STREAM SERIES

eplace(

funct on re

y(b[b.le

-1 < b &/ a.

devnator

Security Comparison SSH and Telnet Protocols

SERGIO CABRAL

});



sergiocabral.com

Topics

1. Analyzing Telnet packets	
2. Analyzing SSH packets	
3. Conclusion	5
4. Network sniffers options	
5. Video demonstration	7

Practical purpose of this demonstration

Verificar com um analisador de pacotes de redes (um sniffer) a criptografia presente via SSH e ausente via Telnet. Justificar o uso do protocolo SSH sempre que possível ao invés do protocolo Telnet.

1. Analyzing Telnet packets

Why does everyone use SSH instead of Telnet? - SSH and Telnet are two protocols that have the same objective, that is, access a server to perform operations on that remote system. But the main difference between the two is encryption.

A simple definition for encryption is the ability to make a message unreadable and only someone with the rollback key could read the original content.

If an unencrypted message travels over a network, anyone who intercepts it can read its contents. Using a network packet analyzer, which is also known as a Sniffer, we can demonstrate this. As an example, we can use Wireshark, in *Figure 1*, and see network packets traveling both on an SSH connection and on a Telnet connection.

📕 The Wireshark Network Analyzer – 🗆 🗙								
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help								
A 🔳 🖉 🐵 📴 🔤 🔍 🗢 🗢 🕾 T 🕹 🧮 🗮 🍳 Q, Q, X								
Apply a display filter <ctrl-></ctrl->								
🖉 About Wireshark X								
Wireshark Authors Folders Plugins Keyboard Shortcuts Acknowledgments License								
WIRESHARK								
Network Protocol Analyzer								
Version 3.2.4 (v3.2.4-0-g893b5a5e1e3e)								
Copyright 1998-2020 Gerald Combs <gerald@wireshark.org> and contributors. License GPLv2+: GNU GPL version 2 or later <https: <br="">www.gnu.org/licenses/gpl-2.0.html> This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.</https:></gerald@wireshark.org>								
Compiled (32-bit) with Qt 5.12.8, with WinPcap SDK (WpdPack) 4.1.2, with GLib 2.52.3, with zlib 1.2.11, with SMI 0.4.8, with c-ares 1.15.0, with Lua 5.2.4, with GnuTLS 3.6.3 and PKCS #11 support, with Gcrypt 1.8.3, with MIT Kerberos, with MaxMind DB resolver, with nghttp2 1.39, 2, with brotil, with LZ4, with Zstandard, with Snappy, with libxml2 2.9.9, with QtMultimedia, with automatic updates using WinSparkle 0.5.7, with AirPcap, with SpeexDSP (using bundled resampler), with SBC, with SpanDSP, with bcg729.								
Running on 64-bit Windows 10 (2009), build 19042, with Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz (with SSE4.2), with 20360 MB of physical memory, with locale Portuguese_Brazil.1252, with light display mode, without HiDPI, with Npcap version 0.9991, based on libpcap version 1.9.1, with GnuTLS 3.6.3, with Grypt 1.8.3, with brotil 1.0.2, with AirPcap 4.1.0 build 1622, binary plugins supported (19 loaded). Built using Microsoft Visual Studio 2019 (VC++ 14.25, build 28614).								
Wireshark is Open Source Software released under the GNU General Public License.								
Check the man page and https://www.wireshark.org for more information.								
Copy To Clipboard								
OK								
Ready to load or capture No Packets Profile: Default								

Figure 1. Wireshark app



See other alternatives to Wireshark in section Network sniffers options.

We need to check which is the server IP and make Wireshark filter IP packets addressed to this using the syntax ip.dst == 191.235.98.138. Now only network packets from that Telnet connection will be displayed.

For the test we can inform the user on the telnet connection and press Enter. Then we clear the

Wireshark history and we can see from here that for each key typed a network packet is sent exposing what you type.

