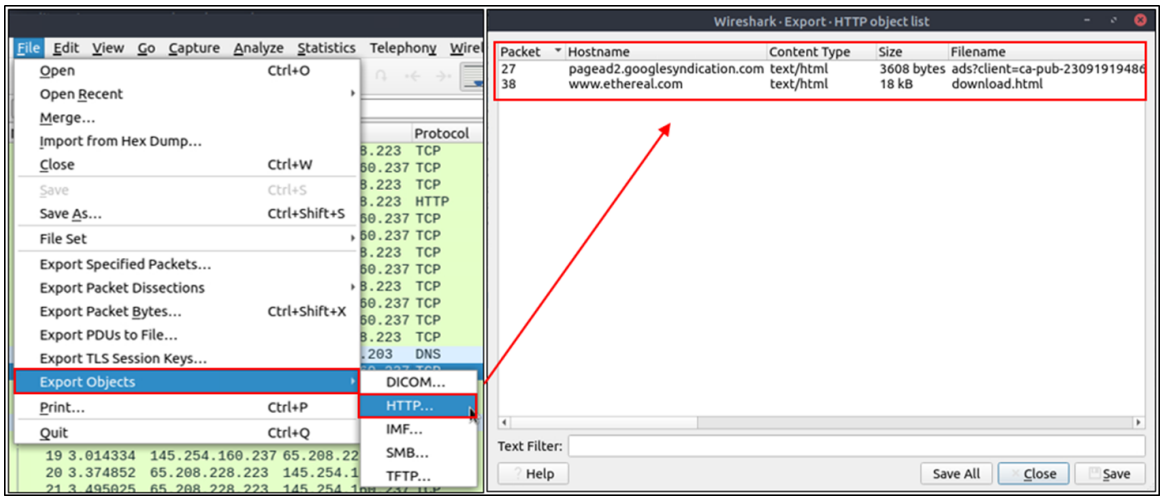
WireShark

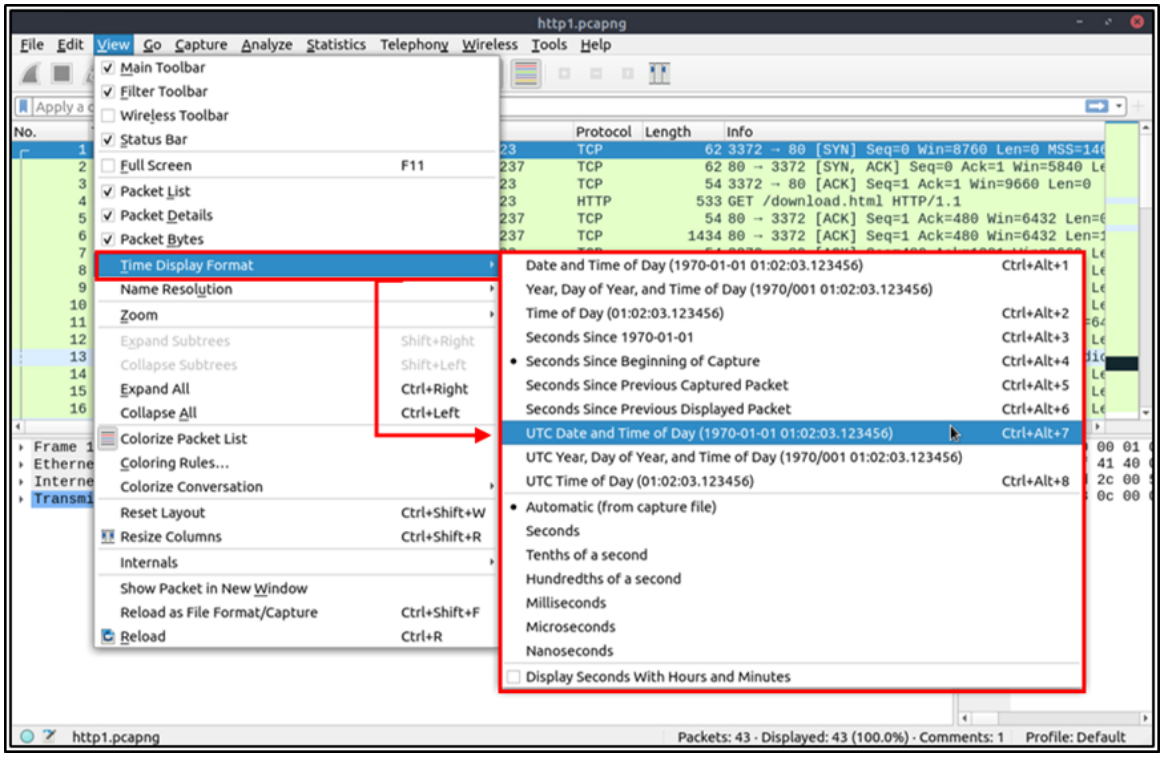
▼ Export Objects (Files)

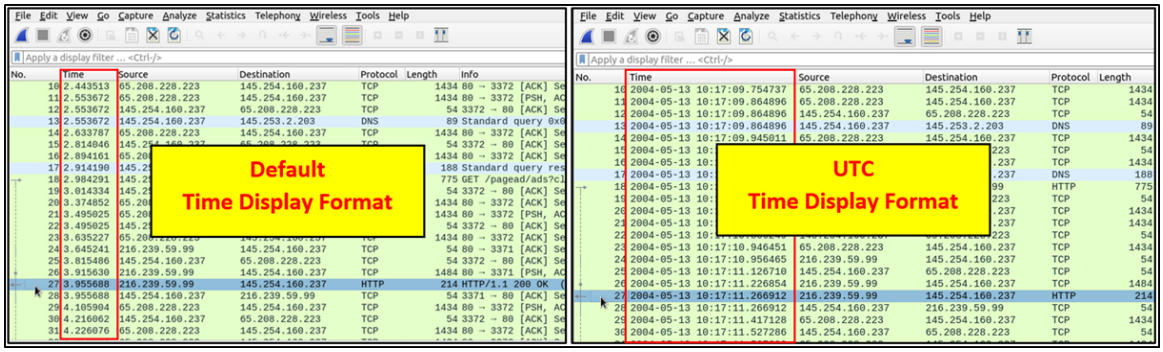
Wireshark can extract files transferred through the wire. For a security analyst, it is vital to discover shared files and save them for further investigation. Exporting objects are available only for selected protocol's streams (DICOM, HTTP, IMF, SMB and TFTP).



▼ Time Display Format

Wireshark lists the packets as they are captured, so investigating the default flow is not always the best option. By default, Wireshark shows the time in "Seconds Since Beginning of Capture", the common usage is using the UTC Time Display Format for a better view. You can use the "View --> Time Display Format" menu to change the time display format.





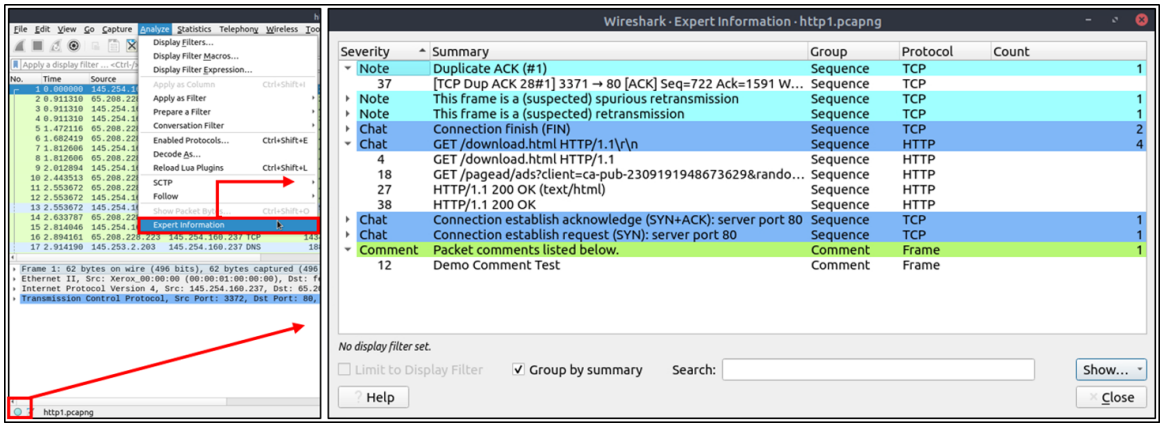
▼ Expert Info

Wireshark also detects specific states of protocols to help analysts easily spot possible anomalies and problems. Note that these are only suggestions, and there is always a chance of having false positives/negatives. Expert info can provide a group of categories in three different severities. Details are shown in the table below.

Severity	Colour	Info
Chat	**Blue**	Information on usual workflow.
Note	**Cyan**	Notable events like application error codes.
Warn	**Yellow**	Warnings like unusual error codes or problem statements.
Error	**Red**	Problems like malformed packets.

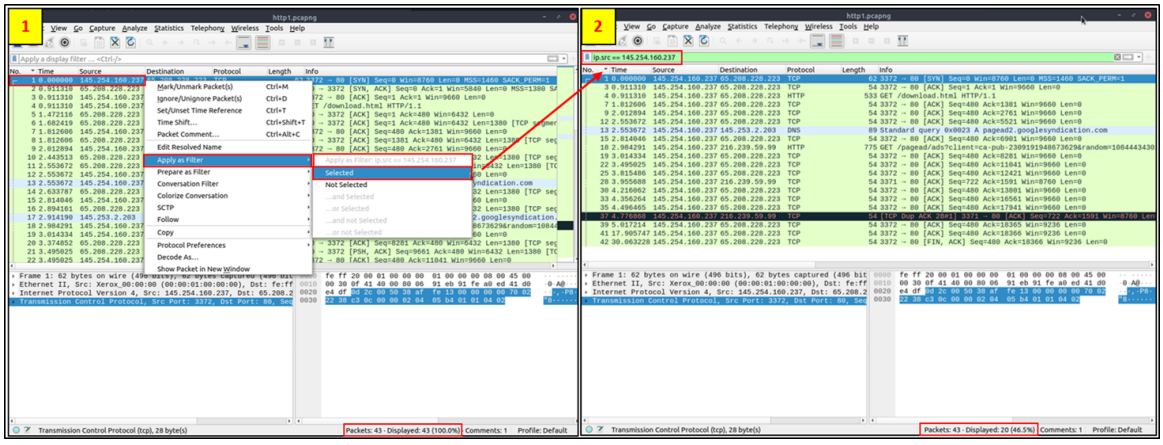
Frequently encountered information groups are listed in the table below. You can refer to Wireshark's official documentation for more information on the expert information entries.

Group	Info	Group	Info
Checksum	Checksum errors.	**Deprecated**	Deprecated protocol usage.
Comment	Packet comment detection.	**Malformed**	Malformed packet detection.



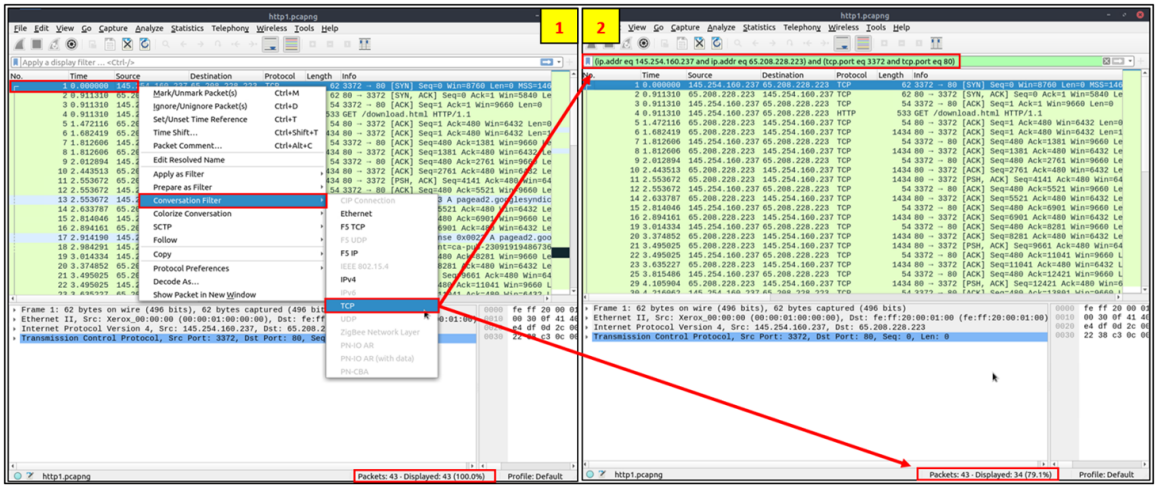
▼ Apply as Filter

This is the most basic way of filtering traffic. While investigating a capture file, you can click on the field you want to filter and use the "right-click menu" or **"Analyse --> Apply as Filter"** menu to filter the specific value. Once you apply the filter, Wireshark will generate the required filter query, apply it, show the packets according to your choice, and hide the unselected packets from the packet list pane. Note that the number of total and displayed packets are always shown on the status bar.



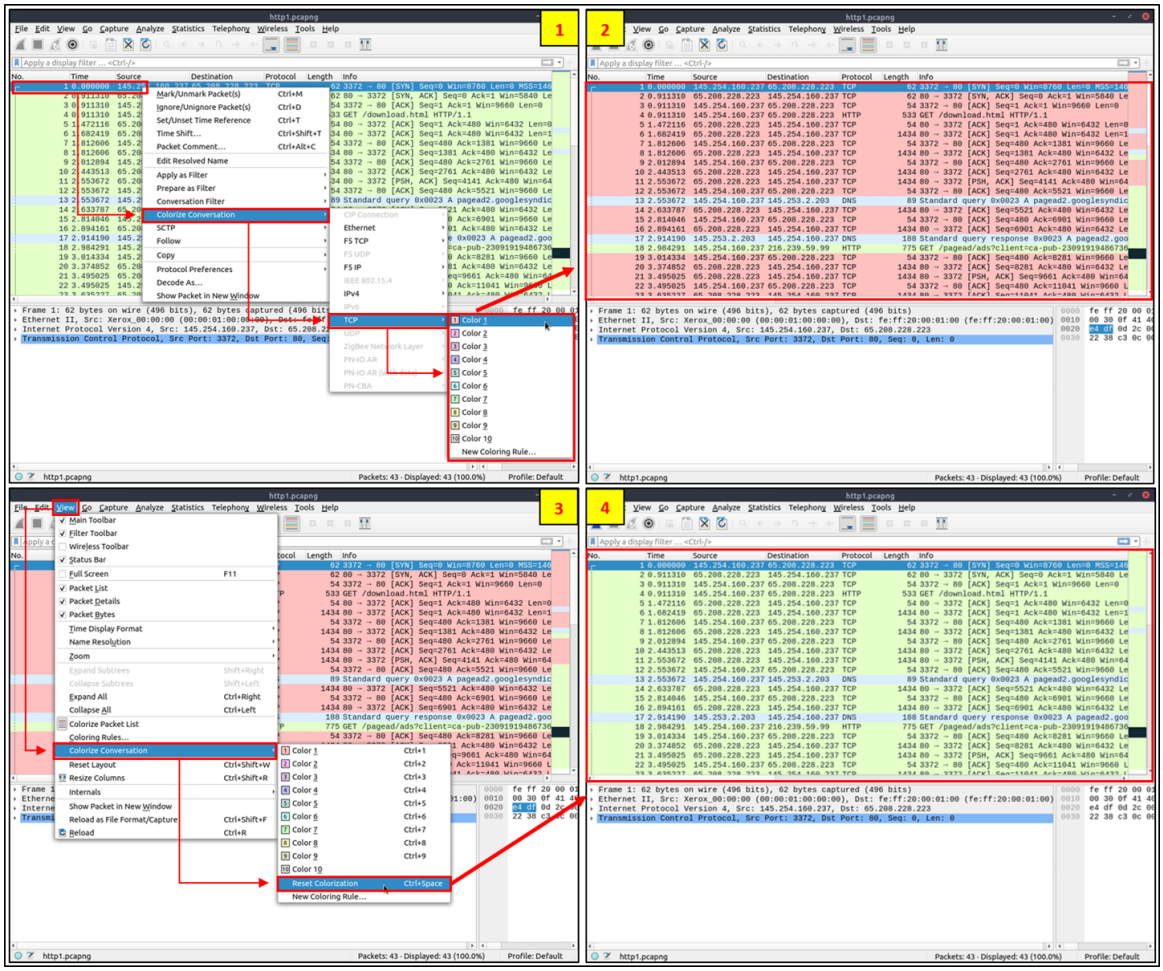
▼ Conversation filter

- "Conversation Filter" option helps you view only the related packets and hide the rest of the packets easily. You can use the"right-click menu" or "**Analyse -> Conversation Filter**" menu to filter conversations.



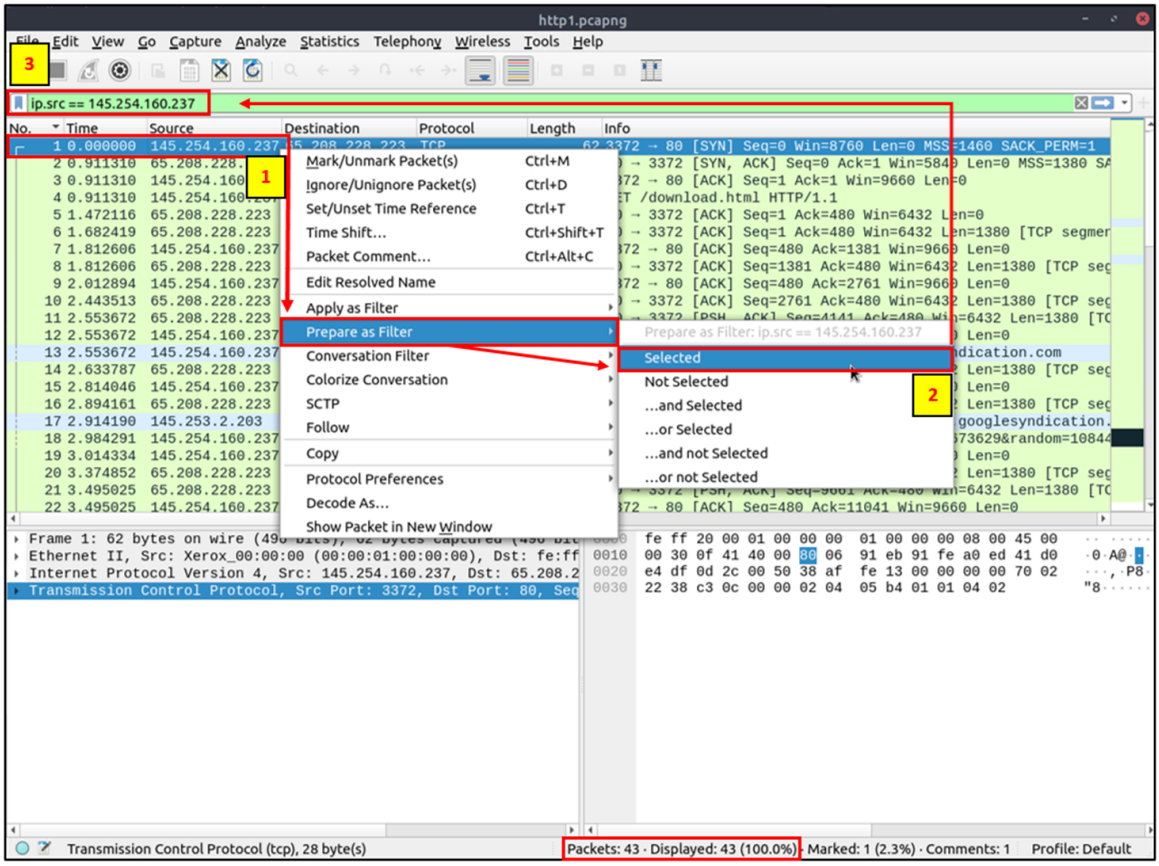
▼ Colourise Conversation

- This option is similar to the "Conversation Filter" with one difference. It highlights the linked packets without applying a display filter and decreasing the number of viewed packets. This option works with the "Colouring Rules" option ad changes the packet colours without considering the previously applied colour rule. You can use the "right-click menu" or **"View --> Colourise Conversation"** menu to colourise a linked packet in a single click. Note that you can use the "View --> Colourise Conversation --> Reset Colourisation" menu to undo this operation.



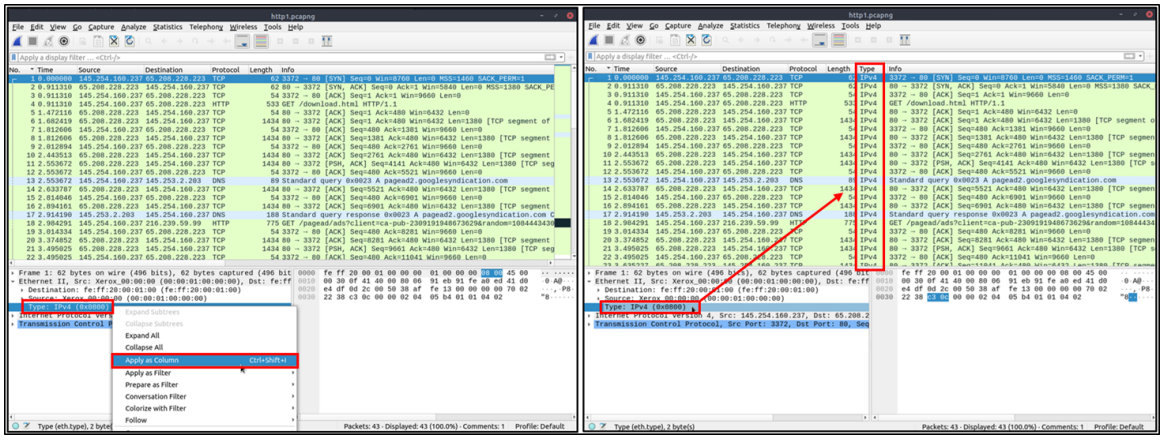
▼ Prepare as Filter

Similar to "Apply as Filter", this option helps analysts create display filters using the "right-click" menu. However, unlike the previous one, **this model doesn't apply the filters after the choice.** It adds the required query **to the pane and waits for the execution command** (enter) or another chosen filtering option by using the **".. and/or.."** from the "right-click menu".



