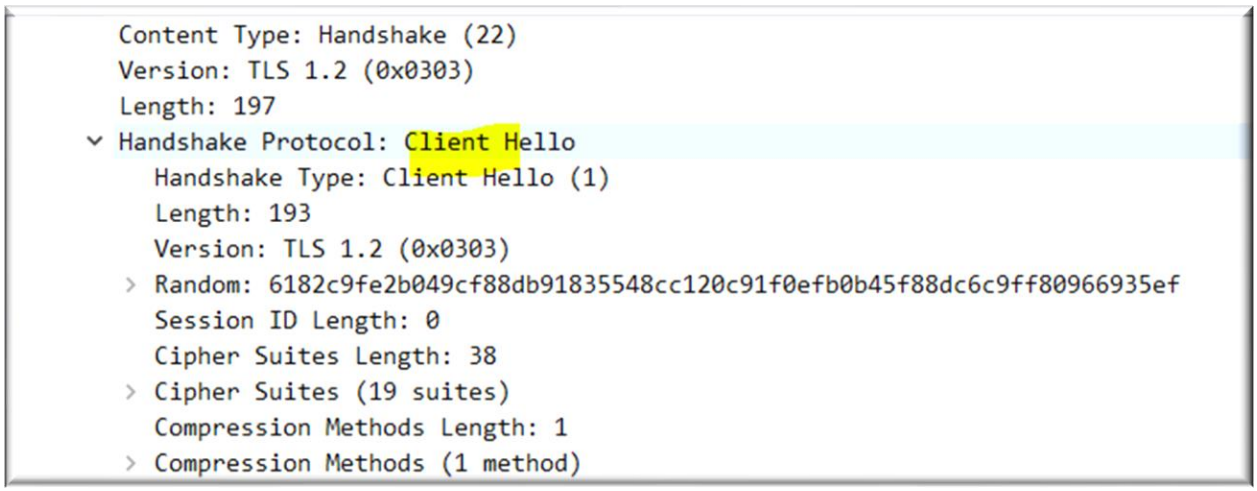
33. A value of 1 indicates this ARP packet is a *Request*, or a value of 2 would indicate this ARP packet is a *Response*
34. There are around 4072 packets with TLS v1.2 protocol. TCP is 3 way handshake whereas TLS v1.2 is 5 way handshake.
35. Let's go through 1 instance of TLS handshake below connecting to www.microsoft.com



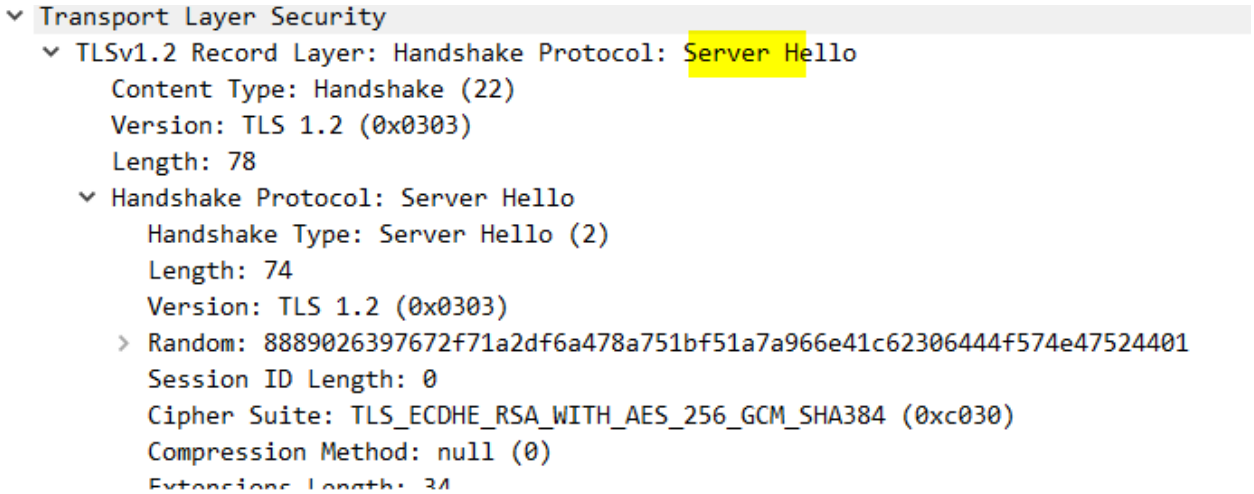
36. We see the DNS request <-> response taking place yielding IP address 23.47.169.181.

37. HTTP uses TCP handshake. HTTPS uses HTTP over TLS wherein it exchanges certificates, keys and client/server Hello's. In our case it is going to be HTTPS.

38. Once the TCP three-way handshake is done. The TLS handshake will kick in with client hello. The first step to start a TSL handshake with '**Client Hello**' from 10.14.135.119



39. After the server receives the client hello, it will examine the supported TLS versions and cipher suites sent by the client. The server will select the highest supported TLS version by both client and the server – TLS v1.2 in our case.



40. Sending Certificate, Server Encryption Key, Server Hello Done to the client

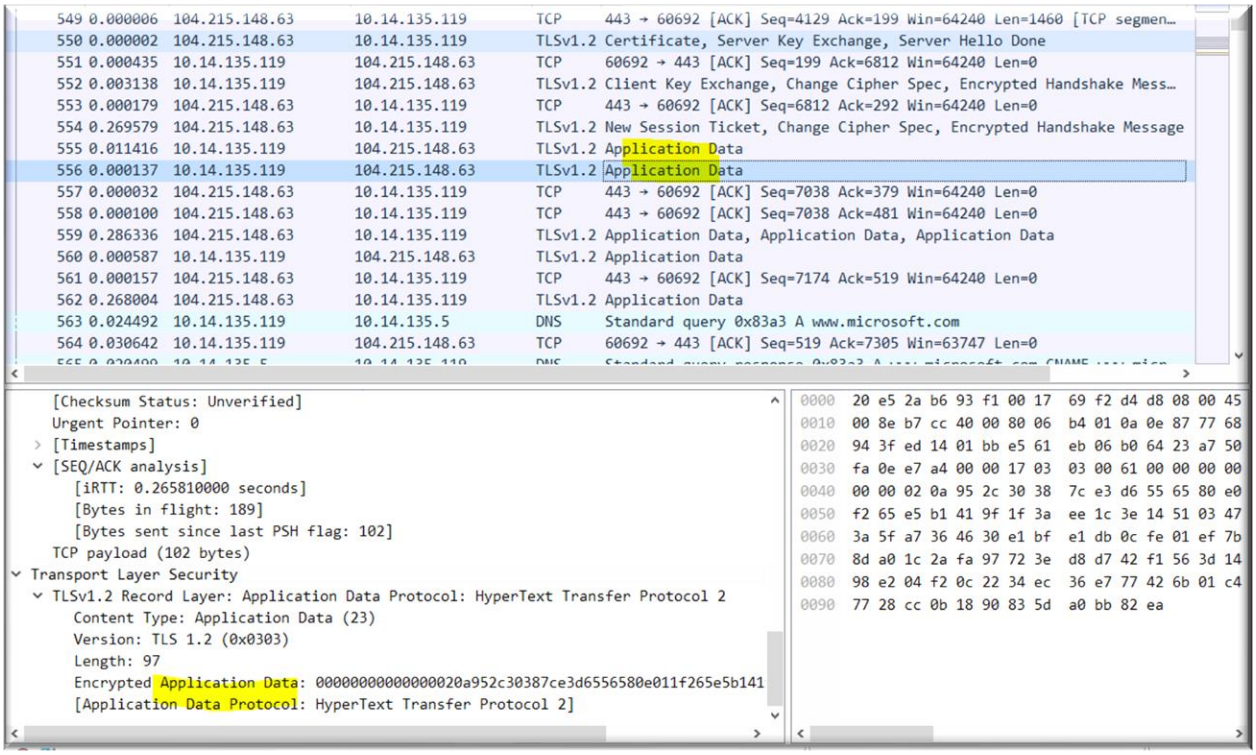
41. The server will send the certificate, server encrypted key, and server hello done message to the client.

42. Sending Certificate, Server Encryption Key, Server Hello Done to the client in Wireshark

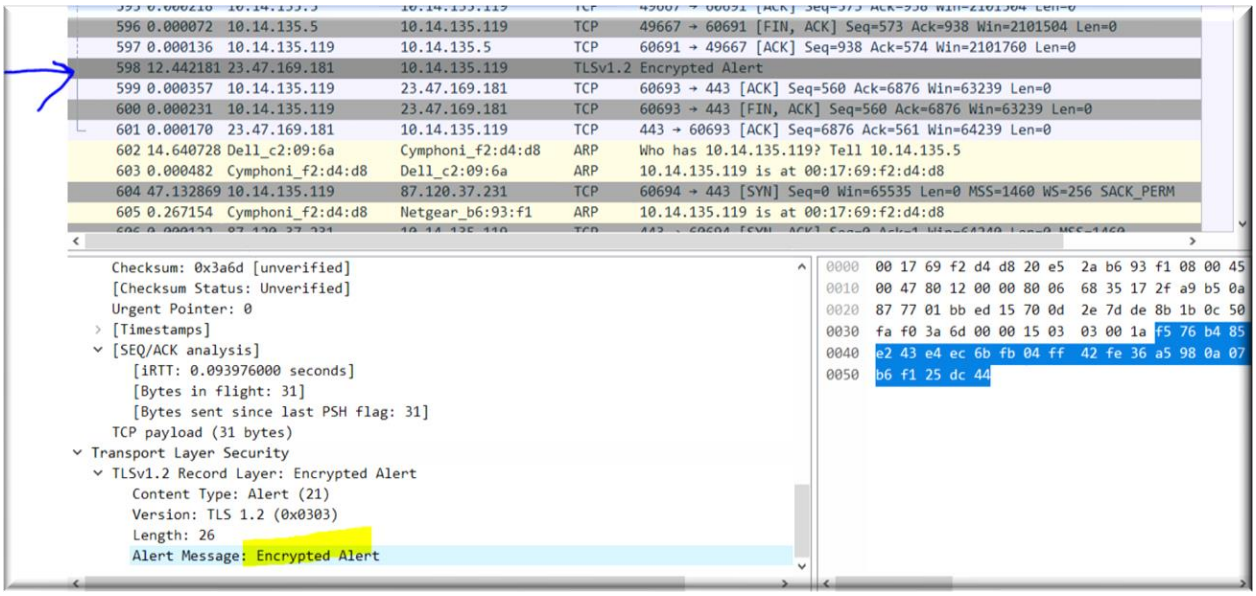
43. Receiving Client encrypted key, change cipher spec, and encrypted handshake message from client

44. After the client receives the server encrypted key. It will respond with the client encrypted key. It also sends change cipher spec. What it means is that it has enough information to start encrypted communication, and it is going to send the data with encryption from now onwards. Till now, the communication was plain text. After this, communication will happen with encryption. After the server receives the change cipher spec message, it expects encrypted data from the client.

