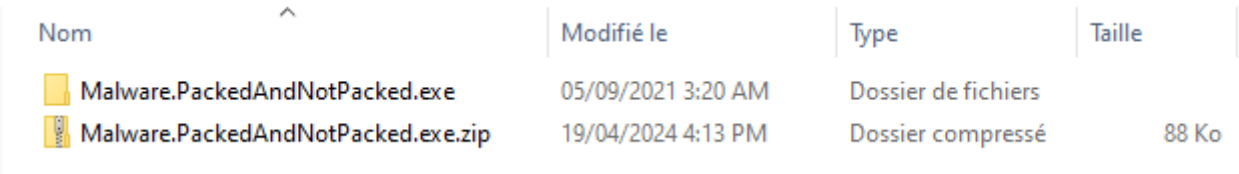
Malware Analysis Lab

This project demonstrates how to set up a secure malware analysis environment using FLARE VM on a Windows virtual machine. You'll explore static and dynamic analysis techniques, along with network monitoring and YARA rule-based detection, to thoroughly examine malware samples and uncover their behavior and potential threats.

Below are the samples we will be working with



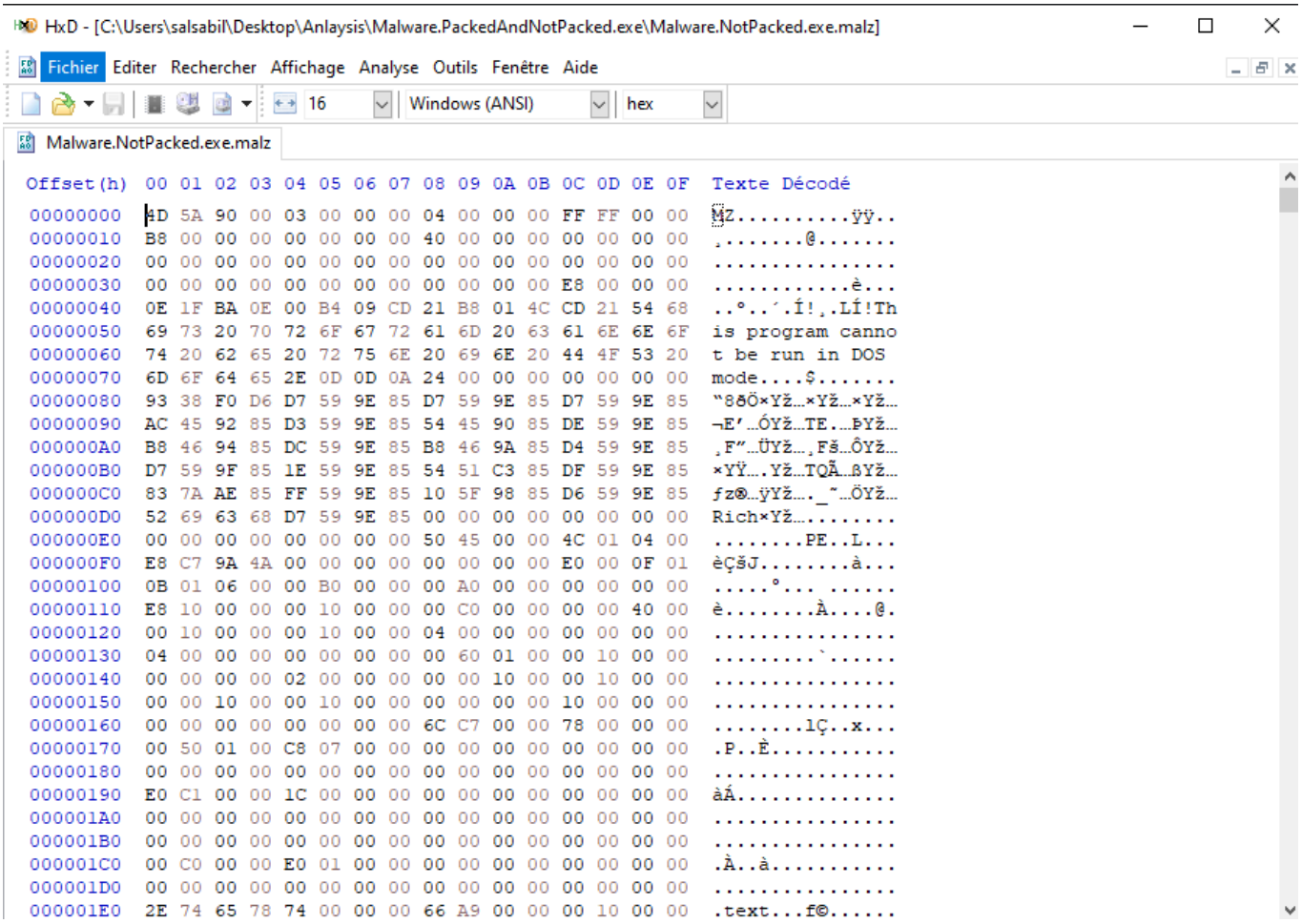
Nom	Modifié le	Type	Taille
Malware.PackedAndNotPacked.exe	05/09/2021 3:20 AM	Dossier de fichiers	
Malware.PackedAndNotPacked.exe.zip	19/04/2024 4:13 PM	Dossier compressé	88 Ko

First, we will conduct basic static analysis, and then move on to dynamic analysis

