Assignment 1: Basic Static and Dynamic Analysis of david and bob Malware Samples

Thomas Green
School of Engineering and Computer Science, Victoria University of Wellington

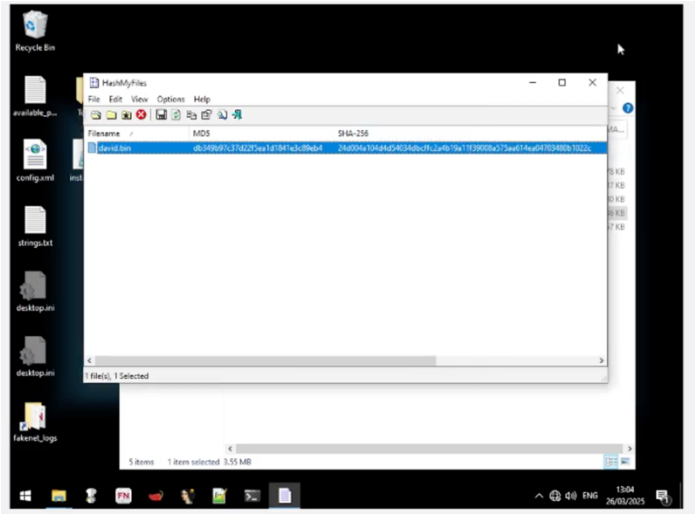
1. Analysis Setup

Analysis environment was set up using VirtualBox on an ECS lab machine with a FlareVM virtual machine for malware execution and a REMnux virtual machine configured with FakeDNS for DNS resolution. Network configuration consisted of NAT and host-only networking to isolate malware behaviour while enabling limited internet simulation.

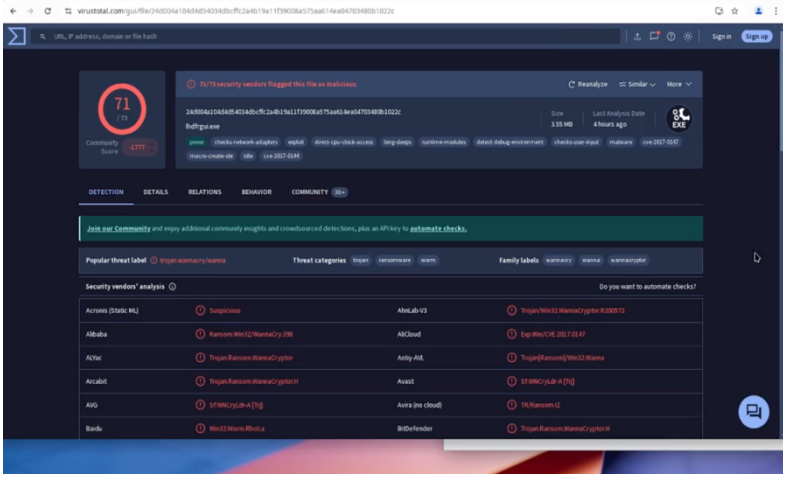
2. David

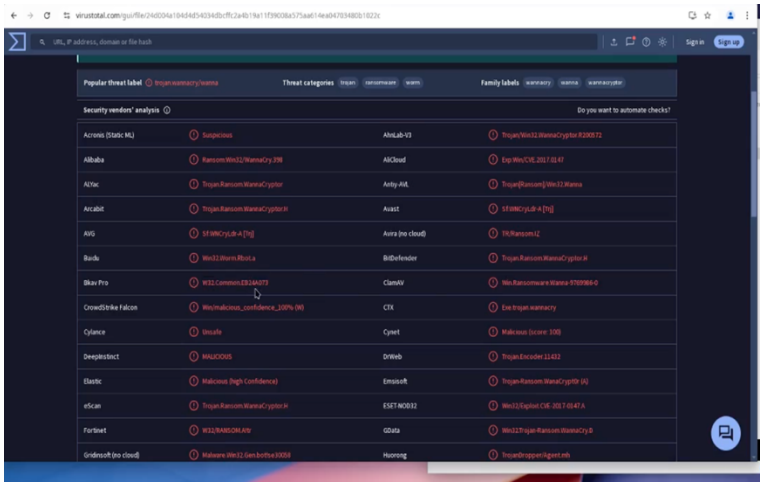
2.1. Basic Static Analysis

VirusTotal

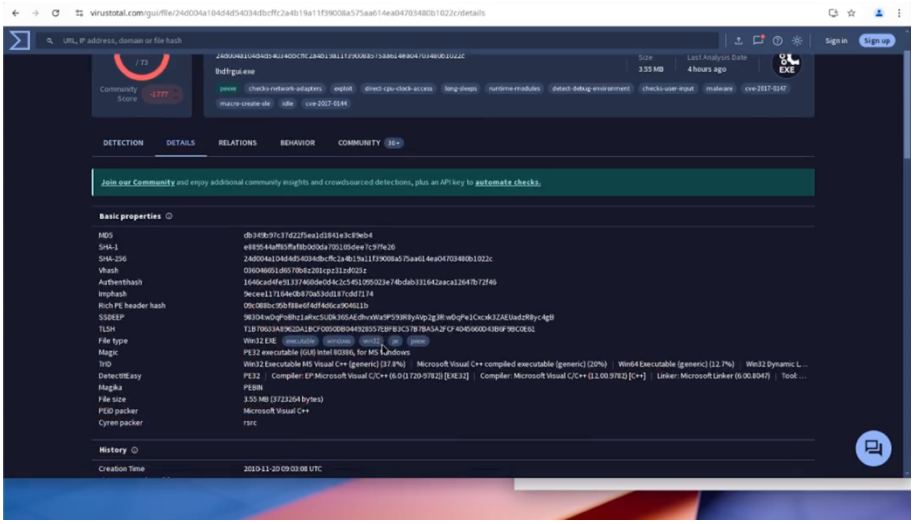


Began investigation acquiring MD5 hash of file to search within VirusTotal.

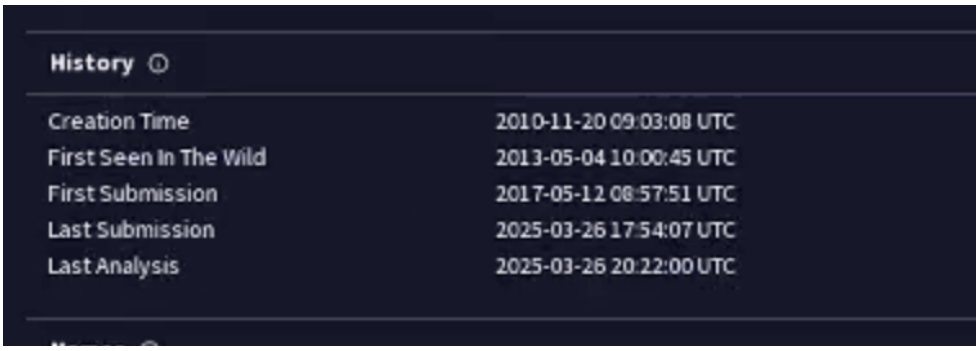




71 out of 73 vendors identified the malware as malicious. Identified by antivirus engines under names such as Trojan.Ransom.WannaCryptor.H and Ransom:Win32/WannaCry.398. I can determine that david.bin is related to WannaCry ransomware, which encrypts files, demands Bitcoin ransom, and has worm-spreading behaviour. Did not manually compare AV signatures with ClamAV, however, VirusTotal includes ClamAV engine, which confirms detection – Win.Ransomware.Wanna-9769986-0.

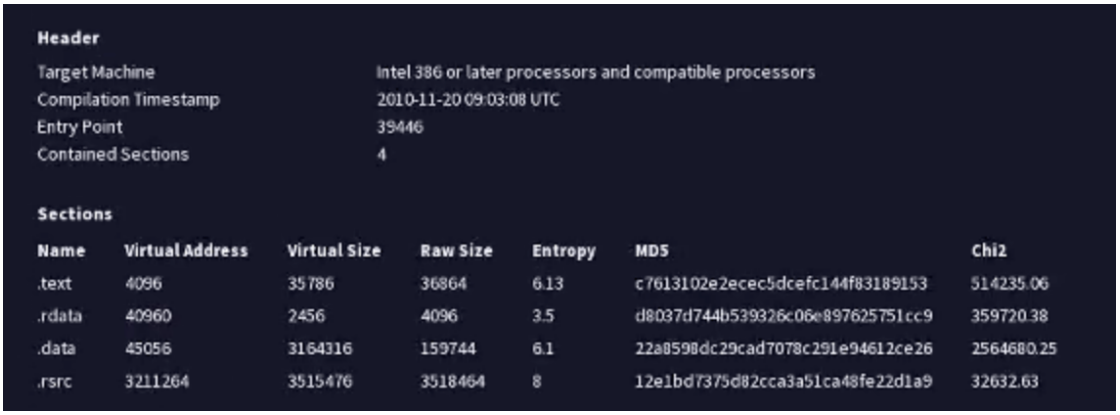


The malware compiler is Microsoft Visual C++. Can identify the MD5 and SHA-256 hashes that match the .bin malware. We can observe that malware is PE32 architecture, meaning it's not immediately obfuscated. Cyren packer indicates a .rsrc packer, signifying that resource packing is used potentially to hide payloads or avoid detection.



The creation time was 2010-11-20 - when the binary was first compiled.

Malware does some sort of file version spoofing to appear legitimate. The original/internal name was lhdfgui.exe. Malware is impersonating a legitimate Windows utility – Windows Disk Defragmenter. It uses this technique to avoid suspicion from casual observers.



Header

Target Machine	Intel 386 or later processors and compatible processors
Compilation Timestamp	2010-11-20 09:03:08 UTC
Entry Point	39446
Contained Sections	4

Sections

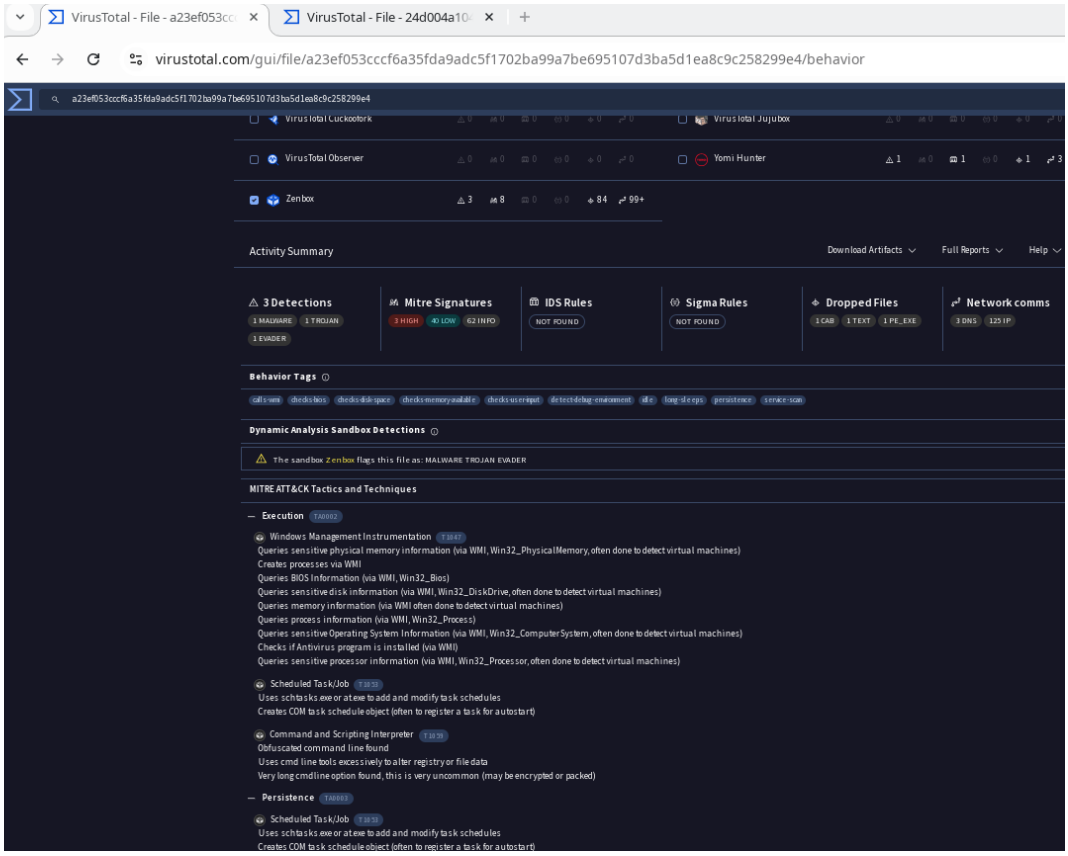
Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	35786	36864	6.13	c7613102e2ecec5dcefc144f83189153	514235.06
.rdata	40960	2456	4096	3.5	d8037d744b539326c06e897625751cc9	359720.38
.data	45056	3164316	159744	6.1	22a8598dc29cad7078c291e94612ce26	2564680.25
.rsrc	3211264	3515476	3518464	8	12e1bd7375d82cca3a51ca48fe22d1a9	32632.63

PE header indicates the malware targets Intel386 or later processes. Has four contained sections: .text, .rdata, .data and .rsrc. Indicates 6.13, 3.5, and 6.1 entropy for .text, .rdata and .data respectively. This is relatively low and doesn't indicate packing. I can see .rsrc has high entropy of 8, indicating a likely use of obfuscation or packing. The MD5 hashes of each section can be observed, which is helpful for threat hunting, signature matching, and detection.

Imports

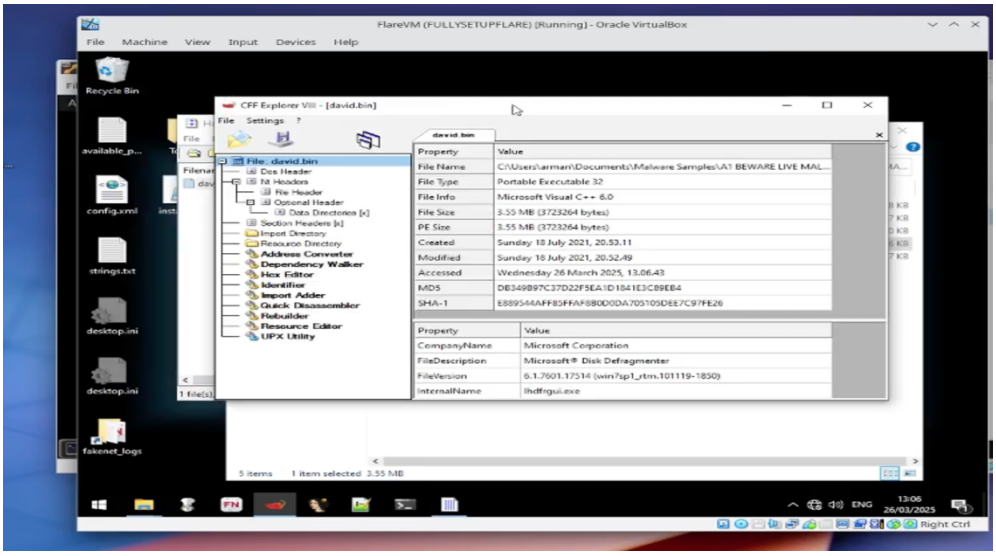
- \+ KERNEL32.dll
- \+ ADVAPI32.dll
- \+ WS2_32.dll
- \+ MSVCP60.dll
- \+ iphlpapi.dll
- \+ WININET.dll
- \+ MSVCRT.dll

Can see the DLLs that the malware uses. Malware to be aware of while analyzing are WS2_32.dll and WININET.dll library, which is the Windows socket API library which implies the use of internet functions such as HTTP requests. Also, iphlpapi.dll allows the program to access information about the system IP configuration and routing table.

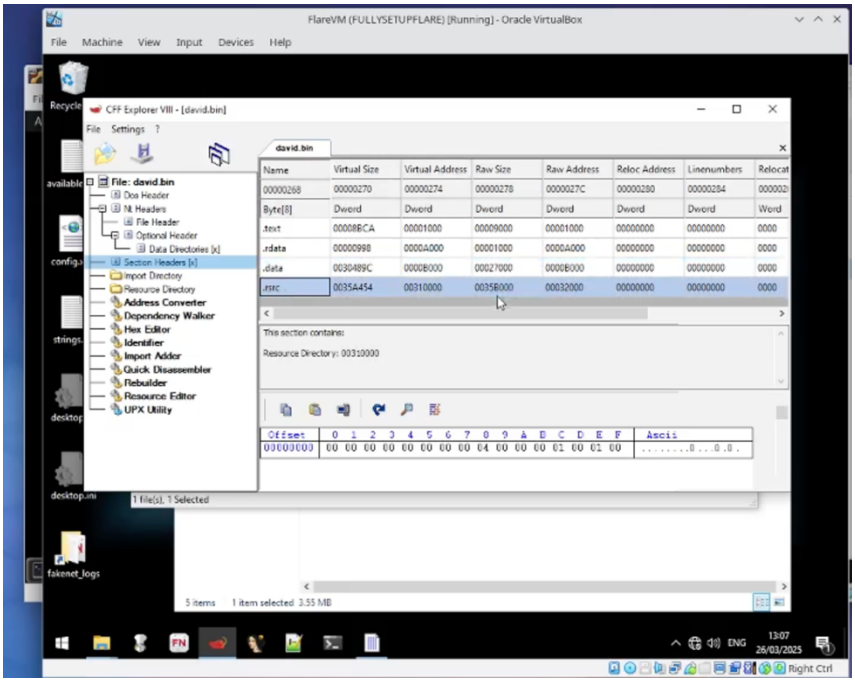


Analyzed the Sandbox report done by Zenbox on VirusTotal. Execution uses command tools extensively for registry/file manipulation and scheduled tasks. For persistence, it uses registry run keys and startup folder to auto-start the malware upon reboot.

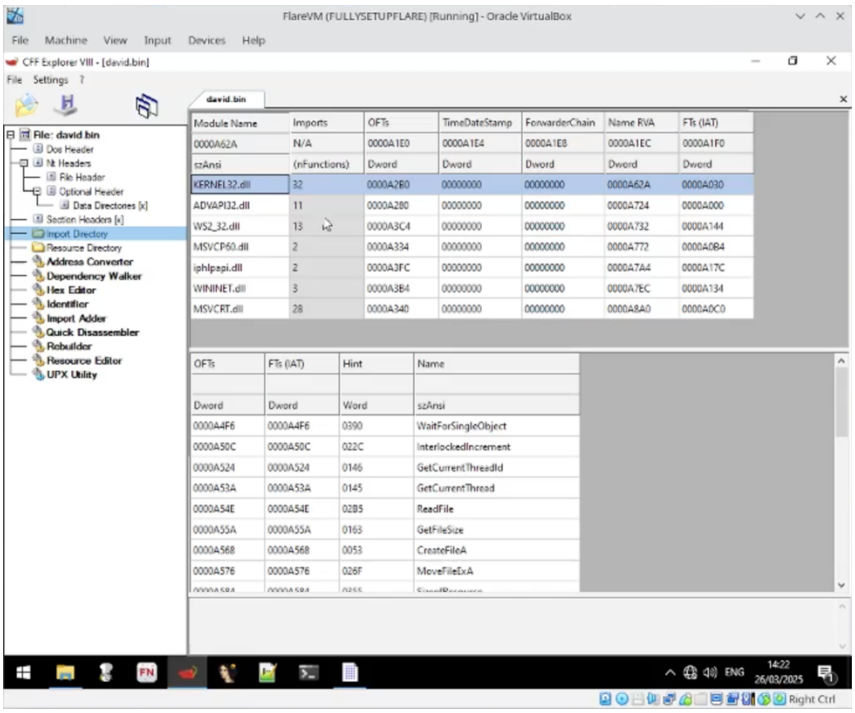
CFF Explorer



Confirms the malware is 32-bit PE and is compiled using Microsoft Visual C++. It also shows the internal name lhdfgui.exe and is trying to pass off as a Microsoft Disk Defragmenter. You can also see creation and modification date, which doesn't match the timestamps we acknowledged earlier in VirusTotal.

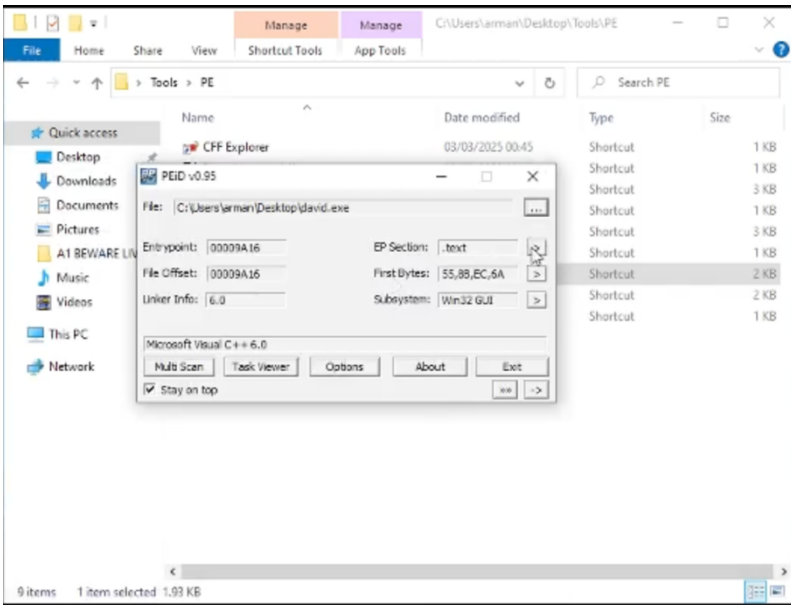


Analyzed PE Section headers. Clear that the malware is not packed with tools like UPX (e.g. UPX0, UPX1). .data section appears larger than the others, which may indicate embedded payloads or buffers.

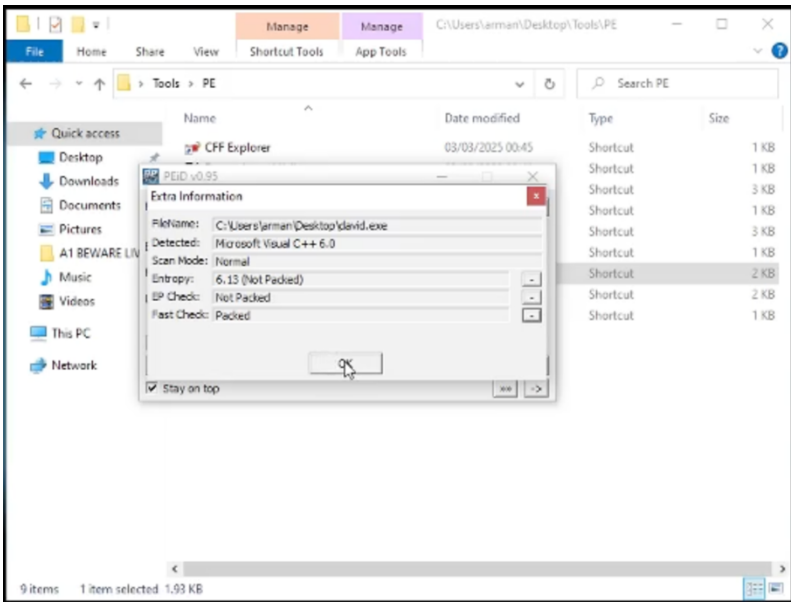


We can see DLLs identified earlier in VirusTotal. Shows that the malware is capable of performing system-level operations such as registry access, networking and file handling. The malware can use functions such as CreateFileA, ReadFile, and MoveFileExA from KERNEL32.dll.