45. The final step in TLS handshake — sending change cipher spec and the final handshake message to the client.
46. This is going to be the last message that the server is going to send, which includes change cipher spec and finishing the message. This indicates that all feature messages are going to be encrypted.
47. Now that the TLS handshake is done we can see the Encrypted application data, and it is shown in the screenshot below:

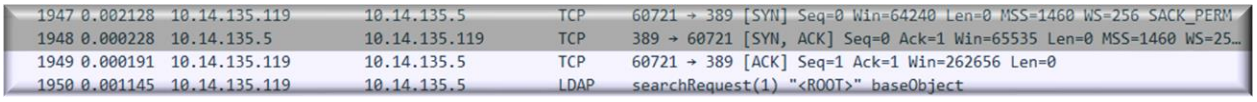


48. One more interesting thing I found is that, there is an Encrypted alert in the info section of the detailed view section. This means that wireshark cannot decrypt the data. Refer the screenshot below:

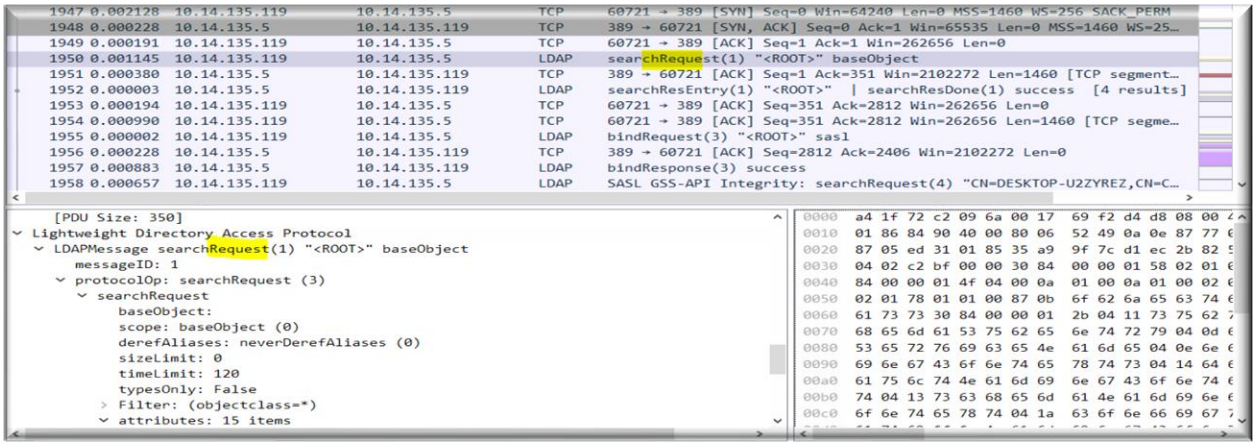


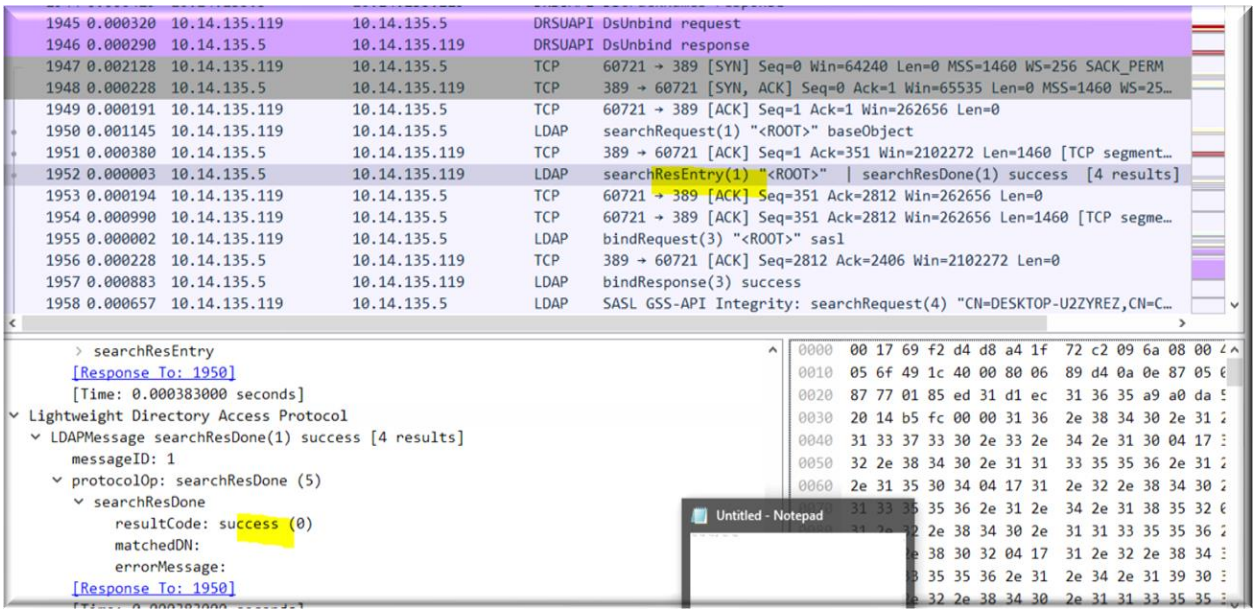
49. Now let's observe the LDAP authentication method from packet number 1947.
50. Authentication is supplied in the "LDAP bind" operation. There are three different authentication methods.
51. We see sasl authentication method in these packets.
52. Simple Authentication and Security Layer, SASL is an extensible framework that makes it possible to plug almost any kind of authentication into LDAP. Beside authentication, it can also provide a data security layer offering data integrity and data confidentiality services. All authentication mechanisms may not support confidentiality (encryption)

53. The first 3 packets belong to TCP 3-way handshaking.

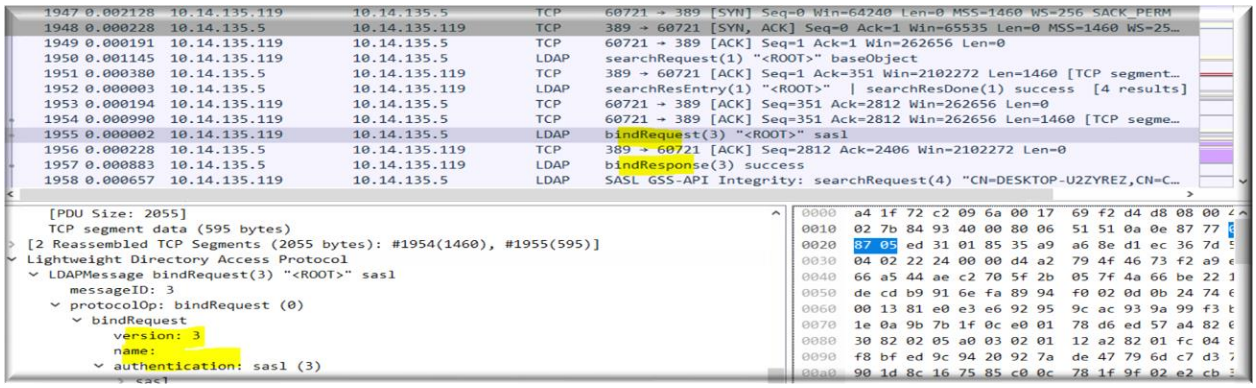


54. Once the TCP connection establishes, It simply creates a search request, searchRequest(1) and search result in received. Refer below screenshots:

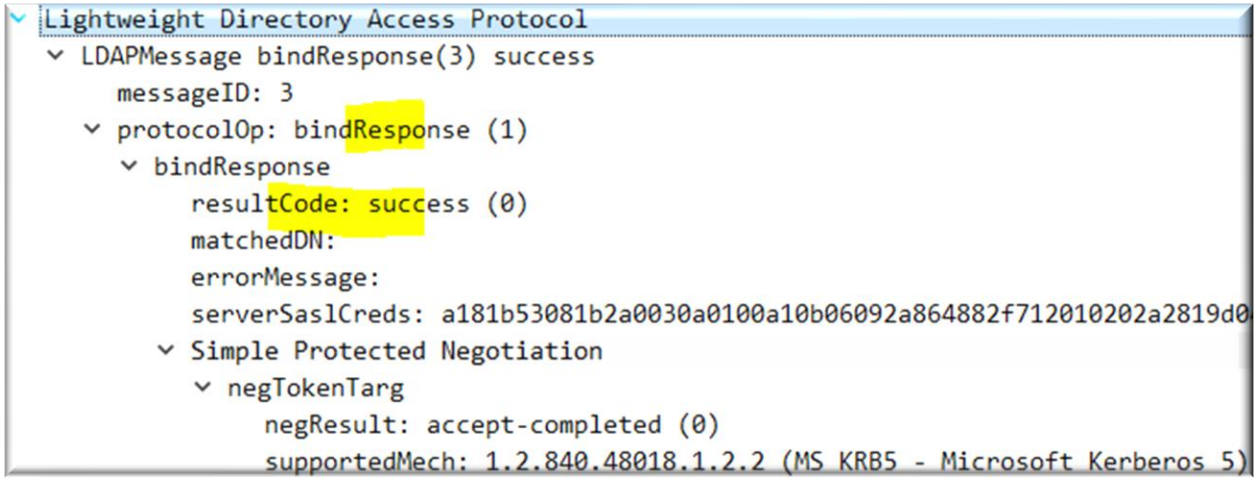




55. The binding operation includes user name and password.
56. In our case it is sasl authentication. Hence, it will show that SASL is used for authentication. Refer below screenshot:



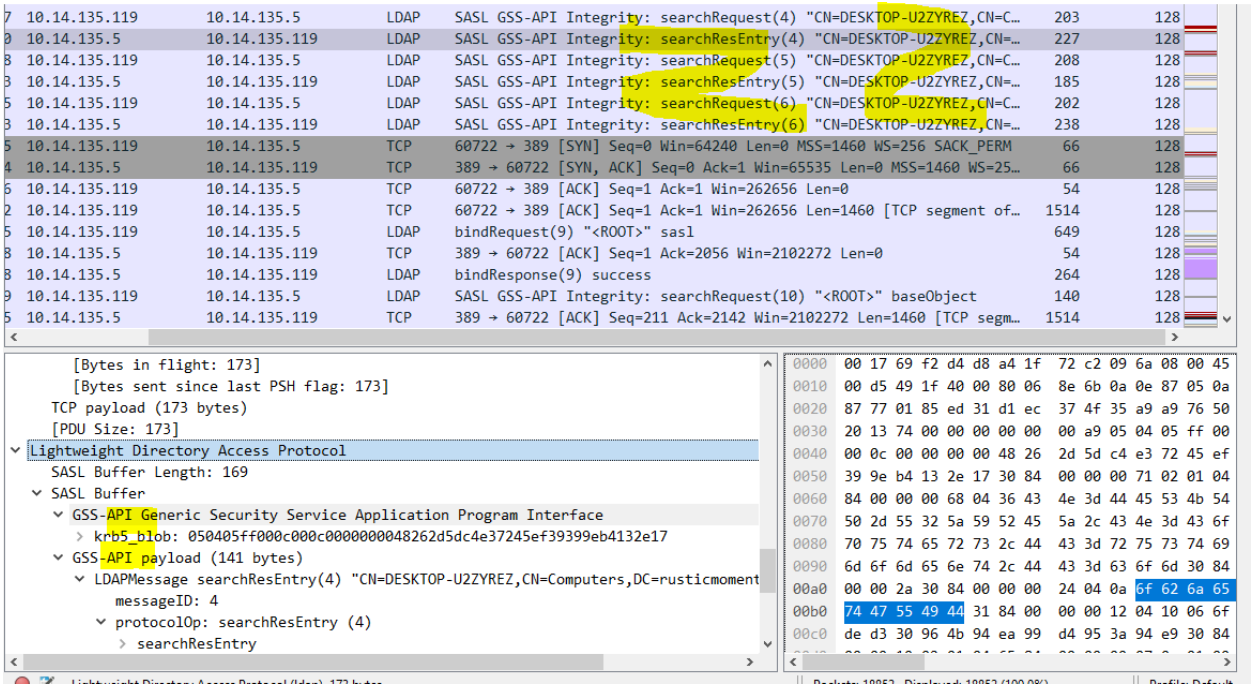
57. The packet also contains which version of LDAP it wants to use (in this case version 3).