File Edit View go Capture Analyze Subsicion Belghorny, Wireless Tools, Heip Image: Source Control Contro Control Control Contro Contro Contro Contro ControCon	🧕 Capturing from Wi-Fi					-	- 0	×	$oldsymbol{\mathbb{Z}}$ Telnet machine.devnator.com $ imes$ + $ imes$
					p				
No. Time Source Destination Protocol Length: Jinfo 1 b2 2,492380 192,166.43,221 191,235.96,138 TELNET 55 Telnet Data TELNET 55 Telnet Data <t< th=""><th>1 🔲 🧕 🧶 🔳 🖾</th><th>X D 9 0 0 0 0</th><th>F 🕹 其 📃 Q. Q.</th><th>Q. 👖</th><th></th><th></th><th></th><th></th><th>Debian GNU/Linux 10</th></t<>	1 🔲 🧕 🧶 🔳 🖾	X D 9 0 0 0 0	F 🕹 其 📃 Q. Q.	Q. 👖					Debian GNU/Linux 10
10 2.692300 192.166.43.221 191.235.96.138 TELNET 15 14.429507 192.166.43.221 191.235.96.138 TELNET 15 14.429507 192.166.43.221 191.235.96.138 TELNET 15 Telnet Data 17 15 15.42000 192.166.43.221 191.235.96.138 TELNET 55 Telnet Data 17 216.172077 192.166.43.221 191.235.96.138 TELNET 55 Telnet Data 17 15 15.4000 192.166.43.221 191.235.96.138 TELNET 55 Telnet Data 17 15 15.4000 192.166.43.221 193.235.96.138 TELNET 55 Telnet Data 17 16.172077 192.166.43.221 193.235.96.138 TELNET 55 Telnet Data 17 16.172077 192.166.43.221 193.235.96.138 TELNET 55 Telnet Data 17 16.172077 192.166.43.221 193.235.96.138 TELNET 55 Telnet Data 17 16.432 193.235.96.138 TELNET 55 Telnet Data 1 17 16.17207 192.166.43.221 193.235.96.138 TELNET 55 Telnet Data 1 17 16.17207 192.166.43.221 193.235.96.138 TELNET 55 Telnet Data 1	ip.dst 191.235.98.138	1					\times	* +	downston login, myuco
Frame 175: 55 bytes on wire (44e bits), 55 bytes captured (44e bits) on interface \Device\UPF_[643859AC_MSF-48] Frame 175: 55 bytes on wire (44e bits), 55 bytes captured (44e bits) on interface \Device\UPF_[643859AC_MSF-48] Eterment II, Src: IntelCor_9xibit2c (64:32:a8):98:b2c; 0, bat: Notorola_b6:17:01 (66:c4:4d:b6:17:01) Intermet Protocol Version 4, Src: 132.168.43.221, Dat: 191.235.98.138 Telnet Data: :					-				devilator togrii. myuse
Frame 175: 55 bytes on wire (44e bits), 55 bytes captured (44e bits) on interface \Device\UPF_[643859AC_MSF-48] Frame 175: 55 bytes on wire (44e bits), 55 bytes captured (44e bits) on interface \Device\UPF_[643859AC_MSF-48] Eterment II, Src: IntelCor_9xibit2c (64:32:a8):98:b2c; 0, bat: Notorola_b6:17:01 (66:c4:4d:b6:17:01) Intermet Protocol Version 4, Src: 132.168.43.221, Dat: 191.235.98.138 Telnet Data: :								m	Dassword.