PEiD

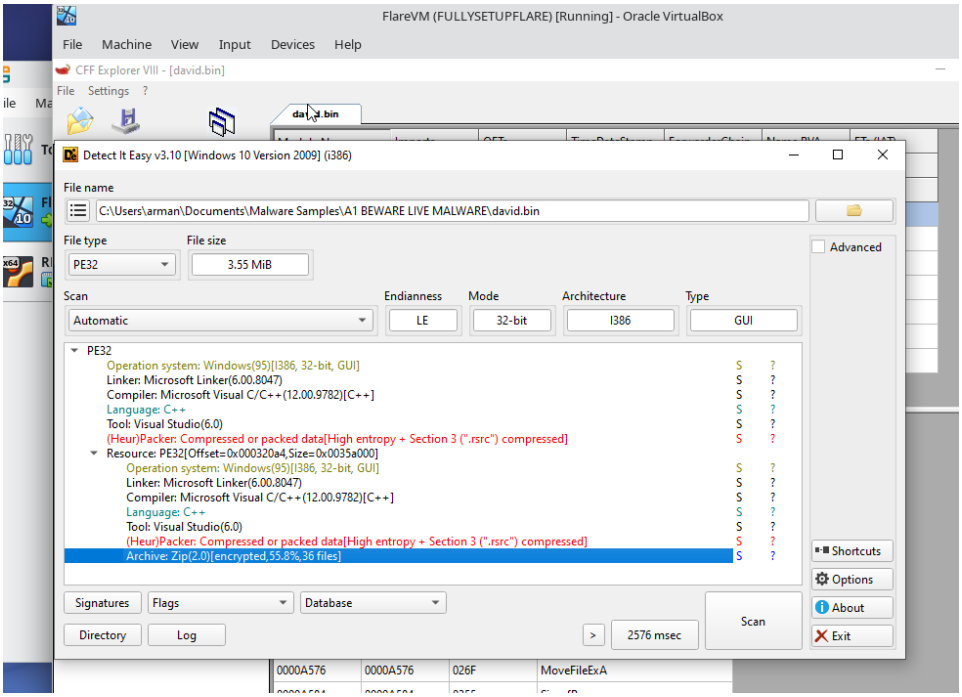


Identifies the entry point of file, which lies in the .text section. Identified malware was compiled using Microsoft Visual C++.



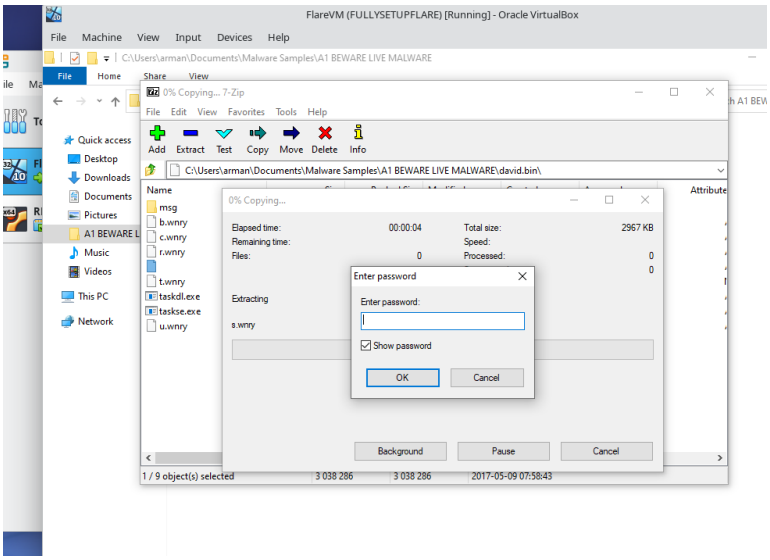
Malware has an entropy of 6.13, which generally falls within the range of unpacked/lightly obfuscated binaries. EP Check also reported not packed. Fast Check says packed, but this sometimes produces false positive results. Cyren Packer, an antivirus engine that detects characteristics of packed/obfuscated files, indicated that .rsrc was packed.

Detect it Easy



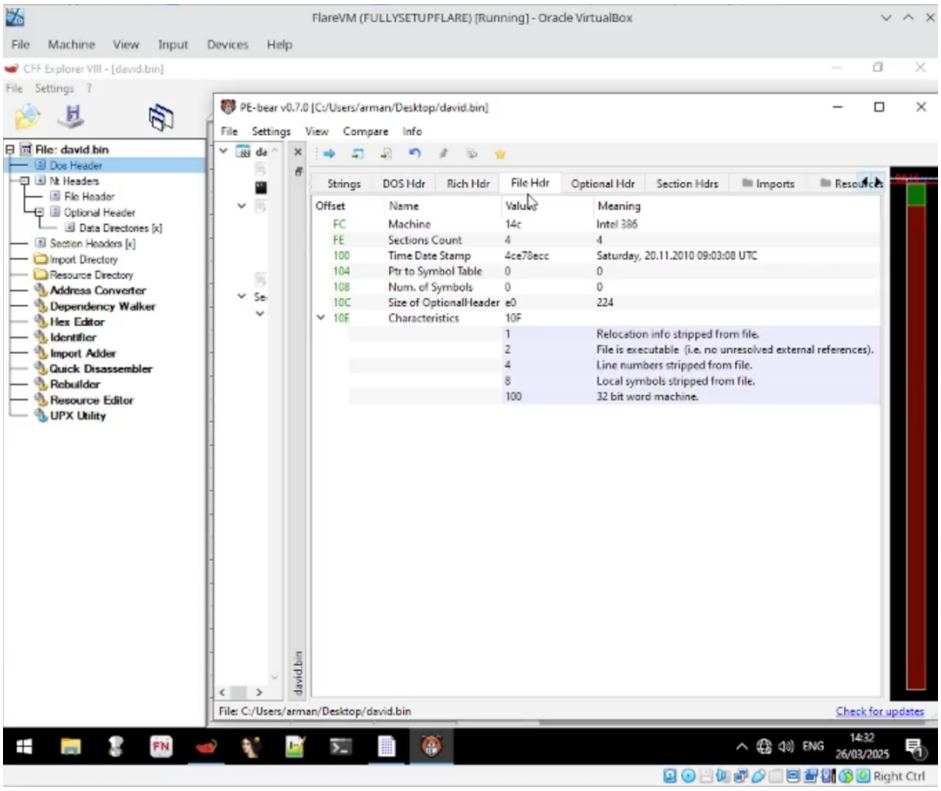
Section 3 (.rsrc) is both compressed and has high entropy, which suggests that it's compressed or packed data. Additionally identified, the .rsrc section contains a ZIP archive with 36 encrypted files. This may become unpacked/executed at runtime.

7-Zip

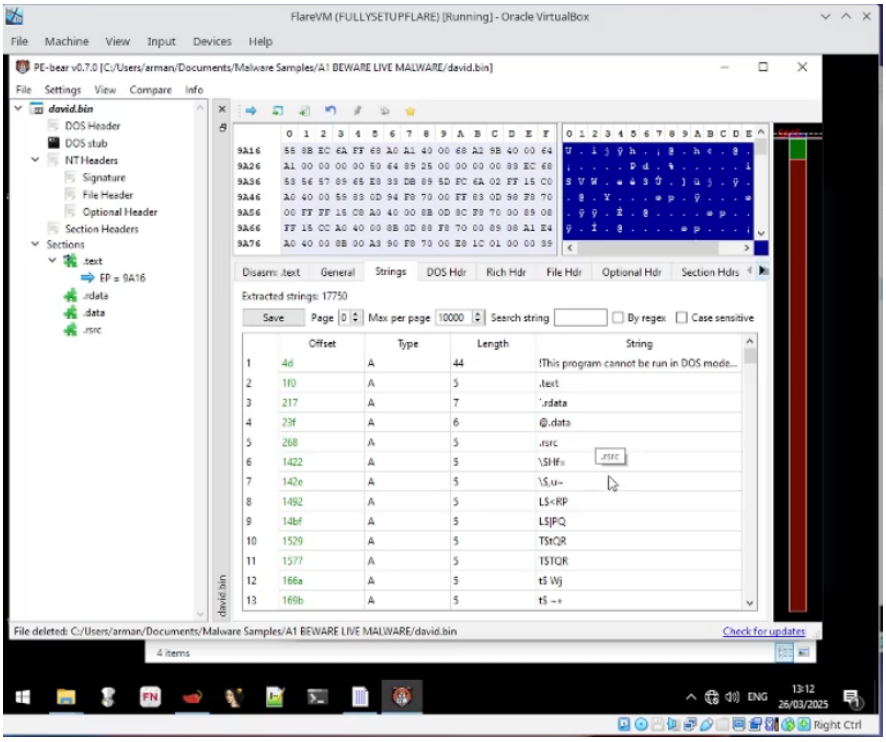


I tried to use 7-zip tool to extract the .rsrc zip contents; however, the archives were password-protected.

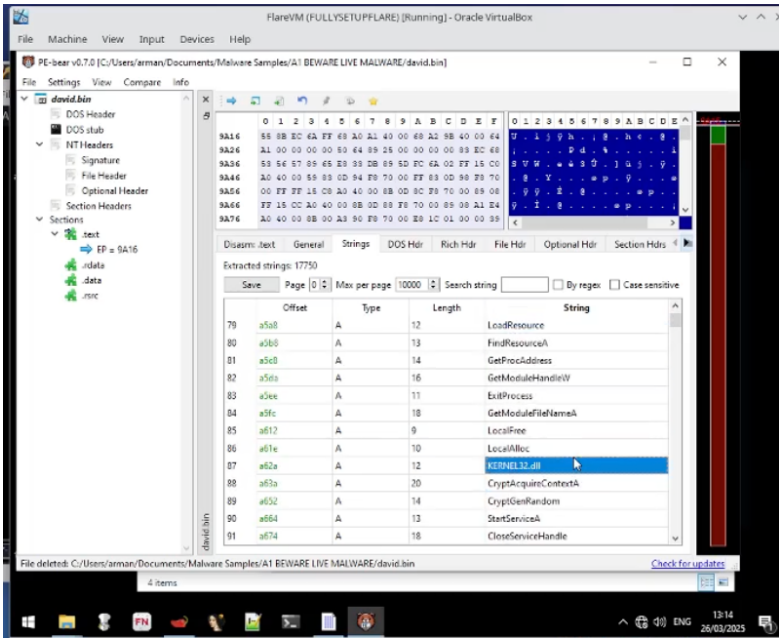
PE-Bear

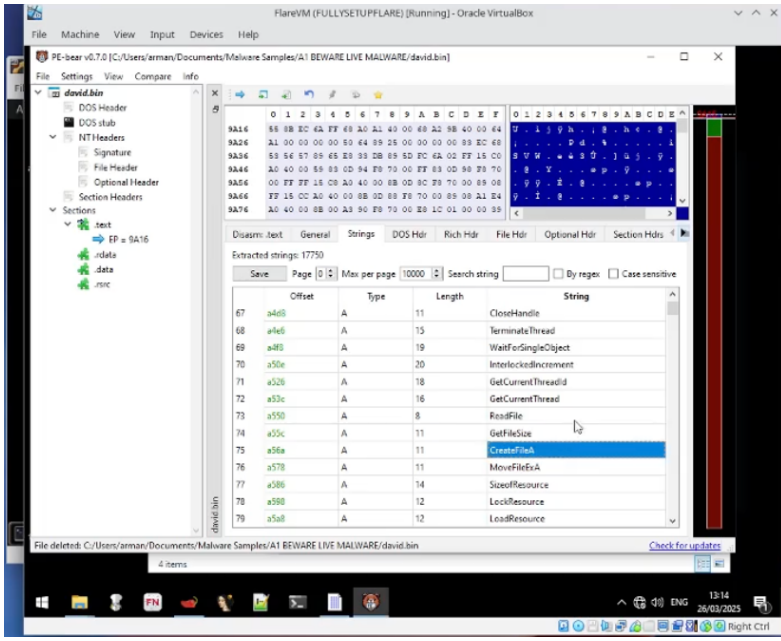


File header confirmed compilation date of david.bin. Characteristics revealed the file is executable, and several components have been stripped, such as relocation info, local symbols, and line numbers. Confirmed compilation date 20.11.2010, as seen earlier in VirusTotal.

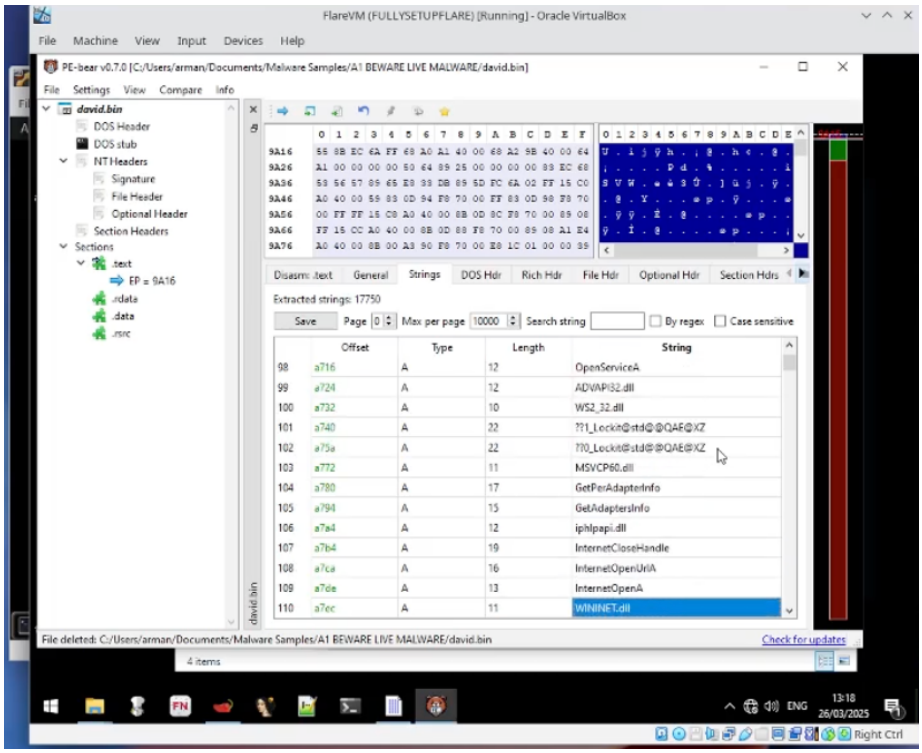


Identified 17750 extracted strings. See initial PE Section references for .text, .rdata, .data and .rsrc. You can also see a bunch of encoded or obfuscated strings at the start as well.

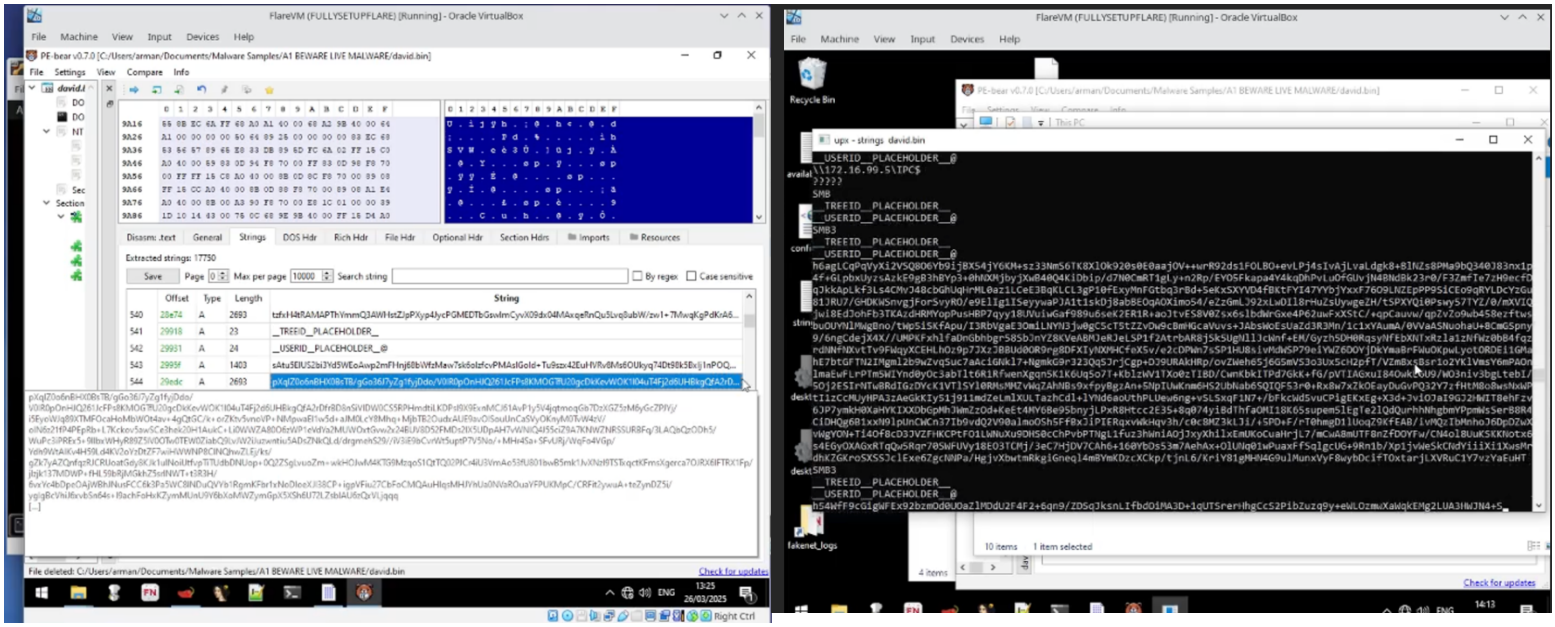




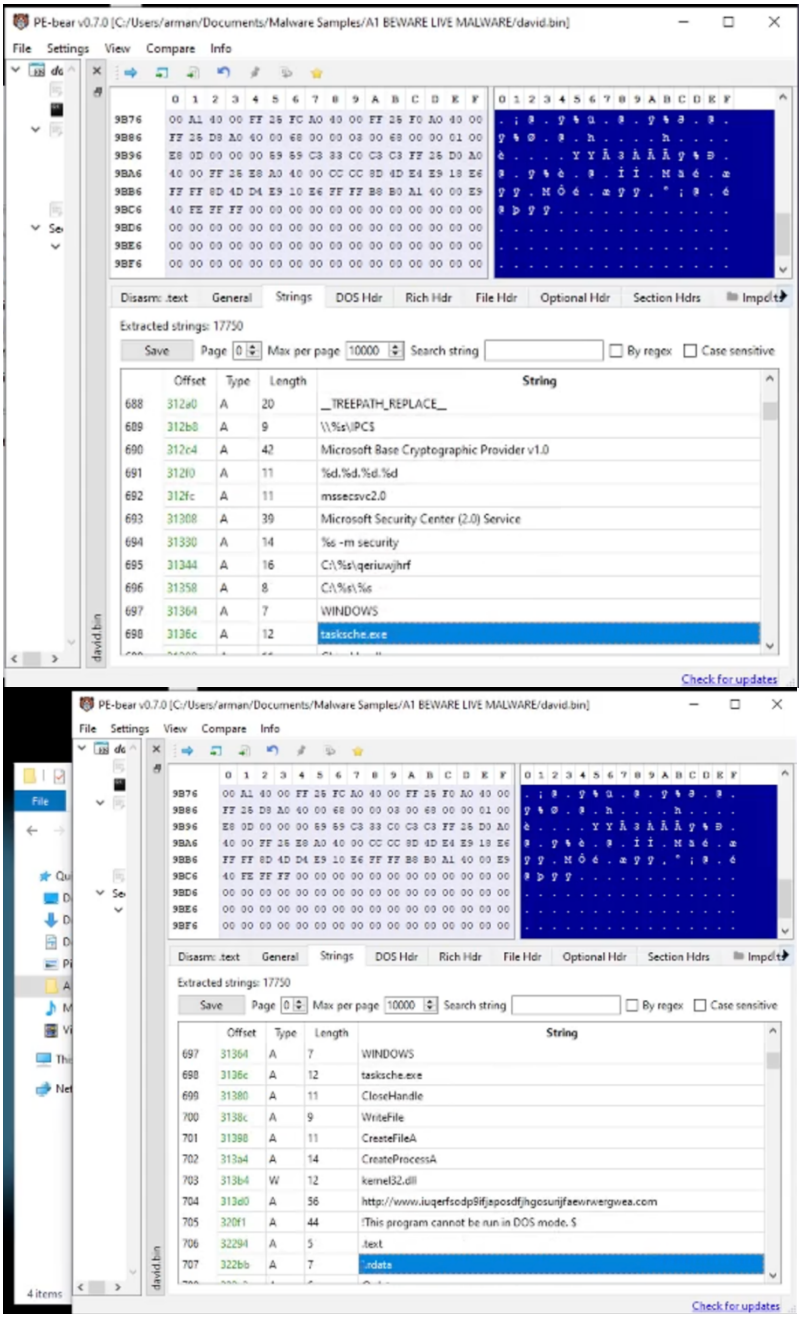
Can identify functions ReadFile, GetFileSize, CreateFileA, and MoveFileExA indicating interaction with files, most likely to read, modify or move data. Presence of KERNEL32.dll reinforces this, as it's commonly used for system operations like file and memory management. Additionally, CryptAcquireContextA and CryptGenRandom, which are part of advapi32.dll, shows the malware is performing encryption.



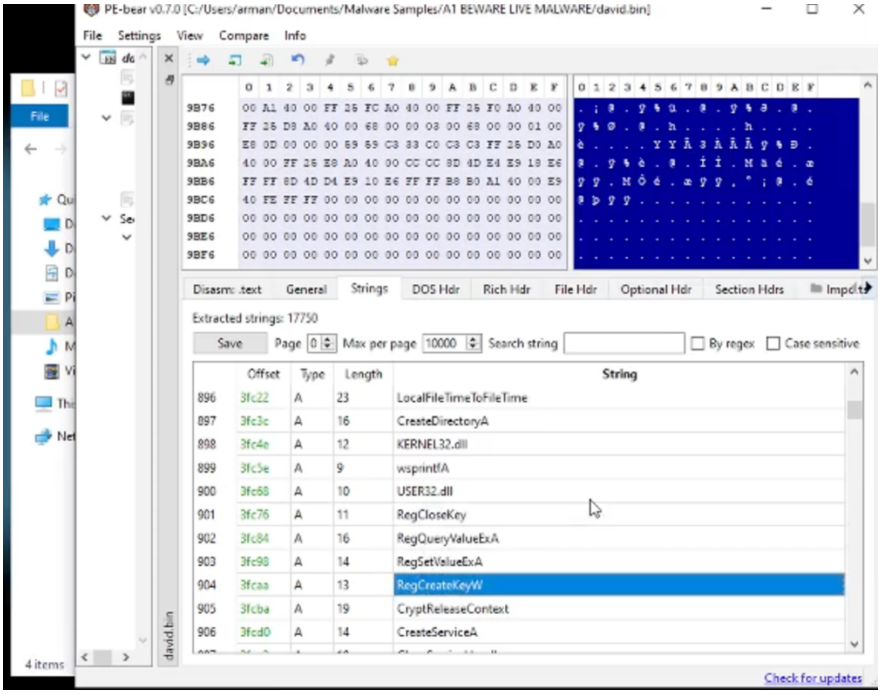
Can see network-based indicators; GetPerAdapterInfo, GetAdaptersInfo, InternetOpenA, InternetOpenUrlA, and InternetCloseHandle. Used to retrieve network configuration details and establish HTTP connections.