▼ Apply as Column

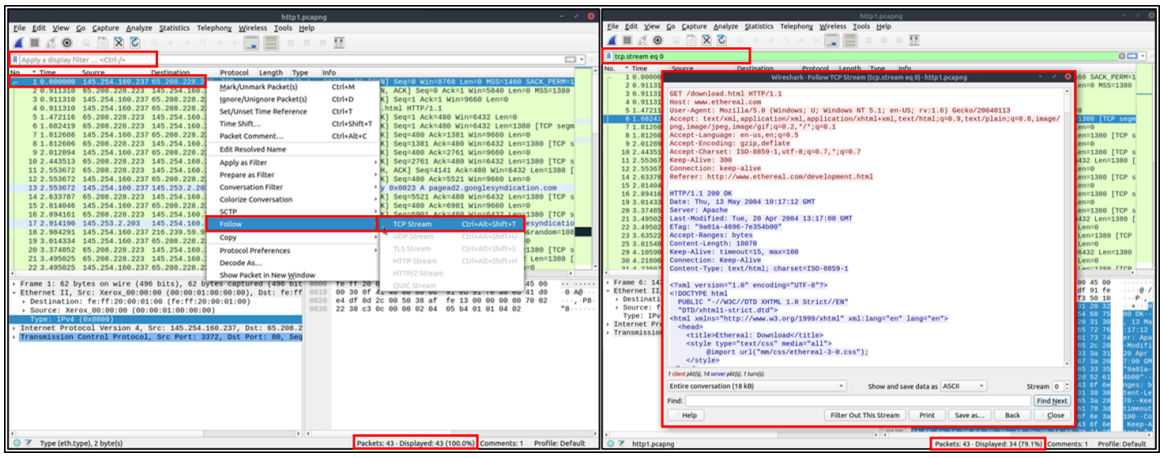
By default, the packet list pane provides basic information about each packet. You can use the "right-click menu" or "Analyse **-->** Apply as Column" menu to add columns to the packet list pane. Once you click on a value and apply it as a column, it will be visible on the packet list pane. This function helps analysts examine the appearance of a specific value/field across the available packets in the capture file. You can enable/disable the columns shown in the packet list pane by clicking on the top of the packet list pane.



Follow Stream

Wireshark displays everything in packet portion size. However, it is possible to reconstruct the streams and view the raw traffic as it is presented at the application level. Following the protocol, streams help analysts recreate the application-level data and understand the event of interest. It is also possible to view the unencrypted protocol data like usernames, passwords and other transferred data.

You can use the"right-click menu" or **"Analyse --> Follow TCP/UDP/HTTP Stream"** menu to follow traffic streams. Streams are shown in a separate dialogue box; packets originating from the server are highlighted with blue, and those originating from the client are highlighted with red.



Once you follow a stream, Wireshark automatically creates and applies the required filter to view the specific stream. Remember, once a filter is applied, the number of the viewed packets will change. You will need to use the "**X button**" located on the right upper side of the display filter bar to remove the display filter and view all available packets in the capture file.

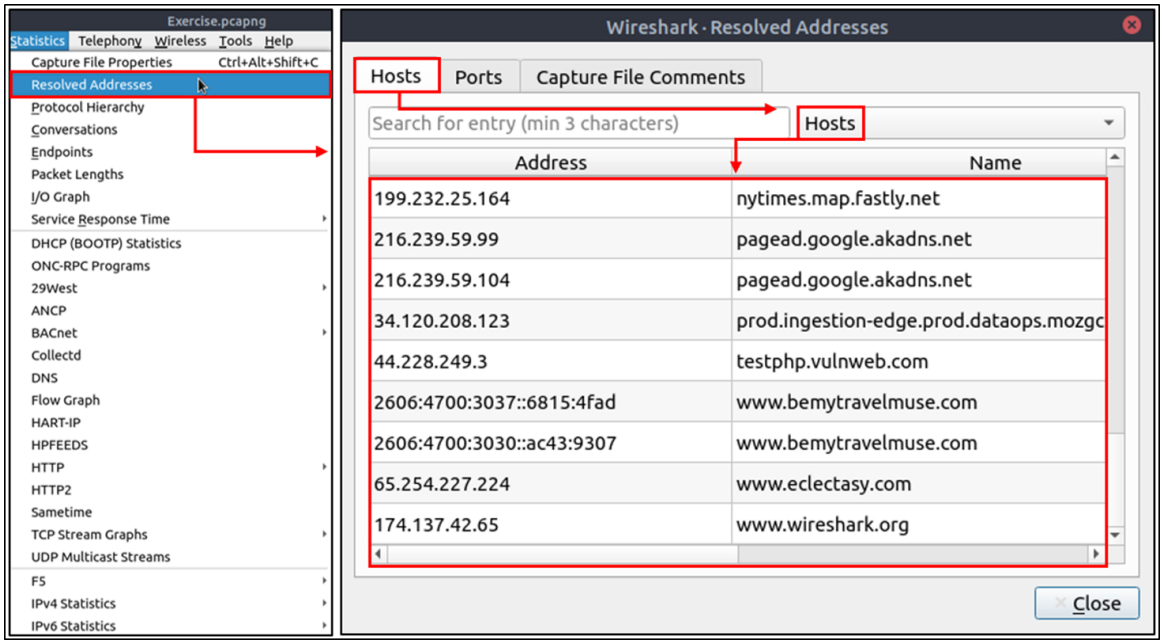
▼ Statistics

- This menu provides multiple statistics options ready to investigate to help users see the big picture in terms of the scope of the traffic, available protocols, endpoints and conversations, and some protocol-specific details like DHCP, DNS and HTTP/2. For a security analyst, it is crucial to know how to utilise the statical information. This section provides a quick summary of the processed pcap, which will help analysts create a hypothesis for an investigation. You can use the **"Statistics"** menu to view all available

options. Now start the given VM, open the Wireshark, load the "Exercise.pcapng" file and go through the walkthrough.

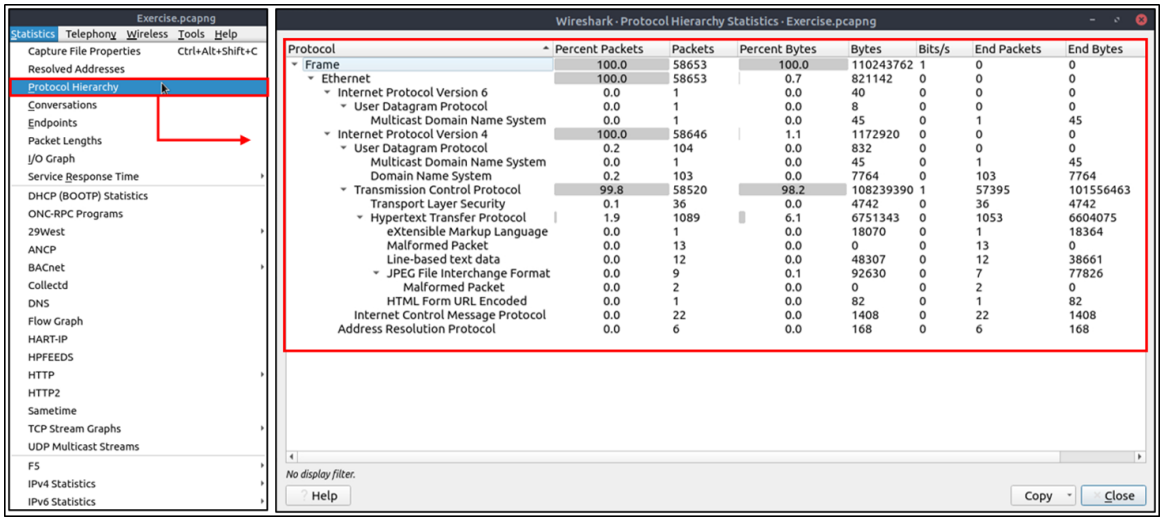
Resolved Addresses

This option helps analysts identify IP addresses and DNS names available in the capture file by providing the list of the resolved addresses and their hostnames. Note that the hostname information is taken from DNS answers in the capture file. Analysts can quickly identify the accessed resources by using this menu. Thus they can spot accessed resources and evaluate them according to the event of interest. You can use the **"Statistics --> Resolved Addresses"** menu to view all resolved addresses by Wireshark.



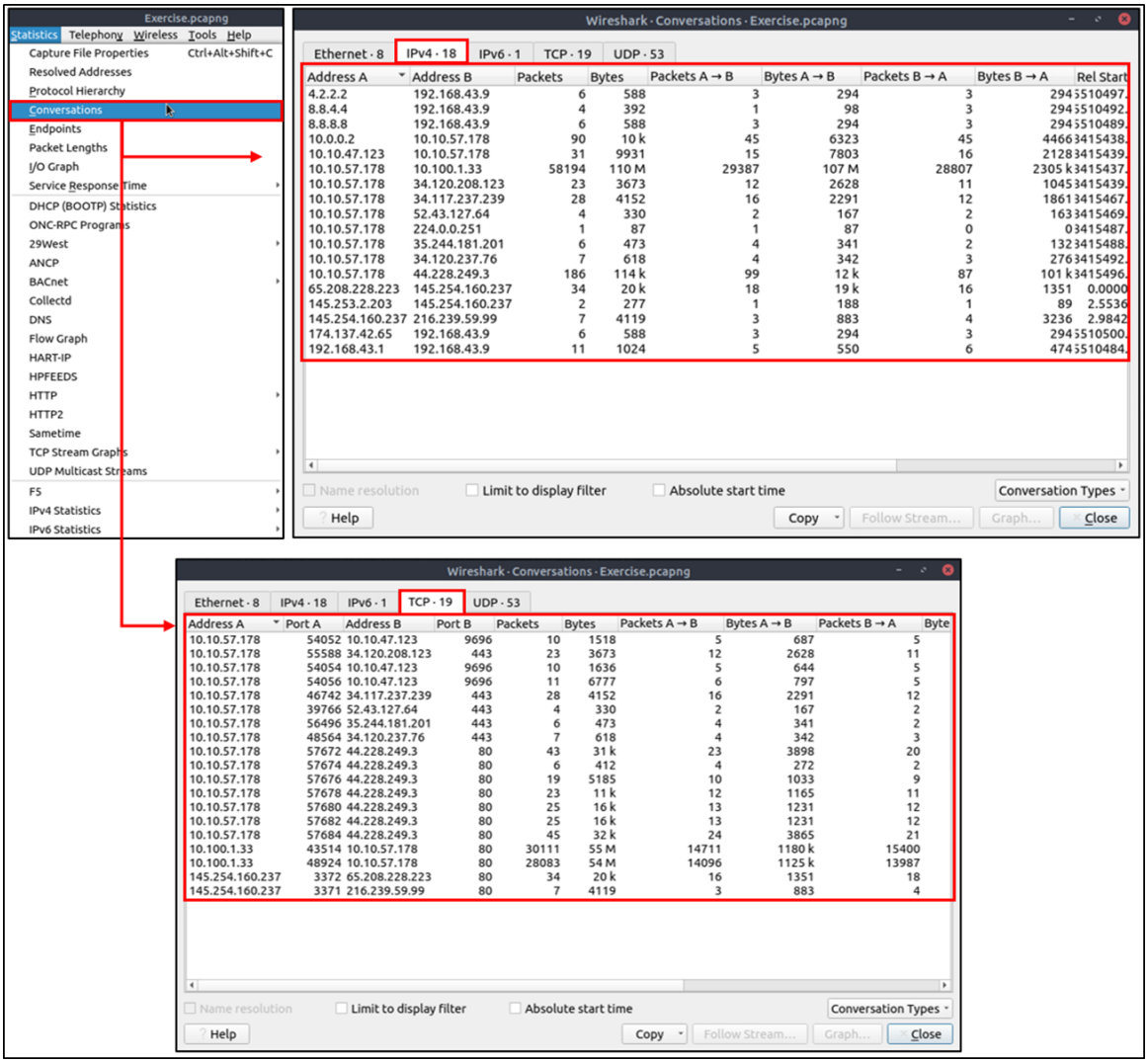
Protocol Hierarchy

This option breaks down all available protocols from the capture file and helps analysts view the protocols in a tree view based on packet counters and percentages. Thus analysts can view the overall usage of the ports and services and focus on the event of interest. The golden rule mentioned in the previous room is valid in this section; you can right-click and filter the event of interest. You can use the **"Statistics --> Protocol Hierarchy"** menu to view this info.



Conversations

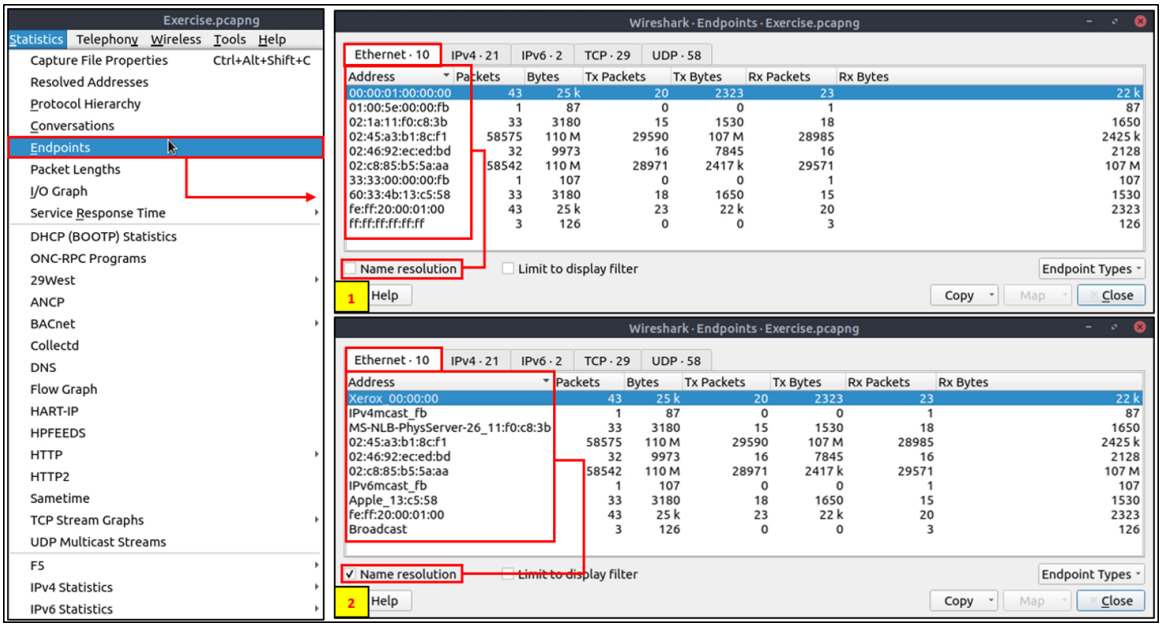
Conversation represents traffic between two specific endpoints. This option provides the list of the conversations in five base formats; ethernet, IPv4, IPv6, TCP and UDP. Thus analysts can identify all conversations and contact endpoints for the event of interest. You can use the "Statistic --> Conversations" menu to view this info.



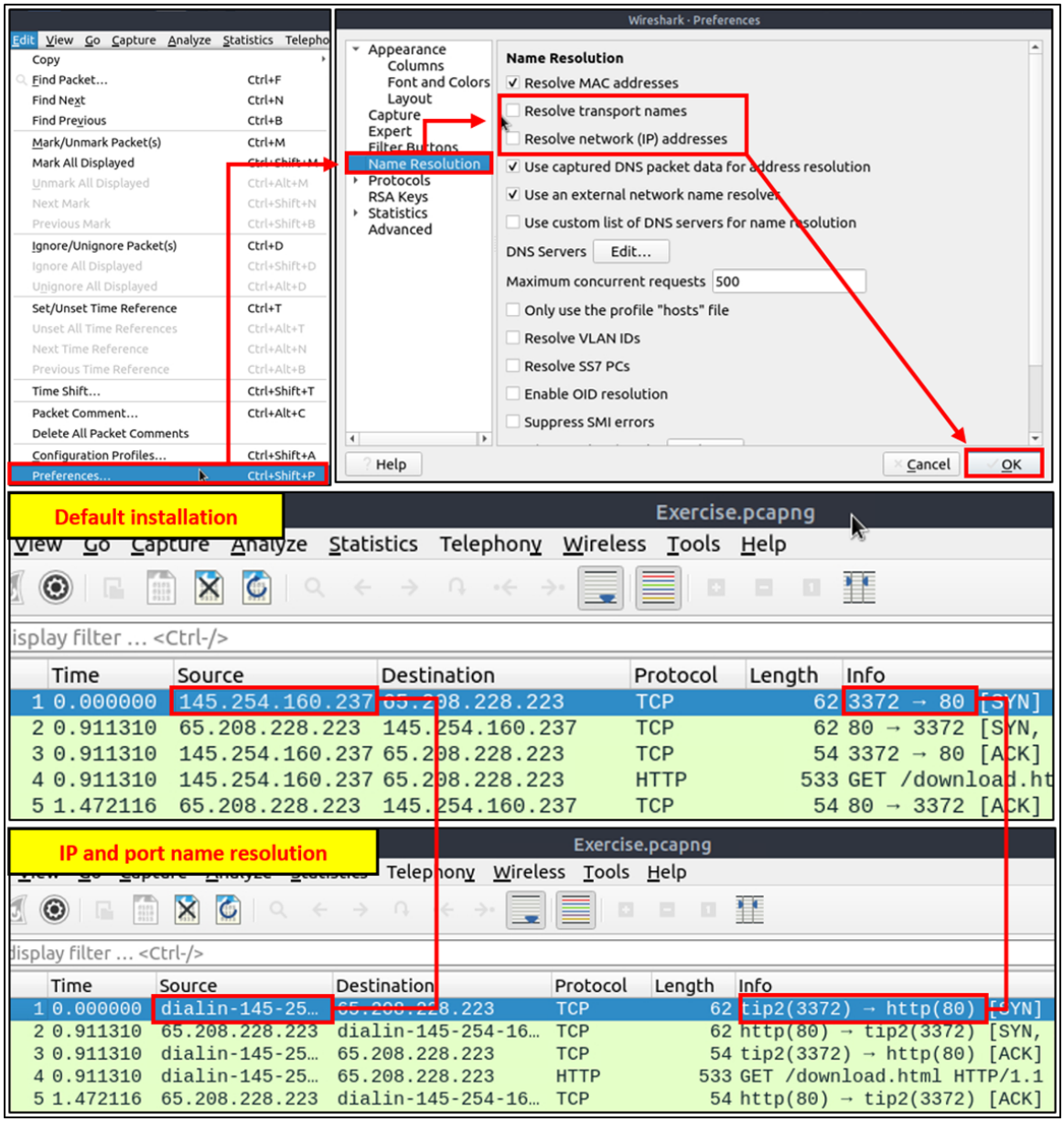
Endpoints

The endpoints option is similar to the conversations option. The only difference is that this option provides unique information for a single information field (Ethernet, IPv4, IPv6, TCP and UDP). Thus analysts can identify the unique endpoints in the capture file and use it for the event of interest. You can use the "Statistics --> Endpoints" menu to view this info.

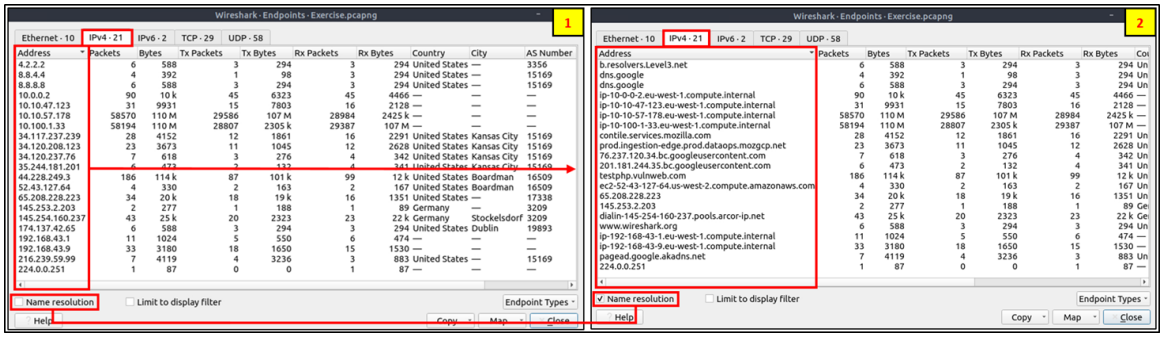
Wireshark also supports resolving MAC addresses to human-readable format using the manufacturer name assigned by IEEE. Note that this conversion is done through the first three bytes of the MAC address and only works for the known manufacturers. When you review the ethernet endpoints, you can activate this option with the **"Name resolution"** button in the lower-left corner of the endpoints window.



