Wireshark: Traffic Analysis

Nmap Scans

Nmap is an industry-standard tool for mapping networks, identifying live hosts and discovering the services. As it is one of the most used network scanner tools, a security analyst should identify the network patterns created with it. This section will cover identifying the most common Nmap scan types.

- TCP connect scans
- SYN scans
- UDP scans

It is essential to know how Nmap scans work to spot scan activity on the network. However, it is impossible to understand the scan details without using the correct filters. Below are the base filters to probe Nmap scan behaviour on the network.

TCP flags in a nutshell.

Notes	Wireshark Filters
Global search.	• `tcp` • `udp`
• Only SYN flag. • SYN flag is set. The rest of the bits are not important.	• `tcp.flags == 2` • `tcp.flags.syn == 1`
• Only ACK flag. • ACK flag is set. The rest of the bits are not important.	• `tcp.flags == 16` • `tcp.flags.ack == 1`

• Only SYN, ACK flags. • SYN and ACK are set. The rest of the bits are not important.	• `tcp.flags == 18` • `(tcp.flags.syn == 1) and (tcp.flags.ack == 1)`
• Only RST flag. • RST flag is set. The rest of the bits are not important.	• `tcp.flags == 4` • `tcp.flags.reset == 1`
• Only RST, ACK flags. • RST and ACK are set. The rest of the bits are not important.	• `tcp.flags == 20` • `(tcp.flags.reset == 1) and (tcp.flags.ack == 1)`
• Only FIN flag • FIN flag is set. The rest of the bits are not important.	• `tcp.flags == 1` • `tcp.flags.fin == 1`

TCP Connect Scans

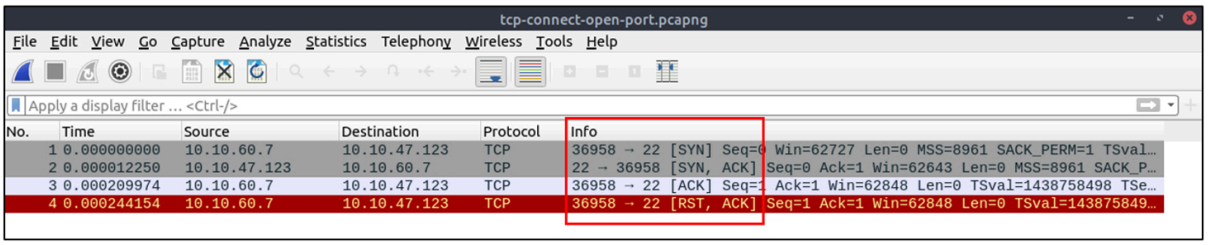
TCP Connect Scan in a nutshell:

- Relies on the three-way handshake (needs to finish the handshake process).
- Usually conducted with `nmap -sT` command.
- Used by non-privileged users (only option for a non-root user).
- Usually has a windows size larger than 1024 bytes as the request expects some data due to the nature of the protocol.

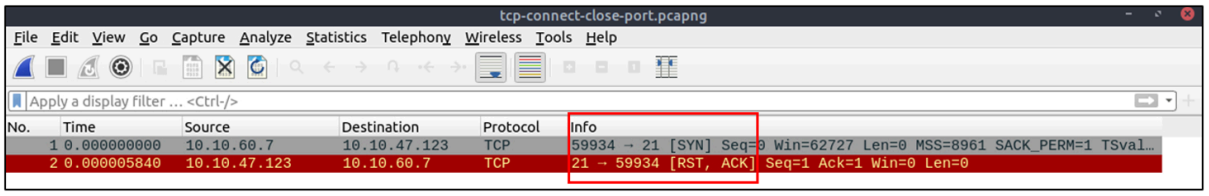
Open TCP Port	Open TCP Port	Closed TCP Port
• SYN --> • <-- SYN, ACK • ACK -->	• SYN --> • <-- SYN, ACK • ACK --> • RST, ACK -->	• SYN --> • <-- RST, ACK

The images below show the three-way handshake process of the open and close TCP ports. Images and pcap samples are split to make the investigation easier and understand each case's details.

Open TCP port (Connect):



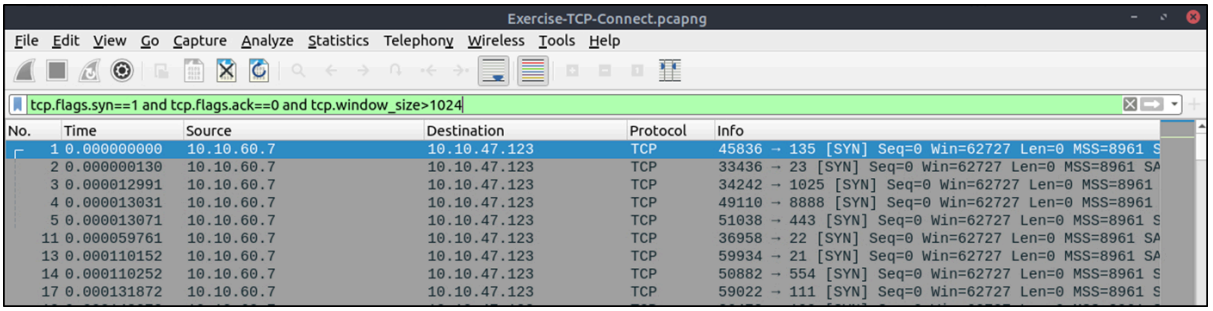
Closed TCP port (Connect):



The above images provide the patterns in isolated traffic. However, it is not always easy to spot the given patterns in big capture files. Therefore analysts need to use a generic filter to view the initial anomaly patterns, and then it will be easier to

focus on a specific traffic point. The given filter shows the TCP Connect scan patterns in a capture file.

```
tcp.flags.syn==1 and tcp.flags.ack==0 and tcp.window_size > 1024
```



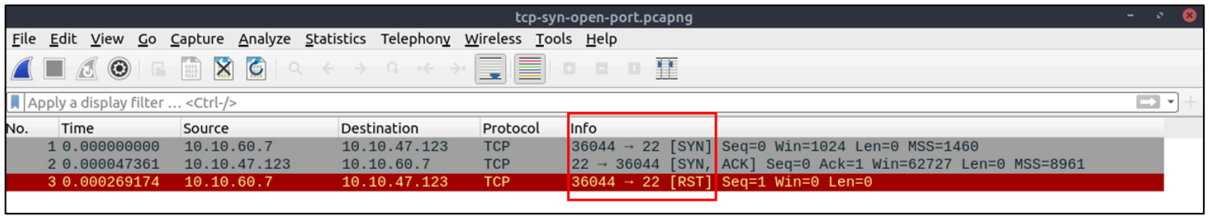
SYN Scans

TCP SYN Scan in a nutshell:

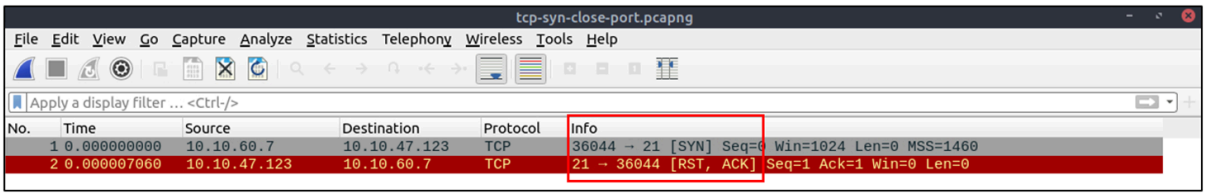
- Doesn't rely on the three-way handshake (no need to finish the handshake process).
- Usually conducted with `nmap -sS` command.
- Used by privileged users.
- Usually have a size less than or equal to 1024 bytes as the request is not finished and it doesn't expect to receive data.

Open TCP Port	Close TCP Port
• SYN --> • <-- SYN,ACK • RST-->	• SYN --> • <-- RST,ACK

Open TCP port (SYN):

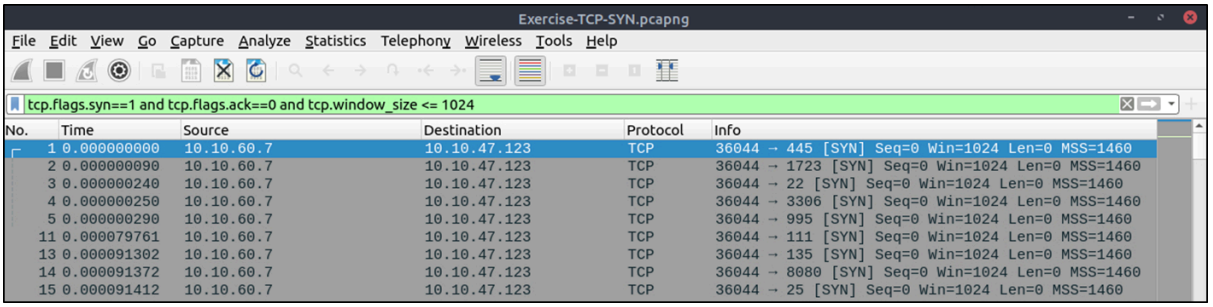


Closed TCP port (SYN):



The given filter shows the TCP SYN scan patterns in a capture file.

```
tcp.flags.syn==1 and tcp.flags.ack==0 and tcp.window_size <= 1024
```


Exercise-TCP-SYN.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.flags.syn==1 and tcp.flags.ack==0 and tcp.window_size <= 1024
No. Time Source Destination Protocol Info
1 0.000000000 10.10.60.7 10.10.47.123 TCP 36044 → 445 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
2 0.000000090 10.10.60.7 10.10.47.123 TCP 36044 → 1723 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3 0.000000240 10.10.60.7 10.10.47.123 TCP 36044 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4 0.000000250 10.10.60.7 10.10.47.123 TCP 36044 → 3306 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5 0.000000290 10.10.60.7 10.10.47.123 TCP 36044 → 995 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
11 0.000079761 10.10.60.7 10.10.47.123 TCP 36044 → 111 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
13 0.000091302 10.10.60.7 10.10.47.123 TCP 36044 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
14 0.000091372 10.10.60.7 10.10.47.123 TCP 36044 → 8080 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
15 0.000091412 10.10.60.7 10.10.47.123 TCP 36044 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

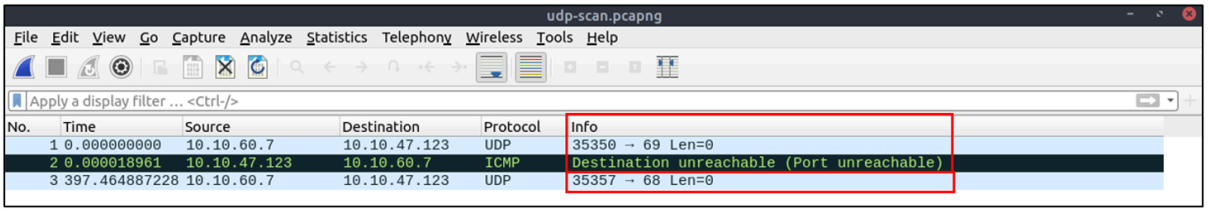
UDP Scans

UDP Scan in a nutshell:

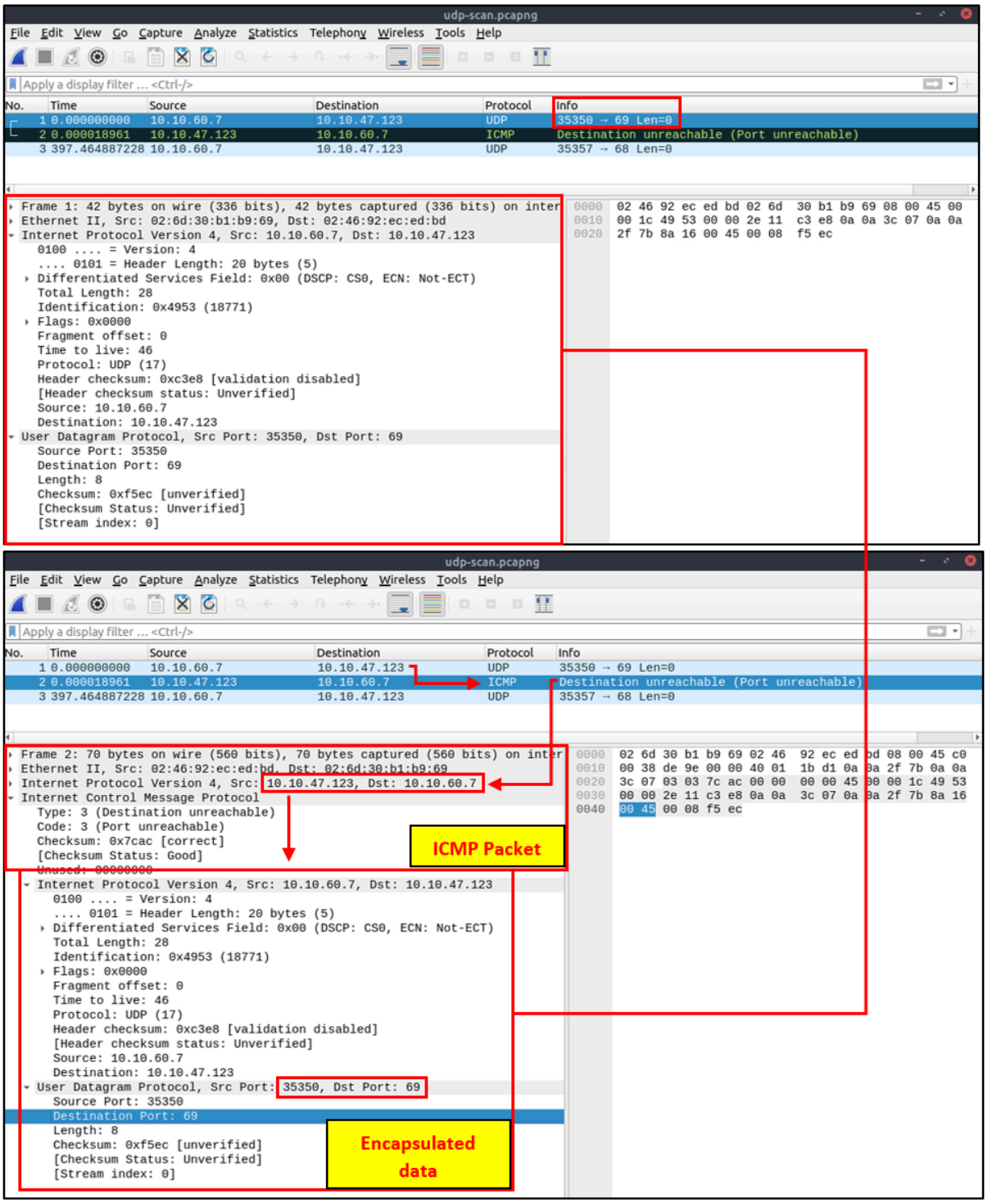
- Doesn't require a handshake process
- No prompt for open ports
- ICMP error message for close ports
- Usually conducted with `nmap -sU` command.