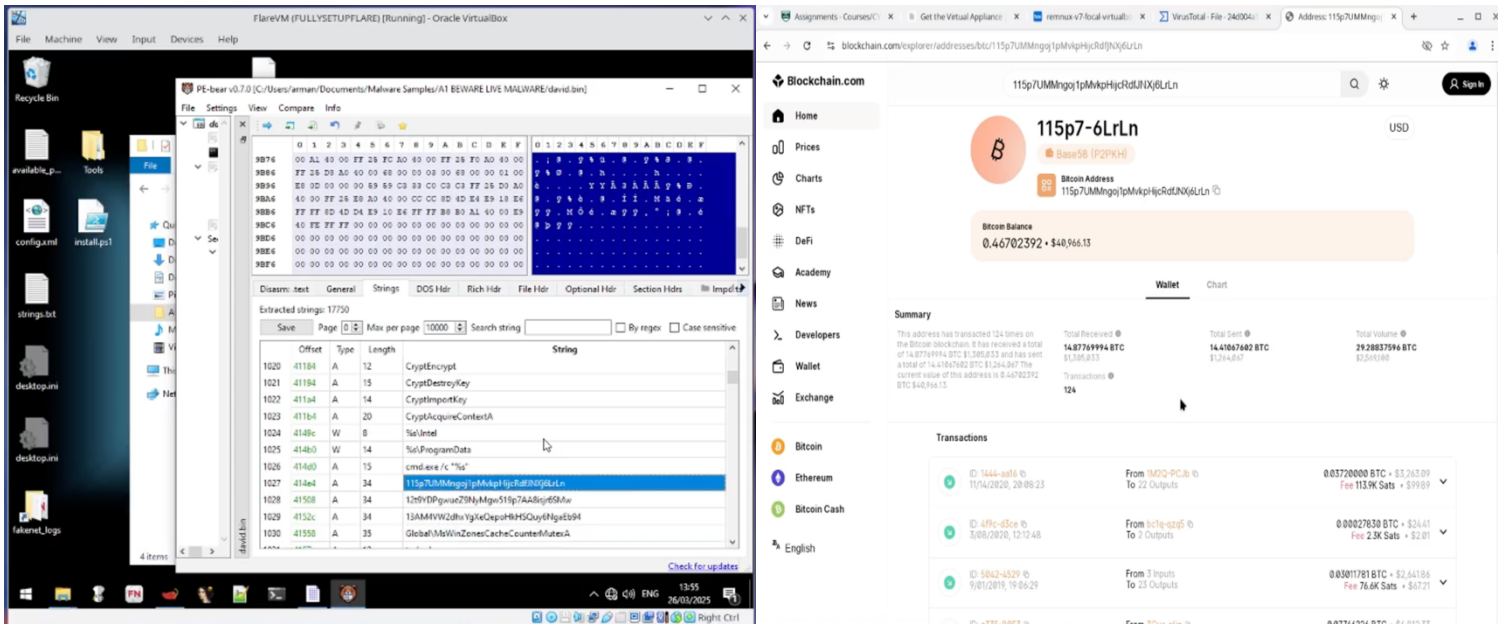
Here, we can see what appears to be payloads. Use of placeholders suggests they are used for injecting user or system values. Strings were too large for PE-bear to display so printed strings on UPX terminal. Can observe SMB3 at the end of the payload. SMB is used for sharing information over a network connection. Suggests the malware uses SMB-based exploits or features to move laterally within a network and deliver payloads to other vulnerable machines.



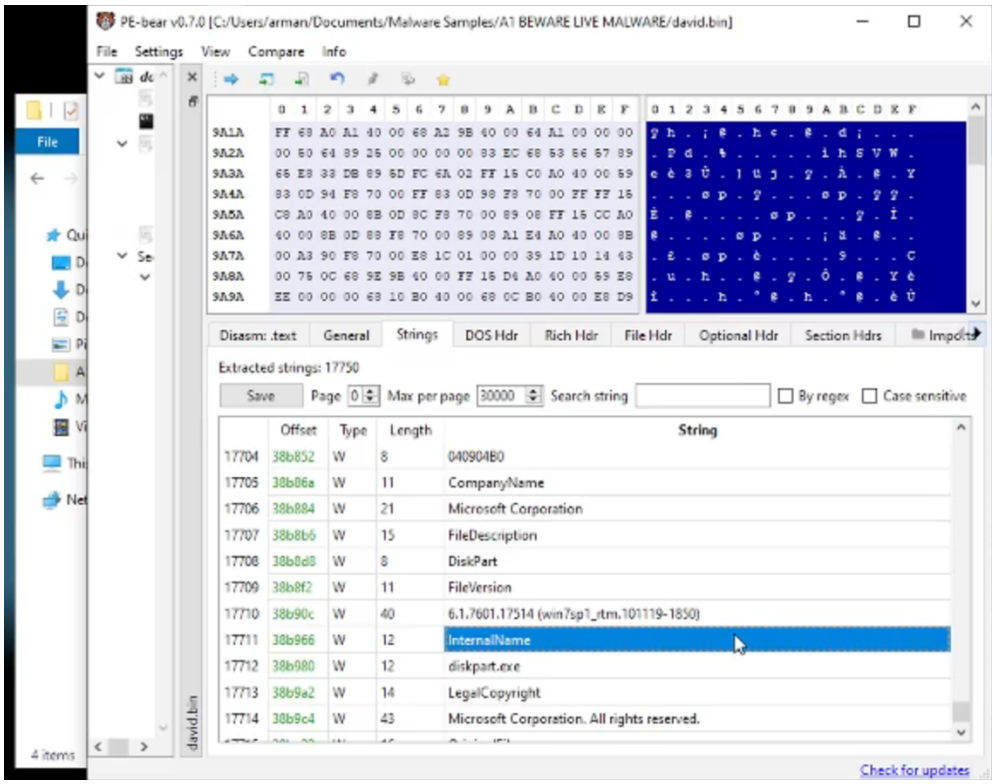
Can see a HTTP domain, which is a kill switch. When malware is executed, it attempts to request an HTTP to this URL. If connection is successful, it will drop/terminate execution, and if it fails, dropper will infect the system with ransomware. Malware is designed to run as a service with the parameters -m security.



Can see registry-related functions; RegCloseKey, RegQueryValueExA and RegCreateKeyW. Functions can enable malware to maintain persistency at startup through registry manipulation by creating/modifying keys so malware runs after system reboot.



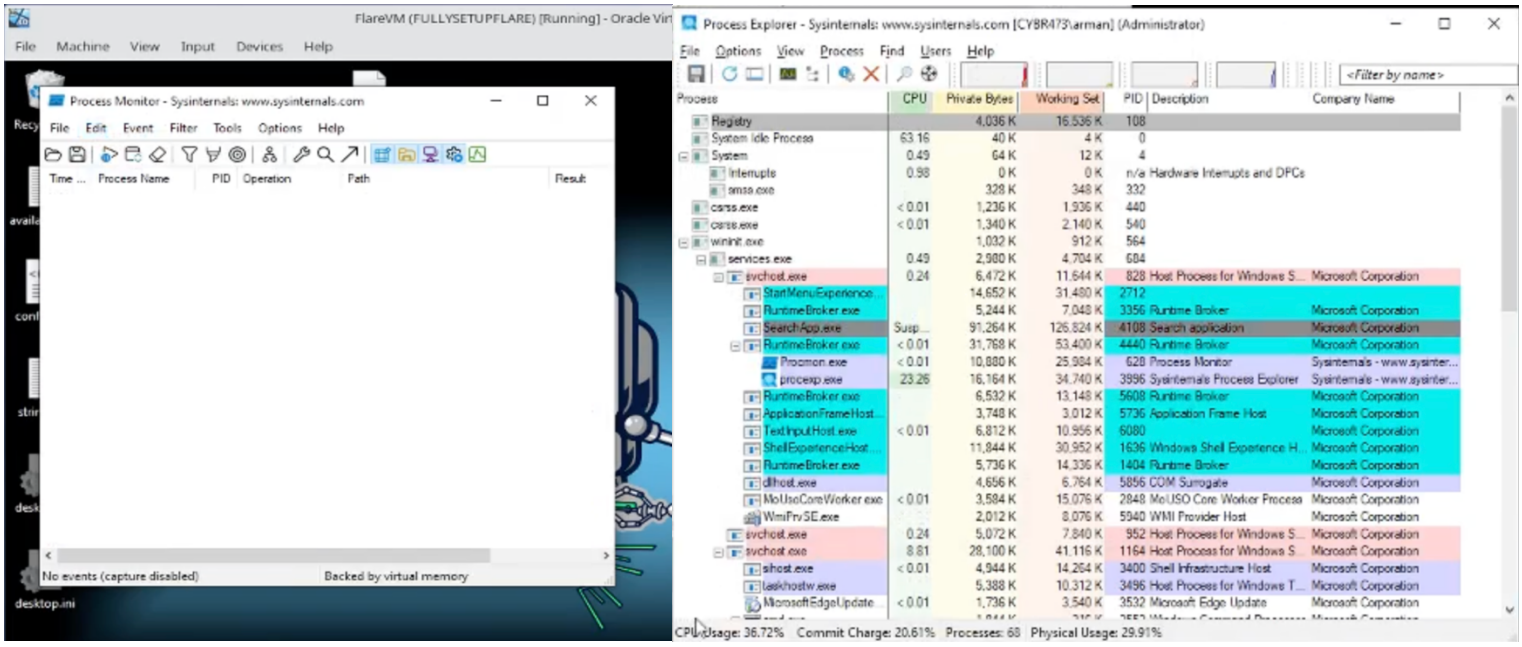
Can observe hardcoded Bitcoin wallet addresses. Can use the addresses to look up the actual Bitcoin wallets on www.blockchain.com.



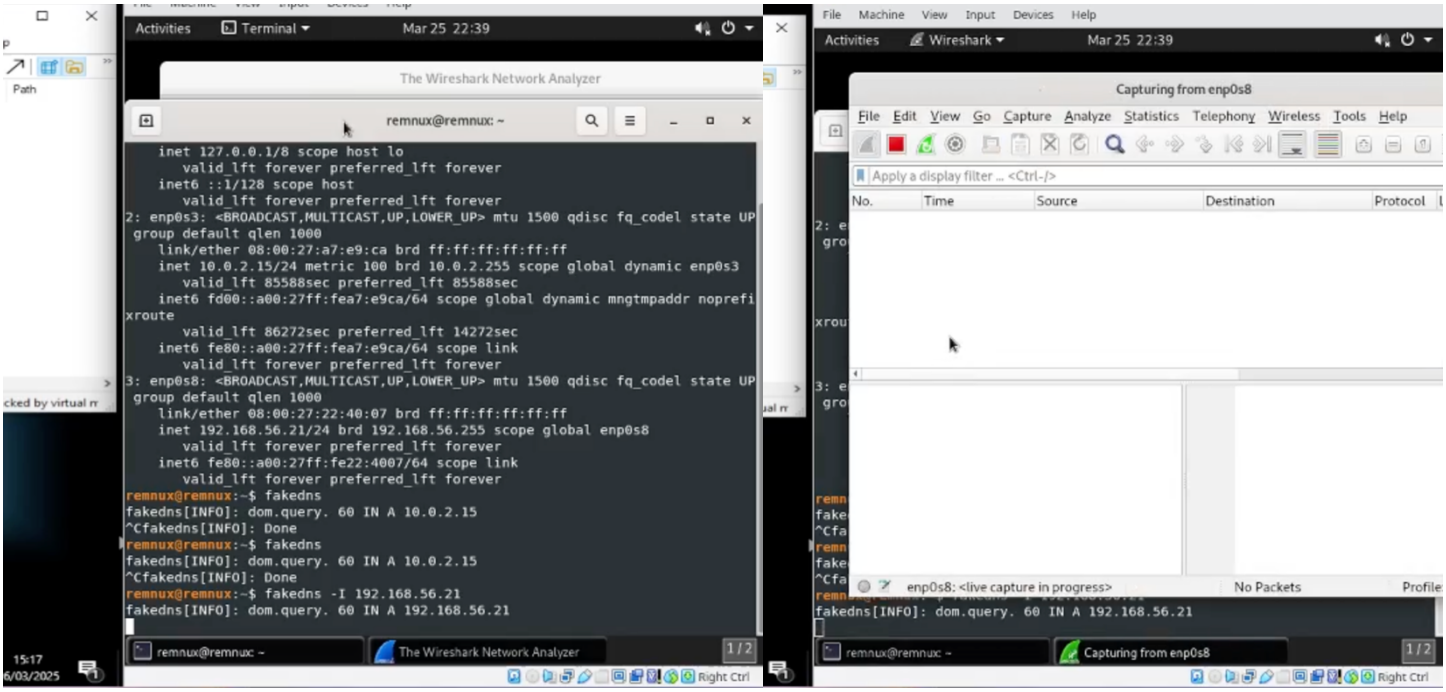
Shows strings DiskPart and diskpart.exe. Is used by the malware to disguise itself as a legitimate Windows System utility by adopting the internal name diskpart.exe. Allows it to evade detection and appear as a typical system process.

2.2. Basic Dynamic Analysis

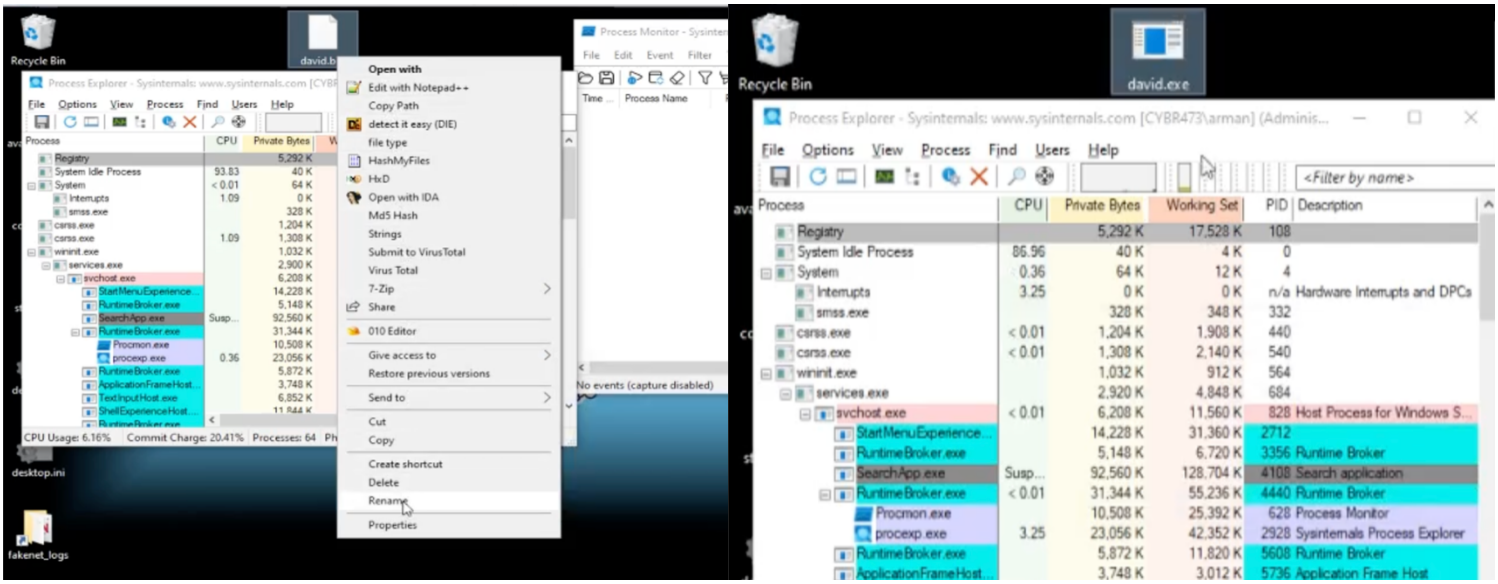
Setup and Running Malware



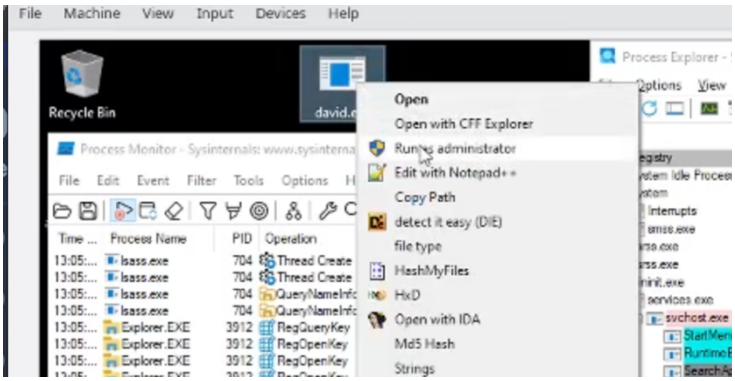
First launched Process Monitor and Process Explorer to monitor system processes in real-time.

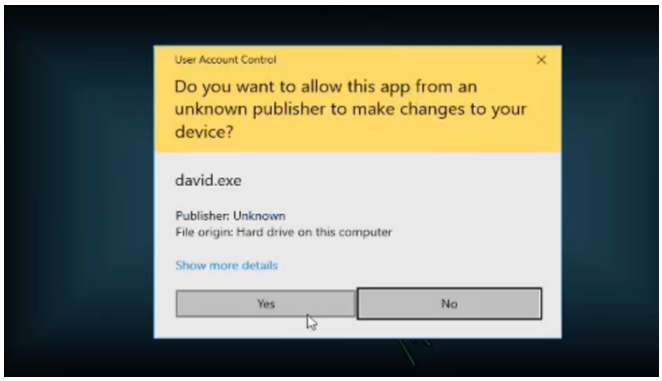


Setup a FakeDNS server to respond to DNS requests and Wireshark to listen to network traffic on enp0s8.

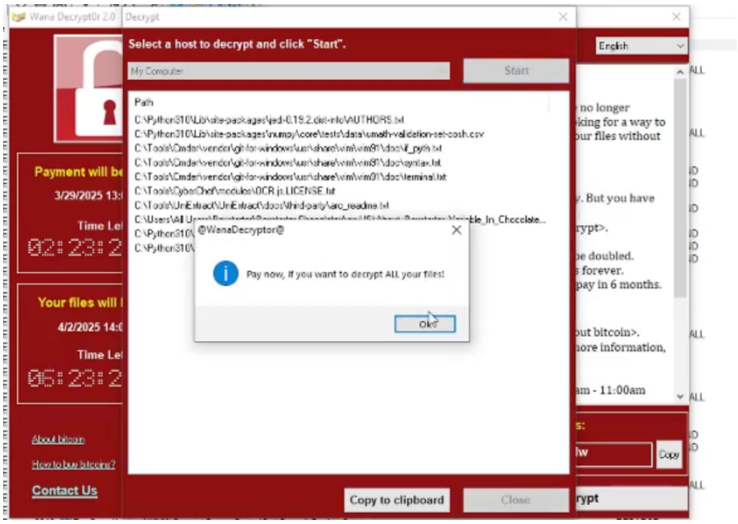


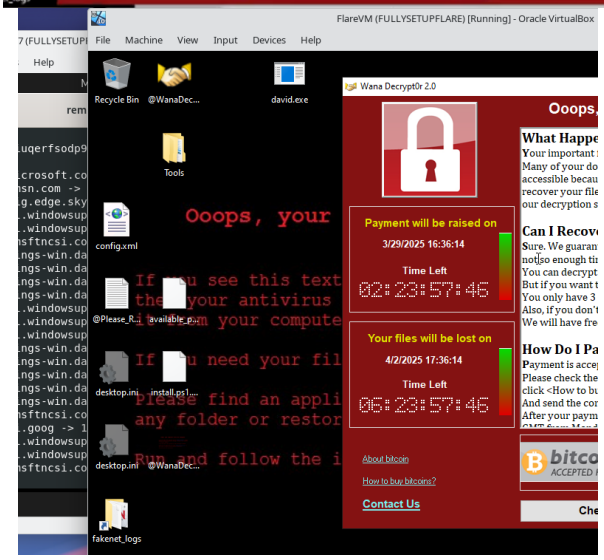
Changed david.bin to an executable.





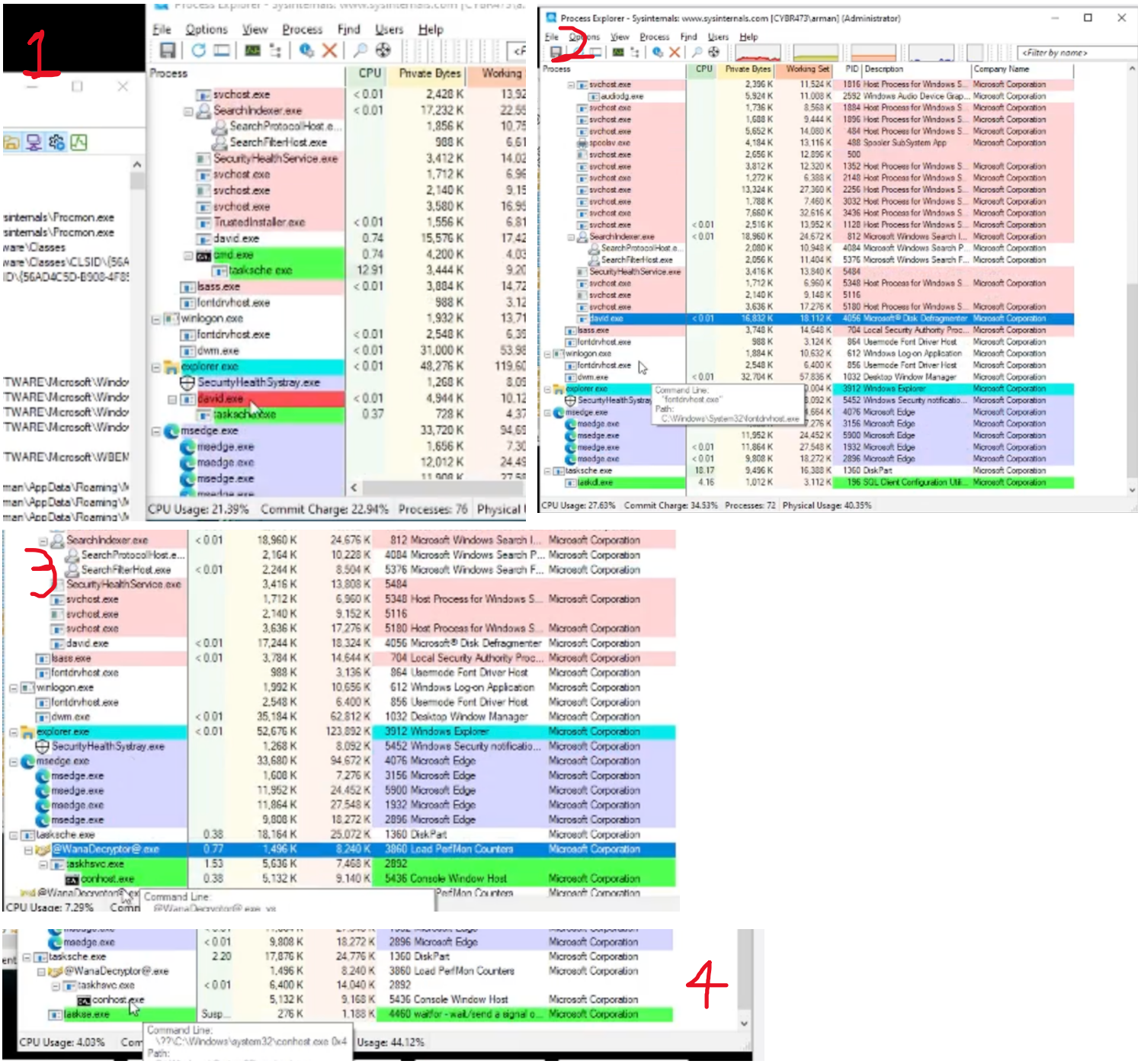
Executed the malware with administrative privileges.



