ESCUELA COLOMBIANA DE INGENIERIA
JULIO GARAVITO

IT SECURITY AND PRIVACY
GROUP 1L

CRYTOGRAPHY PRACTICE
LABORATORY 8

SUBMITTED BY:
JUAN PABLO FERNANDEZ GONZALES
MARIA VALENTINA TORRES MONSALVE

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.

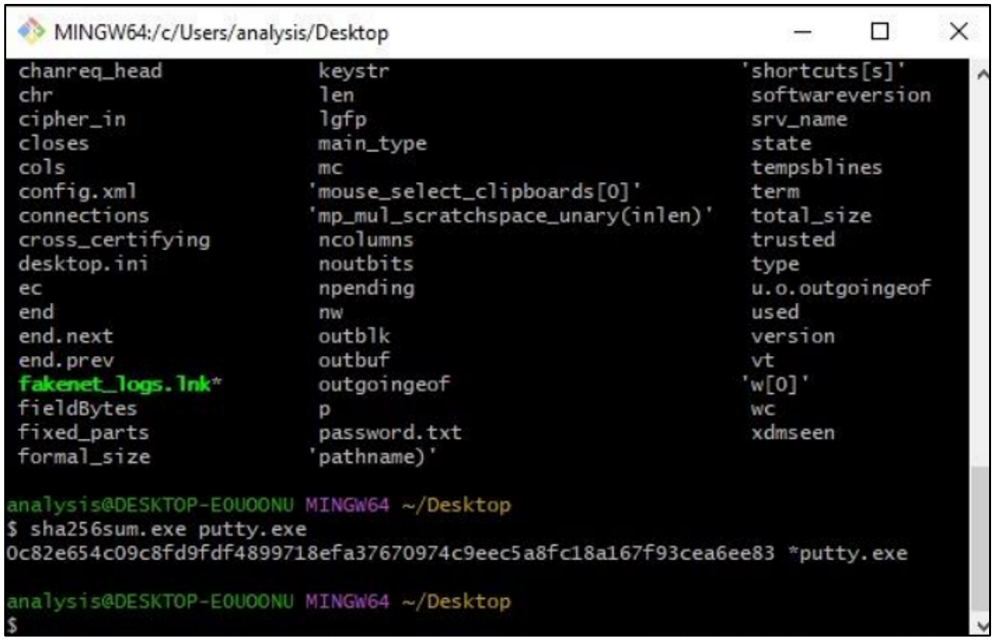
The static analysis provided key insights into the file's internal characteristics, including its architecture, identification hash, and potential malicious behavior.

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***



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.