Open UDP Port	Closed UDP Port
• UDP packet -->	• UDP packet --> • ICMP Type 3, Code 3 message. (Destination unreachable, port unreachable)

Closed (port no 69) and open (port no 68) UDP ports:

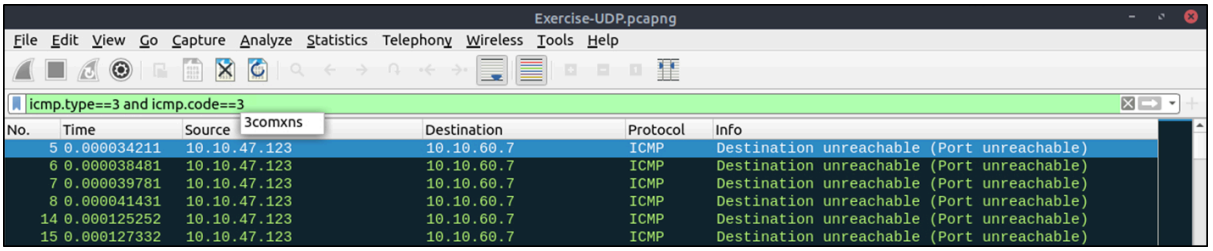


The above image shows that the closed port returns an ICMP error packet. No further information is provided about the error at first glance, so how can an analyst decide where this error message belongs? The ICMP error message uses the original request as encapsulated data to show the source/reason of the packet. Once you expand the ICMP section in the packet details pane, you will see the encapsulated data and the original request, as shown in the below image.



The given filter shows the UDP scan patterns in a capture file.

```
icmp.type==3 and icmp.code==3
```



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

Answer the questions below

Use the "Desktop/exercise-pcaps/nmap/Exercise.pcapng" file.
What is the total number of the "TCP Connect" scans?

ARP Poisoning/Spoofing (A.K.A. Man In The Middle Attack)

ARP protocol, or Address Resolution Protocol (ARP), is the technology responsible for allowing devices to identify themselves on a network. Address Resolution Protocol Poisoning (also known as ARP Spoofing or Man In The Middle (MITM) attack) is a type of attack that involves network jamming/manipulating by sending malicious ARP packets to the default gateway. The ultimate aim is to manipulate the "IP to MAC address table" and sniff the traffic of the target host.

There are a variety of tools available to conduct ARP attacks. However, the mindset of the attack is static, so it is easy to detect such an attack by knowing the ARP protocol workflow and Wireshark skills.

ARP analysis in a nutshell:

- Works on the local network
- Enables the communication between MAC addresses
- Not a secure protocol
- Not a routable protocol
- It doesn't have an authentication function
- Common patterns are request & response, announcement and gratuitous packets.

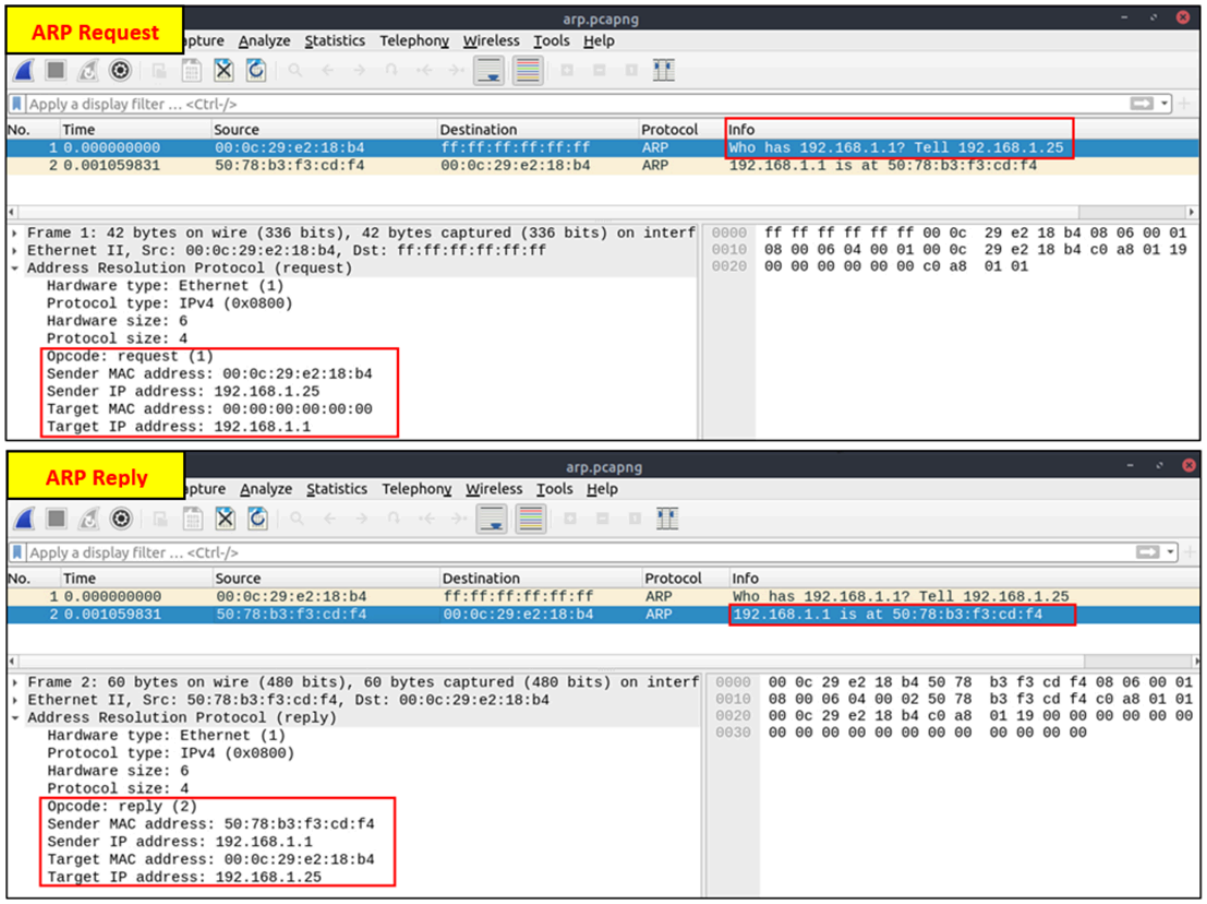
Before investigating the traffic, let's review some legitimate and suspicious ARP packets. The legitimate requests are similar to the shown picture: a broadcast request that asks if any of the available hosts use an IP address and a reply from the host that uses the particular IP address.

Notes	Wireshark filter
Global search	• `arp`

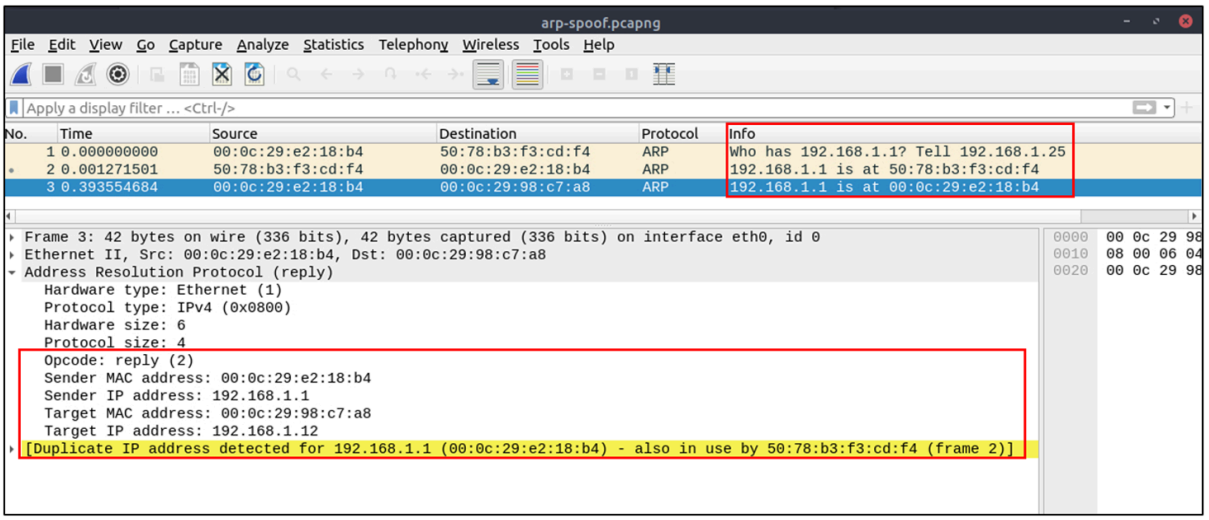
"ARP" options for grabbing the low-hanging fruits:

- Opcode 1: ARP requests.
- Opcode 2: ARP responses.
- Hunt: Arp scanning
- Hunt: Possible ARP poisoning detection
- Hunt: Possible ARP flooding from detection:

- `arp.opcode == 1`
- `arp.opcode == 2`
- `arp.dst.hw_mac==00:00:00:00:00:00`
- `arp.duplicate-address-detected or arp.duplicate-address-frame`
- `((arp) && (arp.opcode == 1)) && (arp.src.hw_mac == target-mac-address)`



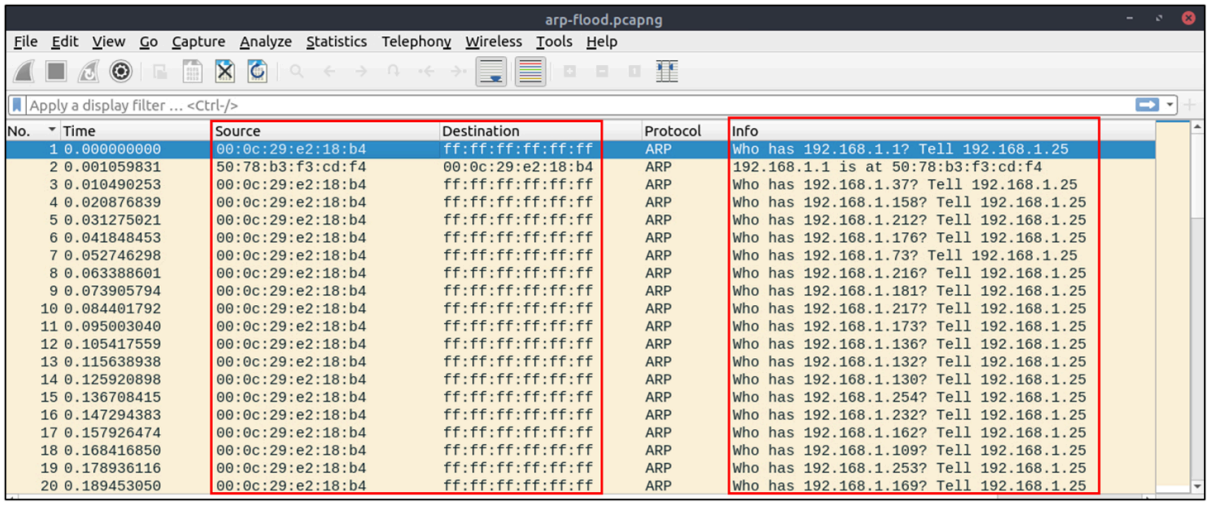
A suspicious situation means having two different ARP responses (conflict) for a particular IP address. In that case, Wireshark's expert info tab warns the analyst. However, it only shows the second occurrence of the duplicate value to highlight the conflict. Therefore, identifying the malicious packet from the legitimate one is the analyst's challenge. A possible IP spoofing case is shown in the picture below.



Here, knowing the network architecture and inspecting the traffic for a specific time frame can help detect the anomaly. As an analyst, you should take notes of your findings before going further. This will help you be organised and make it easier to correlate the further findings. Look at the given picture; there is a conflict; the MAC address that ends with "b4" crafted an ARP request with the "192.168.1.25" IP address, then claimed to have the "192.168.1.1" IP address.

Notes	**Detection Notes**	Findings
Possible IP address match.	1 IP address announced from a MAC address.	• MAC: 00:0c:29:e2:18:b4 • IP: 192.168.1.25
Possible ARP spoofing attempt.	2 MAC addresses claimed the same IP address (192.168.1.1). The " 192.168.1.1" IP address is a possible gateway address.	• MAC1: 50:78:b3:f3:cd:f4 • MAC 2: 00:0c:29:e2:18:b4
Possible ARP flooding attempt.	The MAC address that ends with "b4" claims to have a different/new IP address.	• MAC: 00:0c:29:e2:18:b4 • IP: 192.168.1.1

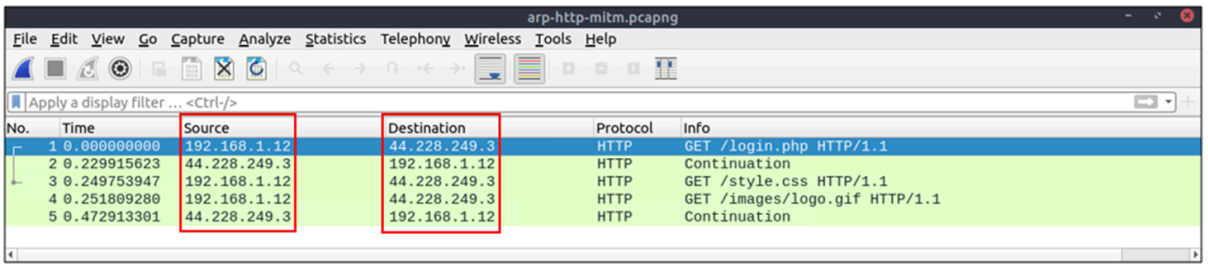
Let's keep inspecting the traffic to spot any other anomalies. Note that the case is split into multiple capture files to make the investigation easier.



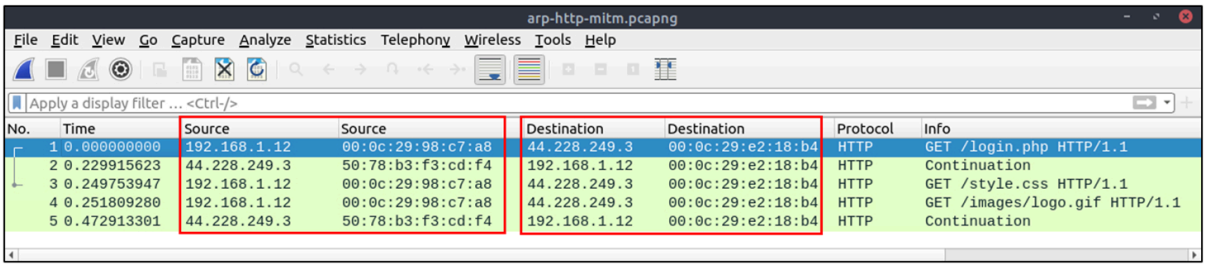
At this point, it is evident that there is an anomaly. A security analyst cannot ignore a flood of ARP requests. This could be malicious activity, scan or network problems. There is a new anomaly; the MAC address that ends with "b4" crafted multiple ARP requests with the "192.168.1.25" IP address. Let's focus on the source of this anomaly and extend the taken notes.

