

**ESCUOLA COLOMBIANA DE INGENIERIA
JULIO GARAVITO**

**IT SECURITY AND PRIVACY
GROUP 1L**

**CRYPTOGRAPHY PRACTICE
LABORATORY 8**

**SUBMITTED BY:
JUAN PABLO FERNANDEZ GONZALES
MARIA VALENTINA TORRES MONSALVE**

1

**SUBMITTED TO:
Eng. DANIEL ESTEBAN VELA LOPEZ**

**BOGOTÁ D.C.
DATE:
10/03/2025**

Introduction

This report presents the static analysis performed on the putty.exe executable file. After several reports of unusual behavior of the system where this file was executed, it was suspected that putty.exe had been modified for malicious purposes. The analysis focused on identifying the file by calculating its hash, evaluating its architecture, inspecting its imports, and analyzing the internal chains and behavior of the binary. This report provides a detailed description of the analysis process, the tools used and the results obtained, offering well-founded conclusions about its nature and behavior.

Static Analysis

An inspection of the file was performed without running it in an active environment. To do this, tools such as:

- **PEStudio:** Allows you to extract and analyze the imports and the structure of the file.
- **FLOSS:** Facilitates the extraction of internal strings that could indicate suspicious behavior.
- **VirusTotal:** Helps to verify if the file has been previously identified as malicious by different antivirus engines.

The static analysis provided key insights into the file's internal characteristics, including its architecture, identification hash, and potential malicious behavior.

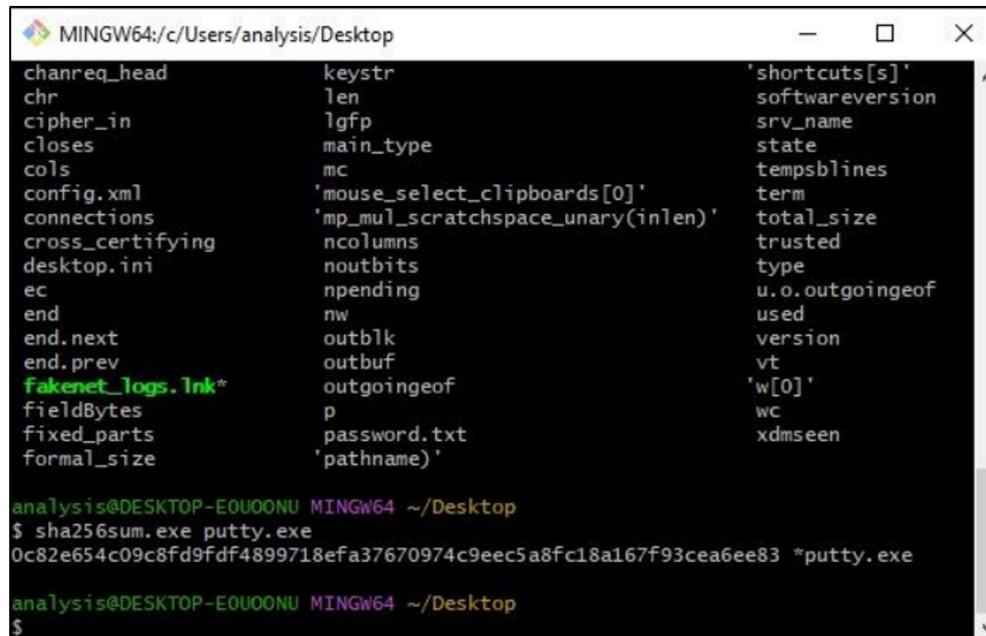
2

Static Analysis: Detailed Answers

- ***What is the SHA256 hash of the sample?***

The purpose of this question is to calculate the SHA256 hash of the file to generate a unique "digital signature." This signature facilitates the unique identification of the file, allows you to verify whether it has been previously scanned and classified as malicious, and makes it possible to compare it with recognized malware databases.

To be able to answer this question from a Linux terminal in the virtual machine we use the ***command "sha256sum.exe putty.exe"*** which is the one that calculates the ***SHA256 hash***



The terminal window shows the following output:

```
MINGW64:/c/Users/analysis/Desktop
chanreq_head      keystr          'shortcuts[s]'
chr               len              softwareversion
cipher_in         lgfp             srv_name
closes            main_type       state
cols              mc               tempsblines
config.xml        'mouse_select_clipboards[0]'
connections       'mp_mul_scratchspace_unary(inlen)'
cross_certifying ncolumns        term
desktop.ini       noutbits        total_size
ec                npending        trusted
end               nw               type
end.next          outblk          u.o.outgoingeof
end.prev          outbuf          used
fakenet_logs.lnk* outgoingeof    version
fieldBytes        p               vt
fixed_parts       password.txt   wc
formal_size       'pathname'     xdmseen

analysis@DESKTOP-E0UOONU MINGW64 ~/Desktop
$ sha256sum.exe putty.exe
0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83 *putty.exe

analysis@DESKTOP-E0UOONU MINGW64 ~/Desktop
$
```

This command generates a SHA256 hash, which is a unique 64-character hexadecimal representation of the file. This "digital signature" can be used to compare the file in different environments and check if it has been modified.