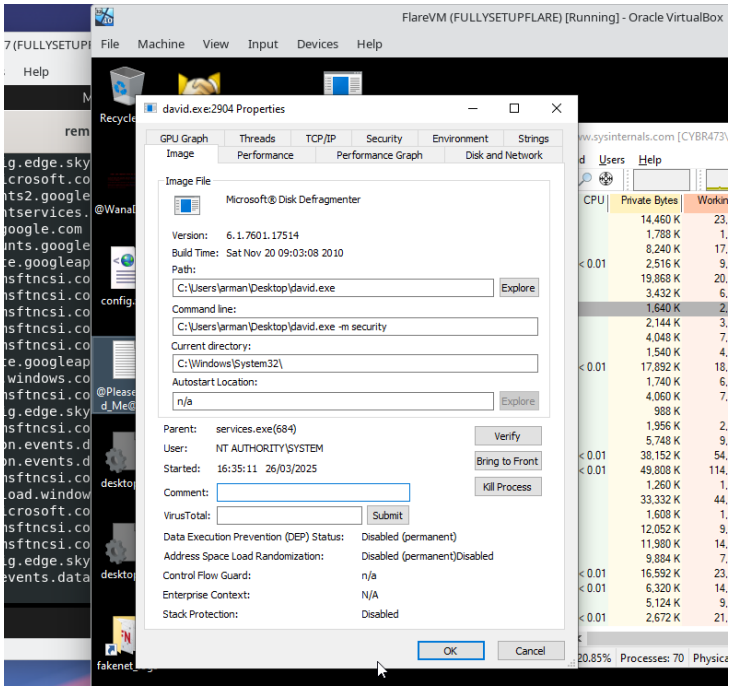


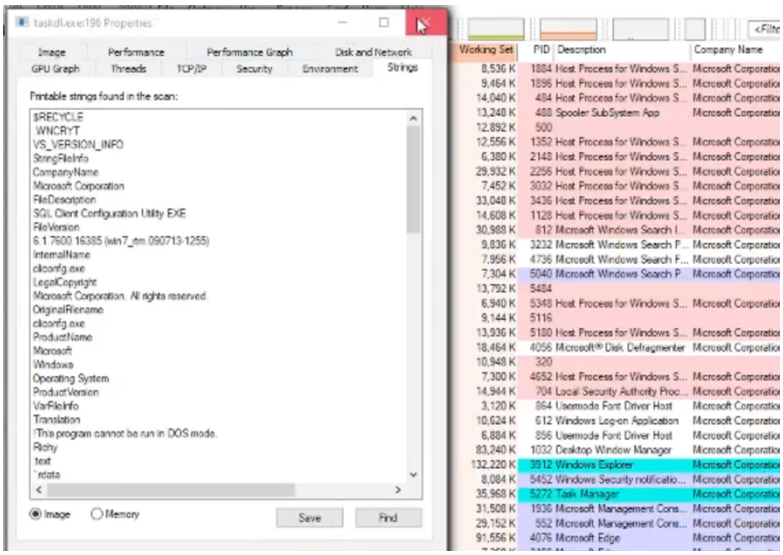
After running, we are greeted with a ransomnote indicating files are encrypted and a Bitcoin wallet address to pay. Has a decrypt button allowing partial file decryption as proof, but full decryption requires payment. Ransom-related files are on the desktop, and the desktop background has changed. This is an indicator malware has finished running.

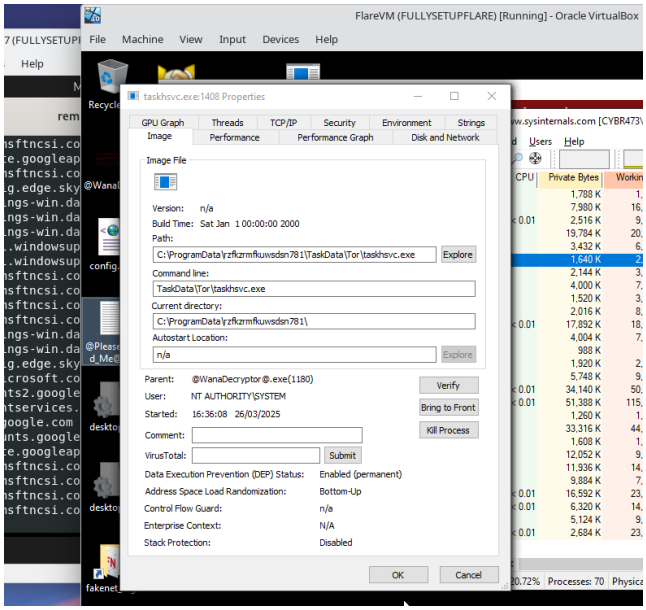
Process Explorer

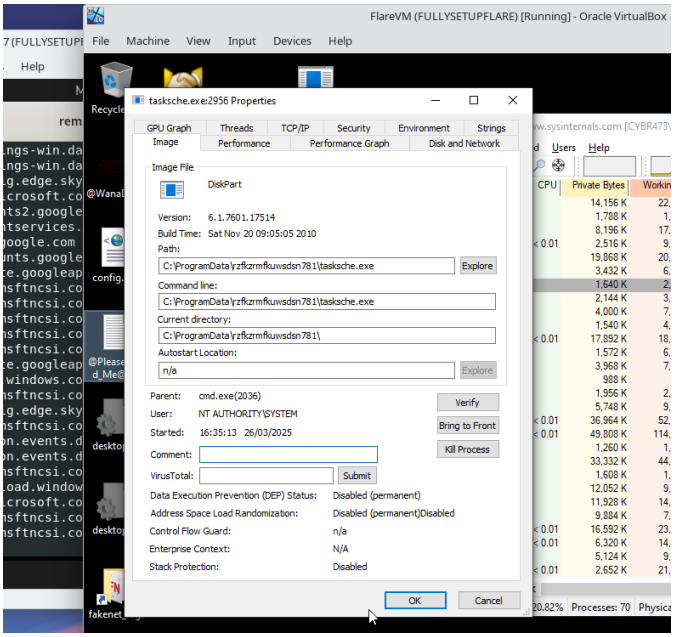


During malware run, tasksche.exe is spawned as a child process of david.exe. Shortly after, taskdl.exe appears as a child process of tasksche.exe but disappears within a few seconds. See the creation of @WanaDecryptor@.exe, which is the ransomware interface, along with taskhsvc.exe and child process conhost.exe.



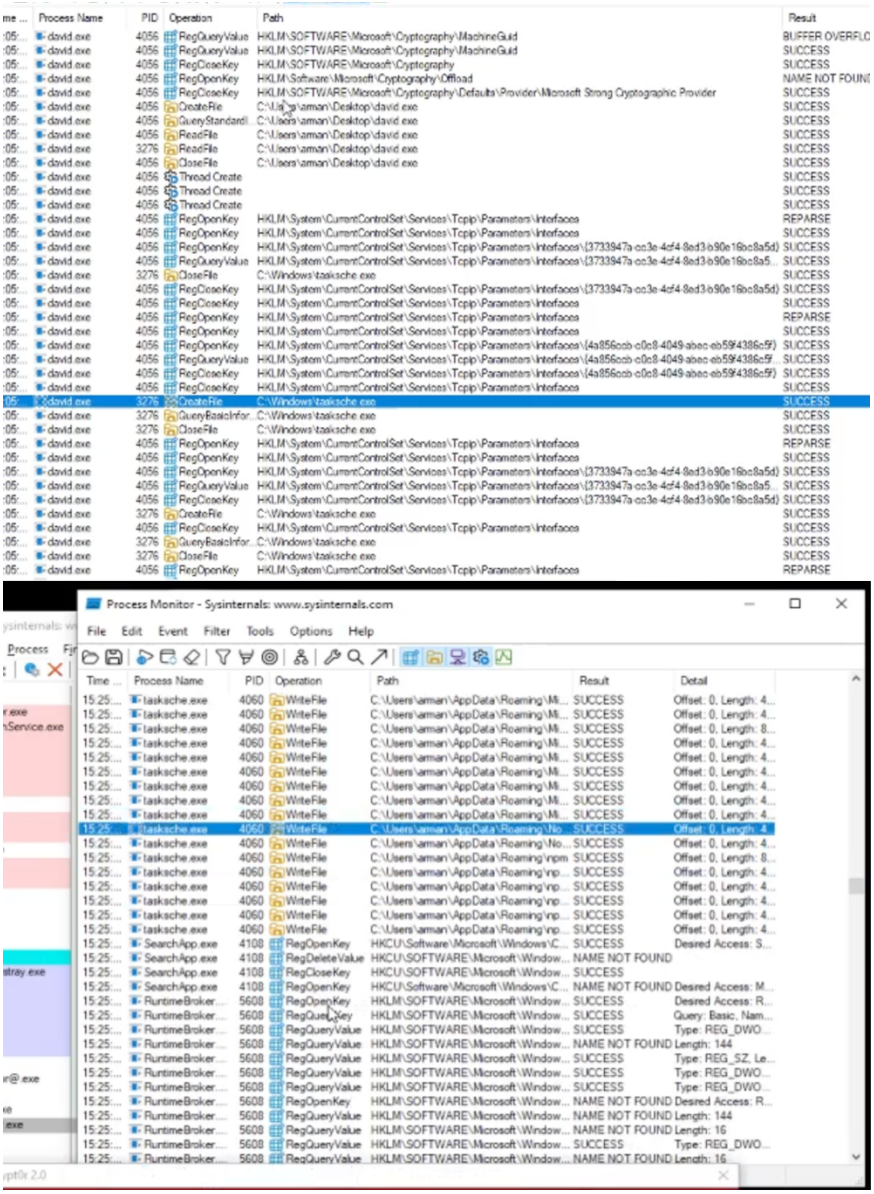




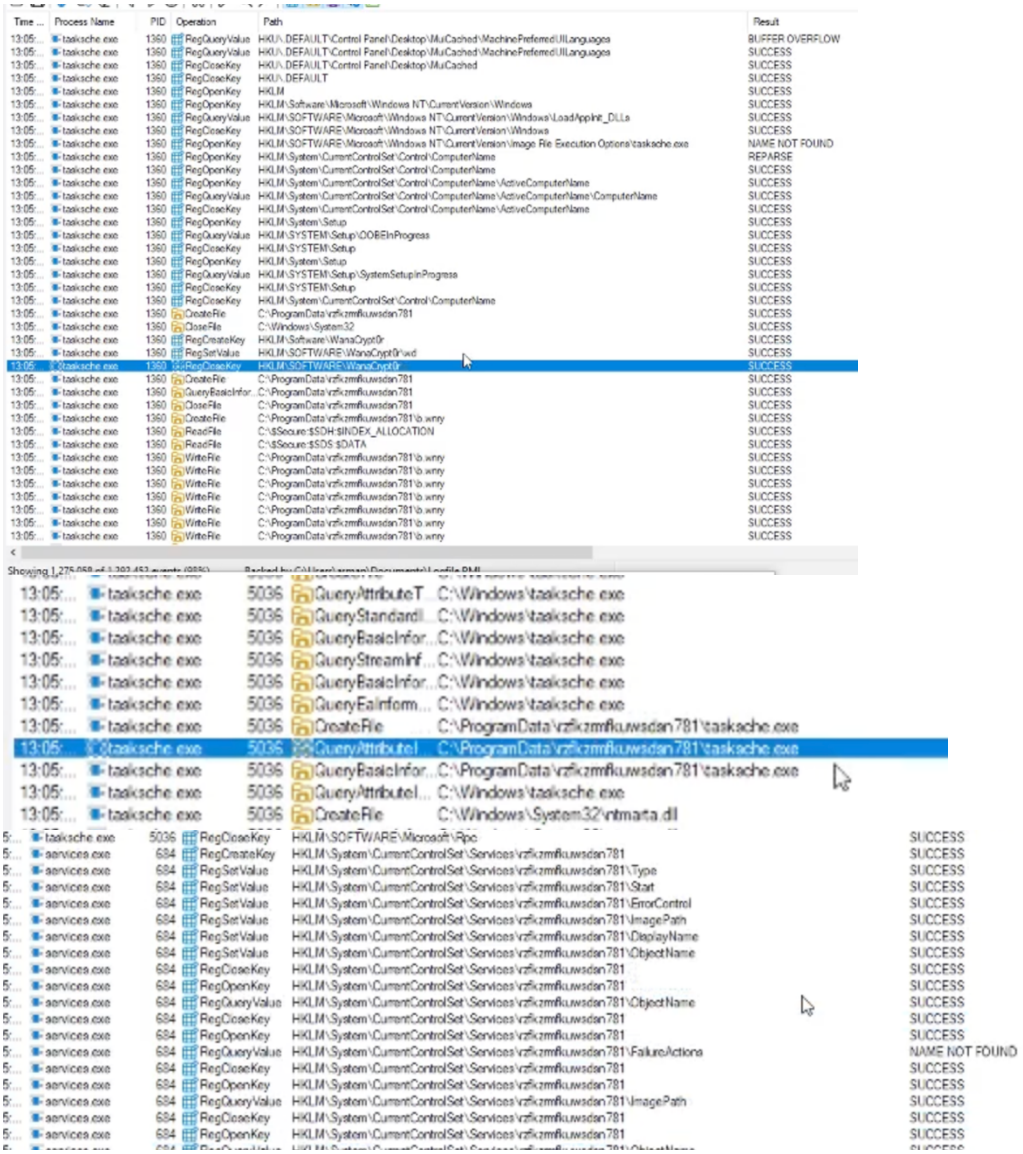


Have captured executables; tasksche.exe, taskhsvc.exe, taskdl.exe and David.exe. I can see the parameter -m security on david.exe confirming malware attempting to make an HTTP request to kill-switch domain. Observed tasksche.exe has the name DiskPart. Another interesting thing is the command line that the executables are run from, rzfkzrmfkuwsdsn781, which appears not to represent an actual/typical ProgramData name.

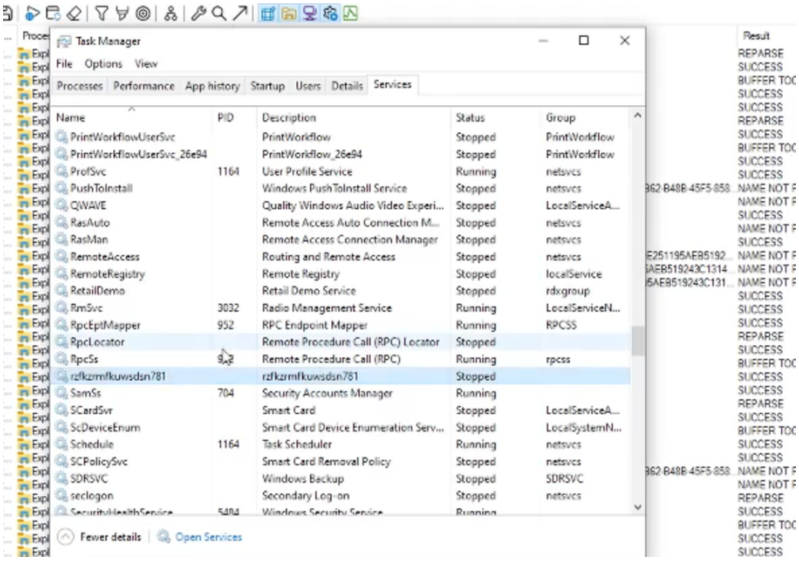
Process Monitor



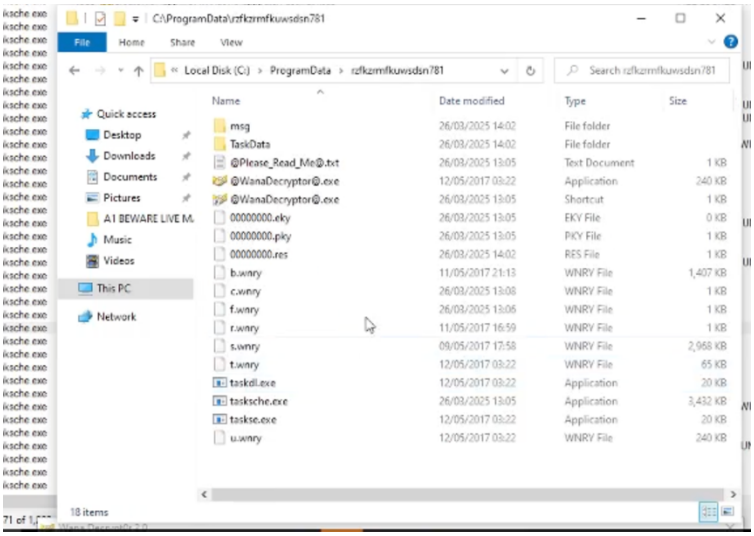
Can see David.exe interacting with tasksche.exe, creating the executable and modifying registry paths, embedding itself into the system. Can also see tasksche.exe performing key operations such as WriteFile, RegOpenKey, and RegDeleteValue, showing this executable writing files and modifying registry keys to change system configurations, either as payload deployment or a persistence mechanism.



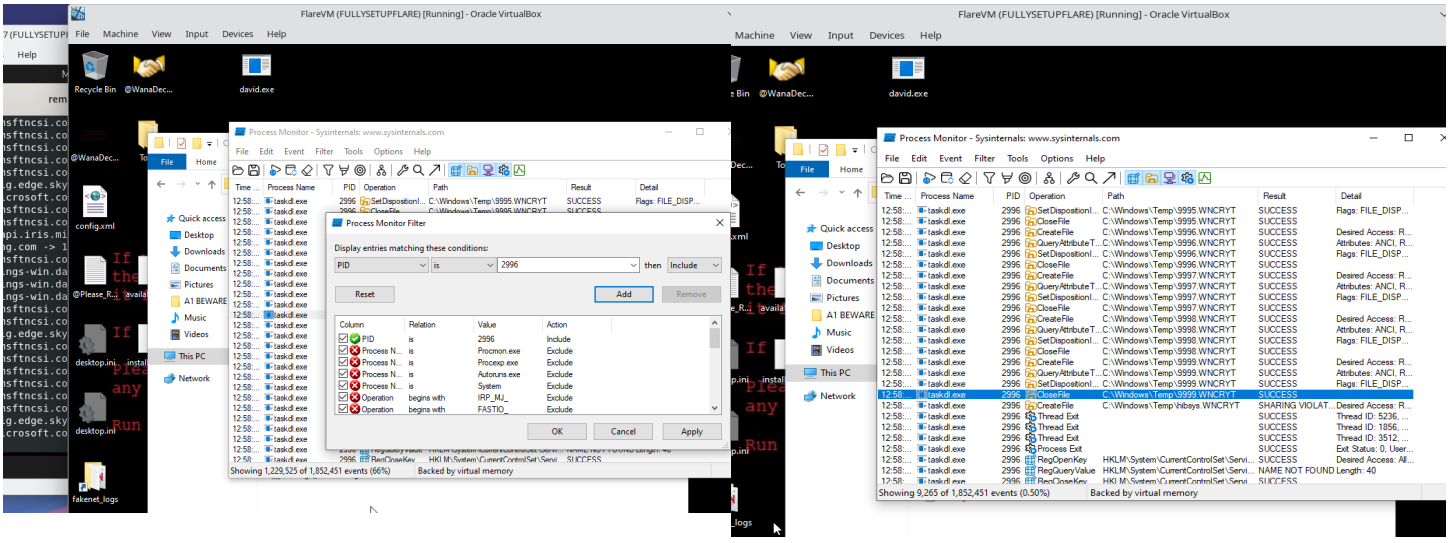
Can see registry rzfkzrmfkuwsdsn781 being created, which is the registry we saw tasksche.exe being run under. Service entry registry key configurations such as Type, Start, ErrorControl and ImagePath can be seen. Can also see creation of tasksche.exe under the same directory.



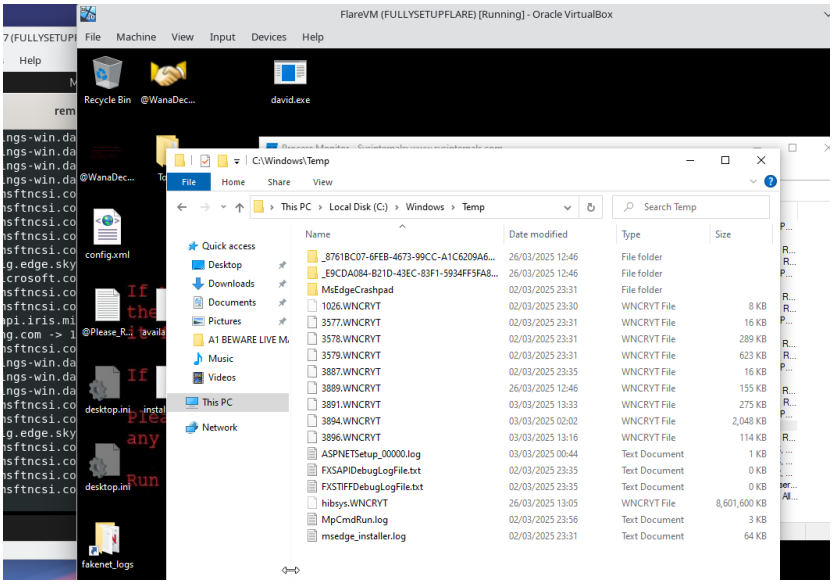
I can see rzfkzrmfkuwsdsn781 as a registered service. It shows this is one of the persistent services the malware created. Although marked "Stopped" this still validates it was registered on the system and capable of being executed on startup.



Can identify rzfkzrmfkuwsdsn781 directory in File Explorer. Directory contains the .exe that was previously analyzed as part of the process (i.e. taskdl.exe, tasksche.exe). Can see PLEASE_READ_ME.txt file and the WannaDecryptor.exe, as well as .eky, .pky, and .res files, which are likely encryption/decryption-related files for ransomware. TaskData directory also contains a Tor folder, suggesting malware may have intended to communicate via Tor network for anonymity.