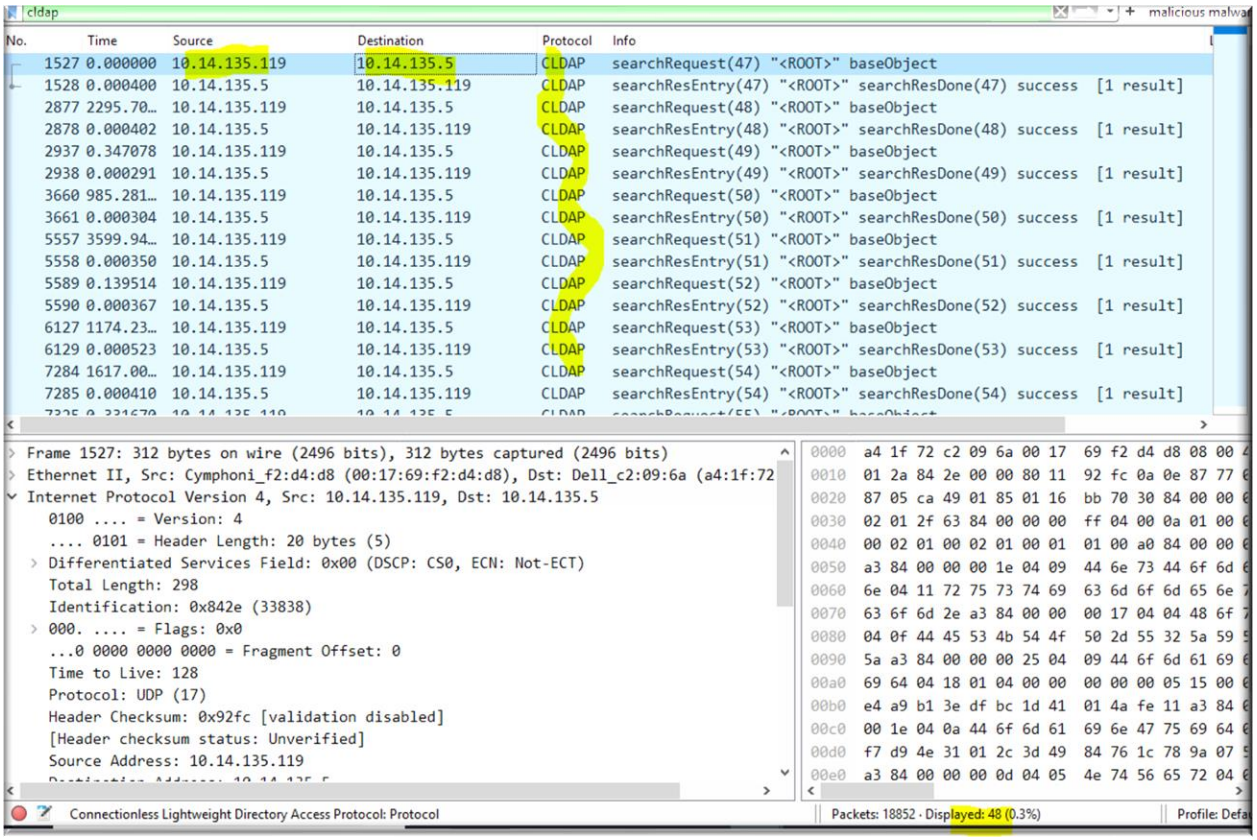
58. The server authenticates the client successfully and returns "success" as "resultCode" using bindResponse operation as shown in the screenshot above.
59. From this moment, the client can query the ldap server.

60. The client uses unbind operation to end the session

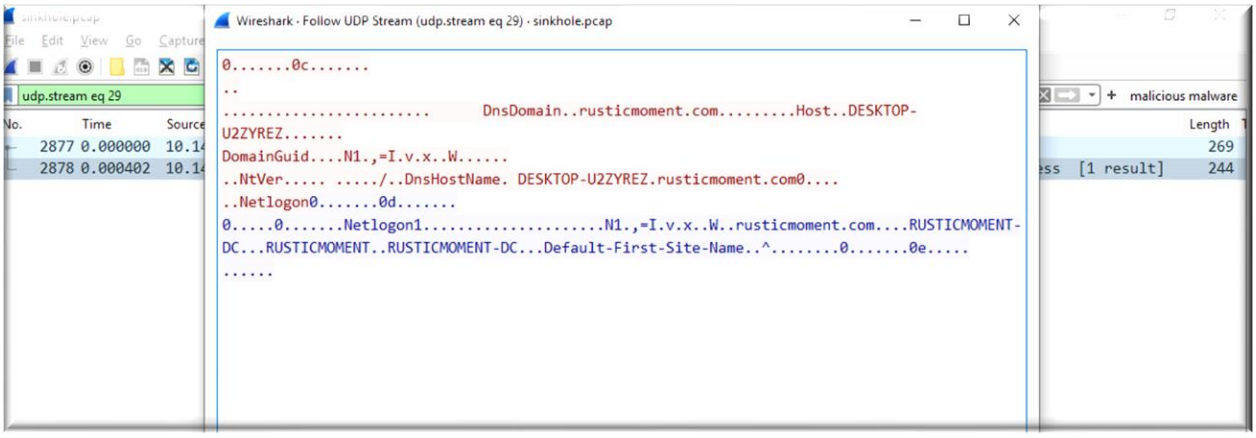
61. SASL supports multiple authentication mechanism. Few are briefly depicted in the screenshot below:



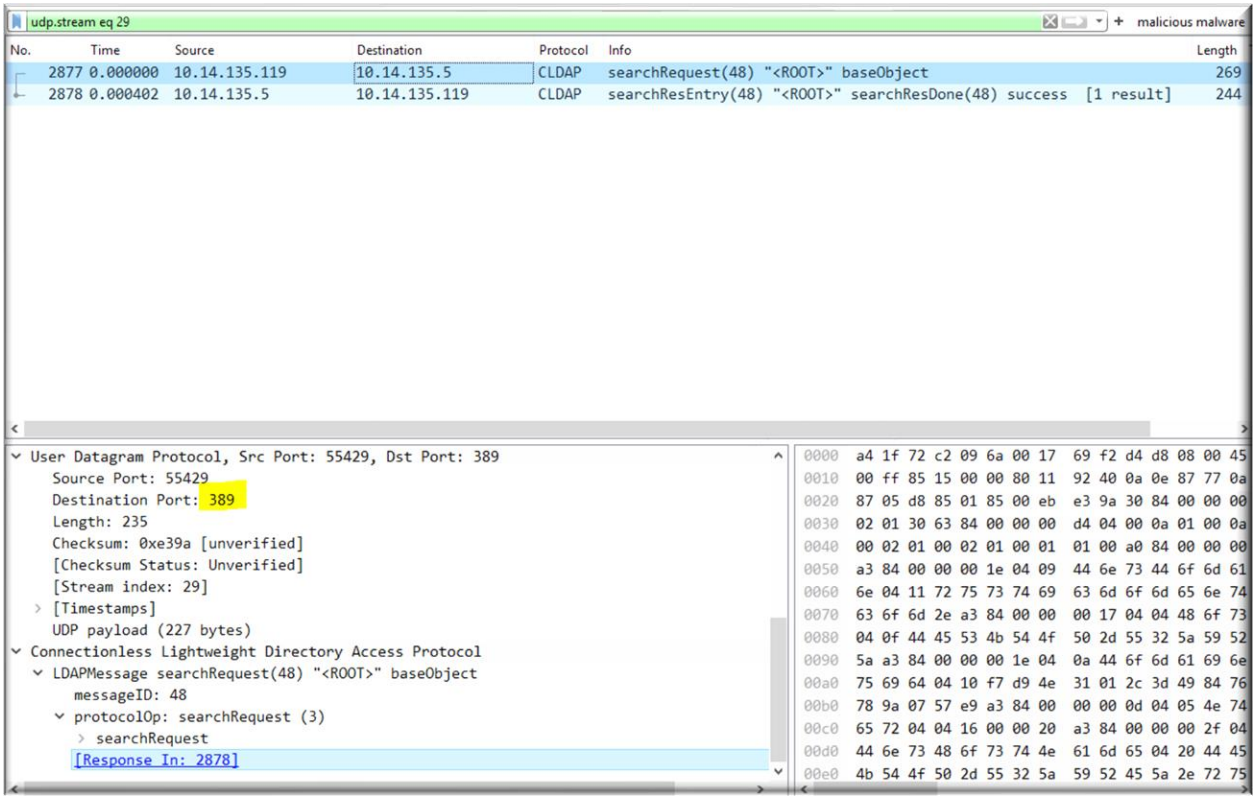
62. Now let's look into CLDAP protocol. There is one more interesting observation which I've learnt which is regarding CLDAP protocol. CLDAP is UDP based protocol and is under radar for DDoS attacks since many years in the past.
63. There are around 48 packets with cldap protocol in the provided pcap file. Refer screenshot below:



64. The communication is taking place between the client 10.14.135.119 and server 10.14.135.5
65. Looking at the UDP stream as shown in the screenshot below, I think this is a DDoS attack.



66. Even the port number 389 gives us the idea that there may be a possibility of an attack through this vulnerable port.



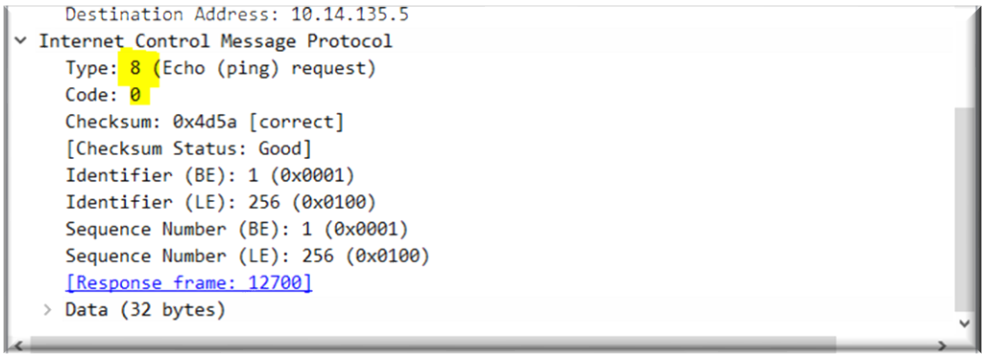
67. LDAP over TCP is safe whereas LDAP over UDP known as CLDAP is unsecure with 389 port being vulnerable.
68. Most LDAP servers and clients use the TCP protocol, which prevents amplification because of a connection handshake that verifies the source and destination can communicate with one another. UDP does not perform this verification, so the LDAP server can be convinced to send traffic to a destination that is unverified.
69. The easiest way to solve this issue is to enable a firewall on your server that blocks the LDAP port 389 from being accessed via UDP. LDAP is most used on Windows servers running Active Directory services.
70. Furthermore, there are ICMP protocols provided in the pcap file. Unlike the Transport Control Protocol (TCP) and User Datagram Protocol (UDP), the Internet Control Message Protocol (ICMP) is not designed for carrying data.
71. While ICMP packets do have a data section, their purpose is not to wrap and carry protocols like HTTP and DNS. Instead, ICMP is designed as a low-level management protocol for the internet. It carries error messages and implements simple management functions.
72. As an error messaging protocol, the structure of an ICMP packet is designed to provide the necessary information to the recipient. Error data in ICMP is carried in two values: the type and the code. Various types and codes are portrayed below:

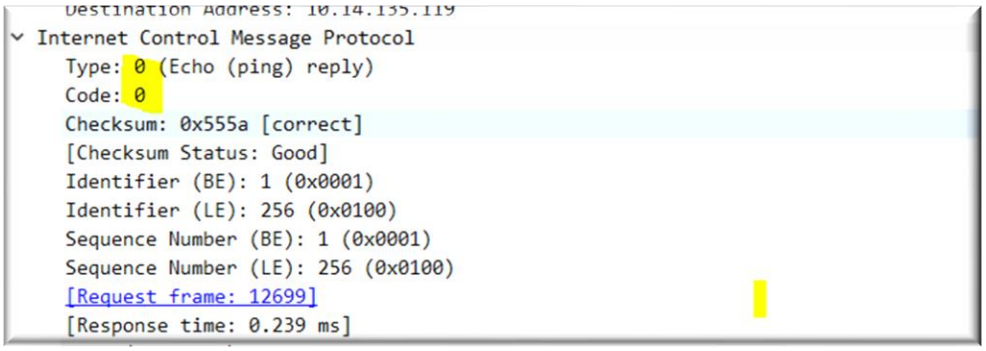
Type	Code	Description
0 – Echo Reply	0	Echo reply
3 – Destination Unreachable	0	Destination network unreachable
	1	Destination host unreachable
	2	Destination protocol unreachable
	3	Destination port unreachable
	4	Fragmentation needed and DF flag set
	5	Source route failed
5 – Redirect Message	0	Redirect datagram for the Network
	1	Redirect datagram for the host
	2	Redirect datagram for the Type of Service and Network
	3	Redirect datagram for the Service and Host
8 – Echo Request	0	Echo request
9 – Router Advertisement	0	Use to discover the addresses of operational routers
10 – Router Solicitation	0	
11 – Time Exceeded	0	Time to live exceeded in transit
	1	Fragment reassembly time exceeded
12 – Parameter Problem	0	Pointer indicates error
	1	Missing required option
	2	Bad length
13 – Timestamp	0	Used for time synchronization
14 – Timestamp Reply	0	Reply to Timestamp message

73. As shown above, the type of an ICMP packet contains the overall message that the message is intended to convey. For example, a type value of 3 means that the intended destination is unreachable.
74. For some types, there are multiple code values intended to provide additional information. For example, a type 3 ICMP message with a 0 code points to issues with the destination network, while a 1 code means that the issue is that the particular host is unreachable.
75. While many ICMP messages are designed to be sent as error messages in response to packets of other protocols, some are designed to implement standalone functionality.