3

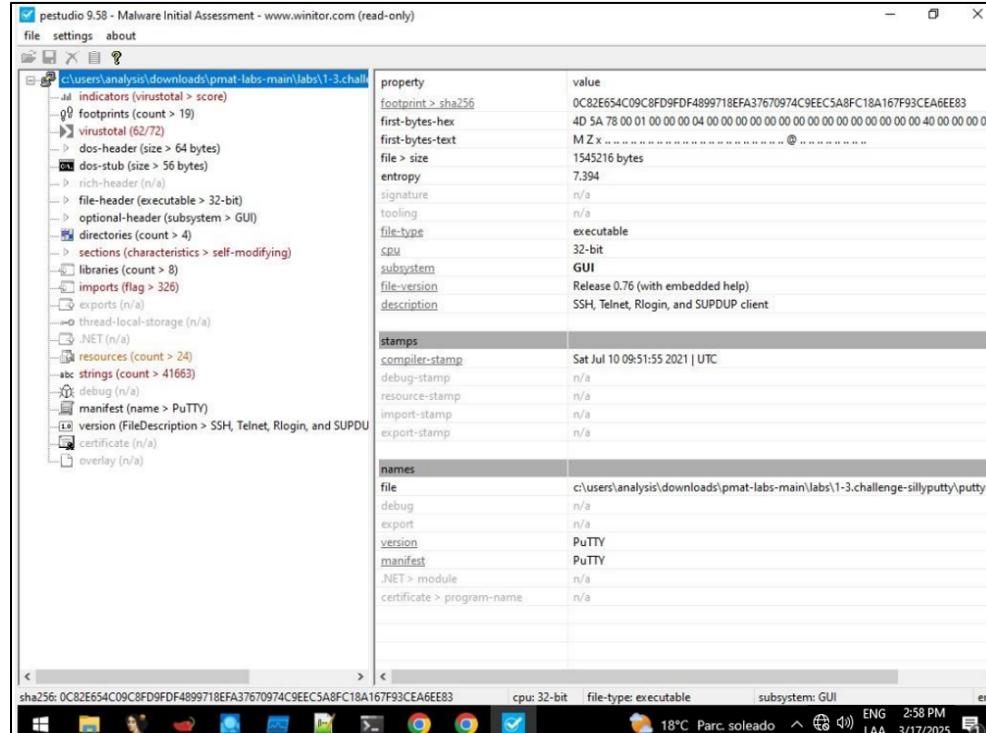
Result

"0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83"

- *What architecture is this binary?*

The architecture of an executable file defines what types of systems it can run on (for example, 32-bit or 64-bit). Identifying your architecture is essential to determine your potential constraints and assess your compatibility with target systems.

PEStudio was used to examine the binary and gain insight into its architecture. This tool allows you to analyze the file headers and extract technical details without having to run it.



4

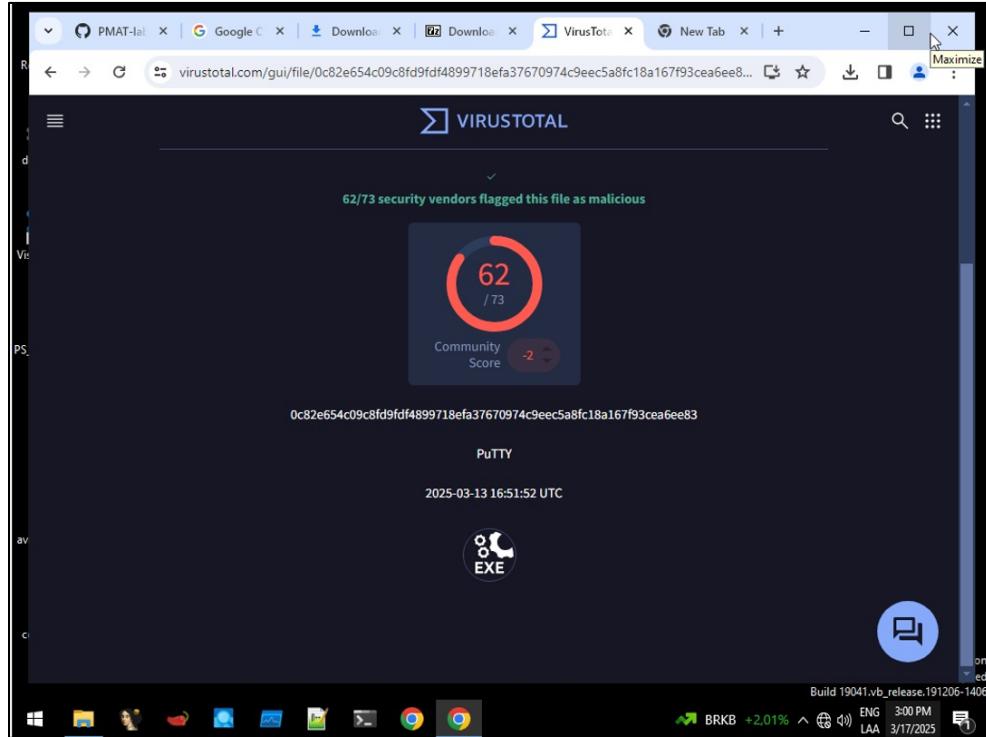
The analysis carried out with PEStudio revealed that the putty.exe file has a 32-bit architecture. This means that it is compatible with 32-bit Windows systems and can also run on 64-bit systems that support 32-bit programs. In addition, the following characteristics were observed:

- It has a graphical user interface (GUI).
 - Version: Corresponds to version 0.76 of PuTTY, a legitimate tool used for SSH, Telnet, Rlogin and SUPDUP connections.
 - Imports: More than 326 imported functions were detected, suggesting considerable interaction with the operating system.
 - *Are there any results from submitting the SHA256 hash to VirusTotal?*

- Are there any results from submitting the SHA256 hash to VirusTotal?

The purpose is to check if the file has been previously classified as malicious on various malware analysis platforms, such as ***VirusTotal***. This allows you to assess the threat level of the file and get an overview of its behavior in different environments.

The SHA256 hash was entered into *VirusTotal*, a service that analyzes files using dozens of antivirus engines.



62/73 security vendors flagged this file as malicious

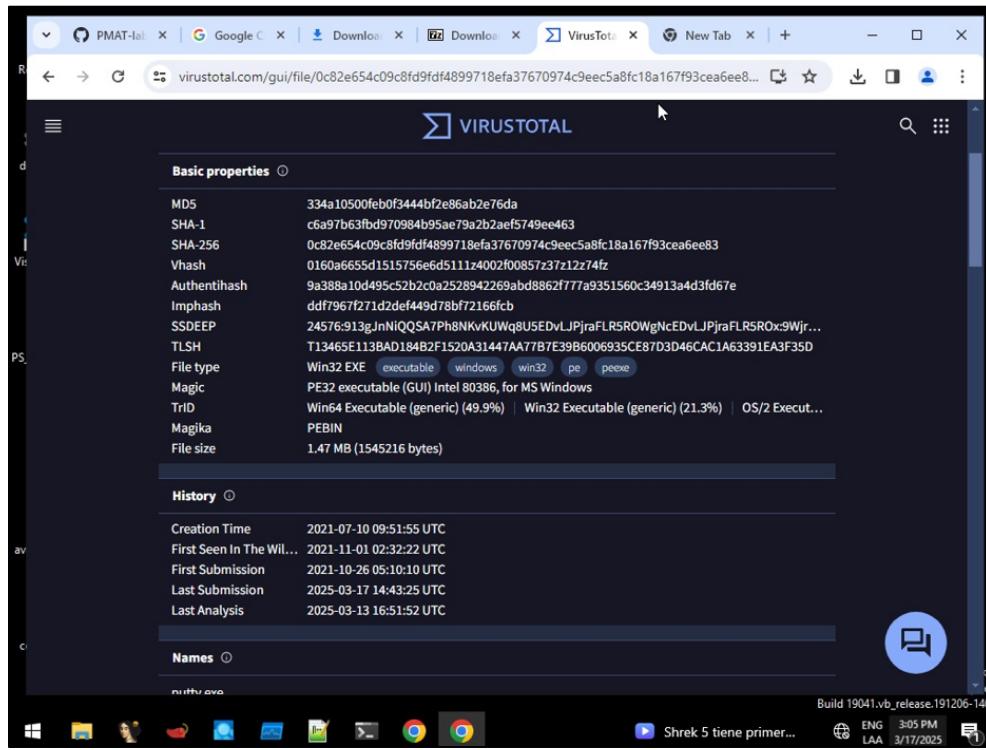
Community Score -2

PuTTY

2025-03-13 16:51:52 UTC

EXE

Build 19041.vb_release.191206-1406



Basic properties

MD5	334a10500feb0f3444bf2e86ab2e76da
SHA-1	c6a97b63fb970984b95ae79a2baef5749ee463
SHA-256	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83
Vhash	0160a6655d15157566ed5111z4002f00857z37z12274fz
Authentihash	9a388a10d495c52b2c0a2528942269abd862f777a9351560c34913a4d3fd67e
ImpHash	dd7967f271d2def449d78bf72166fc
SSDeep	24576913gJnNiQOSA7PhBNKvKUWq8U5EDvLJPjraFLR5ROWgNcEDvLJPjraFLRSROx:9Wjr...
TLSH	T13465E113BAD184B2F1520A31447AA77B7E39B6006935CE87D3D46CAC1A63391EA3F35D
File type	Win32 EXE executable windows win32 pe pexe
Magic	PE32 executable (GUI) Intel 80386, for MS Windows
TrID	Win64 Executable (generic) (49.9%) Win32 Executable (generic) (21.3%) OS/2 Execut...