- Name resolution is not limited only to MAC addresses. Wireshark provides IP and port name resolution options as well. **However, these options are not enabled by default**.
    - If you want to use these functionalities, you need to activate them through the **"Edit --> Preferences --> Name Resolution"** menu. Once you enable IP and port name resolution, you will see the resolved IP address and port names in the packet list pane and also will be able to view resolved names in the "Conversations" and "Endpoints" menus as well.



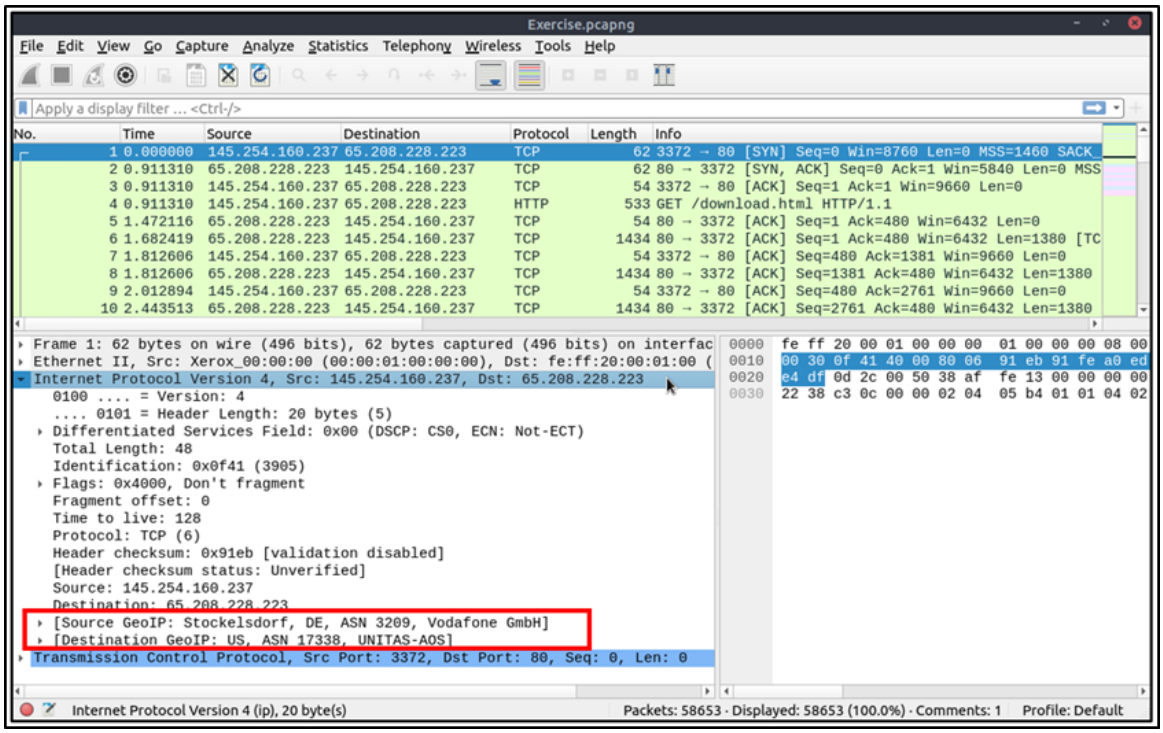
Endpoint menu view with name resolution:



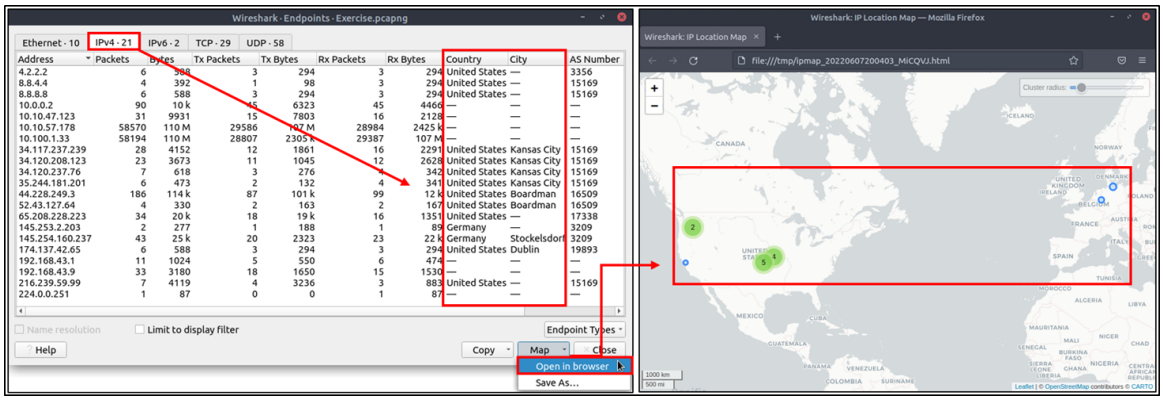
- Besides name resolution, Wireshark also provides an **IP geolocation mapping** that helps analysts identify the map's source and destination

addresses.

- But **this feature is not activated by default** and needs supplementary data like the **GeoIP database**. Currently, Wireshark supports MaxMind databases, and the latest versions of the Wireshark come configured MaxMind DB resolver. However, you still need MaxMind DB files and provide the database path to Wireshark by using the "Edit --> Preferences --> Name Resolution --> MaxMind database directories" menu. Once you download and indicate the path, Wireshark will automatically provide GeoIP information under the IP protocol details for the matched IP addresses.

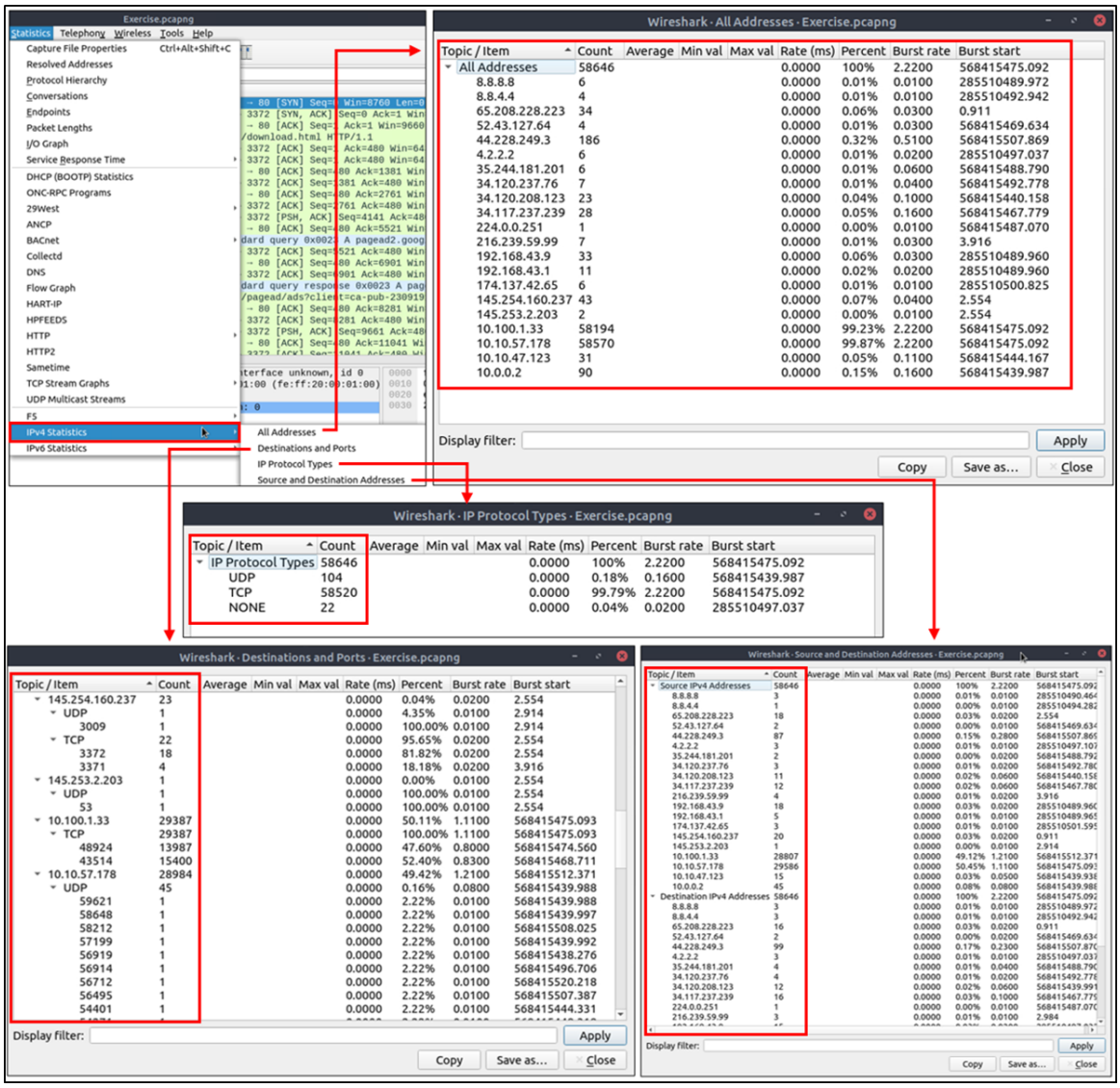


Endpoints and GeoIP view.



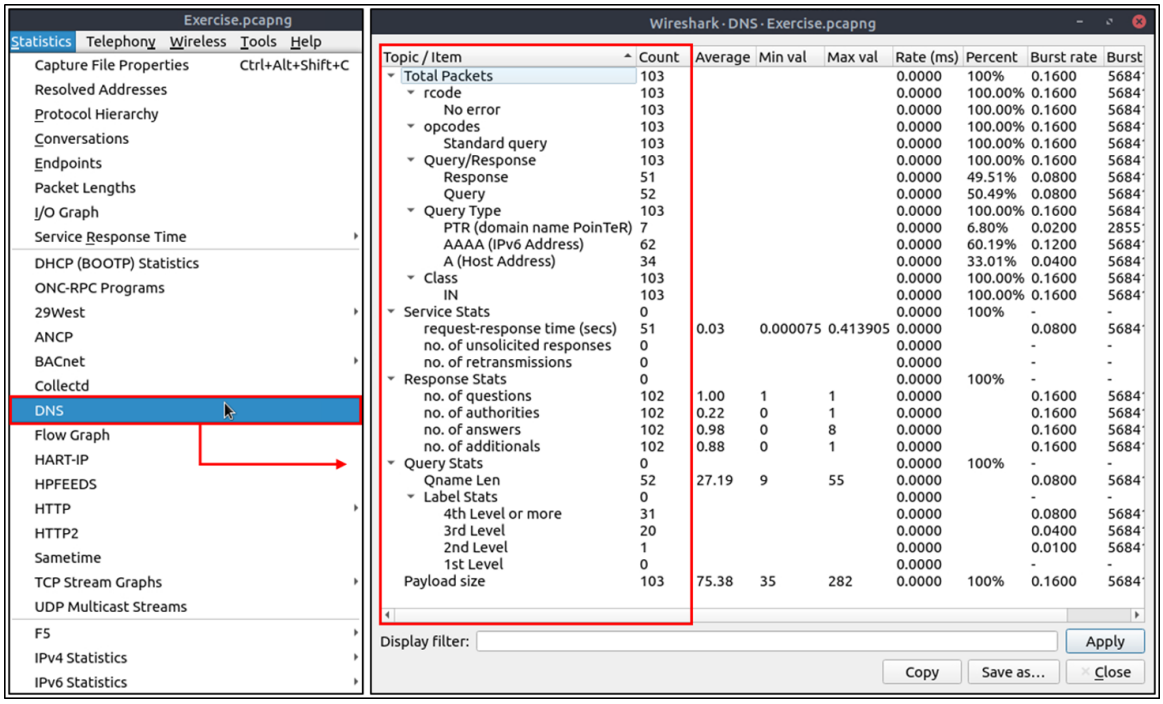
IPv4 and IPv6

Up to here, almost all options provided information that contained both versions of the IP addresses. The statistics menu has two options for narrowing the statistics on packets containing a specific IP version. Thus, analysts can identify and list all events linked to specific IP versions in a single window and use it for the event of interest. You can use the **"Statistics --> IPvX Statistics"** menu to view this info.



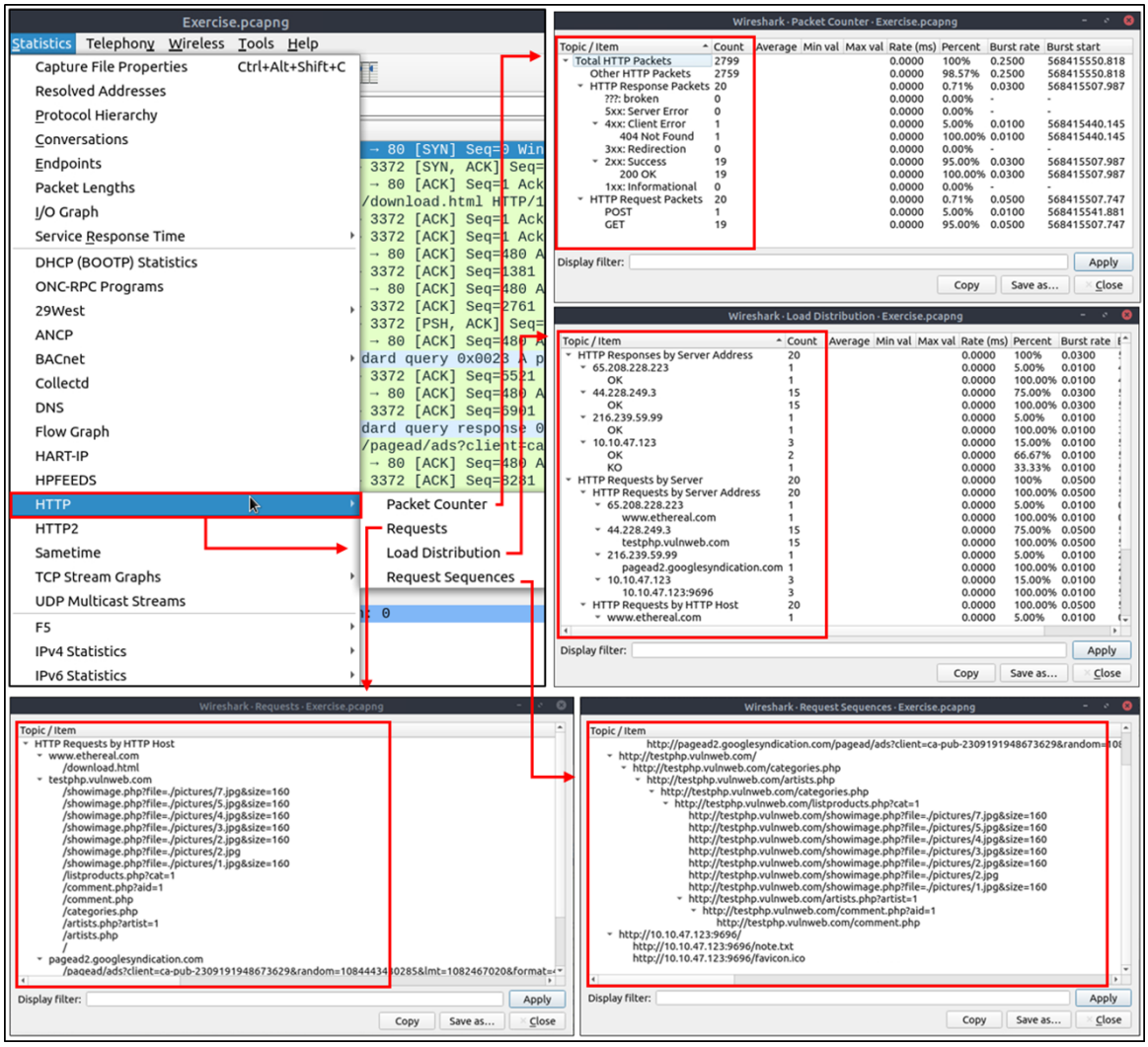
DNS

This option breaks down all DNS packets from the capture file and helps analysts view the findings in a tree view based on packet counters and percentages of the DNS protocol. Thus analysts can view the DNS service's overall usage, including rcode, opcode, class, query type, service and query stats and use it for the event of interest. You can use the **"Statistics --> DNS"** menu to view this info.



HTTP

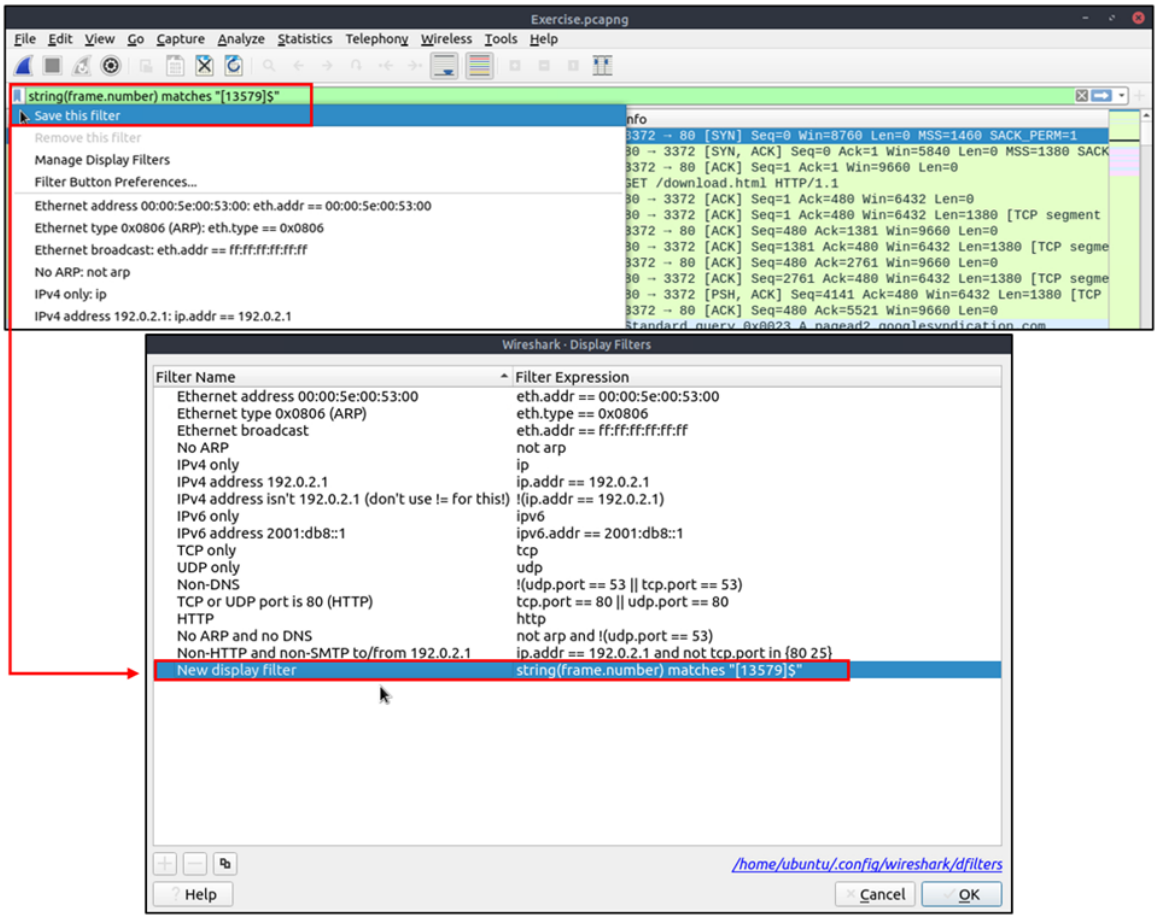
This option breaks down all HTTP packets from the capture file and helps analysts view the findings in a tree view based on packet counters and percentages of the HTTP protocol. Thus analysts can view the HTTP service's overall usage, including request and response codes and the original requests. You can use the **"Statistics --> HTTP"** menu to view this info.