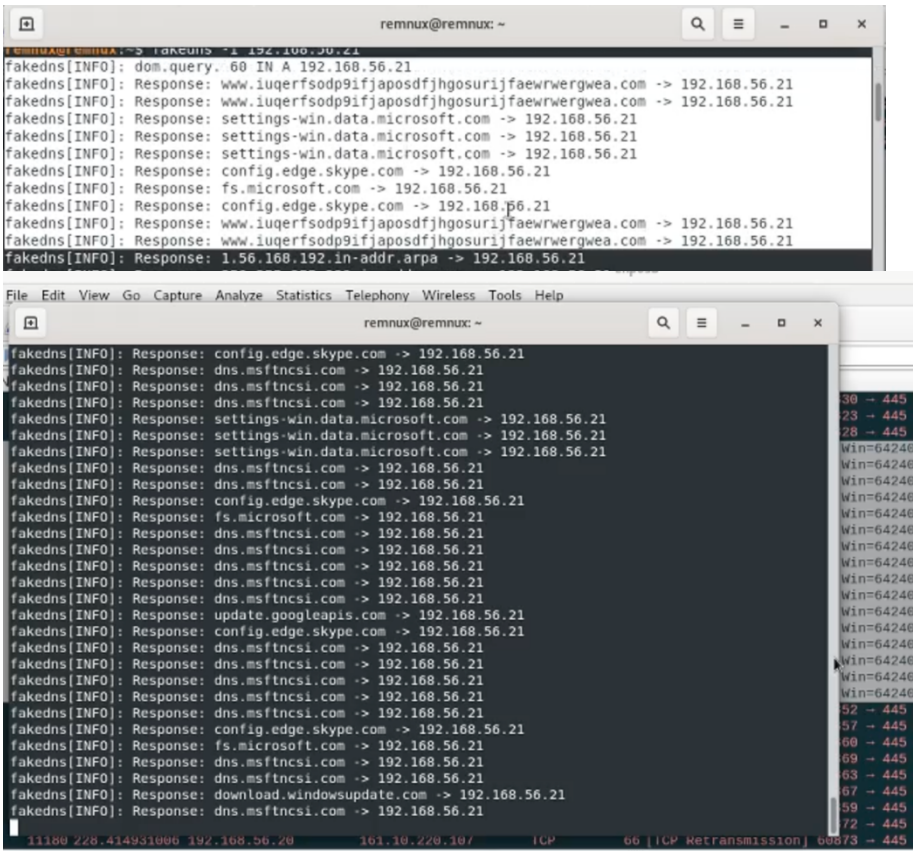


Applied PID filter of taskdl.exe to analyze and observe the process encrypting files and locating them in the Windows\Temp directory appending a .WNCRYT extension.



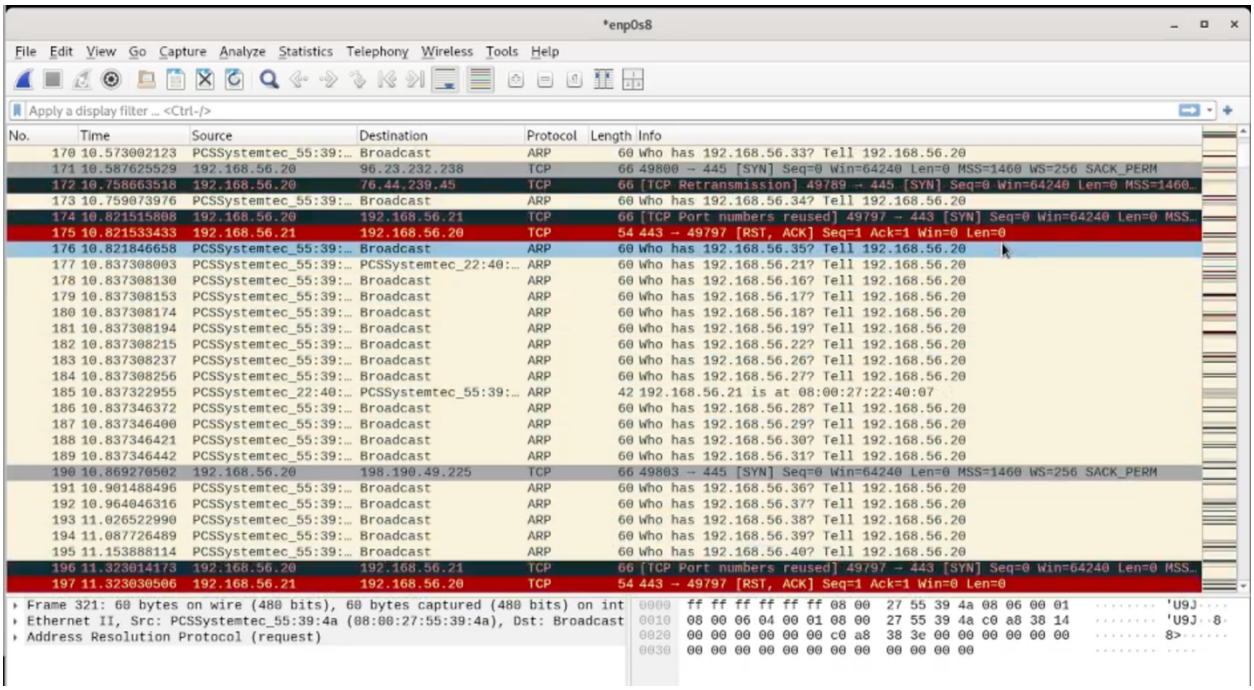
Can observe .WNCRYT files in identified folder using File Explorer.

FakeDNS

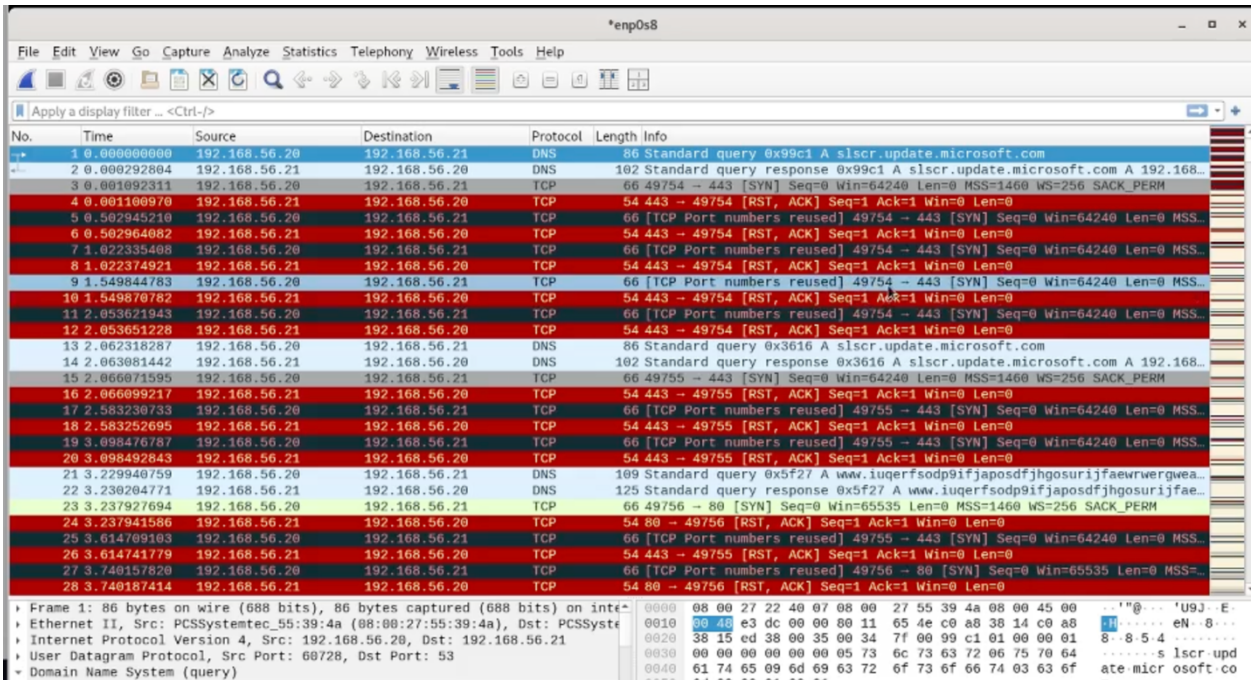


Can observe the first domain the malware attempts to connect with is the kill switch domain identified during string analysis. FakeDNS responded to this, simulating a successful connection, however, malware still executes, indicating connection may not have been treated as valid by the malware. Malware then continues querying other domains.

Wireshark



Large number of ARP packets being broadcasted, attempting to discover other devices on the subnet. This visual indicates the malware spreading mechanism, scanning 192.168.0-255 IP range to see other potential victims.

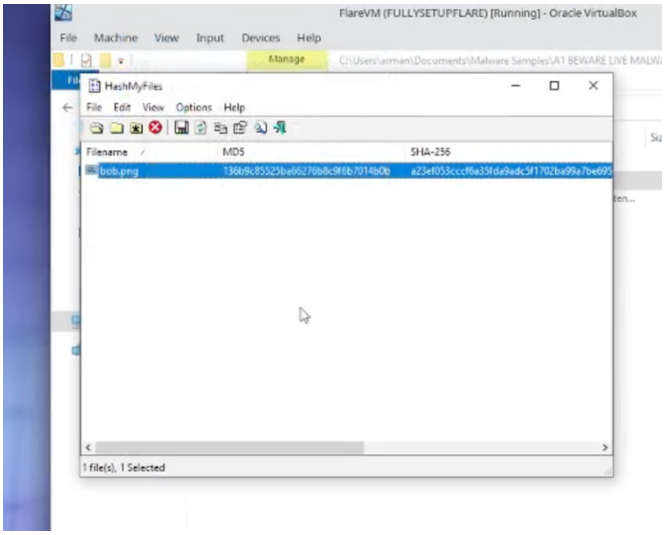


I can see DNS queries being made to the kill switch domain. Although it simulates a DNS response, the malware still does not receive a proper HTTP response from the server since we are not hosting the actual domain, resulting in it believing the kill switch is unreachable.

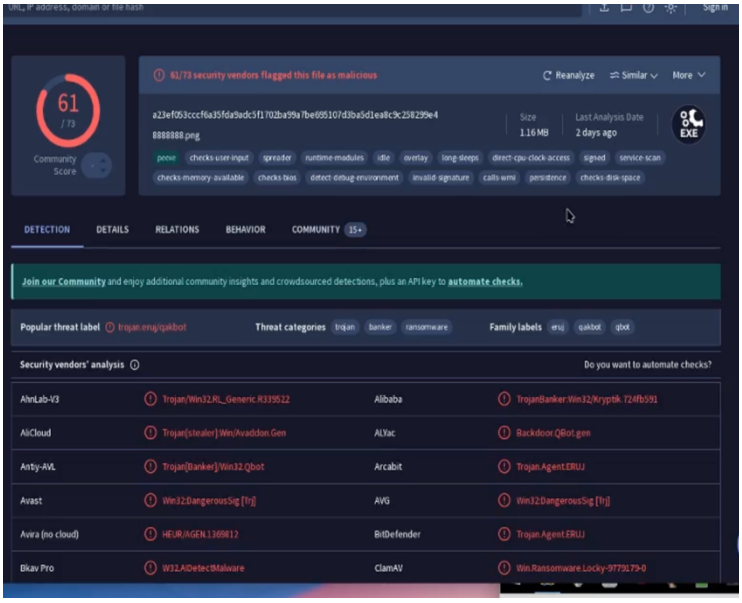
3. Bob

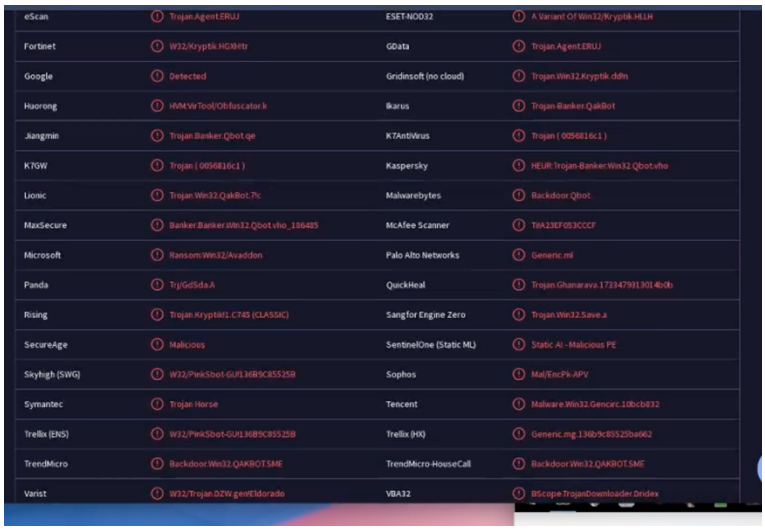
3.1. Basic Static Analysis

VirusTotal

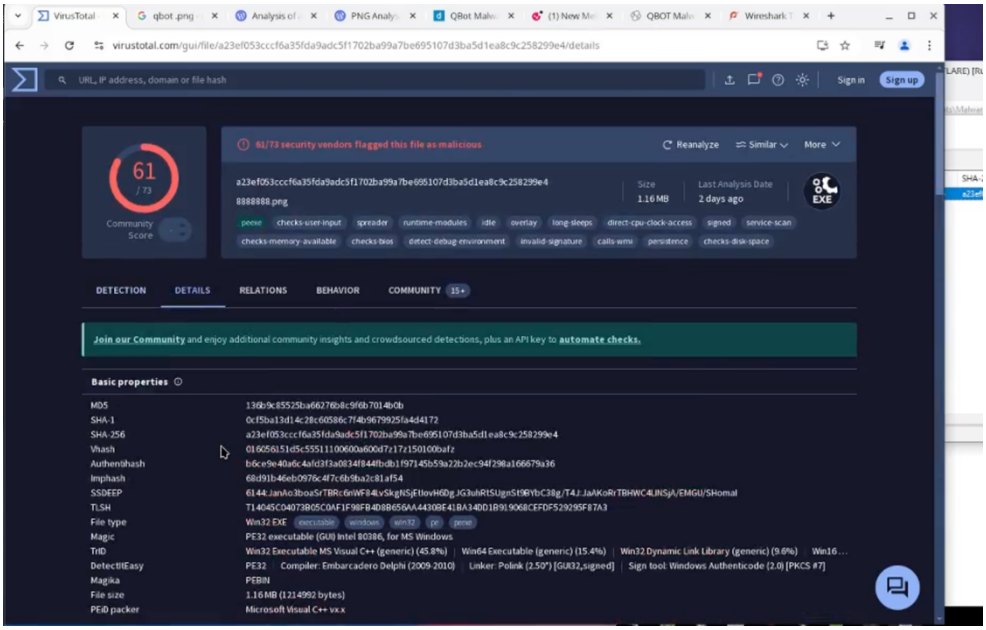


I began investigating acquiring MD5 hash of files to search within VirusTotal. This would provide information on the malware if it had been previously analyzed and flagged by antivirus engines.

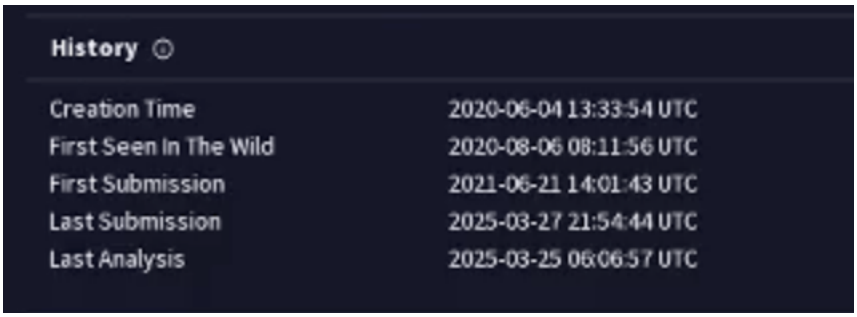




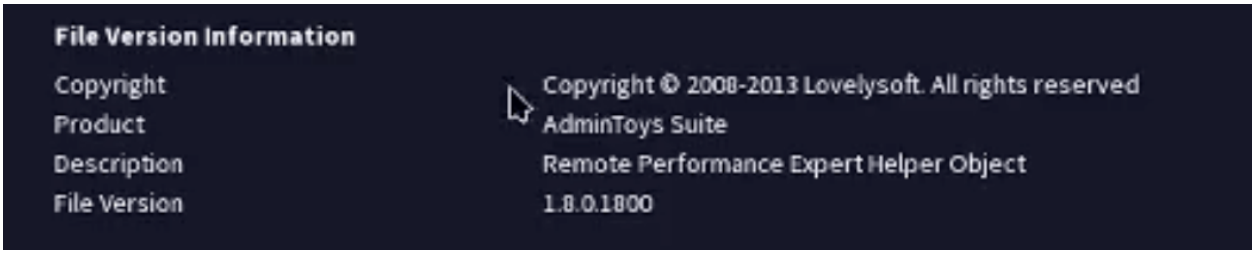
61 out of 73 vendors identified the malware as malicious. Identified by antivirus engines under Backdoor.QBot.gen and Trojan.Agent.ERUJ. Can determine that bob.png is related to Qbot ransomware, which steals banking data (e.g. credentials, session information). Did not manually compare AV signatures with ClamAV, however, VirusTotal includes ClamAV engine, which confirms detection, Win.Ransomware.Locky-9779179-0.



Compiler of the malware is Embarcadero Delphi. However, was flagged being compiled with Microsoft Visual C++. Might suggest multiple components or packing. PEiD packer states using a Microsoft Visual C++ vx.x packer. Can observe that it's PE32 architecture. Can also see VirusTotal tags, persistence, checks-debug-environment and service scan, suggesting it employs anti-analysis, persistence and network scanning techniques.



Can see creation time was 2020-06-04, which is likely when the binary was compiled for the first time.



Malware does file spoofing to appear as legitimate software. Claims to be part of the AdminToys Suite by Lovelysoft, describing itself as a "Remote Performance Helper Object".

Portable Executable Info

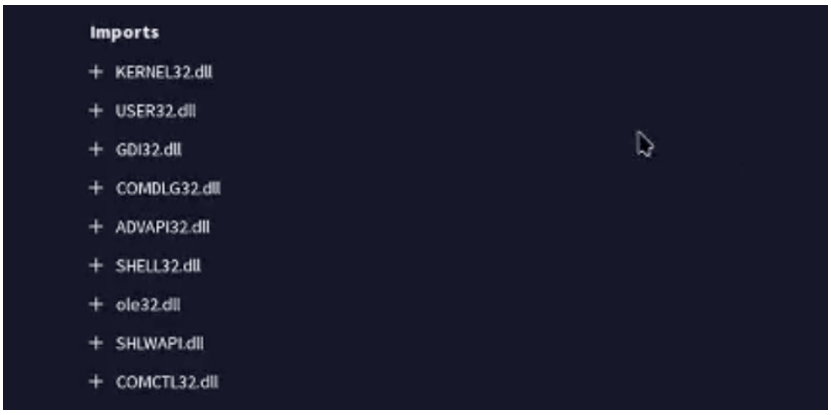
Header

Target Machine	Intel 386 or later processors and compatible processors
Compilation Timestamp	2020-06-04 13:33:54 UTC
Entry Point	10576
Contained Sections	5

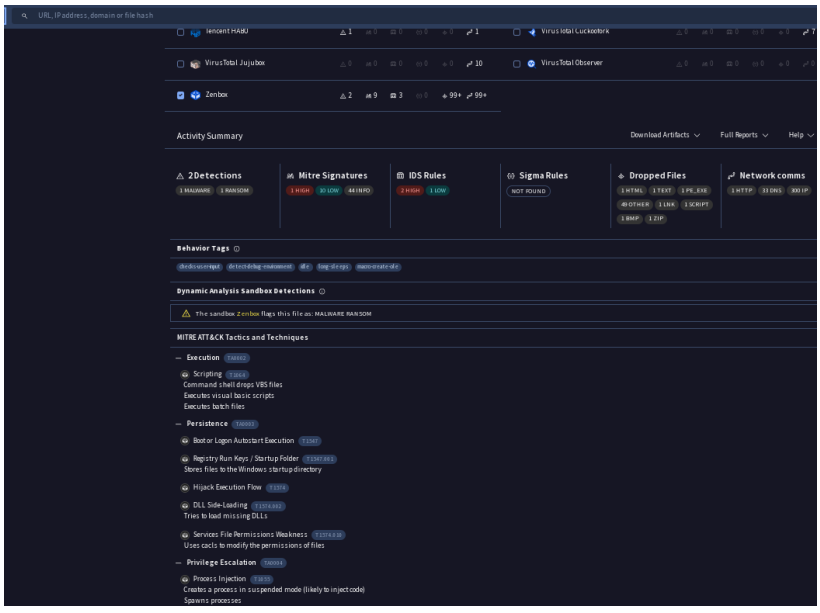
Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	1077644	1077760	2.96	cfb90e8a6aa6adb0067eacd6ec851015	142733296
.rdata	1085440	261	512	2.56	6394519a6326430bbdfdcc4ed9a31b43	71031
.data	1089536	20868	20992	5.67	dd275be55c007ec8311be1360f7f1e56	311888.19
r2	1114112	43415	43520	5.16	f4e0a11b39757039da7b4a8cef59453f	2422658.25
.rsrc	1159168	69636	70144	5.77	afaf61c81fed312ab375d6904ce42032	1854768.62

PE header indicates the malware targets Intel386 or later processes. Has five contained sections: .text, .rdata, .data, .r2, and .rsrc. Indicates entropy of 2.96 and 2.56 for .text and .rdata respectively. This is relatively low and doesn't indicate packing. The entropy of 5.67, 5.16 and 5.77 for .data, .r2 and .rsrc, respectively, indicates a possible use of obfuscation or packing. .r2 refers to the Rich Header, which contains information about the build environment used to compile/link the file.

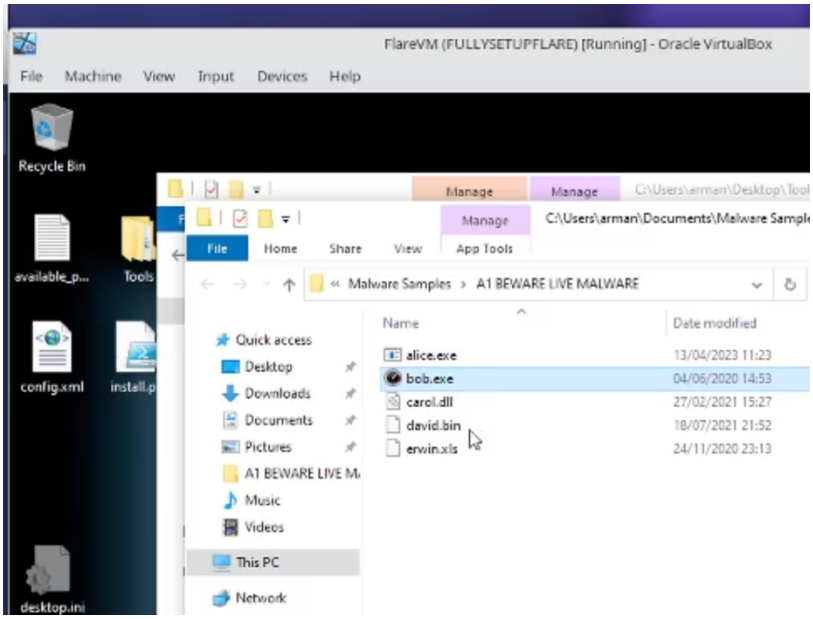


Can see the DLLs that the malware uses. SHELL32.dll and SHLWAPI.dll imply malware interacting with Windows Shell to launch executables and manage files for executing payloads. ole32.dll is used for object linking and embedding, which is used in malware to interact with documents/applications.

