

Wireshark for Pentester: Password Sniffing

April 28, 2021 By Raj Chandel

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 article.

Table of Contents

- Plain text network protocols
- Trace Files
- Capture HTTP Password
- Monitoring HTTPS Packets over SSL or TLS
- Capture Telnet Password
- Capture FTP Password
- Capture SMTP Password
- Analyzing SNMP Community String
- Capture MSSQL Password
- Capture PostgreSQL Password
- Creating Firewall Rules with Wireshark
- Conclusion

Plain text network protocols

So, how is it possible for Wireshark to capture passwords? This is due to the fact that some network protocols do not use encryption. These protocols are referred to as clear text (or plain text) protocols. 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.

Disclaimer: To protect client data, all screenshots have been censored and/or modified.

Trace Files

To, get hands-on with these labs you can download all the trace files from [here](#).

1. Capture HTTP Password
2. Monitoring HTTPS Packets over SSL or TLS
3. Capture Telnet Password
4. Capture FTP Password
5. Capture SMTP Password
6. Analyzing SNMP Community String
7. Capture MSSQL Password
8. Capture PostgreSQL Password

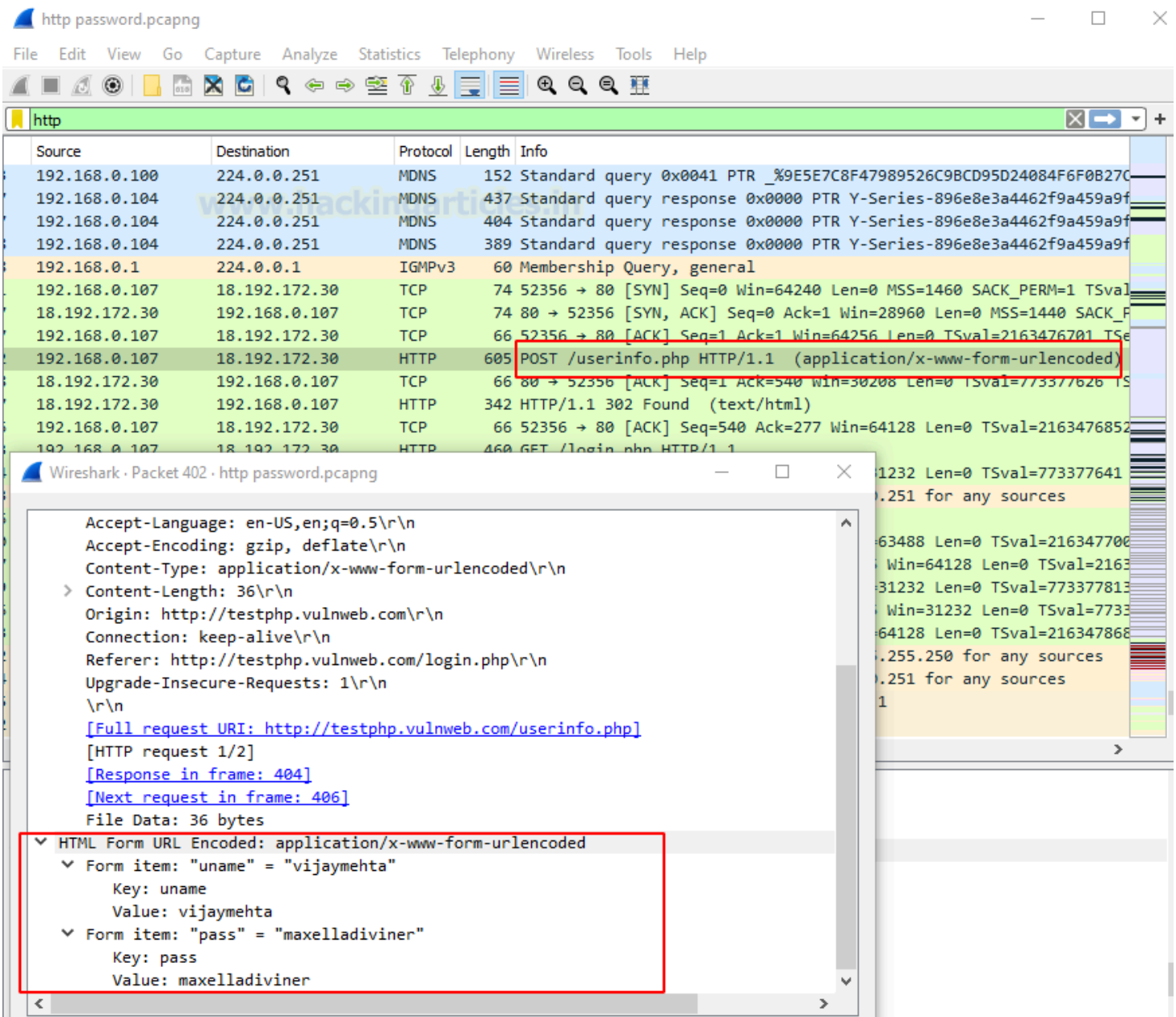
Source of some of the trace files: – [Wireshark.org](https://www.wireshark.org)