76. The purpose of ping is to determine if the system at a certain IP address exists and is currently functional, and that a route to that system can be found. Typing ping into the Windows or Linux terminal will send a series of ping packets and provide a percentage value for the reachability of the destination based upon the number of ping requests that received a response.

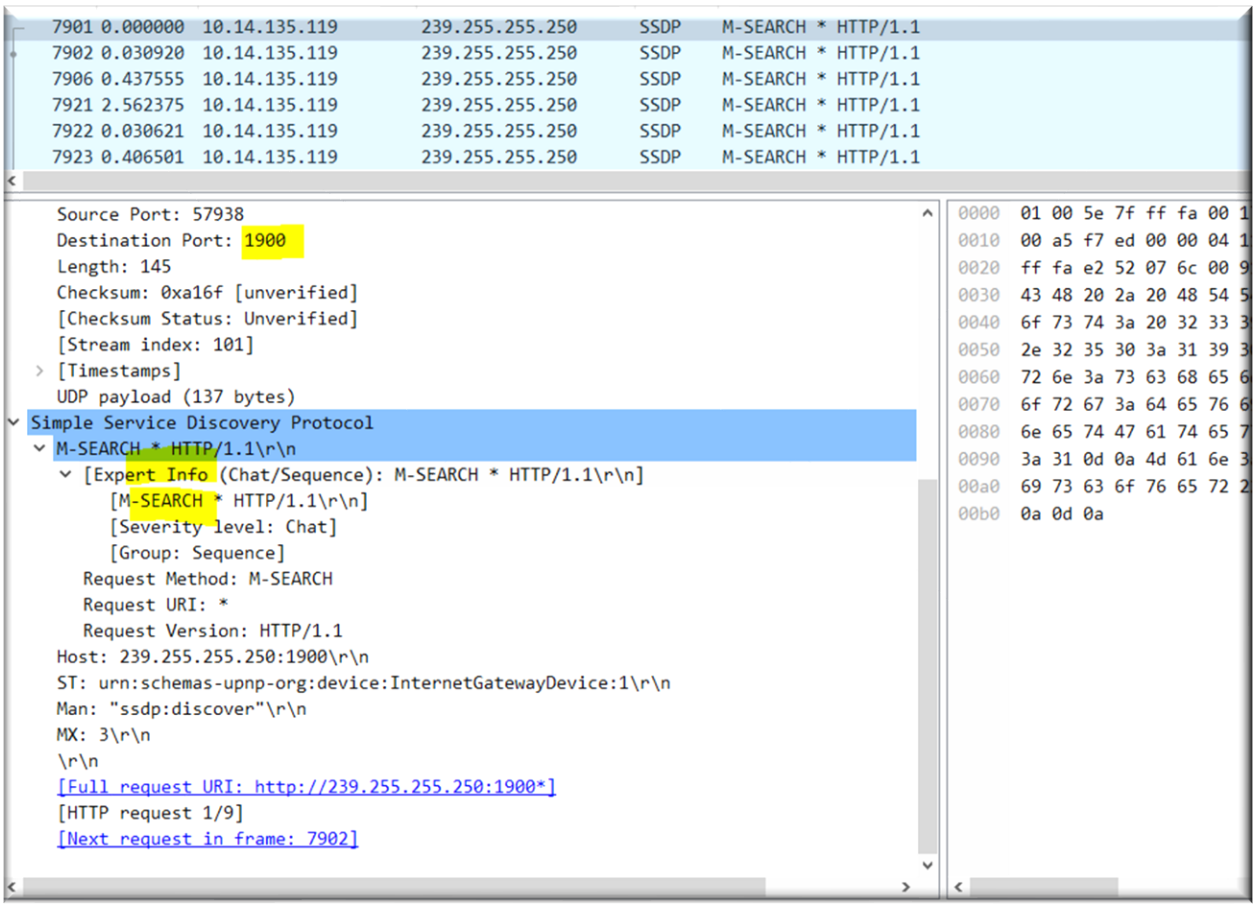
77. The images below shows an ICMP ping request and response in Wireshark.



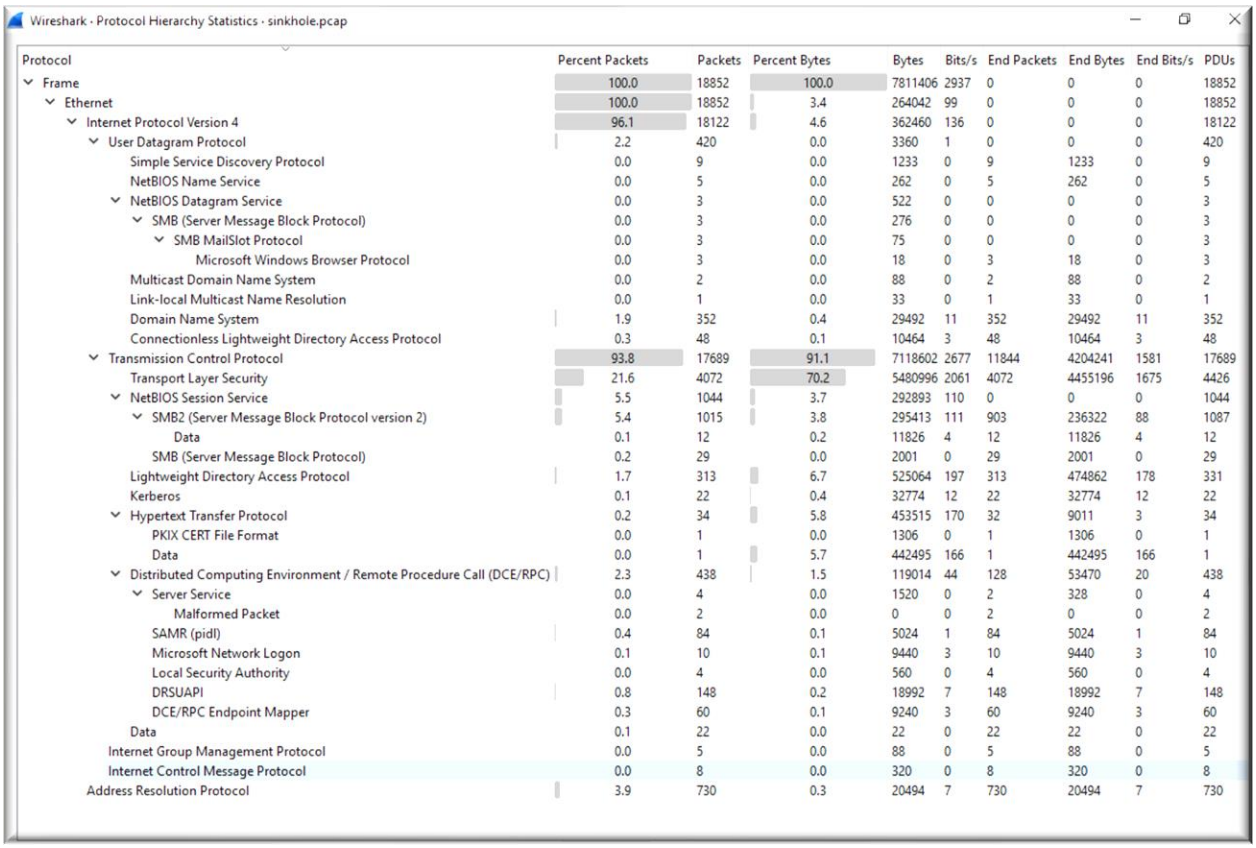


78. Other protocols which I have learnt and observed in the pcap file are:
    Internet Group Management Protocol (IGMP): IGMP is used by IP hosts to manage their dynamic multicast group membership.
79. It is also used by connected routers to discover these group members.

80. Simple Service Discovery Protocol (SSDP): The SSDP protocol can discover Plug & Play devices, with uPnP (Universal Plug and Play).
81. SSDP is HTTP like protocol and works with NOTIFY and M-SEARCH methods.
82. SSDP uses UDP transport protocol on port 1900. Refer below screenshot:

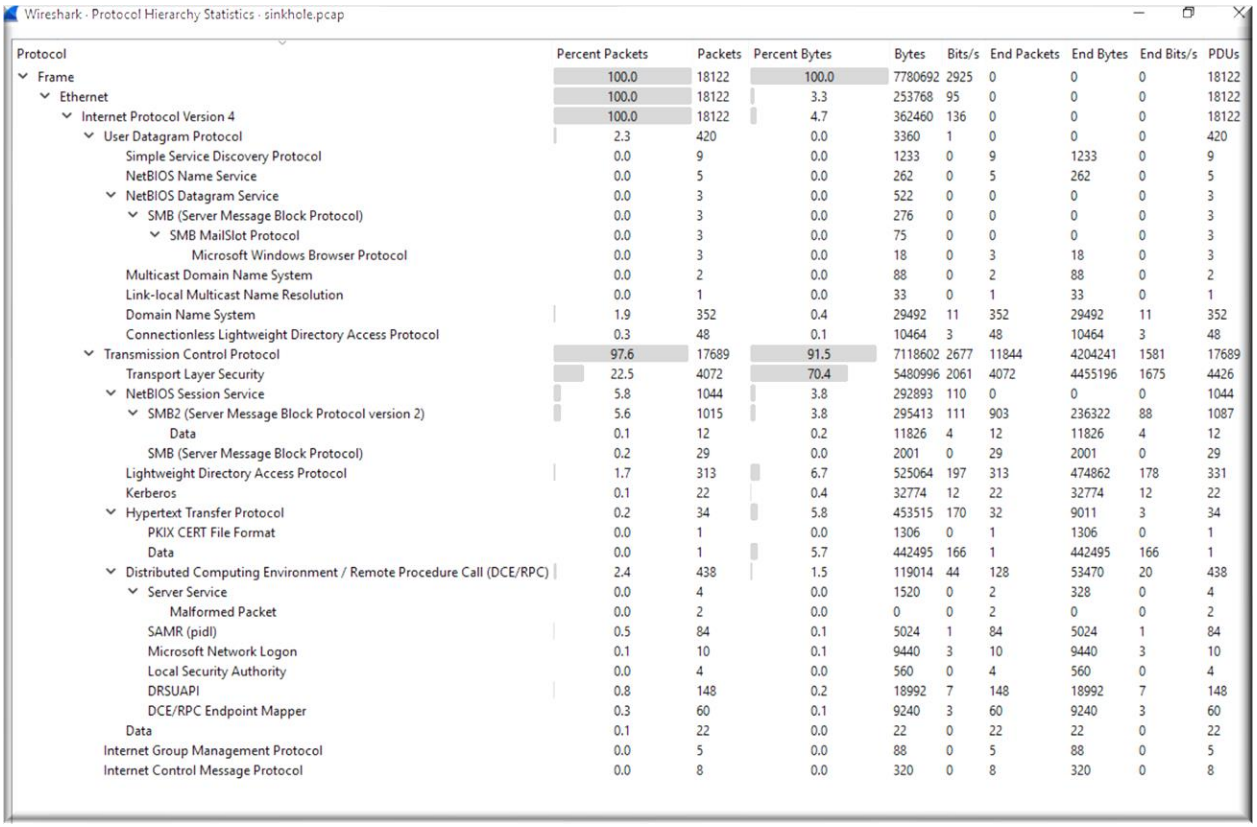


83. Now, let's observe some graphical study in wireshark "Statistics menu" which I've learnt additionally.
84. A wide range of tools related to network stats is available in the menu, which facilitate users in gaining information ranging from general info to specific protocol related info in detail.
85. general details with respect to the packets captured, filters applied, marked packets, and various other stats can be checked in the Statistics menu.
86. The Protocol Hierarchy window provides us with an overview regarding distribution of protocols used in the communication process and how to spot unusual activities in your network that do not follow the benchmark as expected. By distribution of protocols, I mean in what percentage a certain protocol has been used in the communication between two hosts, and statistics, for example, how many bytes and packets are being sent and received for every protocol, are collected easily. Any form of unusual activity can be easily figured out by matching our current traffic with the baseline created.



