Wireshark

A Pentester Guide

Table of Contents

Abstract

Many people wonder if Wireshark can capture passwords. The answer is undoubtedly yes! Wireshark can capture not only passwords, but any type of data passing through a network – usernames, email addresses, personal information, pictures, videos, or anything else.

Wireshark can sniff the passwords passing through as long as we can capture network traffic. But the question is, what kind of passwords are they? Or, more precisely, which network protocols' passwords can we obtain? That is the subject of this report.

Disclaimer: This report is provided for educational and informational purpose only (Penetration Testing). Penetration Testing refers to legal intrusion tests that aim to identify vulnerabilities and improve cybersecurity, rather than for malicious purposes.

Introduction

In the first section of this report, we'll delve into 'Network Packet Forensics,' exploring vital aspects such as data transfer between nodes, the 'OSI 7-layer model,' and how Wireshark stores information across layers when capturing network traffic.

Moving on to the second part, we'll examine how Wireshark can capture passwords, a result of certain unencrypted network protocols known as clear text protocols. These protocols expose all data, including passwords, making it visible to anyone with the ability to intercept the communication, such as a man-in-the-middle.

Network Packet Forensic

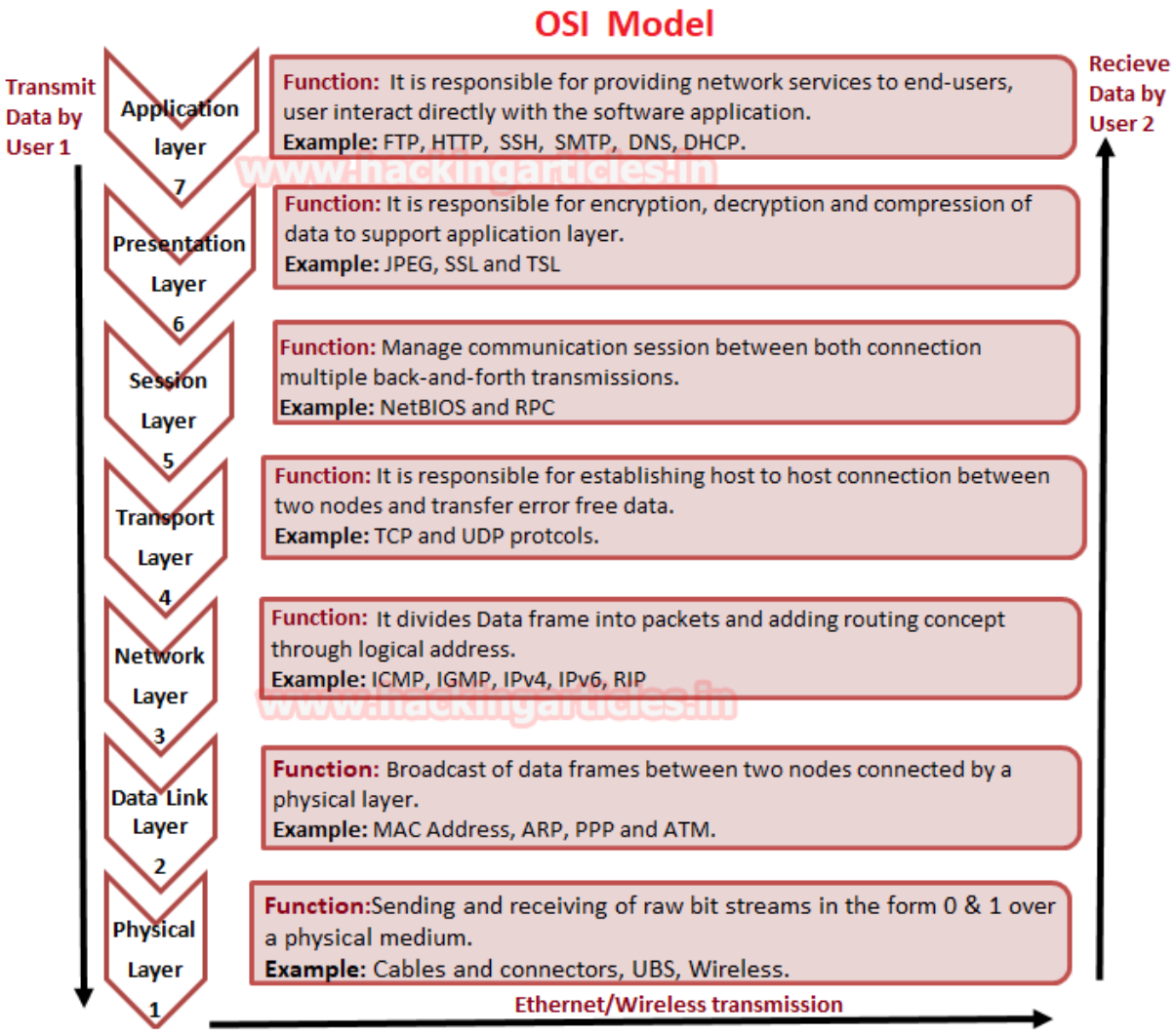
As we know for transferring the data from one system to other, we need a network connection which can be wired or wireless connection. But in the actual transmission of data does not only depend upon network connection apart from that it involves several phases for transmitting data from one system to another which was explained by the OSI model.

OSI stands for **O**pen **S**ystems **I**nterconnection model which is a conceptual model that defines and standardizes the process of communication between the sender's and receiver's system. The data is transfer through 7 layers of architecture where each layer has a specific function in transmitting data over the next layer.

Now have a look over given below image where we had explained the functionality of each layer in the OSI model. So, when data is transmitted by sender's network then it will go in downward direction and data move from application layer to physical layer whereas when the receiver will receive the transmitted data it will come in an upward direction from physical layer to application layer.

Flow of Data from Sender's network: **Application** > **Presentation** > **Session** > **Transport** > **Network** > **Data Link** > **Physical**

Flow of Data from Receiver's network: **Physical** > **Data Link** > **Network** > **Transport** > **Session** > **Presentation** > **Application**



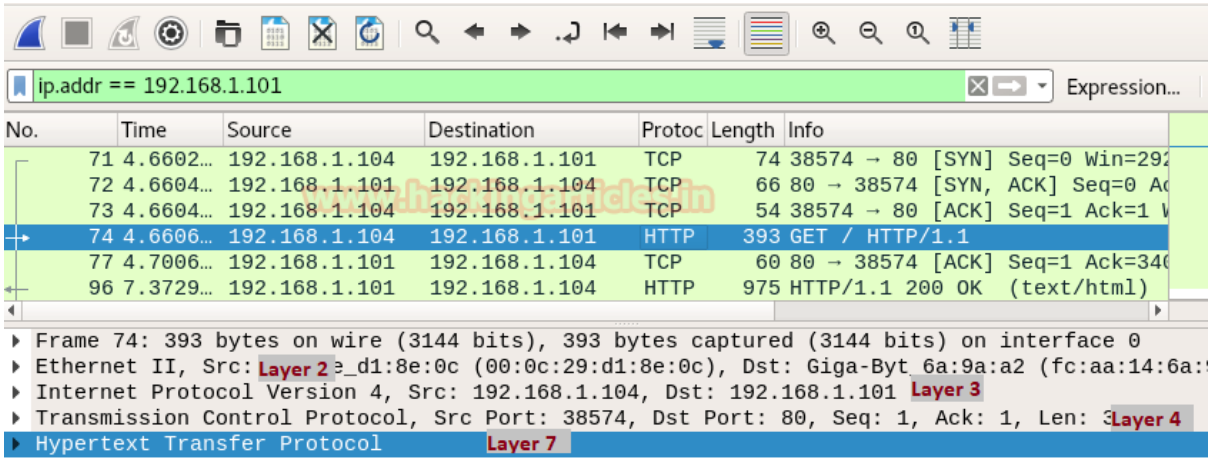
Examine Layers captured by Wireshark

Basically, when a user opens an application for sending or receiving Data then he directly interacts with the application layer for both operations either sending or receiving of data. For example, we act as a client when use Http protocol for uploading or Downloading a Game; FTP for downloading a File; SSH for accessing the shell of the remote system.

While connecting with any application for sharing data between server and client we make use of Wireshark for capturing the flow of network traffic stream to examine the OSI model theory through captured traffic.

From given below image you can observe that Wireshark has captured the traffic of four layers in direction of the source (sender) to destination (receiver) network.

Here it has successfully captured **Layer 2** > **Layer 3** > **Layer 4** and then **Layer 7** information.



Ethernet Header (Data Link)

Data link layer holds 6 bytes of **Mac address** of sender's system and receiver's system with 2 bytes of **Ether type** is used to indicate which protocol is encapsulated i.e. IPv4/IPv6 or ARP.

In Wireshark Ethernet II layer represent the information transmitted over the data link layer. From given below image you can observe that highlighted lower part of Wireshark is showing information in Hexadecimal format where the first row holds information of Ethernet headers details.

So here you can get the source and destination Mac address which also available in Ethernet Header.

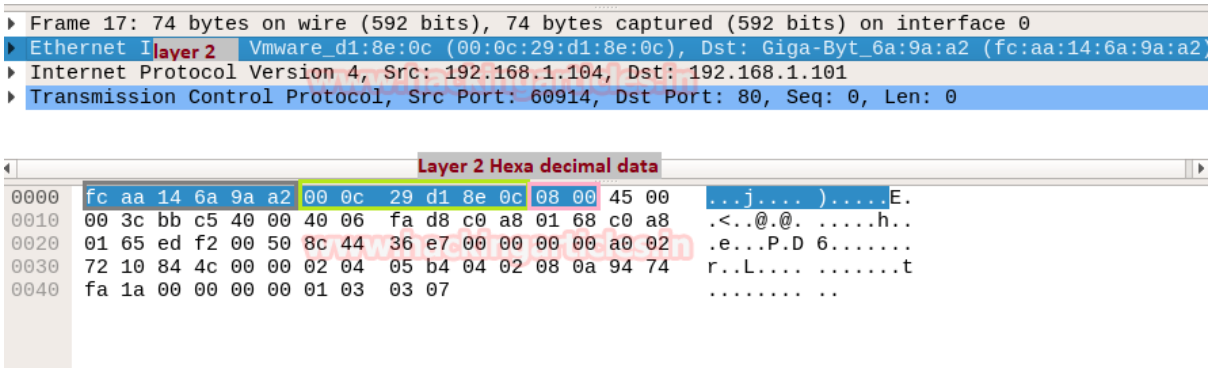
The row is divided into three columns as described below:

Ethernet header **14 bytes**	Destination MAC Address 6 Bytes	Source MAC Address 6 Bytes	Ether Type 2 Bytes
Bits Color	Gray	Light Green	Pink
Hexadecimal value	Fc:aa:14:6a:9a:a2	00:0c:29:d1:8e:0c	0800

As we know the MAC address of the system is always represented in Hexadecimal format but both types are generally categorized in the ways given below:

Ether Type	Hexadecimal Value
ARP: Address Resolution Protocol	0x0806
IPv4: Internet Protocol version 4	0x0800
IPv6: Internet Protocol version 6	0x86dd
IEEE 802.1Q	0x8100

Once again if you notice the given below image then you can observe the highlighted text in pink colour is showing hex value **08 00** which indicates that here **IPv4** is used.