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

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	74	38713 → 443 [SYN] Seq=0 Win=32767 Len=0 MSS=16396 SAC
2	0.000021	127.0.0.1	127.0.0.1	TCP	74	443 → 38713 [SYN, ACK] Seq=0 Ack=1 Win=32767 Len=0 MS
3	0.000037	127.0.0.1	127.0.0.1	TCP	66	38713 → 443 [ACK] Seq=1 Ack=1 Win=32767 Len=0 TSval=5
4	0.000158	127.0.0.1	127.0.0.1	SSLv2	171	Client Hello
5	0.000178	127.0.0.1	127.0.0.1	TCP	66	443 → 38713 [ACK] Seq=1 Ack=106 Win=32767 Len=0 TSval
6	0.002160	127.0.0.1	127.0.0.1	SSLv3	995	Server Hello, Certificate, Server Hello Done
7	0.002609	127.0.0.1	127.0.0.1	TCP	66	38713 → 443 [ACK] Seq=106 Ack=930 Win=32767 Len=0 TSv
8	2.808933	127.0.0.1	127.0.0.1	SSLv3	278	Client Key Exchange, Change Cipher Spec, Encrypted Ha
9	2.822770	127.0.0.1	127.0.0.1	SSLv3	141	Change Cipher Spec, Encrypted Handshake Message
10	2.822809	127.0.0.1	127.0.0.1	TCP	66	38713 → 443 [ACK] Seq=318 Ack=1005 Win=32767 Len=0 TS
11	2.833071	127.0.0.1	127.0.0.1	SSLv3	503	Application Data
12	2.873275	127.0.0.1	127.0.0.1	TCP	66	443 → 38713 [ACK] Seq=1005 Ack=755 Win=32767 Len=0 TS
13	2.938485	127.0.0.1	127.0.0.1	SSLv3	103	Encrypted Handshake Message
14	2.938750	127.0.0.1	127.0.0.1	SSLv3	183	Encrypted Handshake Message
15	2.938761	127.0.0.1	127.0.0.1	TCP	66	443 → 38713 [ACK] Seq=1042 Ack=872 Win=32767 Len=0 TS
16	2.938999	127.0.0.1	127.0.0.1	SSLv3	1073	Encrypted Handshake Message, Encrypted Handshake Mess
17	2.940026	127.0.0.1	127.0.0.1	SSLv3	337	Encrypted Handshake Message, Change Cipher Spec, Encr
18	2.943406	127.0.0.1	127.0.0.1	SSLv3	172	Change Cipher Spec, Encrypted Handshake Message
19	2.944825	127.0.0.1	127.0.0.1	SSLv3	5756	Application Data, Application Data

> Frame 11: 503 bytes on wire (4024 bits), 503 bytes captured (4024 bits)

> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 38713, Dst Port: 443, Seq: 318, Ack: 1005, Len: 437

> Transport Layer Security

```

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00  .....E.
0010  01 e9 49 72 40 00 40 06 f1 9a 7f 00 00 01 7f 00  ..Ir@.@.....
0020  00 01 97 39 01 bb 78 8c 3a d4 78 c5 28 c5 80 18  ...9..x..:x(
0030  7f ff ff dd 00 00 01 01 08 0a 1f 53 7c 14 1f 53  .....S|..S
0040  7c 0a 17 03 00 01 b0 4a c3 3e 9d 77 78 01 2c b4  |.....J..>wx,
0050  bc 4c 9a 84 d7 b9 90 0c 21 10 f0 fa 00 7c 16 bb  .L.....!...|
0060  77 fb 72 42 4f ad 50 4a d0 aa 6f aa 44 6c 62 94  w.rBOPJ...oDlb
0070  1b c5 fe e9 1c 5e de 85 0b 0e 05 e4 18 6e d2 d3  .....^.....n

```

rsasnoeol2.cap | Packets: 58 · Displayed: 58 (100.0%) | Profile: Default

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.*

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	74	38713 → 443 [SYN] Seq=0 Win=32767 Len=0
2	0.000021	127.0.0.1	127.0.0.1	TCP	74	443 → 38713 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0
3	0.000037	127.0.0.1	127.0.0.1	TCP	66	38713 → 443 [ACK] Seq=1 Ack=1 Win=32767 Len=0
4	0.000158	127.0.0.1	127.0.0.1	SSLv2	171	Client Hello
5	0.000178	127.0.0.1	127.0.0.1	TCP	66	443 → 38713 [ACK] Seq=1 Ack=106 Win=32767 Len=0
6	0.002160	127.0.0.1	127.0.0.1	SSLv3	995	Server Hello, Certificate, Server Hello Done
7	0.002609	127.0.0.1	127.0.0.1	TCP	66	38713 → 443 [ACK] Seq=106 Ack=930 Win=32767 Len=0
8	2.808933	127.0.0.1	127.0.0.1	SSLv3	278	Client Key Exchange, Change Cipher Spec
9	2.822770	127.0.0.1	127.0.0.1	SSLv3	141	Change Cipher Spec, Encrypted Handshake Message
10	2.822809	127.0.0.1	127.0.0.1	TCP	66	38713 → 443 [ACK] Seq=318 Ack=1005 Win=32767 Len=0
11	2.833071	127.0.0.1	127.0.0.1	SSLv3	503	Application Data
12	2.873275	127.0.0.1	127.0.0.1	TCP	66	443 → 38713 [ACK] Seq=1005 Ack=755 Win=32767 Len=0
13	2.938485	127.0.0.1	127.0.0.1	SSLv3	103	Encrypted Handshake Message
14	2.938750	127.0.0.1	127.0.0.1	SSLv3	183	Encrypted Handshake Message
15	2.938761	127.0.0.1	127.0.0.1	TCP	66	443 → 38713 [ACK] Seq=1042 Ack=872 Win=32767 Len=0
16	2.938999	127.0.0.1	127.0.0.1	SSLv3	1073	Encrypted Handshake Message, Encrypted Handshake Message
17	2.940026	127.0.0.1	127.0.0.1	SSLv3	337	Encrypted Handshake Message, Change Cipher Spec
18	2.943406	127.0.0.1	127.0.0.1	SSLv3	172	Change Cipher Spec, Encrypted Handshake Message
19	2.944825	127.0.0.1	127.0.0.1	SSLv3	5756	Application Data, Application Data

<

> Frame 11: 503 bytes on wire (4024 bits), 503 bytes captured (4024 bits)

> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 38713, Dst Port: 443, Seq: 318, Ack: 1005, Len: 437

▼ Transport Layer Security

▼ SSLv3 Record Layer: Application Data Protocol: Application Data

Content Type: Application Data (23)