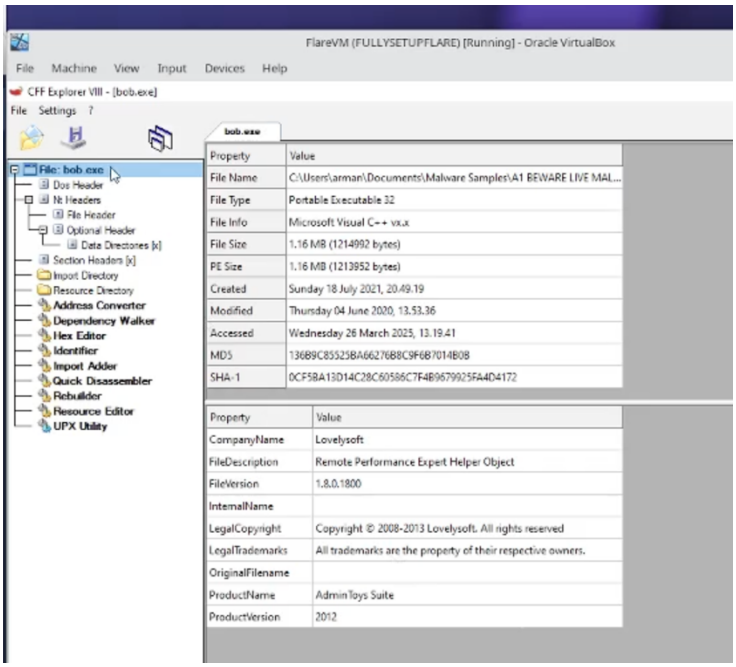


Analyzed the Sandbox report done by Zenbox on VirusTotal. Stated that execution is performed by command shell drops and executes scripts and batch files. It has persistency from startup, boot, and logon execution and registry keys stored in the Windows startup directory. It has also several defence evasion techniques through file manipulation, indicator removal and hiding artifacts and processes.

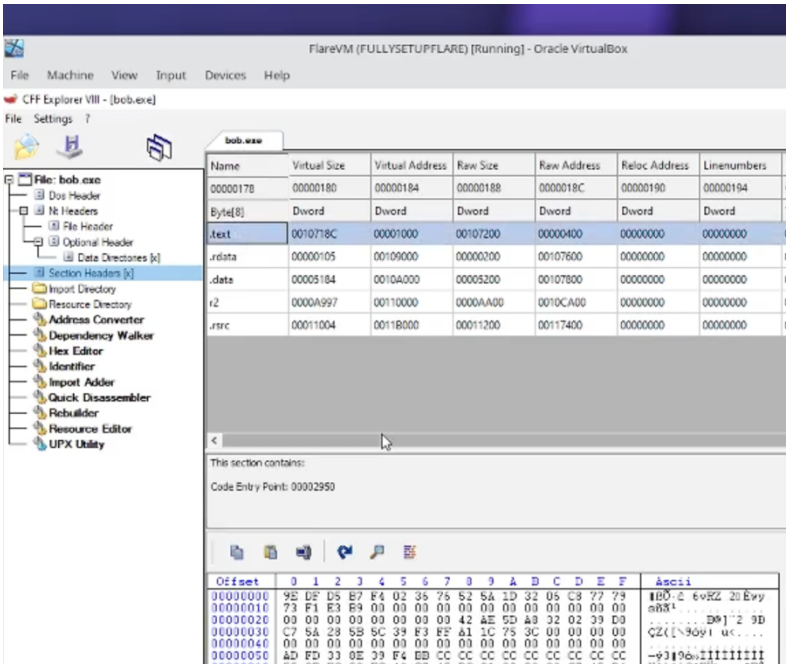
CFF Explorer



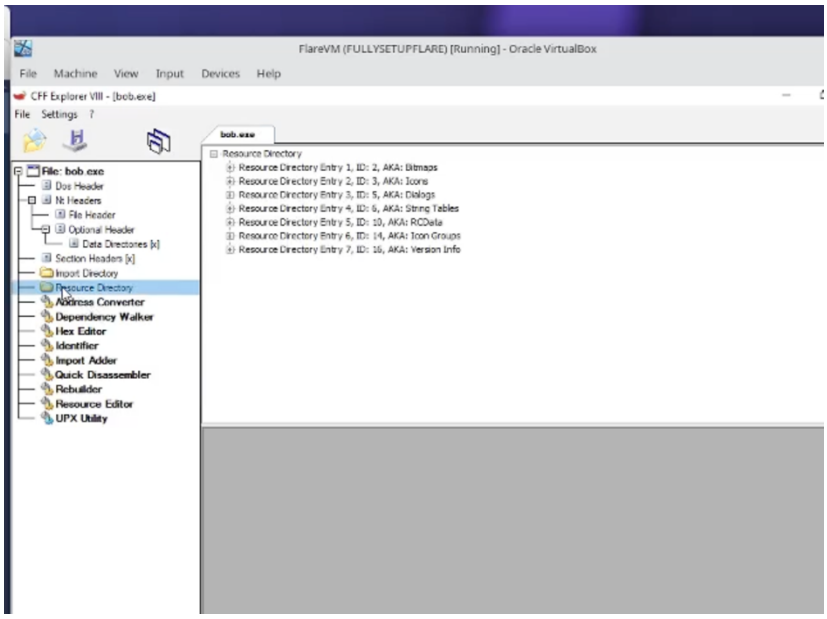
First, I changed the bob.png extension to bob.exe so we can analyze it using the tools we want to use to perform static analysis.



The malware is a 32-bit PE compiled using Microsoft Visual C++ vx.x. Can also see the creation date of July 18 2021, which does not match the creation time seen on VirusTotal.

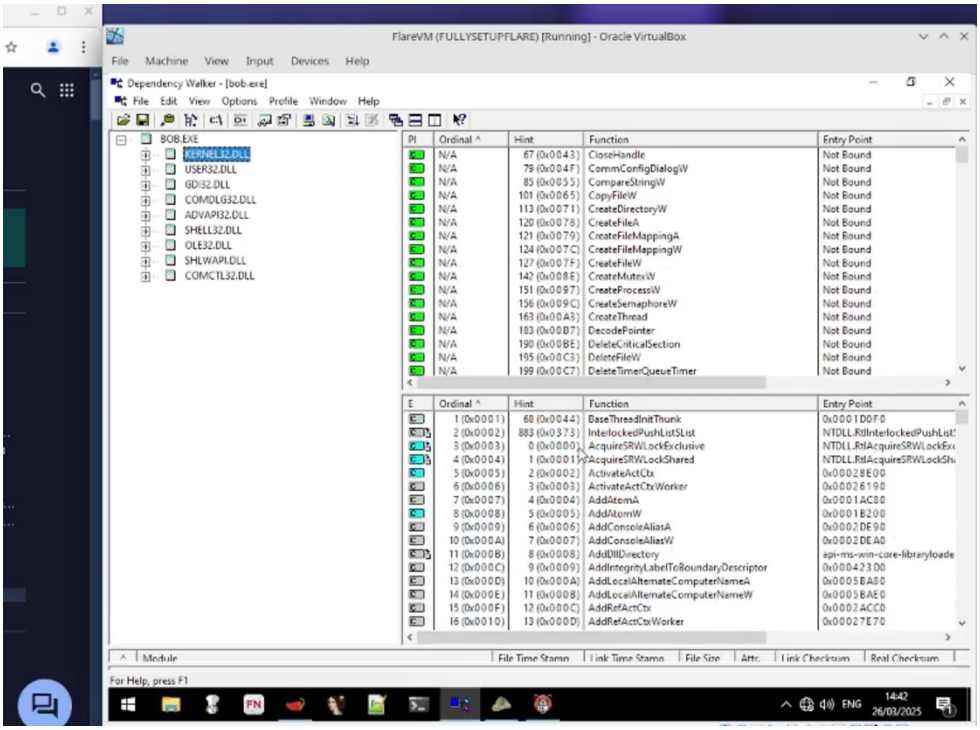


Analyzed PE Section headers. Clear that the malware is not packed with tools like UPX (e.g. UPX0, UPX1). Virtual and raw sizes are relatively similar, which would be unlikely if packed.



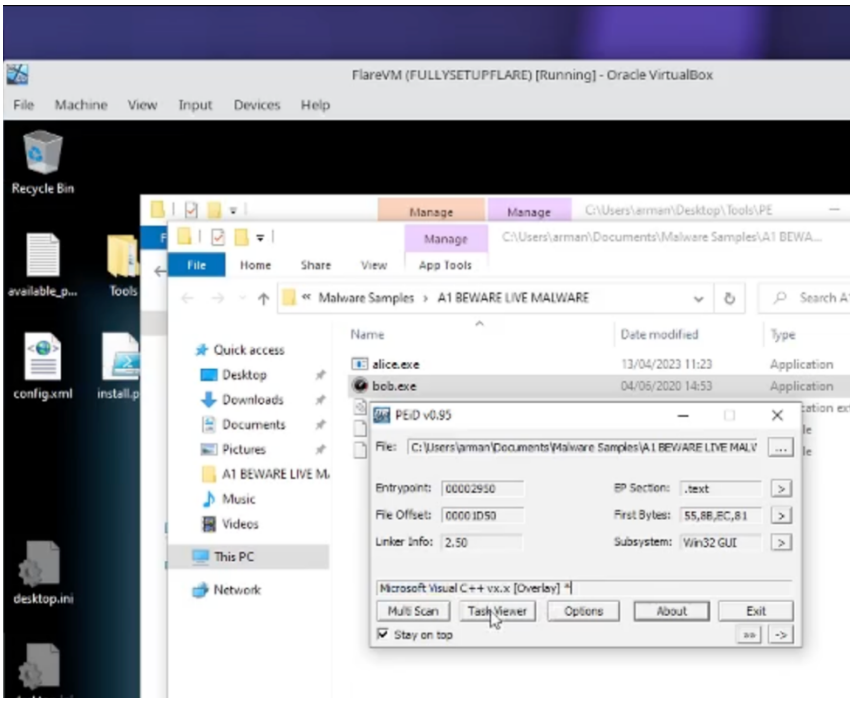
Next, I looked at the resource directory, which included typically GUI-related assets like Bitmaps, icons, dialogs, string tables, and version info.

Dependency Walker

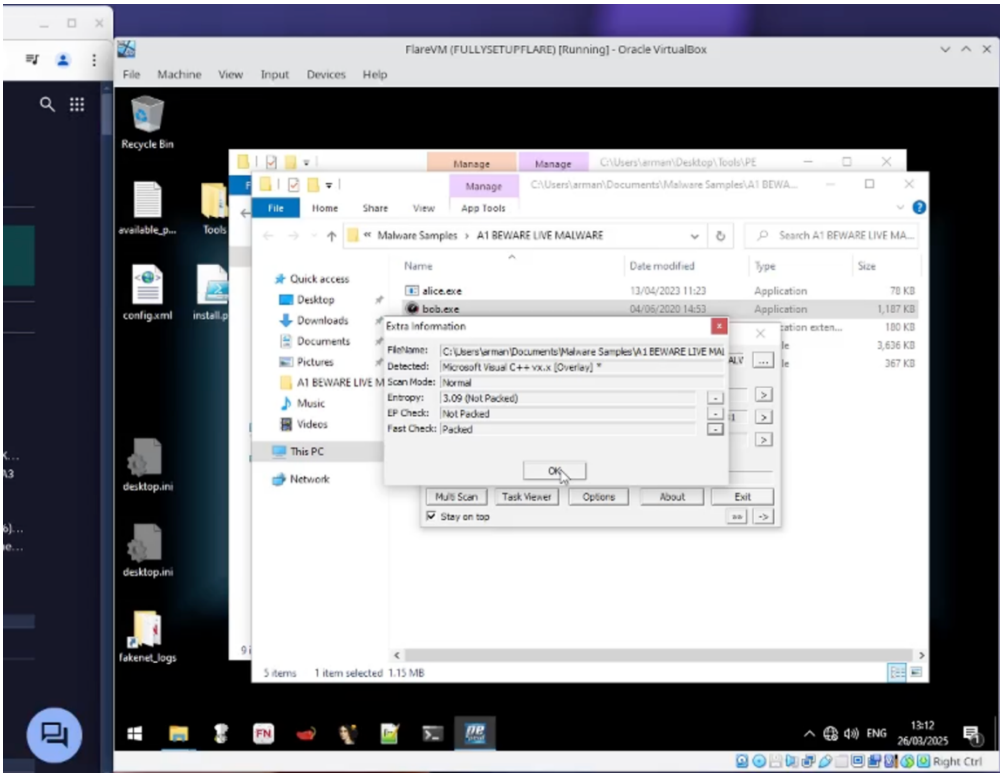


Upper pane displays functions from KERNEL32.dll where we can see dynamically loaded functions at runtime like CreateFileW, CreateThread, CreateProcessW, and DeleteFileW, suggesting it performs file manipulation, etc. From the extent of DLLs, I found it better to analyze them through string analysis.

PEiD

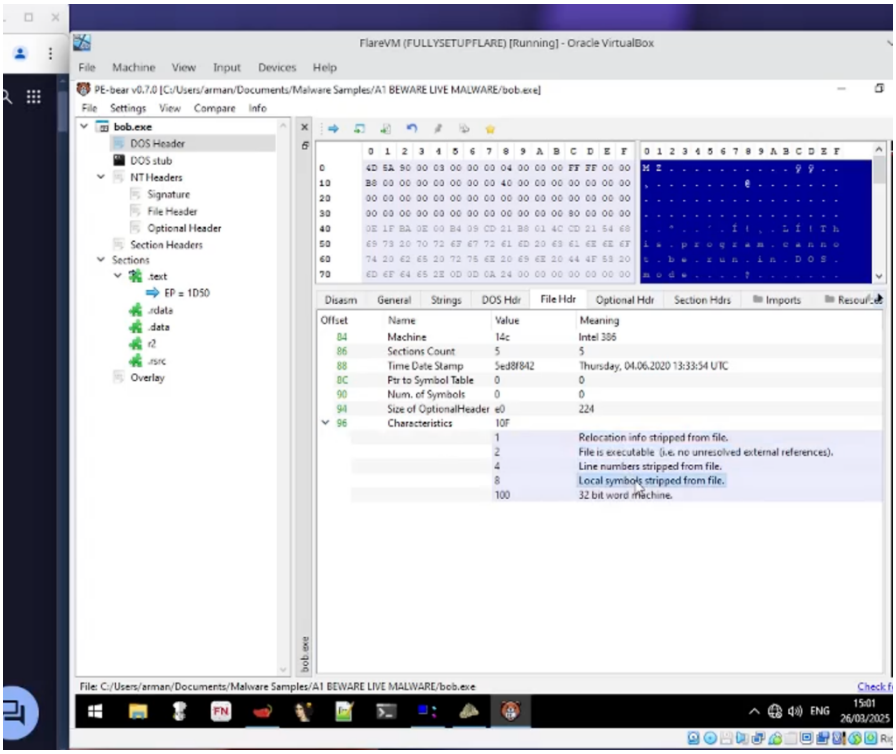


Identifies the entry point of file, which lies in the .text section. Identifies malware was compiled using Microsoft Visual C++ vx.x



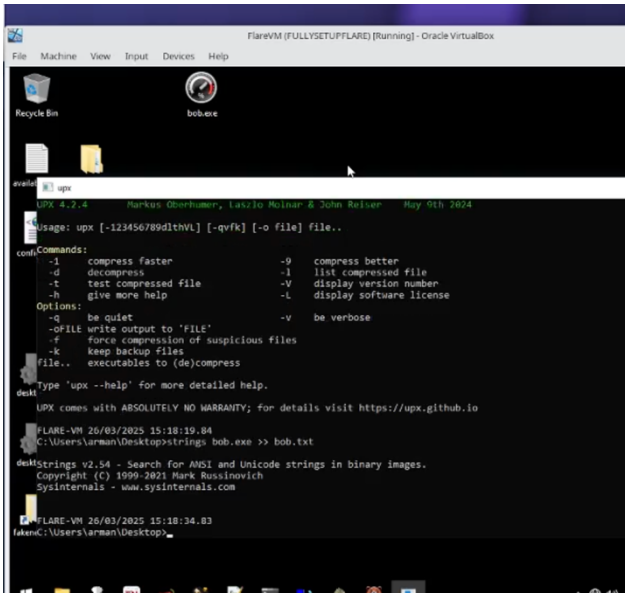
According to PEiD, the malware has an entropy of 3.09, which generally falls within the range of unpacked and non-obfuscated binaries. EP Check also reported not packed. However, the fast check says packed, sometimes producing false positive results.

PE-Bear

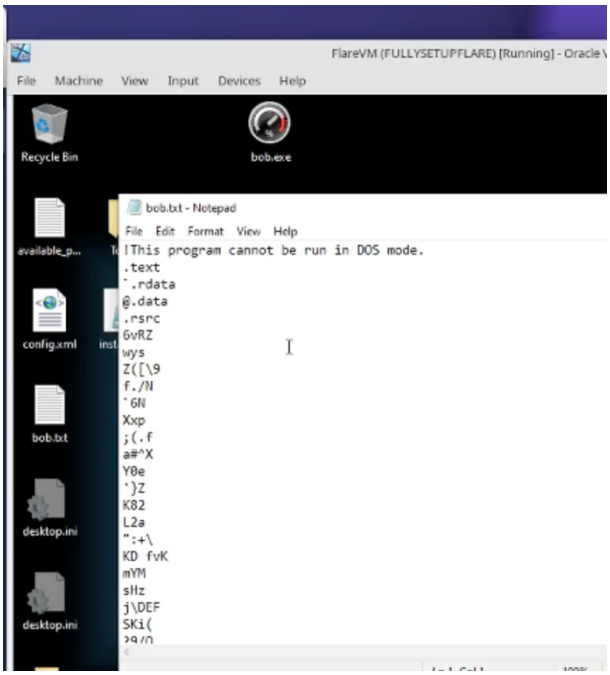


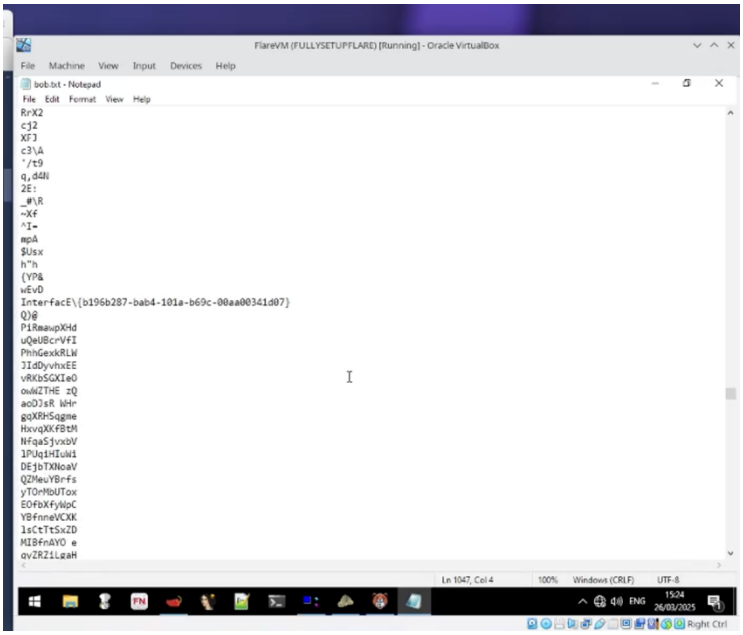
File header confirmed timestamp of compilation date of bob.exe. Characteristics revealed the file is an executable, and several components had been stripped, such as; relocation info, local symbols and line numbers. Confirms compilation date 04.06.2020 as seen before in VirusTotal.

String Analysis through txt file

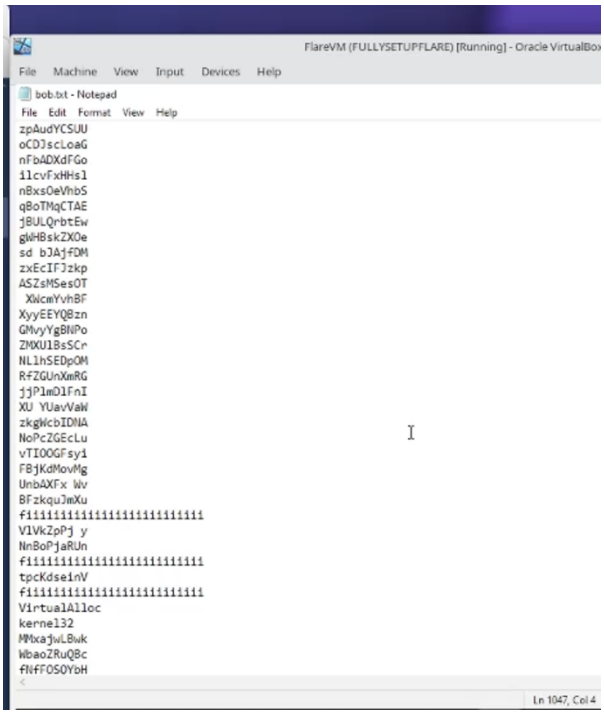


Analyzed strings by outputting the strings to a text file. This was achieved through UPX terminal using command line "strings bob.exe >> bob.txt". Then, I opened file in Notepad to analyze.

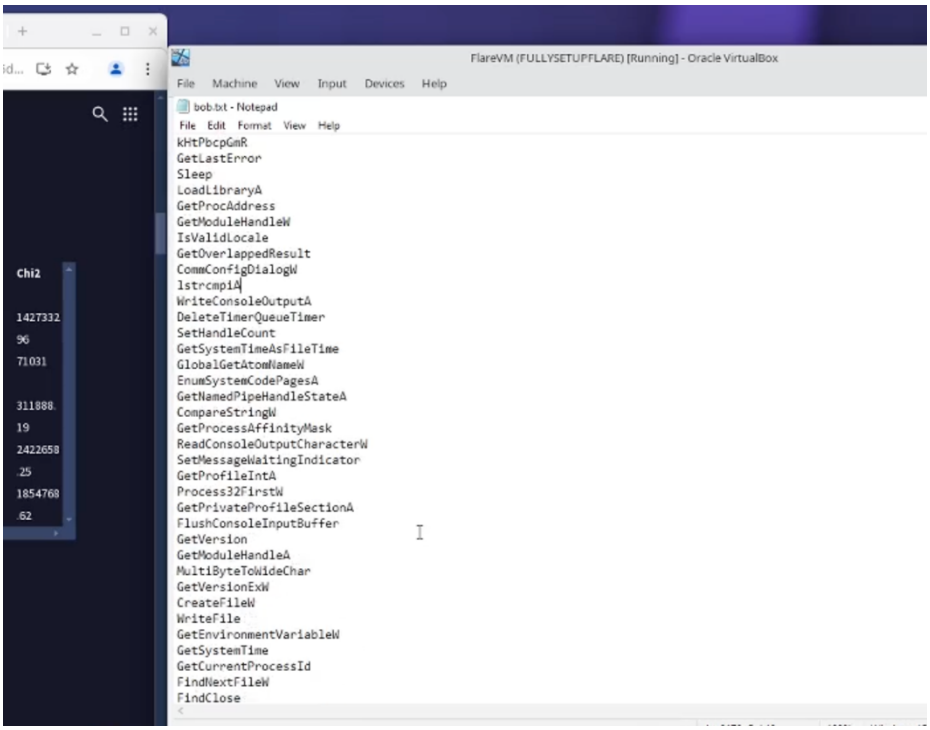




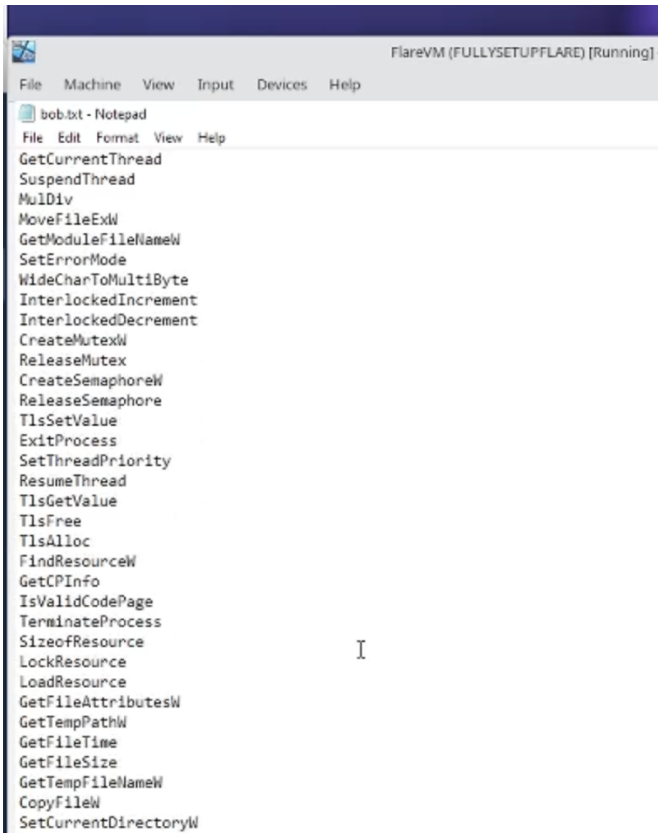
I can see that the strings at the start of the file are heavily obfuscated, making it difficult to interpret. We can see human-readable strings just over the halfway point of the text file.