Magika	PEBIN
File size	1.47 MB (1545216 bytes)

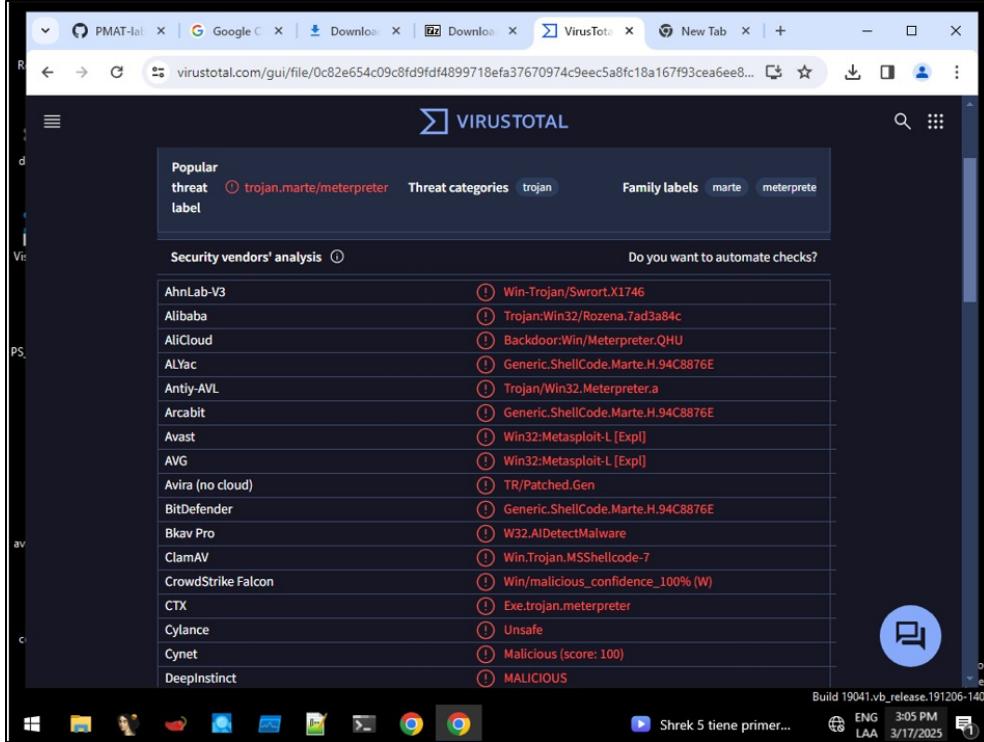
History

Creation Time	2021-07-10 09:51:55 UTC
First Seen In The Wild	2021-11-01 02:32:22 UTC
First Submission	2021-10-26 05:10:10 UTC
Last Submission	2025-03-17 14:43:25 UTC
Last Analysis	2025-03-13 16:51:52 UTC

Names

putty.exe

Build 19041.vb_release.191206-1406



The screenshot shows the VirusTotal analysis interface. At the top, it displays a threat label: "Popular threat: trojan.marte/meterpreter". Below this, under "Security vendors' analysis", there is a table listing 73 antivirus engines and their detection results. The table shows that 62 engines detected the file as malicious, while 11 detected it as benign and 10 detected it as unsafe or malicious with lower confidence. The engines listed include AhnLab-V3, Alibaba, AliCloud, ALYac, Antiy-AVL, Arcabit, Avast, AVG, Avira (no cloud), BitDefender, Bkav Pro, ClamAV, CrowdStrike Falcon, CXT, Cylance, Cynet, and DeepInstinct. The bottom right corner of the interface shows the build number: "Build 19041.vb_release.191206-1406".

Out of a total of 73 antivirus engines, 62 detected the file as malicious, identifying it as:

- trojan.mars/meterpreter
- Trojan/Rozena
- Liftgate/Meterpreter

The results obtained from VirusTotal indicate that the file is very likely to possess backdoor and Trojan functionalities. Such features suggest that the file could allow remote control of the infected system and the execution of malicious commands using secure SSH connections. The high number of detections confirms that most antivirus engines consider the file to be a critical threat.