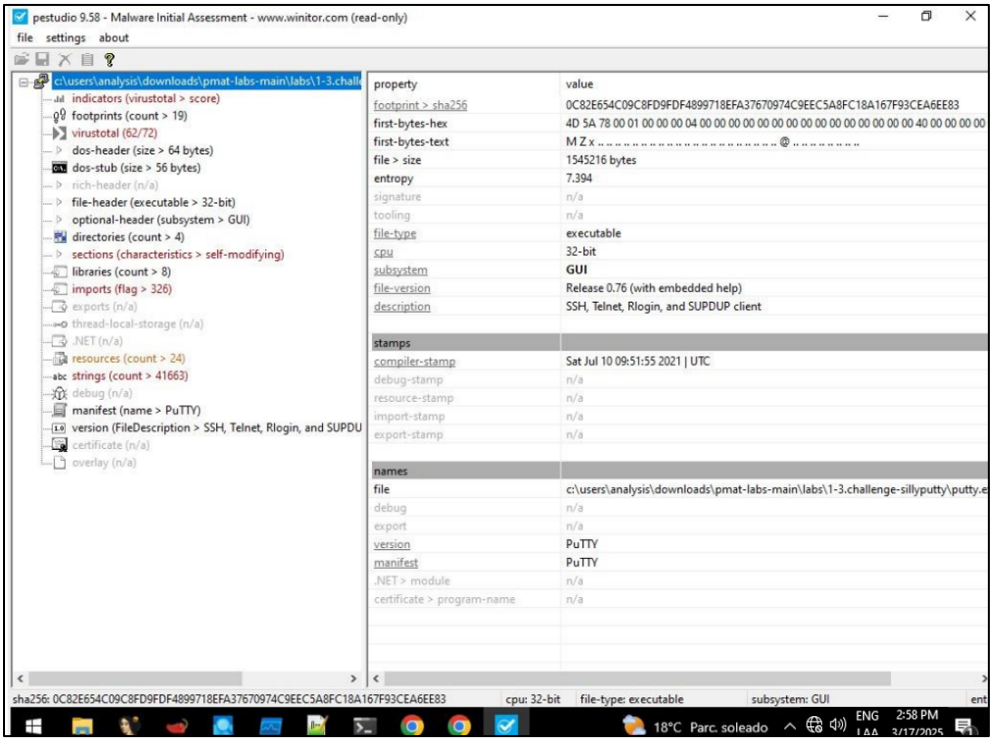
Result

"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.



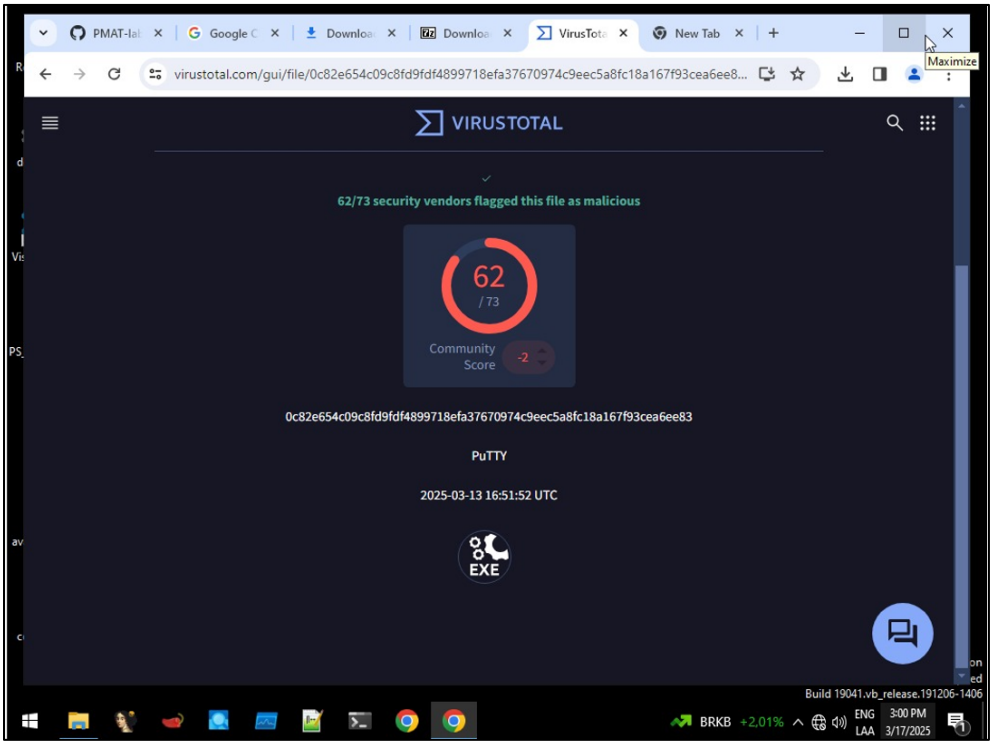
The analysis carried out with PEStudio revealed that the putty.exe file has a 32-bit architecture. This means that it is compatible with 32-bit Windows systems and can also run on 64-bit systems that support 32-bit programs. In addition, the following characteristics were observed:

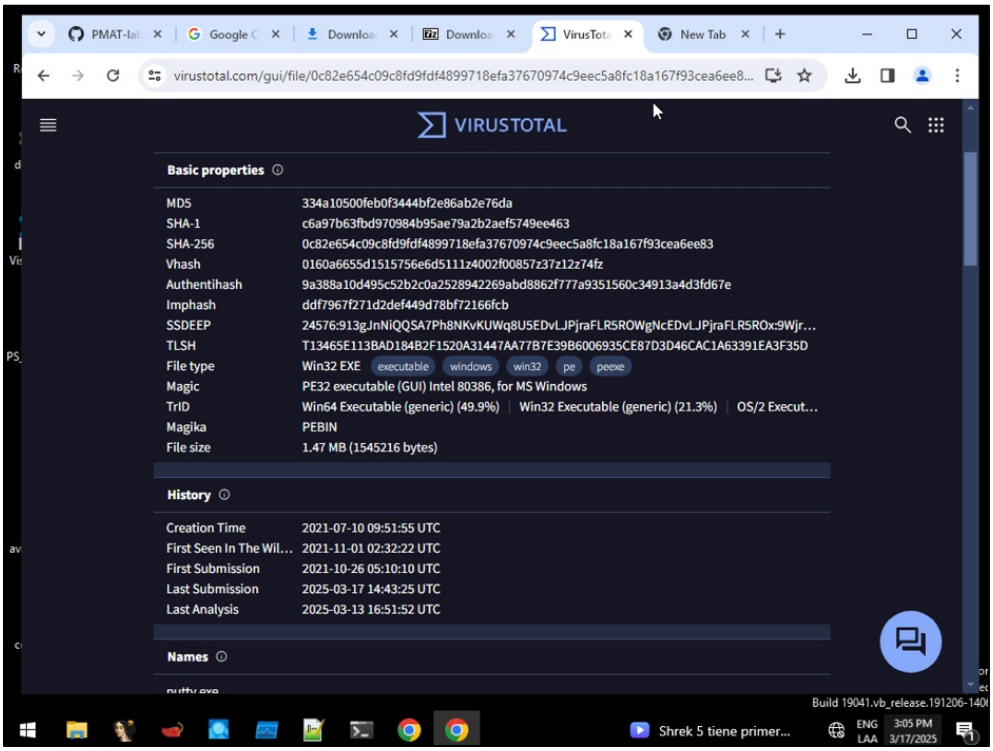
- It has a graphical user interface (GUI).
- Version: Corresponds to version 0.76 of PuTTY, a legitimate tool used for SSH, Telnet, Rlogin and SUPDUP connections.
- Imports: More than 326 imported functions were detected, suggesting considerable interaction with the operating system.

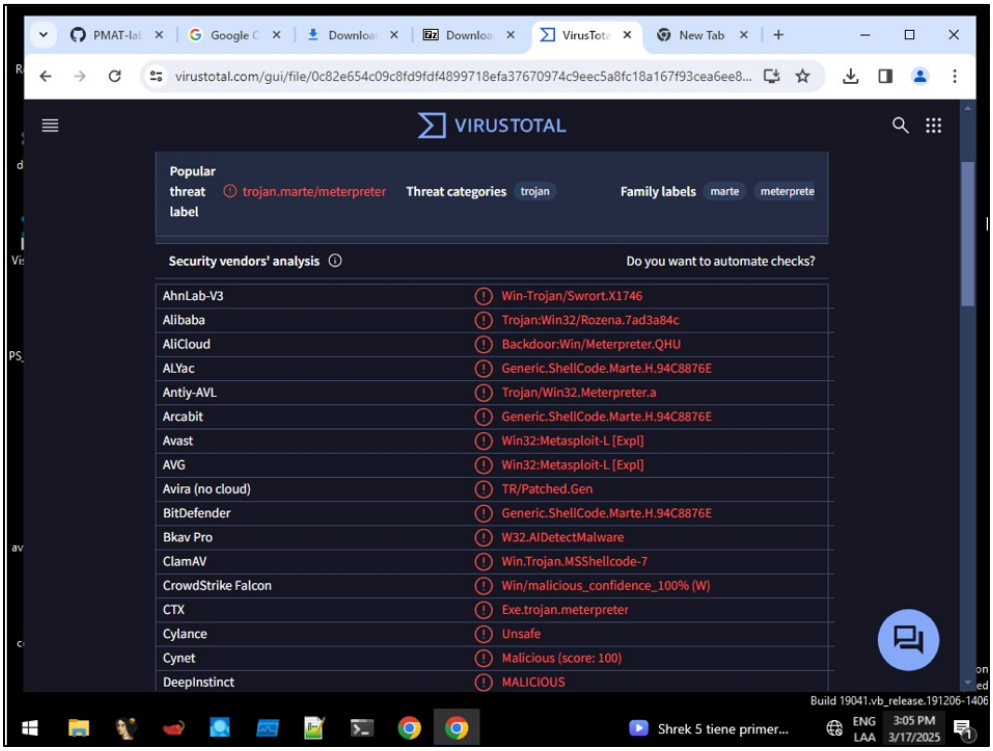
- ***Are there any results from submitting the SHA256 hash to VirusTotal?***

