GOVERNMENT OF INDIA

MINISTRY OF SKILL DEVELOPMENT & ENTREPRENEURSHIP

DIRECTORATE GENERAL OF TRAINING

NATIONAL SKILL TRAINING INSTITUTE

NSTI (W) Salt Lake, Kolkata-700091

CERTIFICATE

This is to certify that following trainees have completed their project titled

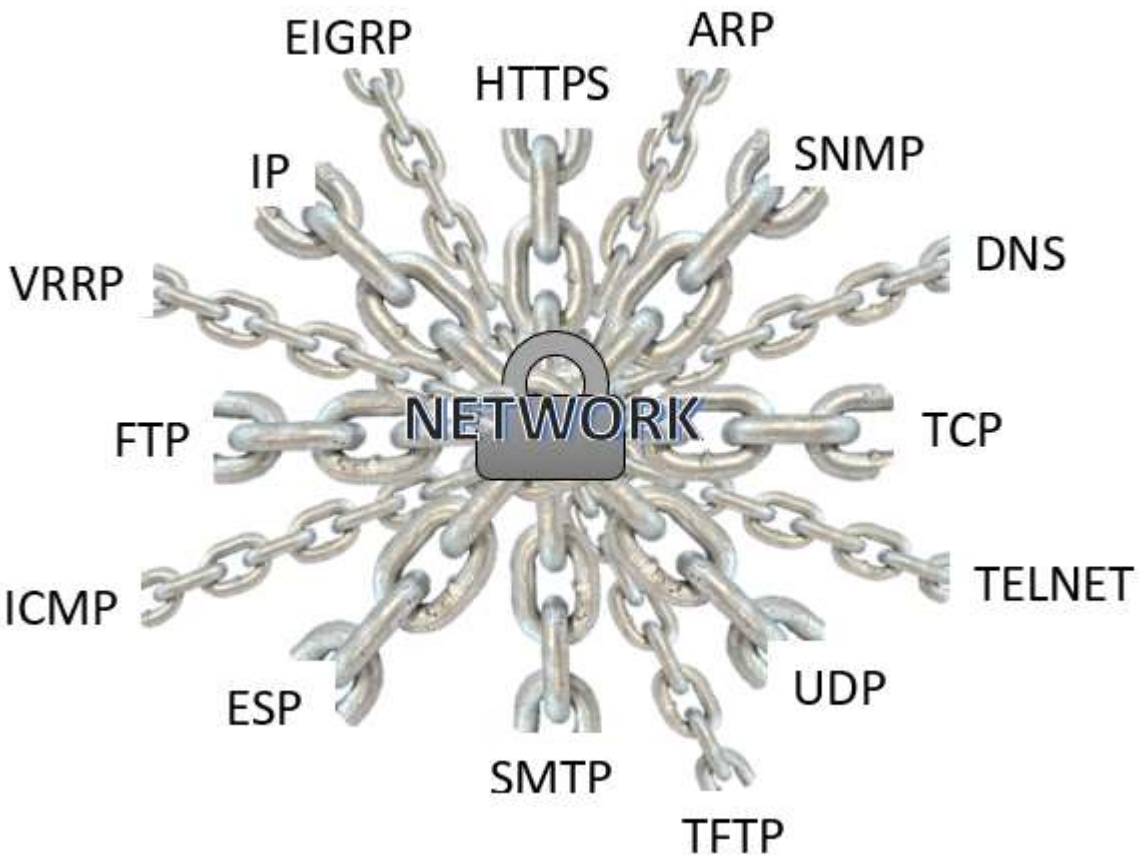
"NSTI Smart Cloud Campus"

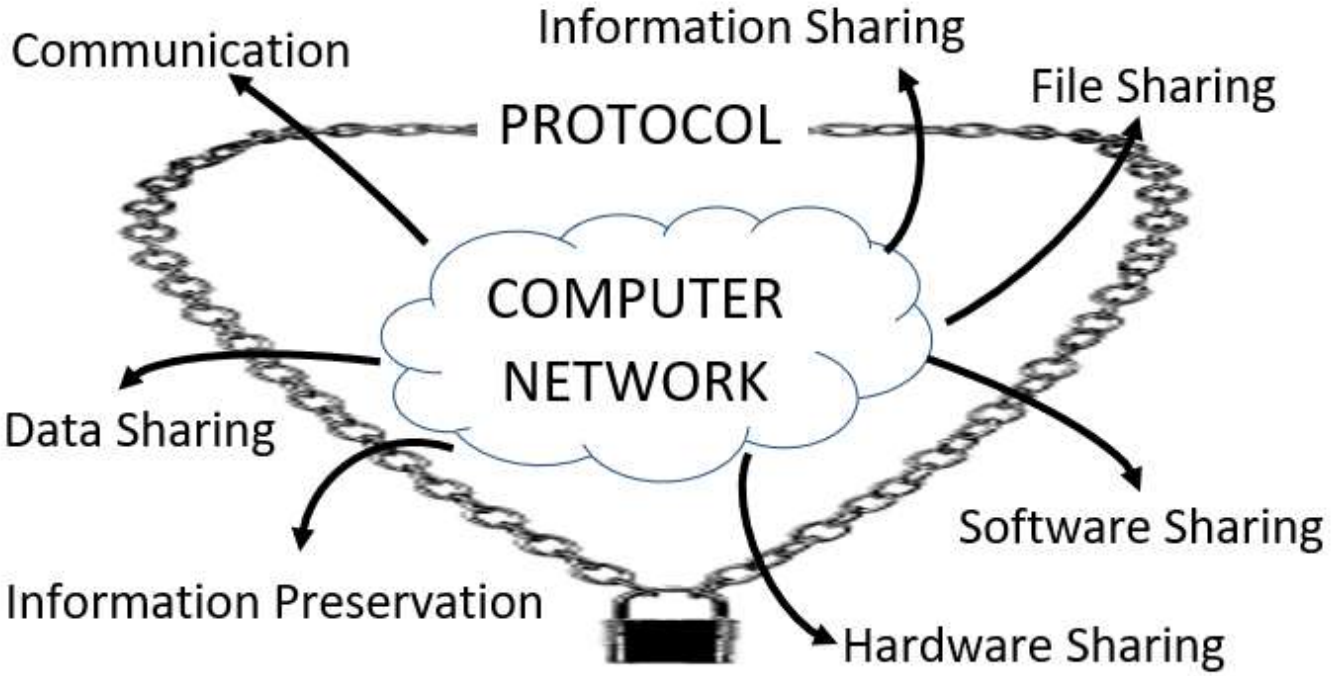
For IBM Program – IT, Networking and Cloud (Technical Diploma)

ROLL NO	NAME
ADIT-19AU03241	ERAM PERWEZ
ADIT-19AU03683	PUSHPA KUMARI
ADIT-19AU00243	SUDHA KUMARI DAS
ADIT-19AU00945	PARAMITA DEWEN

Miss. Arpita Roy	Mr. K.L.Kuli	Mr.G.C. Ramamurthy	Mr. Sarbojit Neogi
IBM Faculty	ADIT Director	Head of Office/Principal	Section In-charge

NETWORK PROTOCOL ANALYZER




Communication
Information Sharing
File Sharing
PROTOCOL
COMPUTER
NETWORK
Data Sharing
Software Sharing
Information Preservation
Hardware Sharing

ACKNOWLEDGEMENT

It is great pleasure for us to undertaken this project. We feel highly doing this project entitled "NETWORK PROTOCOL ANALYZER".

I would like to express profound gratitude to our sir Mr. Abhishek Raman for his invaluable support, encouragement, supervision and useful suggestion throughout this project. Our sincere appreciation also goes to our principal sir Shri G.C.Ramamurthy and Mr.Sarbojit Neogi for their care and moral support.

I would like to thanks whole group members Pushpa Kumari, Sudha Kumari Das and Paramita Dewan helped a lot in finishing this project within the limited time. It helped us increase our knowledge and skills.

DATE: 02/07/2021

Eram Perwez (Group A Leader)

Pushpa Kumari

Sudha Kumari Das

Paramita Dewan

TABLE OF CONTENTS

Introduction of Wireshark

A **packet analyzer** or **packet sniffer** is a computer program or computer hardware such as a packet capture appliance, that can intercept and log traffic that passes over a computer network or part of a network. **Packet capture** is the process of intercepting and logging traffic. As data streams flow across the network, the analyzer captures each packet and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyzes its content according to the appropriate RFC or other specifications.

A packet analyzer used for intercepting traffic on wireless networks is known as a **wireless analyzer** or **WIFI analyzer**. A packet analyzer can also be referred to as a network analyzer or protocol analyzer though these terms also have other meanings.

Software Requirement to Analyze Network Traffic

❋ Open-Source Software

Wireshark is an open-source software project, and is released under the GNU General Public License (GPL). You can freely use Wireshark on any number of computers you like, without worrying about license keys or fees or such. In addition, all source code is freely available under the GPL. Because of that, it is very easy for people to add new protocols to Wireshark, either as plugins, or built into the source, and they often do!

❋ Wireshark (Software)

In this project we are using Wireshark Software to analyze the network traffic over internet.

Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible.

You could think of a network packet analyzer as a measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable (but at a higher level, of course).

In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, that has changed. Wireshark is available for free, is open source, and is one of the best packet analyzer available today.

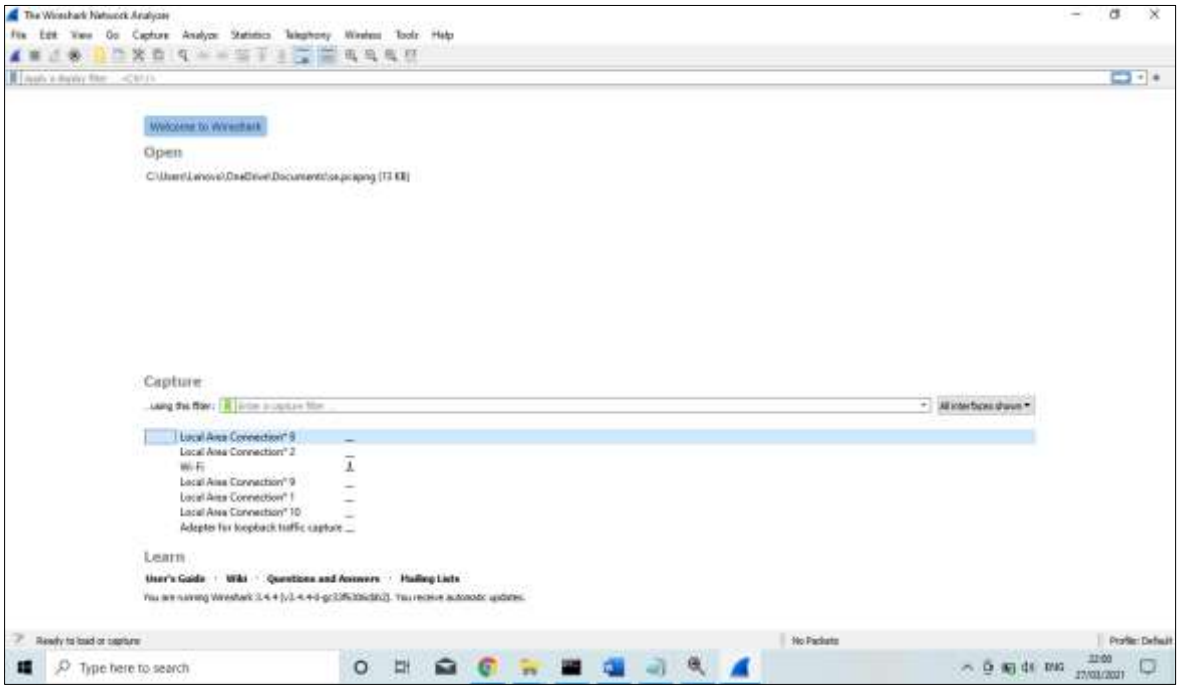
Importance of Wireshark

Here are some reasons people use Wireshark:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems Pg-1
- QA engineers use it to verify network applications
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