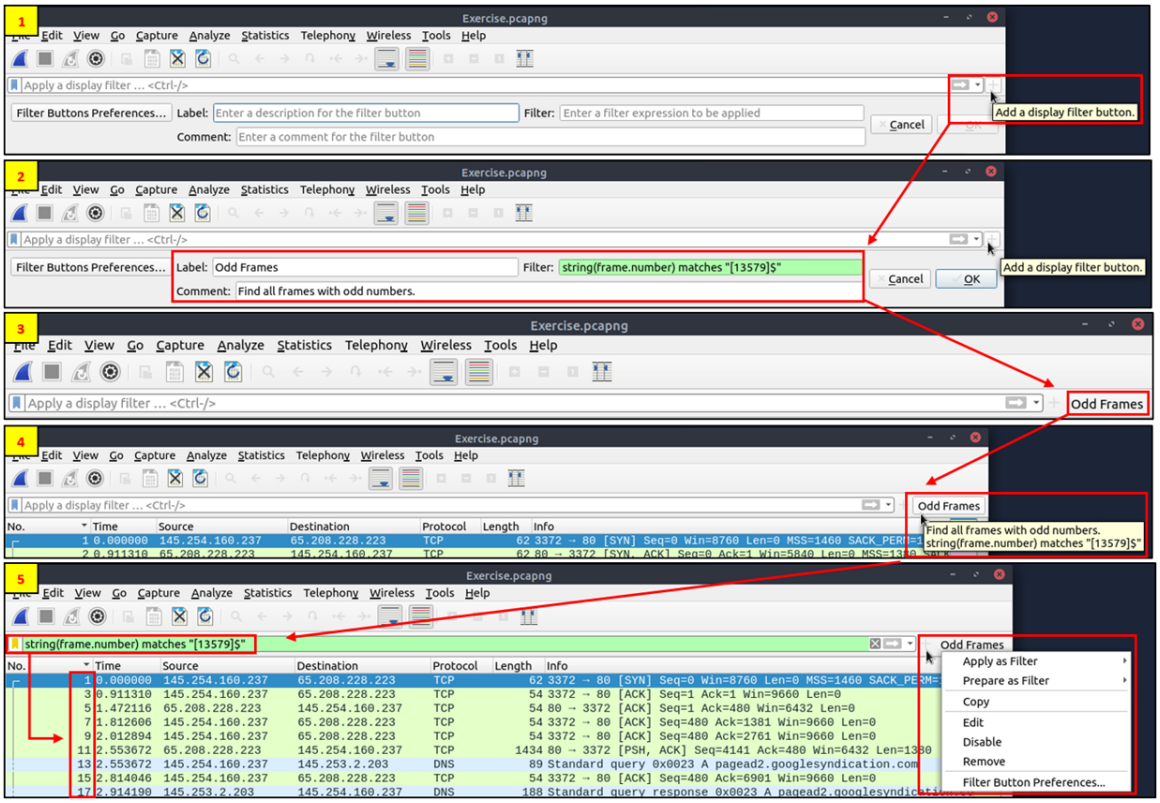
▼ Bookmarks and Filtering Buttons

We've covered different types of filtering options, operators and functions. It is time to create filters and save them as bookmarks and buttons for later usage. As mentioned in the previous task, the filter toolbar has a filter bookmark section to save user-created filters, which helps analysts re-use favourite/complex filters with a couple of clicks. Similar to bookmarks, you can create filter buttons ready to apply with a single click.

Creating and using bookmarks.

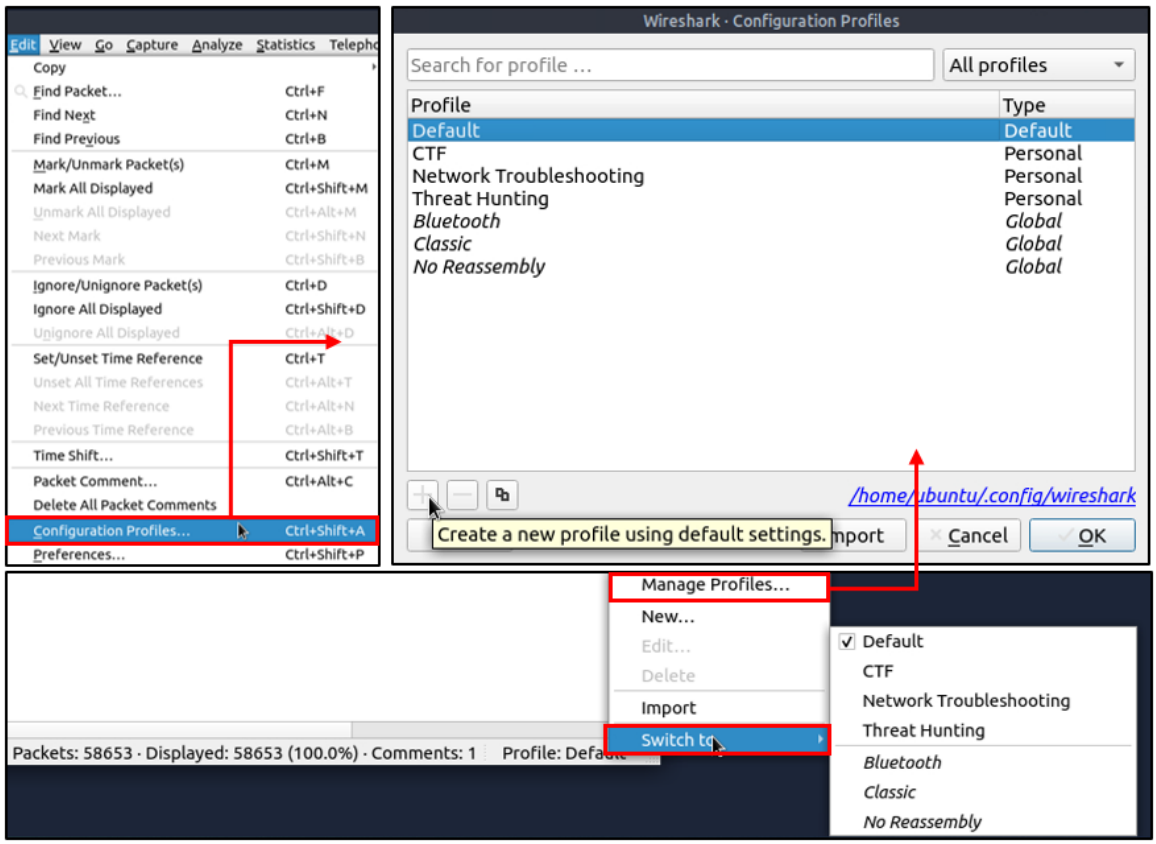


Creating and using display filter buttons.



▼ Profiles

Wireshark is a multifunctional tool that helps analysts to accomplish in-depth packet analysis. As we covered during the room, multiple preferences need to be configured to analyse a specific event of interest. It is cumbersome to re-change the configuration for each investigation case, which requires a different set of colouring rules and filtering buttons. This is where Wireshark profiles come into play. You can create multiple profiles for different investigation cases and use them accordingly. You can use the **"Edit --> Configuration Profiles"** menu or the **"lower right bottom of the status bar --> Profile"** section to create, modify and change the profile configuration.



▼ Packet Filtering

Capture Filters	This type of filter is used to save only a specific part of the traffic. It is set before capturing traffic and not changeable during the capture.
Display Filters	This type of filter is used to investigate packets by reducing the number of visible packets, and it is changeable during the capture.

Note: You cannot use the display filter expressions for capturing traffic and vice versa.

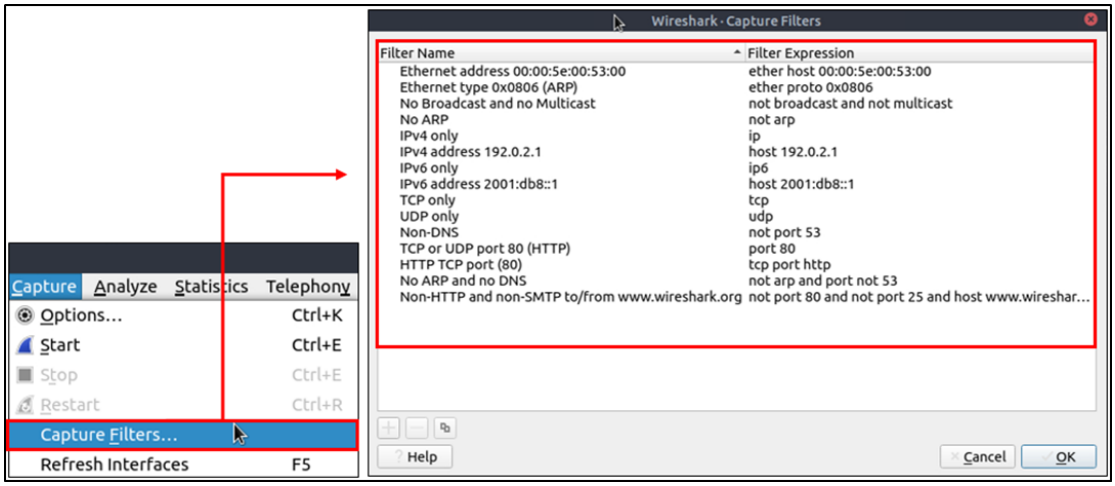
The typical use case is capturing everything and filtering the packets according to the event of interest. Only experienced professionals use capture filters and sniff traffic. This is why Wireshark supports more protocol types in display filters.

▼ Capture Filter Syntax

These filters use byte offsets hex values and masks with boolean operators, and it is not easy to understand/predict the filter's purpose at first glance. The base syntax is explained below:

- **Scope**: host, net, port and portrange.
- **Direction**: src, dst, src or dst, src and dst,
- **Protocol**: ether, wlan, ip, ip6, arp, rarp, tcp and udp.
- **Sample filter to capture port 80 traffic:** `tcp port 80`

You can read more on capture filter syntax from here and here. A quick reference is available under the **"Capture --> Capture Filters"** menu.



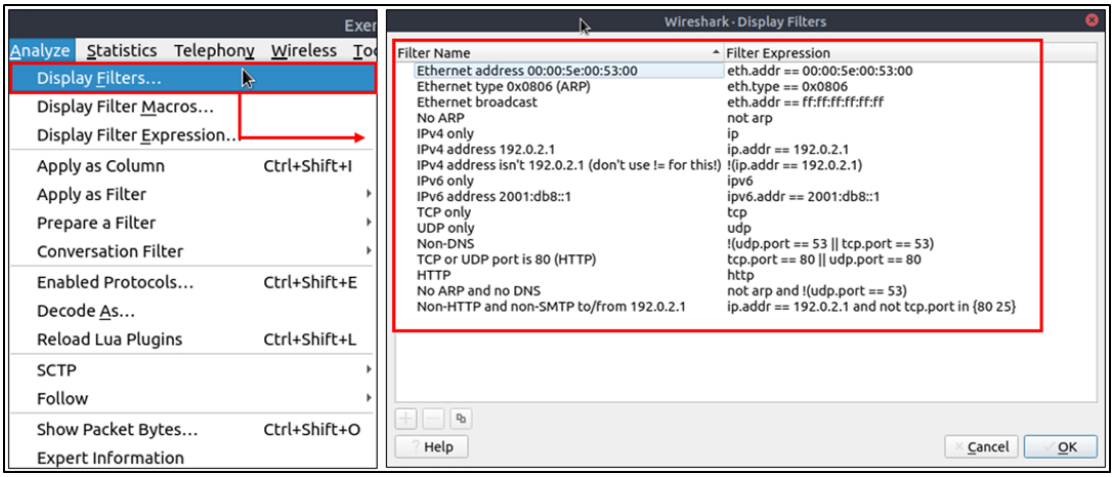
▼ Display Filter Syntax

This is Wireshark's most powerful feature. It supports 3000 protocols and allows conducting packet-level searches under the protocol breakdown. The official "Display Filter Reference" provides all supported protocols breakdown for filtering.

- **Sample filter to capture port 80 traffic:** `tcp.port == 80`

Wireshark has a built-in option (Display Filter Expression) that stores all supported protocol structures to help analysts create display filters. We will cover the "Display Filter Expression" menu later. Now let's understand the

fundamentals of the display filter operations. A quick reference is available under the **"Analyse --> Display Filters"** menu.



Comparison Operators

You can create display filters by using different comparison operators to find the event of interest. The primary operators are shown in the table below.

English	C-Like	Description	Example
eq	==	Equal	`ip.src == 10.10.10.100`
ne	!=	Not equal	`ip.src != 10.10.10.100`
gt	>	Greater than	`ip.ttl > 250`
lt	<	Less Than	`ip.ttl < 10`
ge	>=	Greater than or equal to	`ip.ttl >= 0xFA`
le	<=	Less than or equal to	`ip.ttl <= 0xA`

Note: Wireshark supports decimal and hexadecimal values in filtering. You can use any format you want according to the search you will conduct.

Logical Expressions

Wireshark supports boolean syntax. You can create display filters by using logical operators as well.

English	C-Like	Description	Example
and	&&	Logical AND	`(ip.src == 10.10.10.100) AND (ip.src == 10.10.10.111)`
or	||	Logical OR	`(ip.src == 10.10.10.100) OR (ip.src == 10.10.10.111)`

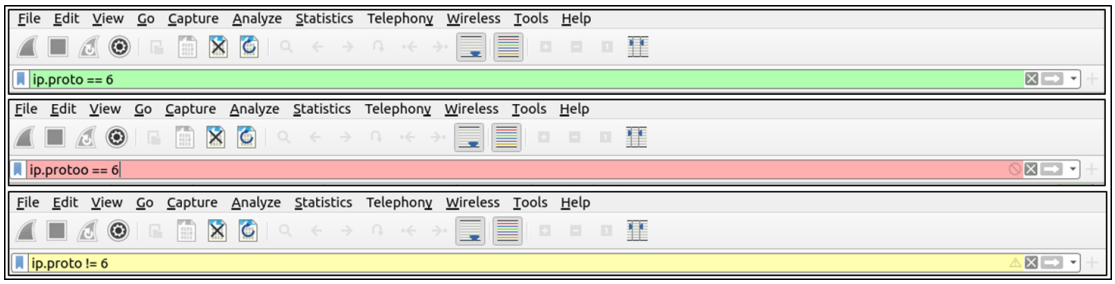
not	**!**	Logical NOT	`!(ip.src == 10.10.10.222)` **Note:** Usage of `!=value` is deprecated; using it could provide inconsistent results. Using the `!(value)` style is suggested for more consistent results.

▼ Packet Filter Toolbar

The filter toolbar is where you create and apply your display filters. It is a smart toolbar that helps you create valid display filters with ease. Before starting to filter packets, here are a few tips:

- Packet filters are defined in lowercase.
- Packet filters have an autocomplete feature to break down protocol details, and each detail is represented by a "dot".
- Packet filters have a three-colour representation explained below.

Green	Valid filter
Red	Invalid filter
Yellow	Warning filter. This filter works, but it is unreliable, and it is suggested to change it with a valid filter.



▼ Protocol Filters

As mentioned in the previous task, Wireshark supports 3000 protocols and allows packet-level investigation by filtering the protocol fields. This task shows the creation and usage of filters against different protocol fields.

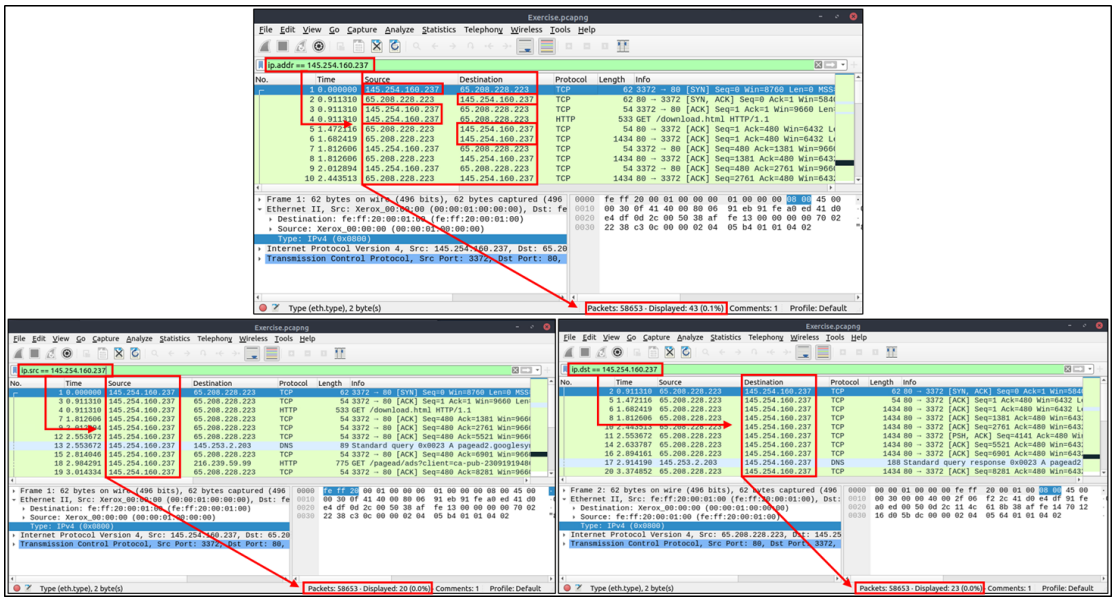
IP Filters

IP filters help analysts filter the traffic according to the IP level information from the packets (Network layer of the OSI model). This is one of the most commonly used filters in Wireshark. These filters filter network-level

information like IP addresses, version, time to live, type of service, flags, and checksum values.

The common filters are shown in the given table.

Filter	Description
`ip`	Show all IP packets.
`ip.addr == 10.10.10.111`	Show all packets containing IP address 10.10.10.111.
`ip.addr == 10.10.10.0/24`	Show all packets containing IP addresses from 10.10.10.0/24 subnet.
`ip.src == 10.10.10.111`	Show all packets originated from 10.10.10.111
`ip.dst == 10.10.10.111`	Show all packets sent to 10.10.10.111
ip.addr vs ip.src/ip.dst	Note: The ip.addr filters the traffic without considering the packet direction. The ip.src/ip.dst filters the packet depending on the packet direction.

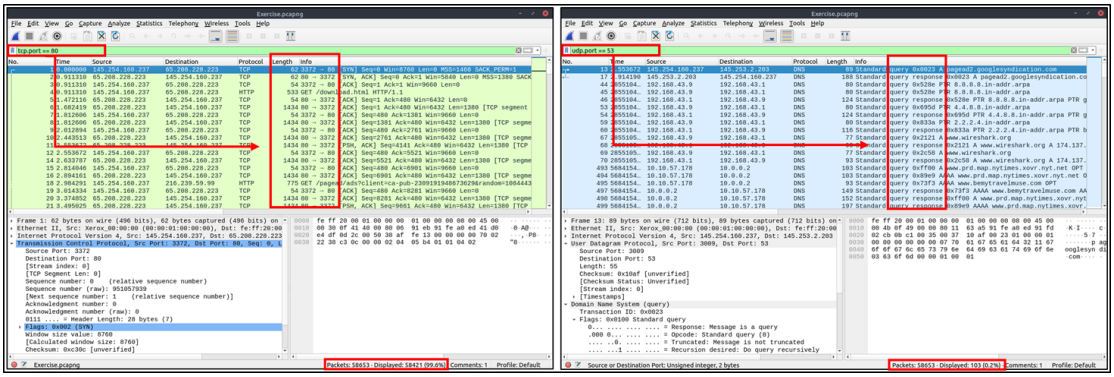


TCP

TCP filters help analysts filter the traffic according to protocol-level information from the packets (Transport layer of the OSI model). These filters filter transport protocol level information like source and destination ports, sequence number, acknowledgement number, windows size, timestamps, flags, length and protocol errors.

Filter	**Description**	Filter	Expression
`tcp.port == 80`	Show all TCP packets	`udp.port == 53`	Show

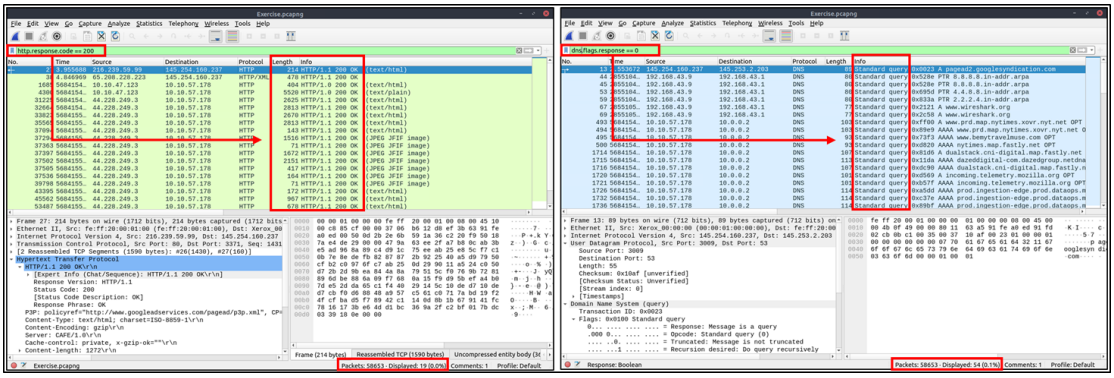
	with port 80		all UDP packets with port 53
`tcp.srcport == 1234`	Show all TCP packets originating from port 1234	`udp.srcport == 1234`	Show all UDP packets originating from port 1234
`tcp.dstport == 80`	Show all TCP packets sent to port 80	`udp.dstport == 5353`	Show all UDP packets sent to port 5353



Application Level Protocol Filters | and DNS

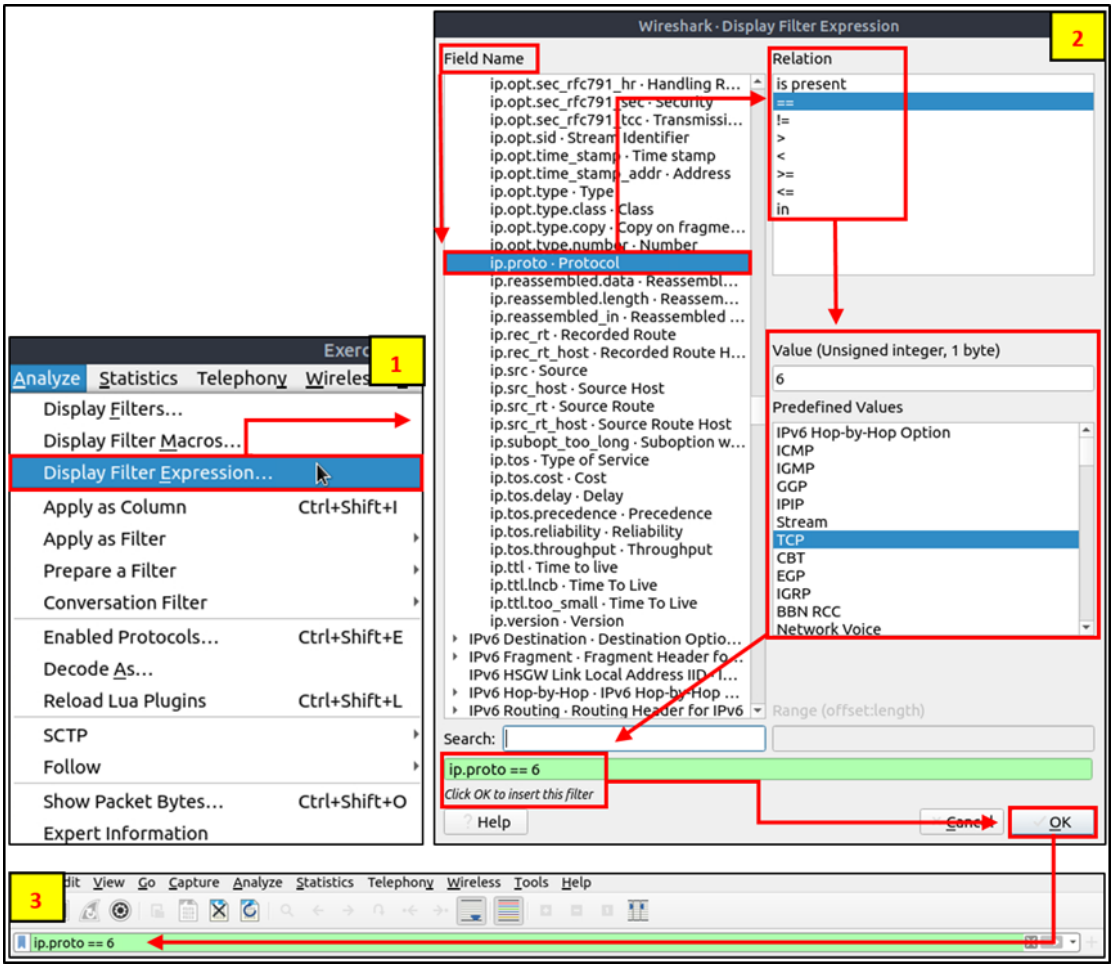
Application-level protocol filters help analysts filter the traffic according to application protocol level information from the packets (Application layer of the OSI model). These filters filter application-specific information, like payload and linked data, depending on the protocol type.