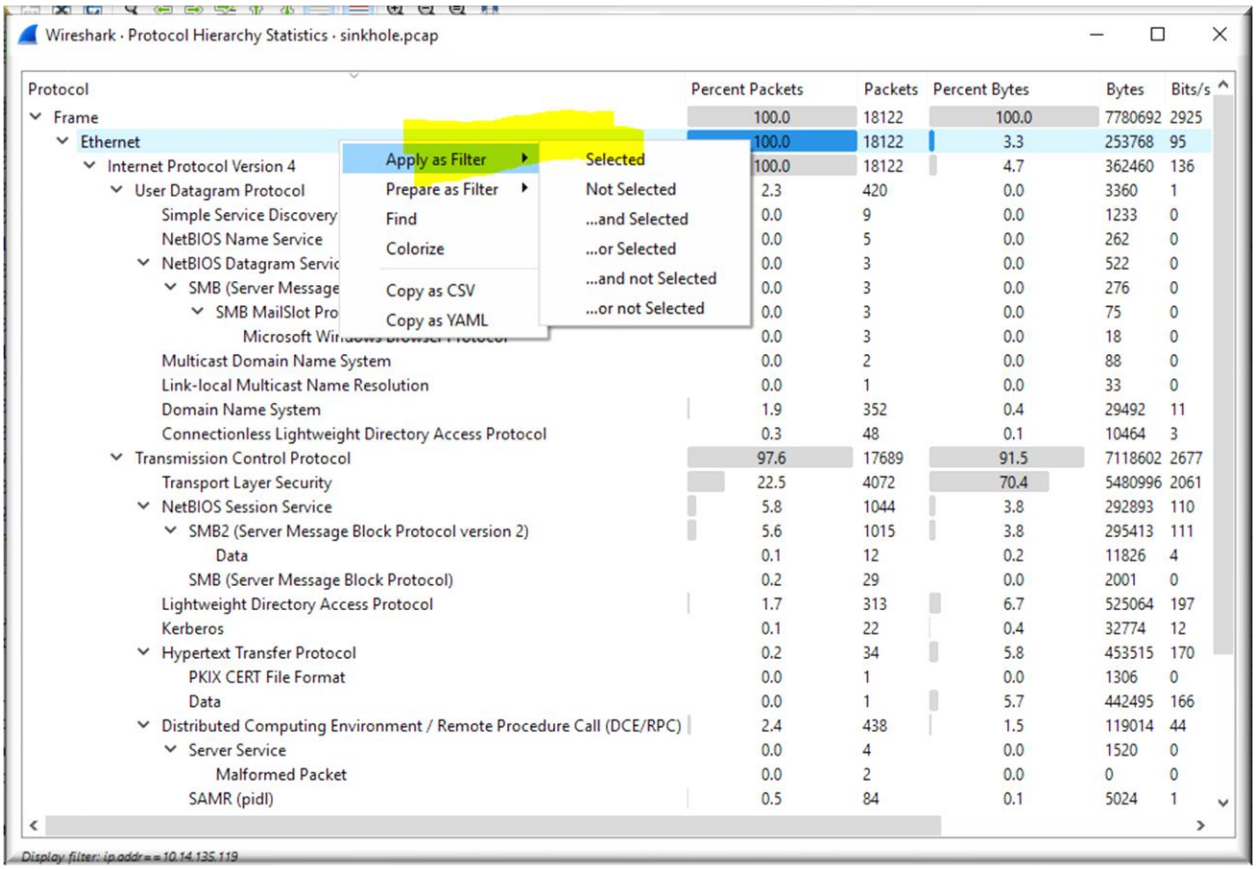
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	18852	100.0	7811406	2937	0	0	0	18852
Ethernet	100.0	18852	3.4	264042	99	0	0	0	18852
Internet Protocol Version 4	96.1	18122	4.6	362460	136	0	0	0	18122
User Datagram Protocol	2.2	420	0.0	3360	1	0	0	0	420
Simple Service Discovery Protocol	0.0	9	0.0	1233	0	9	1233	0	9
NetBIOS Name Service	0.0	5	0.0	262	0	5	262	0	5
NetBIOS Datagram Service	0.0	3	0.0	522	0	0	0	0	3
SMB (Server Message Block Protocol)	0.0	3	0.0	276	0	0	0	0	3
SMB MailSlot Protocol	0.0	3	0.0	75	0	0	0	0	3
Microsoft Windows Browser Protocol	0.0	3	0.0	18	0	3	18	0	3
Multicast Domain Name System	0.0	2	0.0	88	0	2	88	0	2
Link-local Multicast Name Resolution	0.0	1	0.0	33	0	1	33	0	1
Domain Name System	1.9	352	0.4	29492	11	352	29492	11	352
Connectionless Lightweight Directory Access Protocol	0.3	48	0.1	10464	3	48	10464	3	48
Transmission Control Protocol	93.8	17689	91.1	7118602	2677	11844	4204241	1581	17689
Transport Layer Security	21.6	4072	70.2	5480996	2061	4072	4455196	1675	4426
NetBIOS Session Service	5.5	1044	3.7	292893	110	0	0	0	1044
SMB2 (Server Message Block Protocol version 2)	5.4	1015	3.8	295413	111	903	236322	88	1087
Data	0.1	12	0.2	11826	4	12	11826	4	12
SMB (Server Message Block Protocol)	0.2	29	0.0	2001	0	29	2001	0	29
Lightweight Directory Access Protocol	1.7	313	6.7	525064	197	313	474862	178	331
Kerberos	0.1	22	0.4	32774	12	22	32774	12	22
Hypertext Transfer Protocol	0.2	34	5.8	453515	170	32	9011	3	34
PKIX CERT File Format	0.0	1	0.0	1306	0	1	1306	0	1
Data	0.0	1	5.7	442495	166	1	442495	166	1
Distributed Computing Environment / Remote Procedure Call (DCE/RPC)	2.3	438	1.5	119014	44	128	53470	20	438
Server Service	0.0	4	0.0	1520	0	2	328	0	4
Malformed Packet	0.0	2	0.0	0	0	2	0	0	2
SAMR (pidl)	0.4	84	0.1	5024	1	84	5024	1	84
Microsoft Network Logon	0.1	10	0.1	9440	3	10	9440	3	10
Local Security Authority	0.0	4	0.0	560	0	4	560	0	4
DRSUAPI	0.8	148	0.2	18992	7	148	18992	7	148
DCE/RPC Endpoint Mapper	0.3	60	0.1	9240	3	60	9240	3	60
Data	0.1	22	0.0	22	0	22	22	0	22
Internet Group Management Protocol	0.0	5	0.0	88	0	5	88	0	5
Internet Control Message Protocol	0.0	8	0.0	320	0	8	320	0	8
Address Resolution Protocol	3.9	730	0.3	20494	7	730	20494	7	730

87. If you want to check the protocol distribution for a specific host, then before you open the Protocol Hierarchy window, apply a display filter, for example, ip.addr==10.14.135.119. The same filter will be visible at the top of the Hierarchy window just below the title bar. This makes it easy for us to figure out what kind of traffic is actually generated from a certain host, and any malicious traffic from a certain host can be easily figured out. Refer below screenshot:



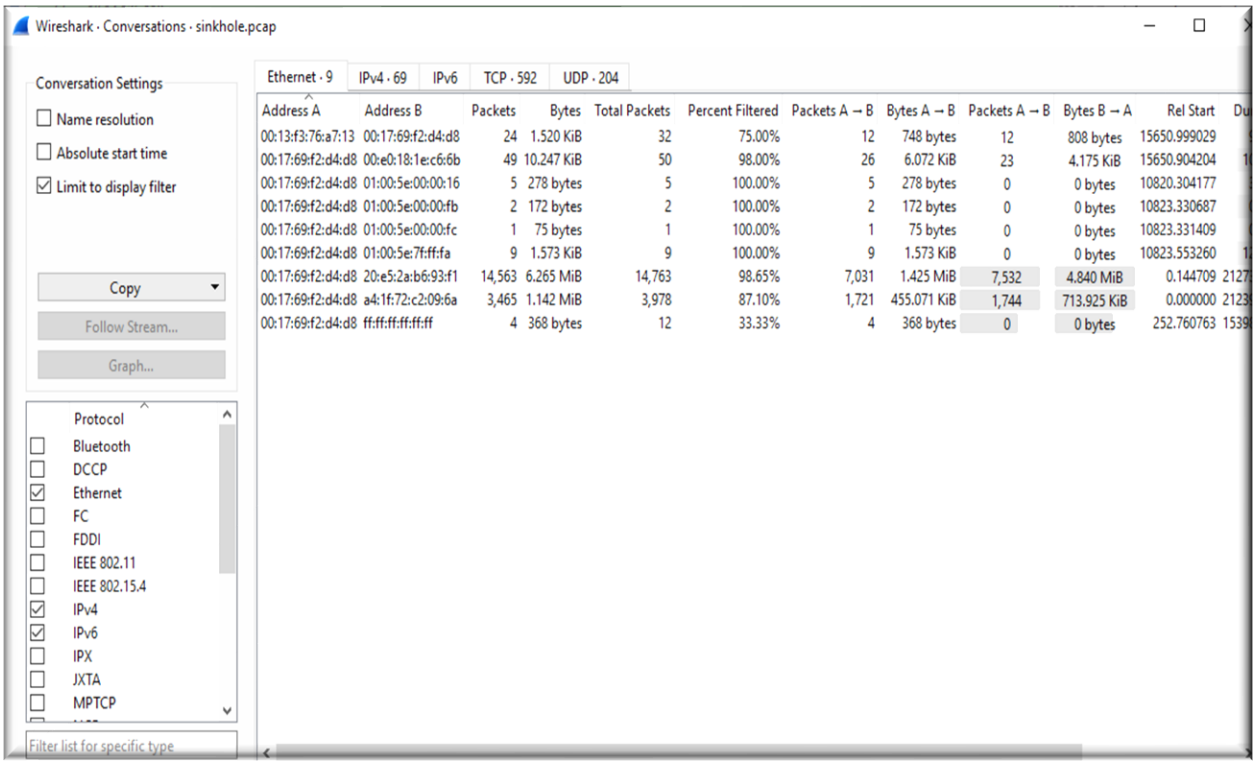
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	18122	100.0	7780692	2925	0	0	0	18122
Ethernet	100.0	18122	3.3	253768	95	0	0	0	18122
Internet Protocol Version 4	100.0	18122	4.7	362460	136	0	0	0	18122
User Datagram Protocol	2.3	420	0.0	3360	1	0	0	0	420
Simple Service Discovery Protocol	0.0	9	0.0	1233	0	9	1233	0	9
NetBIOS Name Service	0.0	5	0.0	262	0	5	262	0	5
NetBIOS Datagram Service	0.0	3	0.0	522	0	0	0	0	3
SMB (Server Message Block Protocol)	0.0	3	0.0	276	0	0	0	0	3
SMB MailSlot Protocol	0.0	3	0.0	75	0	0	0	0	3
Microsoft Windows Browser Protocol	0.0	3	0.0	18	0	3	18	0	3
Multicast Domain Name System	0.0	2	0.0	88	0	2	88	0	2
Link-local Multicast Name Resolution	0.0	1	0.0	33	0	1	33	0	1
Domain Name System	1.9	352	0.4	29492	11	352	29492	11	352
Connectionless Lightweight Directory Access Protocol	0.3	48	0.1	10464	3	48	10464	3	48
Transmission Control Protocol	97.6	17689	91.5	7118602	2677	11844	4204241	1581	17689
Transport Layer Security	22.5	4072	70.4	5480996	2061	4072	4455196	1675	4426
NetBIOS Session Service	5.8	1044	3.8	292893	110	0	0	0	1044
SMB2 (Server Message Block Protocol version 2)	5.6	1015	3.8	295413	111	903	236322	88	1087
Data	0.1	12	0.2	11826	4	12	11826	4	12
SMB (Server Message Block Protocol)	0.2	29	0.0	2001	0	29	2001	0	29
Lightweight Directory Access Protocol	1.7	313	6.7	525064	197	313	474862	178	331
Kerberos	0.1	22	0.4	32774	12	22	32774	12	22
Hypertext Transfer Protocol	0.2	34	5.8	453515	170	32	9011	3	34
PKIX CERT File Format	0.0	1	0.0	1306	0	1	1306	0	1
Data	0.0	1	5.7	442495	166	1	442495	166	1
Distributed Computing Environment / Remote Procedure Call (DCE/RPC)	2.4	438	1.5	119014	44	128	53470	20	438
Server Service	0.0	4	0.0	1520	0	2	328	0	4
Malformed Packet	0.0	2	0.0	0	0	2	0	0	2
SAMR (pidl)	0.5	84	0.1	5024	1	84	5024	1	84
Microsoft Network Logon	0.1	10	0.1	9440	3	10	9440	3	10
Local Security Authority	0.0	4	0.0	560	0	4	560	0	4
DRSUAPI	0.8	148	0.2	18992	7	148	18992	7	148
DCE/RPC Endpoint Mapper	0.3	60	0.1	9240	3	60	9240	3	60
Data	0.1	22	0.0	22	0	22	22	0	22
Internet Group Management Protocol	0.0	5	0.0	88	0	5	88	0	5
Internet Control Message Protocol	0.0	8	0.0	320	0	8	320	0	8