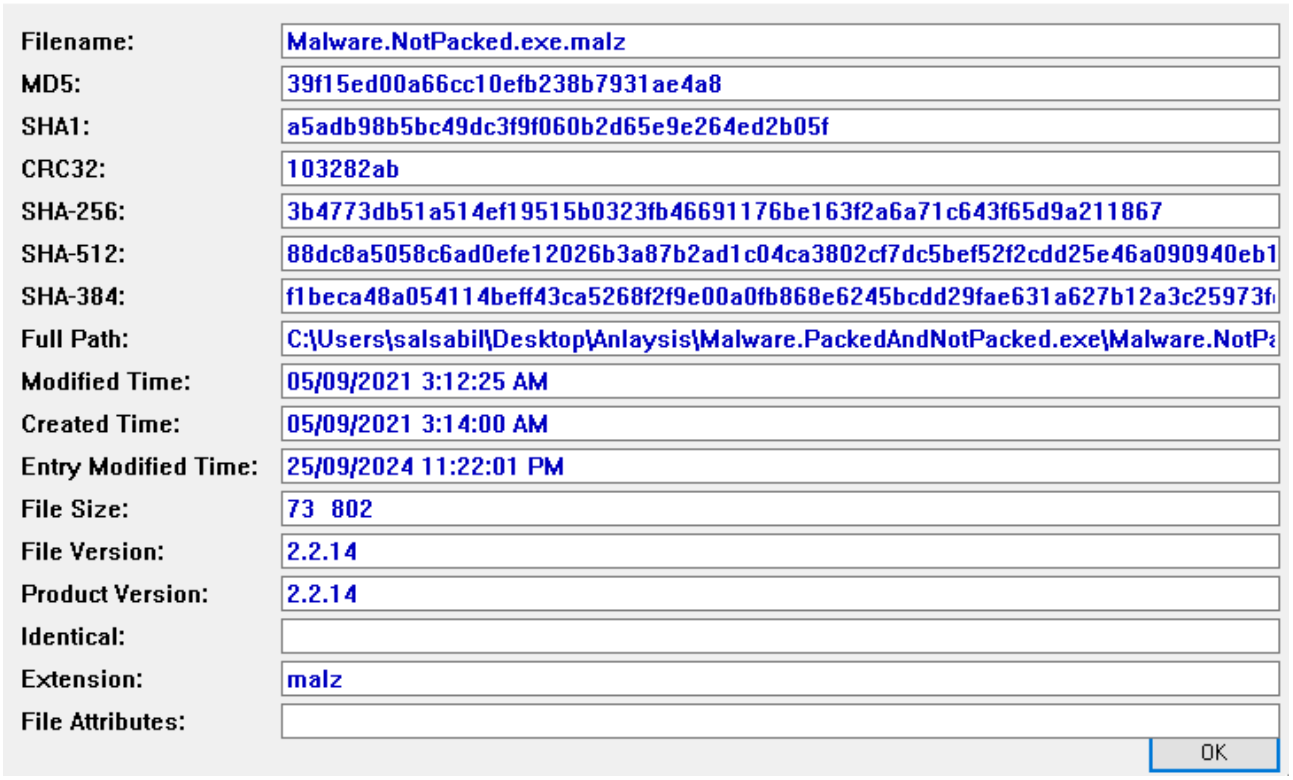
Basic Static Analysis

So with basic static analysis, what we are trying to do is examine the program or code and identify any malicious artifacts without running the actual program. We will be employing a few different tools to gather information

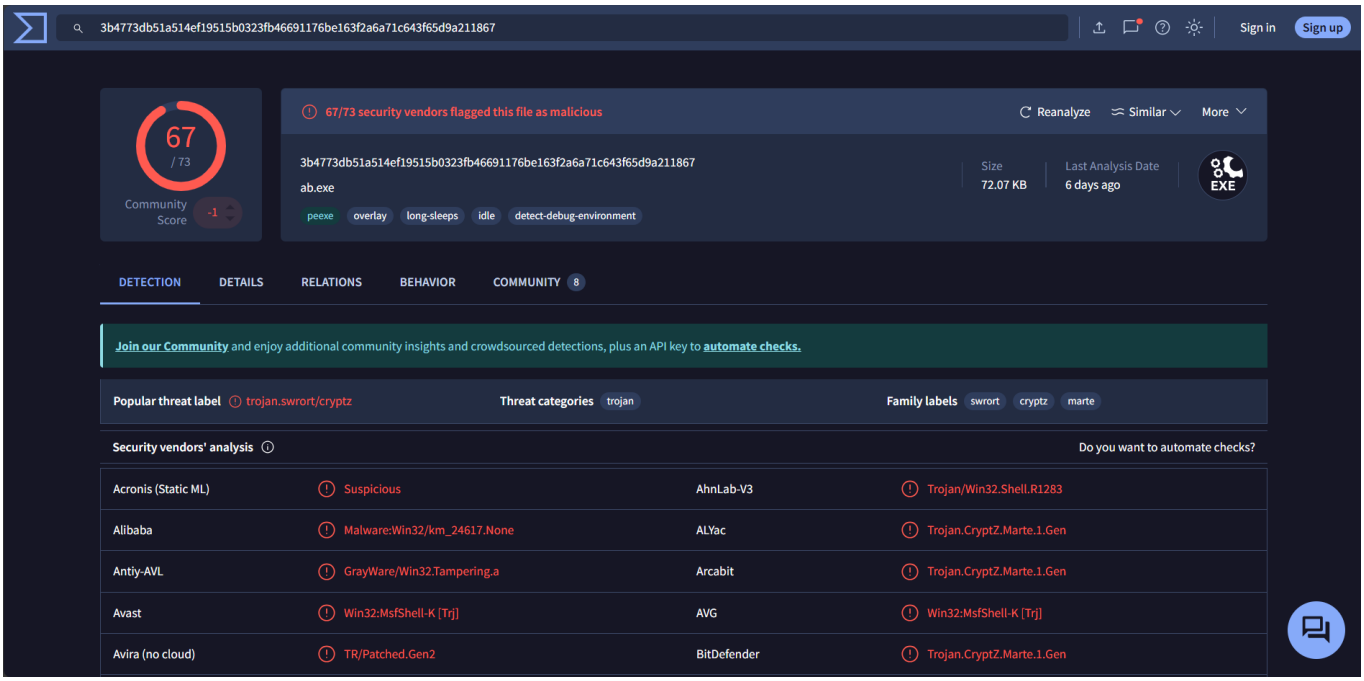
Starting with the HxD tool, we notice that the first two bytes are 4D 5A (MZ), indicating that this is an executable file.