Filter	Description	Filter	Description
`http`	Show all HTTP packets	`dns`	Show all DNS packets
`http.response.code == 200`	Show all packets with HTTP response code "200"	`dns.flags.response == 0`	Show all DNS requests
`http.request.method == "GET"`	Show all HTTP GET requests	`dns.flags.response == 1`	Show all DNS responses
`http.request.method == "POST"`	Show all HTTP POST requests	`dns.qry.type == 1`	Show all DNS "A" records



Display Filter Expressions

- Wireshark has a built-in option (Display Filter Expression) that stores all supported protocol structures to help analysts create display filters. When an analyst can't recall the required filter for a specific protocol or is unsure about the assignable values for a filter, the Display Filter Expressions menu provides an easy-to-use display filter builder guide. It is available under the "**Analyse --> Display Filter Expression**" menu.

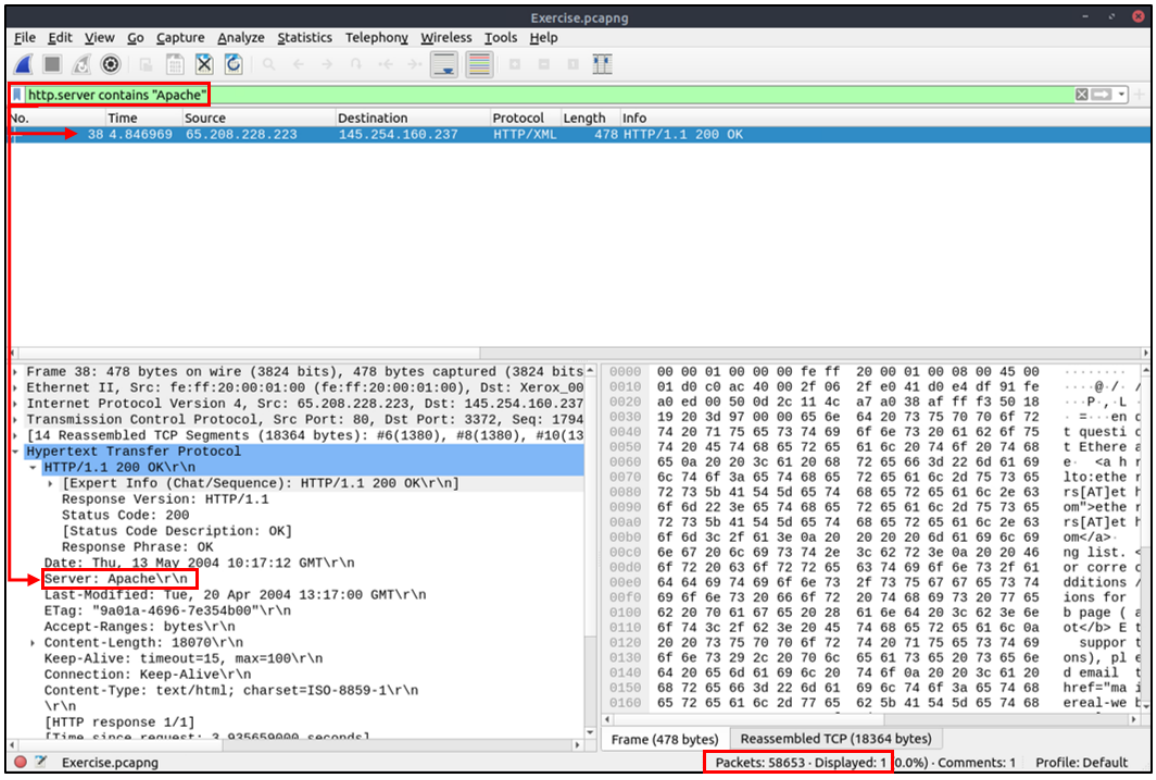


▼ Advanced Filtering

So far, you have learned the basics of packet filtering operations. Now it is time to focus on specific packet details for the event of interest. Besides the operators and expressions covered in the previous room, Wireshark has advanced operators and functions. These advanced filtering options help the analyst conduct an in-depth analysis of an event of interest.

Filter: "contains"

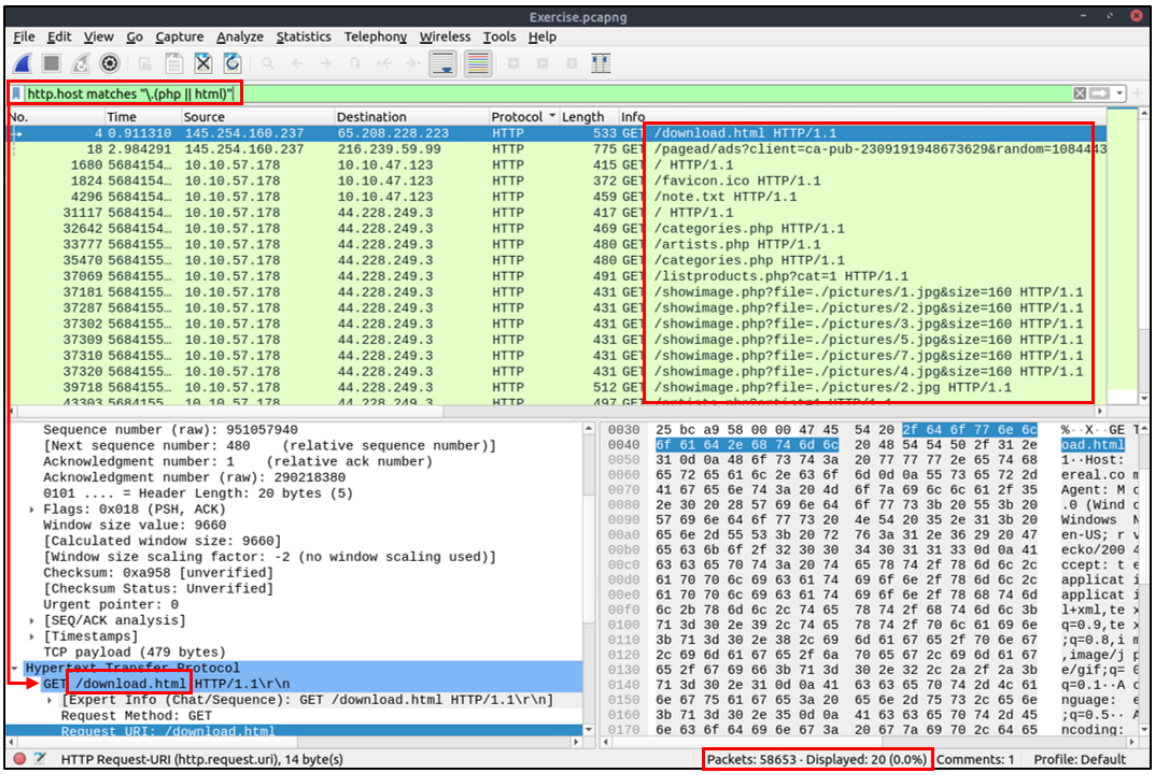
Filter	**contains**
Type	Comparison Operator
Description	Search a value inside packets. It is case-sensitive and provides similar functionality to the "Find" option by focusing on a specific field.
Example	Find all "Apache" servers.
Workflow	List all HTTP packets where packets' "server" field contains the "Apache" keyword.
Usage	`http.server contains "Apache"`



Filter: "matches"

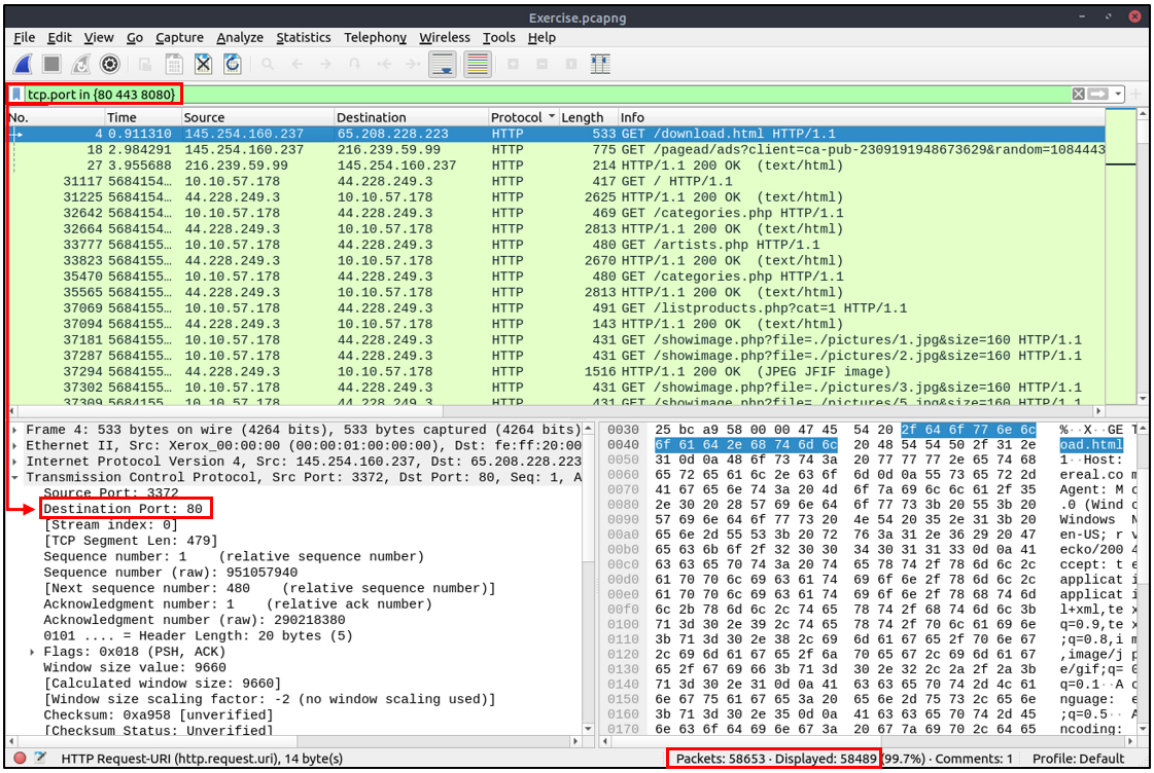
Filter	**matches**

Type	Comparison Operator
Description	Search a pattern of a regular expression. It is case insensitive, and complex queries have a margin of error.
Example	Find all .php and .html pages.
Workflow	List all HTTP packets where packets' "host" fields match keywords ".php" or ".html".
Usage	`http.host matches "\.(php|html)"`



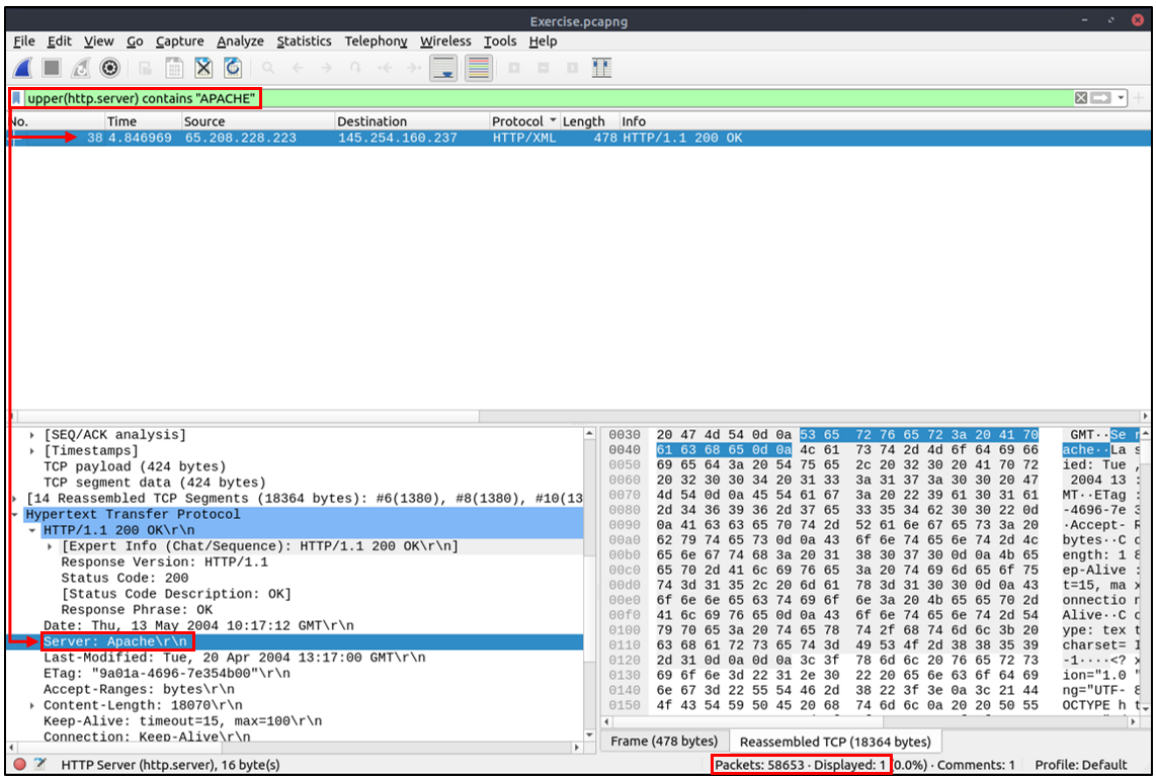
Filter: "in"

Filter	**in**
Type	Set Membership
Description	Search a value or field inside of a specific scope/range.
Example	Find all packets that use ports 80, 443 or 8080.
Workflow	List all TCP packets where packets' "port" fields have values 80, 443 or 8080.
Usage	`tcp.port in {80 443 8080}`



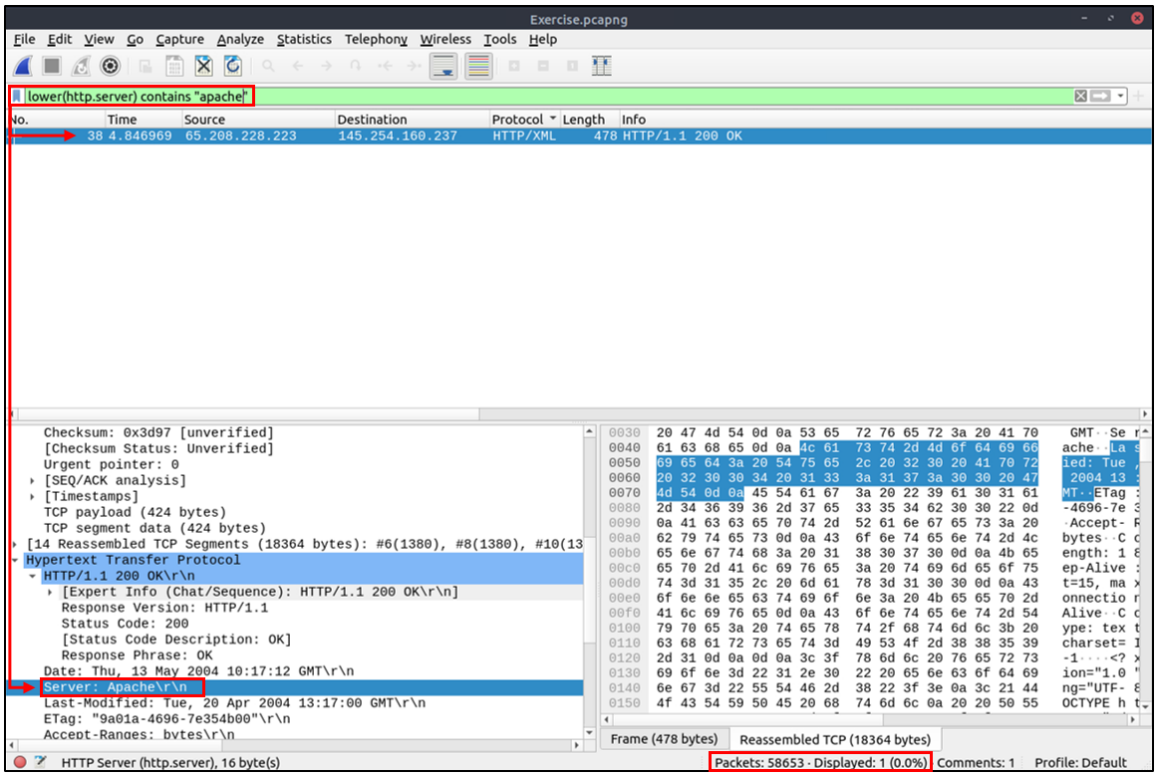
Filter: "upper"

Filter	**upper**
Type	Function
Description	Convert a string value to uppercase.
Example	Find all "APACHE" servers.
Workflow	Convert all HTTP packets' "server" fields to uppercase and list packets that contain the "APACHE" keyword.
Usage	`upper(http.server) contains "APACHE"`



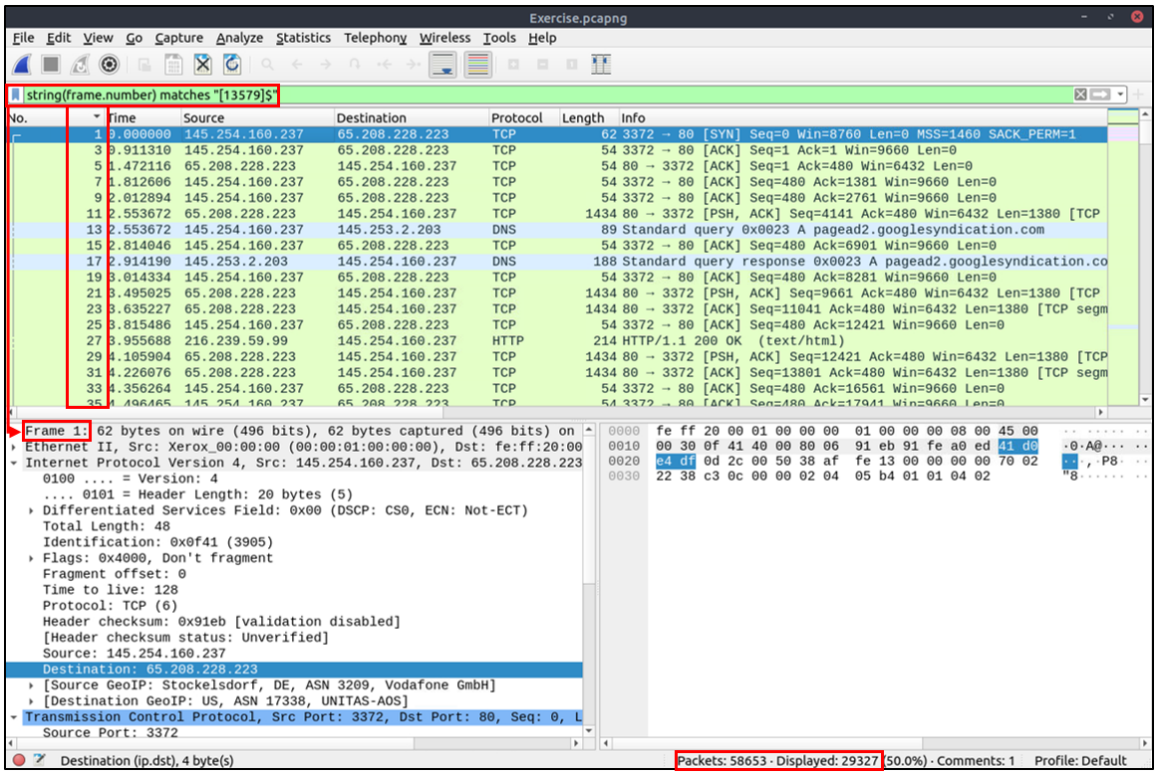
Filter: "lower"