IP Header (Network Layer)

IP header in Wireshark has described the network layer information which is also known as the backbone of the OSI model as it holds Internet Protocol version 4's complete details. Network layer divides data frame into packets and defines its routing path through some hardware devices such as routers, bridges, and switches. These packets are identified through their logical address i.e. source or destination network IP address.

In the image of Wireshark, I have highlighted six most important values which contain vital information of a data packet and this information always flows in the same way as they are encapsulated in the same pattern for each IP header.

Now here, **45** represent IP header length where "4" indicates **IP version 4** and "5" is header length of **5 bits**. while **40** is time to live (**TTL**) of packet and **06** is hex value for **TCP** protocol which means these values changes if anything changes i.e. TTL, Ipv4 and Protocol.

Therefore, you can take help of given below table for examining TTL value for the different operating system.

Operating System	Hex Value TTL	Decimal value TTL
Windows	80	128
Linux	40	64
MAC	39	57

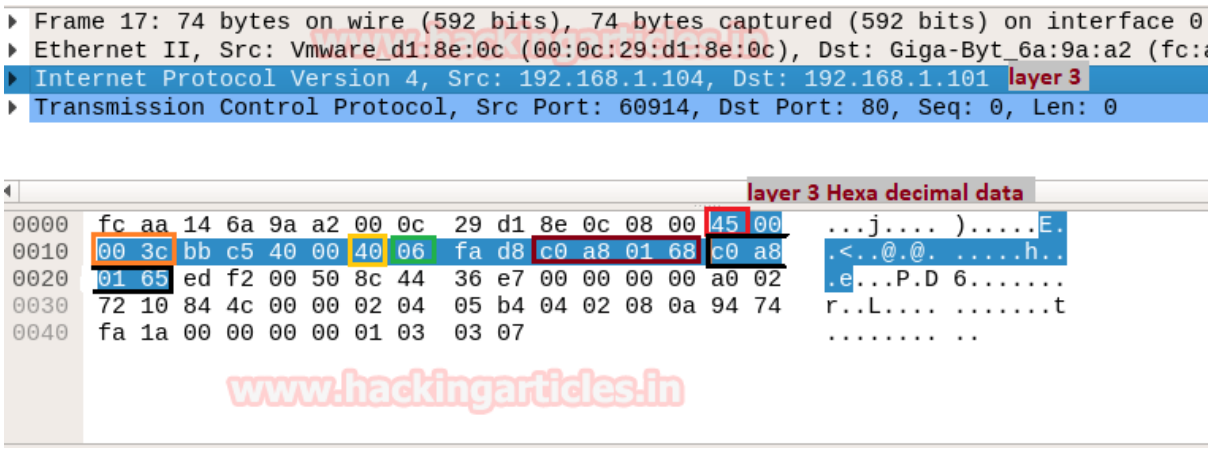
Similarly, you can take help of given below table for examining other Protocol value.

Protocol	Hex Value	Decimal Value
ICMP	1	1
TCP	6	6
EGP	8	8
UDP	11	17

From given below image you can observe Hexadecimal information of the IP header field and using a given table you can study this value to obtain their original value.

IP header (20 bytes)	Header length	Total Length	TTL	Protocol	Source IP	Destination IP
Bits Color	Red	Orange	Yellow	Dark Green	Dark Brown	Black
Hex Value	5	3c	40	06	C0.a8.01.68	C0.a8.01.65
Decimal value	5	60	64	6	192.168.1.104	192.168.1.105

The IP header length is always given in form of the bit and here it is 5 bytes which are also minimum IP header length and to make it 20 bytes, multiply 4 with 5 i.e., 20 bytes.



TCP Header (Transport Layer)

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) are the major protocols as it gives host-to-host connectivity at the Transport Layer of the OSI model. It is also known as Heart of OSI model as it plays a major role in transmitting errors free data.

By examining Network Layer information through Wireshark, we found that here TCP is used for establishing a connection with destination network.

We knew that a computer communicates with another device like a modem, printer, or network server; it needs to handshake with it to establish a connection.

TCP follows **Three-Way-Handshakes** as describe below:

- A client sends a TCP packet to the server with the **SYN flag**
- A server responds to the client request with the **SYN** and **ACK** flags set.
- Client completes the connection by sending a packet with the **ACK** flag set

Structure of TCP segment

Transmission Control Protocol accepts data from a data stream, splits it into chunks, and adds a TCP header creating a TCP segment. A TCP segment only carries the sequence number of the first byte in the segment.

A TCP segment consists of a segment header and a data section. The TCP header contains mandatory fields and an optional extension field.

Source Port	The 16-bit source port number, Identifies the sending port.
Destination Port	The 16-bit destination port number. Identifies the receiving port
Sequence Number	The sequence number of the first data byte in this segment. If the SYN control bit is set, the sequence number is the initial sequence number (n) and the first data byte is n+1.
Acknowledgment Number	If the ACK control bit is set, this field contains the value of the next sequence number that the receiver is expecting to receive.
Data Offset	The number of 32-bit words in the TCP header. It indicates where the data begins.
Reserved	Six bits reserved for future use; must be zero.

Flags	CWR, ECE, URG, ACK, PSH, RST, SYN, FIN
Window	Used in ACK segments. It specifies the number of data bytes, beginning with the one indicated in the acknowledgment number field that the receiver (the sender of this segment) is willing to accept.
Checksum	The 16-bit one's complement of the one's complement sum of all 16-bit words in a pseudo-header, the TCP header, and the TCP data. While computing the checksum, the checksum field itself is considered zero.
Urgent Pointer	Points to the first data octet following the urgent data. Only significant when the URG control bit is set.
Options	Just as in the case of IP datagram options, options can be either: – A single byte containing the option number – A variable length option in the following format
Padding	The TCP header padding is used to ensure that the TCP header ends and data begins on a 32-bit boundary. The padding is composed of zeros.

Different Types of TCP flags

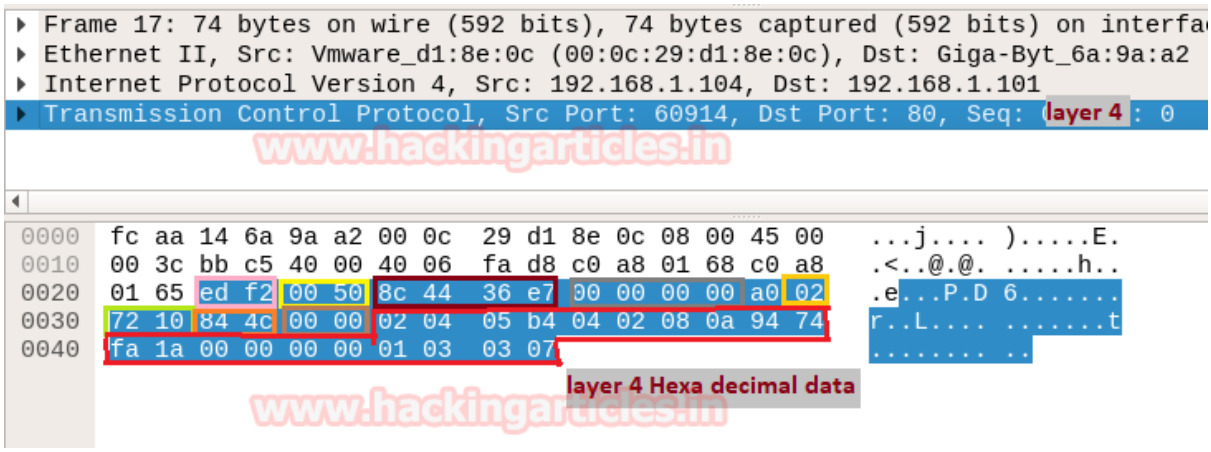
TCP flags are used within TCP header as these are control bits that specify particular connection states or information about how a packet should be set. TCP flag field in a TCP segment will help us to understand the function and purpose of any packet in the connection.

List of flags	Description	Decimal Value	Hex Value
CWR	Congestion Window Reduced (CWR) flag is set by the sending host to shows that it received a TCP segment with the ECE flag set	128	80
ECE	ECN-Echo indicate that the TCP peer is ECN capable during 3-way handshake	64	40
URG	Indicates that the urgent pointer field is significant in this segment.	32	20
ACK	Indicates that the acknowledgment field is significant in this segment.	16	10
PSH	Push function to transfer data	08	08
RST	Resets the connection.	04	04
SYN	Synchronizes the sequence numbers.	02	02
FIN	Last packet from sender which means there is no more data.	01	01
NS	Nonce Sum flag used for concealment protection.	00	00

From given below image you can observe Hexadecimal information of TCP header field and using the given table you can study this value to obtain their original value.

Sequence and acknowledgment numbers are is a major part of TCP, and they act as a way to guarantee that all data is transmitted consistently since all data transferred through a TCP connection must be acknowledged by the receiver in a suitable way. When an acknowledgment is not received, then the sender will again send all data that is unacknowledged.

TCP Header	Bits Color	Hex Value	Decimal value
Source Port	Pink	ed f2	60914
Destination Port (**HTTP**)	Lemon Yellow	00 50	80
Sequence Number	Dark Brown	8c 44 36 e7	2353280743
Acknowledgment Number	Grey	00 00 00 00	0
Flag (**SYN**)	Dark Yellow	02	02
Window size	Green	72 10	29,200
Checksum	Orange	84 4c	33,868
Urgent Pointer	Light Brown	00 00	00
Options	Red	*	*



Using given below table you can read Hex value of other Port Number and their Protocol services. Although these services operate after getting acknowledgment from the destination network and explore at application layer OSI model.

In this way, you can examine every layer of Wireshark for Network Packet Forensic.

Ports Number	Services	Hex Value	Decimal Value
21	FTP	15	21
22	SSH	16	22
23	Telnet	17	23
25	SMTP	19	25
53	DNS	35	53
80	HTTP	50	80

Password Sniffing

Because clear text protocols do not encrypt communication, all data, including passwords, is visible to the naked eye. Anyone who is in a position to see the communication (for example, a man in the middle) can eventually see everything.