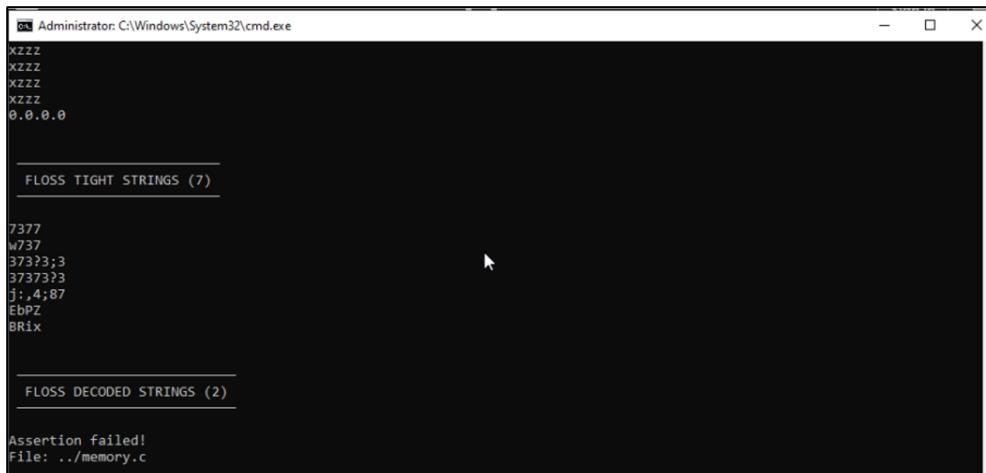
- *Describe the results of pulling the strings from this binary. Record and describe any strings that are potentially interesting. Can any interesting information be extracted from the strings?*

Extracting text strings from a binary is very useful for identifying readable fragments that can give clues about its functionalities or intentions. In malware analysis, these strings can reveal function names, URLs, commands, and other encoded data that the malware employs in its operation.

The floss putty.exe command was run to extract the strings from the putty.exe file.

This command detects the text fragments embedded in the binary, which may have been included in the original code or encoded directly into the executable.

```
:\Users\analysis\Desktop\putty>floss putty.exe
INFO: floss: extracting static strings
WARNING: viv_utils: cfg: incomplete control flow graph
INFO: floss: decoding function features: 100% [ 2786/2786 [00:03<00:00, 809.73 functions/s, skipped 675 library functions ]
INFO: floss: stackstrings: extracting stackstrings from 1993 functions
INFO: floss: results: Proxy error:
INFO: floss: results: 0WB4
INFO: floss: results: 1WB4
INFO: floss: results: xzzz
INFO: floss: results: 0.0.0.0
extracting stackstrings: 100% [ 1993/1993 [00:09<00:00, 215.82 functions/s]
INFO: floss: tightstrings: extracting tightstrings from 105 functions...
INFO: floss: results: 7377
INFO: floss: results: w737
INFO: floss: results: 373?3;3
INFO: floss: results: 37373?3
INFO: floss: results: j;4;87
INFO: floss: results: EbPZ
INFO: floss: results: BRlx
extracting tightstrings from function 0x49e120: 100% [ 105/105 [00:03<00:00, 28.99 functions/s]
INFO: floss: string_decoder: decoding strings
simulating function 0x462cf4 (call 11/11): 27% [ 15/56 [00:02<00:08, 4.72 functions/s]
```



```
Administrator: C:\Windows\System32\cmd.exe
XZZZ
XZZZ
XZZZ
XZZZ
0.0.0.0

FLOSS TIGHT STRINGS (7)

7377
w737
373?3;3
37373?3
j;4;87
EbPZ
BRlx

FLOSS DECODED STRINGS (2)

Assertion failed!
File: .../memory.c
```

The most relevant strings found in the binary and their implications are detailed below:

1. Strings related to remote connections and SSH

- "ppl->vt == &ssh2_transport_vtable", "ppl->vt == &ssh1_connection_vtable", "ssh->version == major_version": These strings indicate that the malware is configured to operate with both SSH1 and SSH2, suggesting the ability to establish secure connections using these protocols. SSH is frequently used in malware to create encrypted tunnels and maintain communication with command-and-control (C2) servers. The reference to the SSH version reinforces the hypothesis that the malware could be establishing secure remote

connections to transfer data or execute commands on the compromised system.

- `"ssh_key_alg(s->hkey) == s->cross_certifying":`
This string suggests that the malware uses cryptographic keys to authenticate SSH connections, which is a technique used by attackers to encrypt communications and evade detection by network monitoring tools.

2. Strings related to system persistence and manipulation

- `"RegCreateKeyA", "RegSetValueExA", "RegDeleteKeyA":`
These are standard calls to the Windows Registry that allow you to create, modify, or delete entries. In the context of malware, they indicate attempts to ensure persistence in the system, modifying the Registry so that the malware runs on system startup, or even to disable system features, such as antivirus or firewall.