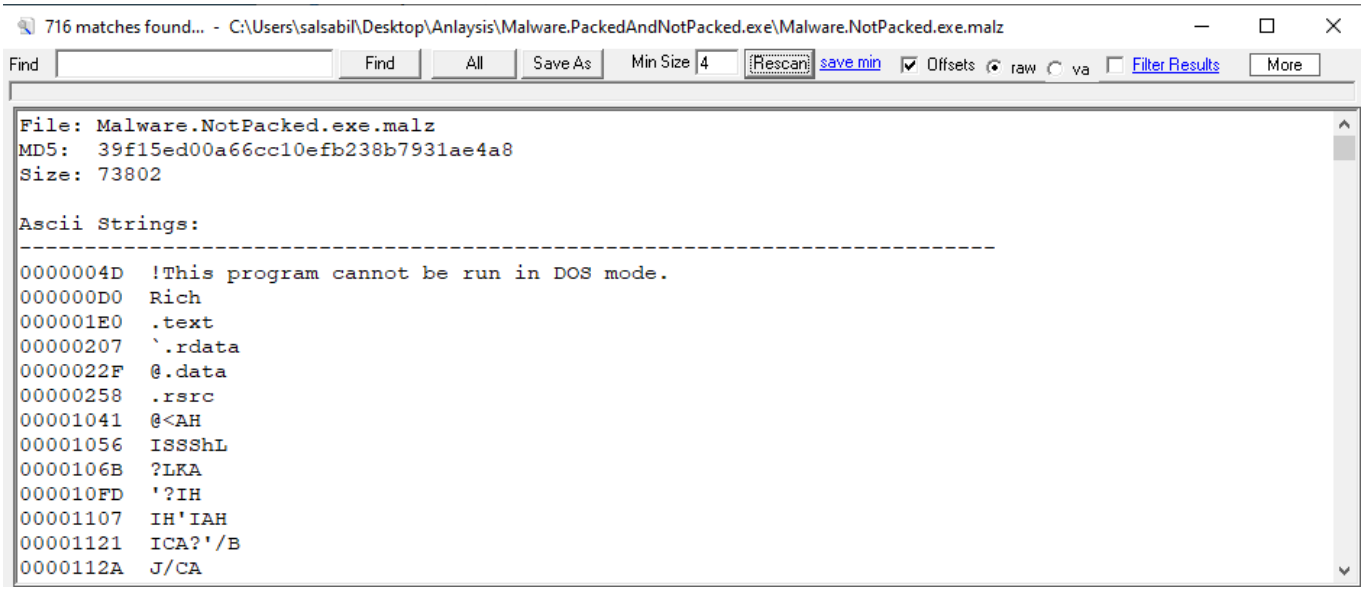
- We utilize HashmyFile to obtain the hashes



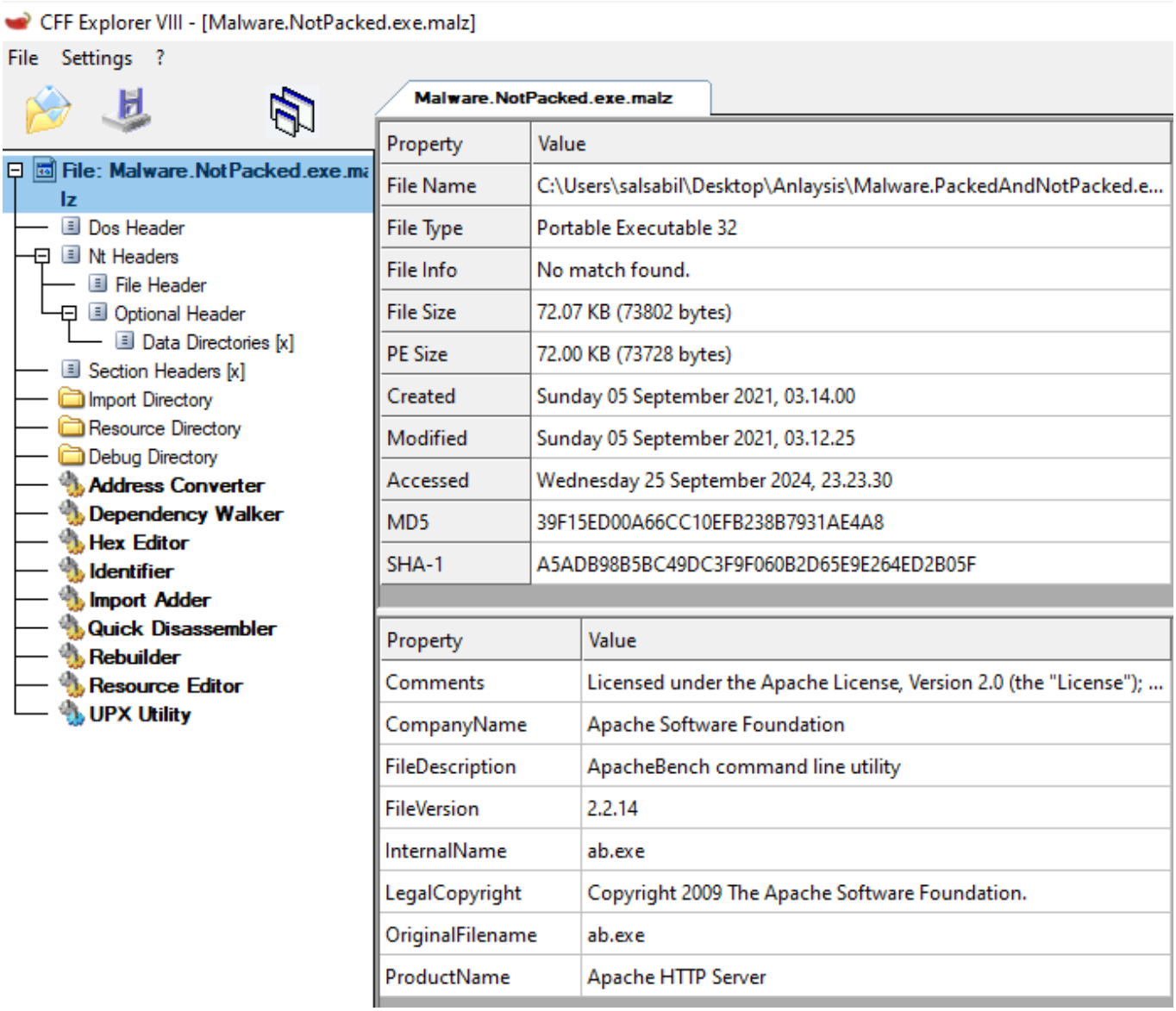
- We looked up the MD5 hash in VirusTotal and discovered that it’s a suspicious file