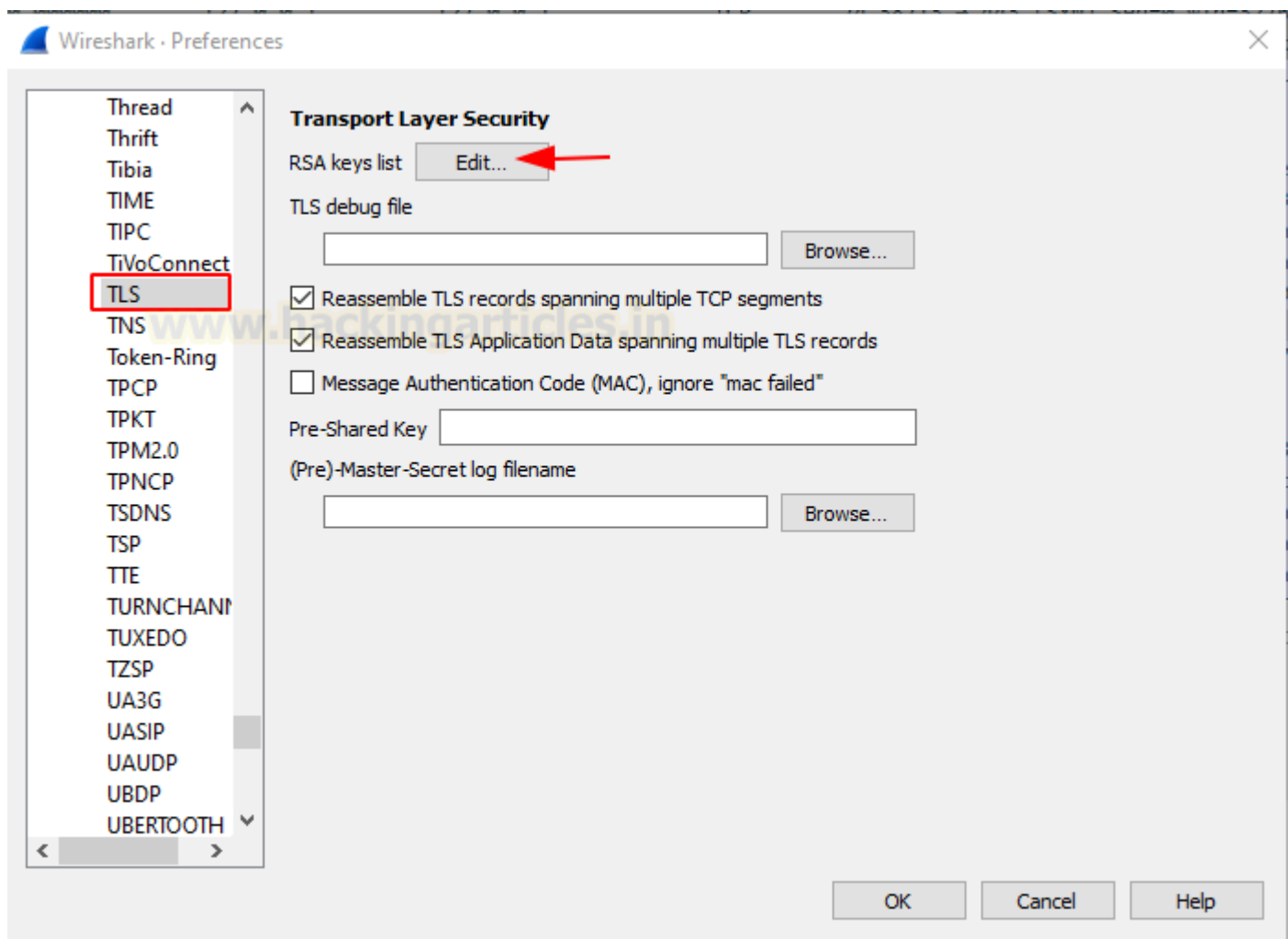
Version: SSL 3.0 (0x0300)

Length: 432

Encrypted Application Data: 4ac33e9d7778012cb4bc4c9a84d7b9900c2110f0fa007c16bb77fb72424fad504ad0aa6f...