3. Strings related to file and memory handling

- `"WriteFile", "ReadFile", "GlobalMemoryStatus":`
These functions suggest that malware can interact directly with system files, reading and writing to them, which is common in malware designed to exfiltrate information. In addition, access to system memory could be used to capture sensitive data, such as passwords or session keys, stored temporarily.

4. Strings related to source code and configuration

- `.. /ssh2transport.c", .. /sshutils.c", .. /sshp rng.c":`
These references to source files related to SSH components suggest that the malware may have been developed by modifying legitimate SSH tools, adapting them for malicious purposes. This reinforces the idea that malware is designed to take advantage of SSH capabilities and perform malicious activities while presenting itself as a trustworthy tool.

The study of the extracted strings indicates that the malware putty.exe possesses advanced functionalities to manipulate secure remote connections via SSH, interact with the file system and memory, and remain persistent on the system by modifying the Windows Registry. These strings suggest that the malware operates covertly and with encrypted communications, making it difficult to detect using conventional security tools.

- *Describe the results of inspecting the IAT for this binary. Are there any imports*

The Import Address Table (IAT) of an executable file lists all the external functions that the program needs to import from the system libraries. Analyzing these imports allows you to identify potential suspicious binary activity.

To inspect the imports from the putty.exe file, PEStudio was used. The tool identified several critical functions that suggest that the file has a high level of interaction with the operating system and could perform malicious activities.

pestudio 9.58 - Malware Initial Assessment - www.winitor.com (read-only)

	imports (326)	flag (52)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (16)
is1\downloads\pmat-labs-main\labs\1-3.challenge	GetDesktopWindow	x	0x006C0065	325 (0x0145)	windowing	
virustotal > score	GetForegroundWindow	x	0x002E002E	342 (0x0156)	windowing	
count > 19)	GetQueueStatus	x	0x00740075	429 (0x01AD)	windowing	
52/72)	GetWindowTextA	x	0x00730073	492 (0x01EC)	windowing	
(size > 64 bytes)	GetOverlappedResult	x	0x002F002E	660 (0x0294)	synchronization	
ize > 56 bytes)	AllocateAndInitializeSid	x	0x0073002F	32 (0x0020)	security	
r (n/a)	CopySid	x	0x006B0073	133 (0x0065)	security	
(executable > 32-bit)	EqualSid	x	0x00720061	282 (0x011A)	security	
aader (Subsystem > GUI)	GetLengthSid	x	0x00660063	331 (0x014B)	security	
(count > 4)	SetSecurityDescriptorDacl	x	0x0063002E	744 (0x02E8)	security	
haracteristics > self-modifying)	SetSecurityDescriptorOwner	x	0x002E0000	746 (0x02EA)	security	
unt > 8)	RegCreateKeyA	x	0x00690077	610 (0x0262)	registry	
g > 320)	RegCreateKeyExA	x	0x0064006E	611 (0x0263)	registry	
a)	RegDeleteKeyA	x	0x0077006F	616 (0x0268)	registry	
il-storage (n/a)	RegDeleteValueA	x	0x002F0073	626 (0x0272)	registry	
count > 24)	RegEnumKeyA	x	0x00690077	632 (0x0278)	registry	
nt > 41663)	RegSetValueExA	x	0x00650072	680 (0x02A8)	registry	
)	GetCurrentProcessId	x	0x00610063	534 (0x0216)	reconnaissance	
iame > PuTTY)	GetEnvironmentVariableA	x	0x00730073	564 (0x0234)	reconnaissance	
eDescription > SSH, Telnet, Rlogin, and SUPDUP cli	GlobalMemoryStatus	x	0x002E0063	821 (0x0335)	memory	
n/a)	GetKeyboardState	x	0x006B0073	363 (0x016B)	input-output	
a)	SetKeyboardState	x	0x006E006F	829 (0x033D)	input-output	
DeleteFileA	x	0x002E0000	272 (0x0110)	file		
FindFirstFileA	x	0x002E0065	375 (0x0177)	file		
FindFirstFileExW	x	0x00000063	377 (0x0179)	file		
FindNextFileA	x	0x002E002E	392 (0x0188)	file		
FindNextFileW	x	0x0070002F	394 (0x018A)	file		
MapViewOfFile	x	0x006C007A	983 (0x03D7)	file		
UnmapViewOfFile	x	0x002F002E	1448 (0x05A8)	file		
WriteFile	x	0x002E0000	1546 (0x060A)	file		
ShellExecuteA	x	0x002E002E	434 (0x01B2)	execution		

sha256: 0C82E654C09C8FD9FDF4899718EFA37670974C9EEC5A8FC18A167F93CEA6EE83 cpu: 32-bit file-type: executable subsystem: GUI