Notes	Detection Notes	Findings
Possible IP address match.	1 IP address announced from a MAC address.	• MAC: 00:0c:29:e2:18:b4 • IP: 192.168.1.25
Possible ARP spoofing attempt.	2 MAC addresses claimed the same IP address (192.168.1.1). The " 192.168.1.1" IP address is a possible gateway address.	• MAC1: 50:78:b3:f3:cd:f4 • MAC 2: 00:0c:29:e2:18:b4
Possible ARP spoofing attempt.	The MAC address that ends with "b4" claims to have a different/new IP address.	• MAC: 00:0c:29:e2:18:b4 • IP: 192.168.1.1
Possible ARP flooding attempt.	The MAC address that ends with "b4" crafted multiple ARP requests against a range of IP addresses.	• MAC: 00:0c:29:e2:18:b4 • IP: 192.168.1.xxx

Up to this point, it is evident that the MAC address that ends with "b4" owns the "192.168.1.25" IP address and crafted suspicious ARP requests against a range of IP addresses. It also claimed to have the possible gateway address as well. Let's focus on other protocols and spot the reflection of this anomaly in the following sections of the time frame.



There is HTTP traffic, and everything looks normal at the IP level, so there is no linked information with our previous findings. Let's add the MAC addresses as columns in the packet list pane to reveal the communication behind the IP addresses.



One more anomaly! The MAC address that ends with "b4" is the destination of all HTTP packets! It is evident that there is a MITM attack, and the attacker is the host with the MAC address that ends with "b4". All traffic linked to "192.168.1.12" IP addresses is forwarded to the malicious host. Let's summarise the findings before concluding the investigation.

Detection Notes	Findings	
IP to MAC matches.	3 IP to MAC address matches.	• MAC: 00:0c:29:e2:18:b4 = IP: 192.168.1.25 • MAC: 50:78:b3:f3:cd:f4 = IP: 192.1681.1 • MAC: 00:0c:29:98:c7:a8 = IP: 192.168.1.12

Attacker	The attacker created noise with ARP packets.	• MAC: 00:0c:29:e2:18:b4 = IP: 192.168.1.25
Router/gateway	Gateway address.	• MAC: 50:78:b3:f3:cd:f4 = IP: 192.1681.1
Victim	The attacker sniffed all traffic of the victim.	• MAC: 50:78:b3:f3:cd:f4 = IP: 192.1681.12

Detecting these bits and pieces of information in a big capture file is challenging. However, in real-life cases, you will not have "tailored data" ready for investigation. Therefore you need to have the analyst mindset, knowledge and tool skills to filter and detect the anomalies.

Note: In traffic analysis, there are always alternative solutions available. The solution type and the approach depend on the analyst's knowledge and skill level and the available data sources.

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

Identifying Hosts: DHCP, NetBIOS and Kerberos

Identifying Hosts

When investigating a compromise or malware infection activity, a security analyst should know how to identify the hosts on the network apart from IP to MAC address match. One of the best methods is identifying the hosts and users on the network to decide the investigation's starting point and list the hosts and users associated with the malicious traffic/activity.

Usually, enterprise networks use a predefined pattern to name users and hosts. While this makes knowing and following the inventory easier, it has good and bad sides. The good side is that it will be easy to identify a user or host by looking at the name. The bad side is that it will be easy to clone that pattern and live in the enterprise network for adversaries. There are multiple solutions to avoid these kinds of activities, but for a security analyst, it is still essential to have host and user identification skills.

Protocols that can be used in Host and User identification:

- Dynamic Host Configuration Protocol (DHCP) traffic
- NetBIOS (NBNS) traffic
- Kerberos traffic

DHCP Analysis

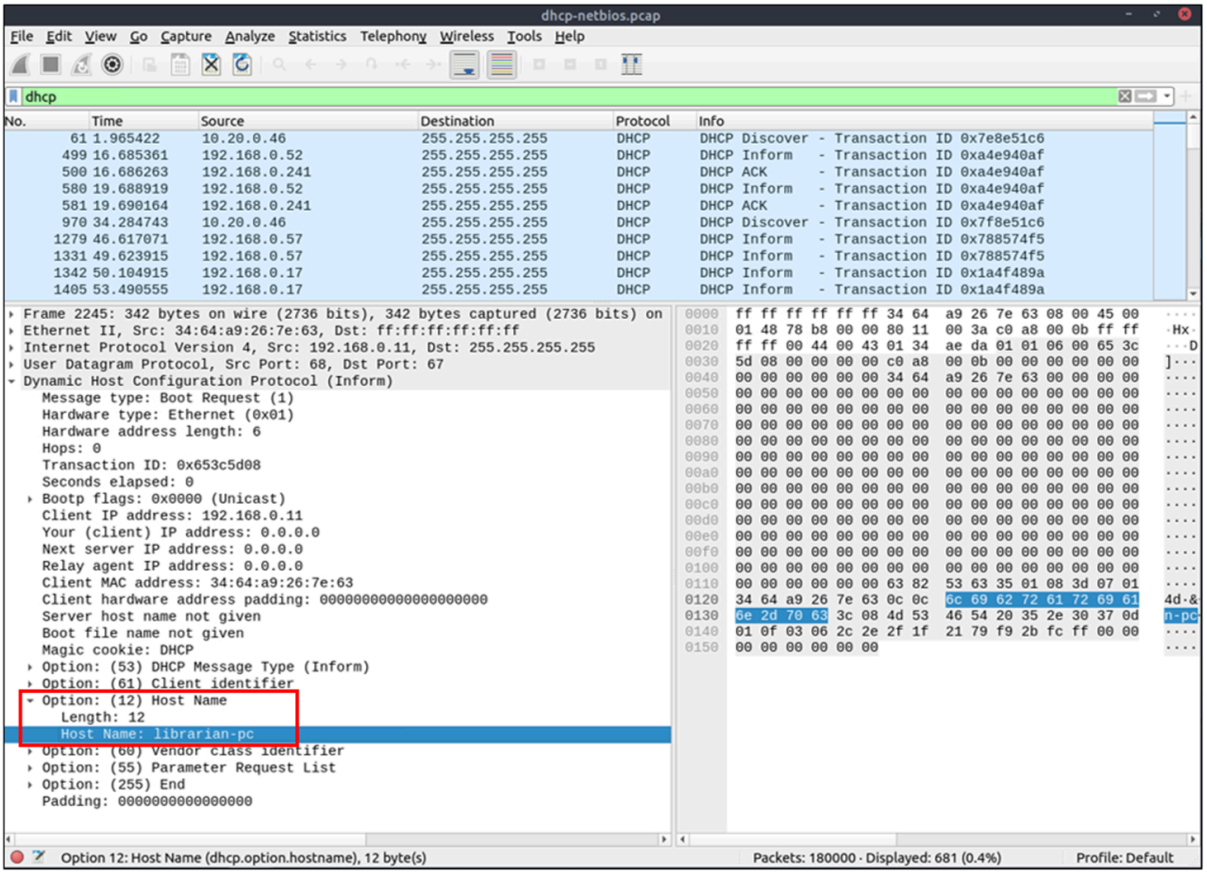
DHCP protocol, or Dynamic Host Configuration Protocol (DHCP), is the technology responsible for managing automatic IP address and required communication parameters assignment.

DHCP investigation in a nutshell:

Notes	Wireshark Filter
Global search.	• `dhcp` or `bootp`

Filtering the proper DHCP packet options is vital to finding an event of interest. • "DHCP Request" packets contain the hostname information • "DHCP ACK" packets represent the accepted requests • "DHCP NAK" packets represent denied requests Due to the nature of the protocol, only "Option 53" (request type) has predefined static values. You should filter the packet type first, and then you can filter the rest of the options by "applying as column" or use the advanced filters like "contains" and "matches".	• Request: `dhcp.option.dhcp == 3` • ACK: `dhcp.option.dhcp == 5` • NAK: `dhcp.option.dhcp == 6`
"DHCP Request" options for grabbing the low-hanging fruits: • Option 12: Hostname. • Option 50: Requested IP address. • Option 51: Requested IP lease time. • Option 61: Client's MAC address.	• `dhcp.option.hostname contains "keyword"`
"DHCP ACK" options for grabbing the low-hanging fruits: • Option 15: Domain name. • Option 51: Assigned IP lease time.	• `dhcp.option.domain_name contains "keyword"`

"DHCP NAK" options for grabbing the low-hanging fruits: • Option 56: Message (rejection details/reason).	As the message could be unique according to the case/situation, It is suggested to read the message instead of filtering it. Thus, the analyst could create a more reliable hypothesis/result by understanding the event circumstances.



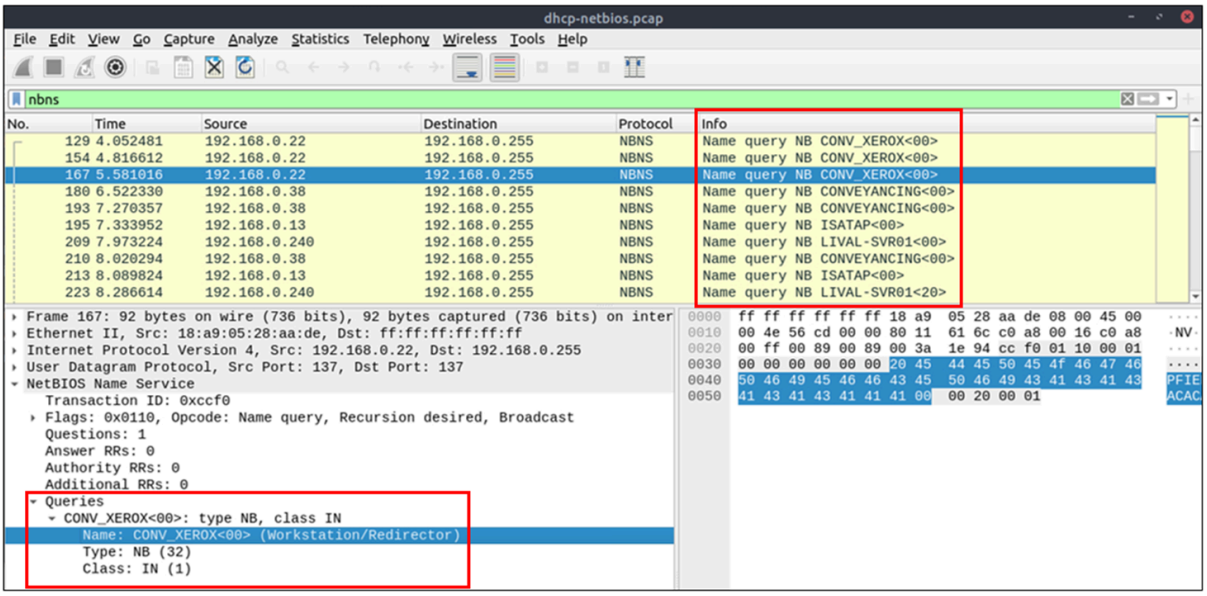
NetBIOS (NBNS) Analysis

NetBIOS or Network Basic Input/Output System is the technology responsible for allowing applications on different hosts to communicate with each other.

NBNS investigation in a nutshell:

Notes	Wireshark Filter

Global search.	• `nbns`
"NBNS" options for grabbing the low-hanging fruits: • Queries: Query details. • Query details could contain "name, Time to live (TTL) and IP address details"	• `nbns.name contains "keyword"`



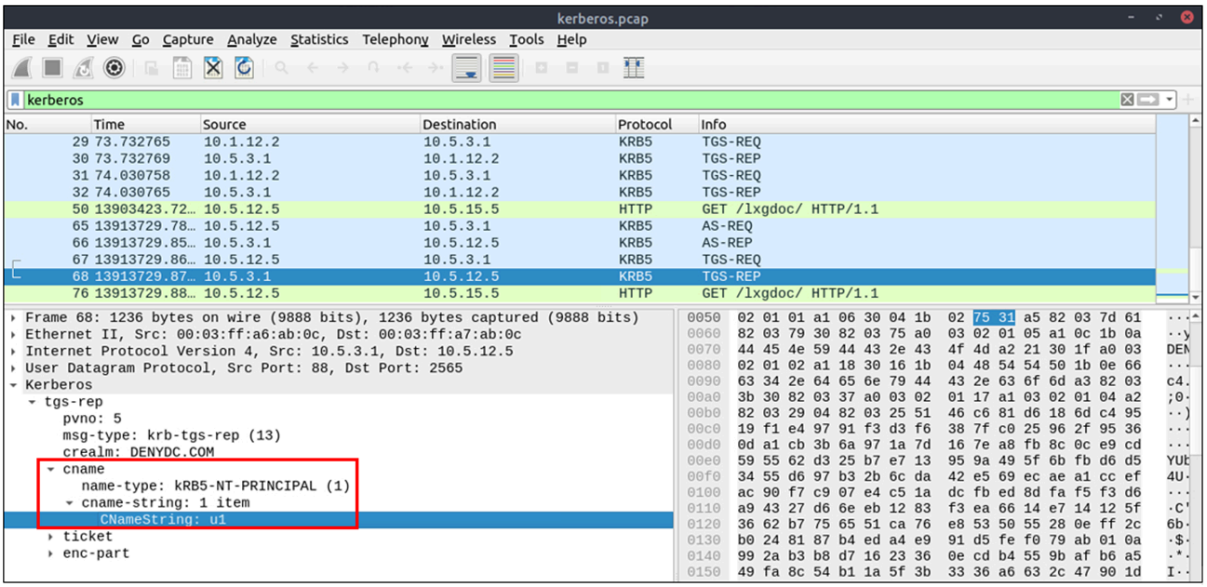
Kerberos Analysis

Kerberos is the default authentication service for Microsoft Windows domains. It is responsible for authenticating service requests between two or more computers over the untrusted network. The ultimate aim is to prove identity securely.

Kerberos investigation in a nutshell:

Notes	Wireshark Filter
Global search.	• `kerberos`

User account search: • CNameString: The username. Note: Some packets could provide hostname information in this field. To avoid this confusion, filter the "$" value. The values end with "$" are hostnames, and the ones without it are user names.	• `kerberos.CNameString contains "keyword"` • `kerberos.CNameString and !(kerberos.CNameString contains "$")`
"Kerberos" options for grabbing the low-hanging fruits: • pvno: Protocol version. • realm: Domain name for the generated ticket. • sname: Service and domain name for the generated ticket. • addresses: Client IP address and NetBIOS name. Note: the "addresses" information is only available in request packets.	• `kerberos.pvno == 5` • `kerberos.realm contains ".org"` • `kerberos.SNameString == "krbtg"`



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

Identifying Hosts

When investigating a compromise or malware infection activity, a security analyst should know how to identify the hosts on the network apart from IP to MAC address match. One of the best methods is identifying the hosts and users on the network to decide the investigation's starting point and list the hosts and users associated with the malicious traffic/activity.

Usually, enterprise networks use a predefined pattern to name users and hosts. While this makes knowing and following the inventory easier, it has good and bad sides. The good side is that it will be easy to identify a user or host by looking at the name. The bad side is that it will be easy to clone that pattern and live in the enterprise network for adversaries. There are multiple solutions to avoid these kinds of activities, but for a security analyst, it is still essential to have host and user identification skills.