Filter	**lower**
Type	Function
Description	Convert a string value to lowercase.
Example	Find all "apache" servers.
Workflow	Convert all HTTP packets' "server" fields info to lowercase and list packets that contain the "apache" keyword.
Usage	`lower(http.server) contains "apache"`



Filter: "string"

Filter	**string**
Type	Function
Description	Convert a non-string value to a string.
Example	Find all frames with odd numbers.
Workflow	Convert all "frame number" fields to string values, and list frames end with odd values.
Usage	`string(frame.number) matches "[13579]$"`



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

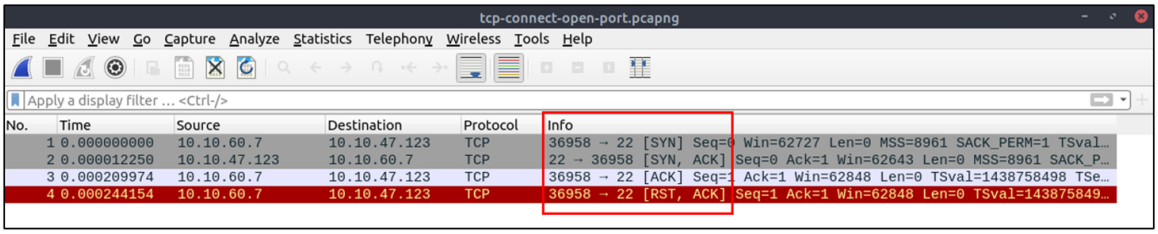
TCP Connect Scan in a nutshell:

- 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.**

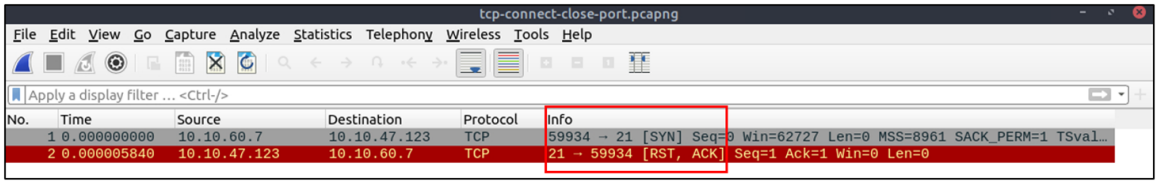
Open TCP Port	Open TCP Port	Closed TCP Port
• SYN --> • <-- SYN, ACK • **ACK -->**	• SYN --> • <-- SYN, ACK • ACK --> • RST, ACK -->	• SYN --> • <-- RST, ACK

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):



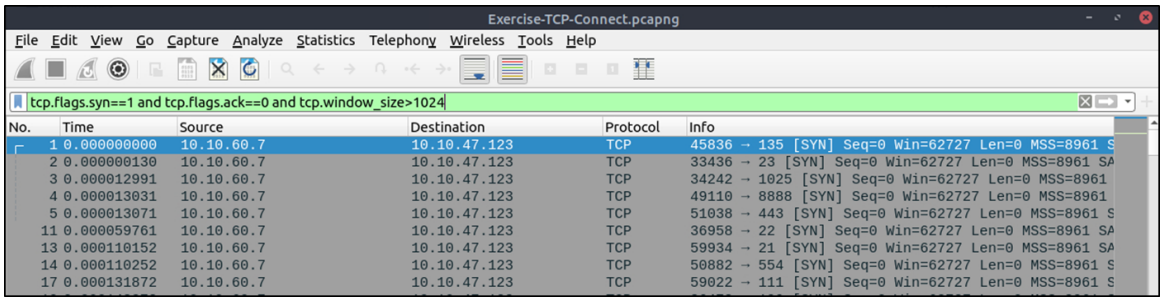
Closed TCP port (Connect):



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

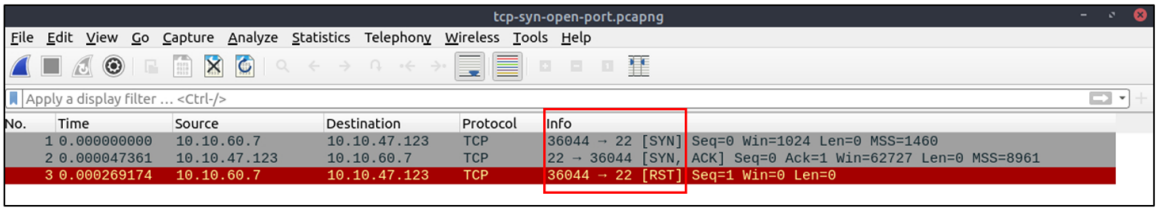
TCP SYN Scan in a nutshell:

- 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.

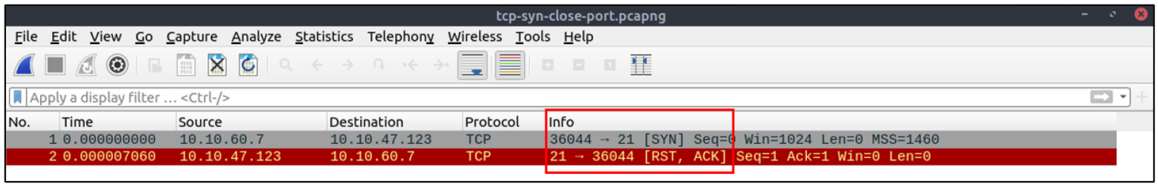
Open TCP Port	Close TCP Port

• SYN --> • <-- SYN,ACK • **RST-->**	• SYN --> • <-- RST,ACK

Open TCP port (SYN):

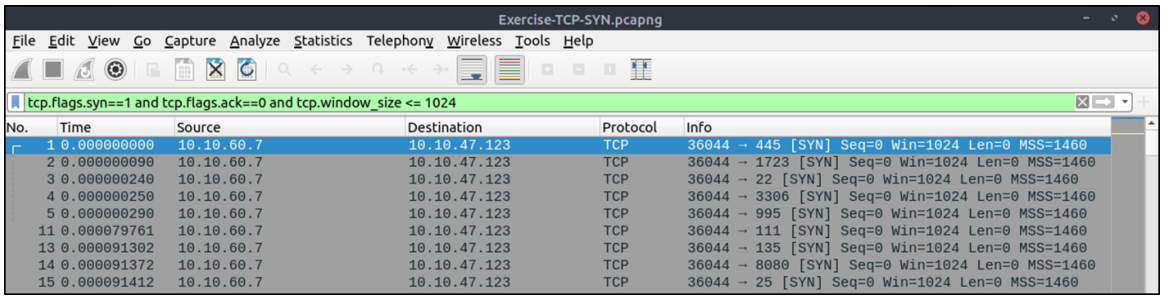


Closed TCP port (SYN):



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
```



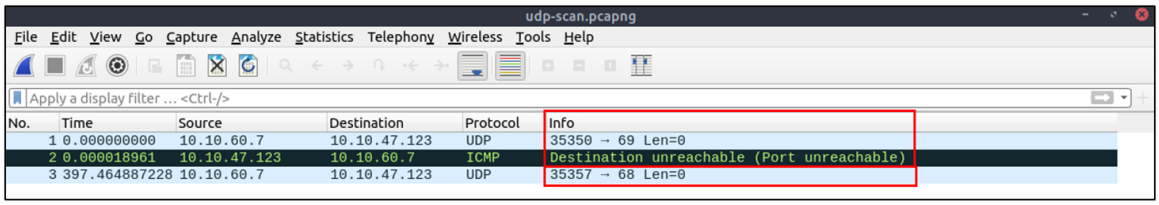
UDP

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.

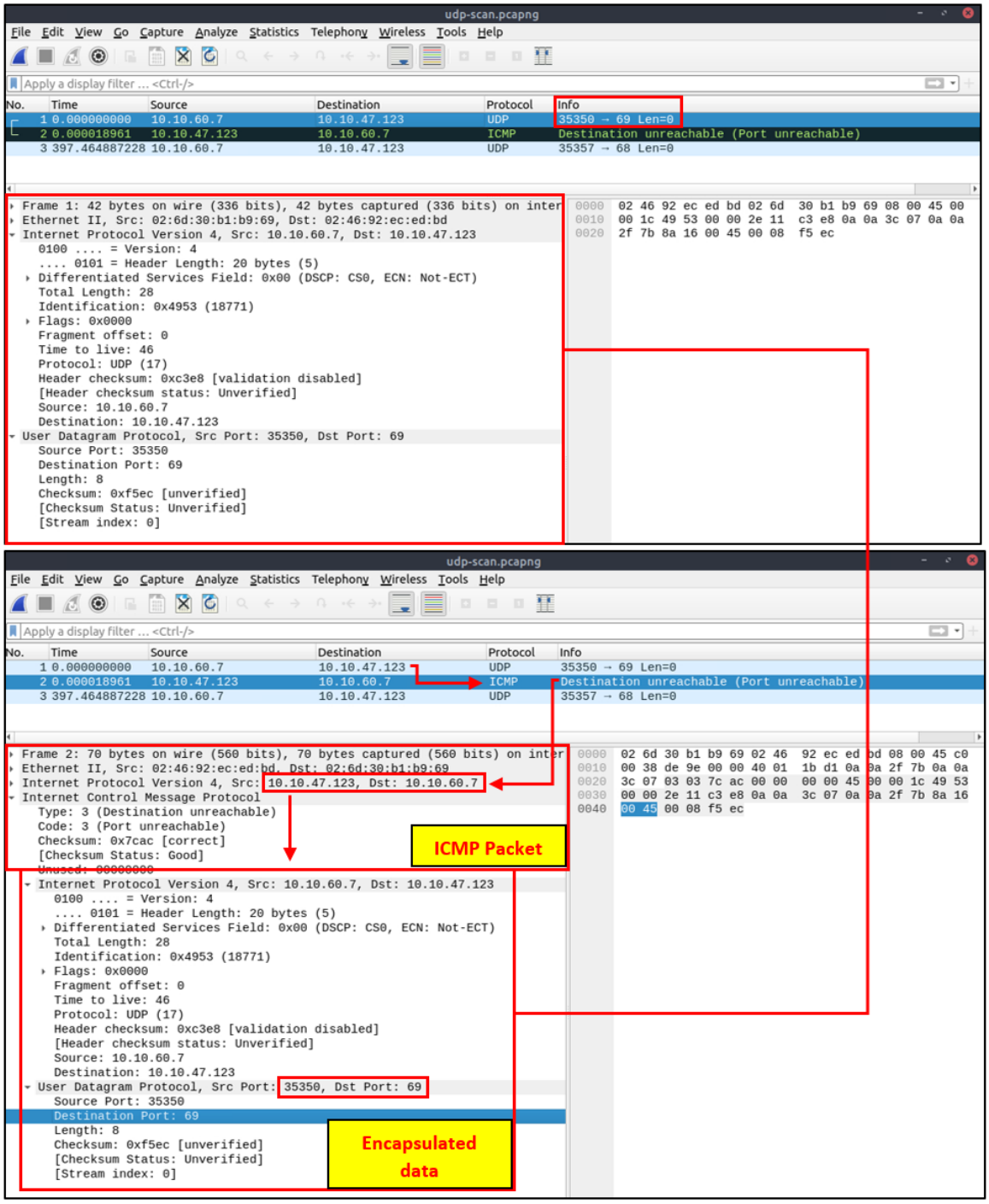
Open UDP Port	Closed UDP Port
• UDP packet -->	• UDP packet --> • ICMP Type 3, Code 3 message. (Destination unreachable, port unreachable)

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



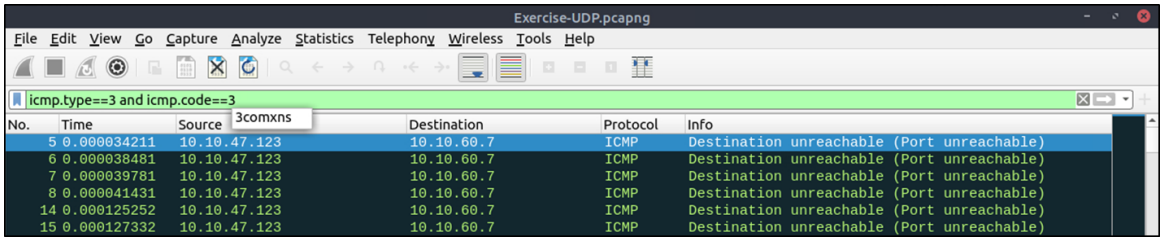
The above image shows that the closed port returns an ICMP error packet. No further information is provided about the error at first glance, so how can an analyst decide where this error message belongs?

- 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
```



▼ ARP Poisoning & Man In The Middle!

- **ARP** protocol, or **A**ddress **R**esolution **P**rotocol (**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.

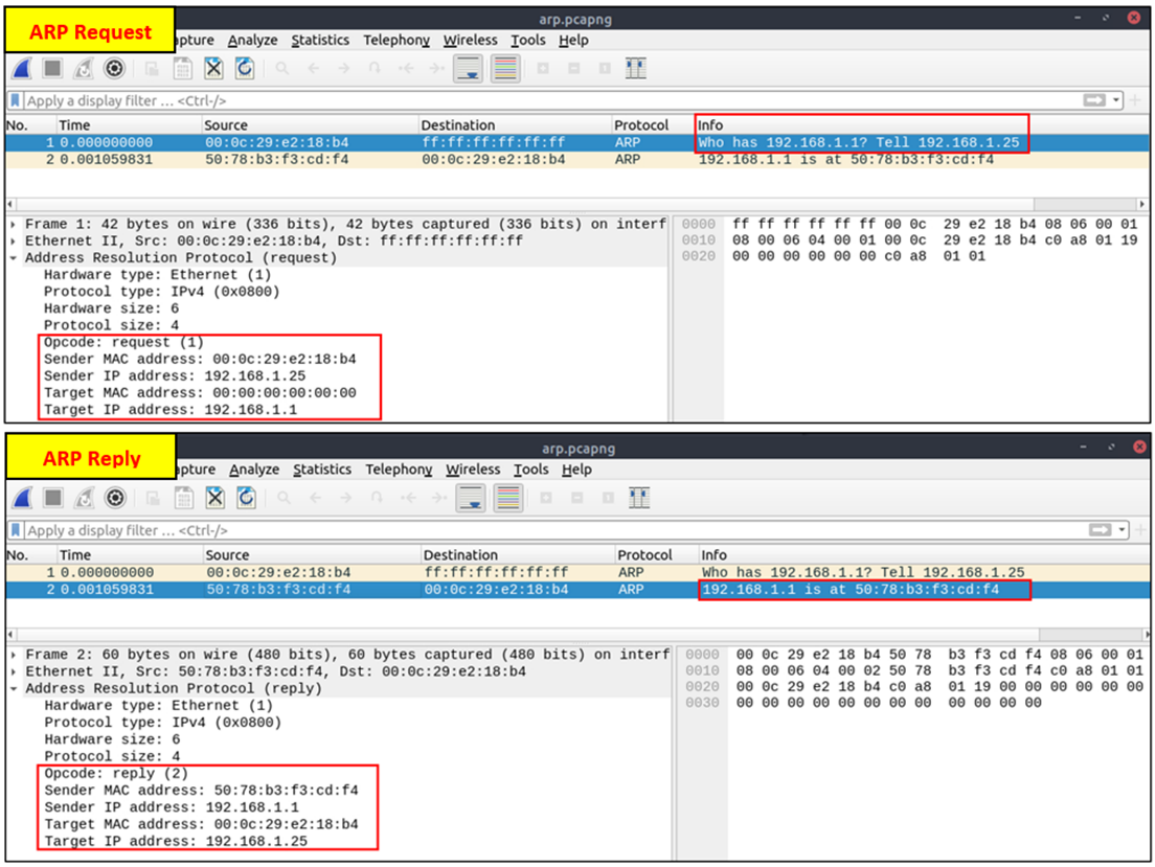
There are a variety of tools available to conduct ARP attacks. However, the mindset of the attack is static, so it is easy to detect such an attack by knowing the ARP protocol workflow and Wireshark skills.

ARP analysis in a nutshell:

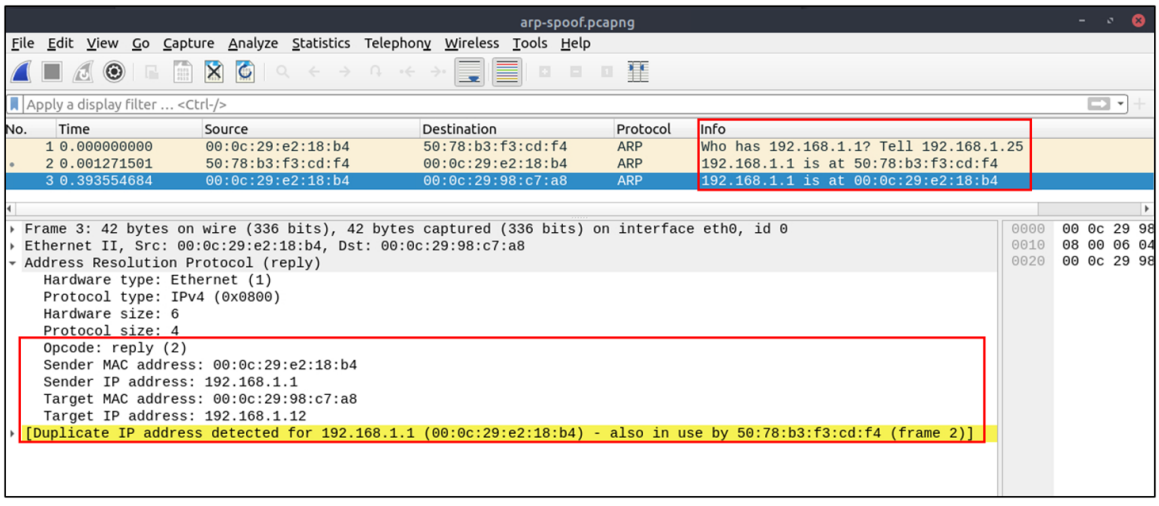
- Works on the local network
- Enables the communication between MAC addresses
- Not a secure protocol
- Not a routable protocol
- It doesn't have an authentication function
- Common patterns are request & response, announcement and gratuitous packets.

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.

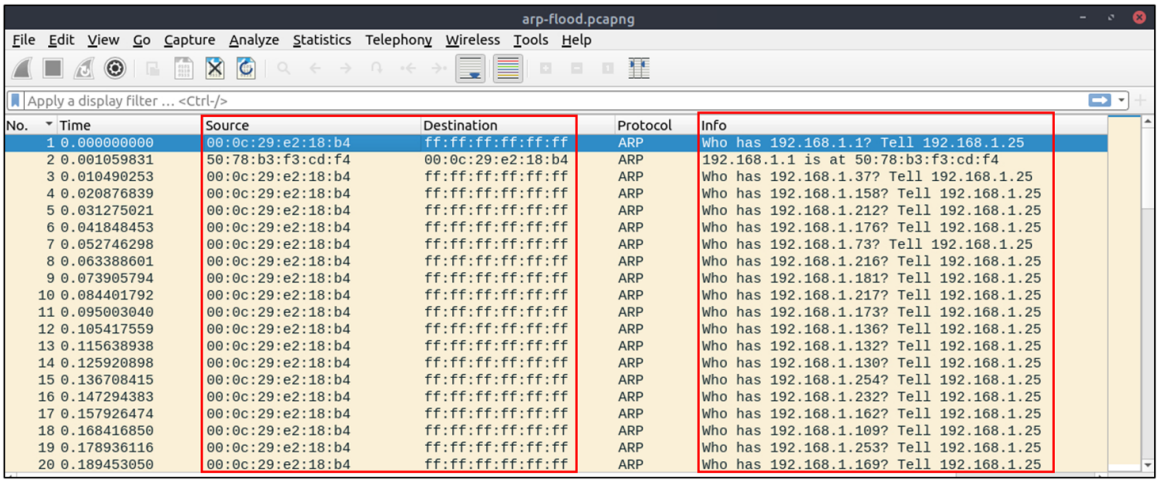


Here, knowing the network architecture and inspecting the traffic for a specific time frame can help detect the anomaly. As an analyst, you should take notes of your findings before going further. This will help you be organised and make it easier to correlate the further findings.

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

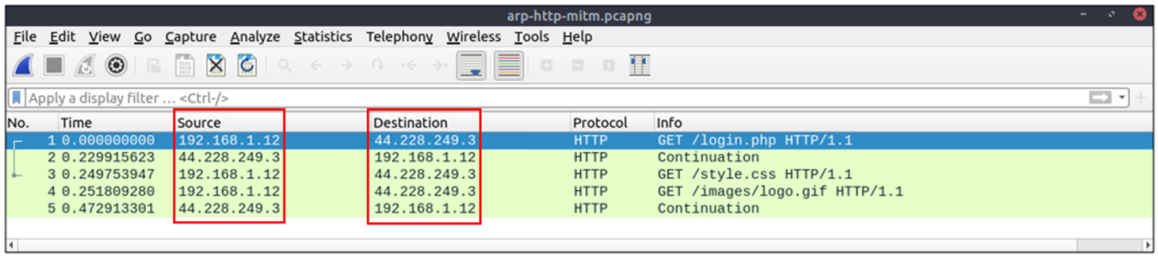
Let's keep inspecting the traffic to spot any other anomalies. Note that the case is split into multiple capture files to make the investigation easier.