-Using the Strings tool, we identified the well-known message: '!This program cannot be run in DOS mode.



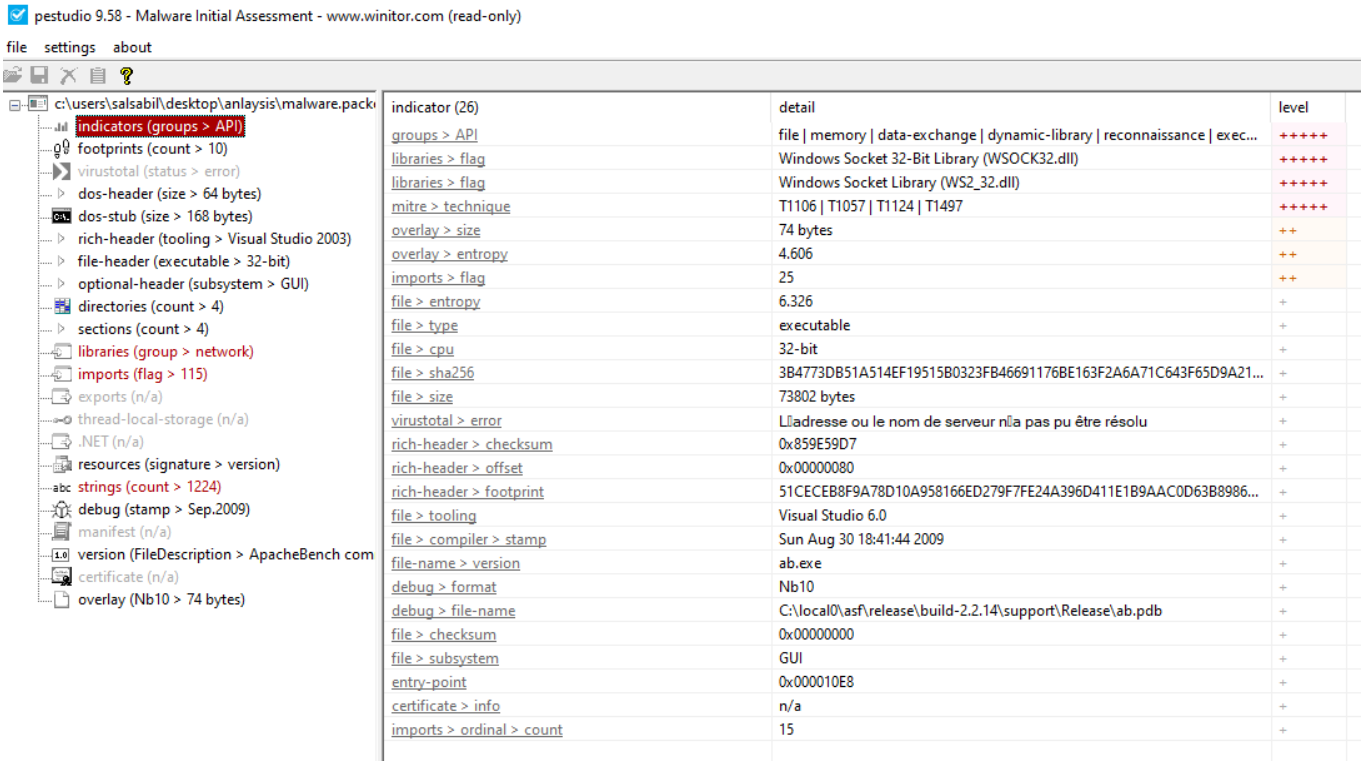
Further information can be gathered using CFF Explorer, including details such as creation date, version, and origin etc.