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

Edit > Preference > Protocol > TLS

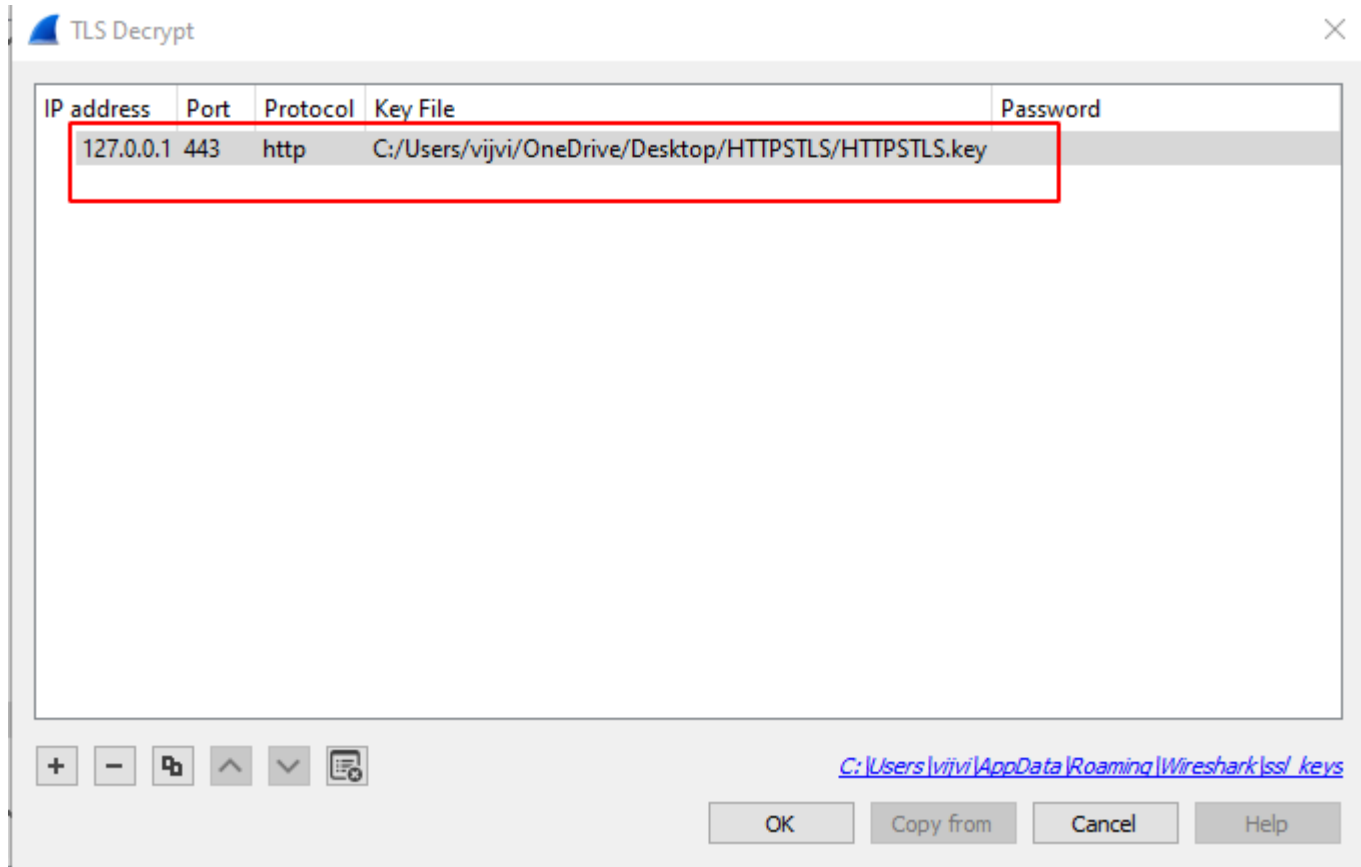


And add these values

IP address: 127.0.0.1

Port: 443

Key File:



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

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
26	2.964819	127.0.0.1	127.0.0.1	TCP	66	443 → 38714 [ACK] Seq=1 Ack=121 Win=32767 Len=0
27	2.992274	127.0.0.1	127.0.0.1	SSLv3	220	Server Hello, Change Cipher Spec, Finished
28	2.992312	127.0.0.1	127.0.0.1	TCP	66	38714 → 443 [ACK] Seq=121 Ack=155 Win=32767 Len=0
29	2.992855	127.0.0.1	127.0.0.1	HTTP	562	GET /icons/debian/openlogo-25.jpg HTTP/1.1
30	2.993501	127.0.0.1	127.0.0.1	HTTP	596	HTTP/1.1 404 Not Found (text/html)
31	2.993840	127.0.0.1	127.0.0.1	HTTP	471	GET /icons/apache_pb.png HTTP/1.1
32	2.994179	127.0.0.1	127.0.0.1	HTTP	1828	HTTP/1.1 200 OK (PNG)
33	3.004256	127.0.0.1	127.0.0.1	TCP	66	443 → 38713 [ACK] Seq=7845 Ack=1548 Win=32767 Len=0
34	3.033250	127.0.0.1	127.0.0.1	TCP	66	38714 → 443 [ACK] Seq=1022 Ack=2447 Win=32767 Len=0
35	3.501643	127.0.0.1	127.0.0.1	HTTP	588	HTTP/1.1 404 Not Found (text/html)
36	3.507001	127.0.0.1	127.0.0.1	HTTP	439	GET /favicon.ico HTTP/1.1
37	3.507541	127.0.0.1	127.0.0.1	HTTP	580	HTTP/1.1 404 Not Found (text/html)
38	3.507555	127.0.0.1	127.0.0.1	TCP	66	38714 → 443 [ACK] Seq=1395 Ack=2961 Win=32767 Len=0
39	3.541174	127.0.0.1	127.0.0.1	TCP	66	38713 → 443 [ACK] Seq=1548 Ack=8367 Win=32767 Len=0
40	6.037880	127.0.0.1	127.0.0.1	HTTP	511	GET /test HTTP/1.1
41	6.037932	127.0.0.1	127.0.0.1	TCP	66	443 → 38713 [ACK] Seq=8367 Ack=1993 Win=32767 Len=0
42	6.041185	127.0.0.1	127.0.0.1	HTTP	644	HTTP/1.1 301 Moved Permanently (text/html)
43	6.041367	127.0.0.1	127.0.0.1	TCP	66	38713 → 443 [ACK] Seq=1993 Ack=8945 Win=32767 Len=0
44	6.088943	127.0.0.1	127.0.0.1	HTTP	511	GET /test/ HTTP/1.1

Transport Layer Security

- SSLv3 Record Layer: Application Data Protocol: http-over-tls
 - Content Type: Application Data (23)
 - Version: SSL 3.0 (0x0300)
 - Length: 344
 - Encrypted Application Data: 1aab72a99faeed998838fdc3f82627082e5b3aa7b4b71158317031287eba13cbdc3adbdd...
 - [Application Data Protocol: http-over-tls]
 - TLS segment data (317 bytes)
- SSLv3 Record Layer: Application Data Protocol: http-over-tls
 - Content Type: Application Data (23)
 - Version: SSL 3.0 (0x0300)
 - Length: 1408
 - Encrypted Application Data: 1fddf32ae5f46e000f438000e67e46e68e46e00003f30f47b8ed00001e45061558b302...

0000 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK
 0010 0a 44 61 74 65 3a 20 4d 6f 6e 2c 20 32 34 20 41 Date: Mon, 24 Aug 2010 12:00:00 GMT

Frame (1828 bytes) Decrypted TLS (317 bytes) Decrypted TLS (1385 bytes) Reassembled SSL (1702 bytes)