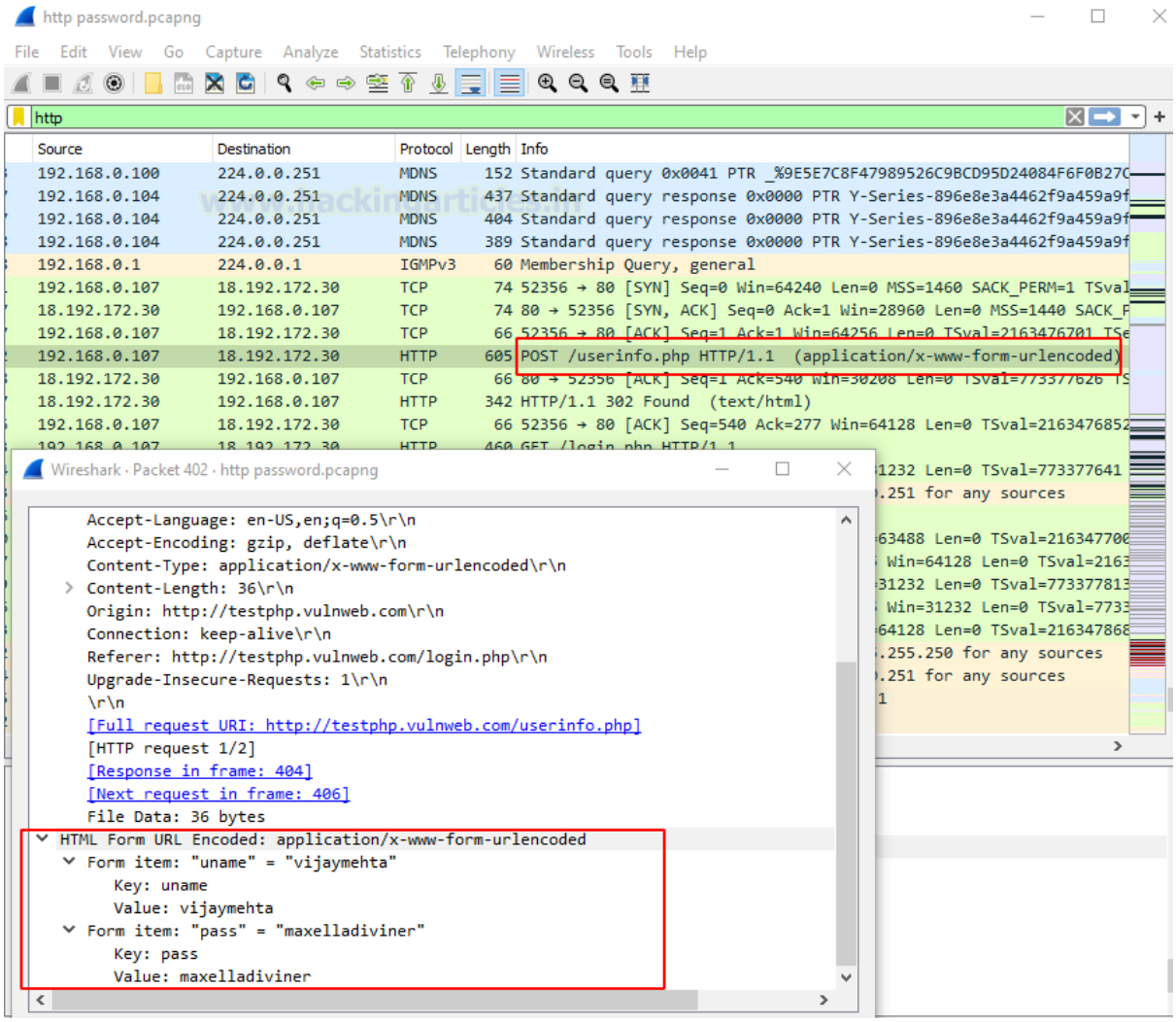
In the sections that follow, we'll take a closer look at these protocols and see examples of captured passwords using Wireshark.

Capture HTTP Password

No introduction is certainly needed for the Hypertext Transfer Protocol (HTTP). It usually works on port 80/TCP, and as it is a text protocol, it does not give the communication parties much or no privacy. Anyone who's able to communicate can catch everything, including passwords, via that channel.

While all major browser vendors have made considerable efforts to prevent the use of HTTP as far as possible, during penetration testing, HTTP can be used on internal media.

Here is an example of login credentials captured in a POST request in an HTTP communication:

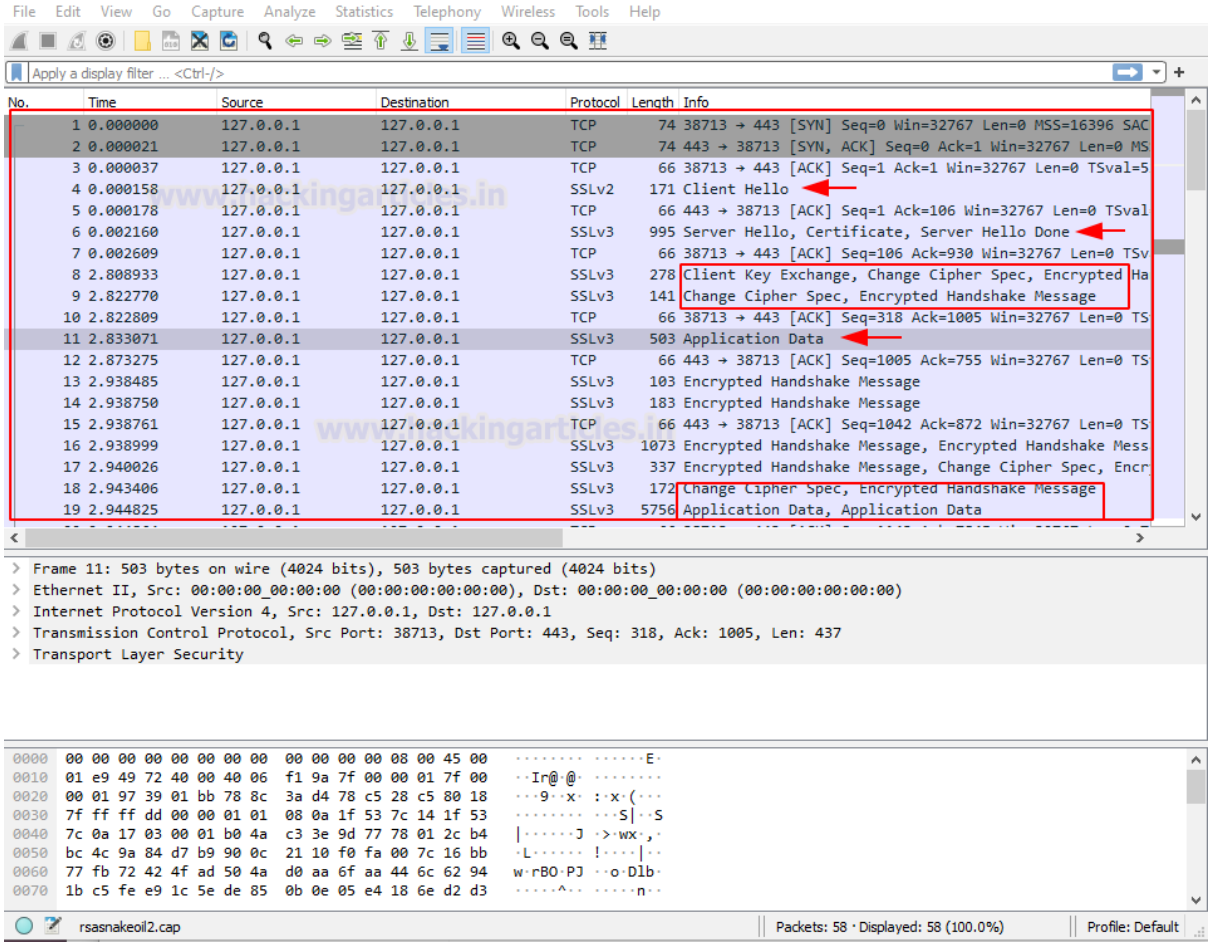


Monitoring HTTPS packets over SSL or TLS

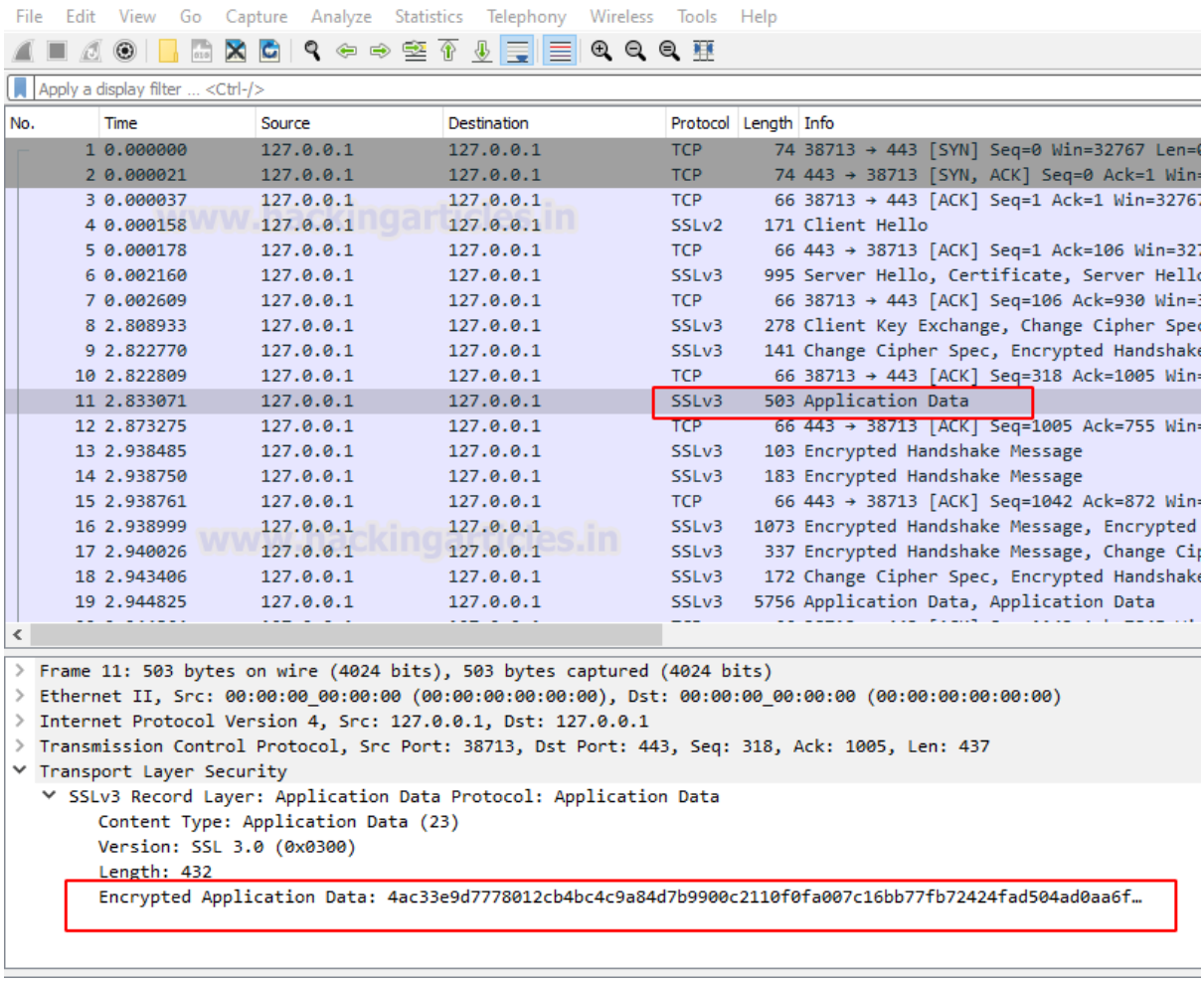
Dissect HTTPS Packet Captures

Open the provided HTTPS/TLS.pcapng file. Where you can see

- The 3-way handshake is happening
- Hello from SSL Client and the ACK from server
- Server Hello and then ACK
- Exchanging some key and Cipher information
- Started Exchanging Data