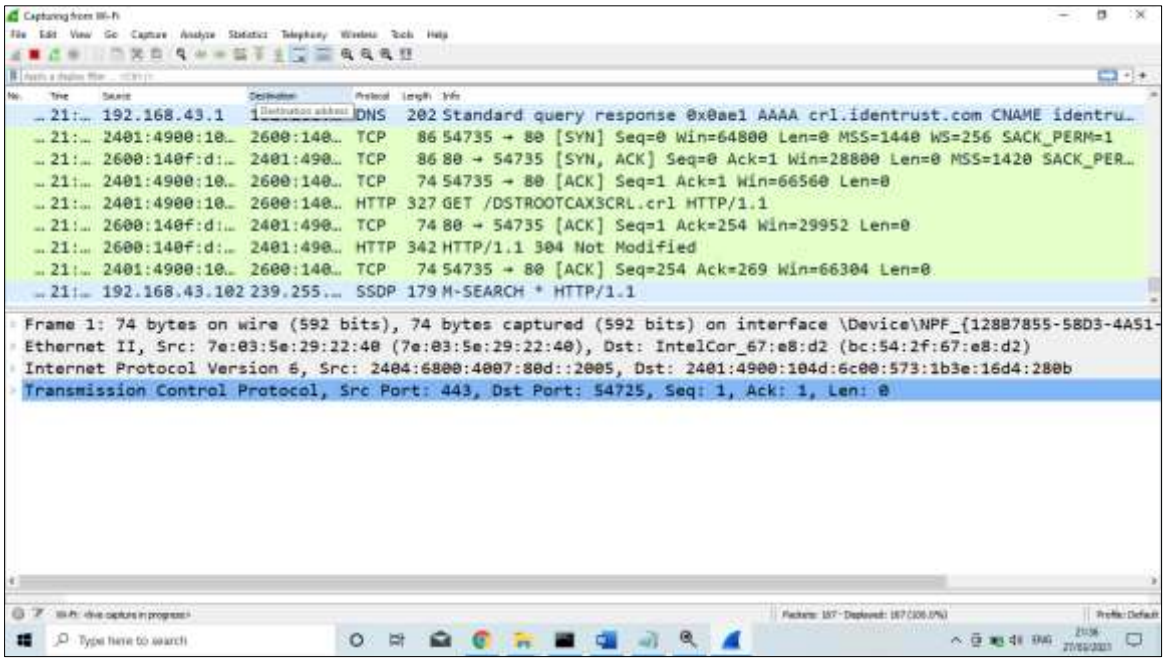
Wireshark can also be helpful in many other situations.

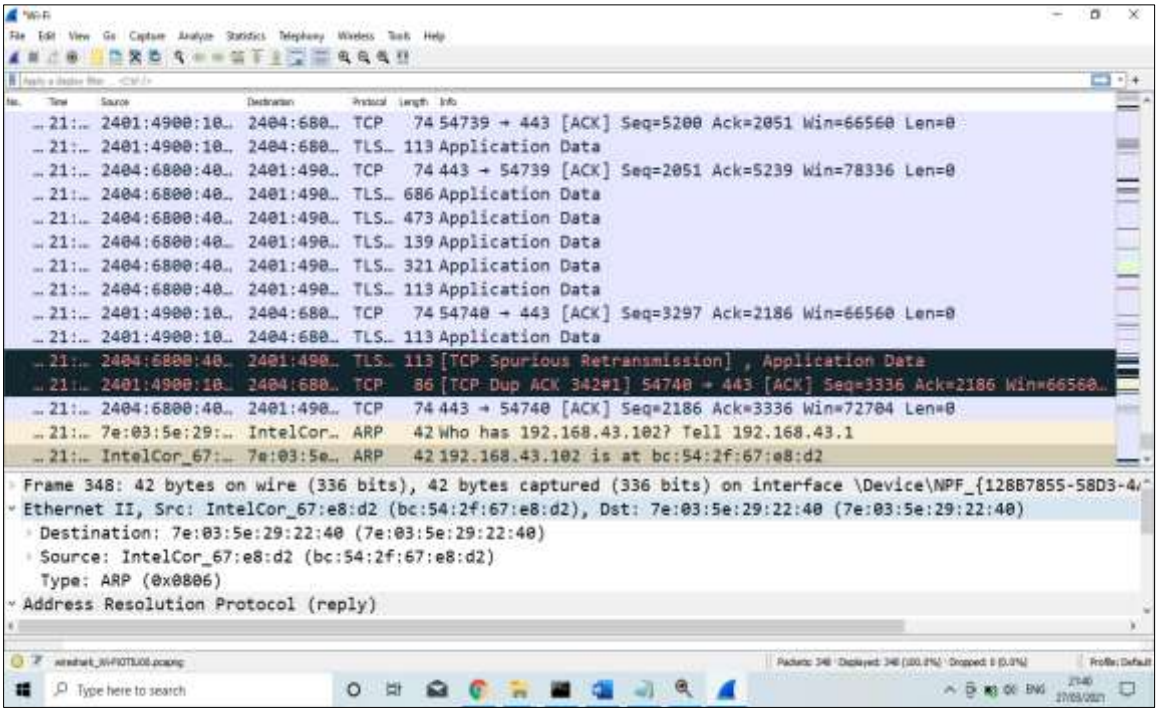
Interface of Wireshark List



❋ Wireshark captures packets

Here we choose Ethernet Adapter because we connect with Wi-Fi.





❋ Types of Network Protocol

There are various types of protocols that support a major and compassionate role in communicating with different devices across the network. These are:

1. Transmission Control Protocol (**TCP**)
2. Internet Protocol (**IP) (IPv4/IPv6)**
3. User Datagram Protocol (**UDP**)
4. Post office Protocol (**POP**)
5. Simple mail transport Protocol (**SMTP**)
6. File Transfer Protocol (**FTP**)
7. Hyper Text Transfer Protocol (**HTTP**)
8. Hyper Text Transfer Protocol Secure (**HTTPS)**
9. Telnet
10. Simple Message Protocol (**SMP**)
11. Address Resolution Protocol (**ARP**)
12. Reverse Address Resolution Protocol (**RARP**)
13. Internet Group Management Protocol (**IGMP**)
14. Gateway-to-Gateway Protocol (**GGP**)
15. Internet Stream Protocol **(ST**)
16. Exterior Gateway Protocol (**EGP**)
17. Interior Gateway Protocol (**IGP**)
18. Datagram Delivery Protocol (**DDP**)
19. Generic Routing Encapsulation (**GRE**)
20. Transport Layer Security Protocol (**TLS**)

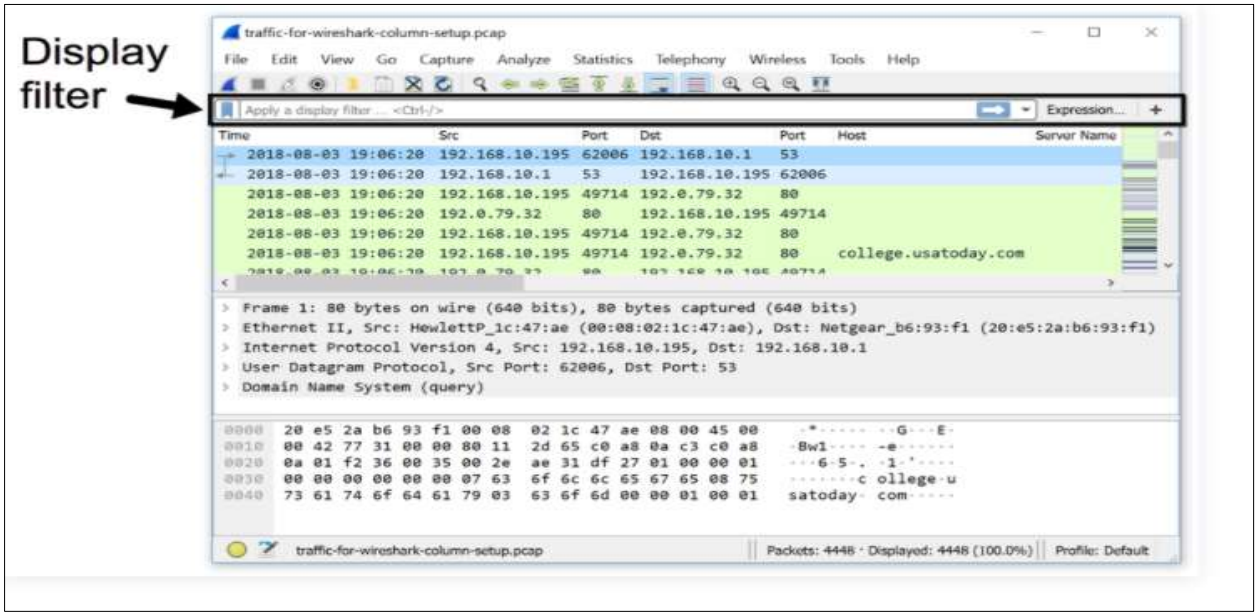
❋ Indicators of Infection Traffic

This tutorial uses examples of Windows infection traffic from commodity malware distributed through mass-distribution methods like malicious spam (malspam) or web traffic. These infections can follow many different paths before the malware, usually a Windows executable file, infects a Windows host.

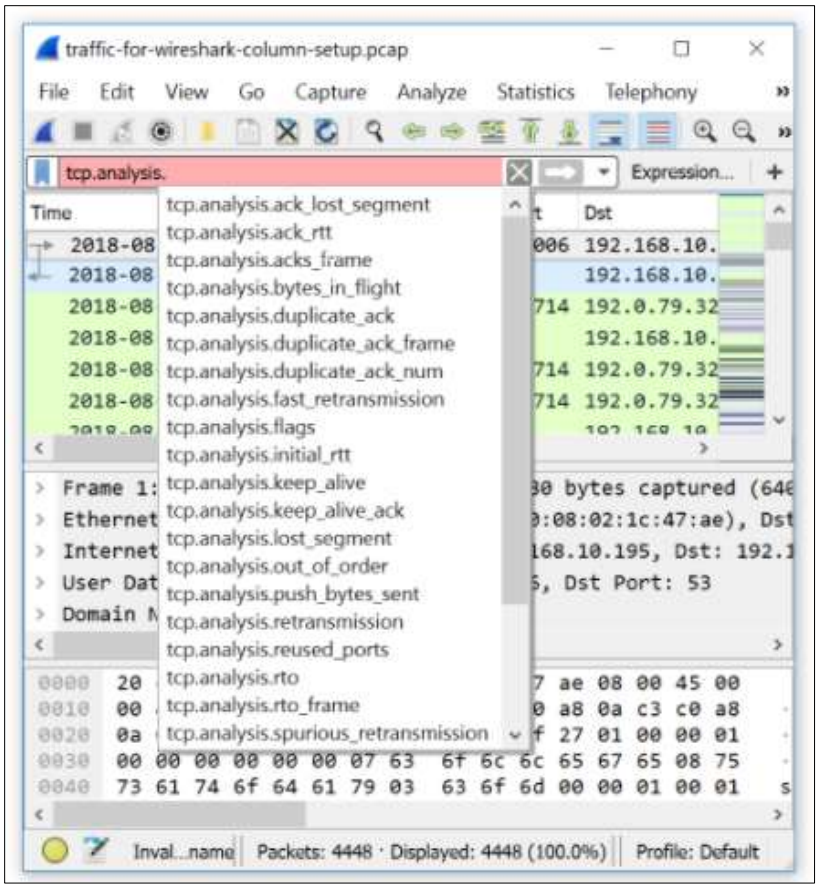
Indicators consist of information derived from network traffic that relates to the infection. These indicators are often referred to as Indicators of Compromise (IOCs). Security professionals often document indicators related to Windows infection traffic such as URLs, domain names, IP addresses, protocols, and ports. Proper use of the Wireshark display filter can help people quickly find these indicators.