Capture Telnet Password

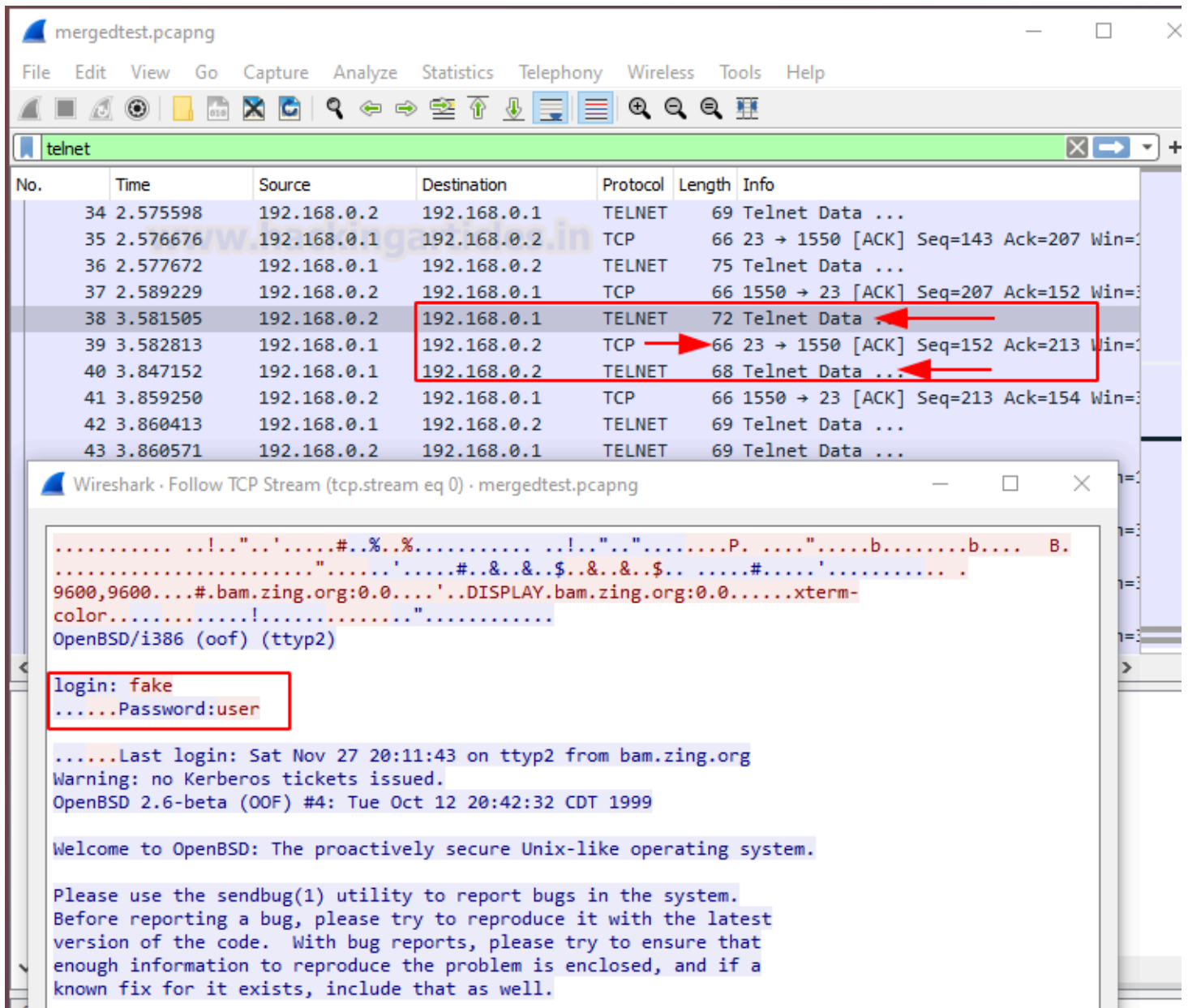
No introduction is required for Telnet protocol using port tcp/23. It is mainly used for administrative convenience and is known for its insecurity. Since encryption is not available, privacy or unauthorized access protection is not available. Telnet is still used today, however...

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:

The image shows a Wireshark packet capture of an FTP session. The packet list pane shows several packets, with packet 18 selected. The packet details pane shows the structure of the selected packet, which is an FTP 'PASS' command. The 'PASS' command is highlighted with a red box, and the password 'jowens@yccc.edu' is also highlighted with a red box. The packet list pane shows the following packets:

No.	Time	Source	Destination	Protocol	Length	Info
9	0.788801	10.0.1.100	198.145.20.140	TCP	78	55919 → 21 [SYN] Seq=0 Win=65535 Len=0 MSS=14
10	0.901830	198.145.20.140	10.0.1.100	TCP	74	21 → 55919 [SYN, ACK] Seq=0 Ack=1 Win=14480 L
11	0.902022	10.0.1.100	198.145.20.140	TCP	66	55919 → 21 [ACK] Seq=1 Ack=1 Win=132480 Len=0
12	1.020035	198.145.20.140	10.0.1.100	FTP	93	Response: 220 Welcome to kernel.org
13	1.020201	10.0.1.100	198.145.20.140	TCP	66	55919 → 21 [ACK] Seq=1 Ack=28 Win=132448 Len=
14	12.651747	10.0.1.100	198.145.20.140	FTP	76	Request: USER ftp
15	12.784279	198.145.20.140	10.0.1.100	TCP	66	21 → 55919 [ACK] Seq=28 Ack=11 Win=14592 Len=
16	12.784290	198.145.20.140	10.0.1.100	FTP	100	Response: 331 Please specify the password.
17	12.784481	10.0.1.100	198.145.20.140	TCP	66	55919 → 21 [ACK] Seq=11 Ack=62 Win=132416 Len
18	19.293792	10.0.1.100	198.145.20.140	FTP	88	Request: PASS jowens@yccc.edu
19	19.400899	198.145.20.140	10.0.1.100	FTP	89	Response: 230 Login successful.
20	19.401116	10.0.1.100	198.145.20.140	TCP	66	55919 → 21 [ACK] Seq=33 Ack=85 Win=132392 Len
21	19.401370	10.0.1.100	198.145.20.140	FTP	72	Request: SYST
22	19.523265	198.145.20.140	10.0.1.100	FTP	85	Response: 215 UNIX Type: L8
23	19.523444	10.0.1.100	198.145.20.140	TCP	66	55919 → 21 [ACK] Seq=39 Ack=104 Win=132376 Le
24	19.523671	10.0.1.100	198.145.20.140	FTP	72	Request: FEAT
25	19.615248	198.145.20.140	10.0.1.100	FTP	81	Response: 211-Features:

The packet details pane for packet 18 shows the following structure:

- > Frame 18: 88 bytes on wire (704 bits), 88 bytes captured (704 bits)
- > Ethernet II, Src: Apple_23:15:a4 (f0:b4:79:23:15:a4), Dst: Cisco-Li_fe:bd:f9 (00:23:69:fe:bd:f9)
- > Internet Protocol Version 4, Src: 10.0.1.100, Dst: 198.145.20.140
- > Transmission Control Protocol, Src Port: 55919, Dst Port: 21, Seq: 11, Ack: 62, Len: 22
- ▼ File Transfer Protocol (FTP)
 - ▼ PASS jowens@yccc.edu\r\n
 - Request command: PASS
 - Request arg: jowens@yccc.edu
 - [Current working directory:]

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.