88. Using the Protocol Hierarchy window, you can create filters too. Just right-click on the protocol you wish to use and then go ahead and specify the expression, as shown in the following screenshot:

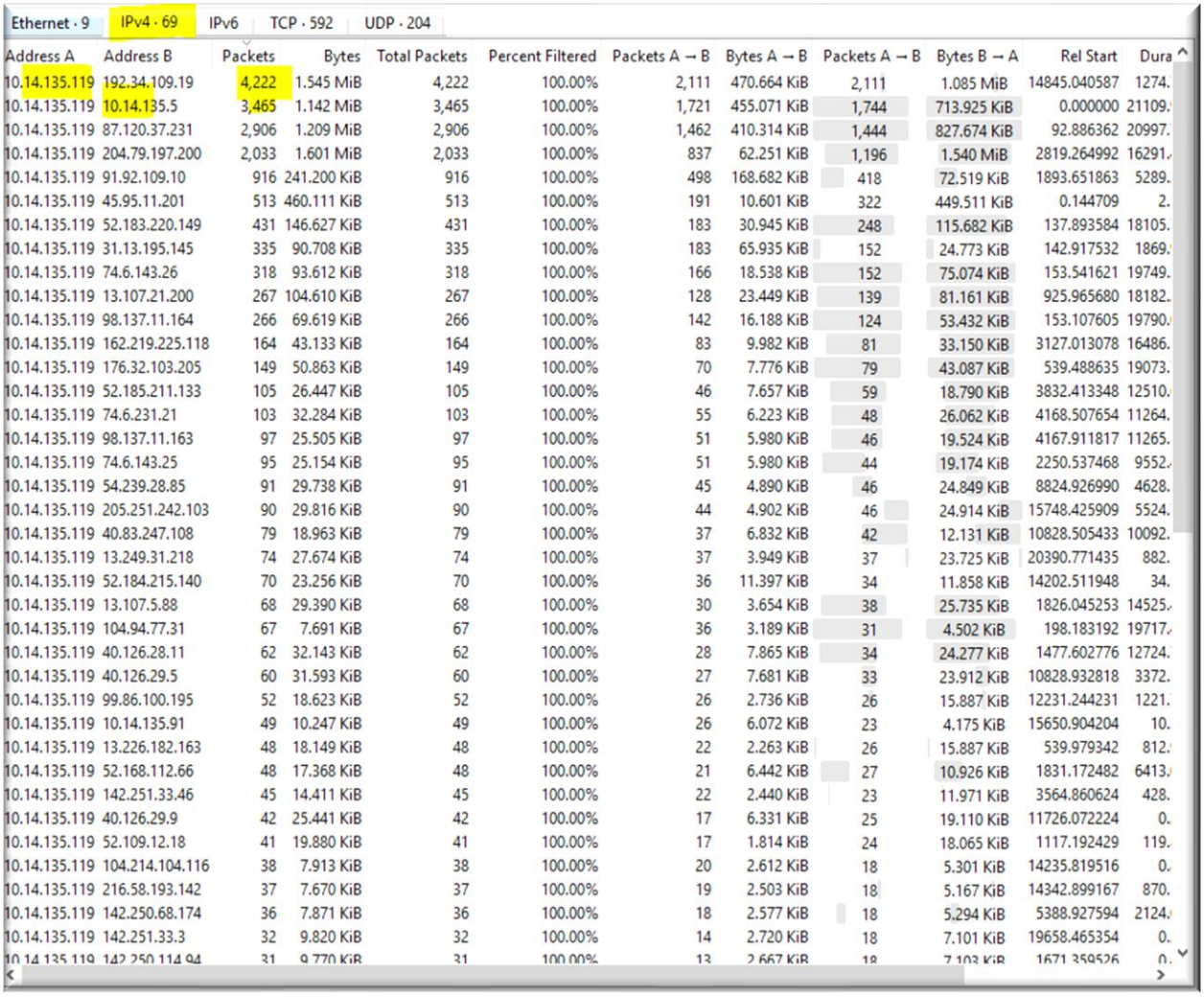


89. There will be situations when a certain host in your network has been breached and you might be observing some unusual traffic associated with a particular host. In such situations, the Protocol Hierarchy window will prove worthy.
90. **Conversations** When two devices are connected to each other on the network, they are supposed to communicate; this is considered normal behavior. However, suppose you have thousands of devices connected to your network and you want to figure out the most active device that is generating too much traffic, then in that instance, the Conversations window will be quite useful.

91. To access this nice tool, click on Statistics -> Conversations. After this, you will be presented with a window like the one shown in the following screenshot, which lists various details in terms of several columns listing the packets that were transferred, the bytes that were transferred, the flow of traffic, devices' MAC addresses, and various other details. At the top, you will observe various protocols displayed individually in separate tabs, and along with each active protocol tab, you will notice a number that denotes the number of unique conversations.

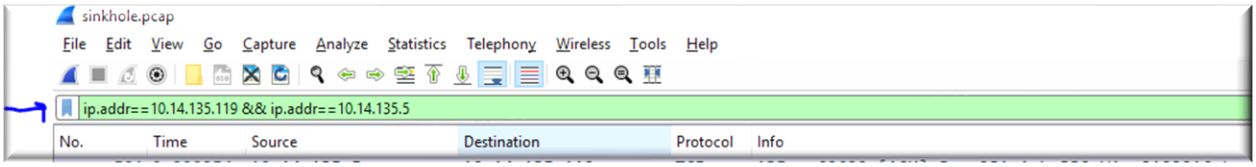


92. For example, if you are looking for the devices that generated a lot of packets and from where major data transfer has happened, then open the Conversations dialog, go to the IPv4 tab, and sort the packets column in a descending order. Here, the device listed in the first row is your answer. Take a look at the following screenshot that illustrates the same.



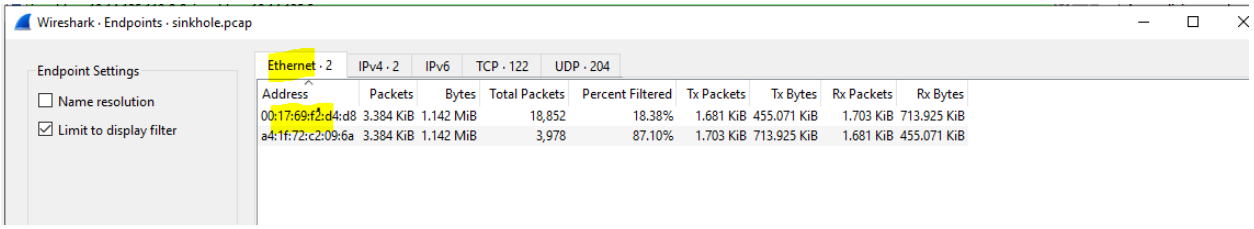
Ethernet · 9	IPv4 · 69	IPv6	TCP · 592	UDP · 204							
Address A	Address B	Packets	Bytes	Total Packets	Percent Filtered	Packets A → B	Bytes A → B	Packets A → B	Bytes B → A	Rel Start	Dura
10.14.135.119	192.34.109.19	4,222	1.545 MiB	4,222	100.00%	2,111	470.664 KiB	2,111	1.085 MiB	14845.040587	1274.
10.14.135.119	10.14.135.5	3,465	1.142 MiB	3,465	100.00%	1,721	455.071 KiB	1,744	713.925 KiB	0.000000	21109.
10.14.135.119	87.120.37.231	2,906	1.209 MiB	2,906	100.00%	1,462	410.314 KiB	1,444	827.674 KiB	92.886362	20997.
10.14.135.119	204.79.197.200	2,033	1.601 MiB	2,033	100.00%	837	62.251 KiB	1,196	1.540 MiB	2819.264992	16291.
10.14.135.119	91.92.109.10	916	241.200 KiB	916	100.00%	498	168.682 KiB	418	72.519 KiB	1893.651863	5289.
10.14.135.119	45.95.11.201	513	460.111 KiB	513	100.00%	191	10.601 KiB	322	449.511 KiB	0.144709	2.
10.14.135.119	52.183.220.149	431	146.627 KiB	431	100.00%	183	30.945 KiB	248	115.682 KiB	137.893584	18105.
10.14.135.119	31.13.195.145	335	90.708 KiB	335	100.00%	183	65.935 KiB	152	24.773 KiB	142.917532	1869.
10.14.135.119	74.6.143.26	318	93.612 KiB	318	100.00%	166	18.538 KiB	152	75.074 KiB	153.541621	19749.
10.14.135.119	13.107.21.200	267	104.610 KiB	267	100.00%	128	23.449 KiB	139	81.161 KiB	925.965680	18182.
10.14.135.119	98.137.11.164	266	69.619 KiB	266	100.00%	142	16.188 KiB	124	53.432 KiB	153.107605	19790.
10.14.135.119	162.219.225.118	164	43.133 KiB	164	100.00%	83	9.982 KiB	81	33.150 KiB	3127.013078	16486.
10.14.135.119	176.32.103.205	149	50.863 KiB	149	100.00%	70	7.776 KiB	79	43.087 KiB	539.488635	19073.
10.14.135.119	52.185.211.133	105	26.447 KiB	105	100.00%	46	7.657 KiB	59	18.790 KiB	3832.413348	12510.
10.14.135.119	74.6.231.21	103	32.284 KiB	103	100.00%	55	6.223 KiB	48	26.062 KiB	4168.507654	11264.
10.14.135.119	98.137.11.163	97	25.505 KiB	97	100.00%	51	5.980 KiB	46	19.524 KiB	4167.911817	11265.
10.14.135.119	74.6.143.25	95	25.154 KiB	95	100.00%	51	5.980 KiB	44	19.174 KiB	2250.537468	9552.
10.14.135.119	54.239.28.85	91	29.738 KiB	91	100.00%	45	4.890 KiB	46	24.849 KiB	8824.926990	4628.
10.14.135.119	205.251.242.103	90	29.816 KiB	90	100.00%	44	4.902 KiB	46	24.914 KiB	15748.425909	5524.
10.14.135.119	40.83.247.108	79	18.963 KiB	79	100.00%	37	6.832 KiB	42	12.131 KiB	10828.505433	10092.
10.14.135.119	13.249.31.218	74	27.674 KiB	74	100.00%	37	3.949 KiB	37	23.725 KiB	20390.771435	882.
10.14.135.119	52.184.215.140	70	23.256 KiB	70	100.00%	36	11.397 KiB	34	11.858 KiB	14202.511948	34.
10.14.135.119	13.107.5.88	68	29.390 KiB	68	100.00%	30	3.654 KiB	38	25.735 KiB	1826.045253	14525.
10.14.135.119	104.94.77.31	67	7.691 KiB	67	100.00%	36	3.189 KiB	31	4.502 KiB	198.183192	19717.
10.14.135.119	40.126.28.11	62	32.143 KiB	62	100.00%	28	7.865 KiB	34	24.277 KiB	1477.602776	12724.
10.14.135.119	40.126.29.5	60	31.593 KiB	60	100.00%	27	7.681 KiB	33	23.912 KiB	10828.932818	3372.
10.14.135.119	99.86.100.195	52	18.623 KiB	52	100.00%	26	2.736 KiB	26	15.887 KiB	12231.244231	1221.
10.14.135.119	10.14.135.91	49	10.247 KiB	49	100.00%	26	6.072 KiB	23	4.175 KiB	15650.904204	10.
10.14.135.119	13.226.182.163	48	18.149 KiB	48	100.00%	22	2.263 KiB	26	15.887 KiB	539.979342	812.
10.14.135.119	52.168.112.66	48	17.368 KiB	48	100.00%	21	6.442 KiB	27	10.926 KiB	1831.172482	6413.
10.14.135.119	142.251.33.46	45	14.411 KiB	45	100.00%	22	2.440 KiB	23	11.971 KiB	3564.860624	428.
10.14.135.119	40.126.29.9	42	25.441 KiB	42	100.00%	17	6.331 KiB	25	19.110 KiB	11726.072224	0.
10.14.135.119	52.109.12.18	41	19.880 KiB	41	100.00%	17	1.814 KiB	24	18.065 KiB	1117.192429	119.
10.14.135.119	104.214.104.116	38	7.913 KiB	38	100.00%	20	2.612 KiB	18	5.301 KiB	14235.819516	0.
10.14.135.119	216.58.193.142	37	7.670 KiB	37	100.00%	19	2.503 KiB	18	5.167 KiB	14342.899167	870.
10.14.135.119	142.250.68.174	36	7.871 KiB	36	100.00%	18	2.577 KiB	18	5.294 KiB	5388.927594	2124.
10.14.135.119	142.251.33.3	32	9.820 KiB	32	100.00%	14	2.720 KiB	18	7.101 KiB	19658.465354	0.
10.14.135.119	142.250.114.94	31	9.770 KiB	31	100.00%	13	2.667 KiB	18	7.103 KiB	1671.359526	0.