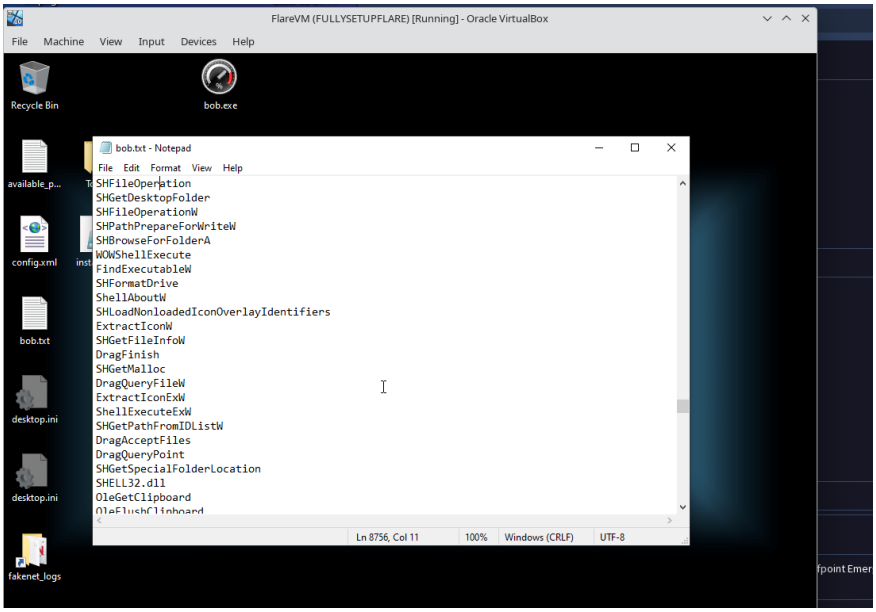
The repeated 'fiiiiii' string appears to serve as a placeholder or anti-analysis technique. Can see function VirtualAlloc, which is used for memory allocation and is often linked with shellcode loading or unpacking routines.



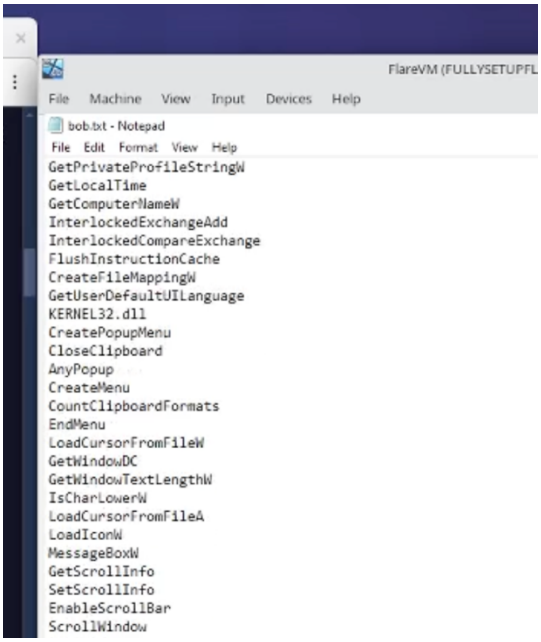
The GetLastError function retires most recent error code for debugging. The Sleep function is often used for anti-analysis, introducing delays to evade analysis. LoadLibraryA and GetProcAddress are frequently used for dynamic function resolution so the malware can load DLLs and resolve pointers at runtime. Can also see CreateFile and WriteFile, which suggests file system interaction.



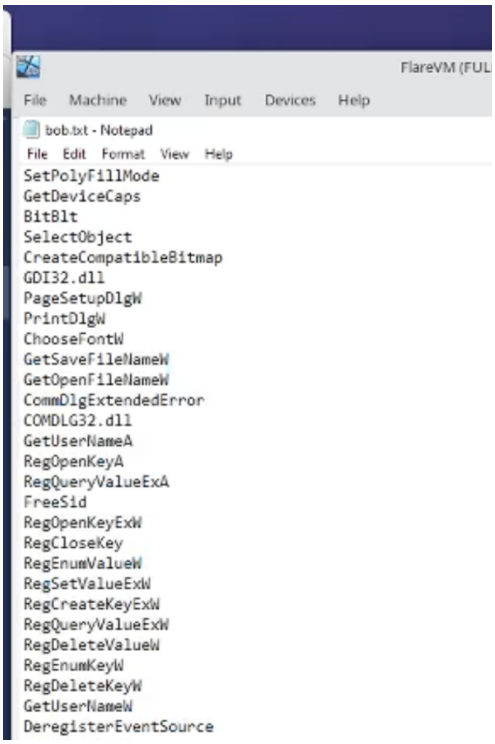
I can see thread-related functions: GetCurrentThread, SuspendThread, and ResumeThread, which allow malware to control thread execution. CreateMutexW and TLSAlloc, implying synchronization to manage multiple threads. Memory-focused functions; LoadResource, SizeofResource and LockResource - points toward unpacking/loading payloads during runtime.



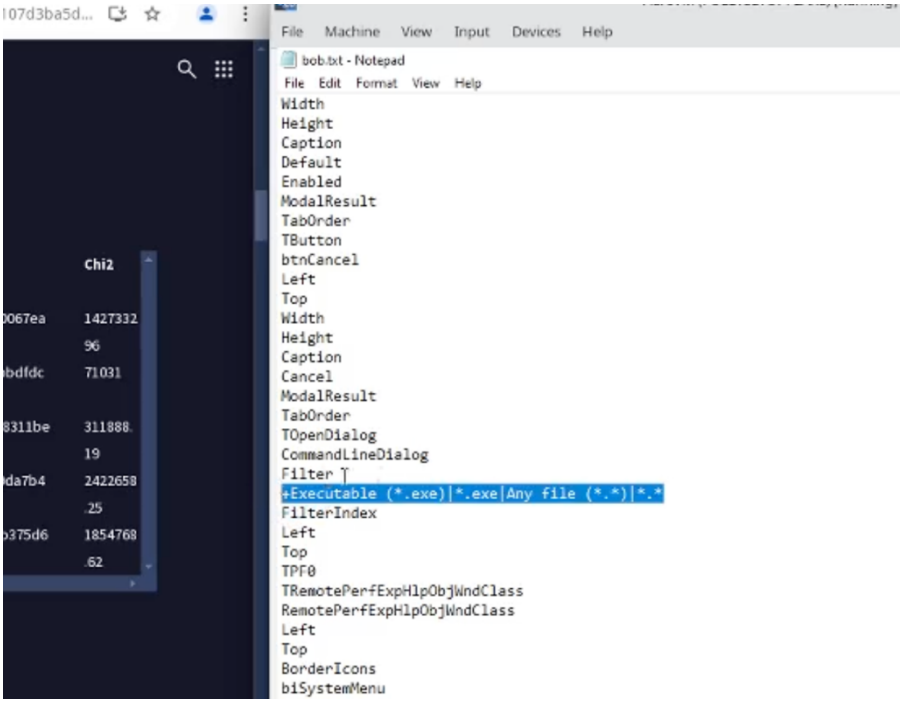
Can see functions ShellExecuteW, ShellExecuteExW, and WOWShellExecute - provides malware with ability to execute additional programs or scripts.



GUI-related functions, CreatePopupMenu, CreateMenu, ScrollWindow and IsWindowVisible, indicates attempts to trick the user or display fake system messages. Functions CloseClipboard and CountClipboardFormats suggest the malware attempting to monitor/interact with clipboard contents.



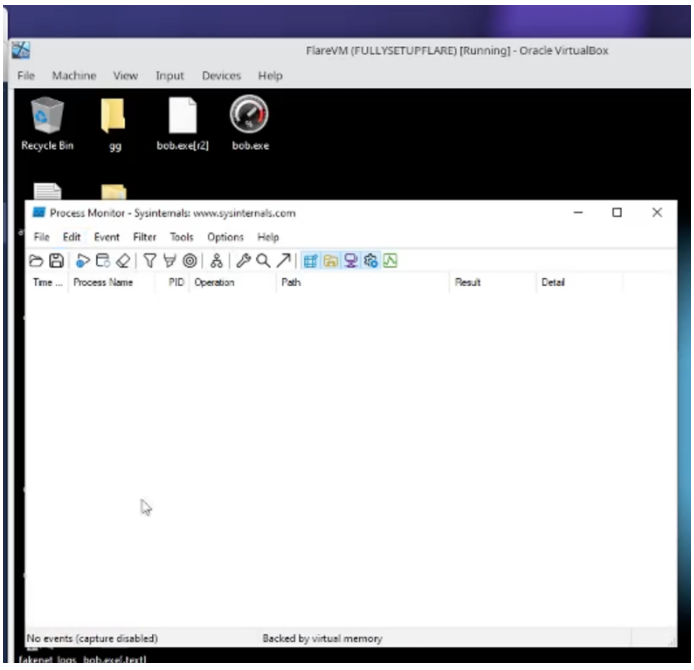
Registry-related functions such as RegOpenKeyA, RegOpenKeyExA, RegCreateKeyExW, RegSetValueExw and RegDeleteKey.

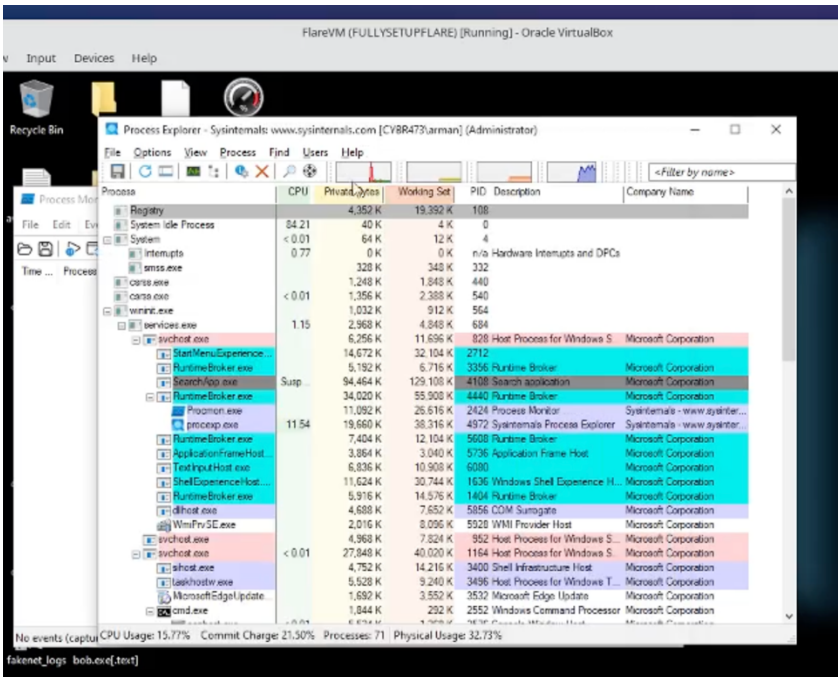


The +Executable function suggests malware may invoke fake/real-life dialog to trick users into selecting a .exe (based on GUI aspect) for execution or programmatically selecting .exe files for injection/execution.

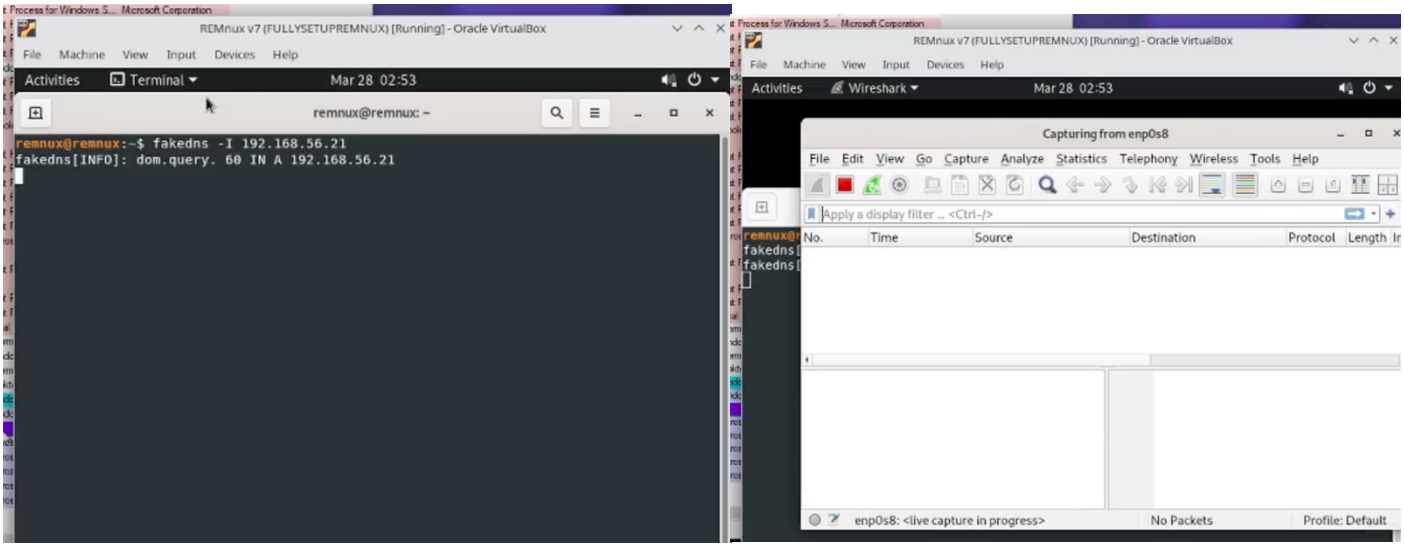
3.2. Basic Dynamic Analysis

Setup and Running Malware

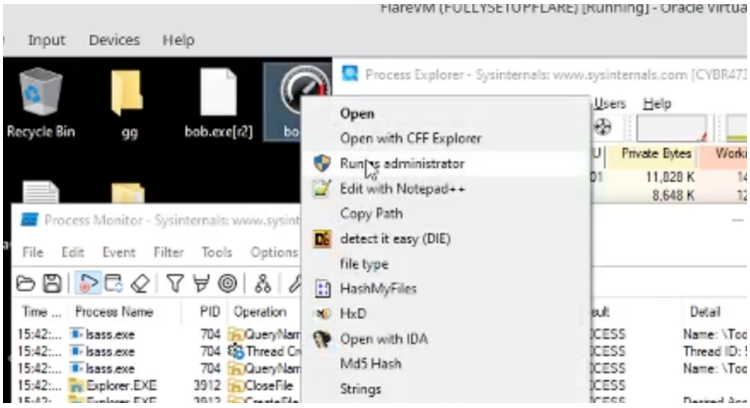


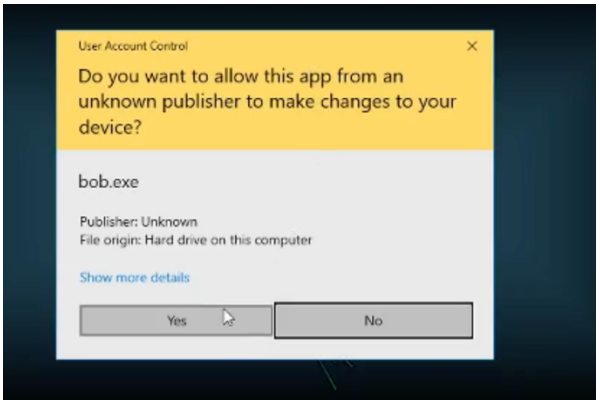


First launched Process Monitor and Process Explorer to monitor system processes in real-time.



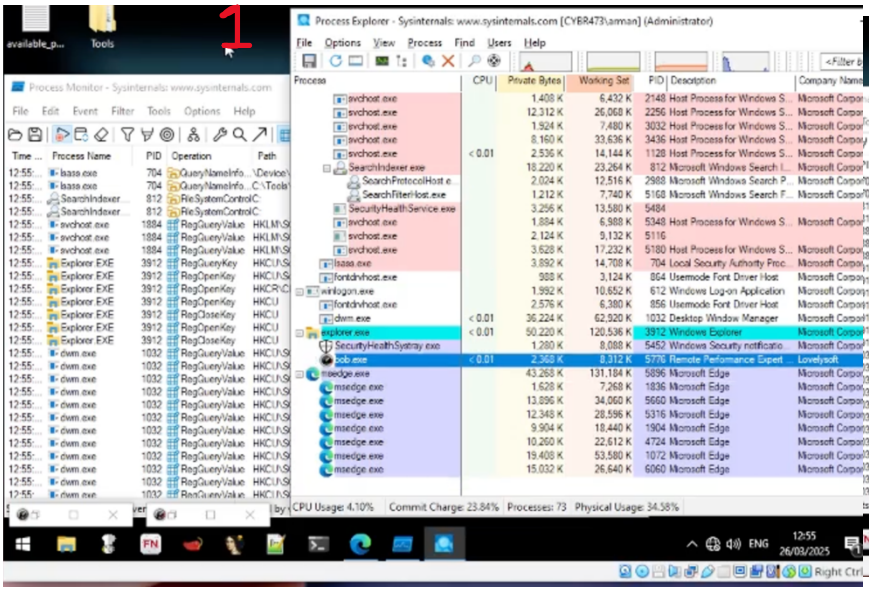
Setup a FakeDNS server to respond to DNS requests and Wireshark to listen to network traffic on enp0s8.

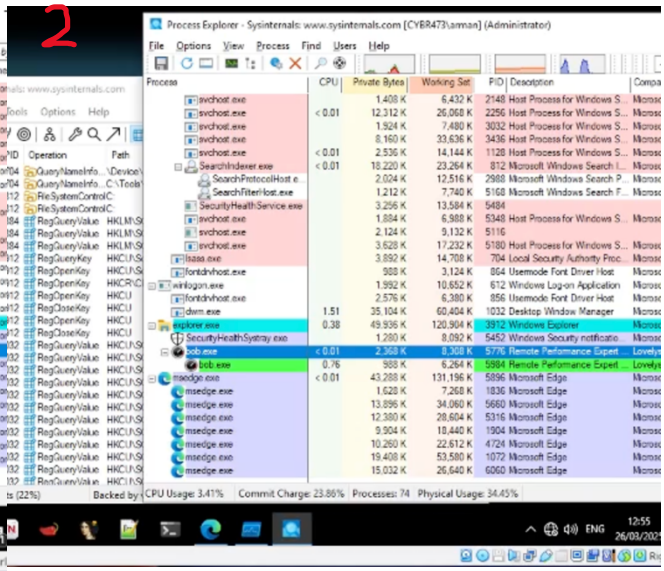


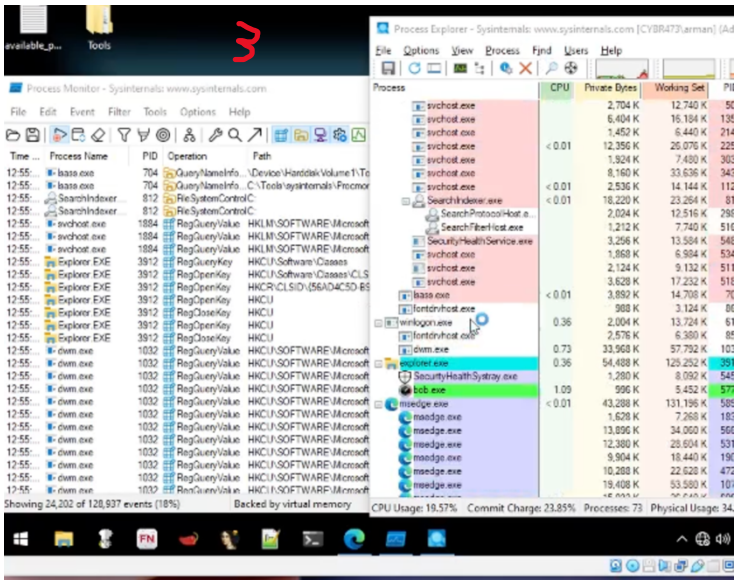


Executed bob.exe with administrative privilege.