166 15.906912 192.166.43.221 191.235.90.138 TELNET 55 Telnet Data 172 16.372677 192.168.43.221 191.235.90.138 TELNET 55 Telnet Data 175 16.809698 192.168.43.221 191.235.90.138 TELNET 55 Telnet Data 9 9 9 9 9 9 9 9 9 175 16.80960 2 Frame 175: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface \Device\UPF_(643359aC) hase-48 1								~	14330014.
177 16.37497 197.168.43.221 191.235.96.138 TELMET 55 Telmet Data 175 16.809538 192.168.43.221 191.235.96.138 TELMET 55 Telmet Data 2 5 Telmet 175: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface \Device\NPF.(643859AC:A5F-48) 2 Ethernet II, Src: IntelCor_9e:b6:2c (64:32:a8:9e:b6:2c), Dst: Notorola_b6:17:01 (68:c4:46:b6:17:01) 2 Internet Protocol Version 4, Src: 192.168.4.3:21, Dst: 191.235.96.188 7 Transmission Control Protocol, Src Port: 51950, Dst Port: 23, Seq: 6, Ack: 1, Len: 1 Data: s Seq: 6, Ack: 1, Len: 1								V	
175 16.803630 192.168.43.221 103.225.06.138 TELNET 55 Telnet Data 0 0 0 0 0 0 0 2 Frame 175: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on Interface \Device\UPF_(643859AC-ASF-48) 0 0 5 Ethernet II, Src: IntelCor_9e:b6:2c (64:32:a8:9a:b6:2c), Dst: Motorola_b6:17:01 (66:c4:4d:b6:17:01) 1 1 1 Internet Protocol Version 4, Src: 192.168.43.221, Dst: 191.235.96.138 1 1 1 7elnet Data: s 0 68:c4:4d:b6:17:01.65:c4:32:a8:9a:b6:2c 08:00:45:00 http://d2 , E									
Frame 175: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface \Device\NPF_(643859AC-A5F-48) S Ethernet II, Src: IntelCor_9eib6;2c (64:32:e8:99:b6:2c), Dst: Notorola_b6:17:01 (66:c4:4d:b6:17:01) Internet Protocol Version 4, Src: 192.166.43.221, Dst: 191.235.98.138 Transmission Control Protocol, Src Port: 51950, Dst Port: 23, Seq: 6, Ack: 1, Len: 1 Telnet Data: s				TELNET					
Internet Protocol Version 4, Src: 192.168.43.221, Dst: 191.235.98.138 Transmission Control Protocol, Src Port: 51950, Dst Port: 23, Seq: 6, Ack: 1, Len: 1 Telnet Data: s 0000 68 c4 4d b6 17 01 64 32 a8 9a b6 2c 08 00 45 00 h-Md2,E-							(643859AC-	5	
	<pre>> Internet Protocol Version 4, Src: 192.168.43.221, Dst: 191.235.98.138 > Transmission Control Protocol, Src Port: 51950, Dst Port: 23, Seq: 6, Ack: 1, Len: 1 Y Telnet</pre>								
	20000 c0 at at bc 17	a		45 5					1