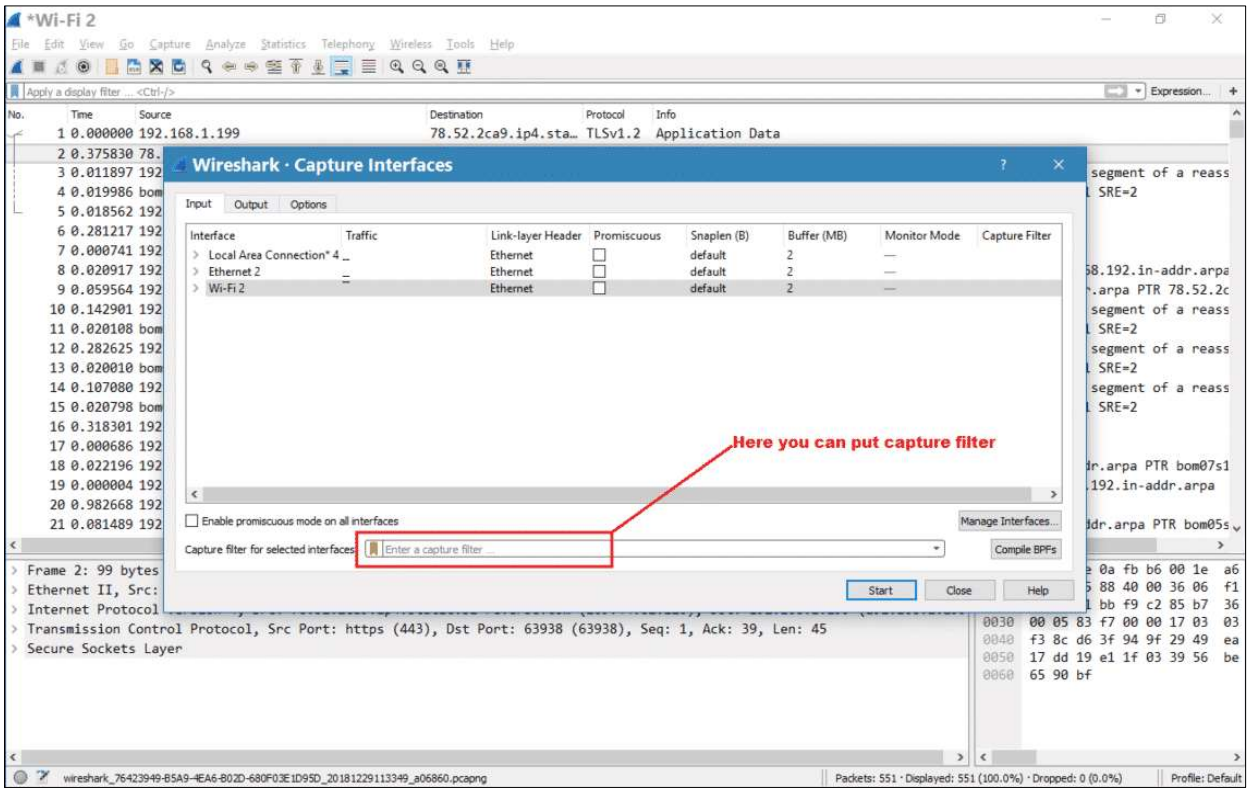
❋ The Wireshark Display Filter

Wireshark's display filter a bar located right above the column display section. This is where you type expressions to filter the frames, IP packets, or TCP segments that Wireshark displays from a cap.

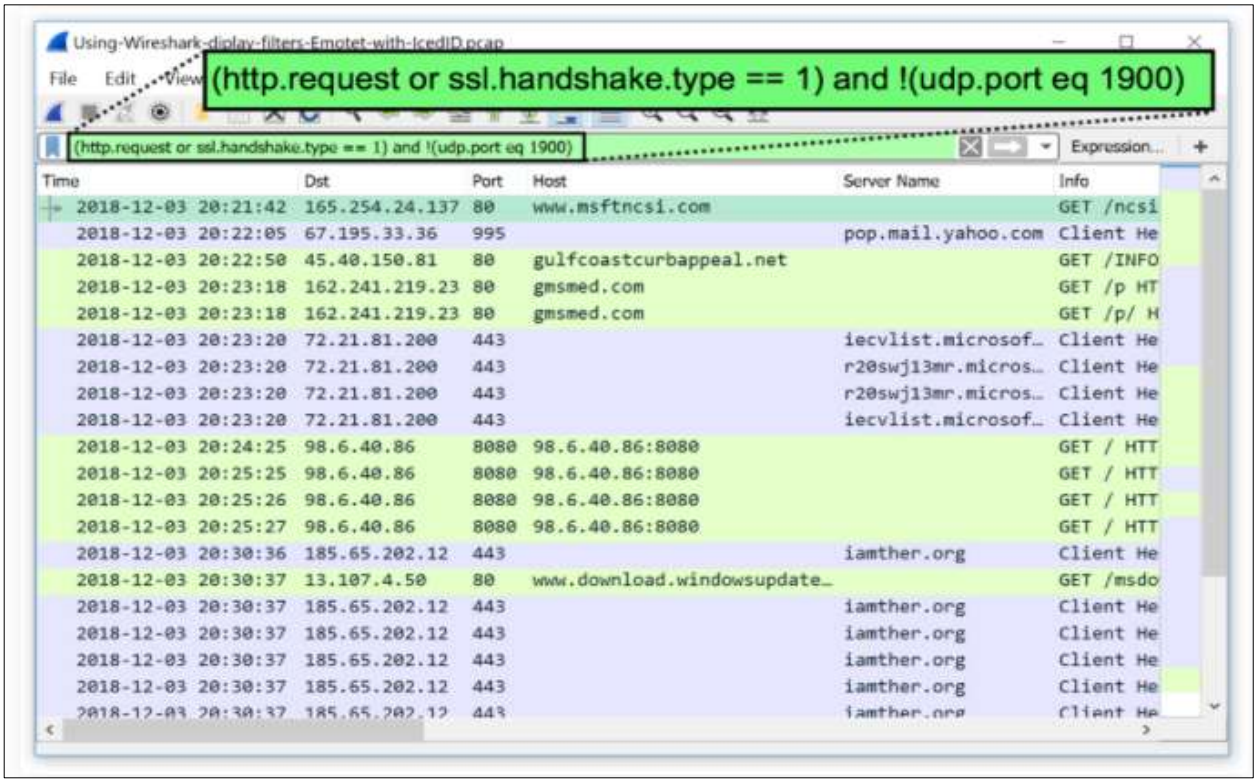


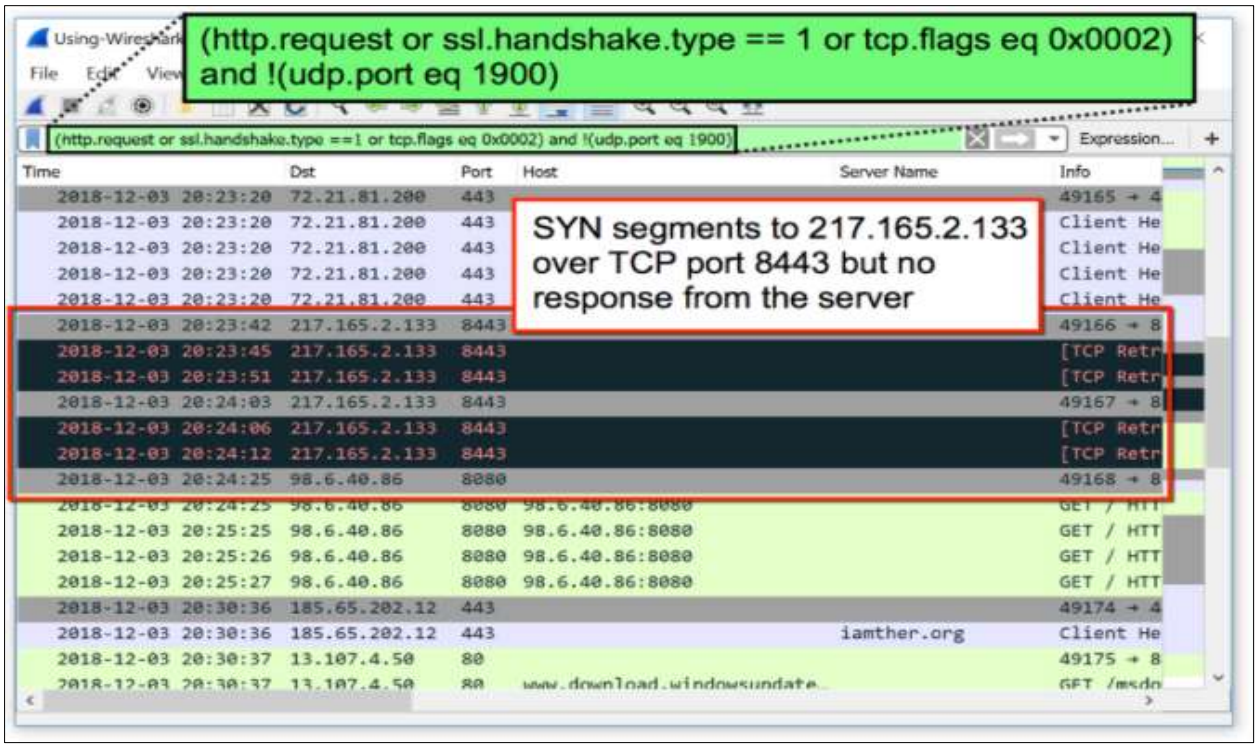
If you type any types of protocol in the display filter or capture filter, Wireshark offers a list of suggestions based on the text you have typed. While the display filter bar remains red, the expression is not yet accepted. If the display filter bar turns green, the expression has been accepted and should work properly. If the display filter bar turns yellow, the expression has been accepted, but it will probably not work as intended.





Filters for Web-Based Infection Traffic





There are two types of filter in Wireshark

1. Display filter
2. Capture filter

Display Filters

- Wireshark uses display filters for general packet filtering while viewing and for its ColoringRules.
- The basics and the syntax of the display filters are described in the User's Guide.
- The master list of display filter protocol fields can be found in the display filter reference.
- If you need a display filter for a specific protocol, have a look for it at the Protocol Reference.

Examples:-

Show only **SMTP** (port 25) and **ICMP** traffic:

- tcp.port eq 25 or icmp

Show only traffic in the **LAN** (192.168.x.x), between **workstations** and **servers** -- **no Internet**:

- ip.src==192.168.0.0/16 and ip.dst==192.168.0.0/16

TCP buffer full -- *Source is instructing Destination to stop sending data*

- tcp.window_size == 0 && tcp.flags.reset != 1

Filter on Windows -- Filter out noise, while watching Windows Client - DC exchanges

- smb || nbns || dcerpc || nbss || dns

Sasser worm: --What sasser really did--

- ls_ads.opnum==0x09

Match packets containing the (arbitrary) 3-byte sequence 0x81, 0x60, 0x03 at the beginning of the **UDP** payload, skipping the 8-byte UDP header. Note that the values for the byte sequence implicitly are in hexadecimal only. *(Useful for matching homegrown packet protocols.)*

- udp[8:3]==81:60:03

The "slice" feature is also useful to filter on the vendor identifier part (OUI) of the MAC address, see the **Ethernet** page for details. Thus you may restrict the display to only packets from a specific device manufacturer. E.g. for DELL machines only:

- eth.addr[0:3]==00:06:5B

It is also possible to search for characters appearing anywhere in a field or protocol by using the contains operator.

Match packets that contains the 3-byte sequence 0x81, 0x60, 0x03 anywhere in the **UDP** header or payload:

- udp contains 81:60:03

Match packets where **SIP** To-header contains the string "a1762" anywhere in the header:

- sip.To contains "a1762"

The matches, or ~, operator makes it possible to search for text in string fields and byte sequences using a regular expression, using Perl regular expression syntax. Note: Wireshark needs to be built with libpcre in order to be able to use the matches operator.

Match **HTTP** requests where the last characters in the uri are the characters "gl=se":

- http.request.uri matches "gl=se$"

Note: The $ character is a **PCRE** punctuation character that matches the end of a string, in this case the end of http.request.uri field.

Filter by a protocol (e.g. SIP) and filter out unwanted IPs:

- ip.src != xxx.xxx.xxx.xxx && ip.dst != xxx.xxx.xxx.xxx && sip

Gotchas

Some filter fields match against multiple protocol fields. For example, "ip.addr" matches against both the **IP** source and destination addresses in the **IP** header. The same is true for "tcp.port", "udp.port", "eth.addr", and others. It's important to note that

- ip.addr == 10.43.54.65

Is equivalent to.

- ip.src == 10.43.54.65 or ip.dst == 10.43.54.65

This can be counterintuitive in some cases. Suppose we want to filter out any traffic to or from 10.43.54.65. We might try the following:

- ip.addr != 10.43.54.65

Which is equivalent to.

- ip.src != 10.43.54.65 or ip.dst != 10.43.54.65

This translates to "pass all traffic except for traffic with a source IPv4 address of 10.43.54.65 **and** a destination IPv4 address of 10.43.54.65", which isn't what we wanted.

Instead we need to negate the expression, like so:

- ! (ip.addr == 10.43.54.65)

Which is equivalent to.