Protocols that can be used in Host and User identification:

- Dynamic Host Configuration Protocol (DHCP) traffic
- NetBIOS (NBNS) traffic
- Kerberos traffic

DHCP Analysis

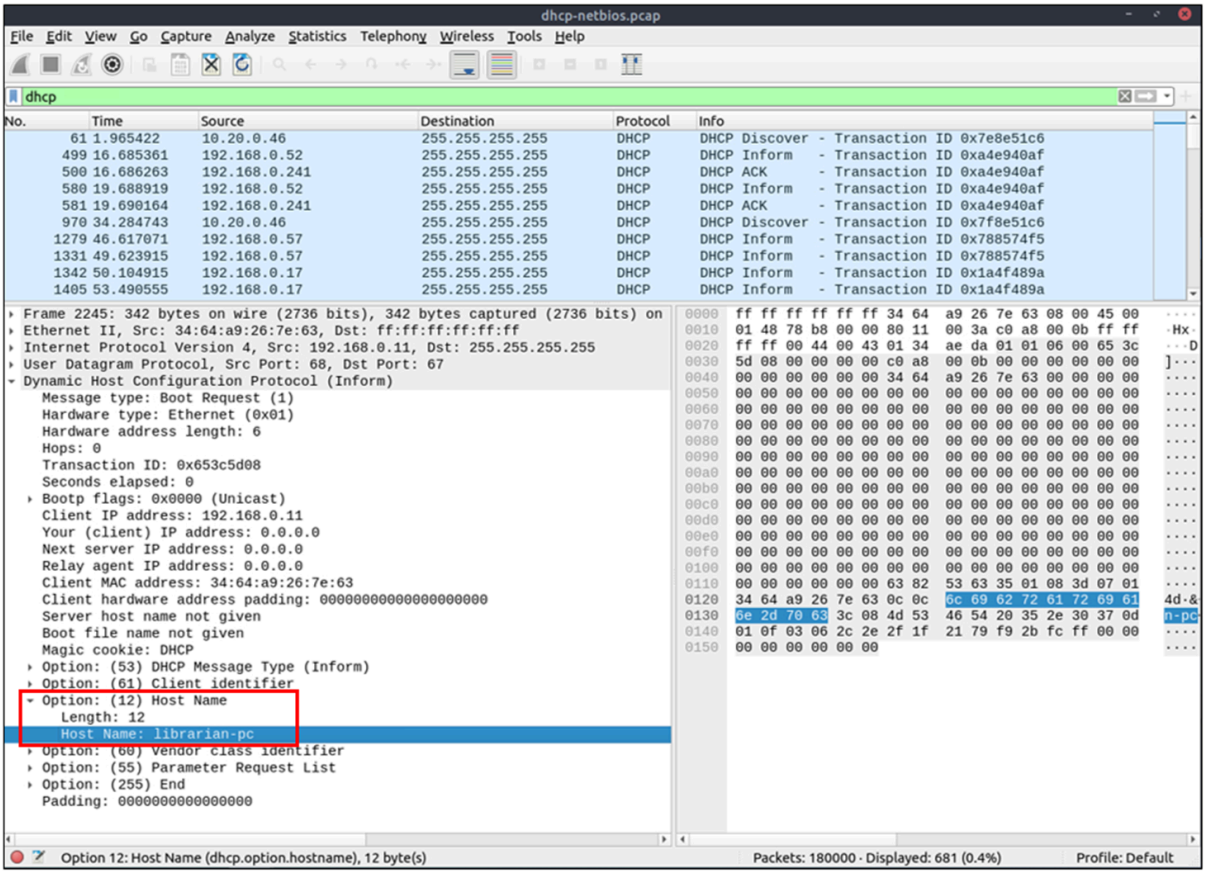
DHCP protocol, or Dynamic Host Configuration Protocol (DHCP), is the technology responsible for managing automatic IP address and required communication parameters assignment.

DHCP investigation in a nutshell:

Notes	Wireshark Filter
Global search.	• `dhcp` or `bootp`

Filtering the proper DHCP packet options is vital to finding an event of interest. • "DHCP Request" packets contain the hostname information • "DHCP ACK" packets represent the accepted requests • "DHCP NAK" packets represent denied requests Due to the nature of the protocol, only "Option 53" (request type) has predefined static values. You should filter the packet type first, and then you can filter the rest of the options by "applying as column" or use the advanced filters like "contains" and "matches".	• Request: `dhcp.option.dhcp == 3` • ACK: `dhcp.option.dhcp == 5` • NAK: `dhcp.option.dhcp == 6`
"DHCP Request" options for grabbing the low-hanging fruits: • Option 12: Hostname. • Option 50: Requested IP address. • Option 51: Requested IP lease time. • Option 61: Client's MAC address.	• `dhcp.option.hostname contains "keyword"`
"DHCP ACK" options for grabbing the low-hanging fruits: • Option 15: Domain name. • Option 51: Assigned IP lease time.	• `dhcp.option.domain_name contains "keyword"`

"DHCP NAK" options for grabbing the low-hanging fruits: • Option 56: Message (rejection details/reason).	As the message could be unique according to the case/situation, It is suggested to read the message instead of filtering it. Thus, the analyst could create a more reliable hypothesis/result by understanding the event circumstances.



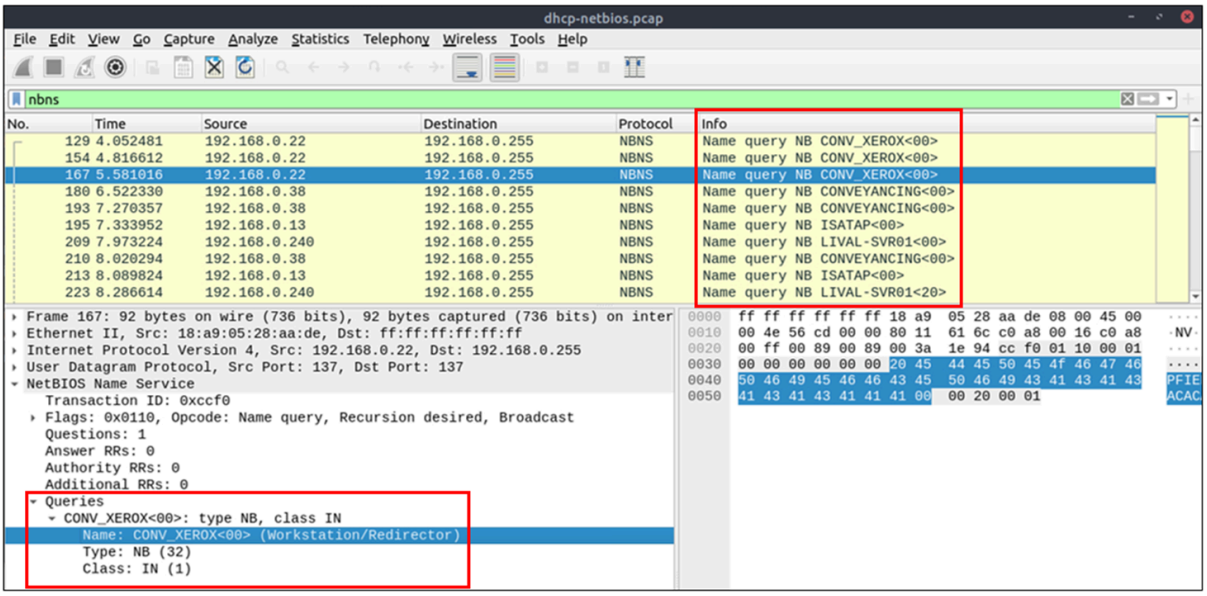
NetBIOS (NBNS) Analysis

NetBIOS or Network Basic Input/Output System is the technology responsible for allowing applications on different hosts to communicate with each other.

NBNS investigation in a nutshell:

Notes	Wireshark Filter

Global search.	• `nbns`
"NBNS" options for grabbing the low-hanging fruits: • Queries: Query details. • Query details could contain "name, Time to live (TTL) and IP address details"	• `nbns.name contains "keyword"`



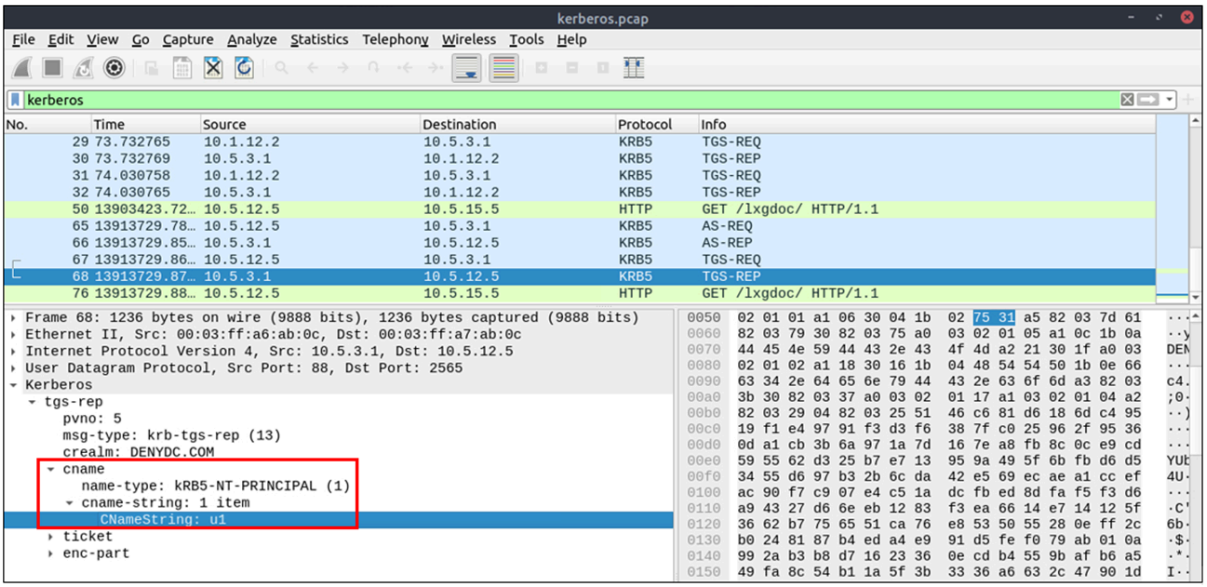
Kerberos Analysis

Kerberos is the default authentication service for Microsoft Windows domains. It is responsible for authenticating service requests between two or more computers over the untrusted network. The ultimate aim is to prove identity securely.

Kerberos investigation in a nutshell:

Notes	Wireshark Filter
Global search.	• `kerberos`

User account search: • CNameString: The username. Note: Some packets could provide hostname information in this field. To avoid this confusion, filter the "$" value. The values end with "$" are hostnames, and the ones without it are user names.	• `kerberos.CNameString contains "keyword"` • `kerberos.CNameString and !(kerberos.CNameString contains "$")`
"Kerberos" options for grabbing the low-hanging fruits: • pvno: Protocol version. • realm: Domain name for the generated ticket. • sname: Service and domain name for the generated ticket. • addresses: Client IP address and NetBIOS name. Note: the "addresses" information is only available in request packets.	• `kerberos.pvno == 5` • `kerberos.realm contains ".org"` • `kerberos.SNameString == "krbtg"`



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

Cleartext Protocol Analysis: FTP

Investigating cleartext protocol traces sounds easy, but when the time comes to investigate a big network trace for incident analysis and response, the game changes. Proper analysis is more than following the stream and reading the cleartext data. For a security analyst, it is important to create statistics and key results from the investigation process. As mentioned earlier at the beginning of the Wireshark room series, the analyst should have the required network knowledge and tool skills to accomplish this. Let's simulate a cleartext protocol investigation with Wireshark!

FTP Analysis

File Transfer Protocol (FTP) is designed to transfer files with ease, so it focuses on simplicity rather than security. As a result of this, using this protocol in unsecured environments could create security issues like:

- MITM attacks
- Credential stealing and unauthorised access
- Phishing
- Malware planting
- Data exfiltration

FTP analysis in a nutshell:

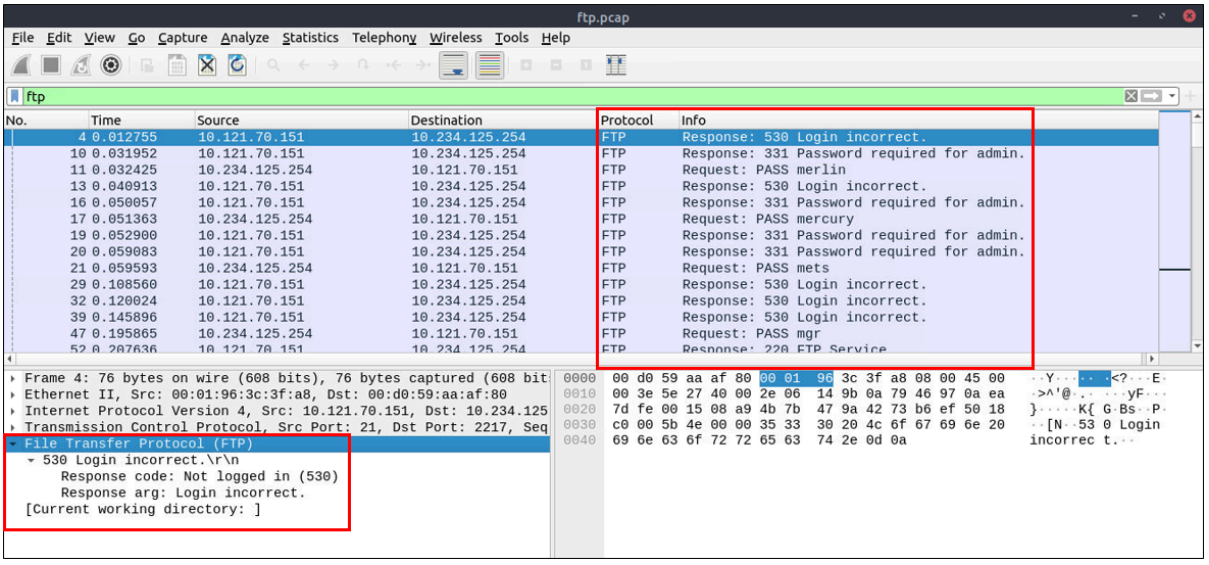
Notes	Wireshark Filter
Global search	• `ftp`
"FTP" options for grabbing the low-hanging fruits: • x1x series: Information request responses. • x2x series: Connection messages. • x3x series: Authentication messages. Note: "200" means command successful.	---

"x1x" series options for grabbing the low-hanging fruits: • 211: System status. • 212: Directory status. • 213: File status	• `ftp.response.code == 211`
"x2x" series options for grabbing the low-hanging fruits: • 220: Service ready. • 227: Entering passive mode. • 228: Long passive mode. • 229: Extended passive mode.	• `ftp.response.code == 227`
"x3x" series options for grabbing the low-hanging fruits: • 230: User login. • 231: User logout. • 331: Valid username. • 430: Invalid username or password • 530: No login, invalid password.	• `ftp.response.code == 230`
"FTP" commands for grabbing the low-hanging fruits: • USER: Username. • PASS: Password. • CWD: Current work directory. • LIST: List.	• `ftp.request.command == "USER"` • `ftp.request.command == "PASS"` • `ftp.request.arg == "password"`

Advanced usages examples for grabbing low-hanging fruits:

- Bruteforce signal: List failed login attempts.
- Bruteforce signal: List target username.
- Password spray signal: List targets for a static password.