test 1.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

smtp

No.	Time	Source	Destination	Protocol	Length	Info
1584...	310967609.832...	74.53.140.153	10.10.1.4	SMTP	235	S: 220-xc90.websitewelcome.com ESMTP Exim 4
1584...	310967609.837...	10.10.1.4	74.53.140.153	SMTP	63	C: EHLO GP
1584...	310967610.178...	74.53.140.153	10.10.1.4	SMTP	191	S: 250-xc90.websitewelcome.com Hello GP [1
1584...	310967610.181...	10.10.1.4	74.53.140.153	SMTP	66	C: AUTH LOGIN
1584...	310967610.523...	74.53.140.153	10.10.1.4	SMTP	72	S: 334 VXNlcm5hbWU6
1584...	310967610.524...	10.10.1.4	74.53.140.153	SMTP	84	C: User: Z3VycGFydGFwQHhhdHJpb3RzLmlu
1584...	310967610.866...	74.53.140.153	10.10.1.4	SMTP	72	S: 334 UGFzc3dvcmQ6
1584...	310967610.866...	10.10.1.4	74.53.140.153	SMTP	72	C: Pass: cHVuamFiQDEyMw==
1584...	310967611.226...	74.53.140.153	10.10.1.4	SMTP	84	S: 235 Authentication succeeded
1584...	310967611.227...	10.10.1.4	74.53.140.153	SMTP	90	C: MAIL FROM: <gurpartap@patriots.in>
1584...	310967611.569...	74.53.140.153	10.10.1.4	SMTP	62	S: 250 OK
1584...	310967611.570...	10.10.1.4	74.53.140.153	SMTP	93	C: RCPT TO: <raj_deol2002in@yahoo.co.in>
1584...	310967611.932...	74.53.140.153	10.10.1.4	SMTP	68	S: 250 Accepted
1584...	310967611.933...	10.10.1.4	74.53.140.153	SMTP	60	C: DATA
1584...	310967612.274...	74.53.140.153	10.10.1.4	SMTP	110	S: 354 Enter message, ending with "." on a
1584...	310967612.305...	10.10.1.4	74.53.140.153	SMTP	1514	S: DATA fragment, 1460 bytes
1584...	310967612.305...	10.10.1.4	74.53.140.153	SMTP	1514	C: DATA fragment, 1460 bytes
1584...	310967612.305...	10.10.1.4	74.53.140.153	SMTP	1514	C: DATA fragment, 1460 bytes
1584...	310967612.305...	10.10.1.4	74.53.140.153	SMTP	1514	[TCP Window Full] C: DATA fragment, 1460 by
1584...	310967612.307...	192.168.1.1	10.10.1.4	ICMP	590	Destination unreachable (Fragmentation need

<

> Frame 158406: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)

> Ethernet II, Src: Cradlepo_3c:17:c2 (00:e0:1c:3c:17:c2), Dst: Netgear_d9:81:60 (00:1f:33:d9:81:60)

> Internet Protocol Version 4, Src: 10.10.1.4, Dst: 74.53.140.153

> Transmission Control Protocol, Src Port: 1470, Dst Port: 25, Seq: 22, Ack: 337, Len: 30

Simple Mail Transfer Protocol

Username: Z3VycGFydGFwQHhhdHJpb3RzLmlu

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

The screenshot shows a web application titled "BASE64" with a green header. It features two main buttons: "Decode" and "Encode". Below the header, a message states: "Do you have to deal with Base64 format? Then this site is perfect for you! Use our super handy online tool to encode or decode your data." The main section is titled "Decode from Base64 format" and includes the instruction "Simply enter your data then push the decode button." There are two large text input areas. The first area contains the Base64 string "Z3VycGFydGFwQHBhdHJpb3RzLmlu", which is highlighted with a red box. Below this area, there is an information icon and text: "For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page." Below that is a dropdown menu set to "UTF-8" with the label "Source character set." and a checkbox labeled "Decode each line separately (useful for when you have multiple entries)." There is also a toggle switch for "Live mode OFF" with the text "Decodes in real-time as you type or paste (supports only the UTF-8 character set)." A green button labeled "< DECODE >" is present, with the text "Decodes your data into the area below." below it. The second text input area contains the decoded text "gurpartap@patriots.in", also highlighted with a red box.

BASE64

Decode and Encode

Decode Encode

Do you have to deal with **Base64** format? Then this site is perfect for you! Use our super handy online tool to encode or **decode** your data.

Decode from Base64 format

Simply enter your data then push the decode button.

Z3VycGFydGFwQHBhdHJpb3RzLmlu

i For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

☒ Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

< DECODE > Decodes your data into the area below.

gurpartap@patriots.in

User: – Z3VycGFydGFwQHBhdHJpb3RzLmlu

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

Password: – cHVuamFiQDEyMw==

BASE64

Decode and Encode

DecodeEncode

Do you have to deal with **Base64** format? Then this site is perfect for you! Use our super handy online tool to encode or **decode** your data.

Decode from Base64 format

Simply enter your data then push the decode button.

cHVuamFiQDEyMw==

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8

Source character set.

☐

Decode each line separately (useful for when you have multiple entries).

☒ Live mode OFF

Decodes in real-time as you type or paste (supports only the UTF-8 character set).

< **DECODE** >

Decodes your data into the area below.

punjab@123

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:

The image shows a Wireshark packet capture of an SNMP trap. The top pane displays a list of packets, with packet 15270 selected. The middle pane shows the packet details for the selected packet, highlighting the 'community: public' field. The bottom pane shows the raw packet data in hexadecimal and ASCII format.