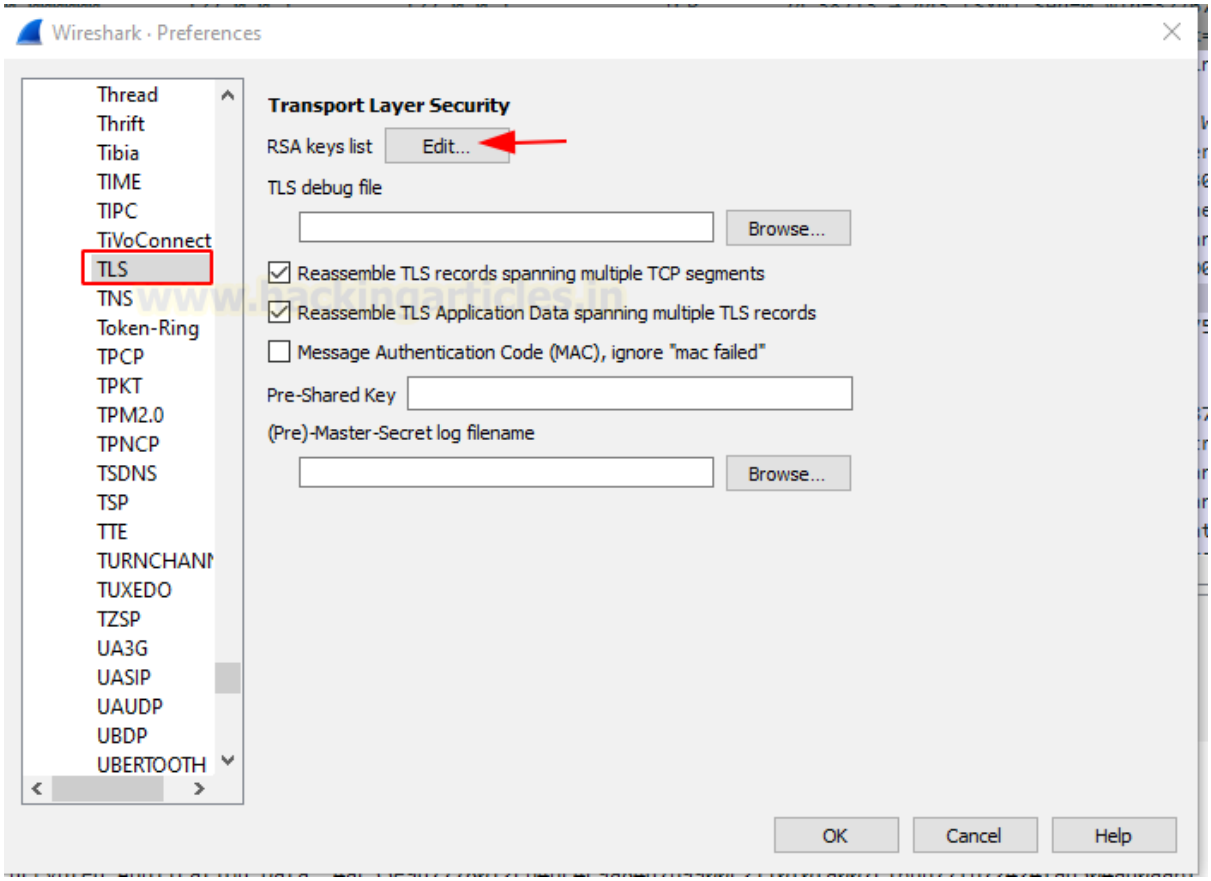


Then, if we click on any application data, that data is unreadable to us. However, with Wireshark, we can decrypt that data… all we need is the server's Private Key. ***Don't worry we have already provided the key along with the PCAP file.***



To Decrypt the Encrypted Application Data over TLS or SSL Navigate to

Edit > Preference > Protocol > TLS

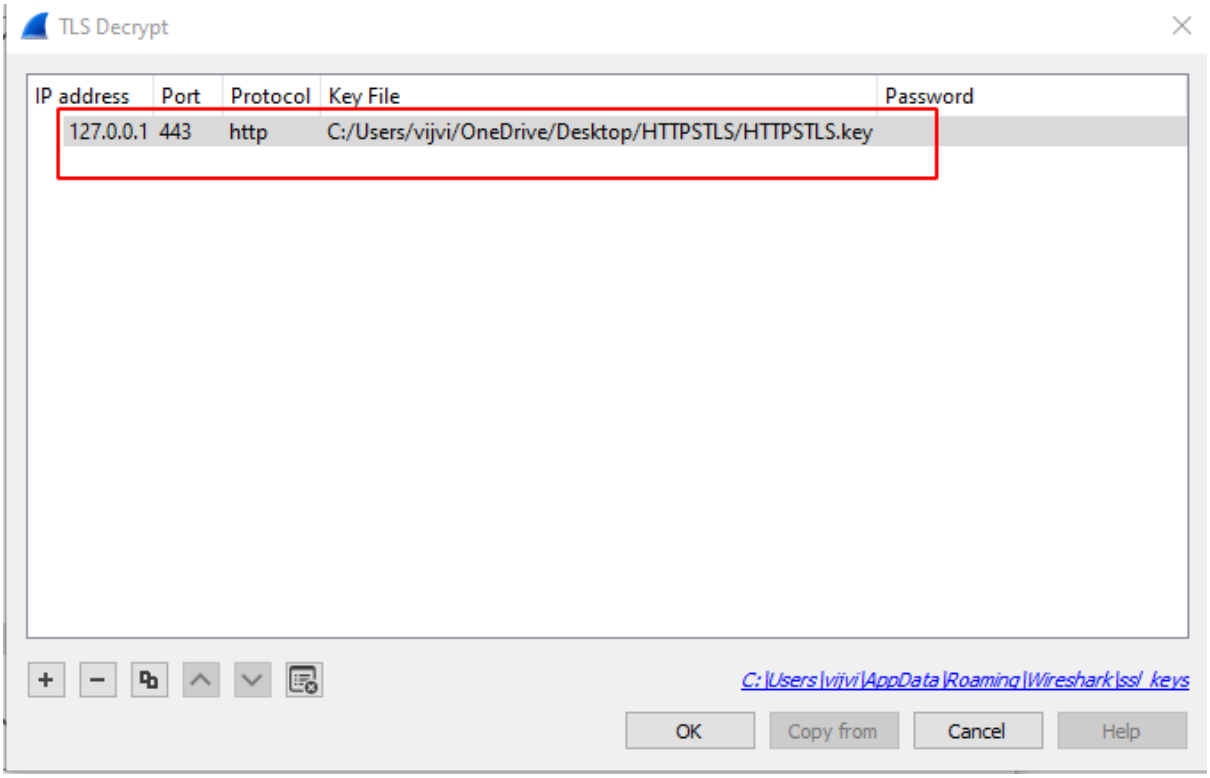


And add these values

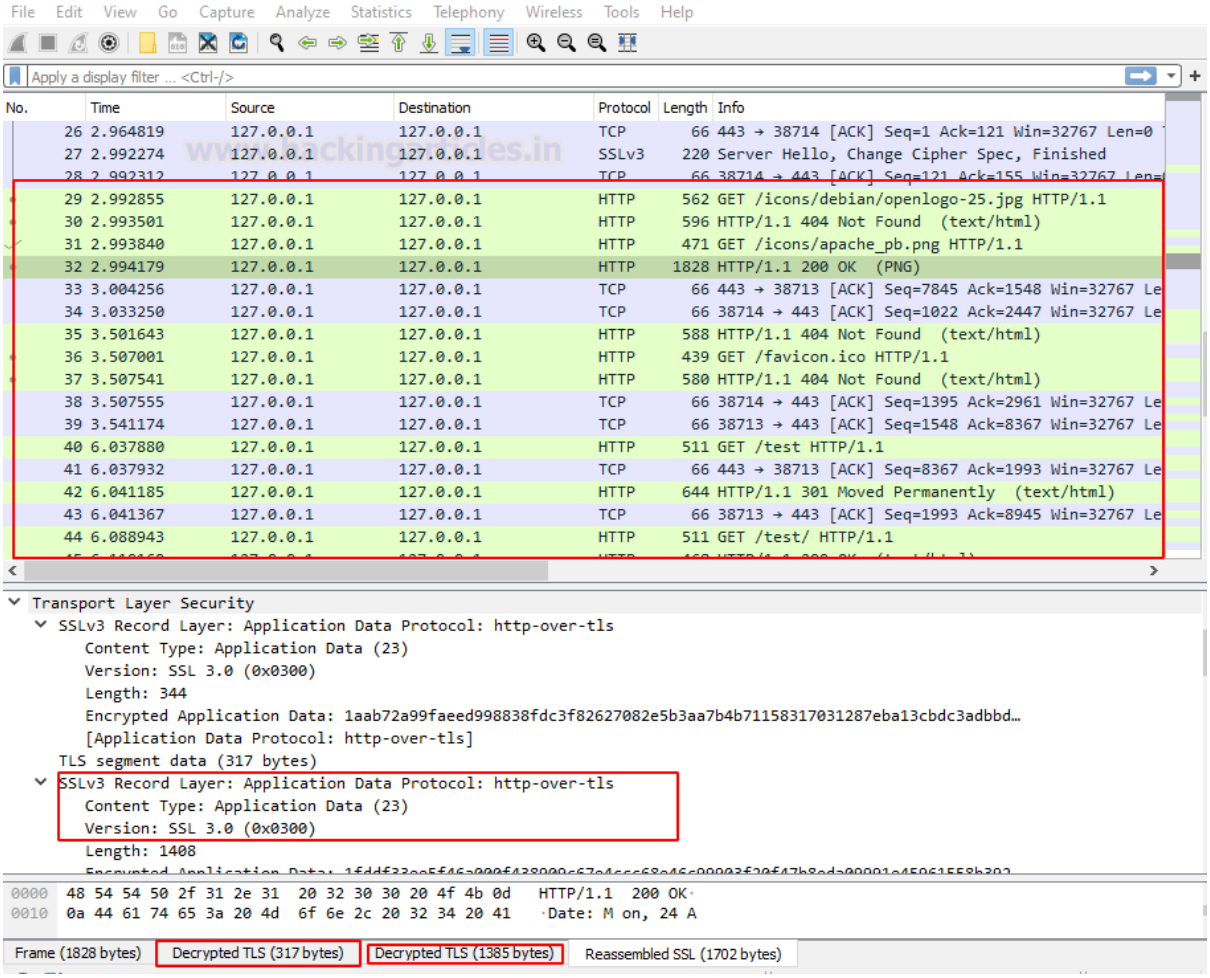
IP address: 127.0.0.1

Port: 443

Key File:



Hurray!!! As you can see, we have Successfully decrypted the Data over the TLS.