- `ftp.response.code == 530`
- `(ftp.response.code == 530) and (ftp.response.arg contains "username")`
- `(ftp.request.command == "PASS" ) and (ftp.request.arg == "password")`



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

Cleartext Protocol Analysis: HTTP

HTTP Analysis

Hypertext Transfer Protocol (HTTP) is a cleartext-based, request-response and client-server protocol. It is the standard type of network activity to request/serve web pages, and by default, it is not blocked by any network perimeter. As a result of being unencrypted and the backbone of web traffic, HTTP is one of the must-to-know protocols in traffic analysis. Following attacks could be detected with the help of HTTP analysis:

- Phishing pages
- Web attacks
- Data exfiltration
- Command and control traffic (C2)

HTTP analysis in a nutshell:

Notes	Wireshark Filter
Global search Note: HTTP2 is a revision of the HTTP protocol for better performance and security. It supports binary data transfer and request&response multiplexing.	• `http` • `http2`
"HTTP Request Methods" for grabbing the low-hanging fruits: • GET • POST • Request: Listing all requests	• `http.request.method == "GET"` • `http.request.method == "POST"` • `http.request`

"HTTP Response Status Codes" for grabbing the low-hanging fruits: • 200 OK: Request successful. • 301 Moved Permanently: Resource is moved to a new URL/path (permanently). • 302 Moved Temporarily: Resource is moved to a new URL/path (temporarily). • 400 Bad Request: Server didn't understand the request. • 401 Unauthorised: URL needs authorisation (login, etc.). • 403 Forbidden: No access to the requested URL. • 404 Not Found: Server can't find the requested URL. • 405 Method Not Allowed: Used method is not suitable or blocked. • 408 Request Timeout: Request look longer than server wait time. • 500 Internal Server Error: Request not completed, unexpected error. • 503 Service Unavailable: Request not completed server or service is down.	• `http.response.code == 200` • `http.response.code == 401` • `http.response.code == 403` • `http.response.code == 404` • `http.response.code == 405` • `http.response.code == 503`
"HTTP Parameters" for grabbing the low-hanging fruits: • User agent: Browser and operating system identification to a web server application. • Request URI: Points the requested resource from the server. • Full *URI: Complete URI information. *URI: Uniform Resource Identifier.	• `http.user_agent contains "nmap"` • `http.request.uri contains "admin"` • `http.request.full_uri contains "admin"`

"HTTP Parameters" for grabbing the low-hanging fruits: • Server: Server service name. • Host: Hostname of the server • Connection: Connection status. • Line-based text data: Cleartext data provided by the server. • HTML Form URL Encoded: Web form information.	• `http.server contains "apache"` • `http.host contains "keyword"` • `http.host == "keyword"` • `http.connection == "Keep-Alive"` • `data-text-lines contains "keyword"`

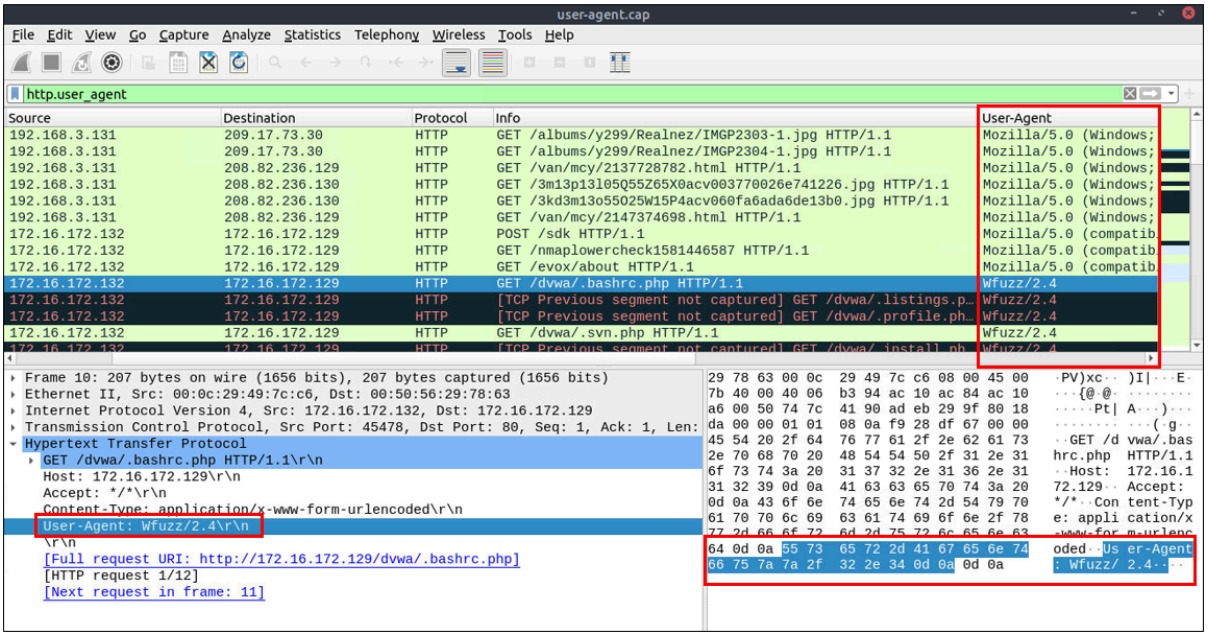
User Agent Analysis

As the adversaries use sophisticated technics to accomplish attacks, they try to leave traces similar to natural traffic through the known and trusted protocols. For a security analyst, it is important to spot the anomaly signs on the bits and pieces of the packets. The "user-agent" field is one of the great resources for spotting anomalies in HTTP traffic. In some cases, adversaries successfully modify the user-agent data, which could look super natural. A security analyst cannot rely only on the user-agent field to spot an anomaly. Never whitelist a user agent, even if it looks natural. User agent-based anomaly/threat detection/hunting is an additional data source to check and is useful when there is an obvious anomaly. If you are unsure about a value, you can conduct a web search to validate your findings with the default and normal user-agent info (example site).

User Agent analysis in a nutshell:

Notes	Wireshark Filter
Global search.	• `http.user_agent`

Research outcomes for grabbing the low-hanging fruits: • Different user agent information from the same host in a short time notice. • Non-standard and custom user agent info. • Subtle spelling differences. ("Mozilla" is not the same as "Mozlilla" or "Mozlila") • Audit tools info like Nmap, Nikto, Wfuzz and sqlmap in the user agent field. • Payload data in the user agent field.	• `(http.user_agent contains "sqlmap") or (http.user_agent contains "Nmap") or (http.user_agent contains "Wfuzz") or (http.user_agent contains "Nikto")`

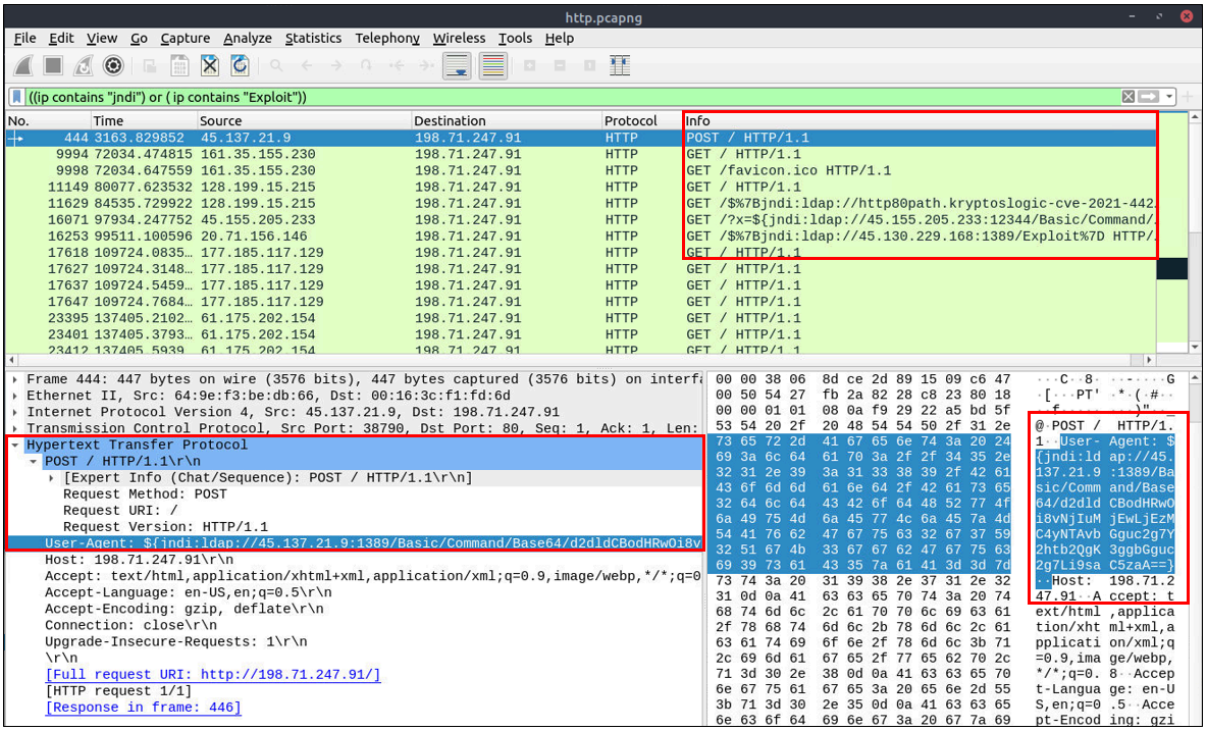


Log4j Analysis

A proper investigation starts with prior research on threats and anomalies going to be hunted. Let's review the knowns on the "Log4j" attack before launching Wireshark.

Log4j vulnerability analysis in a nutshell:

Notes	Wireshark Filters
Research outcomes for grabbing the low-hanging fruits: • The attack starts with a "POST" request • There are known cleartext patterns: "jndi:ldap" and "Exploit.class".	• `http.request.method == "POST"` • `(ip contains "jndi") or ( ip contains "Exploit")` • `(frame contains "jndi") or ( frame contains "Exploit")` • `(http.user_agent contains "$") or (http.user_agent contains "==")`



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

Encrypted Protocol Analysis: Decrypting HTTPS

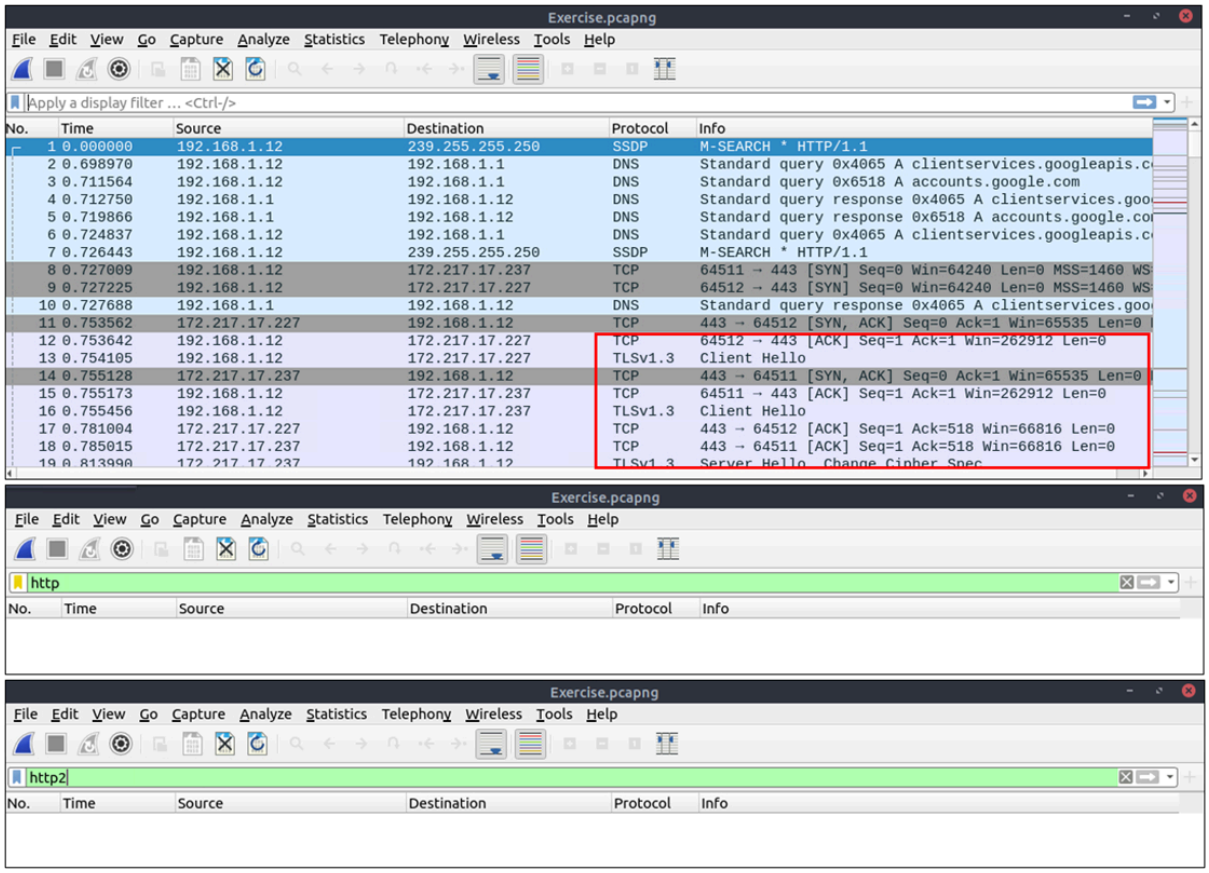
Decrypting HTTPS Traffic

When investigating web traffic, analysts often run across encrypted traffic. This is caused by using the Hypertext Transfer Protocol Secure (HTTPS) protocol for enhanced security against spoofing, sniffing and intercepting attacks. HTTPS uses TLS protocol to encrypt communications, so it is impossible to decrypt the traffic and view the transferred data without having the encryption/decryption key pairs. As this protocol provides a good level of security for transmitting sensitive data, attackers and malicious websites also use HTTPS. Therefore, a security analyst should know how to use key files to decrypt encrypted traffic and investigate the traffic activity.

The packets will appear in different colours as the HTTP traffic is encrypted. Also, protocol and info details (actual URL address and data returned from the server) will not be fully visible. The first image below shows the HTTP packets encrypted with the TLS protocol. The second and third images demonstrate filtering HTTP packets without using a key log file.