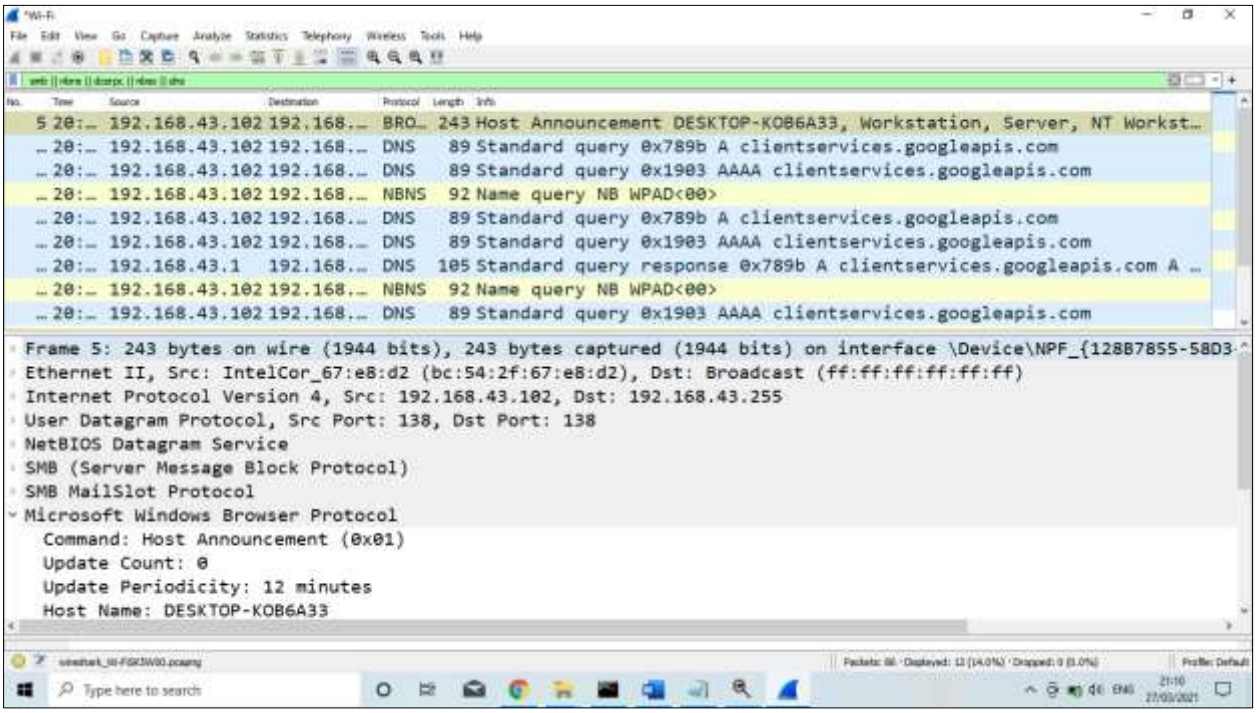
- ! (ip.src == 10.43.54.65 or ip.dst == 10.43.54.65)

This translates to "pass any traffic except with a source IPv4 address of 10.43.54.65 **or** a destination IPv4 address of 10.43.54.65", which is what we wanted.

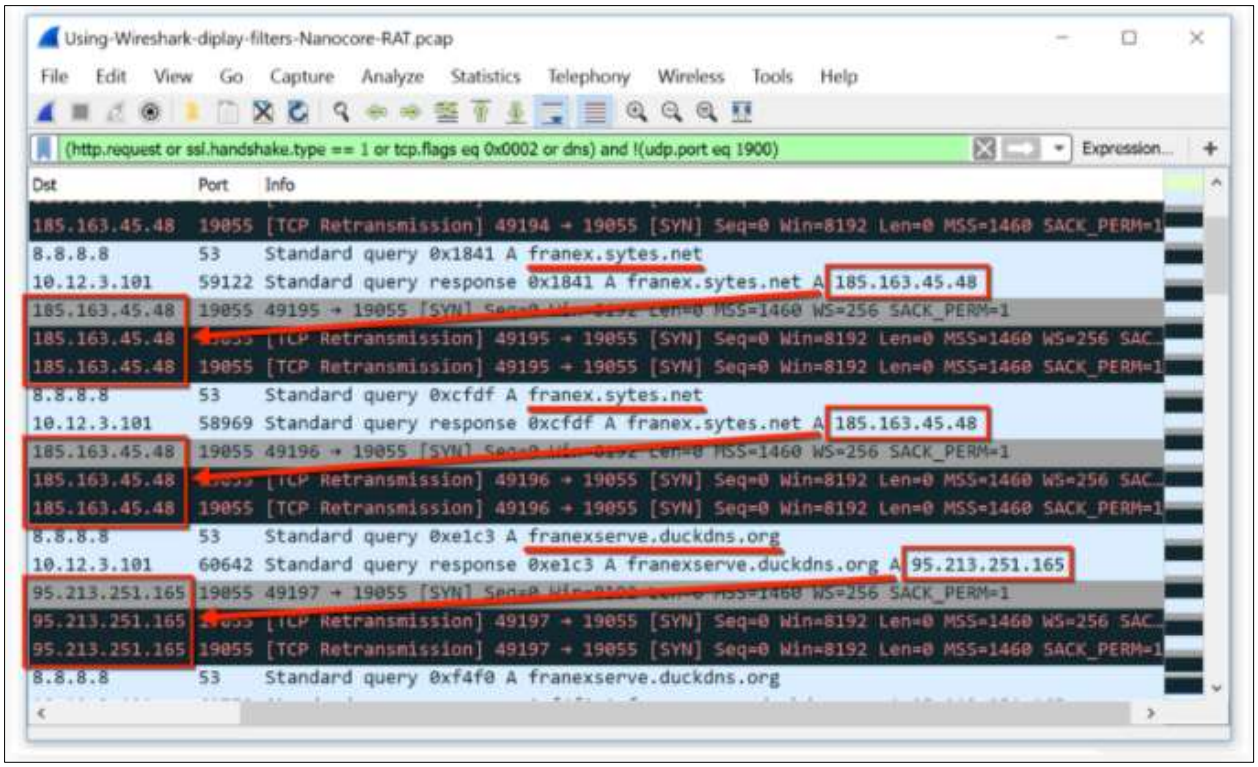
This can also happen if, for example, you have tunneled protocols, so that you might have two separate IPv4 or IPv6 layers and two separate IPv4 or IPv6 headers, or if you have multiple instances of a field for other reasons, such as multiple IPv6 "next header" fields.

If you have a filter expression of the form *name op value*, where *name* is the name of a field, *op* is a comparison operator such as == or!= or < or..., and *value* is a value against which you're comparing, it should be thought of as meaning "match a packet if there is *at least one* instance of the field named *name* whose value is (equal to, not equal to, less than, ...) *value*". The negation of that is "match a packet if there are *no* instances of the field named *name* whose value is (equal to, not equal to, less than,) *value*"; simply negating *op*, e.g. replacing == with! = or < with >=, give you another "if there is at least one" check, which is not the negation of the original check.

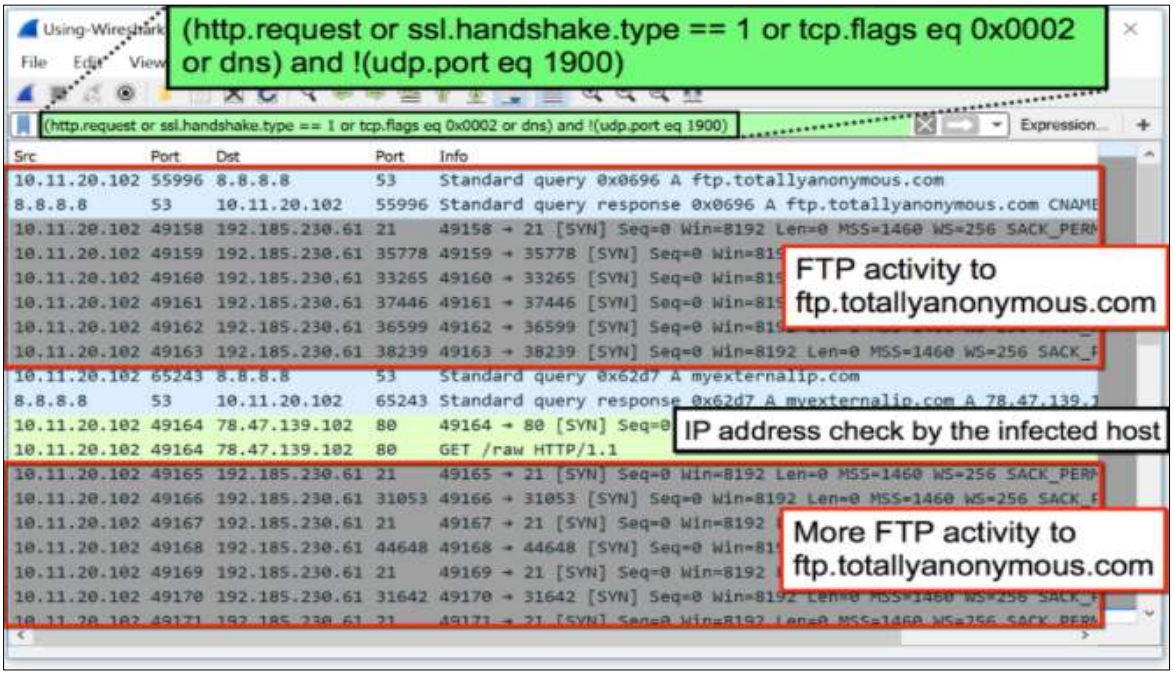
- *Filter on Windows -- Filter out noise, while watching Windows Client - DC exchanges*



- *Nanocore RAT pcap through display filter*



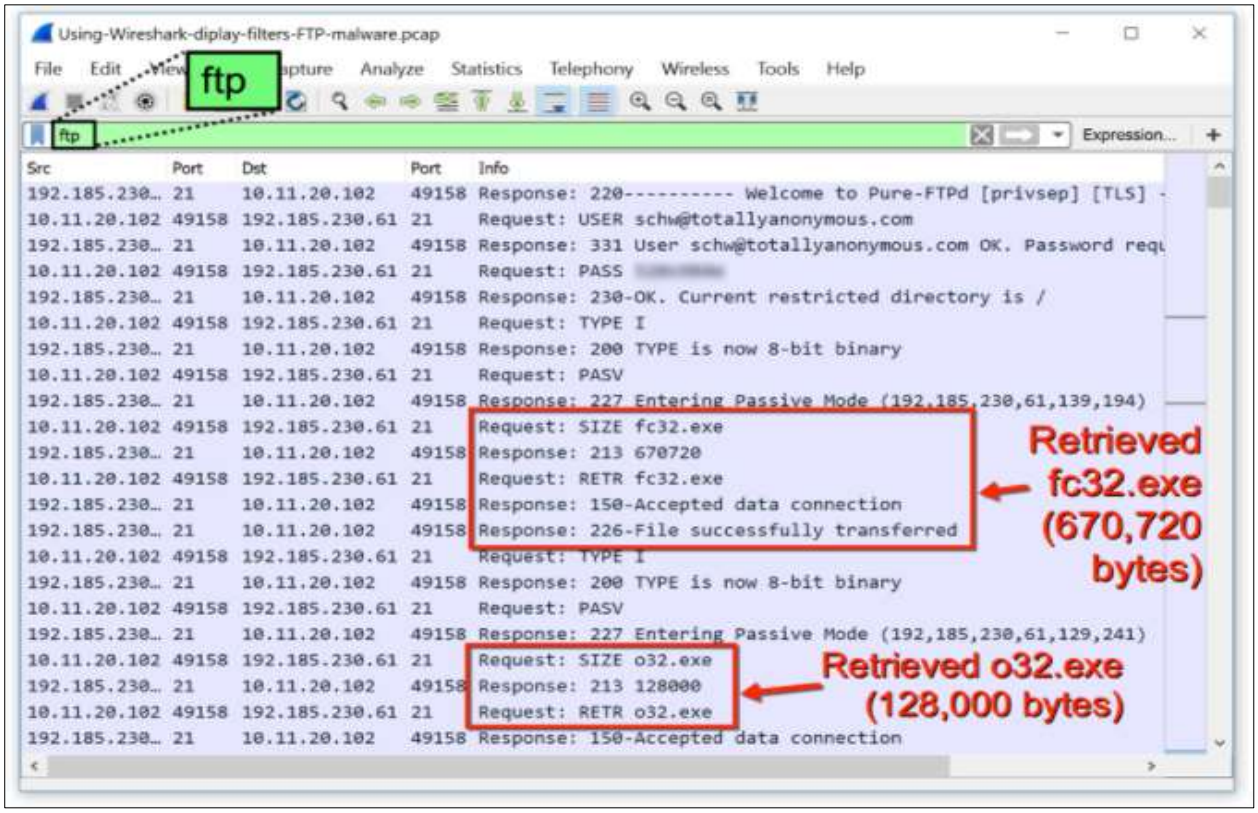
- *Multiple protocol filter through display capture*



❋ File Transfer Protocol (FTP)

The File Transfer Protocol (*FTP*) is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network. *FTP* is built on a client-server model architecture using separate control and data connections between the client and the server.