Now there are many ways to get information about the file that we are suspicious of. One easy-to-use tool is PeStudio, which is installed with the FlareVM. Just search the name and drag and drop the suspicious file into the program:

So with basic static analysis, what we are trying to do is examine the program or code and identify any malicious artifacts without running the actual program. We will be employing a few different tools to gather information. As we are on PeStudio, let's try to gather more information about the malware.

We will look for some interesting strings that may pop out, such as a URL, a domain name, an IP address, or maybe some different types of imports such as DLLs.



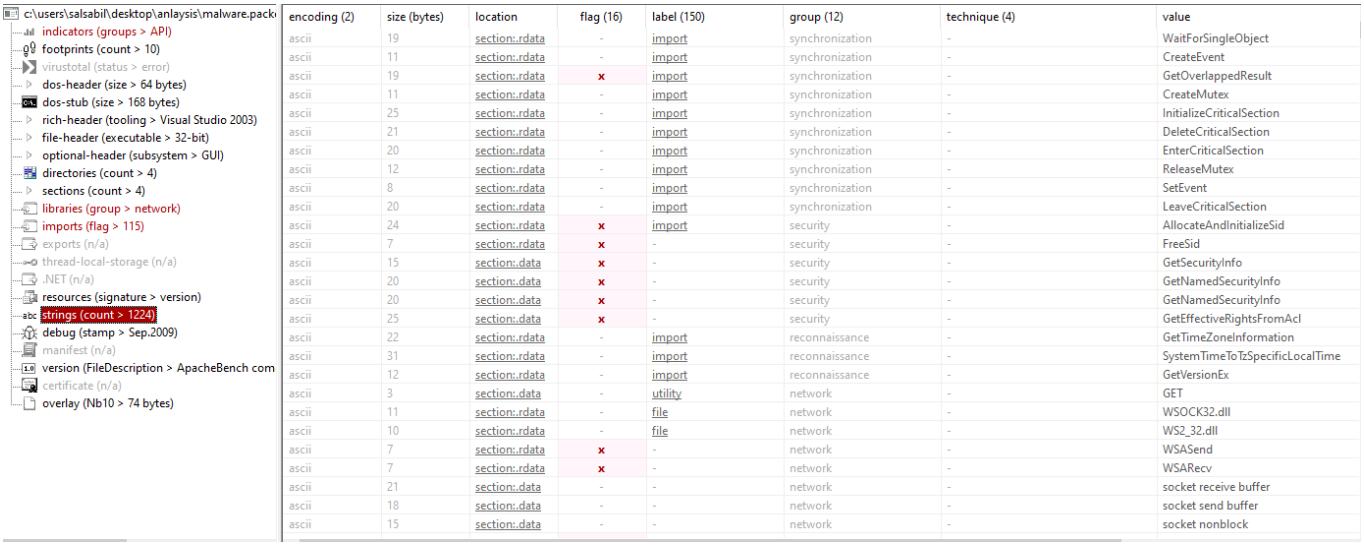
indicator (26)	detail	level
groups > API	file | memory | data-exchange | dynamic-library | reconnaissance | exec...	+++++
libraries > flag	Windows Socket 32-Bit Library (WSOCK32.dll)	+++++
libraries > flag	Windows Socket Library (WS2_32.dll)	+++++
mitre > technique	T1106 | T1057 | T1124 | T1497	+++++
overlay > size	74 bytes	++
overlay > entropy	4.606	++
imports > flag	25	++
file > entropy	6.326	+
file > type	executable	+
file > cpu	32-bit	+
file > sha256	3B4773DB51A514EF19515B0323FB46691176BE163F2A6A71C643F65D9A21...	+
file > size	73802 bytes	+
virustotal > error	L□adresse ou le nom de serveur n□a pas pu être résolu	+
rich-header > checksum	0x859E59D7	+
rich-header > offset	0x00000080	+
rich-header > footprint	51CECEB8F9A78D10A958166ED279F7FE24A396D411E1B9AAC0D63B8986...	+
file > tooling	Visual Studio 6.0	+
file > compiler > stamp	Sun Aug 30 18:41:44 2009	+
file-name > version	ab.exe	+
debug > format	Nb10	+
debug > file-name	C:\local0\asf\release\build-2.2.14\support\Release\ab.pdb	+
file > checksum	0x00000000	+
file > subsystem	GUI	+
entry-point	0x000010E8	+
certificate > info	n/a	+
imports > ordinal > count	15	+