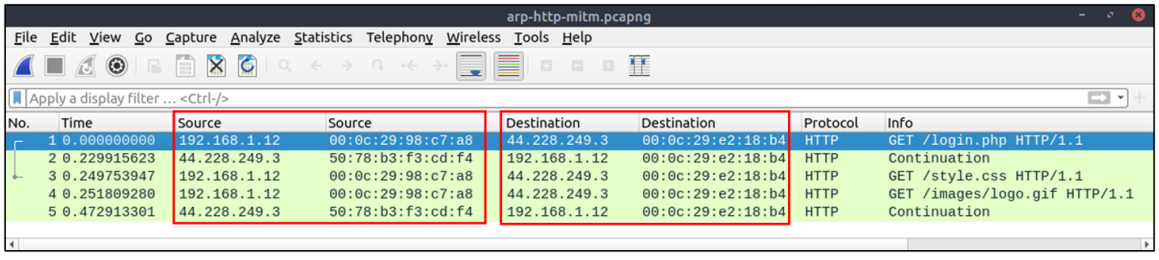
At this point, it is evident that there is an anomaly. A security analyst cannot ignore a flood of ARP requests. This could be malicious activity, scan or network problems. There is a new anomaly; the MAC address that ends with "b4" crafted multiple ARP requests with the "192.168.1.25" IP address. Let's 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.
Attacker	The attacker created noise with ARP packets.
Router/gateway	Gateway address.
Victim	The attacker sniffed all traffic of the victim.

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.

▼ 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

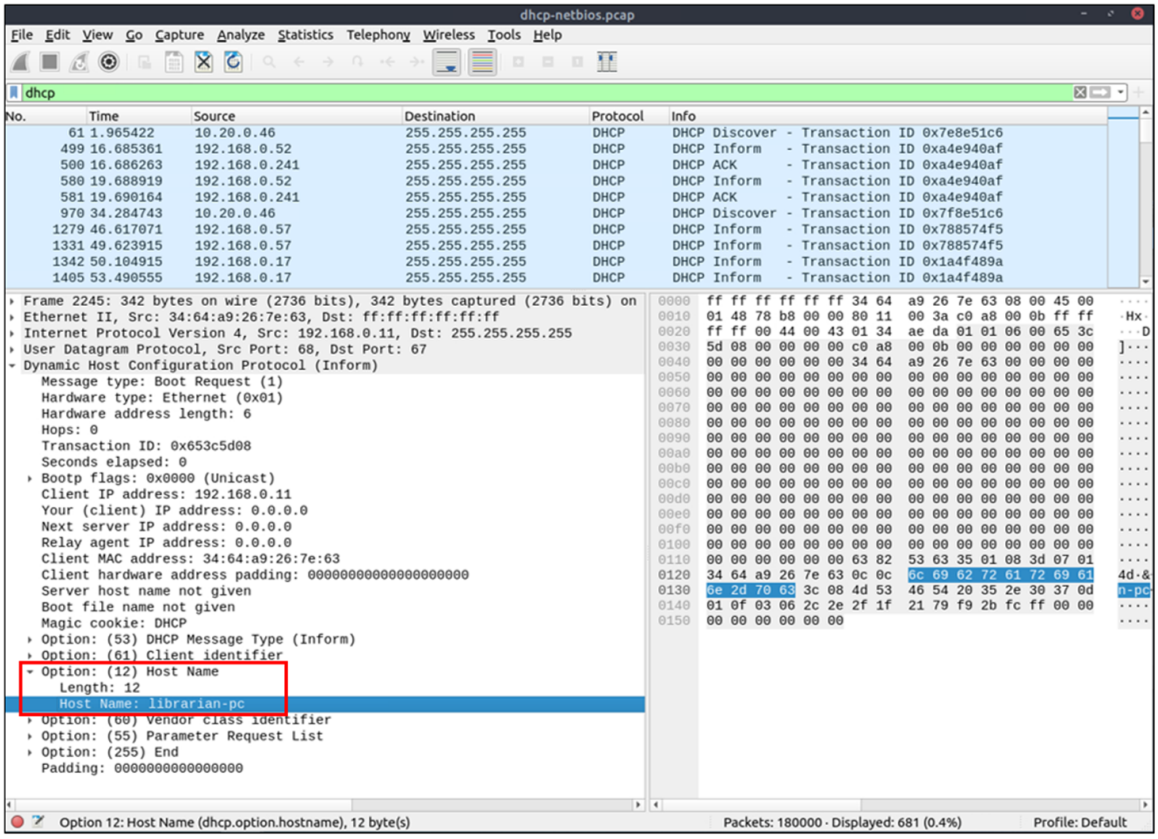
- **NetBIOS** (NBNS) traffic
- **Kerberos** traffic

DHCP

Dynamic **H**ost **C**onfiguration **P**rotocol **(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.

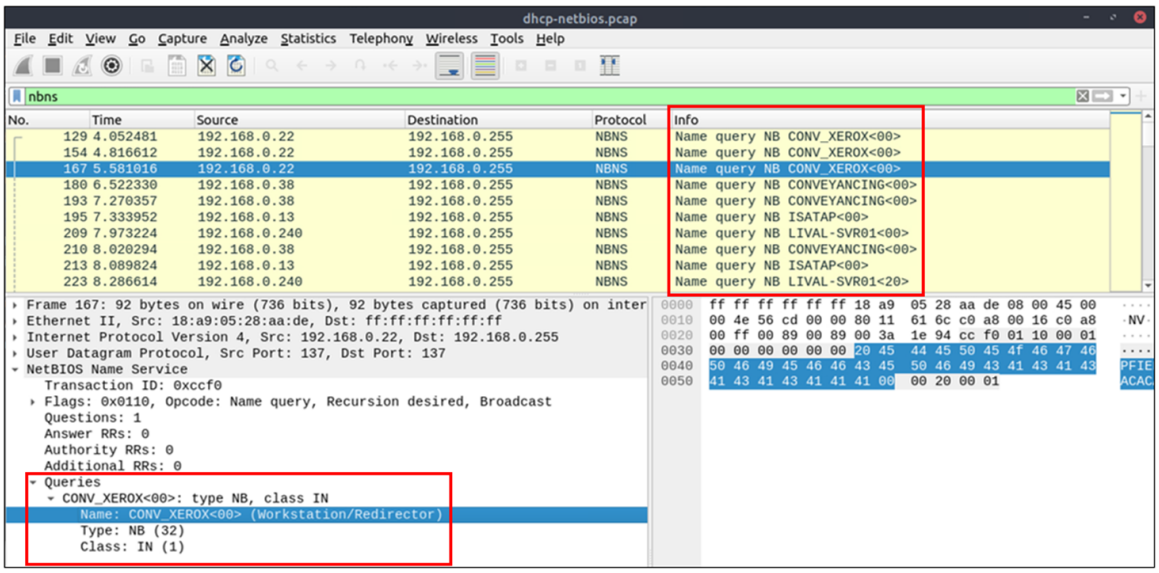


NetBIOS (NBNS) Analysis

NetBIOS or **Net**work **B**asic **I**nput/**O**utput **S**ystem 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"`
NetBIOS registration requests	`nbns.flags.opcode == 5`

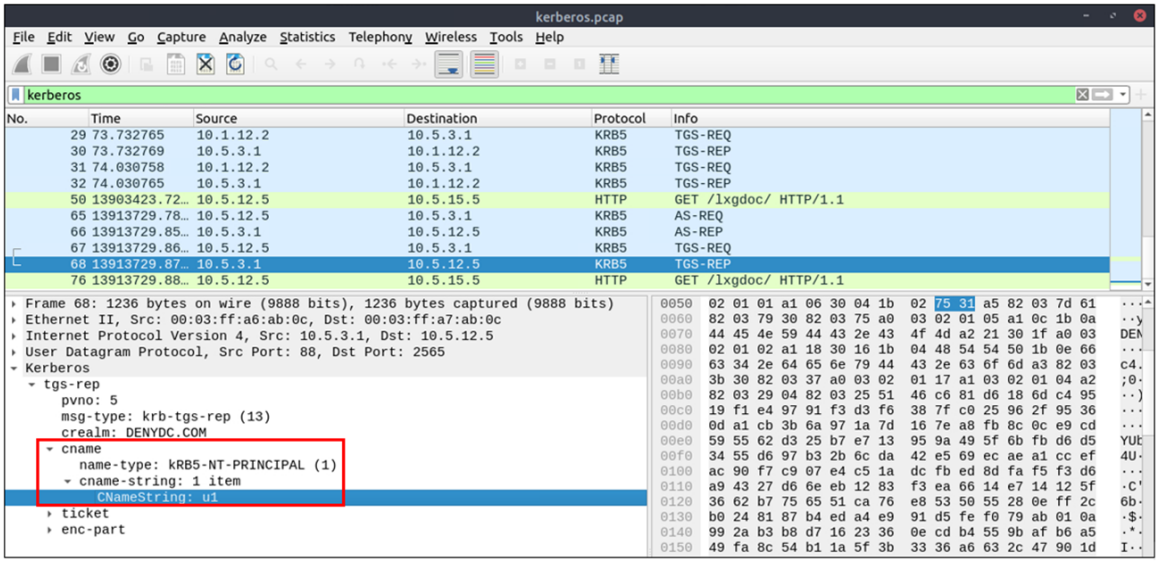


Kerberos Analysis

- **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"`



▼ Tunneling Traffic: DNS and ICMP

Tunnelling Traffic: ICMP and DNS

- Traffic tunnelling is (also known as **"port forwarding"**) transferring the data/resources in a secure method to network segments and zones.
- It can be used for "**internet to private networks**" and "**private networks to internet**" flow/direction.
- There is an **encapsulation process** to **hide** the **data**, so the transferred data appear natural for the case, but it contains private data packets and transfers them to the final destination securely.

Tunnelling provides **anonymity** and traffic security. Therefore it is highly used by enterprise networks. However, as it gives a significant level of data encryption, **attackers** use tunnelling to **bypass security perimeters** using the standard and trusted protocols used in everyday traffic like ICMP and DNS. Therefore, for a security analyst, it is crucial to have the ability to spot ICMP and DNS anomalies.

ICMP Analysis

- Internet Control Message Protocol (ICMP) is designed for diagnosing and reporting network communication issues. It is highly used in error reporting and testing.
- As it is a trusted network layer protocol, sometimes it is used for denial of service (DoS) attacks; also, adversaries use it in **data exfiltration and C2**

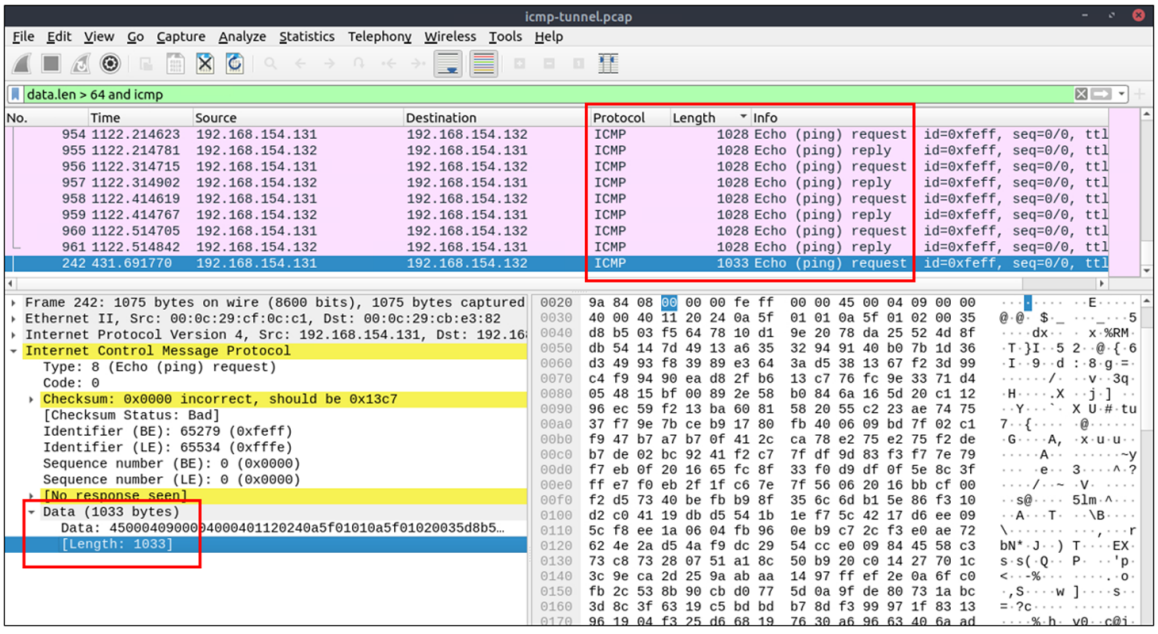
tunnelling activities.

ICMP analysis in a nutshell:

- Usually, ICMP tunnelling attacks are **anomalies** appearing/starting after a malware execution or vulnerability exploitation.
- As the ICMP packets can transfer an additional data payload, adversaries use this section to **exfiltrate data** and establish a C2 connection.
- It could be a TCP, HTTP or SSH data. As the ICMP protocols provide a great opportunity to carry extra data, it also has **disadvantages**.
    - Most enterprise networks **block** custom packets or require administrator privileges to create custom ICMP packets.

A large volume of ICMP traffic or anomalous packet sizes are indicators of ICMP tunnelling. Still, the adversaries could create custom packets that match the regular ICMP packet size (64 bytes), so it is still cumbersome to detect these tunnelling activities. However, a security analyst should know the normal and the abnormal to spot the possible anomaly and escalate it for further analysis.

Notes	Wireshark filters
Global search	• `icmp`
"ICMP" options for grabbing the low-hanging fruits: • **Packet length**. • **ICMP destination addresses**. • **Encapsulated** protocol signs in ICMP payload.	• `data.len > 64 and icmp`



DNS Analysis

- Domain Name System (DNS) is designed to **translate/convert IP domain addresses to IP addresses.**
- It is also known as a phonebook of the internet. As it is the essential part of web services, it is commonly used and trusted, and therefore often ignored. Due to that, **adversaries use it in data exfiltration and C2 activities.**

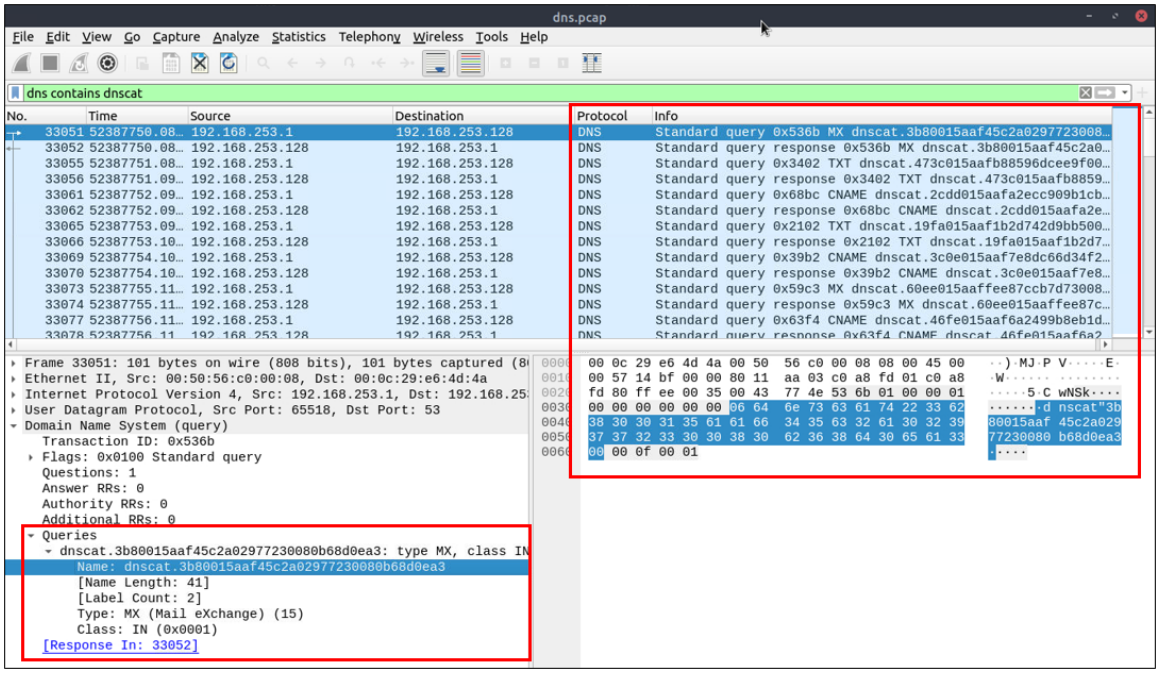
DNS analysis in a nutshell:

- Similar to ICMP tunnels, DNS attacks are anomalies appearing/starting after a malware execution or vulnerability exploitation.
- Adversary creates (or already has) a **domain address** and configures it as a **C2 channel**. The malware or the commands executed after exploitation sends **DNS queries to the C2 server.**
- However, these queries are **longer than default DNS queries** and crafted for subdomain addresses. Unfortunately, these subdomain addresses are not actual addresses; they are encoded commands as shown below:

"encoded-commands.maliciousdomain.com"

- When this query is routed to the C2 server, the server sends the actual malicious commands to the host.
- As the DNS queries are a natural part of the networking activity, these packets have the chance of not being detected by network perimeters. A security analyst should know how to investigate the DNS packet lengths and target addresses to spot these anomalies.

Notes	Wireshark Filter
Global search	• `dns`
"DNS" options for grabbing the low-hanging fruits: • **Query length**. • **Anomalous** and non-regular **names** in DNS addresses. • **Long DNS addresses** with **encoded subdomain addresses**. • Known patterns like **dnscat** and **dns2tcp**. • Statistical analysis like the anomalous **volume** of DNS requests for a particular target. **!mdns: Disable local link device queries.**	• `dns contains "dnscat"` • `dns.qry.name.len > 15 and !mdns`



▼ **Cleartext Protocol Analysis: FTP**

Cleartext Protocol Analysis

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

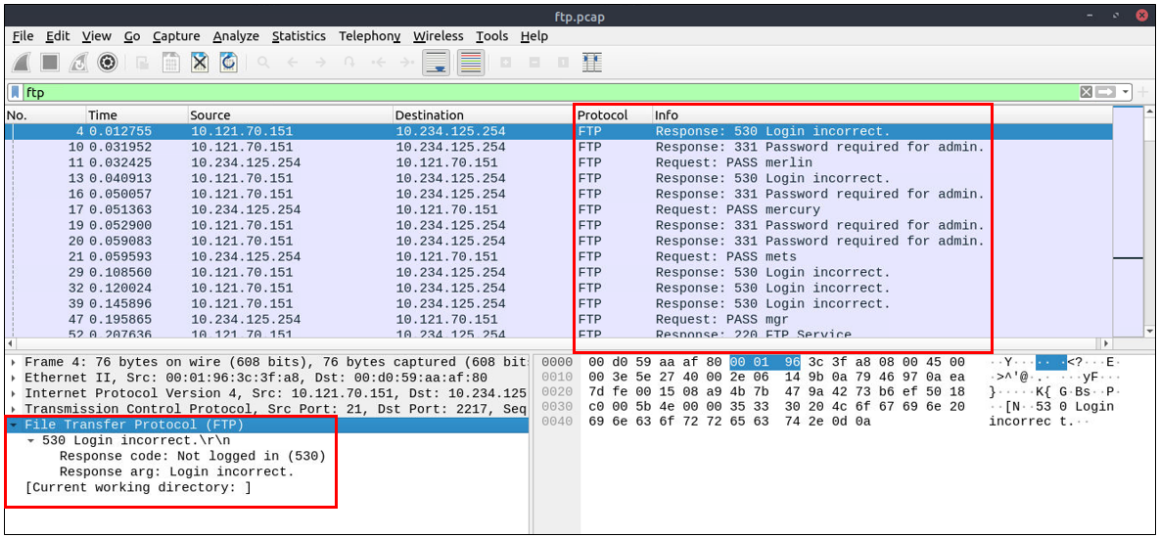
File Transfer Protocol (FTP) is designed to transfer files with ease, so it focuses on simplicity rather than security. As a result of this, using this protocol in unsecured environments could create security issues like:

- 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")`



▼ 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	• `http.response.code ==`

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.	`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"`