Figure 2. Telnet packets with leaked keystrokes

As shown in *Figure 2*, my password being "mypass" and I can see each letter being sent by the network packets: "m", "y", "p", "a", "s", "s".

When proceeding with the connection and sending commands we see the network packets traveling with the data open for reading, without using encryption.

2. Analyzing SSH packets

We now connect using the SSH protocol. Unlike Telnet, which after establishing the connection needs to receive the user's name via keyboard, SSH already sends this information together with the address of the remote computer at the time of connection. Then you only enter the password via the keyboard.

Since it is the same remote computer, we will continue to use the IP filter applied in Wireshark but we clear the history before entering the password. You will notice that a network packet is not sent for each keystroke. It will only be sent when you are finished entering the password and pressing Enter. And the package will be sent in encrypted form.

After login, the SSH protocol also sends a network packet for each key typed, just like Telnet does. But these packages are encrypted, they are not readable, as indicated in yellow in *Figure 3*.

🙇 Capturing from Wi-Fi				- 🗆 X	🔊 SergioCabral $ imes$ + $ imes$
		stics Telephony Wireless		elp	🗲 SergioCabral@DEVN
ip.dst == 191.235.98.13	8			*	e e e e e e e e e e e e e e e e e e e
No. Tme 141 19.915302 145 21.149613 147 21.294759 148 21.333288 159 21.407250 152 21.513617 153 21.519015 154 21.59015 154 21.50454 156 21.747976 161 21.747976 161 21.877631 164 21.918227 169 22.005514 177 22.1162145 173 22.150145 179 22.231743 > Frame 54: 138 byth	Source 192.166.43.221 192.16		TCP SSH TCP SSH SSH TCP SSH TCP SSH TCP SSH TCP SSH TCP SSH TCP SSH TCP	Length Info 54 51952 + 22 [ACK] Seq=825 Ack=1 90 Client: Uncrypted packet [len= 90 Client: Encrypted packet [len= 54 51952 + 22 [ACK] Seq=1095 Ack= 90 Client: Encrypted packet [len= 54 51952 + 22 [ACK] Seq=1095 Ack= 90 Client: Encrypted packet [len= 54 51952 + 22 [ACK] Seq=1095 Ack= 90 Client: Encrypted packet [len= 54 51952 + 22 [ACK] Seq=1095 Ack= 90 Client: Encrypted packet [len= 54 51952 + 22 [ACK] Seq=1041 Ack= 90 Client: Encrypted packet [len= 54 51952 + 22 [ACK] Seq=1143 Ack= 90 Client: Encrypted packet [len= 54 51952 + 22 [ACK] Seq=1143 Ack= 90 Client: Encrypted packet [len= 54 51952 + 22 [ACK] Seq=1143 Ack= 90 Client: Encrypted packet [len= 54 51952 + 22 [ACK] Seq=1143 Ack= 90 Client: Encrypted packet [len= 54 51952 + 22 [ACK] Seq=1143 Ack= 90 Client: Encrypted packet [len=] 54 51952 + 22 [ACK] Seq=1143 Ack= 90 Client: Encrypted packet [len=] 54 51952 + 22 [ACK] Seq=1143 Ack= 90 Client: Encrypted packet [len=] 54 51952 + 22 [ACK] Seq=1143 Ack=] 55 on interface \Device\NPF_[643859AC-D45] a b6:17:01 (68:c4:4d:b6:17:01)	Password: Password: Linux devnator 4.19. 146-1 (2020-09-17) x Last login: Sat Oct 101.user.vivozap.com mvuser@devnator:~\$ u
> Internet Protocol	Version 4, Src: 192.	168.43.221, Dst: 191. t: 51952, Dst Port: 2	235.98.138	Ack: 1, Len: 84	146-1 (2020-09-17) x myuser@devnator:~\$

Figure 3. Encrypted SSH packets

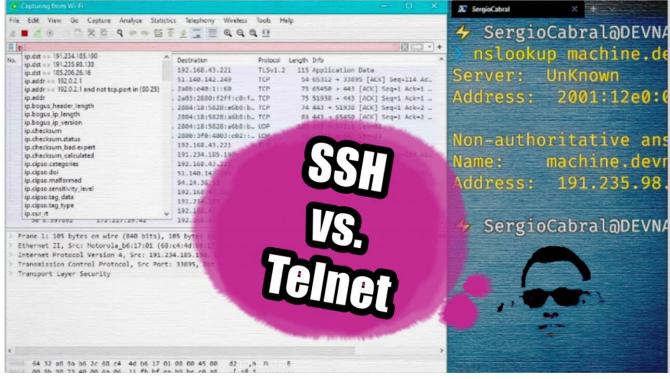
3. Conclusion

That is why SSH is an acronym for Secure Shell. That is, do not use Telnet on insecure networks and always prefer SSH.

4. Network sniffers options

Name	License	Download
Tcpdump	free; open-source	https://www.tcpdump.org/
Cloudshark	trial	https://www.cloudshark.org/
Sysdig	free; open-source	https://sysdig.com/opensource/ inspect/
Ettercap	free; open-source	https://www.ettercap- project.org/downloads.html
SmartSniff	free	https://www.nirsoft.net/utils/ smsniff.html

5. Video demonstration



https://youtu.be/f3FdwUO4v6M

I'll be back.