93. In the first row, we can see how many packets/bytes have been sent and received by each endpoint and the total elapsed duration. If you wish to create a filter for the same, right click on the first row and then create the respective expression you are thinking about. I chose the first option, A<->B, which only shows packets that are associated with Address A and Address B:
94. The respective filter will be inserted in the Display Filter dialog, as shown in the following screenshot:

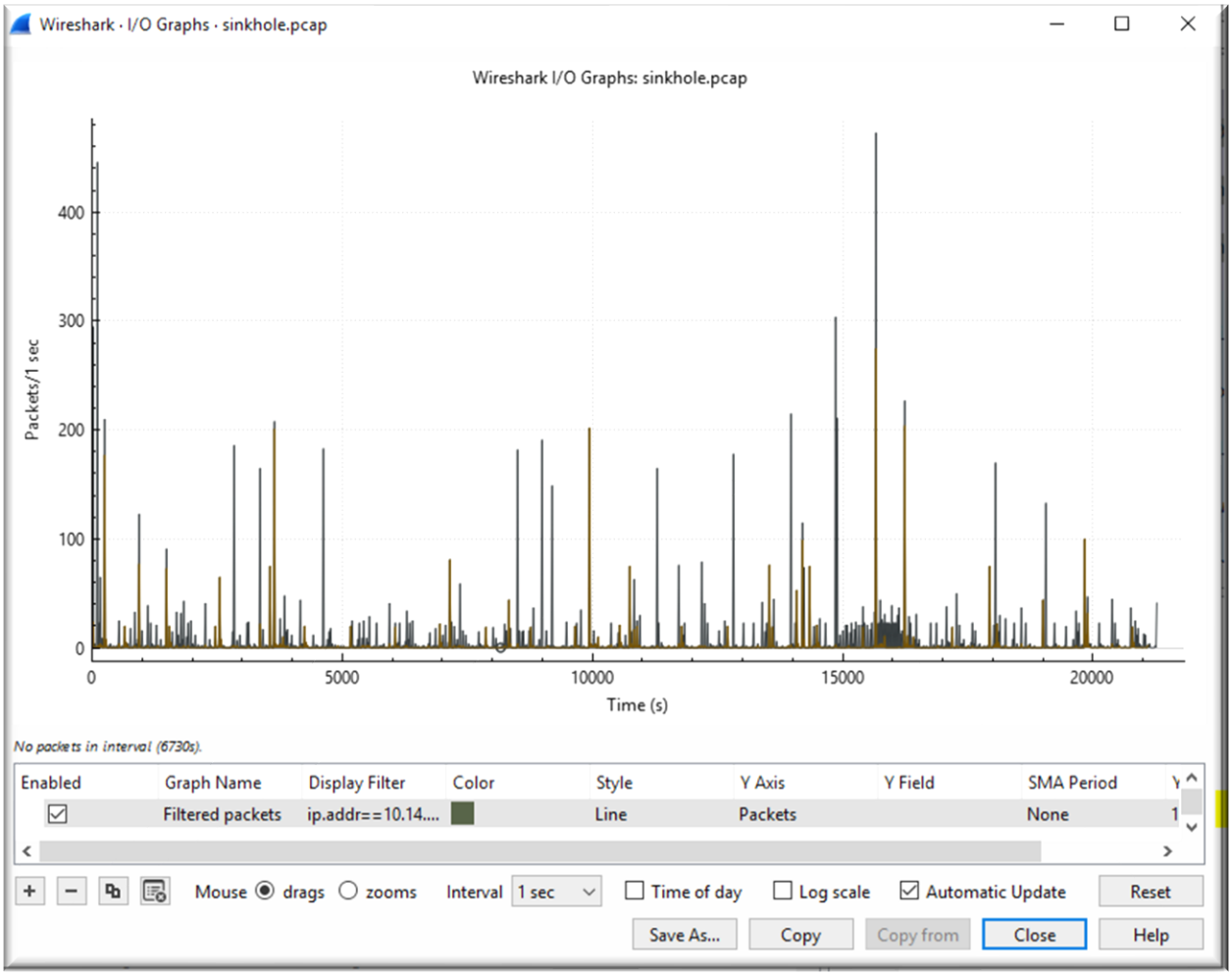


95. **Endpoints**, Two devices that share data with each other are often referred to as endpoints with reference to Wireshark. Every host is able to communicate with the help of an Network Interface Card (NIC) that holds a physical address (often termed as a MAC address), and the same address is used for communication over a local network. Devices that communicate in this

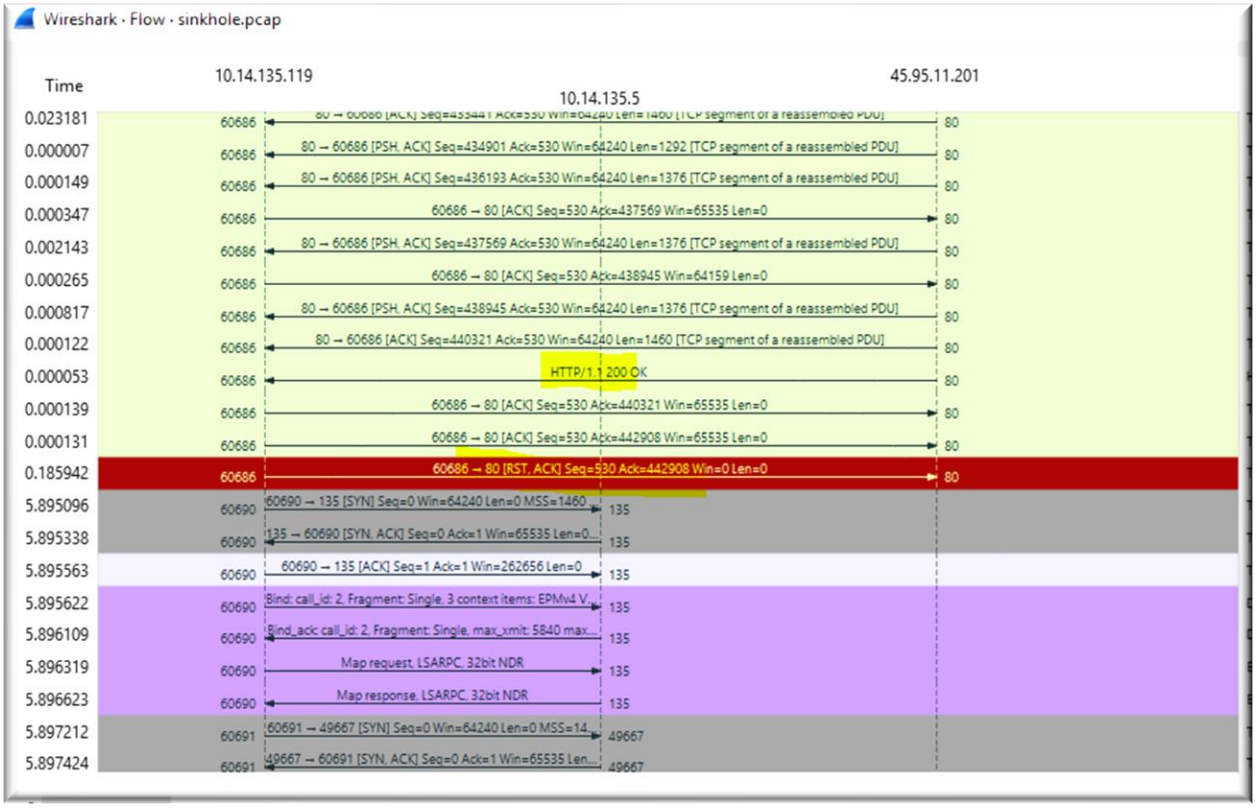
kind of infrastructure are termed as endpoints. Wireshark gives us the facility of analyzing and collecting information regarding these two devices