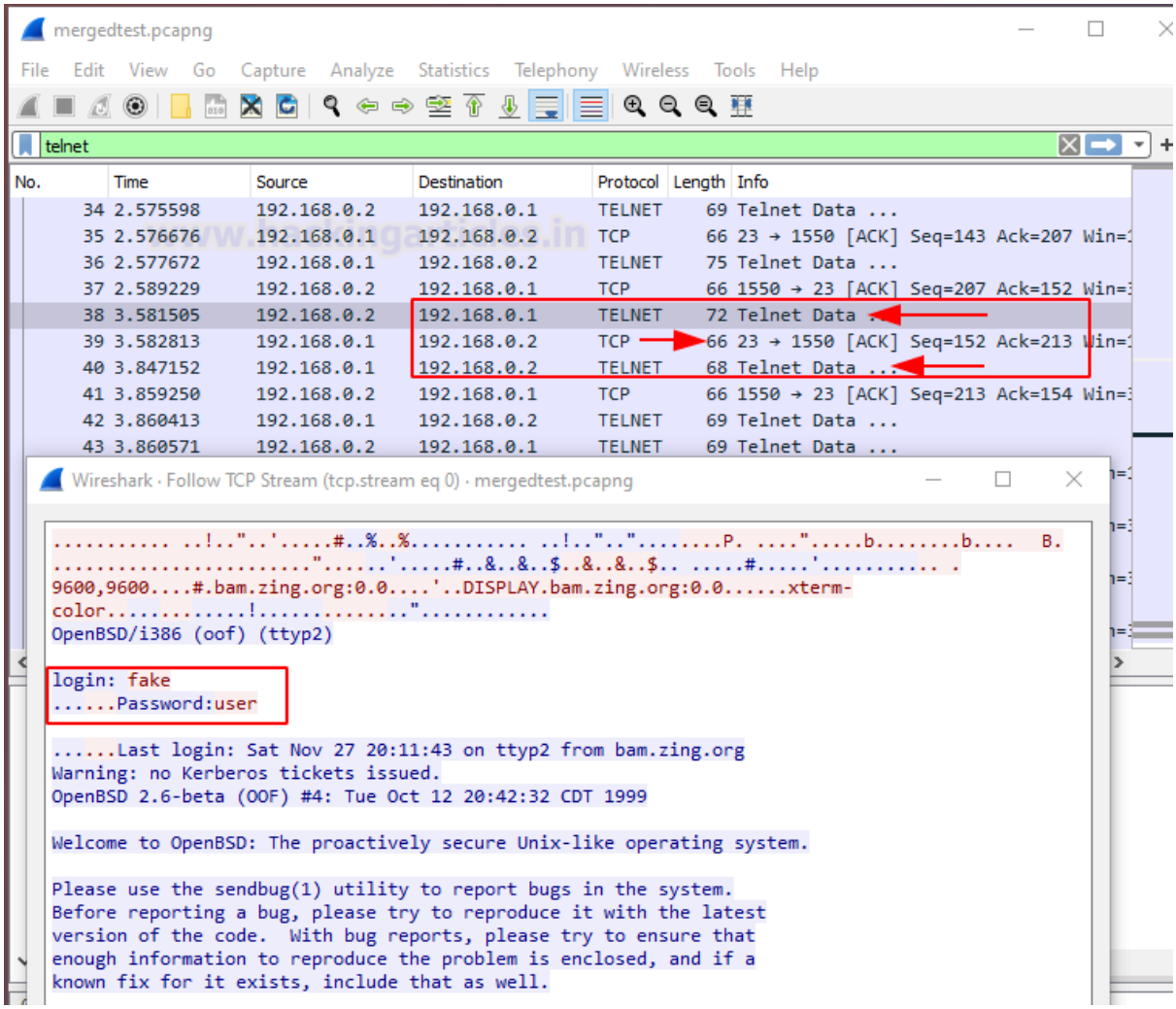
Capture Telnet Password

No introduction is required for Telnet protocol using port tcp/23. It is mainly used for administrative convenience and is known for its insecurity. Since encryption is not available, privacy or unauthorized access protection is not available. Telnet is still used today, however…

Telnet is a protocol used for administration on a wide range of devices. Telnet is the only option for some devices, with no other options (e.g. there is no SSH nor HTTPS web interface available). This makes it extremely difficult for organizations to completely eliminate it. Telnet is commonly seen on:

- Video Conferencing Systems
- Mainframes
- Network equipment
- Storage and Tape systems
- Imaging devices
- Legacy IP based Phones

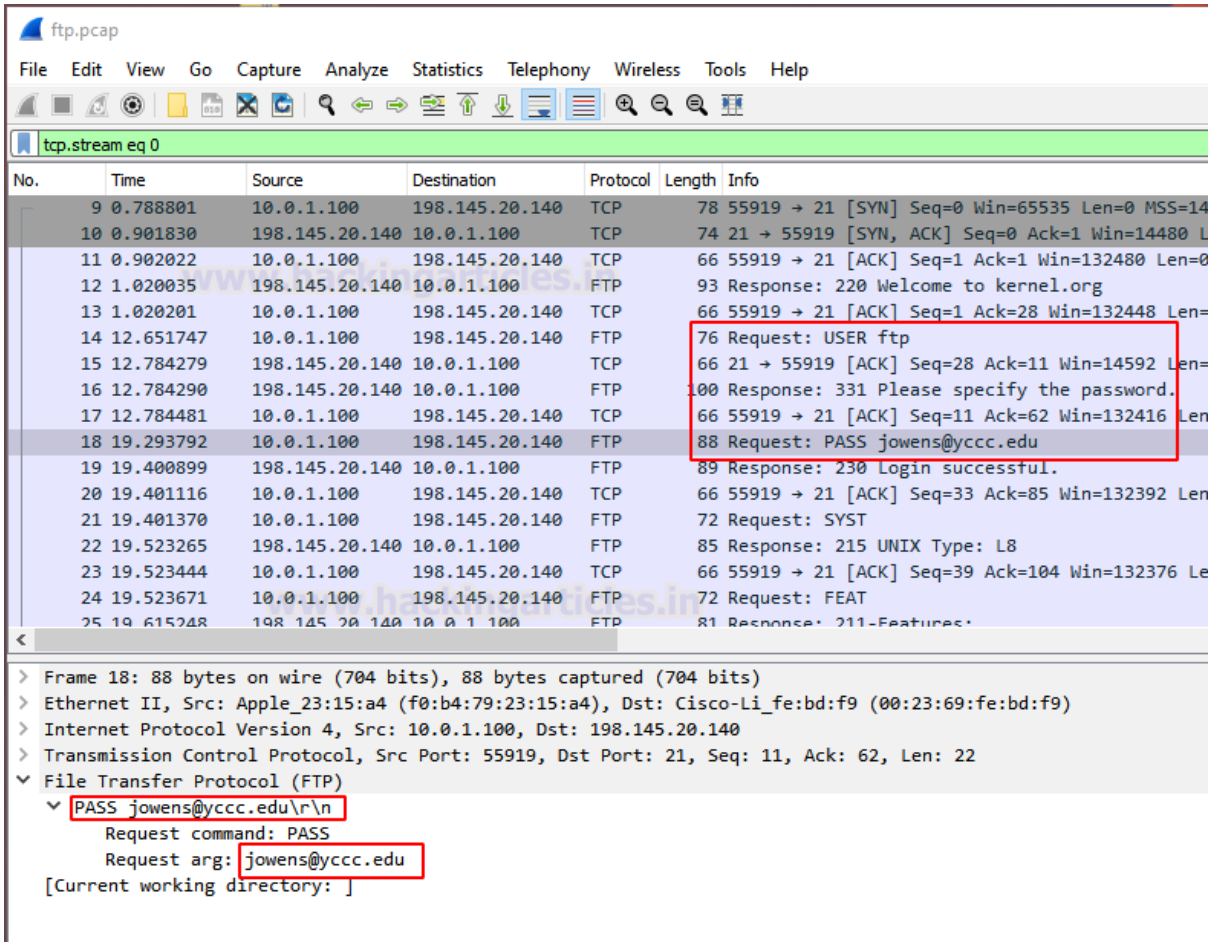
Since telnet is a plain-text protocol, an opponent can wake up to the communication and capture it all, including passwords. The following screenshot shows an example of a telnet communication with the captured password:



So, that now you can see an attacker completely overtake the Mainframe System.

Capture FTP Password

File Transfer Protocol (FTP) usually uses the TCP/20 or the TCP/21 ports. Although this protocol is very old, it is still used in their networks by some organizations. FTP is a plain text protocol so a well-positioned attacker can capture FTP login credentials with Wireshark very easily. This screenshot shows a captured FTP password with Wireshark as an example:



As you can see by sitting in a network, we can easily capture FTP credentials.