9

pestudio 9.58 - Malware Initial Assessment - www.winitor.com (read-only)

file settings about

Imports (326)

imports (326)	flag (52)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (16)
ShellExecuteA	x	0x00124118	0x002E002E	434 (0x01B2)	execution
CreateProcessA	x	0x00124402	0x00730068	223 (0x00DF)	execution
GetCurrentProcess	x	0x001245C4	0x0064006C	533 (0x0215)	execution
GetCurrentThread	x	0x001245EE	0x00640072	537 (0x0219)	execution
GetCurrentThreadId	x	0x00124602	0x0063002E	538 (0x021A)	execution
GetEnvironmentStringsW	x	0x0012462A	0x002F0002	563 (0x0233)	execution
GetThreadTimes	x	0x001247EC	0x0077002F	769 (0x0301)	execution
OpenProcess	x	0x00124A58	0x002E0000	1030 (0x0406)	execution
SetEnvironmentVariableW	x	0x00124B3A	0x002F002E	1292 (0x050C)	execution
TerminateProcess	x	0x00124BD0	0x00730073	1412 (0x0584)	execution
RaiseException	x	0x00124A96	0x00720068	1115 (0x0458)	exception
GetModuleHandleExW	x	0x001246F6	0x00630073	627 (0x0273)	dynamic-library
OutputDebugStringW	x	0x00124A66	0x002F002E	1042 (0x0412)	diagnostic
CloseClipboard	x	0x0012395E	0x002F002E	77 (0x004D)	data-exchange
EmptyClipboard	x	0x00123AAA	0x002E0002	234 (0x00EA)	data-exchange
GetClipboardData	x	0x00123B44	0x0077006F	310 (0x0136)	data-exchange
GetClipboardOwner	x	0x00123B58	0x002F0073	313 (0x0139)	data-exchange
OpenClipboard	x	0x00123E2A	0x00740073	676 (0x02A4)	data-exchange
RegisterClipboardFormatA	x	0x00123EA0	0x00670067	741 (0x02E5)	data-exchange
SetClipboardData	x	0x00123F6A	0x006C0064	806 (0x0326)	data-exchange
SystemParametersInfoA	x	0x00124060	0x006E0069	918 (0x0396)	-
SetCurrentDirectoryA	x	0x00124B12	0x0063002E	1280 (0x0500)	-
CreateWindowExA	-	0x001239C4	0x00680073	115 (0x0073)	windowing
CreateWindowExW	-	0x001239D6	0x002E006B	116 (0x0074)	windowing
DefWindowProcA	-	0x001239F6	0x002E002E	168 (0x0048)	windowing
DefWindowProcW	-	0x00123A08	0x0077002F	169 (0x0049)	windowing
DestroyWindow	-	0x00123A46	0x0077002F	183 (0x00B7)	windowing
DispatchMessageA	-	0x00123A68	0x00730068	190 (0x00BE)	windowing
DispatchMessageW	-	0x00123A7C	0x0063006F	191 (0x00BF)	windowing
EnableWindow	-	0x00123ACE	0x00680073	241 (0x00F1)	windowing
FindWindowA	-	0x00123AF6	0x0063002E	275 (0x0113)	windowing
FindWindowExA	-	0x00123B12	0x0063002E	285 (0x0123)	windowing

sha256: 0C82E654C09C8FD9FDF4899718EFA2767097AC9E5CSA8FC18A167F93CEA6EE83

cpu: 32-bit file-type: executable subsystem: GUI ento

Alerta météo ENG 3:30 PM LAA 3/17/2025

Some of the most relevant imports found in the analysis include:

ShellExecuteA and CreateProcessA: These functions allow you to run files or programs within the system. Its presence in the binary suggests that the file can launch other processes or execute commands without user intervention, which is common in Trojans that download and execute additional payloads.

RegCreateKeyA, RegSetValueExA, RegDeleteKeyA: These functions allow you to modify the Windows Registry, ensuring the persistence of malware on the system. They can be used to execute malware at every system startup or to disable security settings.

FindFirstFileA and FindNextFileA: These functions are used to search for files on the system. Its use is common in malware that searches for sensitive files for exfiltration or in ransomware that encrypts user files.