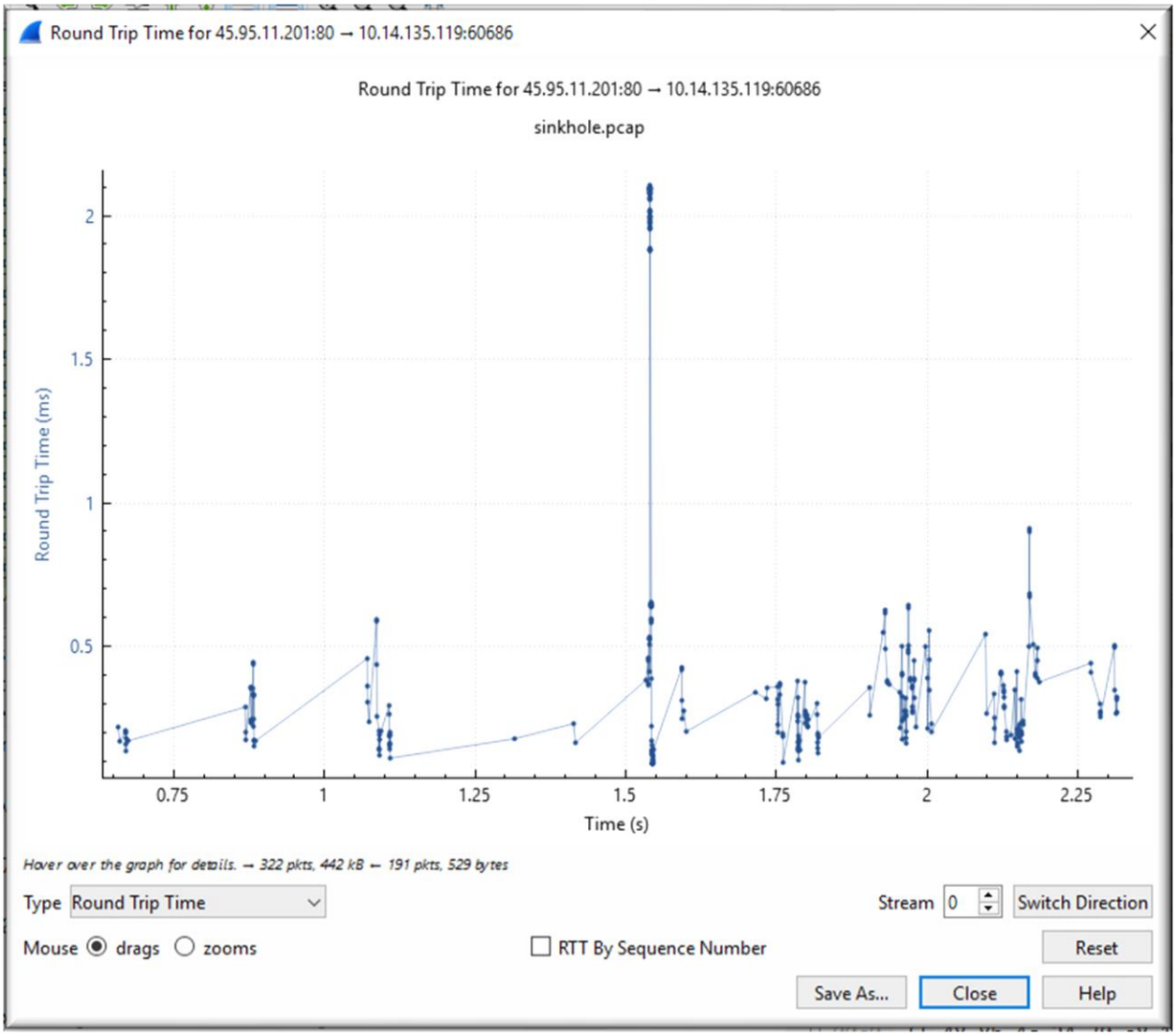
96. **IO Graphs,** is one of the basic graphs that are created using the packets available in the capture file. To create the IO graph, select any TCP packet in your capture file and then click on IO Graph under Statistics. Refer to the following screenshot:



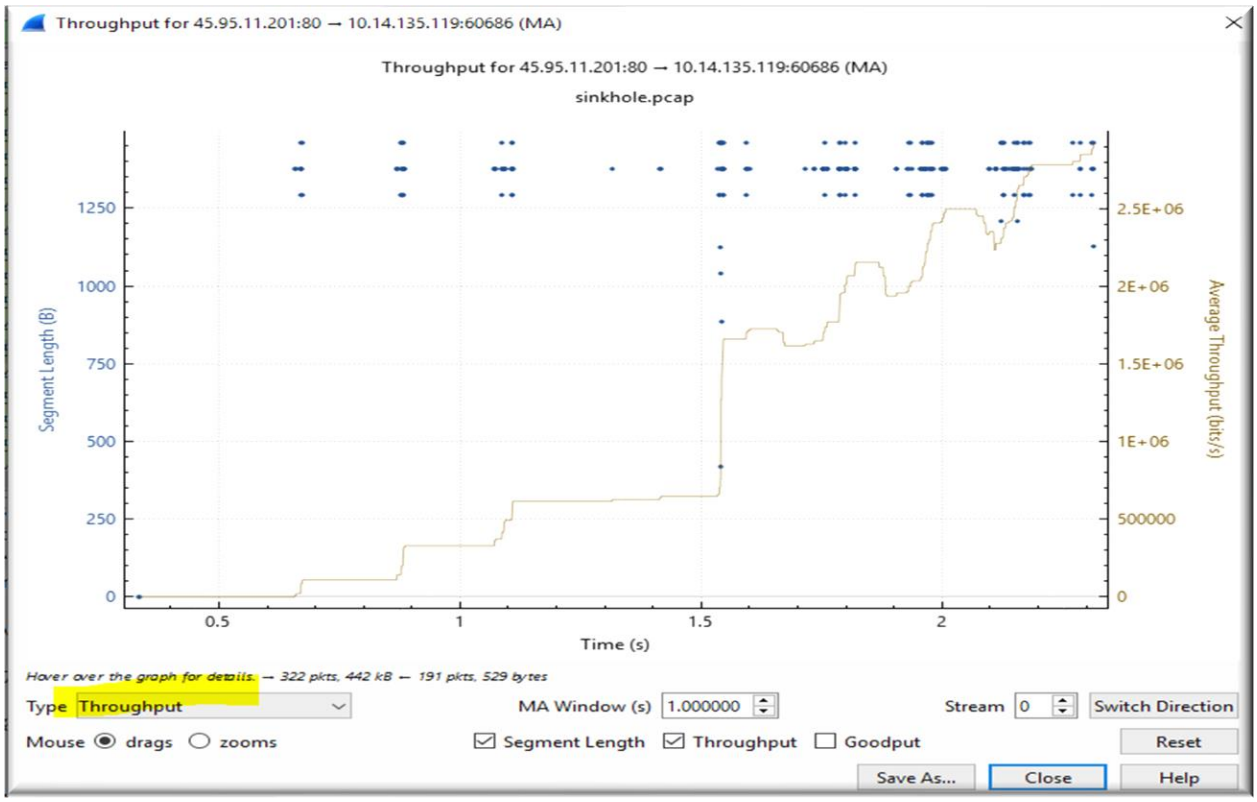
97. **Flow graph,** is one of the nicest features in Wireshark, where we are assisted with troubleshooting capabilities in scenarios like facing a lot of dropped connections, lost frames, retransmission traffic, and more. Flow graphs let us create a column-based graph, which summarizes the flow of traffic between two endpoints, and it even lets us export the results in a simple text-based format. This is the easiest way of verifying the connection between client and server. Refer below screenshot:



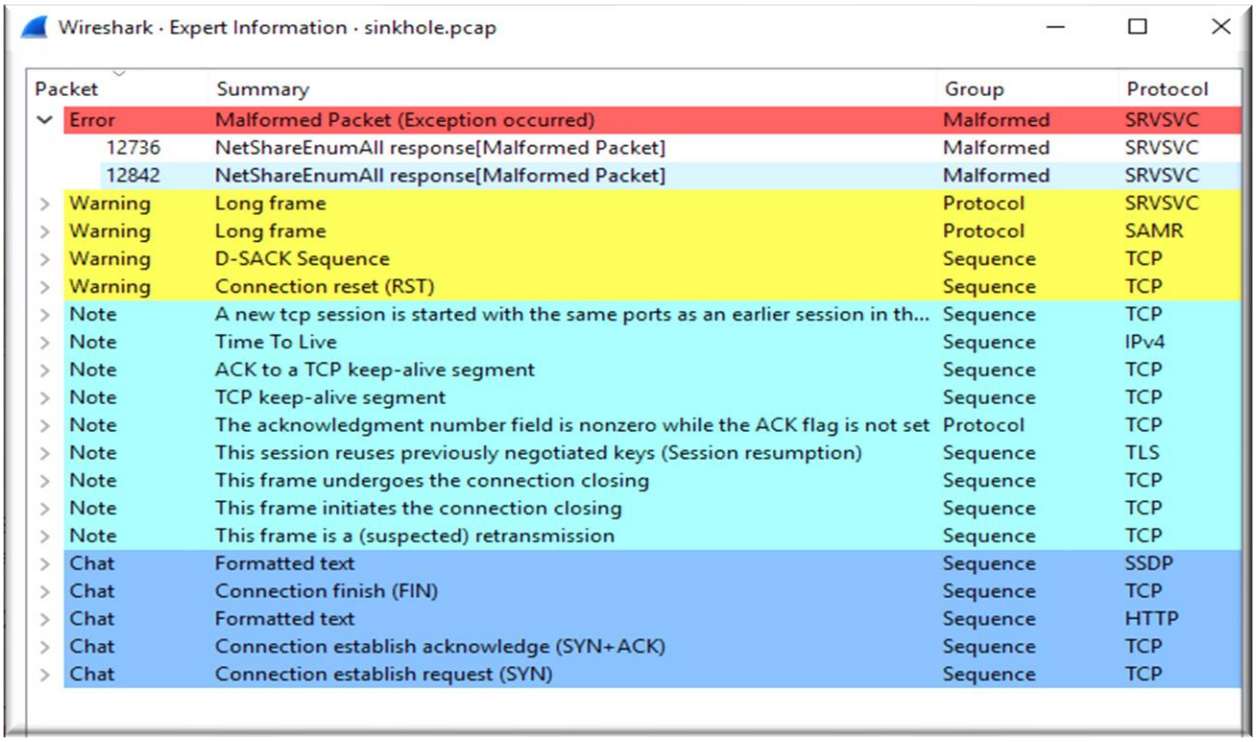
98. As we can see the communication above between client(10.14.135.119) and server(10.14.135.5), post receiving HTTP/1.1 200 OK from the server, wherein malicious software was transferred in the form of an attachment, we observe the packet drop immediately as highlighted in red as the client sends reset to the server (RST, ACK) due to loss in connection.
99. **Round-trip time (RTT)** is the duration in which the ACK for a packet that is sent is received, that is, for every packet sent from a host, there is an ACK received (TCP communication), which determines the successful delivery of the packet. The total time that is consumed from the transfer of the packet to the ACK for the same is called round trip time. Follow these steps to create one for yourself: Select any TCP packet in your packet list pane. Navigate to Statistics | TCP Stream Graph | Round Trip Time Graph. The x axis represents the TCP sequence number, and the y axis represents the RTT in milliseconds. Each plotted point on the graph represents the RTT of a packet. If you are not seeing anything in your graph, then you might have selected an opposite directional packet. RTT graphs are often used by network admins to identify any congestion or latency that can make your network perform slowly.



100. **Throughput graph**, is very similar to the IO graph that depicts the traffic flow. However, it is different in one important aspect that Throughput graphs depict the unidirectional traffic whereas IO graphs depict the traffic in both directions.



101. **Expert Information** The information in the Expert Infos dialog is populated by the dissectors that enable the translation of every protocol that is well known to Wireshark. Refer screenshot below:



Packet	Summary	Group	Protocol
Error	Malformed Packet (Exception occurred)	Malformed	SRVSVC
12736	NetShareEnumAll response[Malformed Packet]	Malformed	SRVSVC
12842	NetShareEnumAll response[Malformed Packet]	Malformed	SRVSVC
Warning	Long frame	Protocol	SRVSVC
Warning	Long frame	Protocol	SAMR
Warning	D-SACK Sequence	Sequence	TCP
Warning	Connection reset (RST)	Sequence	TCP
Note	A new tcp session is started with the same ports as an earlier session in th...	Sequence	TCP
Note	Time To Live	Sequence	IPv4
Note	ACK to a TCP keep-alive segment	Sequence	TCP
Note	TCP keep-alive segment	Sequence	TCP
Note	The acknowledgment number field is nonzero while the ACK flag is not set	Protocol	TCP
Note	This session reuses previously negotiated keys (Session resumption)	Sequence	TLS
Note	This frame undergoes the connection closing	Sequence	TCP
Note	This frame initiates the connection closing	Sequence	TCP
Note	This frame is a (suspected) retransmission	Sequence	TCP
Chat	Formatted text	Sequence	SSDP
Chat	Connection finish (FIN)	Sequence	TCP
Chat	Formatted text	Sequence	HTTP
Chat	Connection establish acknowledge (SYN+ACK)	Sequence	TCP
Chat	Connection establish request (SYN)	Sequence	TCP