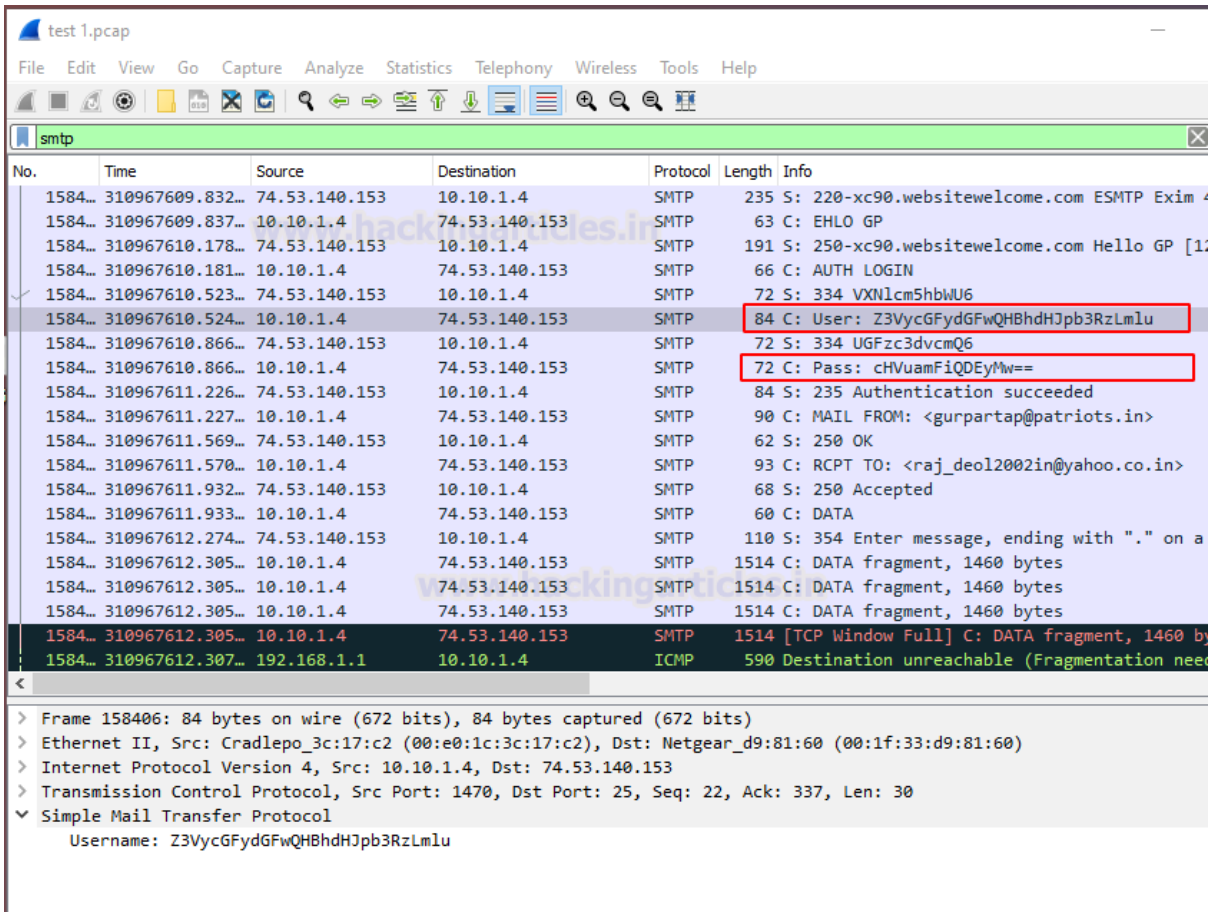
Capture SMTP password

For many decades, we have also been accompanied by SMTP (Simple Mail Transfer Protocol). It uses TCP/25 and although the port TCP/464 is secure, today the port TCP/25 is almost opened on each mail server because of reverse compatibility.

Many TCP/25 servers need the command 'STARTTLS' to begin the encryption of SSL/TLS before any attempts are made to authenticate it. However, mail servers still support plain text authentication across the unencrypted channel within certain organizations. Mostly because of heritage systems in your internal networks.

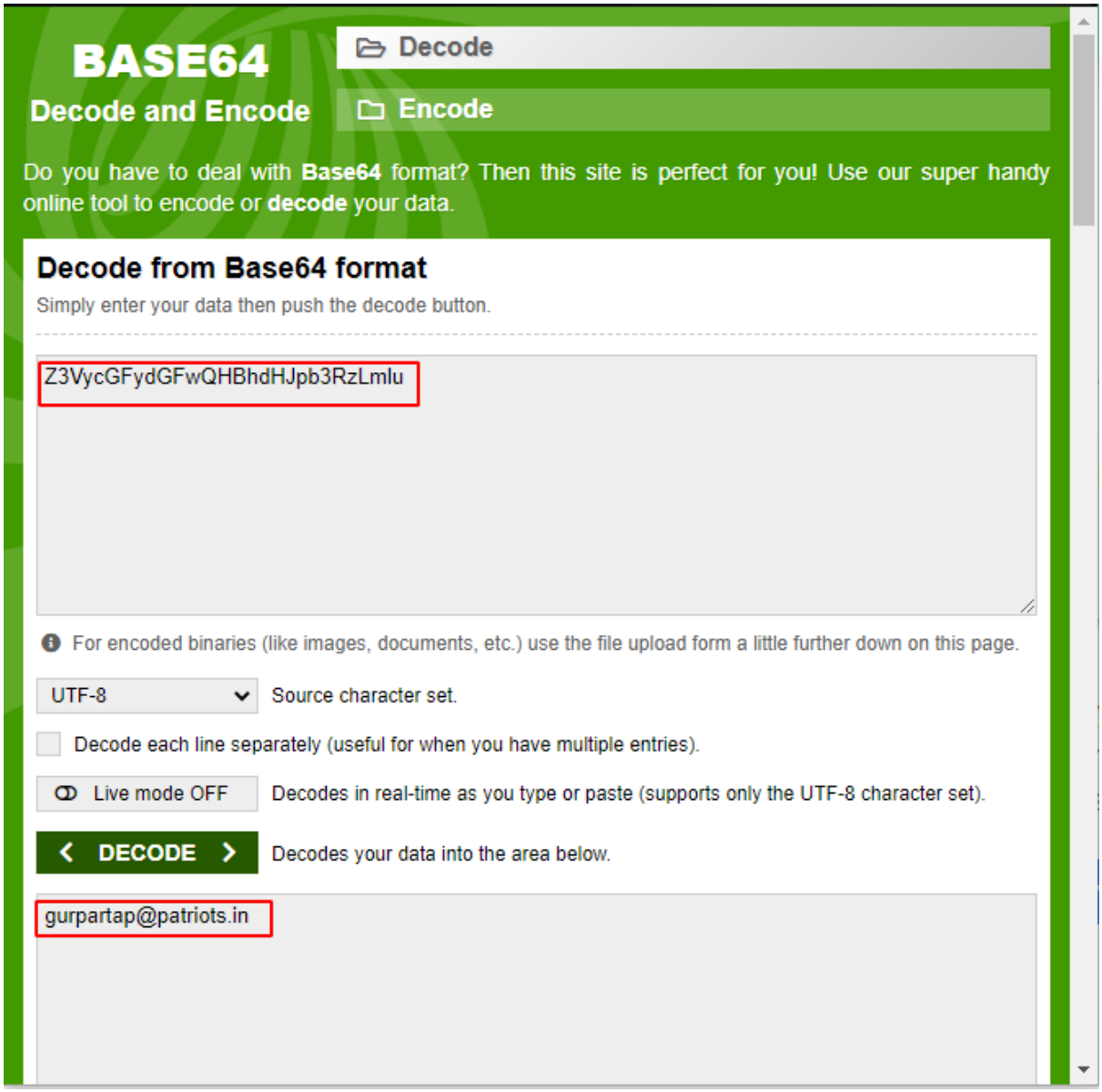
If someone is using plain text authentication during an SMTP transaction, the credentials can be sniffed from a well-positioned attacker. The attacker must only decode the username and password from base64. SMTP uses Base 64 encoding for the transaction to encode the username and password.

A captured SMTP credentials can be seen in the following screenshot with Wireshark and the consequent base64 decoder using the base64 utility.



There are many methods available to decode the base64 strings. For this, I'm using an online tool that is designed specifically for decoding such as base64decode.org or base64decode.net. But we should beware – we may not want to disclose private credentials on the Internet to other parties. In the course of penetration tests and offensive tests, sensitivity and privacy are especially crucial. This is particularly important.

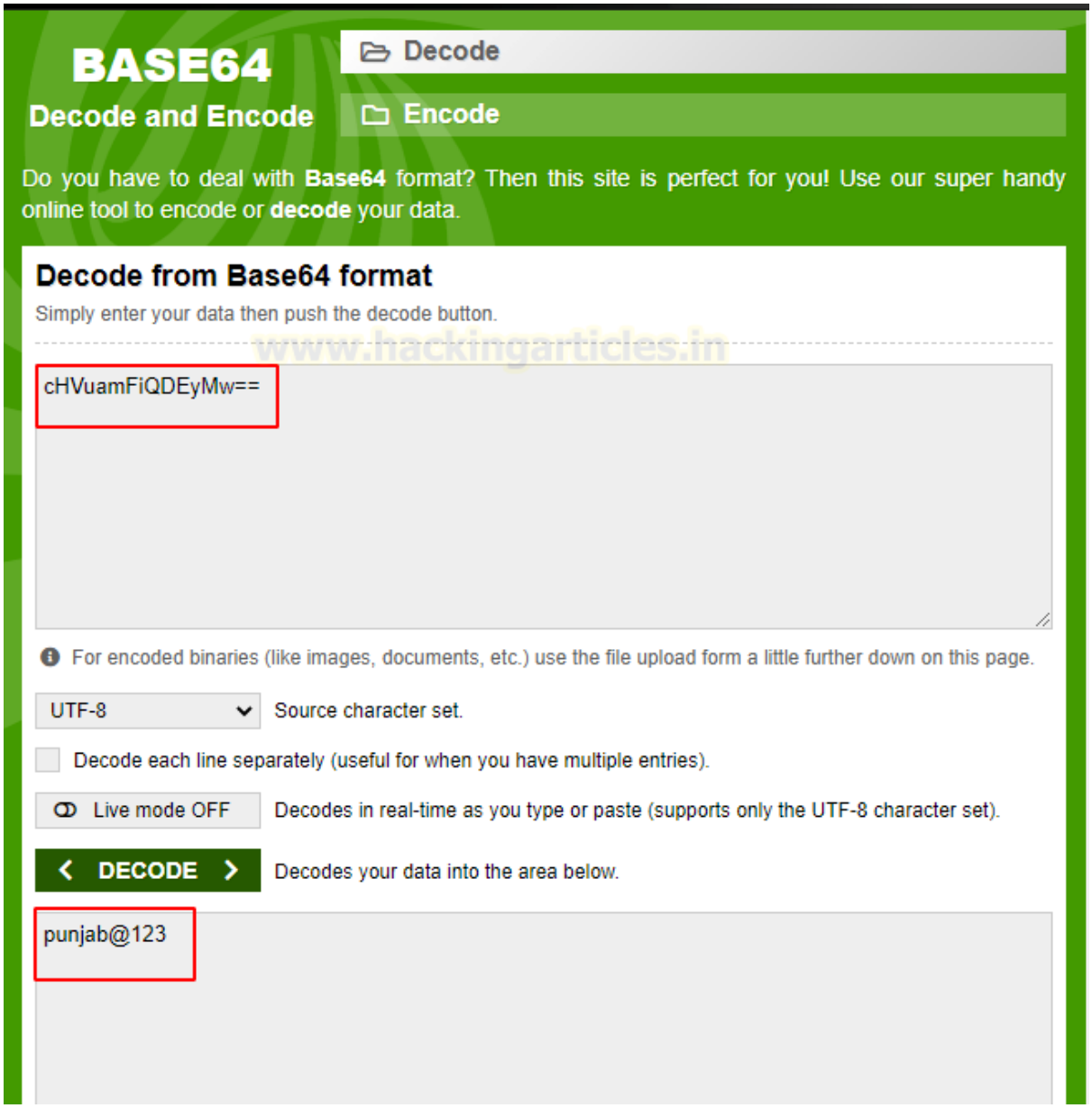
Now, just copy the value of strings of user and password and decode it via base64 decoder as shown below image. As of now, I'm decrypting the user string



User: – Z3VycGFydGFwQHBhdHJpb3RzLmlu

As you can see in the above screenshot, we have successfully able to see the user's name in clear text format. Similarly, we can decrypt the password