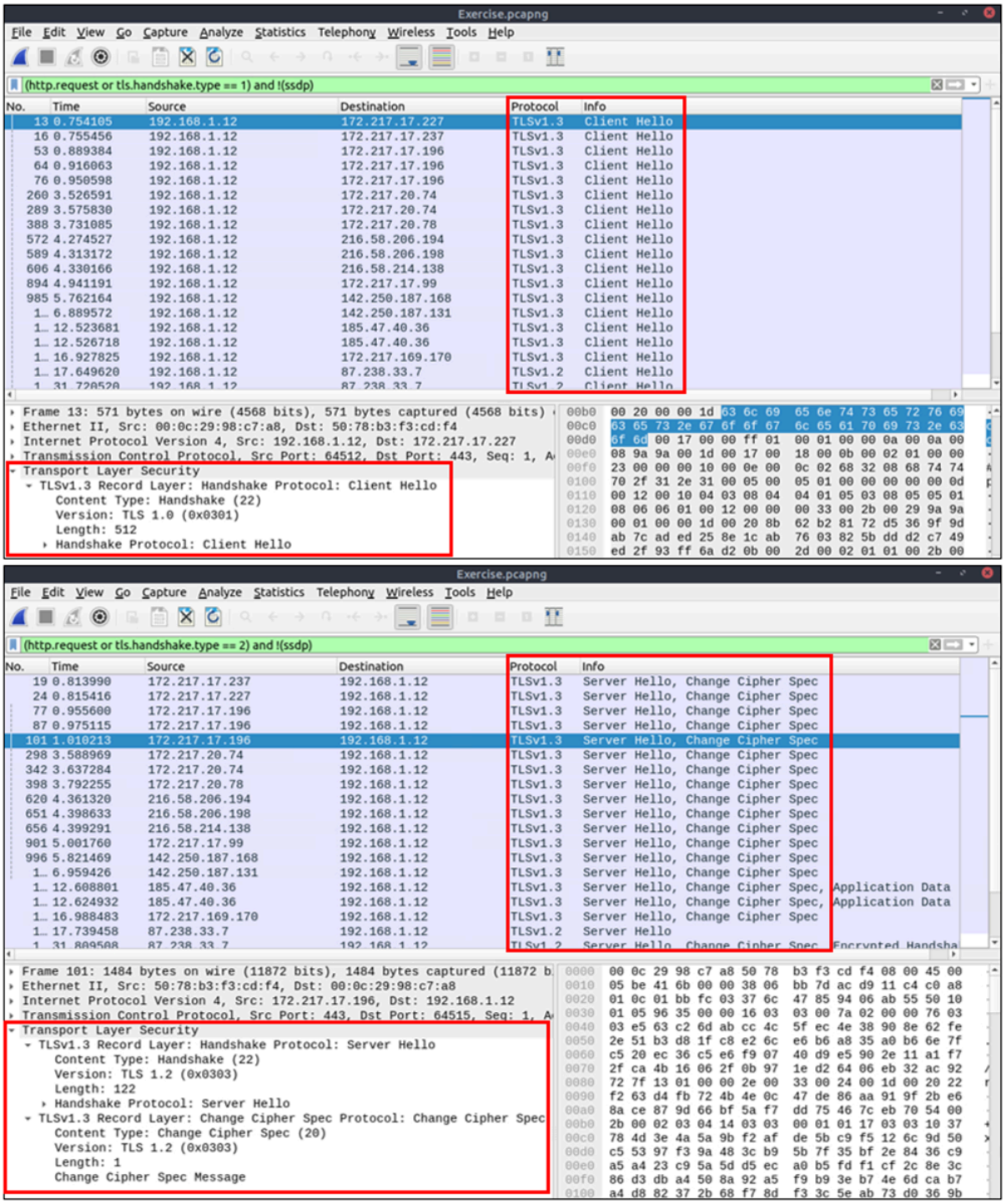
Additional information for HTTPS :

Notes	Wireshark Filter
"HTTPS Parameters" for grabbing the low-hanging fruits: • Request: Listing all requests • TLS: Global TLS search • TLS Client Request • TLS Server response • Local Simple Service Discovery Protocol (SSDP) Note: SSDP is a network protocol that provides advertisement and discovery of network services.	• `http.request` • `tls` • `tls.handshake.type == 1` • `tls.handshake.type == 2` • `ssdp`



Similar to the TCP three-way handshake process, the TLS protocol has its handshake process. The first two steps contain "Client Hello" and "Server Hello" messages. The given filters show the initial hello packets in a capture file. These filters are helpful to spot which IP addresses are involved in the TLS handshake.

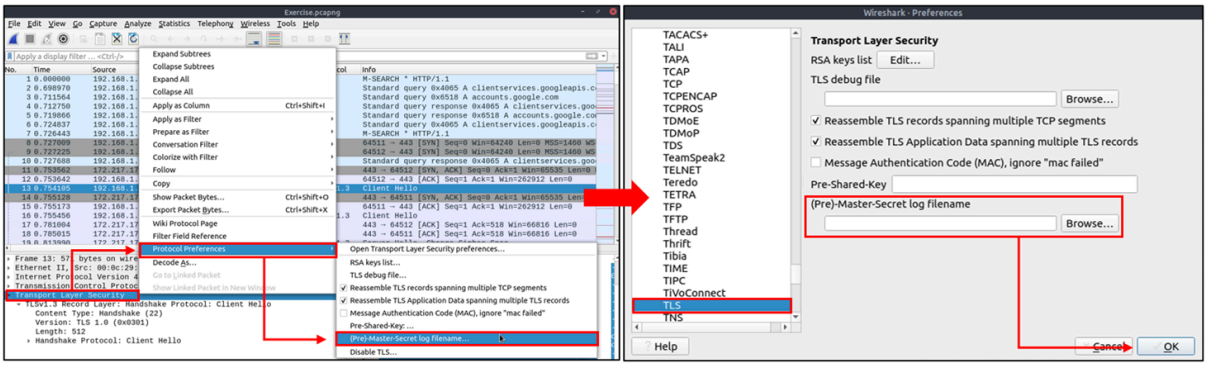
- Client Hello: `(http.request or tls.handshake.type == 1) and !(ssdp)`
- Server Hello: `(http.request or tls.handshake.type == 2) and !(ssdp)`



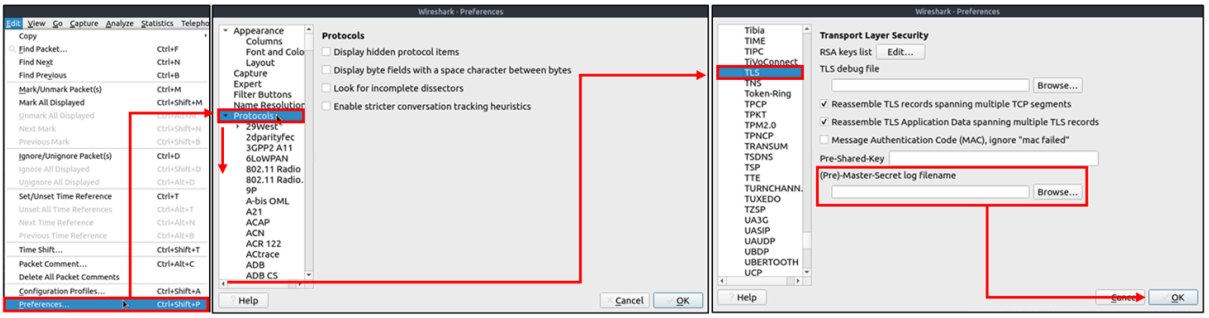
An encryption key log file is a text file that contains unique key pairs to decrypt the encrypted traffic session. These key pairs are automatically created (per session) when a connection is established with an SSL/TLS-enabled webpage. As these processes are all accomplished in the browser, you need to configure your system and use a suitable browser (Chrome and Firefox support this) to save these values as a key log file. To do this, you will need to set up an environment variable and create the SSLKEYLOGFILE, and the browser will dump the keys to this file as you browse the web. SSL/TLS key pairs are created per session at the connection time,

so it is important to dump the keys during the traffic capture. Otherwise, it is not possible to create/generate a suitable key log file to decrypt captured traffic. You can use the "right-click" menu or "Edit --> Preferences --> Protocols --> TLS" menu to add/remove key log files.

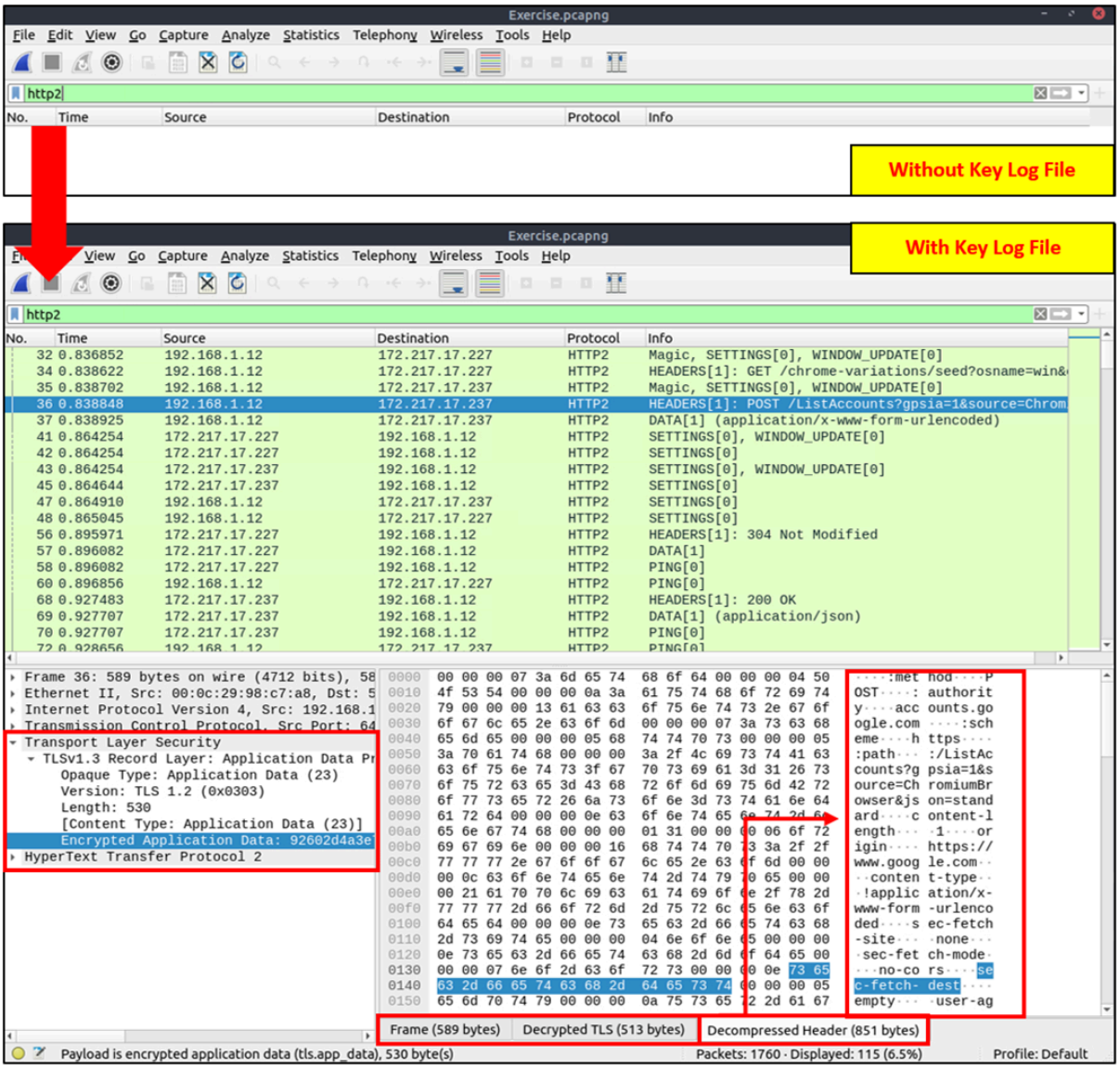
Adding key log files with the "right-click" menu:



Adding key log files with the "Edit --> Preferences --> Protocols --> TLS" menu:



Viewing the traffic with/without the key log files:



The above image shows that the traffic details are visible after using the key log file. Note that the packet details and bytes pane provides the data in different formats for investigation. Decompressed header info and HTTP2 packet details are available after decrypting the traffic. Depending on the packet details, you can also have the following data formats:

- Frame
- Decrypted TLS
- Decompressed Header
- Reassembled TCP
- Reassembled SSL

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

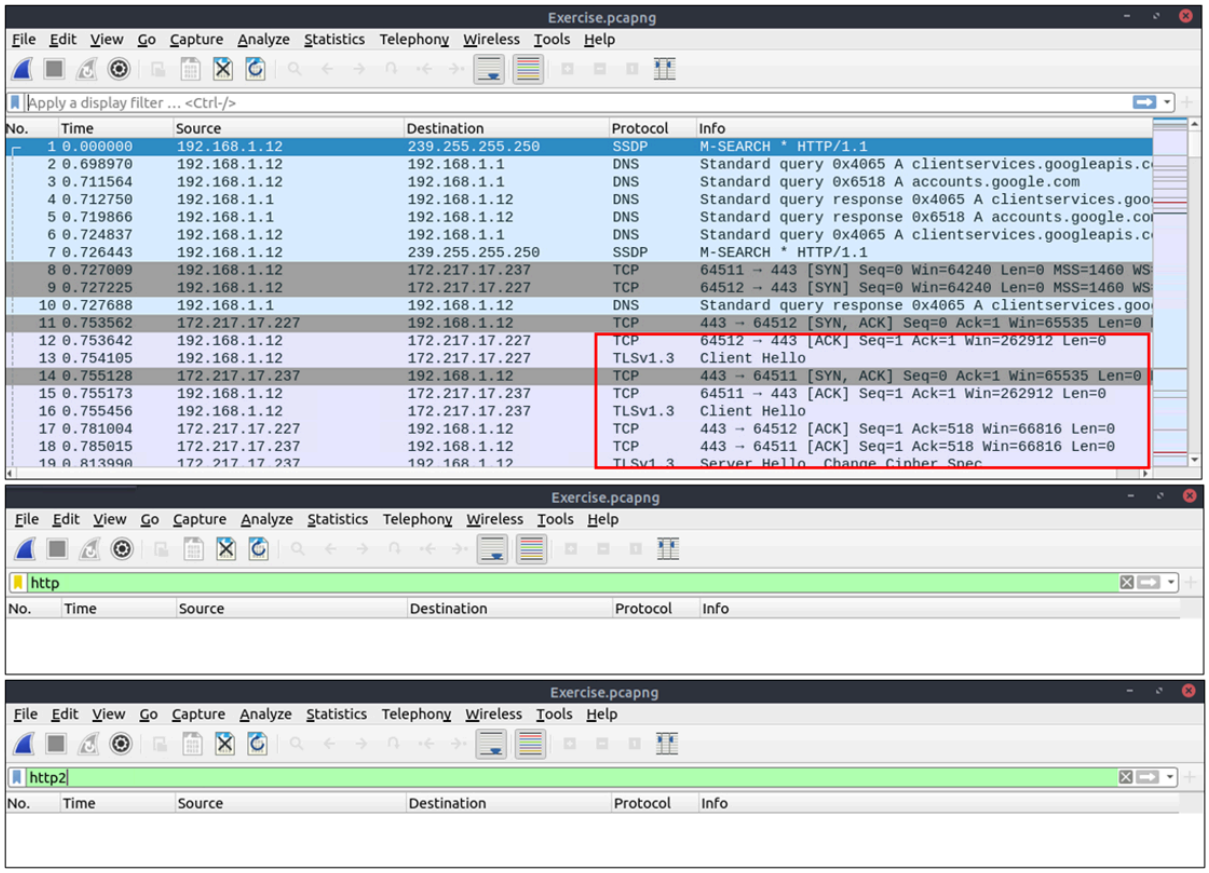
Decrypting HTTPS Traffic

When investigating web traffic, analysts often run across encrypted traffic. This is caused by using the Hypertext Transfer Protocol Secure (HTTPS) protocol for enhanced security against spoofing, sniffing and intercepting attacks. HTTPS uses TLS protocol to encrypt communications, so it is impossible to decrypt the traffic and view the transferred data without having the encryption/decryption key pairs. As this protocol provides a good level of security for transmitting sensitive data, attackers and malicious websites also use HTTPS. Therefore, a security analyst should know how to use key files to decrypt encrypted traffic and investigate the traffic activity.