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.


virustotal.com/gui/file/0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee8...
VIRUSTOTAL
62/73 security vendors flagged this file as malicious
62
/ 73
Community Score
-2
0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83
PuTTY
2025-03-13 16:51:52 UTC
EXE


virustotal.com/gui/file/0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee8...
VIRUSTOTAL
Basic properties
MD5 334a10500feb0f3444bf2e86ab2e76da
SHA-1 c6a97b63fbd970984b95ae79a2b2aef5749ee463
SHA-256 0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83
Vhash 0160a6655d1515756e6d5111z4002f00857z37z12z74fz
Authentihash 9a388a10d495c52b2c0a2528942269abd8862f777a9351560c34913a4d3fd67e
Imphash ddf7967f271d2def449d78bf72166fcb
SSDEEP 24576:913gJnNiQQSA7Ph8NKvKUWq8U5EDvLJPjraFLR5ROWgNcEDvLJPjraFLR5ROx:9Wjr...
TLSH T13465E113BAD184B2F1520A31447AA77B7E39B6006935CE87D3D46CAC1A63391EA3F35D
File type Win32 EXE executable windows win32 pe peexe
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 Wil... 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



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

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

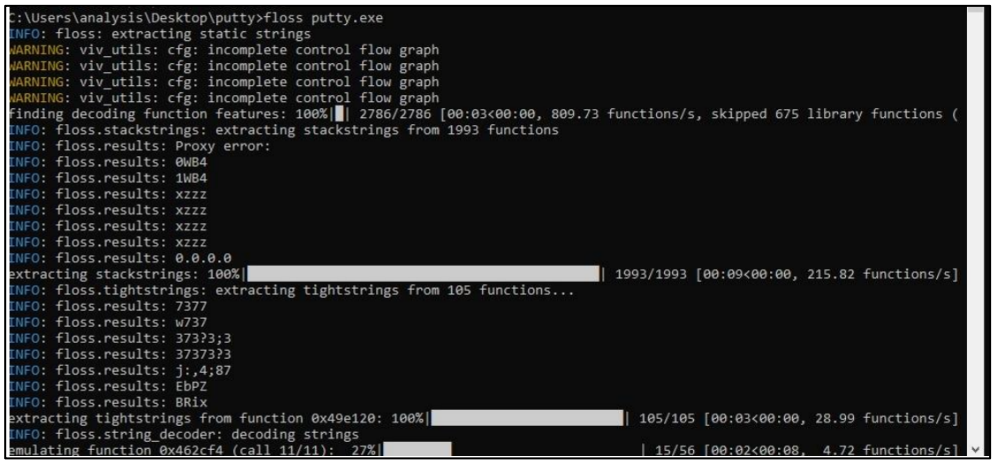
The results obtained from VirusTotal indicate that the file is very likely to possess backdoor and Trojan functionalities. Such features suggest that the file could allow remote control of the infected system and the execution of malicious commands using secure SSH connections. The high number of detections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

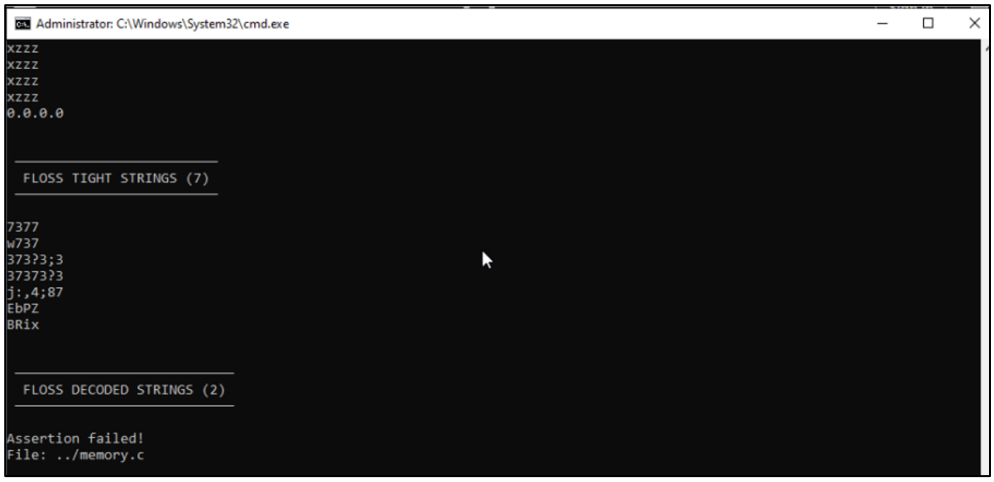
- ***Describe the results of pulling the strings from this binary. Record and describe any strings that are potentially interesting. Can any interesting information be extracted from the strings?***

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.