No.	Time	Source	Destination	Protocol	Length	Info
15262	620.143771	192.168.0.2	192.168.0.1	SNMP	99	iso.3.6.1.4.1.4.1.2.21[Ma]
15263	620.183662	192.168.0.2	192.168.0.1	SNMP	98	iso.3.6.1.4.1.4.1.2.21[Ma]
15264	620.223650	192.168.0.2	192.168.0.1	SNMP	92	trap iso.3.6.1.4.1.4.1.2.21
15265	620.263653	192.168.0.2	192.168.0.1	SNMP	100	trap iso.3.6.1.4.1.4.1.2.21
15266	620.303646	192.168.0.2	192.168.0.1	SNMP	100	trap iso.3.6.1.4.1.4.1.2.21
15267	620.343642	192.168.0.2	192.168.0.1	SNMP	100	iso.3.6.1.4.1.4.1.2.21 1.3
15268	620.383630	192.168.0.2	192.168.0.1	SNMP	100	trap iso.3.6.1.4.1.4.1.2.21
15269	620.423625	192.168.0.2	192.168.0.1	SNMP	100	trap iso.3.6.1.4.1.4.1.2.21
15270	620.463627	192.168.0.2	192.168.0.1	SNMP	100	trap iso.3.6.1.4.1.4.1.2.21
15271	620.503618	192.168.0.2	192.168.0.1	SNMP	100	trap iso.3.6.1.4.1.4.1.2.21
15272	620.543611	192.168.0.2	192.168.0.1	SNMP	102	trap iso.3.6.1.4.1.4.1.2.21
15273	620.583613	192.168.0.2	192.168.0.1	SNMP	86	trap iso.3.6.1.4.1.4.1.2.21
15274	620.623614	192.168.0.2	192.168.0.1	SNMP	86	trap iso.3.6.1.4.1.4.1.2.21
15275	620.663608	192.168.0.2	192.168.0.1	SNMP	154	trap iso.3.6.1.4.1.4.1.2.21
15276	620.703604	192.168.0.2	192.168.0.1	SNMP	225	trap iso.3.6.1.4.1.4.1.2.21
15277	620.743596	192.168.0.2	192.168.0.1	SNMP	259	trap iso.3.6.1.4.1.4.1.2.21
15278	620.783933	192.168.0.2	192.168.0.1	SNMP	432	trap iso.3.6.1.4.1.4.1.2.21
15279	620.823602	192.168.0.2	192.168.0.1	SNMP	636	trap iso.3.6.1.4.1.4.1.2.21

Simple Network Management Protocol

- version: version-1 (0)
- community: public
- data: trap (4)
 - trap
 - enterprise: 1.3.6.1.4.1.4.1.2.21 (iso.3.6.1.4.1.4.1.2.21)
 - agent-addr: 127.0.0.1
 - generic-trap: egpNeighborLoss (5)
 - specific-trap: 0
 - time-stamp: 15270
 - variable-bindings: 1 item
 - 1 3 6 1 2 1 2 1 0

0000 00 e0 29 68 8b fb 00 20 af 1b 07 fa 08 00 45 00 ..)h... ..E.
0010 00 56 4f 6c 00 00 40 11 a9 d7 c0 a8 00 02 c0 a8 .VOL..@.
0020 00 01 04 10 00 a2 00 42 9a 1f 30 38 02 01 00 04B ..08....
0030 06 70 75 62 6c 69 63 a4 2b 06 09 2b 06 01 04 01 .public. ++....

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

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).

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:

The image shows a Wireshark network traffic capture of PostgreSQL communication. The top pane displays a list of packets, with packet 9 highlighted. The bottom pane shows the details of packet 9, which is a PostgreSQL Startup message. The details pane is expanded, showing the following information:

- Type: Startup message
- Length: 38
- Protocol major version: 3
- Protocol minor version: 0
- Parameter name: user
- Parameter value: oryx
- Parameter name: database
- Parameter value: mailstore

The packet data is also visible in the bottom pane, showing the raw bytes and their hexadecimal representation.

No.	Time	Source	Destination	Protocol	Length	Info
7	0.002956	127.0.0.1	127.0.0.1	PGSQL	104	>
9	0.003085	127.0.0.1	127.0.0.1	PGSQL	104	>
11	0.003253	127.0.0.1	127.0.0.1	PGSQL	79	<R
13	0.003340	127.0.0.1	127.0.0.1	PGSQL	79	<R
15	0.003458	127.0.0.1	127.0.0.1	PGSQL	107	>p
16	0.003582	127.0.0.1	127.0.0.1	PGSQL	107	>p
19	0.077078	127.0.0.1	127.0.0.1	PGSQL	227	<R/S/S/S/S/S/K/Z
20	0.077986	127.0.0.1	127.0.0.1	PGSQL	227	<R/S/S/S/S/S/K/Z
21	0.078098	127.0.0.1	127.0.0.1	PGSQL	222	>P/B/D/E/S/P/B/D/E/S
23	0.078598	127.0.0.1	127.0.0.1	PGSQL	192	>P/B/E/P/B/D/E/S
25	0.080906	127.0.0.1	127.0.0.1	PGSQL	164	<1/2/C/1/2/T/D/C/Z
26	0.081378	127.0.0.1	127.0.0.1	PGSQL	118	>P/B/D/E/S
27	0.082503	127.0.0.1	127.0.0.1	PGSQL	99	<1/2/n/C/Z
28	0.083214	127.0.0.1	127.0.0.1	PGSQL	251	<1/2/T/C/Z
29	0.084898	127.0.0.1	127.0.0.1	PGSQL	145	<1/2/T/C/Z
32	8.949436	127.0.0.1	127.0.0.1	PGSQL	327	>P/B/D/E/S
34	9.099196	127.0.0.1	127.0.0.1	PGSQL	348	<1/2/T/C/Z
36	15.399924	127.0.0.1	127.0.0.1	PGSQL	114	>B/D/E/S

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 45931, Dst Port: 5432, Seq: 1, Ack: 1, Len: 38
PostgreSQL
Type: Startup message
Length: 38
Protocol major version: 3
Protocol minor version: 0
Parameter name: user
Parameter value: oryx
Parameter name: database
Parameter value: mailstore

0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00E.
0010 00 5a bd 98 40 00 40 06 7f 03 7f 00 00 01 7f 00 .Z..@.@.....
0020 00 01 b3 6b 15 38 c9 01 b0 bd c9 49 e2 1d 80 18 ...k.8...I....
0030 7f ff a0 48 00 00 01 01 08 0a 13 42 0d 2c 13 42 ...H....B.,B