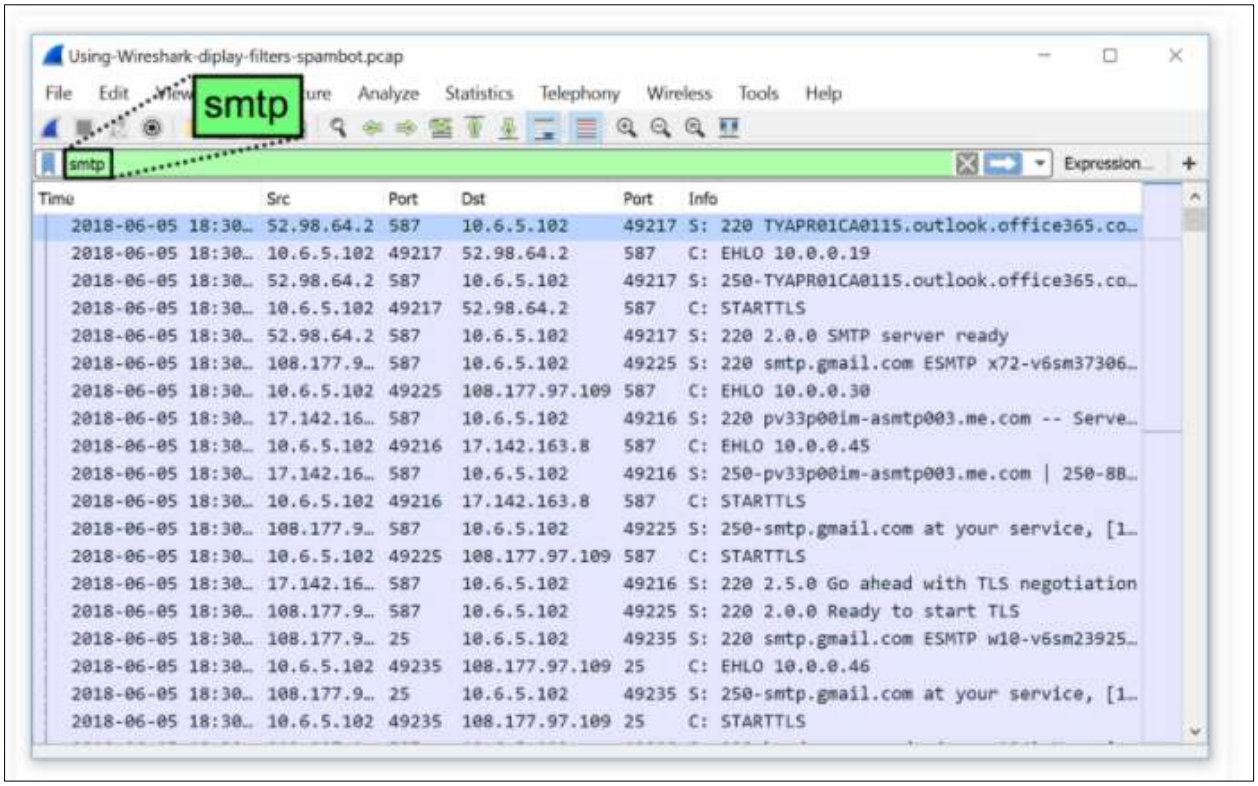
➢ *FTP protocol filter*



❋ Simple mail transport Protocol (SMTP)

SMTP stands for Simple Mail Transfer Protocol. SMTP is a set of communication guidelines that allow software to transmit an electronic mail over the internet is called Simple Mail Transfer Protocol. It is a program used for sending messages to other computer users based on e-mail addresses.

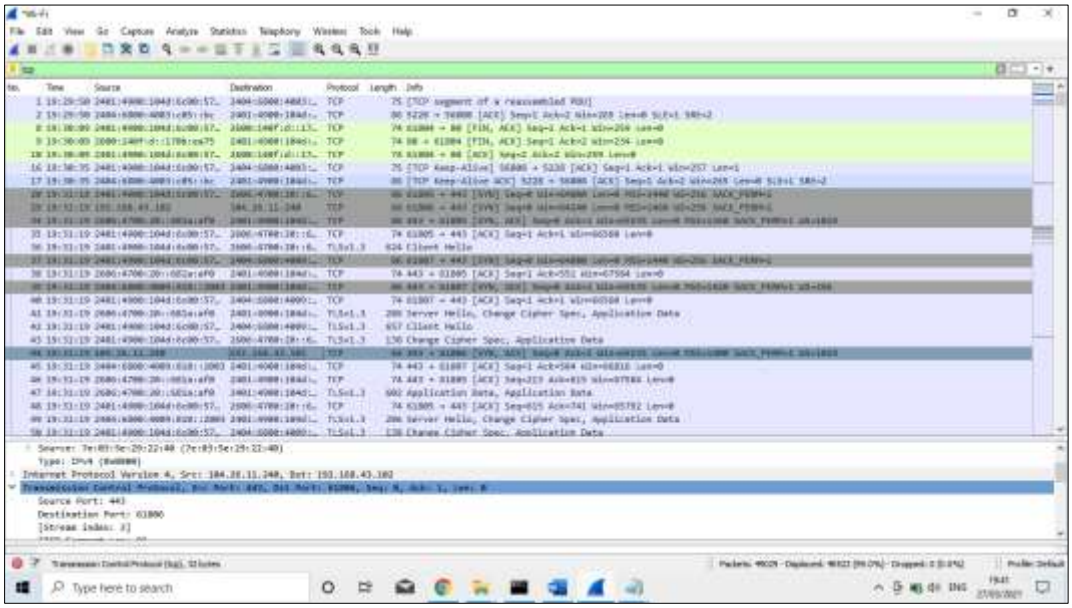
➢ *Capture SMTP Protocol*



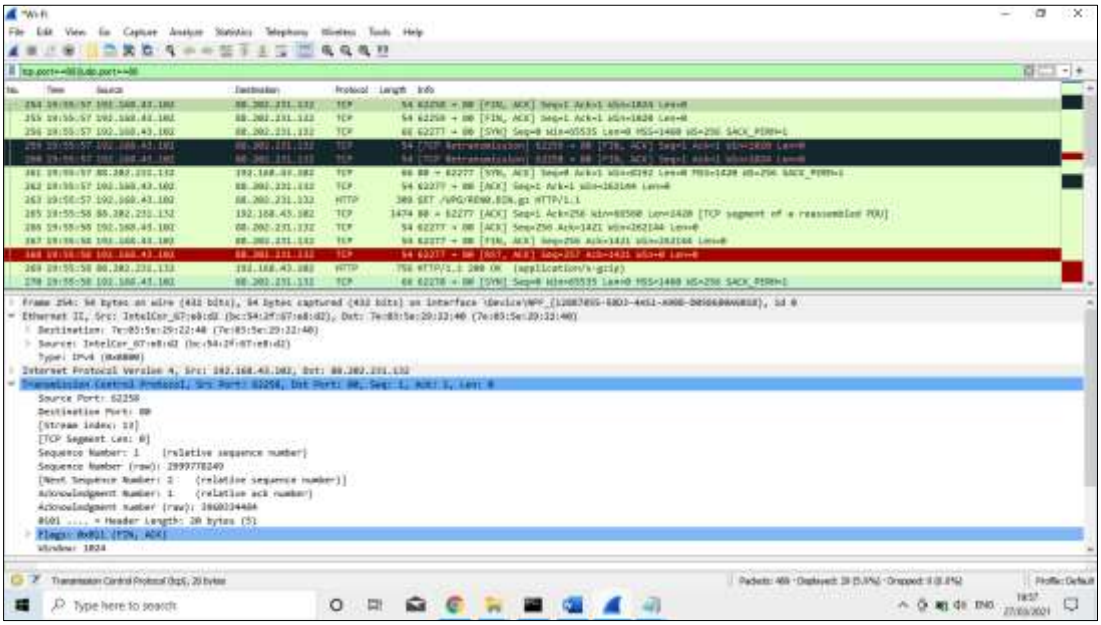
❋ Transmission Control Protocol (TCP)

TCP (Transmission Control Protocol) is a standard that defines how to establish and maintain a network conversation through which application programs can exchange data. **TCP** works with the Internet Protocol (IP), which defines how computers send packets of data to each other.

➢ *Capture Transmission Control Protocol*



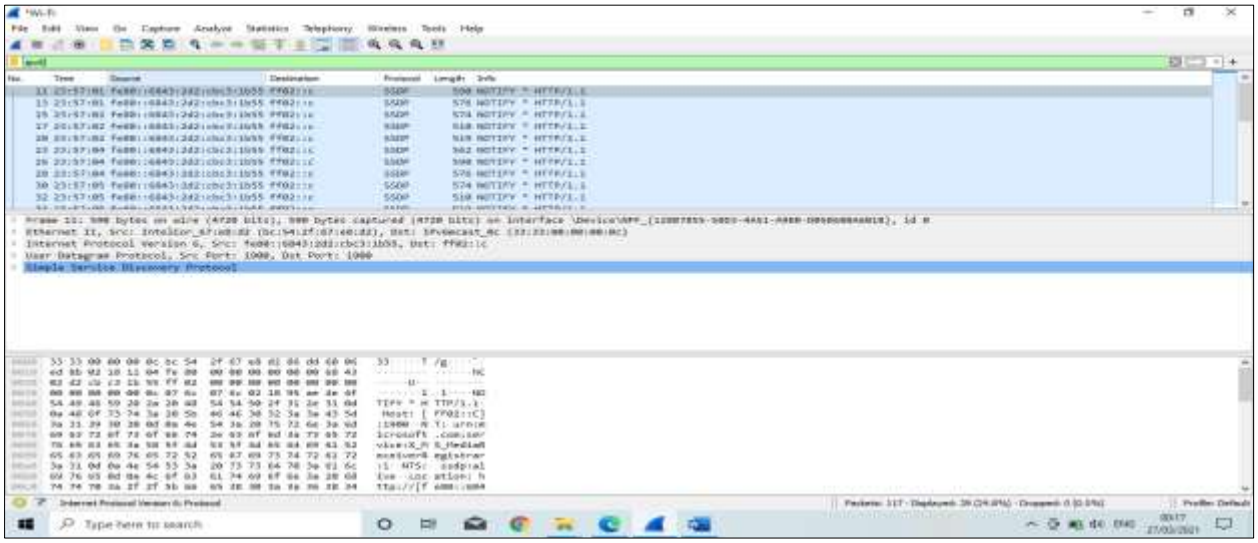
- *TCP protocol ||UDP protocol Filter with port no 80*



Internet Protocol (IP)

The Internet Protocol (IP) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet.