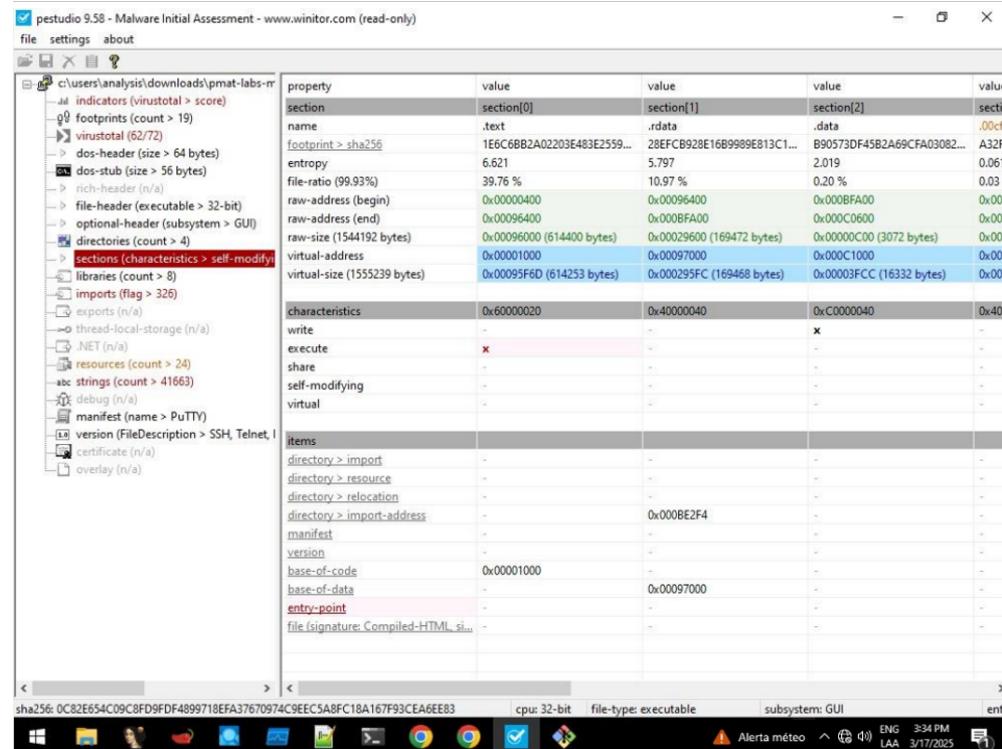
GetClipboardData and EmptyClipboard: These functions allow access to the contents of the system clipboard. Its presence in the binary suggests that the malware could be capturing sensitive information copied by the user, such as passwords or bank details.

GlobalMemoryStatus: This function obtains information about the system's memory usage. It can be used by malware to adjust its behavior and avoid consuming too many resources, reducing the chance of being detected.

Import analysis confirms that the putty.exe file has advanced functionalities to execute processes, manipulate files and registry keys, access the clipboard, and maintain persistence in the system. These characteristics reinforce the suspicion that the binary is malicious.

- *Is it likely that this binary is packed?*

To determine if the putty.exe file was packaged, the .text section was analyzed using PEStudio. The disk and memory sizes of the binary were compared to identify possible indications of compression or encryption.



property	value	value	value	value
section	section[0]	section[1]	section[2]	secti
name	.text	.rdata	.data	.00cf
footprint > sha256	1E6C6BB2A02203E483E2559...	B90573DF45B2A69CFA03082...	B90573DF45B2A69CFA03082...	A32F
entropy	6.621	5.797	2.019	0.061
file-ratio (99.93%)	39.76 %	10.97 %	0.20 %	0.03 %
raw-address (begin)	0x00000400	0x00096400	0x000BFA00	0x00
raw-address (end)	0x00096400	0x000BFA00	0x000C0600	0x00
raw-size (1544192 bytes)	0x00096000 (614400 bytes)	0x00029600 (169472 bytes)	0x00000C00 (3072 bytes)	0x00
virtual-address	0x00001000	0x00097000	0x000C1000	0x00
virtual-size (1555239 bytes)	0x00095F8D (614253 bytes)	0x000295FC (169468 bytes)	0x00003FCC (16332 bytes)	0x00
characteristics	0x60000020	0x40000040	0xC0000040	0x40
write	-	-	x	-
execute	x	-	-	-
share	-	-	-	-
self-modifying	-	-	-	-
virtual	-	-	-	-
items				
directory > import	-	-	-	-
directory > resource	-	-	-	-
directory > relocation	-	-	-	-
directory > import-address	-	0x000BE2F4	-	-
manifest	-	-	-	-
version	-	-	-	-
base-of-code	0x00001000	-	-	-
base-of-data	-	0x00097000	-	-
entry-point	-	-	-	-
file (signature: Compiled-HTML_ si...)	-	-	-	-

11

The analysis revealed that the .text section has a disk size of 614,400 bytes and an in-memory size of 614,253 bytes, with a minimum difference of 147 bytes. In packaged files, the difference between disk size and memory size is often significant, as the code is decompressed or decrypted upon execution.

Since the size difference in this case is negligible, no indication was found that the binary is packed. This suggests that the file does not employ compression or encryption techniques to hide its code, making it easier to analyze.