PostgreSQL: Protocol

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

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

pgsql

No.	Time	Source	Destination	Protocol	Length	Info
7	0.002956	127.0.0.1	127.0.0.1	PGSQL	104	>
9	0.003085	127.0.0.1	127.0.0.1	PGSQL	104	>
11	0.003253	127.0.0.1	127.0.0.1	PGSQL	79	<R
13	0.003340	127.0.0.1	127.0.0.1	PGSQL	79	<R
15	0.003458	127.0.0.1	127.0.0.1	PGSQL	107	>p
16	0.003582	127.0.0.1	127.0.0.1	PGSQL	107	>p
19	0.077078	127.0.0.1	127.0.0.1	PGSQL	227	<R/S/S/S/S/S/K/Z
20	0.077986	127.0.0.1	127.0.0.1	PGSQL	227	<R/S/S/S/S/S/K/Z
21	0.078098	127.0.0.1	127.0.0.1	PGSQL	222	>P/B/D/E/S/P/B/D/E/S
23	0.078598	127.0.0.1	127.0.0.1	PGSQL	192	>P/B/E/P/B/D/E/S
25	0.080906	127.0.0.1	127.0.0.1	PGSQL	164	<1/2/C/1/2/T/D/C/Z
26	0.081378	127.0.0.1	127.0.0.1	PGSQL	118	>P/B/D/E/S
27	0.082503	127.0.0.1	127.0.0.1	PGSQL	99	<1/2/n/C/Z
28	0.083214	127.0.0.1	127.0.0.1	PGSQL	251	<1/2/T/C/Z
29	0.084898	127.0.0.1	127.0.0.1	PGSQL	145	<1/2/T/C/Z
32	8.949436	127.0.0.1	127.0.0.1	PGSQL	327	>P/B/D/E/S
34	9.099196	127.0.0.1	127.0.0.1	PGSQL	348	<1/2/T/C/Z
36	15.399924	127.0.0.1	127.0.0.1	PGSQL	114	>B/D/E/S

<

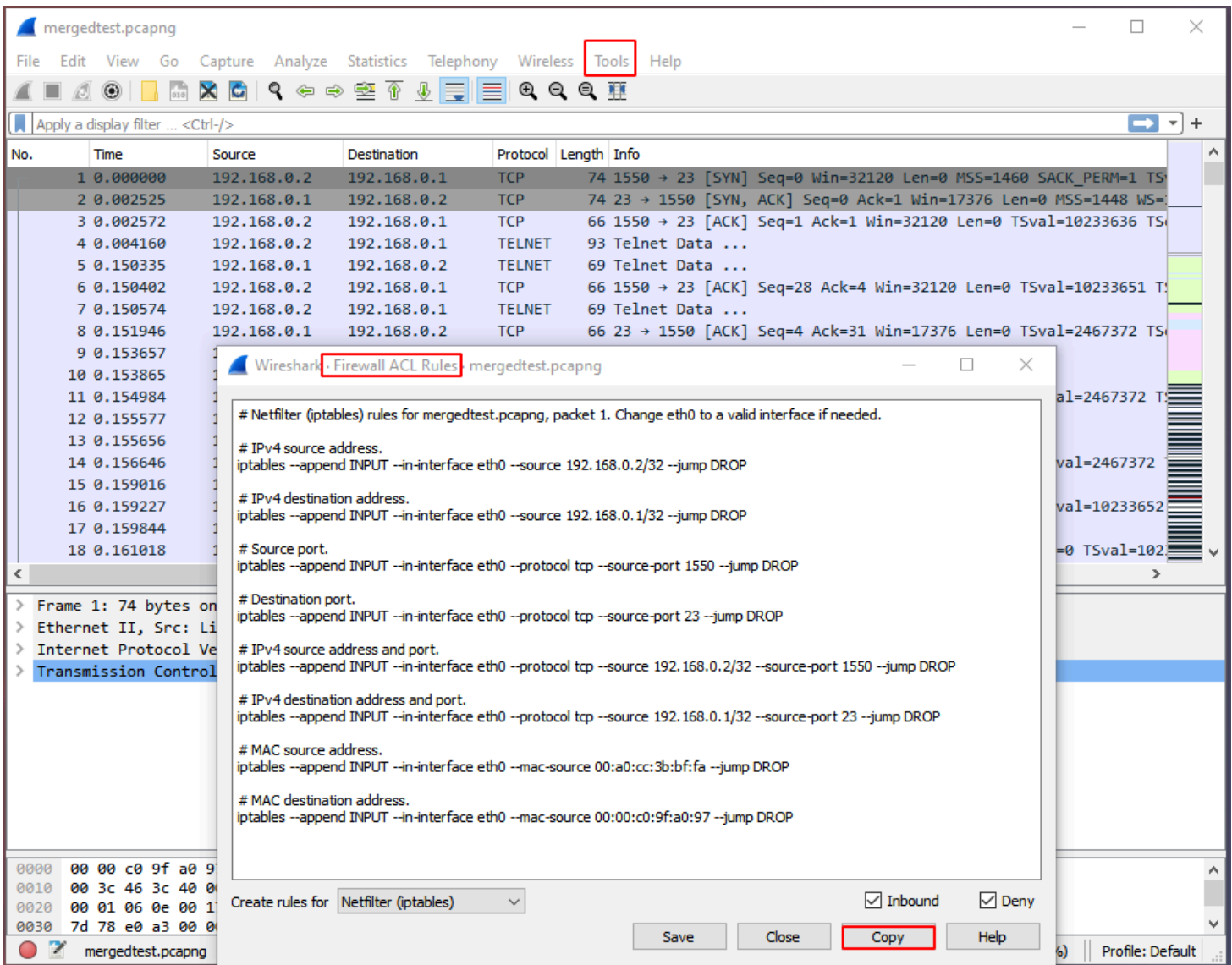
> Frame 15: 107 bytes on wire (856 bits), 107 bytes captured (856 bits)
 > Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
 > Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 > Transmission Control Protocol, Src Port: 45930, Dst Port: 5432, Seq: 39, Ack: 14, Len: 107
 > PostgreSQL
 Type: Password message
 Length: 40
 Password: md5ceffc01dcde7541829deef6b5e9c9142

0040 0d 2c 70 00 00 00 28 6d 64 35 63 65 66 66 63 30 .,p... (m d5ceffc0
 0050 31 64 63 64 65 37 35 34 31 38 32 39 64 65 65 66 1dcde754 1829deef
 0060 36 62 35 65 39 63 39 31 34 32 00 6b5e9c91 42.

A password. (pgsql.password), 36 bytes

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.