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.

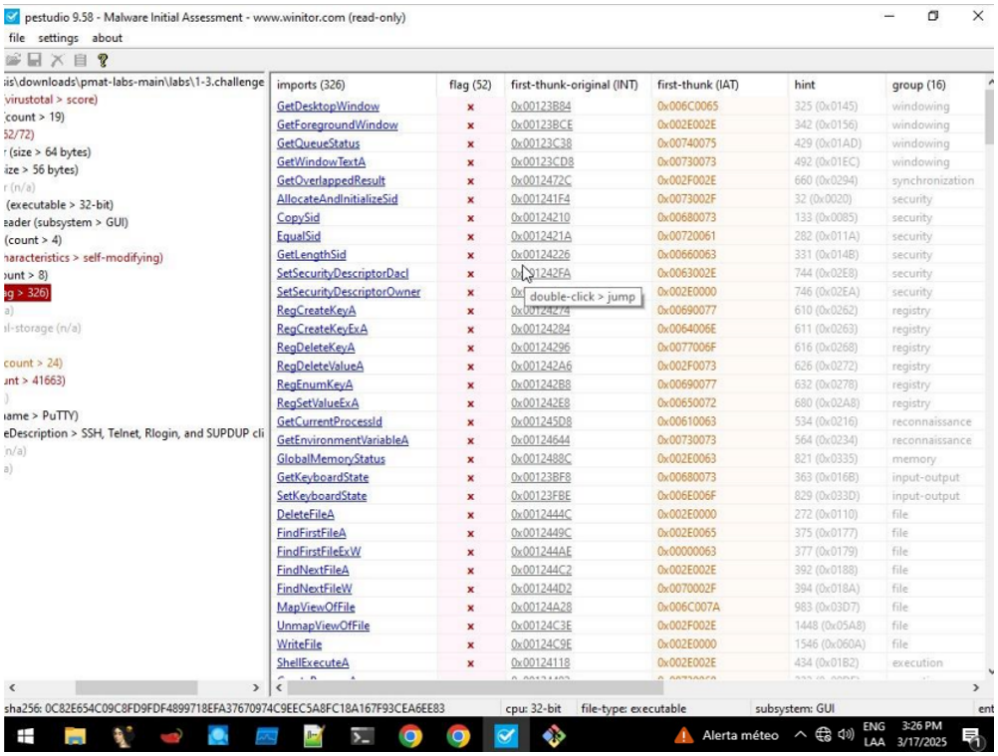
2. **Strings related to system persistence and manipulation**
   - *"RegCreateKeyA", "RegSetValueExA", "RegDeleteKeyA":* These are standard calls to the Windows Registry that allow you to create, modify, or delete entries. In the context of malware, they indicate attempts to ensure persistence in the 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", ".. /sshprng.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.