Password: – cHVuamFiQDEyMw==



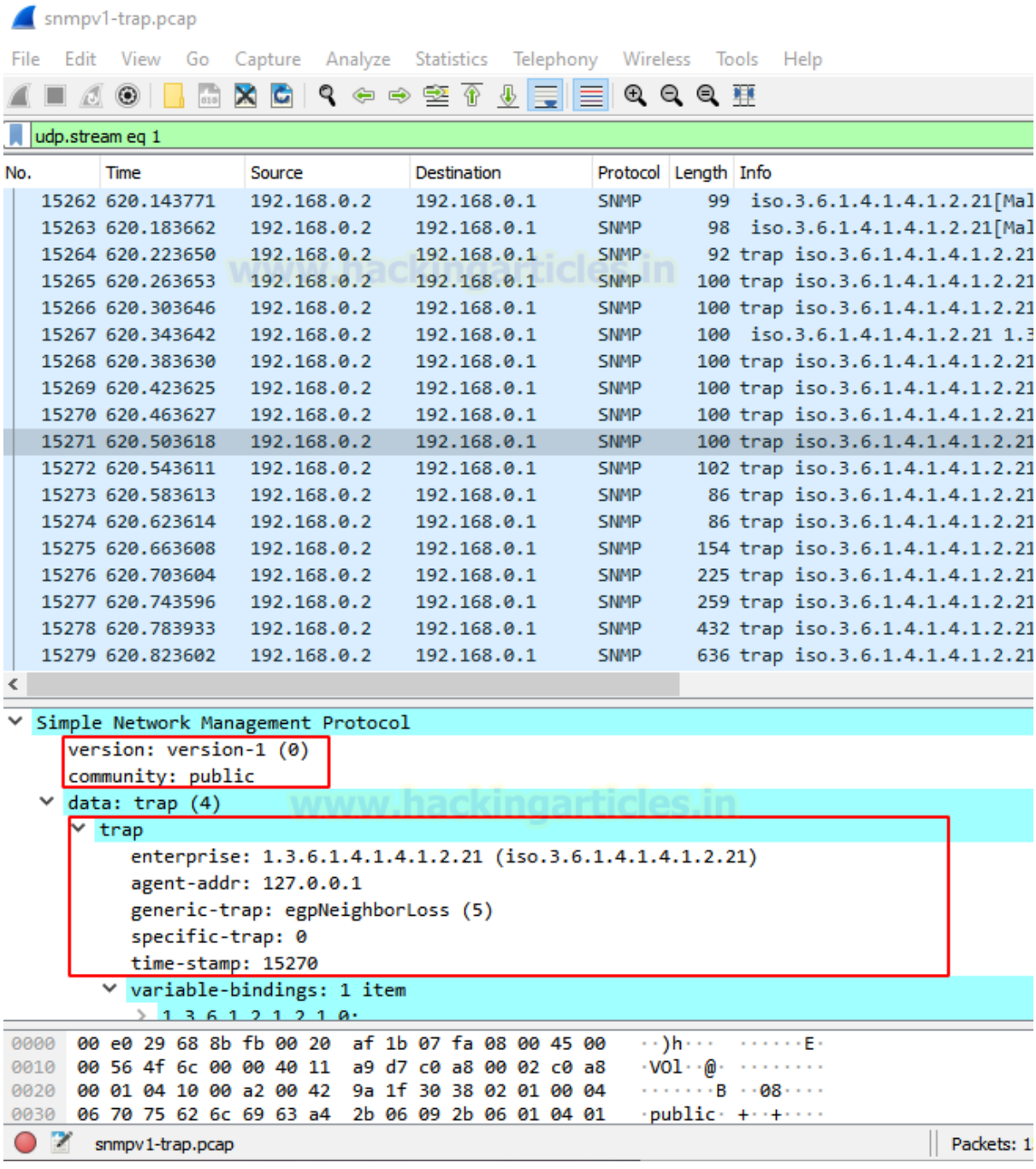
Hurray!!! Now we have got enough credentials to take over a system.

Analyzing SNMP Community String

Simple Network Management Protocol (SNMP) typically runs on port UDP/161. The main objective is network devices and their functions to manage and monitor. SNMP have 3 versions and the first 2 (v1 and v2c) versions are plain text. SNMP uses something that is equivalent to authentication, named community string. Therefore, it is almost the same to capture the SNMP community string as to capture credentials.

While SNMPv3 has been with us for nearly two decades, it takes time. In their internal networks, most organizations still use v1 or v2c. Typically this is due to the backwards compatibility in their networks with legacy systems.

An example of the SNMP community string captured using Wireshark is:

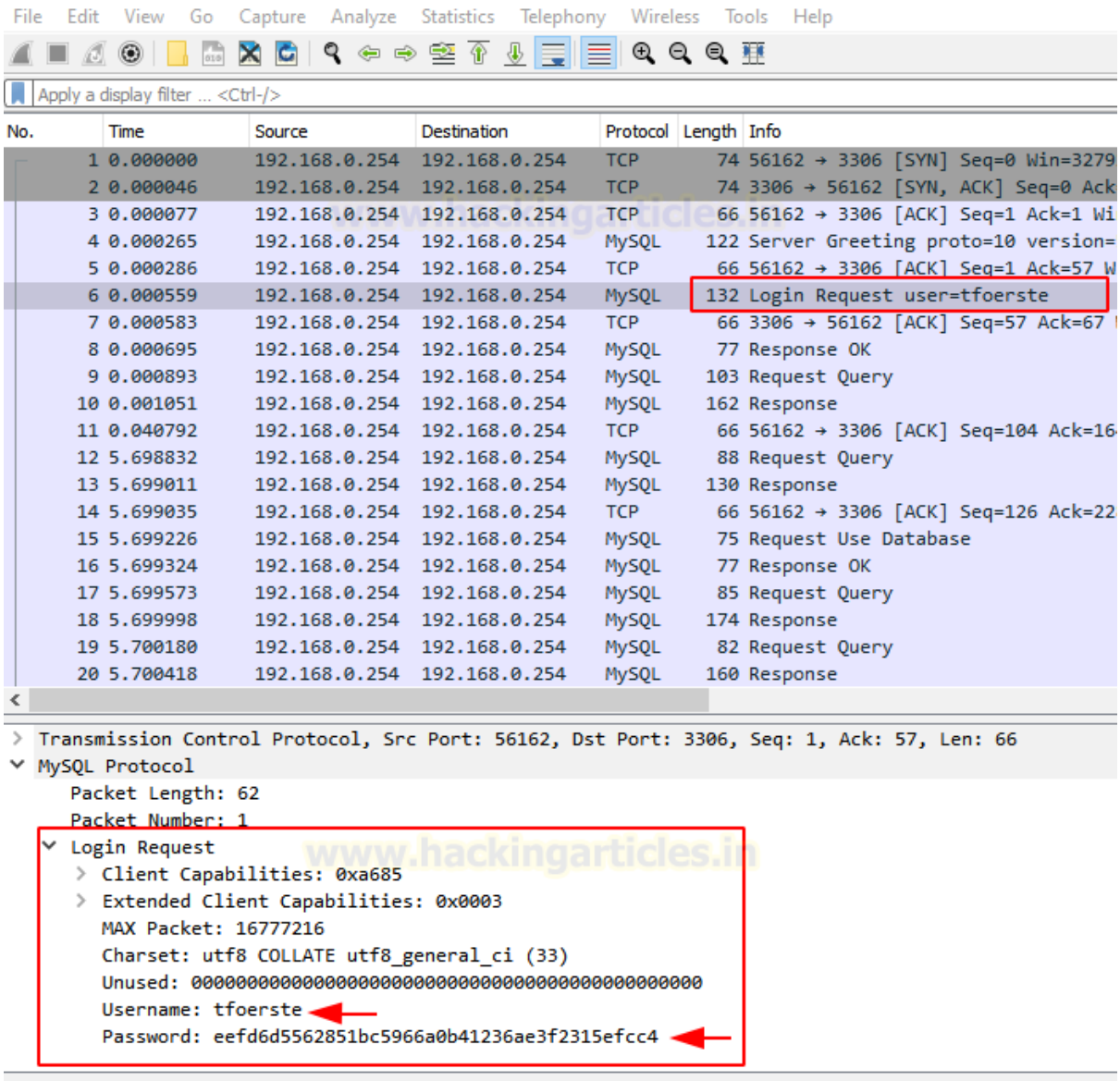


An attacker could now use the community string and collect detailed system information. This could enable the attacker to learn about the system insensitive detail and to make further attempts. Note that the community string sometimes also allows you to modify your remote system configuration (read/write access).

Capture MSSQL Password

The Microsoft SQL server usually runs on TCP/1433 port; this is yet another service we can use with Wireshark to capture the password. If the server is not configured using the ForceEncryption option, it is possible to record plain text authentication directly or via a downgrade attack. MSSQL credentials can be easily captured by a man in the middle.