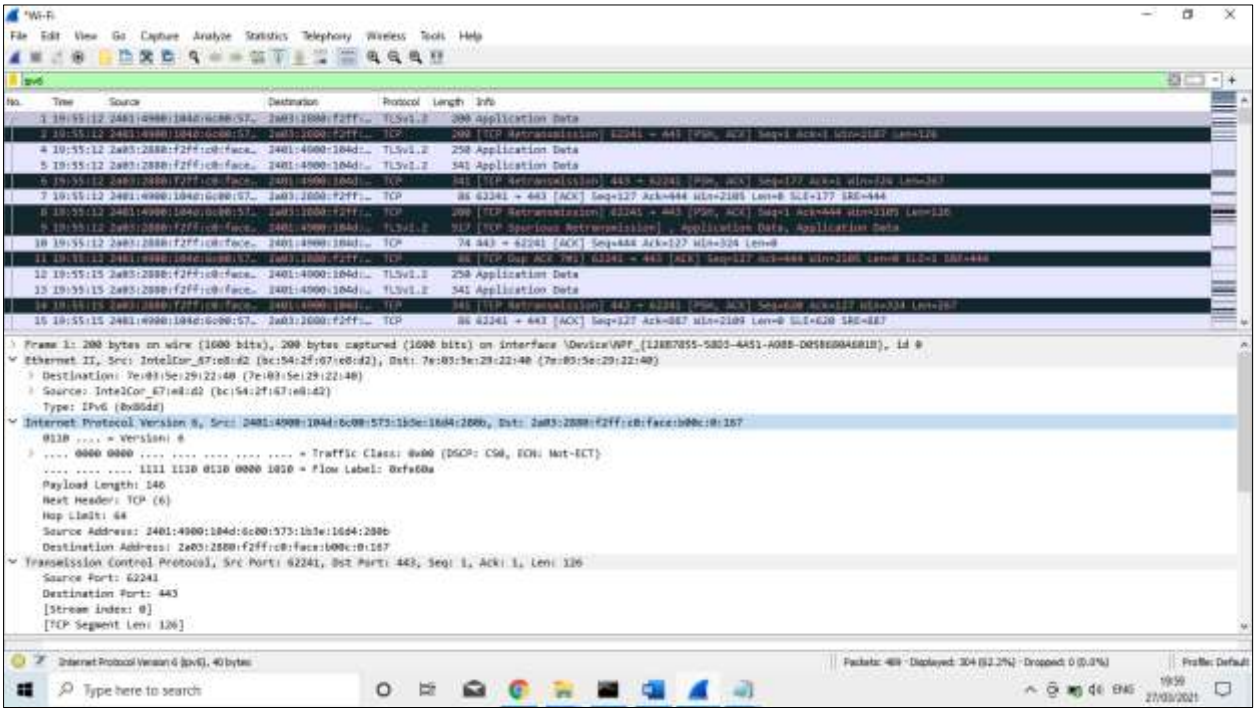
- *Capture IP*



Internet Protocol version 6 (Ipv6)

Internet Protocol version 6 (*IPv6*) is the most recent version of the Internet Protocol (IP), the communications protocol that provides an identification and location system for computers on networks and routes traffic across the Internet.

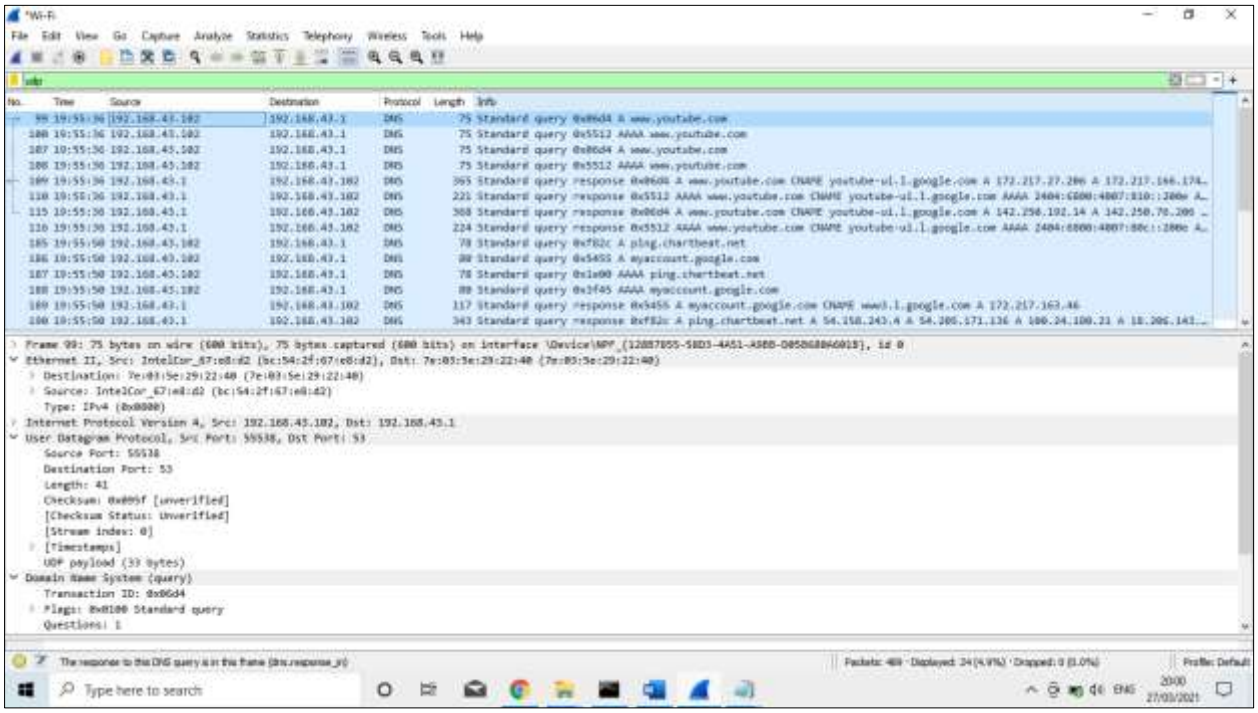
Capture Internet Protocol version 6 (Ipv6)



❋ User Datagram Protocol (UDP)

UDP (User Datagram Protocol) is a communications protocol that is primarily used for establishing low-latency and loss-tolerating connections between applications on the internet. It speeds up transmissions by enabling the transfer of data before an agreement is provided by the receiving party.

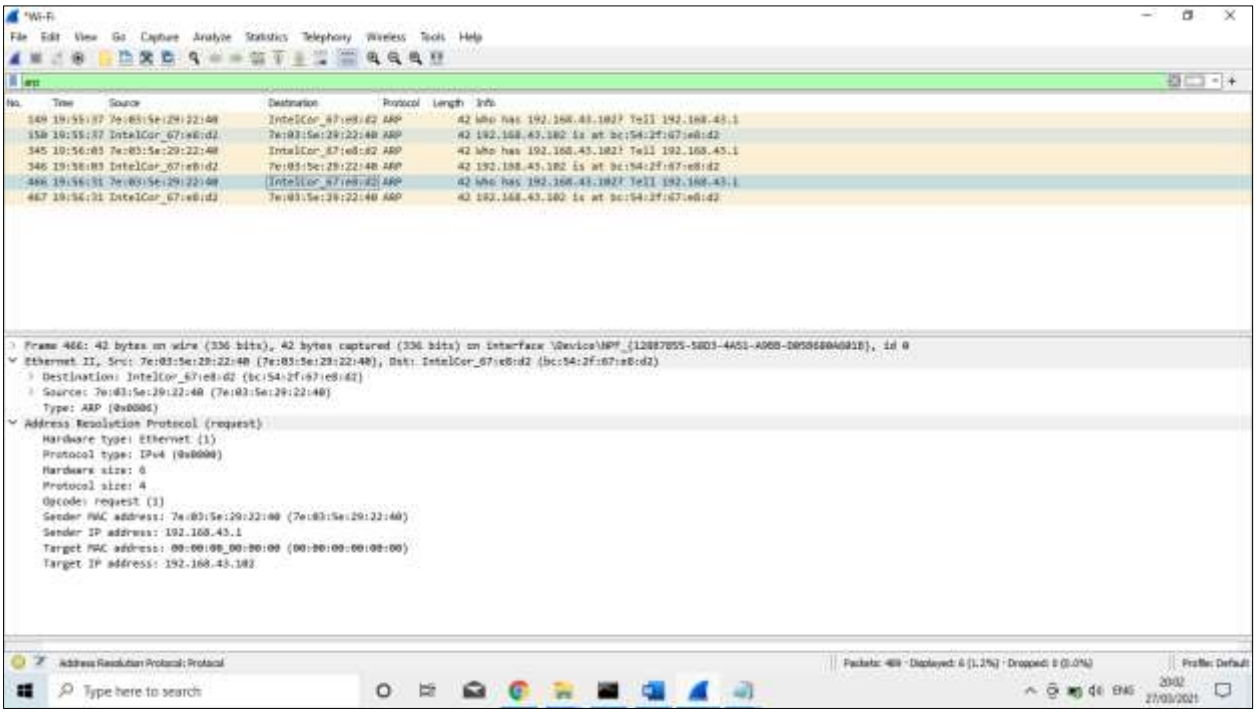
➢ *Capture User Datagram Protocol (UDP)*



❋ Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) is a procedure for mapping a dynamic Internet Protocol address (IP address) to a permanent physical machine address in a local area network (LAN). The physical machine address is also known as a Media Access Control or MAC address.

➢ *Capture Address Resolution Protocol (ARP)*

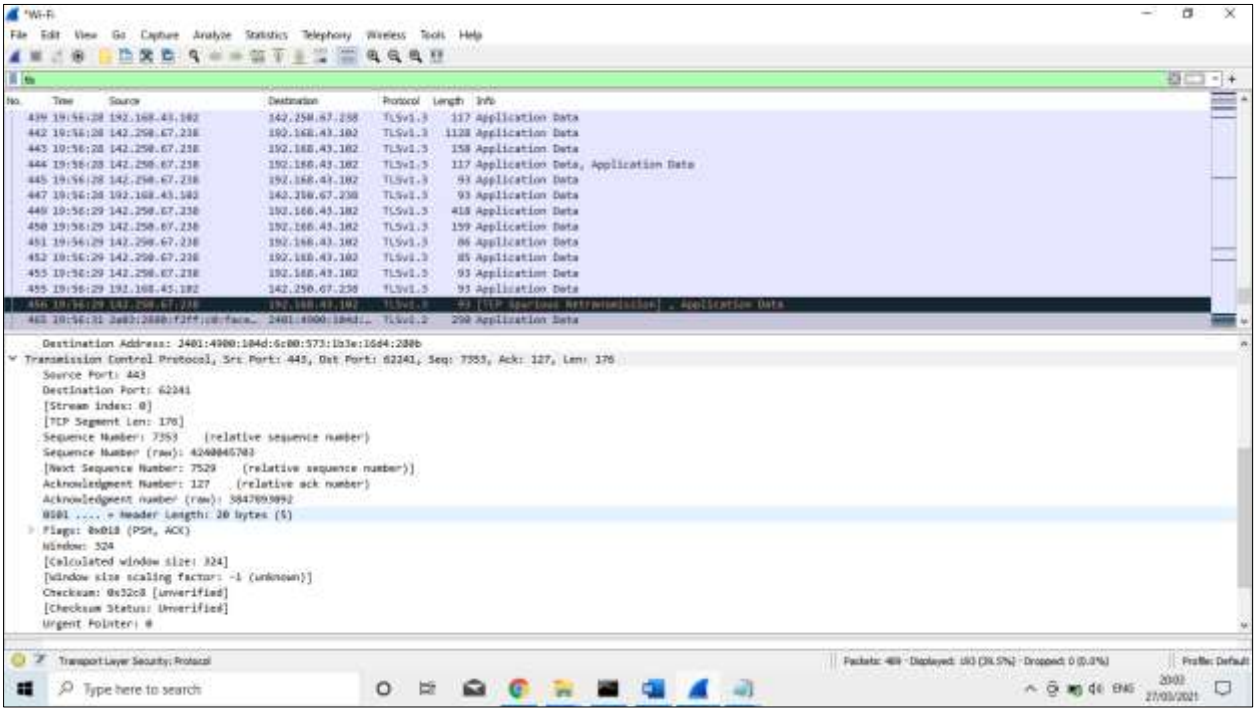


❋ Transport Layer Security Protocol (TLS)

Transport Layer Security (TLS) is the successor protocol to SSL. TLS is an improved version of SSL.

TLS is a security protocol that provides privacy and data integrity for Internet communications. Implementing *TLS* is a standard practice for building secure web apps.