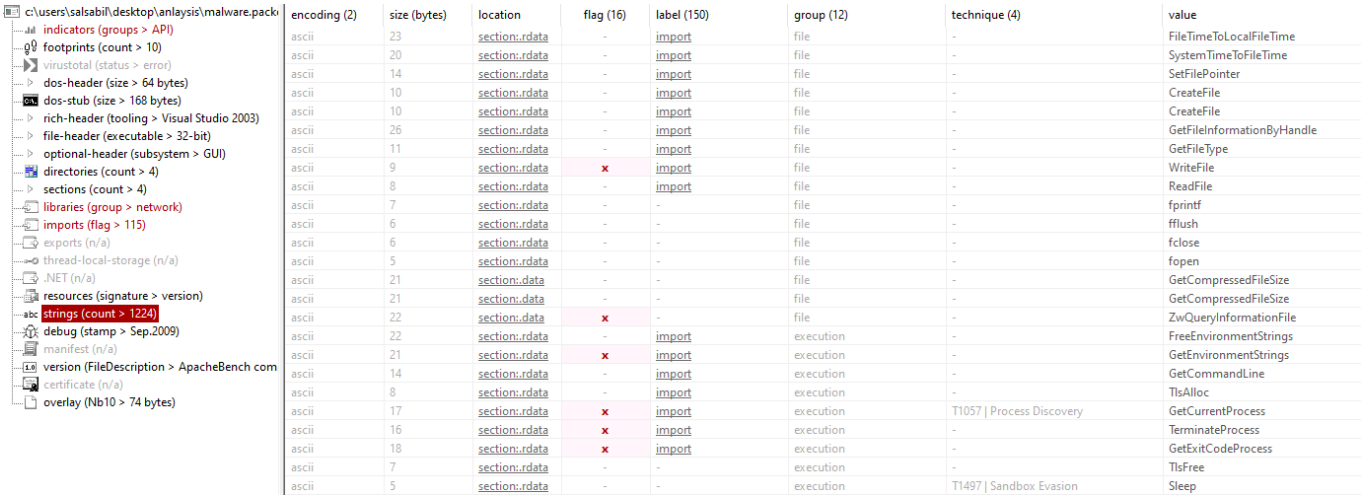
We analyzed the section field and found that it includes write and executable permissions

property	value	value	value	value
section	section[0]	section[1]	section[2]	section[3]
name	.text	.rdata	.data	.rsrc
footprint > sha256	90B179981F9BCEC4D57967A...	C9C158955ADA53055C12E5...	36C0AA22FB65D0F60AB7FC...	77D4D9B7BCF6235AC21DC6...
entropy	7.021	5.318	4.408	1.958
file-ratio (94.35%)	61.05 %	5.55 %	22.20 %	5.55 %
raw-address (begin)	0x00001000	0x0000C000	0x0000D000	0x00011000
raw-address (end)	0x0000C000	0x0000D000	0x00011000	0x00012000
raw-size (69632 bytes)	0x0000B000 (45056 bytes)	0x00001000 (4096 bytes)	0x00004000 (16384 bytes)	0x00001000 (4096 bytes)
virtual-address	0x00001000	0x0000C000	0x0000D000	0x00015000
virtual-size (78192 bytes)	0x0000A966 (43366 bytes)	0x00000FE6 (4070 bytes)	0x0000705C (28764 bytes)	0x000007C8 (1992 bytes)
characteristics	0x60000020	0x40000040	0xC0000040	0x40000040
write	-	-	x	-
execute	x	-	-	-
share	-	-	-	-
self-modifying	-	-	-	-
virtual	-	-	-	-
items				
directory > import	-	0x0000C76C	-	-
directory > resource	-	-	-	0x00015000
directory > debug	-	0x0000C1E0	-	-
directory > import-address	-	0x0000C000	-	-
version	-	-	-	0x00011060
base-of-code	0x00001000	-	-	-
base-of-data	-	0x0000C000	-	-
entry-point	0x000010E8	-	-	-

Next, we will explore the strings section, which is a crucial component of our analysis. The Strings tab in PeStudio allows us to identify human-readable strings within the malware, potentially revealing significant information such as URLs, IP addresses, file paths, registry keys, commands, and other indicators of compromise (IOCs). Analyzing these strings can offer

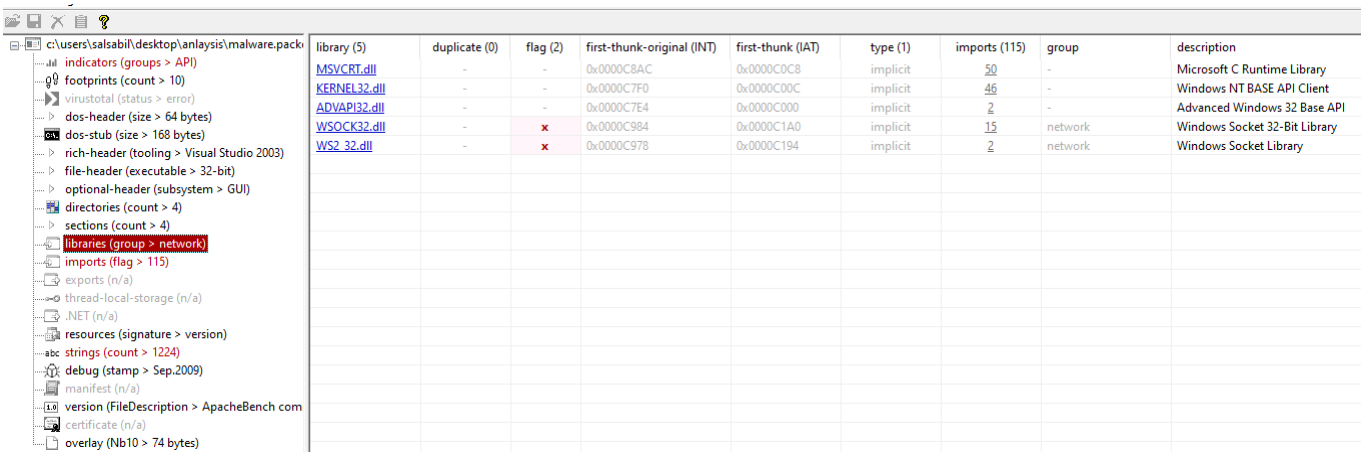
valuable insights into the malware's behavior, potential targets, and communication channels, thereby assisting in our understanding and mitigation of the threat.





By investigating these strings, we can uncover valuable insights into the malware's operations and potential impact.

Next, the Libraries tab shows DLLs that the malware imports, revealing its capabilities.