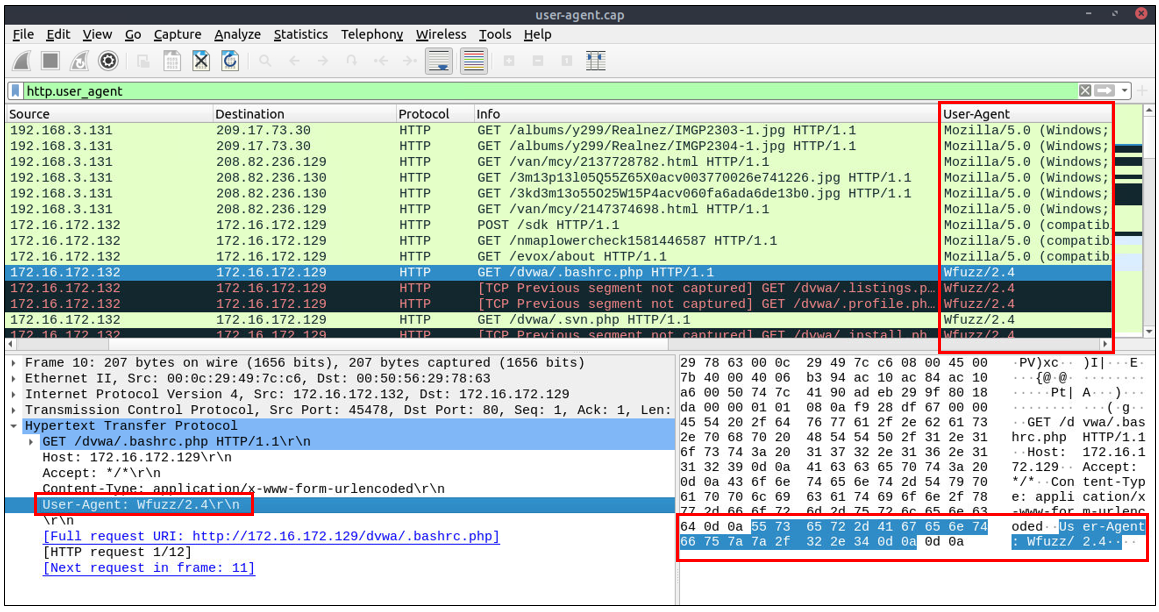
User Agent Analysis

- 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")`



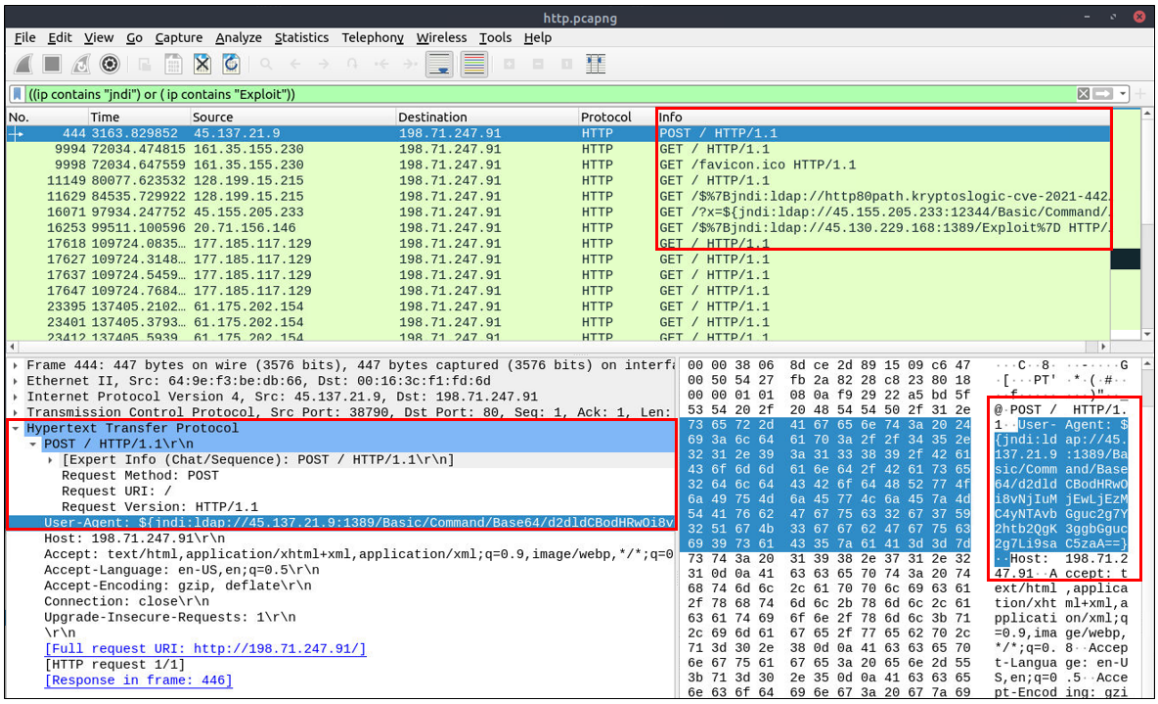
Log4j Analysis

- Log4j Analysis refers to the process of examining and assessing the logs generated by the Apache Log4j framework in order to identify security vulnerabilities, operational issues, or other relevant insights within an application or system.

- 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 "==")`



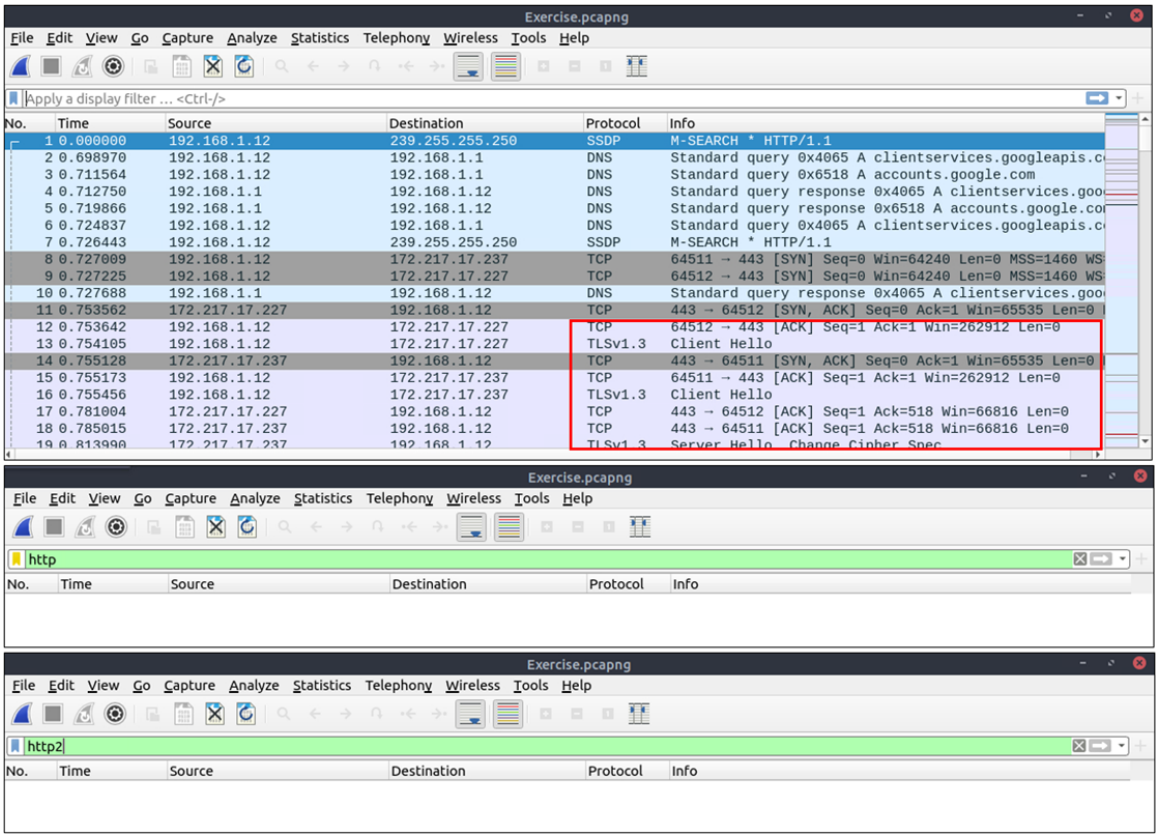
▼ Encrypted Protocol Analysis: Decrypting HTTPS

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.

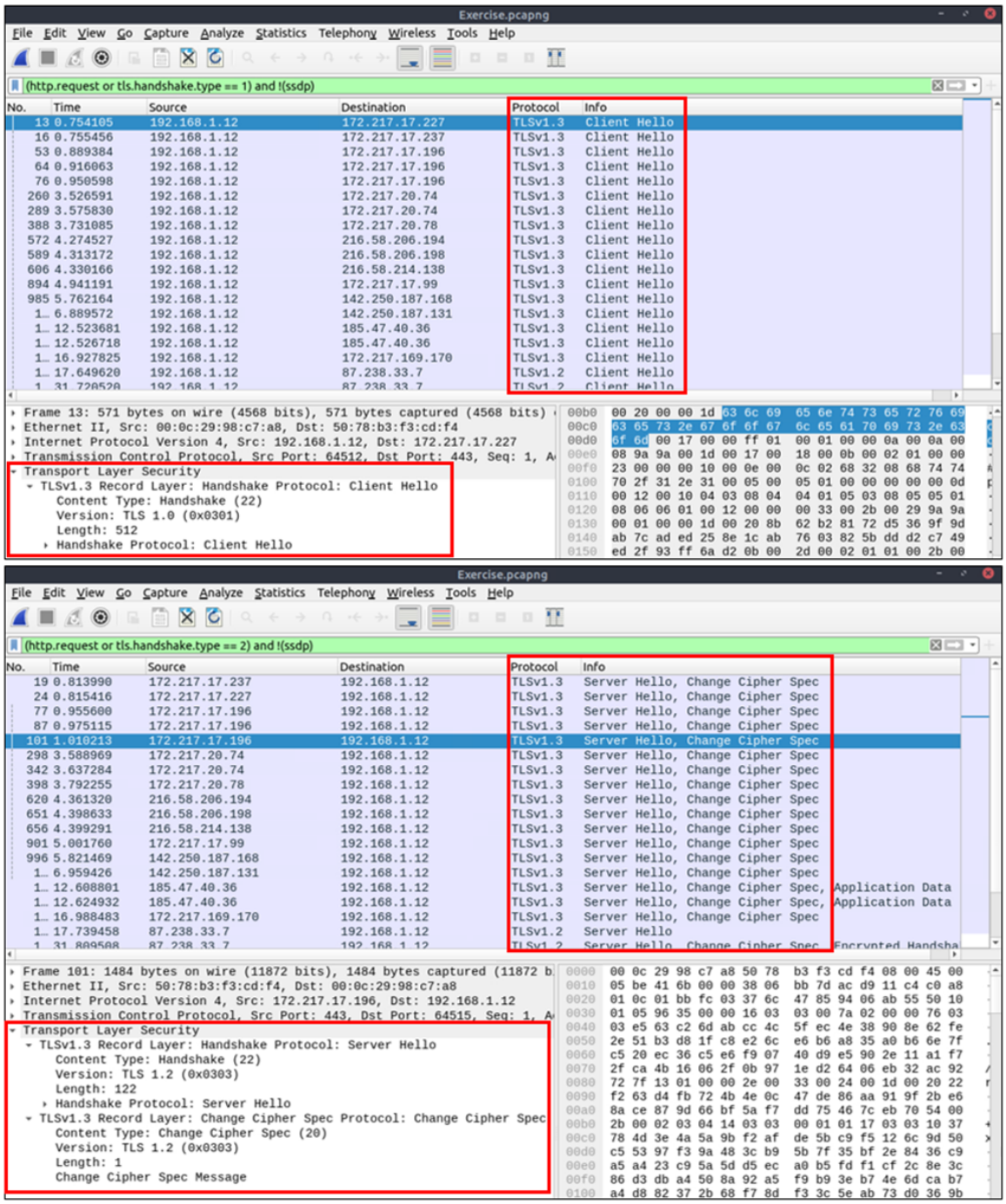


Additional information for HTTPS :

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	• `http.request` • `tls` • `tls.handshake.type == 1` •

Protocol (SSDP) **Note:** SSDP is a network protocol that provides **advertisement and discovery of network services.**	`tls.handshake.type == 2` • `ssdp`

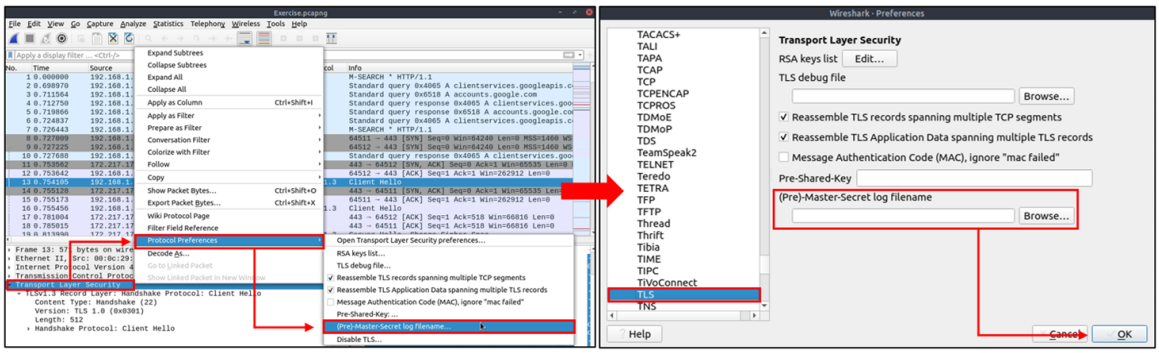
- Similar to the TCP three-way handshake process, the TLS protocol has its handshake process.
- The first two steps contain "**Client Hello**" and "**Server Hello**" messages.
- The given filters show the initial hello packets in a capture file. These filters are helpful to spot which IP addresses are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



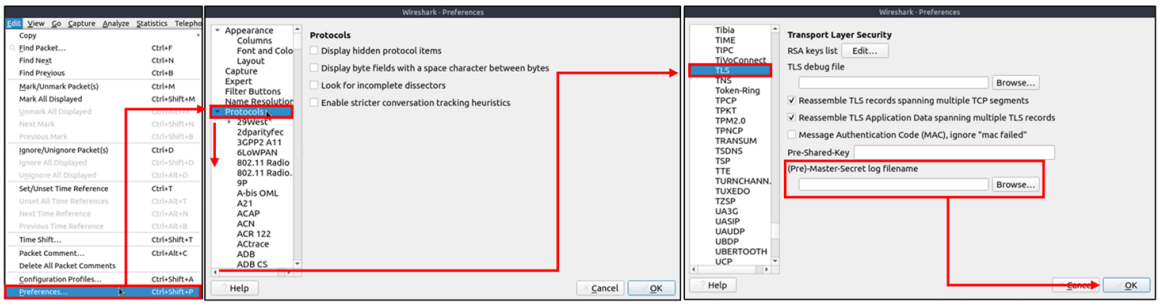
- An **encryption key log file** is a text file that contains unique key pairs to decrypt the encrypted traffic session.
- These key pairs are automatically created (**per session**) when a connection is established with an SSL/TLS-enabled webpage.
- As these processes are all accomplished in the browser, 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.

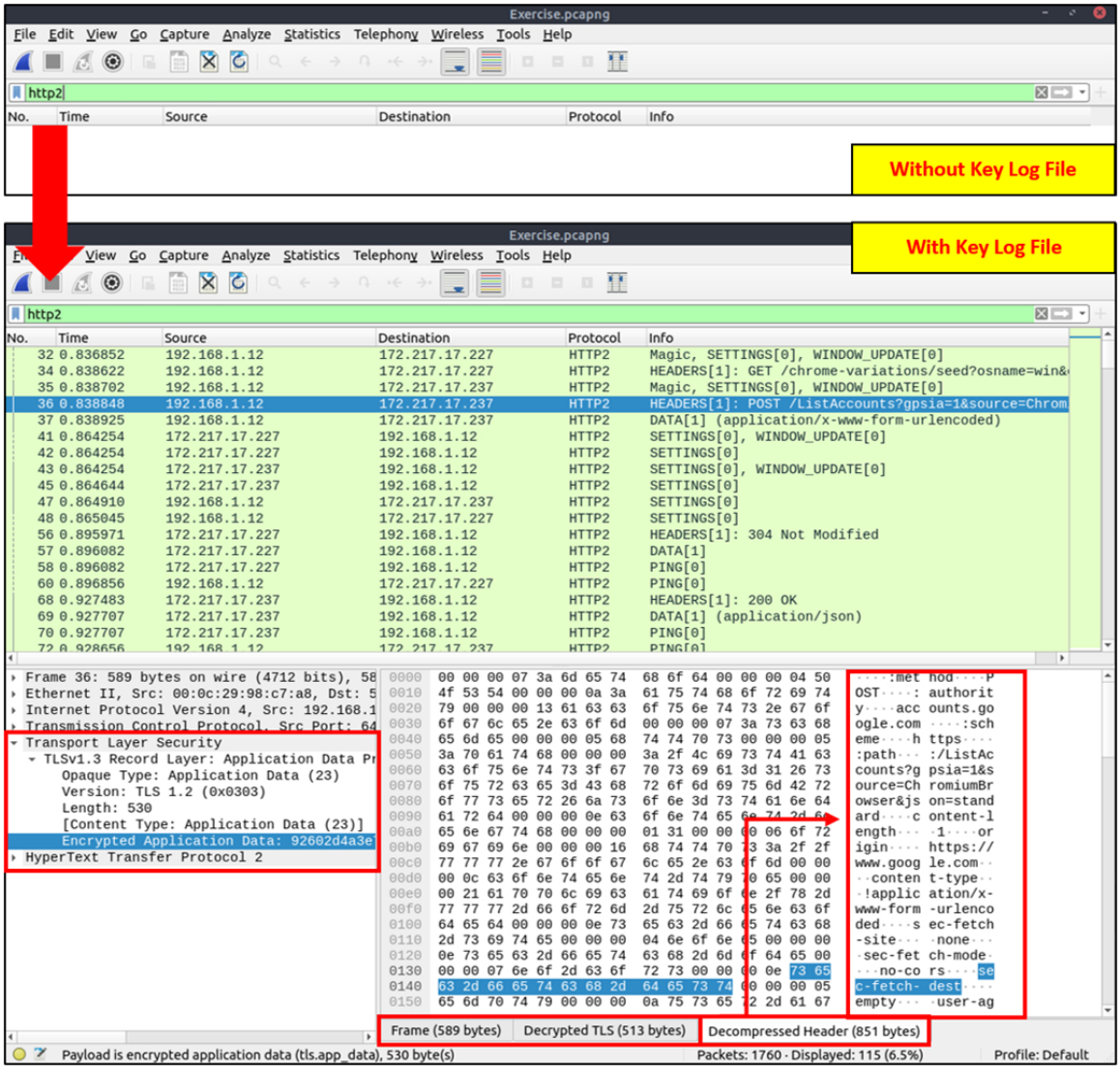
Adding key log files with the "right-click" menu:



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



Viewing the traffic with/without the key log files:



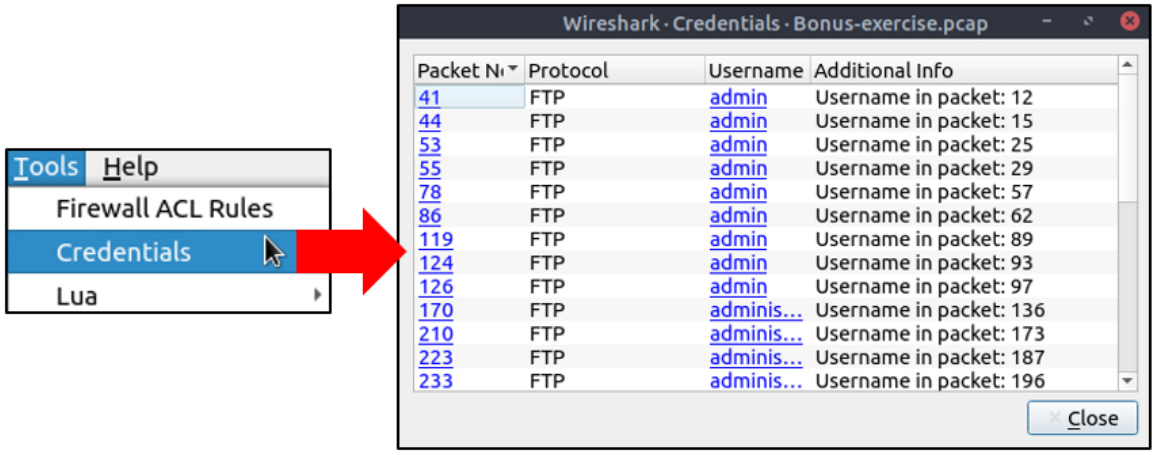
The above image shows that the traffic details are visible after using the key log file. Note that the packet details and bytes pane provides the data in different formats for investigation. Decompressed header info and HTTP2 packet details are available after decrypting the traffic. Depending on the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

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

▼ Hunt Cleartext Credentials!

Bonus: Hunt Cleartext Credentials!

- Some Wireshark dissectors (FTP, HTTP, IMAP, pop and SMTP) are programmed to extract cleartext passwords from the capture file. You can view detected credentials using the **"Tools --> Credentials"** menu. This feature works only after specific versions of Wireshark (v3.1 and later). Since the feature works only with particular protocols, it is suggested to have manual checks and not entirely rely on this feature to decide if there is a cleartext credential in the traffic.
- Once you use the feature, it will open a new window and provide detected credentials. It will show the packet number, protocol, username and additional information. This window is clickable; clicking on the packet number will select the packet containing the password, and clicking on the username will select the packet containing the username info. The additional part prompts the packet number that contains the username.



▼ Actionable Results!

- Wireshark is not all about packet details; it can help you to create **firewall rules** ready to implement with a couple of clicks.
- You can create firewall rules by using the **"Tools --> Firewall ACL Rules"** menu. Once you use this feature, it will open a new window and provide a combination of rules (IP, port and MAC address-based) for different purposes. Note that these rules are generated for implementation on an outside firewall interface.

Currently, Wireshark can create rules for:

- Netfilter (iptables)
- Cisco IOS (standard/extended)

- IP Filter (ipfilter)
- IPFirewall (ipfw)
- Packet filter (pf)
- Windows Firewall (netsh new/old format)