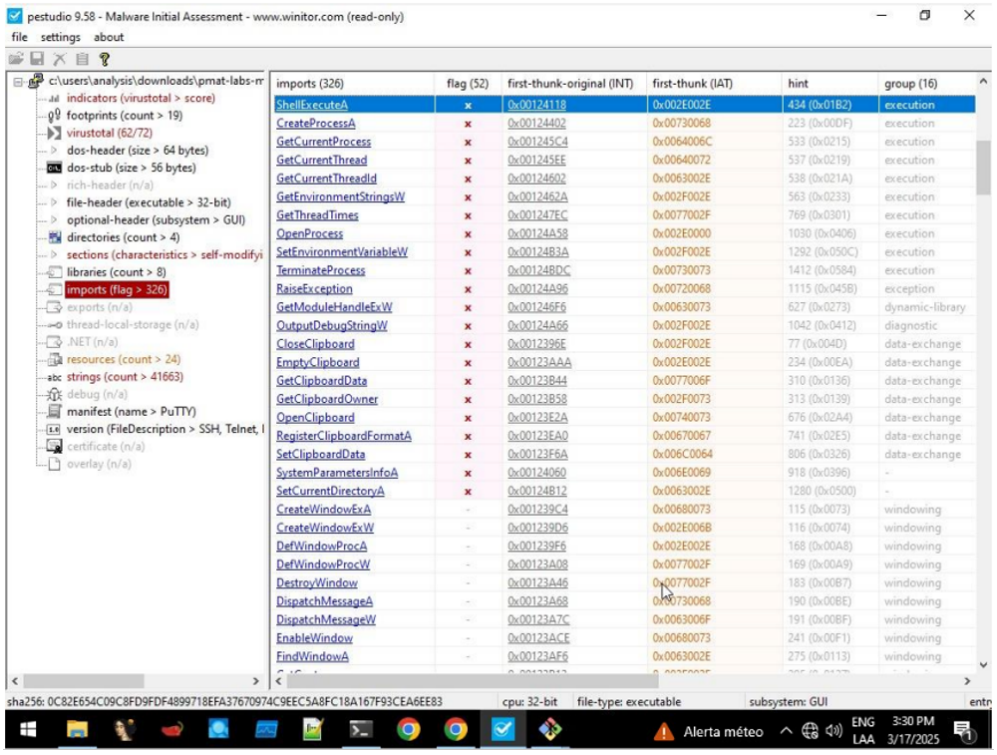
The study of the extracted strings indicates that the malware putty.exe possesses advanced functionalities to manipulate secure remote connections via SSH, interact with the file system and memory, and remain persistent on the system by modifying the Windows Registry. These strings suggest that the malware 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.





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.

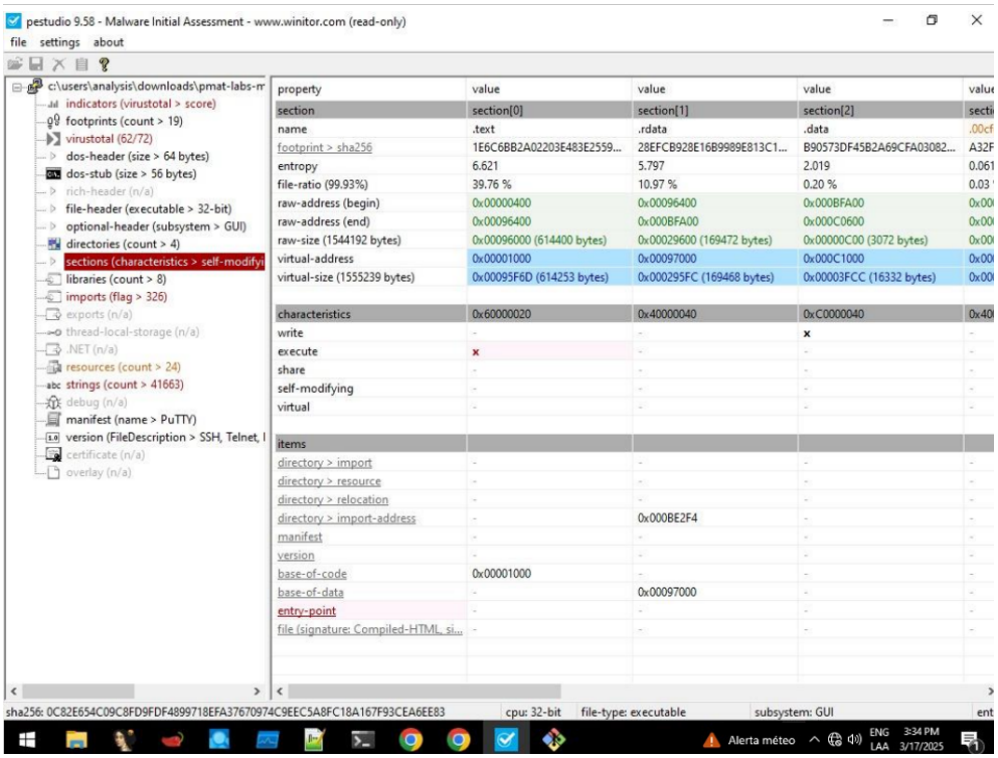
GetClipboardData and EmptyClipboard: These functions allow access to the contents of the system clipboard. Its presence in the binary suggests that the malware could be capturing sensitive information copied by the user, such as passwords or bank details.

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.



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.