Key libraries include:

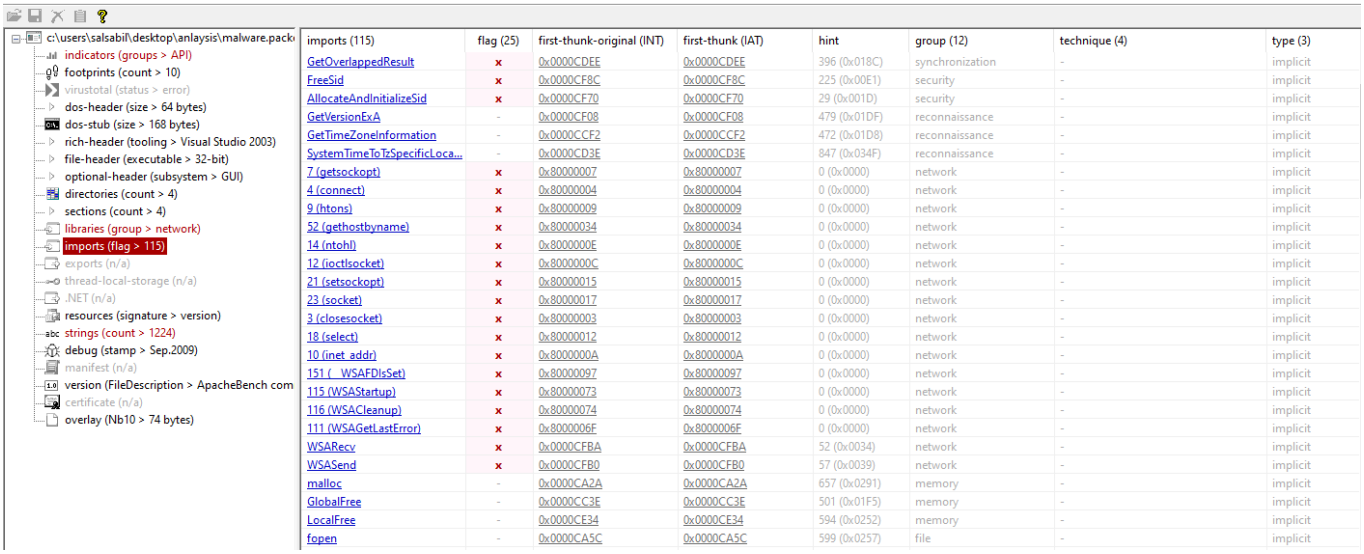
WSOK32.dll : This library is associated with the older Winsock API, which provides basic network functionality such as creating sockets, establishing connections, and data transmission over TCP/IP.

WS2 32.dll : This is the updated version of the Winsock library, offering more advanced networking capabilities compared to **WSOCK32.dll**, including support for both IPv4 and IPv6 protocols, and more complex network interactions.

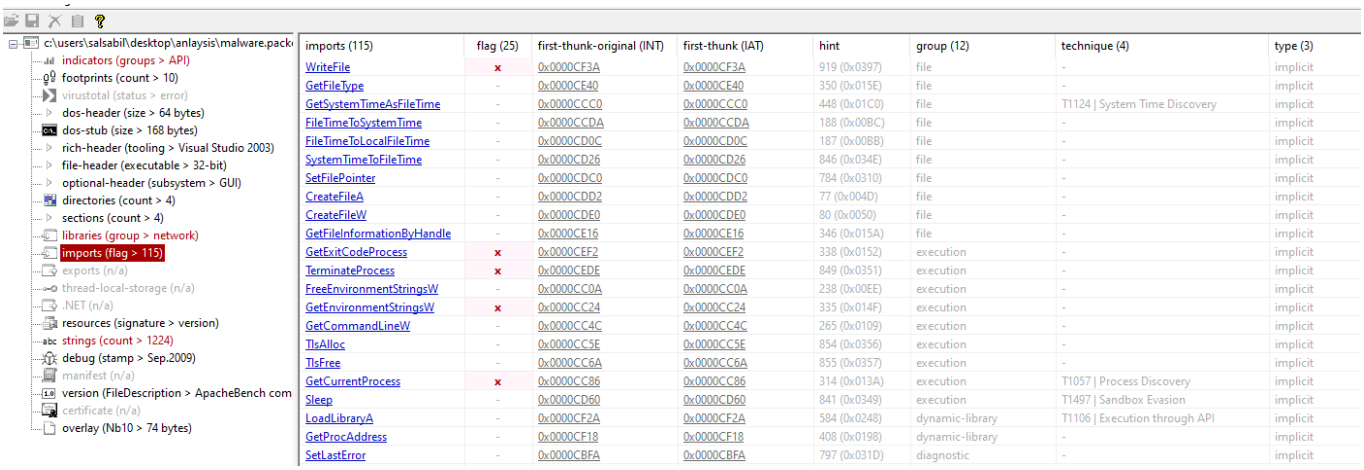
these two libraries are likely engaging in network-based activity, which could include:

- Communicating with a remote C2 server.
- Exfiltrating stolen data.
- Spreading itself over a network.