The packets will appear in different colours as the HTTP traffic is encrypted. Also, protocol and info details (actual URL address and data returned from the server) will not be fully visible. The first image below shows the HTTP packets encrypted with the TLS protocol. The second and third images demonstrate filtering HTTP packets without using a key log file.

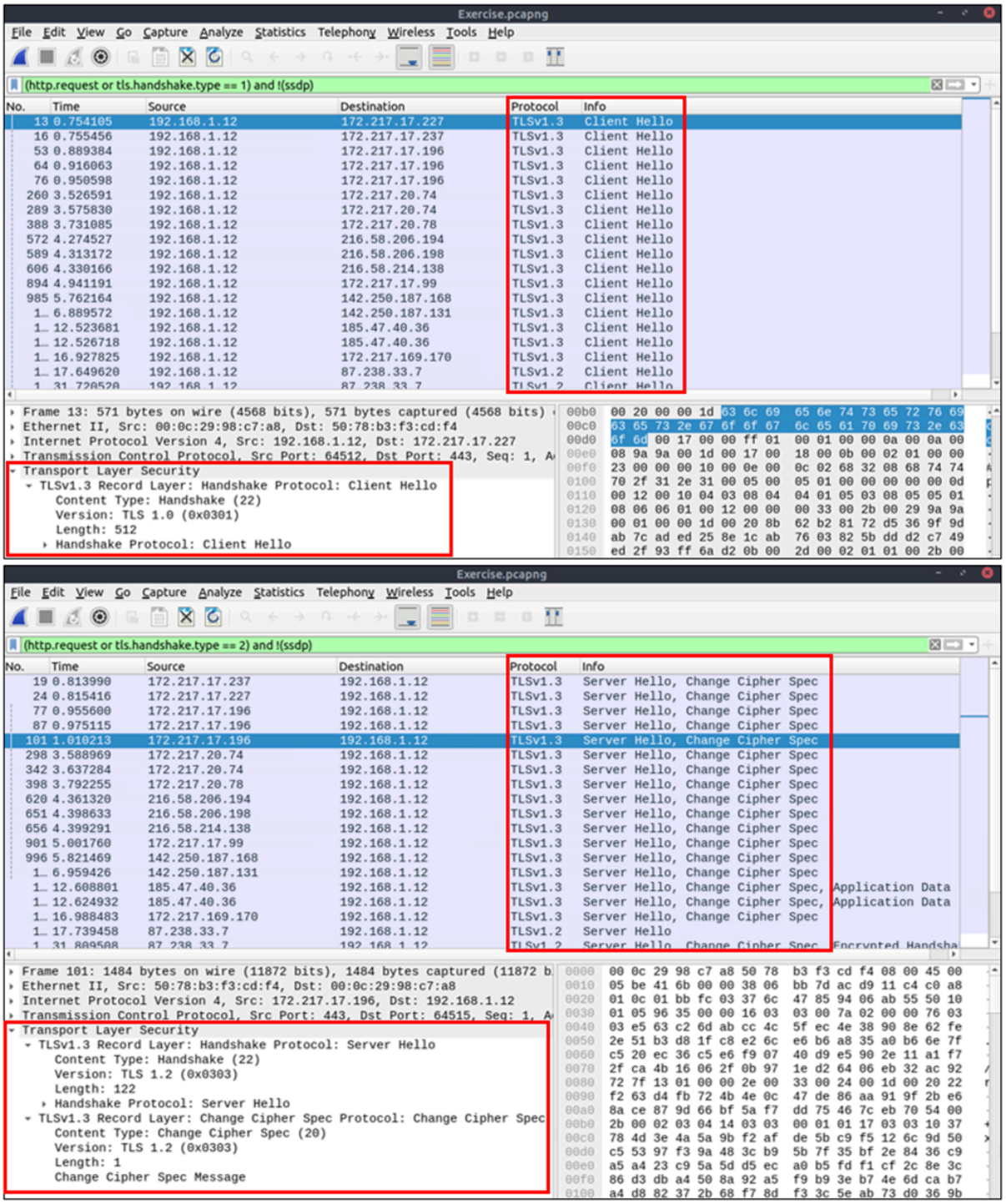
Additional information for HTTPS :

Notes	Wireshark Filter
"HTTPS Parameters" for grabbing the low-hanging fruits: • Request: Listing all requests • TLS: Global TLS search • TLS Client Request • TLS Server response • Local Simple Service Discovery Protocol (SSDP) Note: SSDP is a network protocol that provides advertisement and discovery of network services.	• `http.request` • `tls` • `tls.handshake.type == 1` • `tls.handshake.type == 2` • `ssdp`



Similar to the TCP three-way handshake process, the TLS protocol has its handshake process. The first two steps contain "Client Hello" and "Server Hello" messages. The given filters show the initial hello packets in a capture file. These filters are helpful to spot which IP addresses are involved in the TLS handshake.

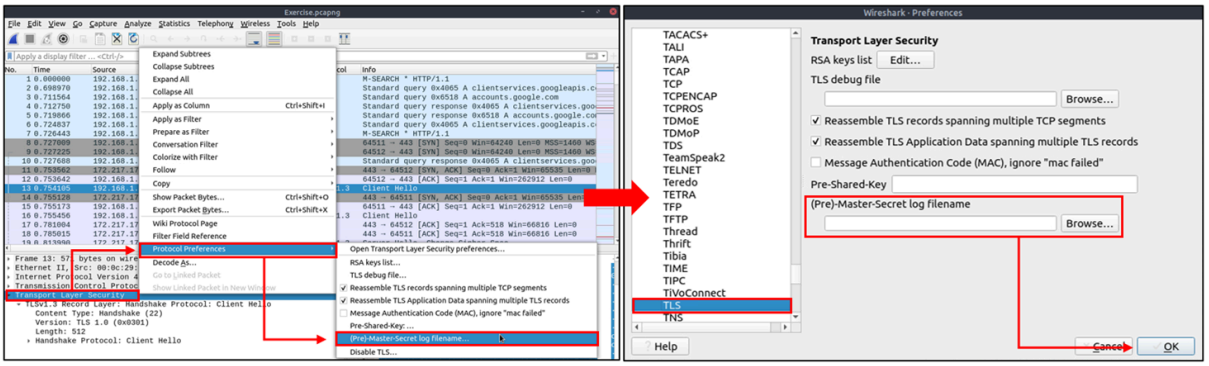
- Client Hello: `(http.request or tls.handshake.type == 1) and !(ssdp)`
- Server Hello: `(http.request or tls.handshake.type == 2) and !(ssdp)`



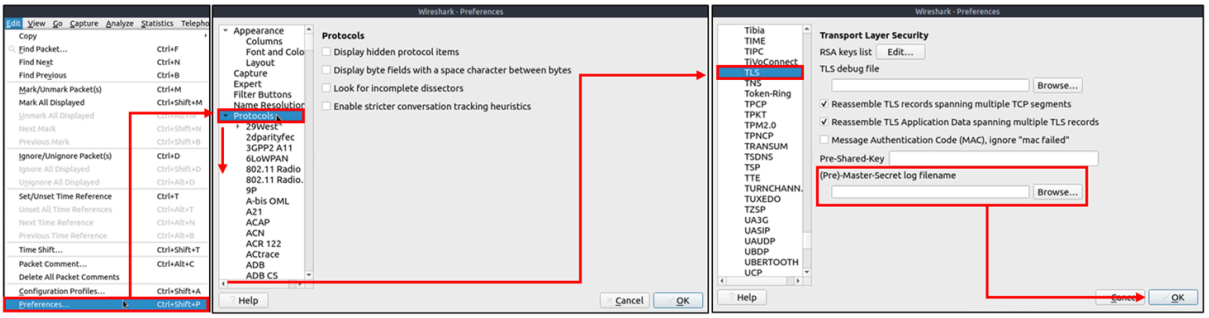
An encryption key log file is a text file that contains unique key pairs to decrypt the encrypted traffic session. These key pairs are automatically created (per session) when a connection is established with an SSL/TLS-enabled webpage. As these processes are all accomplished in the browser, you need to configure your system and use a suitable browser (Chrome and Firefox support this) to save these values as a key log file. To do this, you will need to set up an environment variable and create the SSLKEYLOGFILE, and the browser will dump the keys to this file as you browse the web. SSL/TLS key pairs are created per session at the connection time,

so it is important to dump the keys during the traffic capture. Otherwise, it is not possible to create/generate a suitable key log file to decrypt captured traffic. You can use the "right-click" menu or "Edit --> Preferences --> Protocols --> TLS" menu to add/remove key log files.

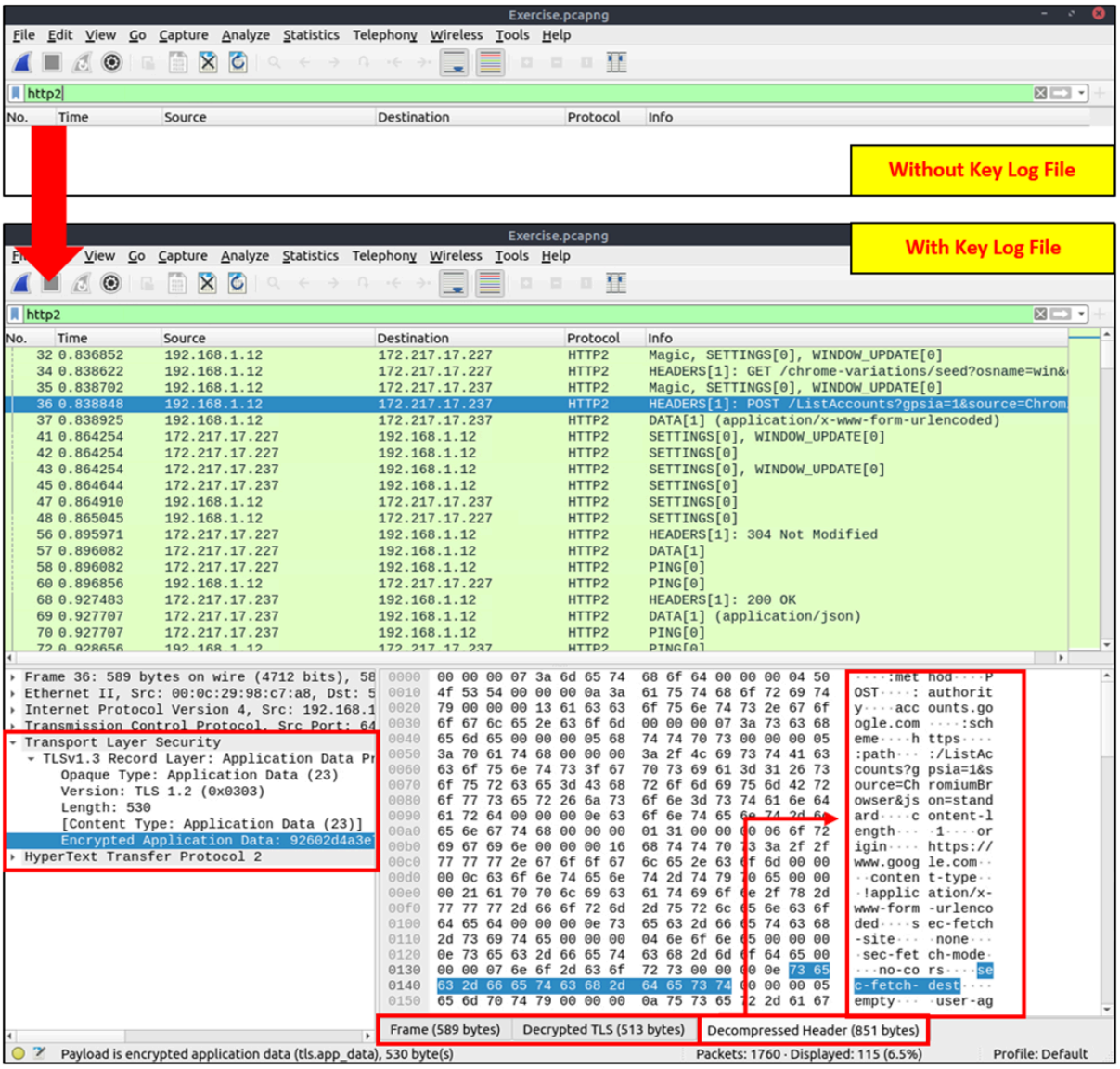
Adding key log files with the "right-click" menu:



Adding key log files with the "Edit --> Preferences --> Protocols --> TLS" menu:



Viewing the traffic with/without the key log files:



The above image shows that the traffic details are visible after using the key log file. Note that the packet details and bytes pane provides the data in different formats for investigation. Decompressed header info and HTTP2 packet details are available after decrypting the traffic. Depending on the packet details, you can also have the following data formats:

- Frame
- Decrypted TLS
- Decompressed Header
- Reassembled TCP
- Reassembled SSL