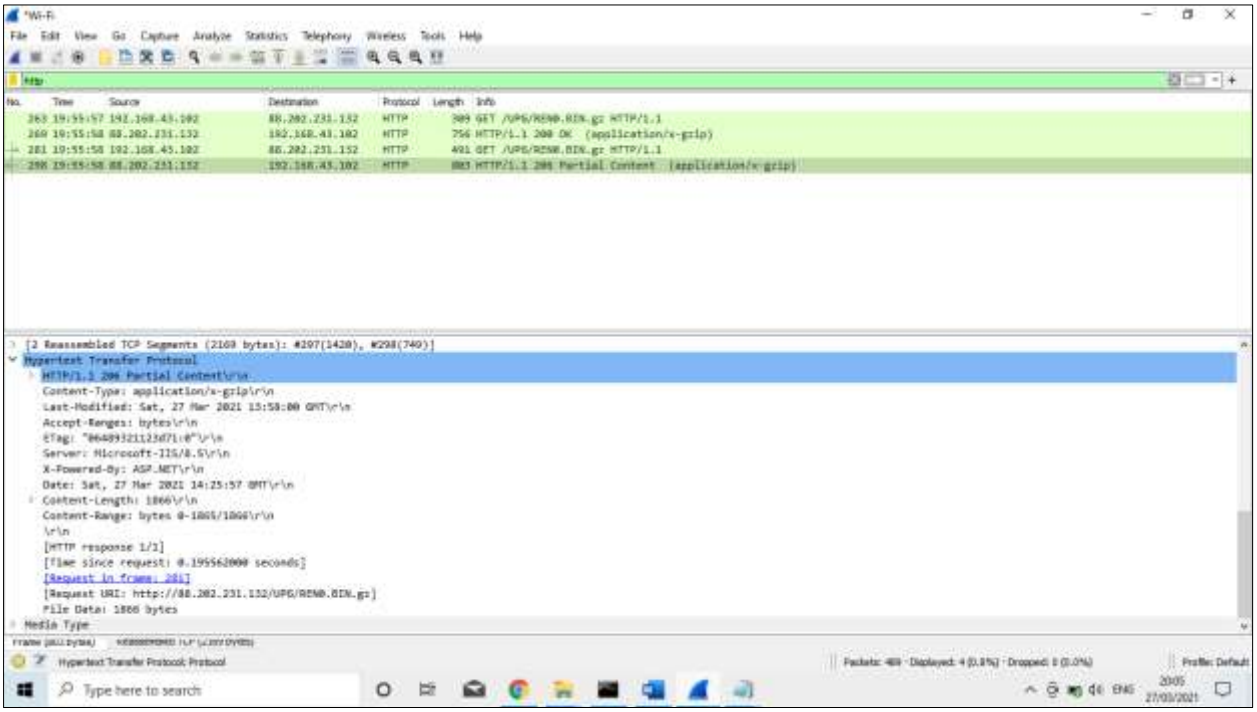
➴ *Capture Transport Layer Security Protocol (TLS)*



HyperText Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes.

Capture HyperText Transfer Protocol (HTTP)



Capture filter

Wireshark uses the same syntax for capture filters as tcpdump, Windpump, Analyzer, and any other program that uses the libpcap/WinPcap library.

If you need a capture filter for a specific protocol, have a look for it at the Protocol Reference.

Capture filters are set before starting a packet capture and cannot be modified during the capture. Display filters on the other hand do not have this limitation and you can change them on the fly.

In the main window, one can find the capture filter just above the interfaces list and in the interfaces dialog. The display filter can be changed above the packet list.

Examples

Capture only traffic to or from IP address 172.18.5.4:

- host 172.18.5.4

Capture traffic to or from a range of IP addresses:

- net 192.168.0.0/24

Or

- net 192.168.0.0 mask 255.255.255.0

Capture traffic from a range of IP addresses:

- src net 192.168.0.0/24

Or

- src net 192.168.0.0 mask 255.255.255.0

Capture traffic to a range of IP addresses:

- dst net 192.168.0.0/24

Or

- dst net 192.168.0.0 mask 255.255.255.0

Capture only DNS (port 53) traffic:

- port 53

Capture non-HTTP and non-SMTP traffic on your server (both are equivalent):

- host www.example.com and not (port 80 or port 25)

And

- host www.example.com and not port 80 and not port 25

Capture except all ARP and DNS traffic:

- port not 53 and not arp

Capture traffic within a range of ports

- (tcp[0:2] > 1500 and tcp[0:2] < 1550) or (tcp[2:2] > 1500 and tcp[2:2] < 1550)

Or, with newer versions of libpcap (0.9.1 and later):

- tcp portrange 1501-1549

Capture only Ethernet type EAPOL:

- ether proto 0x888e

Reject Ethernet frames towards the Link Layer Discovery Protocol Multicast group:

- not ether dst 01:80:c2:00:00:0e

Capture only IPv4 traffic - the shortest filter, but sometimes very useful to get rid of lower layer protocols like ARP and STP:

- Ip

Capture only unicast traffic - useful to get rid of noise on the network if you only want to see traffic to and from your machine, not, for example, broadcast and multicast announcements:

- not broadcast and not multicast

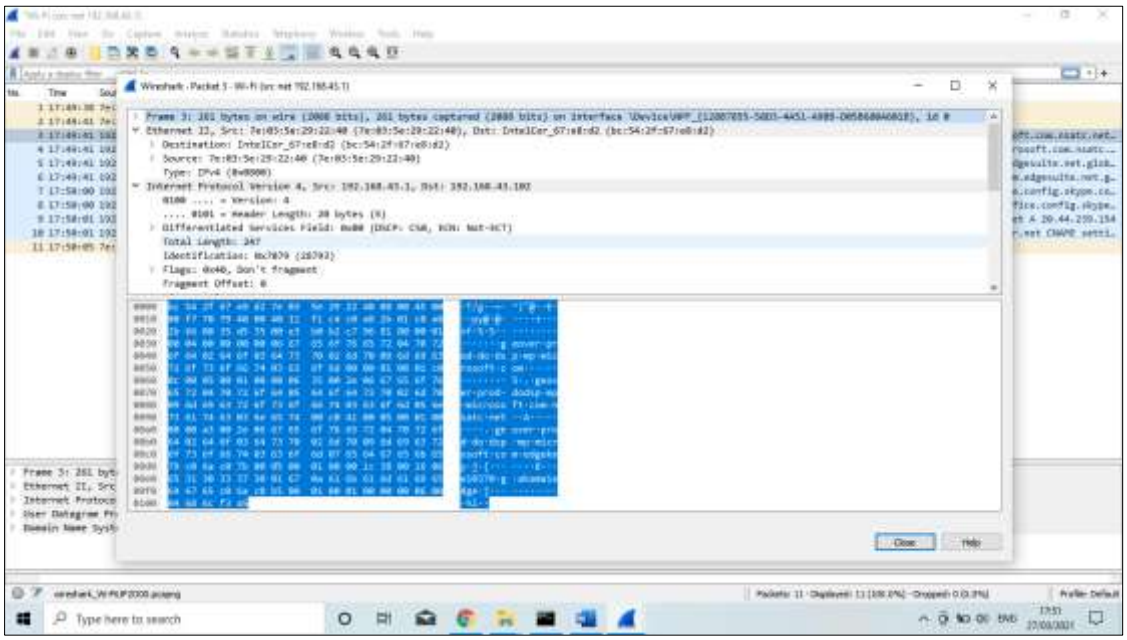
Capture IPv6 "all nodes" (router and neighbor advertisement) traffic. Can be used to find rogue RAs:

- dst host ff02::1

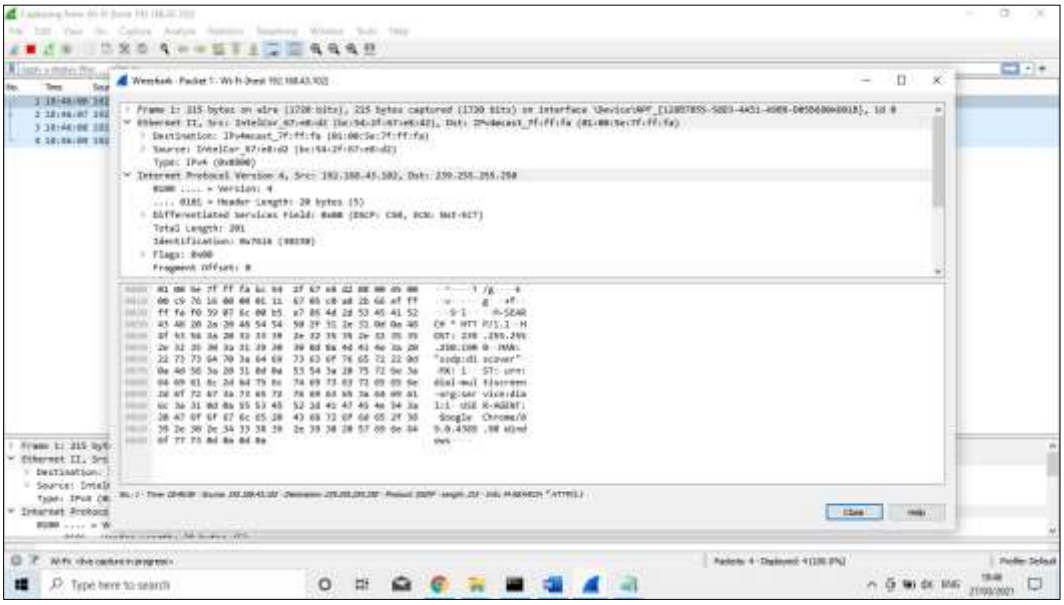
Capture HTTP GET requests. This looks for the bytes 'G', 'E', 'T', and ' ' (hex values 47, 45, 54, and 20) just after the TCP header. "tcp[12:1] & 0xf0) >> 2" figures out the TCP header length. From Jefferson Ogata via the tcpdump-workers mailing list.

- port 80 and tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x47455420

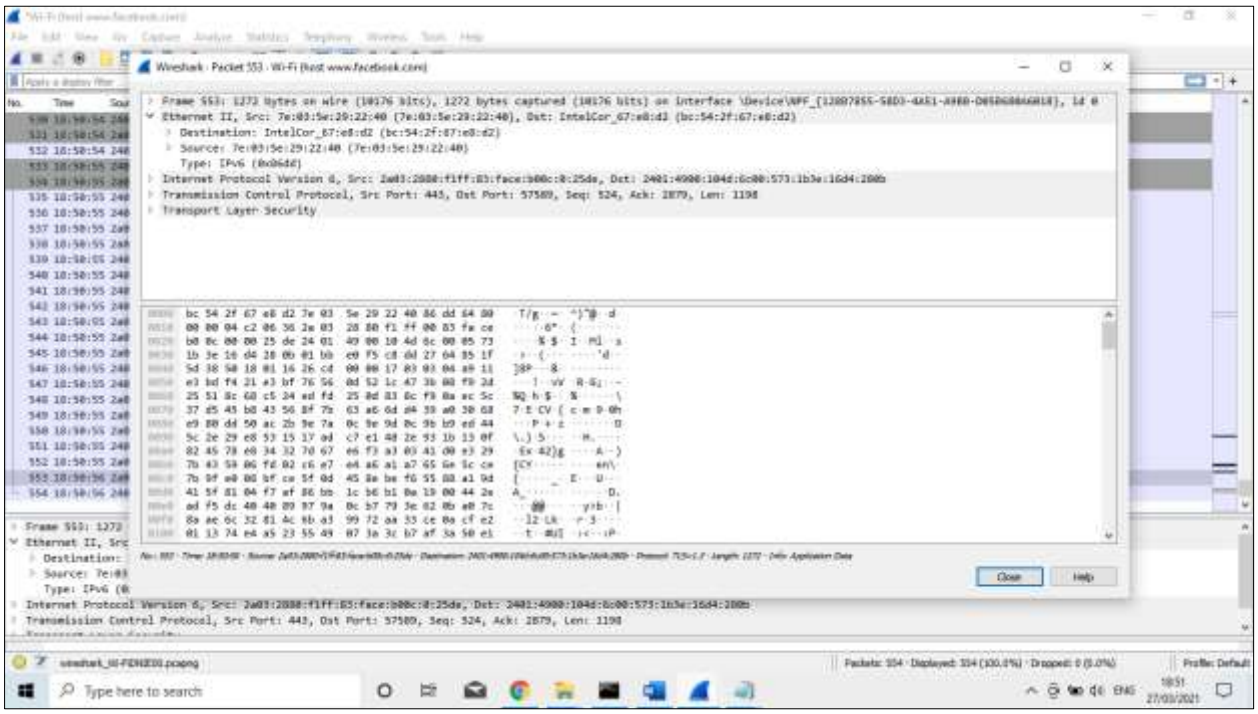
- *Src net DNS server IP filter*



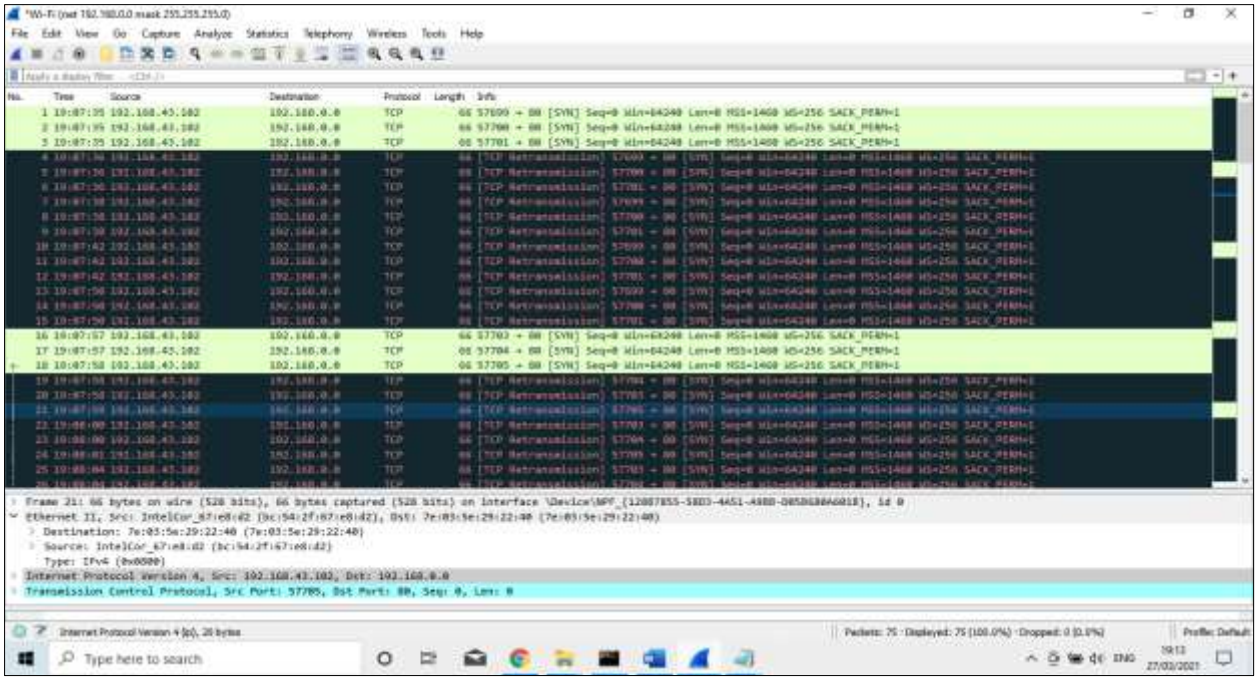
- *Capture only traffic to or from IP address 192.168.43.102*