Here's an example of a Wireshark-captured MSSQL

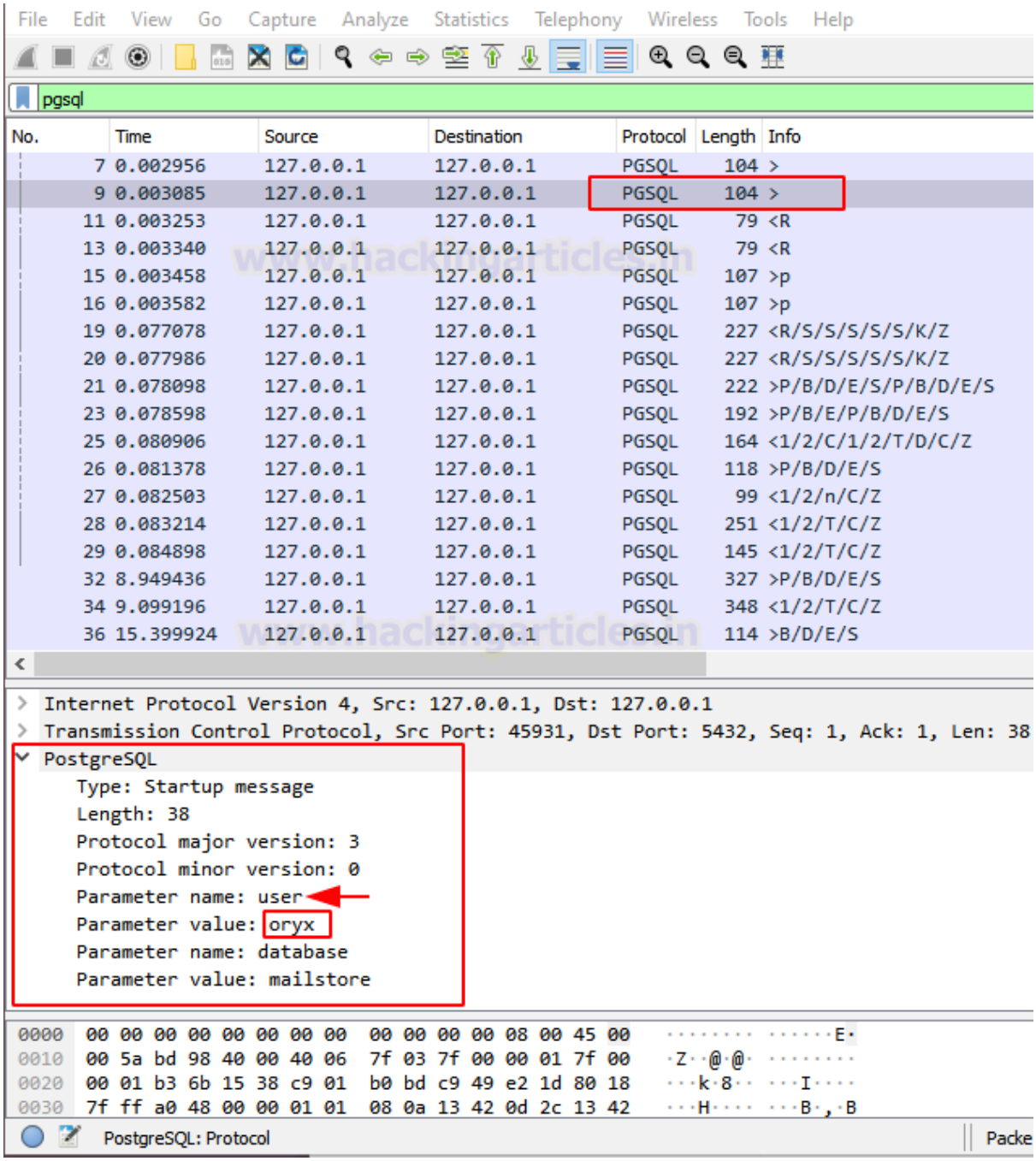


Now, we have a privileged account of the MSSQL server. Therefore, this would have a critical impact allowing the attacker to take complete control over the database server or it could also lead to remote command execution (RCE).

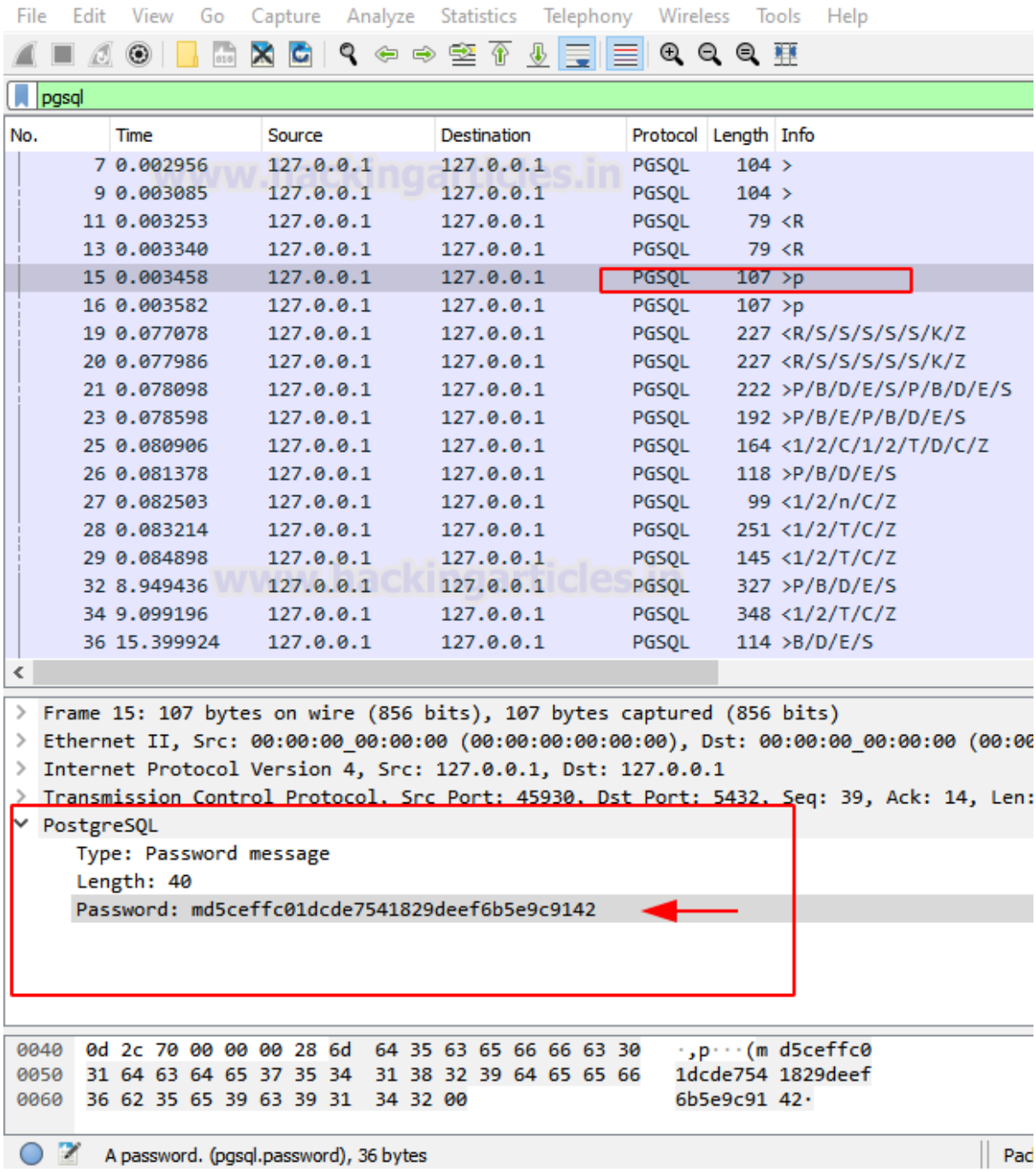
Capture PostgreSQL Password

PostgreSQL is yet another widely used SQL database server. It runs on TCP port 5432 and accepts a variety of authentication methods. It is usually set to disallow clear-text authentication, but it can also be set to allow it. In such cases, a well-positioned attacker could intercept network traffic and obtain the username and password.

It should be noted that PostgreSQL authentication occurs in multiple packets. The username and database name comes first:

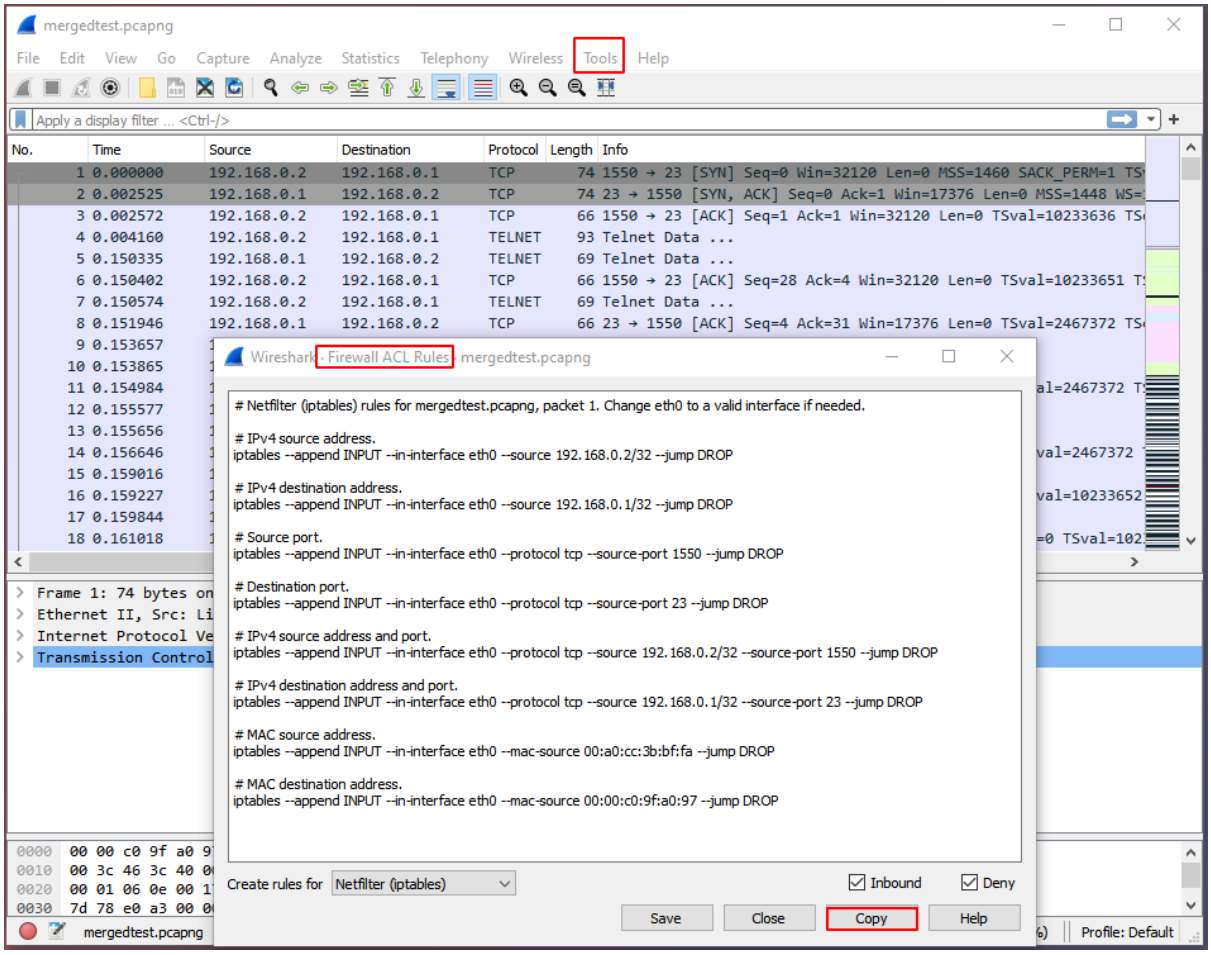


We can also see the PostgreSQL password in the following network packet:



Creating Firewall Rules with Wireshark

Although Wireshark cannot block network traffic, it can assist us in the development of firewall rules for our firewall. Wireshark will create firewall rules based on the traffic we're looking at. To block a packet, all we have to do is pick it and navigate through the menu:



Selected rules can now be copied and pasted directly into our firewall. The following firewalls' syntax is supported by Wireshark:

- Windows Firewall(netsh)
- IP Filter(ipfw)
- NetFilter (iptables)
- Packet Filter(pf)

Conclusion

Wireshark can catch authentication for a wide range of network protocols. There is a possibility as long as we have the ability to eavesdrop on network traffic and the communication is not encrypted. Passwords aren't the only thing that a well-placed attacker can capture; virtually any type of data passing through the network can be captured.

Hence, one can make use of these commands as a cybersecurity professional to assess vulnerabilities on systems and keep these systems away from threat.

References

- https://www.hackingarticles.in/network-packet-forensic-using-wireshark/
- https://www.hackingarticles.in/wireshark-for-pentester-password-sniffing/
- https://www.wireshark.org/docs/