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

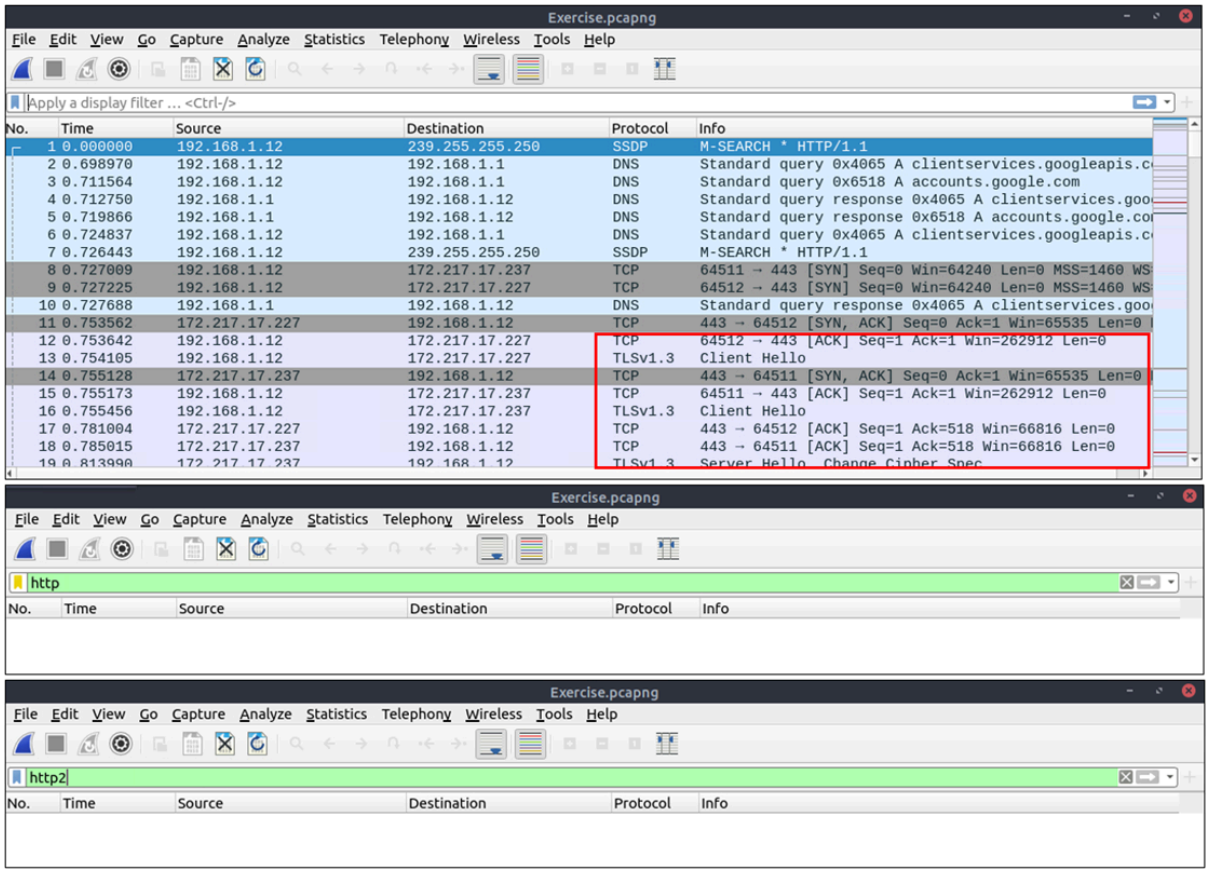
Decrypting HTTPS Traffic

When investigating web traffic, analysts often run across encrypted traffic. This is caused by using the Hypertext Transfer Protocol Secure (HTTPS) protocol for enhanced security against spoofing, sniffing and intercepting attacks. HTTPS uses TLS protocol to encrypt communications, so it is impossible to decrypt the traffic and view the transferred data without having the encryption/decryption key pairs. As this protocol provides a good level of security for transmitting sensitive data, attackers and malicious websites also use HTTPS. Therefore, a security analyst should know how to use key files to decrypt encrypted traffic and investigate the traffic activity.

The packets will appear in different colours as the HTTP traffic is encrypted. Also, protocol and info details (actual URL address and data returned from the server) will not be fully visible. The first image below shows the HTTP packets encrypted with the TLS protocol. The second and third images demonstrate filtering HTTP packets without using a key log file.

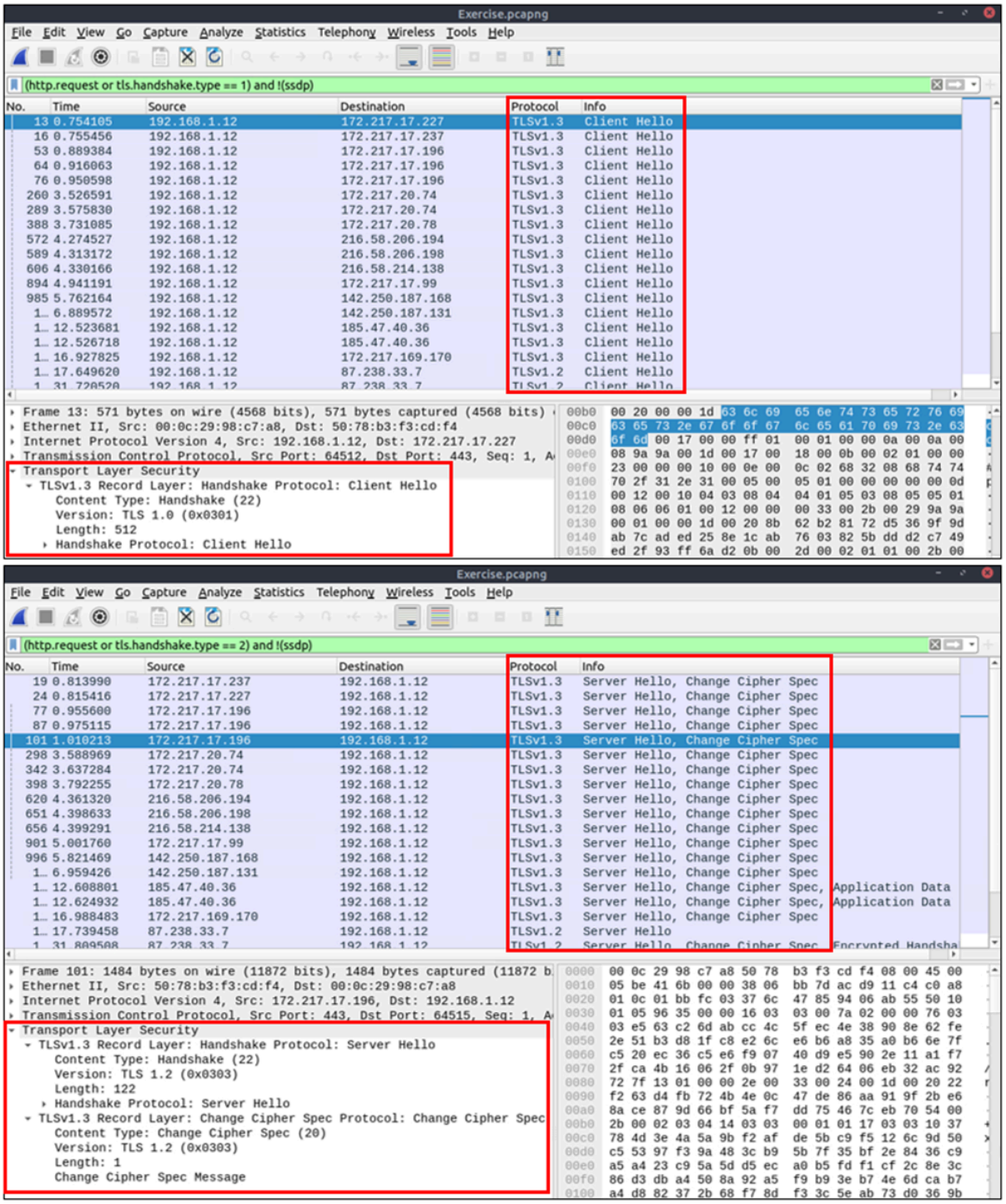
Additional information for HTTPS :

Notes	Wireshark Filter
"HTTPS Parameters" for grabbing the low-hanging fruits: • Request: Listing all requests • TLS: Global TLS search • TLS Client Request • TLS Server response • Local Simple Service Discovery Protocol (SSDP) Note: SSDP is a network protocol that provides advertisement and discovery of network services.	• `http.request` • `tls` • `tls.handshake.type == 1` • `tls.handshake.type == 2` • `ssdp`



Similar to the TCP three-way handshake process, the TLS protocol has its handshake process. The first two steps contain "Client Hello" and "Server Hello" messages. The given filters show the initial hello packets in a capture file. These filters are helpful to spot which IP addresses are involved in the TLS handshake.

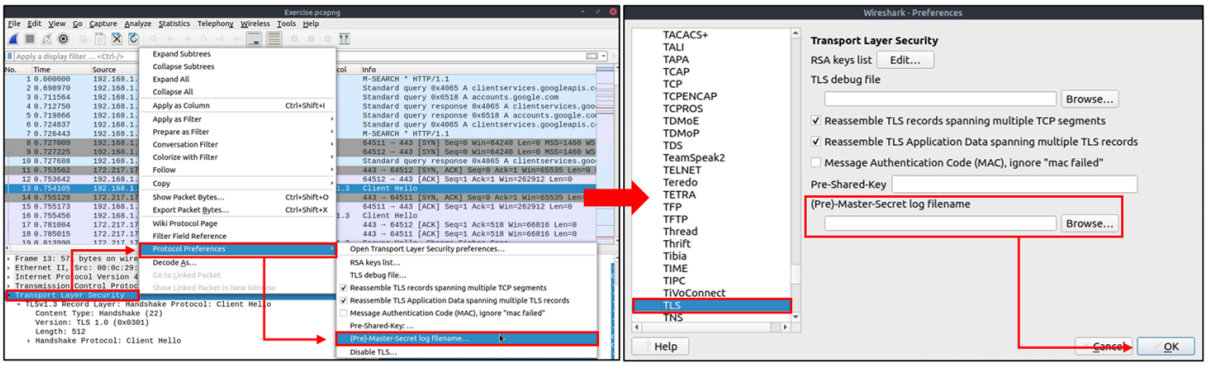
- Client Hello: `(http.request or tls.handshake.type == 1) and !(ssdp)`
- Server Hello: `(http.request or tls.handshake.type == 2) and !(ssdp)`



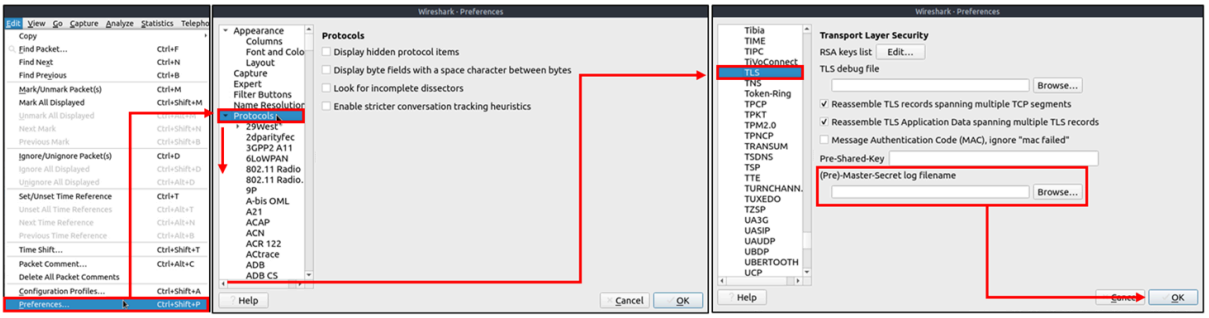
An encryption key log file is a text file that contains unique key pairs to decrypt the encrypted traffic session. These key pairs are automatically created (per session) when a connection is established with an SSL/TLS-enabled webpage. As these processes are all accomplished in the browser, you need to configure your system and use a suitable browser (Chrome and Firefox support this) to save these values as a key log file. To do this, you will need to set up an environment variable and create the SSLKEYLOGFILE, and the browser will dump the keys to this file as you browse the web. SSL/TLS key pairs are created per session at the connection time,

so it is important to dump the keys during the traffic capture. Otherwise, it is not possible to create/generate a suitable key log file to decrypt captured traffic. You can use the "right-click" menu or "Edit --> Preferences --> Protocols --> TLS" menu to add/remove key log files.

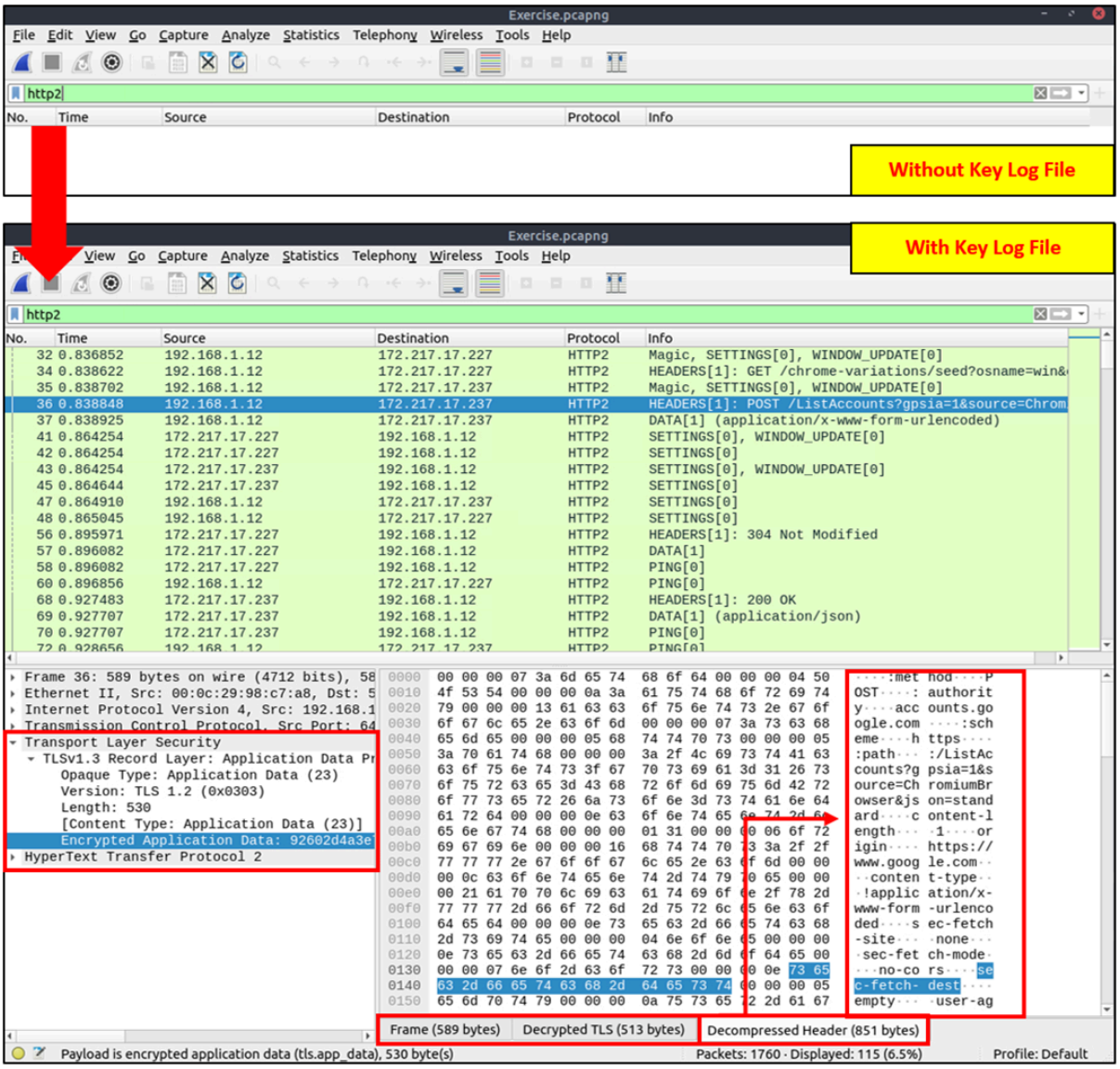
Adding key log files with the "right-click" menu:



Adding key log files with the "Edit --> Preferences --> Protocols --> TLS" menu:



Viewing the traffic with/without the key log files:



The above image shows that the traffic details are visible after using the key log file. Note that the packet details and bytes pane provides the data in different formats for investigation. Decompressed header info and HTTP2 packet details are available after decrypting the traffic. Depending on the packet details, you can also have the following data formats:

- Frame
- Decrypted TLS
- Decompressed Header
- Reassembled TCP
- Reassembled SSL

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