- *Capture non-HTTP and non-SMTP traffic on my server*



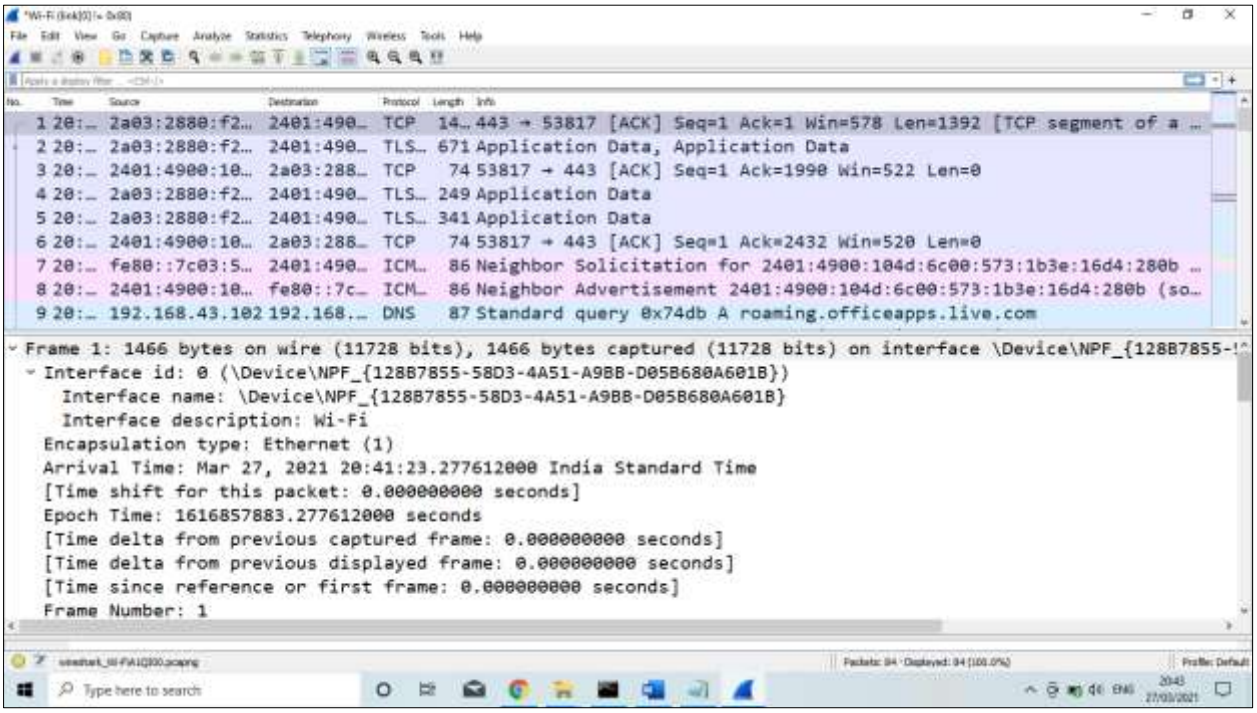
- *Capture traffic to or from a range of IP addresses:*



❋ Wireless Local Area Network (WLAN)

WLAN stands for wireless local area network .It is a wireless computer network that links two or more devices using wireless communication to form a local area network (LAN) within a limited area such as a home, school, computer laboratory, campus, or office building.

➢ *Capture WLAN traffic without Beacons:*

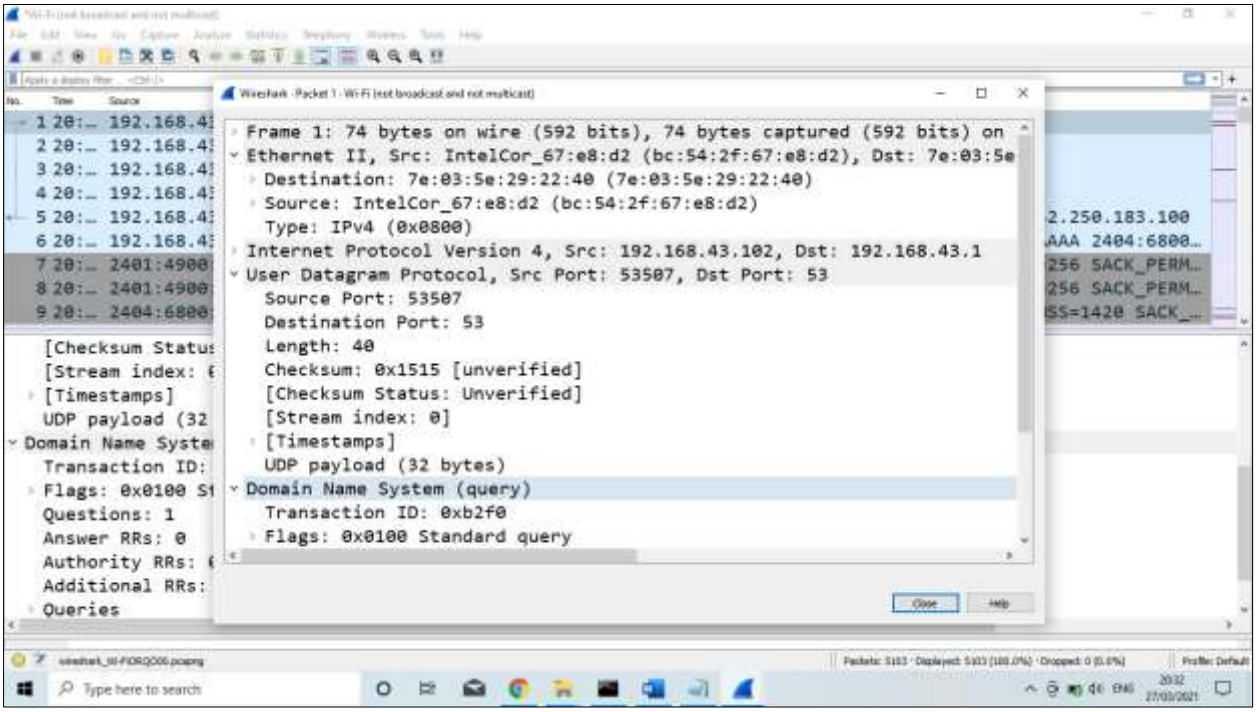


❋ Unicast Traffic

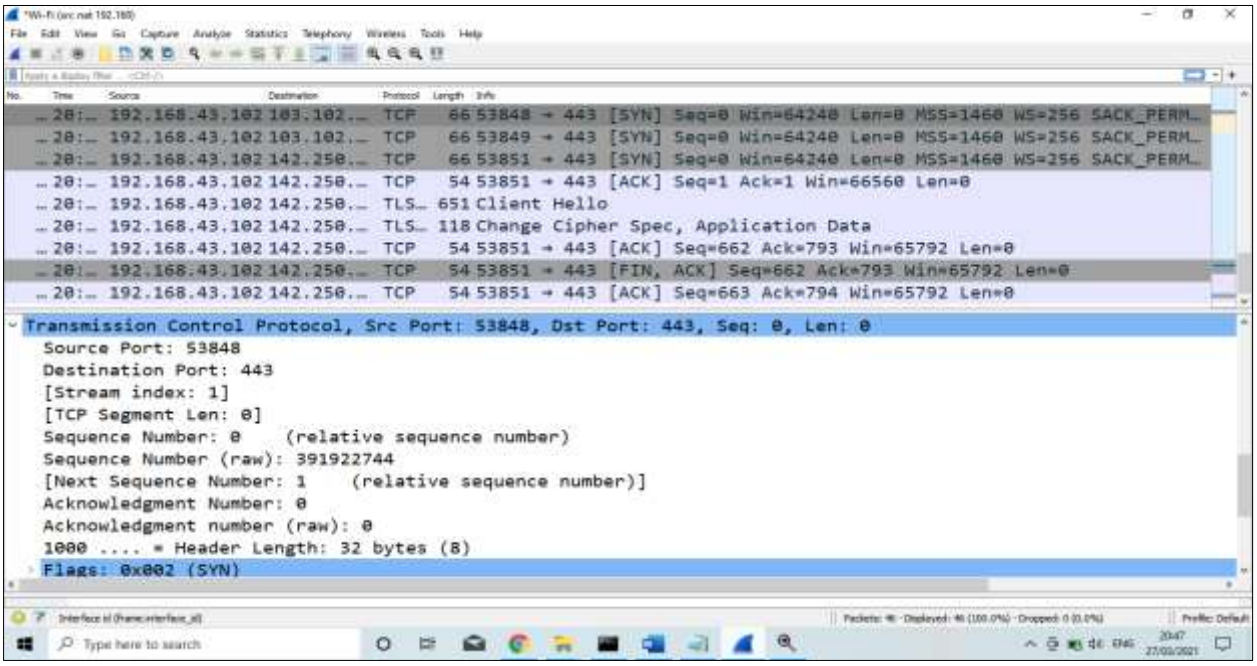
Unicast: traffic, many streams of IP packets that move across networks flow from a single point, such as a website server, to a single endpoint such as a client PC. This is the most common form of information transference on networks.

Traffic is sent from one host to another. A replica of each packet in the data stream goes to every host that requests it.

- *Capture only unicast traffic - useful to get rid of noise on the network if you only want to see traffic to and from your machine, not*



- *Capture all traffic originating (source) in the IP range 192.168.43.102:*



Conclusion

Wireshark is a program that is used to capture data packets to allow a more precise analysis. The main focus of this tool is observing the data traffic within a network. Such a tool allows the user to examine his/her own computer for protocol errors and problems within the network architecture. Accordingly, Wireshark is also gaining significance within the information technology and network-internal communication, because by finding discrepancies, risks to the PC and its components can be prevented. From a security aspect it must be taken into account that such a program is helpful in discovering and stopping hacker attacks. Especially among people working in the industry, this can be of an advantage if sensitive data is stored on their computer that should never reach third parties. A high-quality network analyzer bundled with useful advanced features as the above will help any engineer or administrator diagnose and deal with network problems quickly and efficiently, but also capture suspicious network traffic patterns often associated with hacking attempts. A network protocol analyzer is a tool used to monitor data traffic and analyze captured signals as they travel across communication channels.

Challenges

The challenge of this project is that our team members worked very hard to collect data from different sources for completion of the project, we have used Wireshark open sources software. Some time we have faced so many technical issue with our devices, sometime print screen not working or delay to working when we trying to capture our running filtering image, but our team members are very supportive and hardworking so we have successful to overcome the problematic situation. Our team spirit is the fuel that allows us to accomplish our project.

Bibliography

To accomplish the project of "NETWORK PROTOCOL ANALYZER" we taken help of the following sites:

- www.google.com
- www.wikipedia.com
- www.wireshark.org


Thank
you!