Next we will go with the imports
This combination indicates that the malware is involved in both system manipulation (files, processes) and network-based operations, which could be part of a data exfiltration or command-and-control mechanism.



imports (115)	flag (25)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (12)	technique (4)	type (3)
GetOverlappedResult	x	0x0000CDEE	0x0000CDEE	396 (0x018C)	synchronization	-	implicit
FreeSid	x	0x0000CF8C	0x0000CF8C	225 (0x00E1)	security	-	implicit
AllocateAndInitializeSid	x	0x0000CF70	0x0000CF70	29 (0x001D)	security	-	implicit
GetVersionExA	-	0x0000CF08	0x0000CF08	479 (0x01DF)	reconnaissance	-	implicit
GetTimeZoneInformation	-	0x0000CCF2	0x0000CCF2	472 (0x01D8)	reconnaissance	-	implicit
SystemTimeToTzSpecificLoca...	-	0x0000CD3E	0x0000CD3E	847 (0x034F)	reconnaissance	-	implicit
7 (getsockopt)	x	0x80000007	0x80000007	0 (0x0000)	network	-	implicit
4 (connect)	x	0x80000004	0x80000004	0 (0x0000)	network	-	implicit
9 (htons)	x	0x80000009	0x80000009	0 (0x0000)	network	-	implicit
52 (gethostbyname)	x	0x80000034	0x80000034	0 (0x0000)	network	-	implicit
14 (ntohl)	x	0x8000000E	0x8000000E	0 (0x0000)	network	-	implicit
12 (ioctlsocket)	x	0x8000000C	0x8000000C	0 (0x0000)	network	-	implicit
21 (setsockopt)	x	0x80000015	0x80000015	0 (0x0000)	network	-	implicit
23 (socket)	x	0x80000017	0x80000017	0 (0x0000)	network	-	implicit
3 (closesocket)	x	0x80000003	0x80000003	0 (0x0000)	network	-	implicit
18 (select)	x	0x80000012	0x80000012	0 (0x0000)	network	-	implicit
10 (inet_addr)	x	0x8000000A	0x8000000A	0 (0x0000)	network	-	implicit
151 (__WSAFDIsSet)	x	0x80000097	0x80000097	0 (0x0000)	network	-	implicit
115 (WSAStartup)	x	0x80000073	0x80000073	0 (0x0000)	network	-	implicit
116 (WSACleanup)	x	0x80000074	0x80000074	0 (0x0000)	network	-	implicit
111 (WSAGetLastError)	x	0x8000006F	0x8000006F	0 (0x0000)	network	-	implicit
WSARecv	x	0x0000CFBA	0x0000CFBA	52 (0x0034)	network	-	implicit
WSASend	x	0x0000CFB0	0x0000CFB0	57 (0x0039)	network	-	implicit
malloc	-	0x0000CA2A	0x0000CA2A	657 (0x0291)	memory	-	implicit
GlobalFree	-	0x0000CC3E	0x0000CC3E	501 (0x01F5)	memory	-	implicit
LocalFree	-	0x0000CE34	0x0000CE34	594 (0x0252)	memory	-	implicit
fopen	-	0x0000CA5C	0x0000CA5C	599 (0x0257)	file	-	implicit



imports (115)	flag (25)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (12)	technique (4)	type (3)
WriteFile	x	0x0000CF3A	0x0000CF3A	919 (0x0397)	file	-	implicit
GetFileType	-	0x0000CE40	0x0000CE40	350 (0x015E)	file	-	implicit
GetSystemTimeAsFileTime	-	0x0000CCC0	0x0000CCC0	448 (0x01C0)	file	T1124 | System Time Discovery	implicit
FileTimeToSystemTime	-	0x0000CCDA	0x0000CCDA	188 (0x00BC)	file	-	implicit
FileTimeToLocalFileTime	-	0x0000CD0C	0x0000CD0C	187 (0x00BB)	file	-	implicit
SystemTimeToFileTime	-	0x0000CD26	0x0000CD26	846 (0x034E)	file	-	implicit
SetFilePointer	-	0x0000CDC0	0x0000CDC0	784 (0x0310)	file	-	implicit
CreateFileA	-	0x0000CDD2	0x0000CDD2	77 (0x004D)	file	-	implicit
CreateFileW	-	0x0000CDE0	0x0000CDE0	80 (0x0050)	file	-	implicit
GetFileInformationByHandle	-	0x0000CE16	0x0000CE16	346 (0x015A)	file	-	implicit
GetExitCodeProcess	x	0x0000CEF2	0x0000CEF2	338 (0x0152)	execution	-	implicit
TerminateProcess	x	0x0000CEDE	0x0000CEDE	849 (0x0351)	execution	-	implicit
FreeEnvironmentStringsW	-	0x0000CC0A	0x0000CC0A	238 (0x00EE)	execution	-	implicit
GetEnvironmentStringsW	x	0x0000CC24	0x0000CC24	335 (0x014F)	execution	-	implicit
GetCommandLineW	-	0x0000CC4C	0x0000CC4C	265 (0x0109)	execution	-	implicit
TlsAlloc	-	0x0000CC5E	0x0000CC5E	854 (0x0356)	execution	-	implicit
TlsFree	-	0x0000CC6A	0x0000CC6A	855 (0x0357)	execution	-	implicit
GetCurrentProcess	x	0x0000CC86	0x0000CC86	314 (0x013A)	execution	T1057 | Process Discovery	implicit
Sleep	-	0x0000CD60	0x0000CD60	841 (0x0349)	execution	T1497 | Sandbox Evasion	implicit
LoadLibraryA	-	0x0000CF2A	0x0000CF2A	584 (0x0248)	dynamic-library	T1106 | Execution through API	implicit
GetProcAddress	-	0x0000CF18	0x0000CF18	408 (0x0198)	dynamic-library	-	implicit
SetLastError	-	0x0000CBFA	0x0000CBFA	797 (0x031D)	diagnostic	-	implicit

Conclusion

In this malware analysis lab, we thoroughly examined a suspicious executable file to understand its behavior and impact. We began by performing static analysis to inspect the file's structure, identifying key libraries and functions that suggested malicious activities