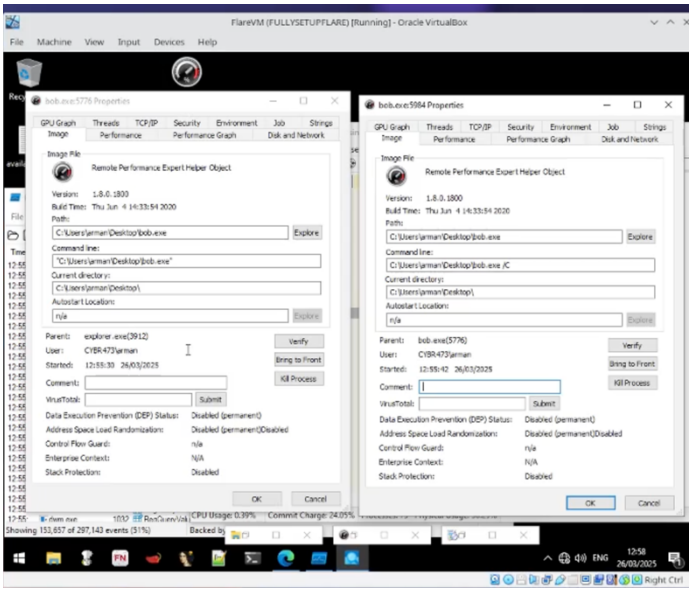
Process Explorer

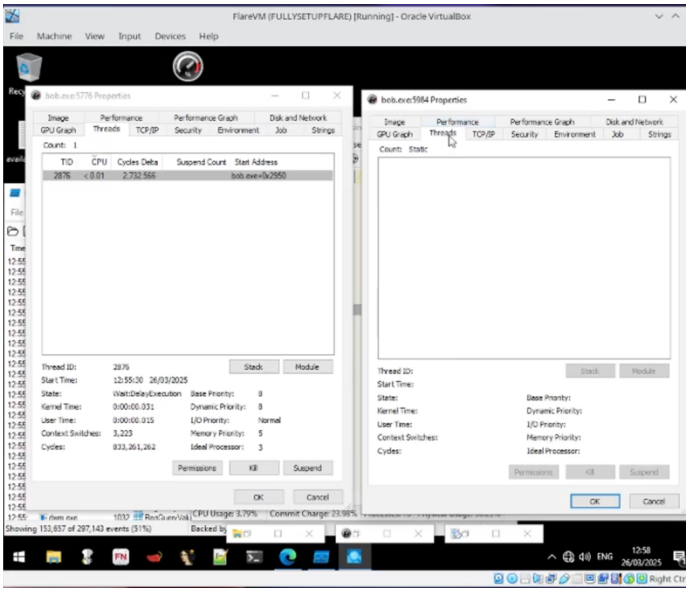


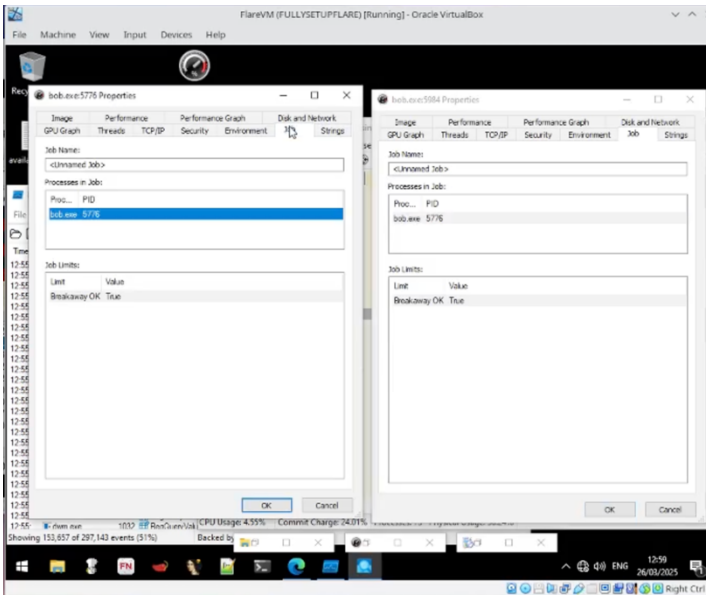




After executing bob.exe, bob.exe (PID 5576) process starts. Shortly after, a child process called bob.exe (PID 5984) is created. After that, the child process disappears, and the parent process (PID: 5776) is left. This behaviour could suggest process manipulation or injection to confuse analysis tools or evade detection.

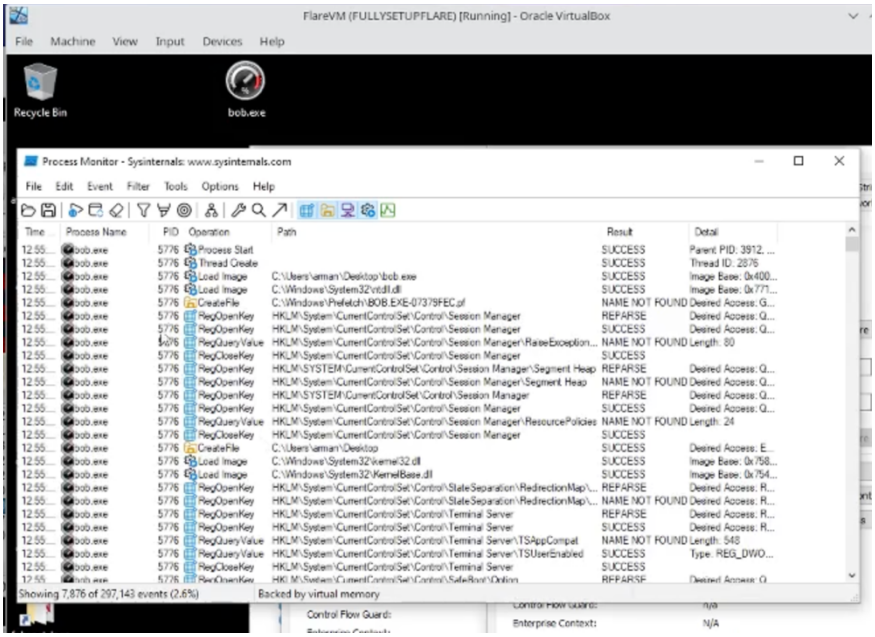




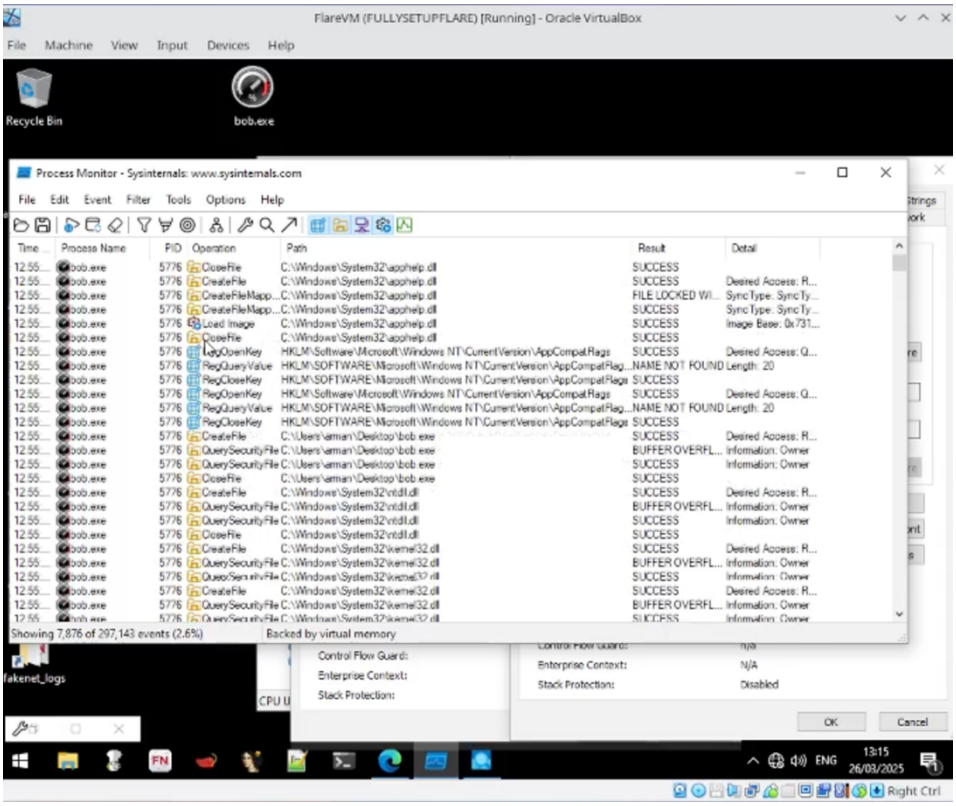


Child process (5984) started with a command-line argument (bob.exe /C). Child process appears to execute its payload due to its short lifespan and absence of threads after termination. Also runs same job PID (5776) as parent, suggesting parent remains running inheriting the same job object as the child. May indicate process hollowing or parent-injection techniques, where child is carrying out malware actions, making the parent appear less suspicious.

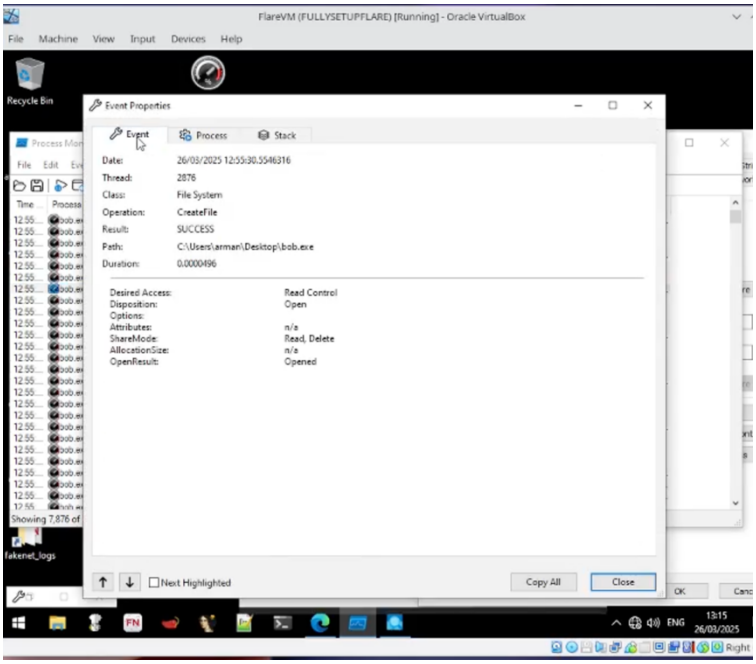
Process Monitor



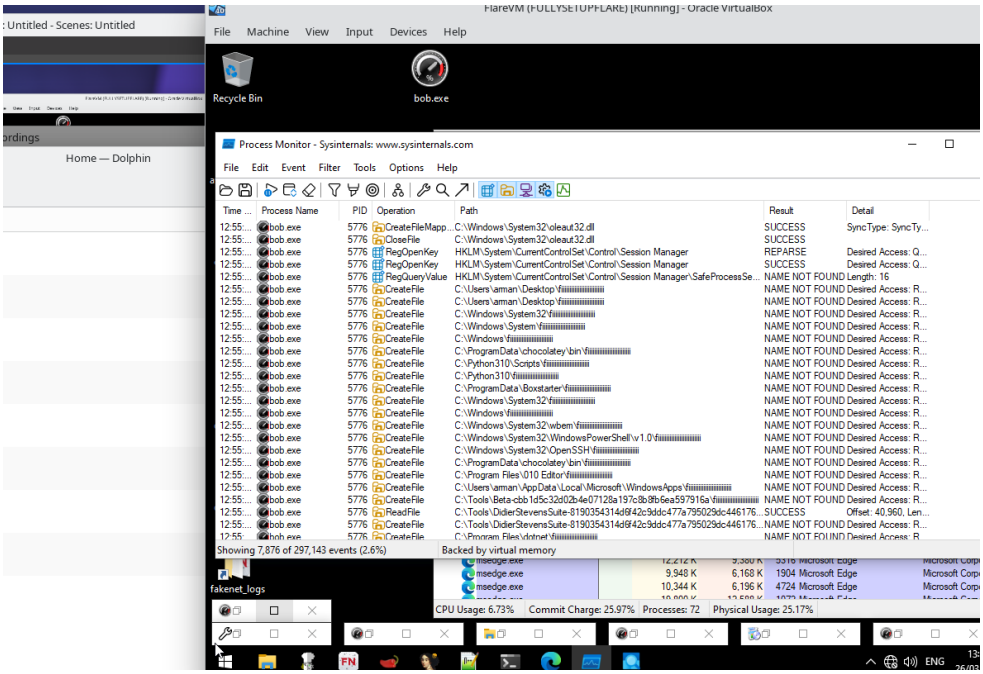
Start of parent process indicated by; Process Start, Thread Create, Load Image. I can see registry operations, RegOpenKey, RegQueryValue, and RegCloseKey, indicating system information gathering and possibly anti-analysis checks.



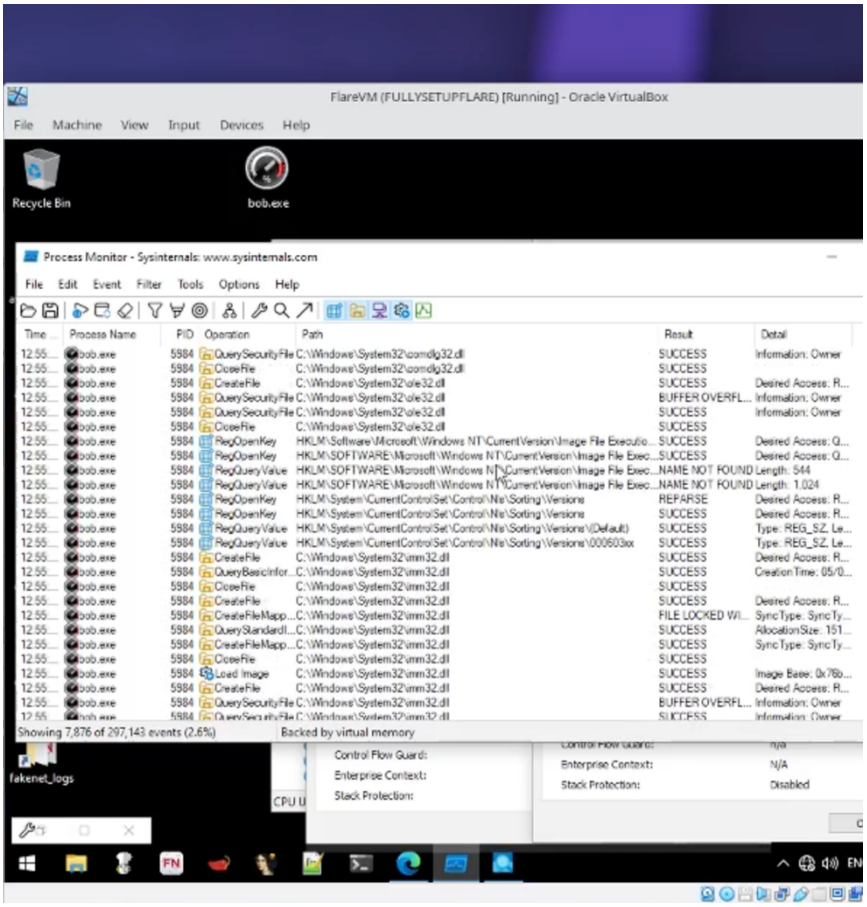
Can see file operations, CreateFile, WriteFile, and DeleteFile, indicating malware interacting with the file system, creating or modifying files and possibly cleaning up to avoid detection.



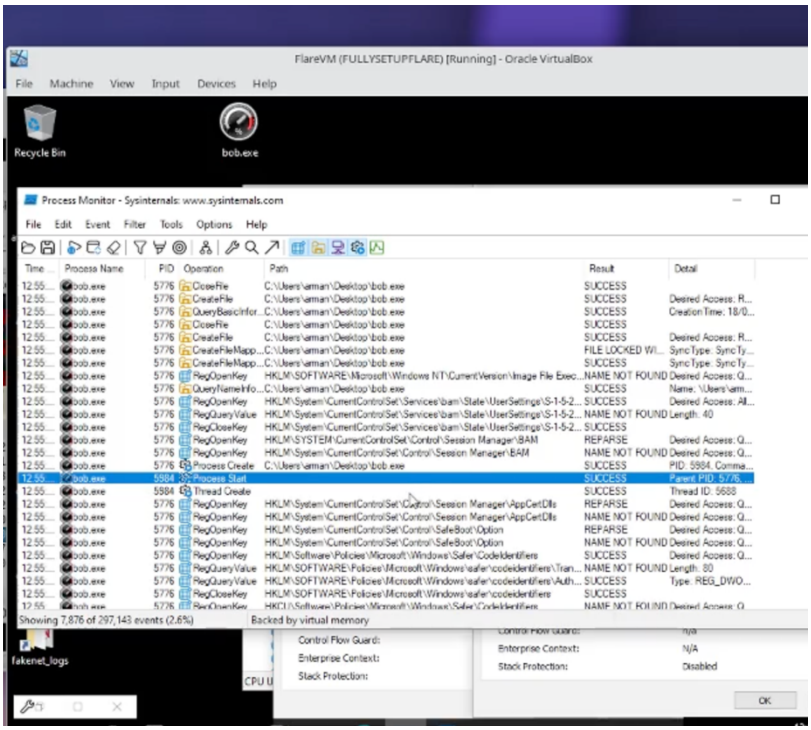
Can see parent process (PID: 5776) create child process (PID: 5984).



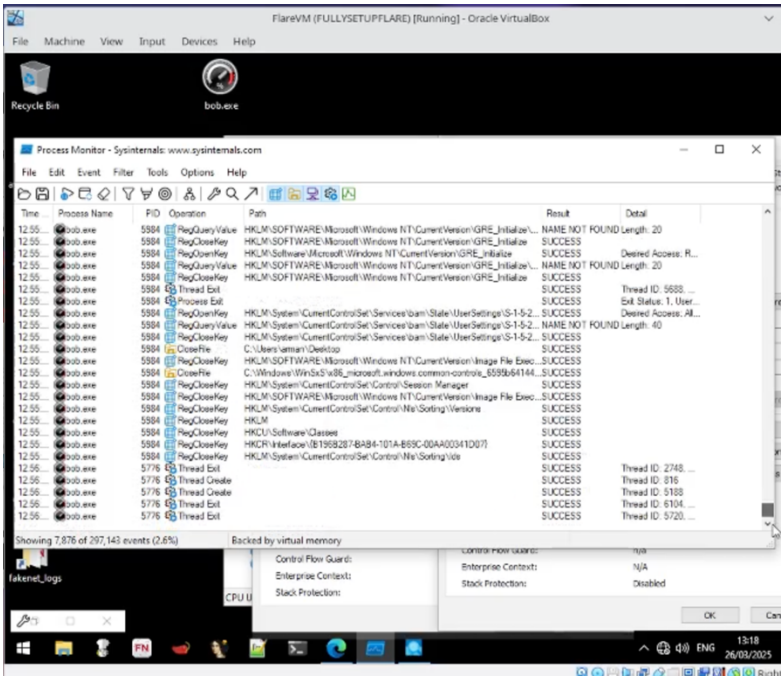
Can see CreateFile process, containing the repetitive string identified earlier, "fiiiiiiiiiiiii". Suggests malware may use this as markers or dummy filenames as a possible anti-analysis technique to check existence of sandbox-specific directories. NAME NOT FOUND indicates the malware is scanning the environment before proceeding.



Parent process (PID: 5776) finishes off making/creating child bob.exe. The child process (PID: 5984) starts its process.

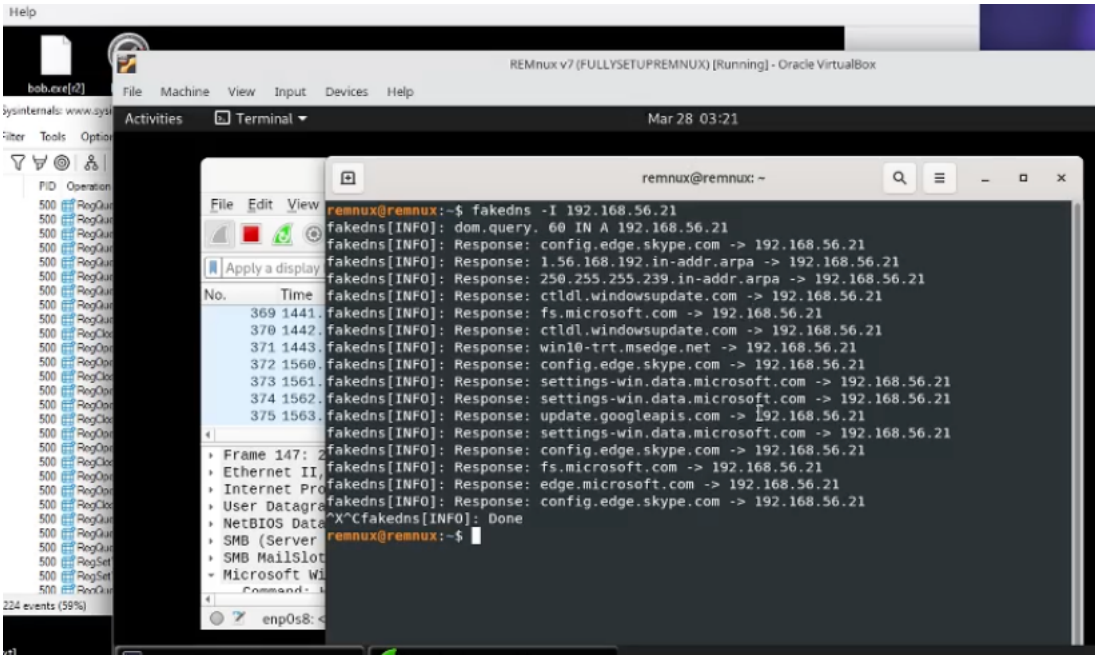


The child process performs similar actions with registry (RegOpenKey, RegQueryValue) and file processes (CreateFile, CloseFile).



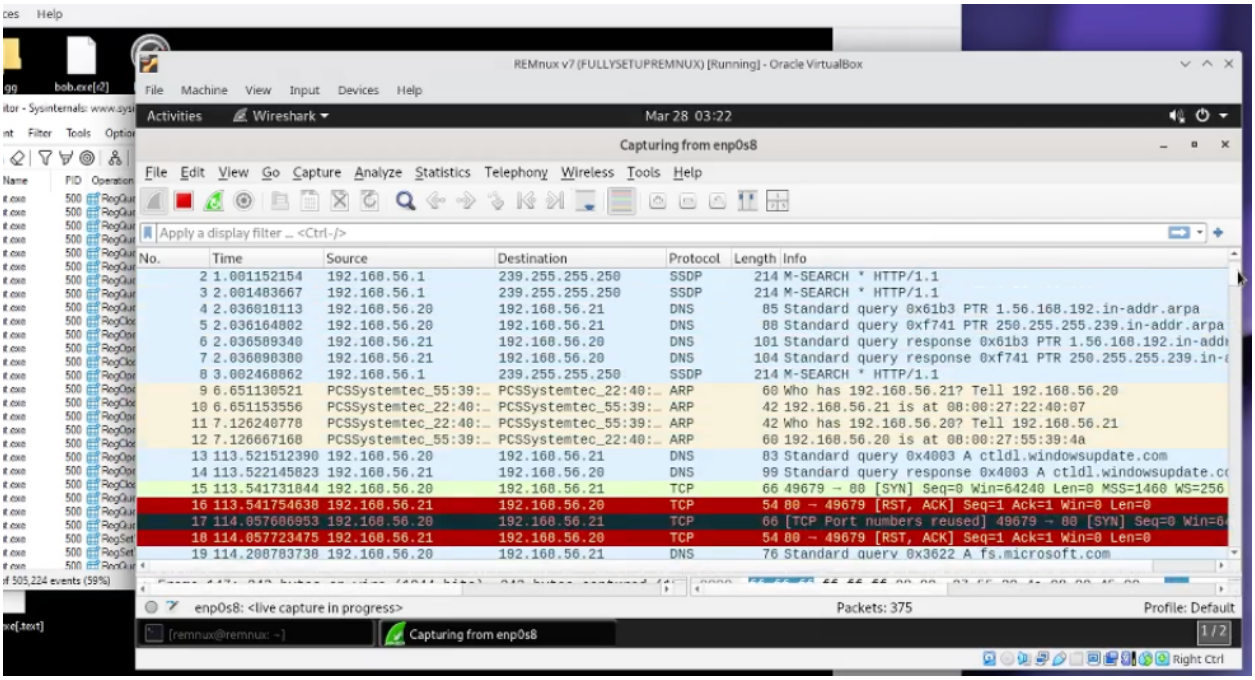
Child process (PID: 5984) performs majority of the actions in the malware. Can see the child's process finishing and exiting its process (Process Exit). After, we can see the parent process finishing its thread as well.

FakeDNS



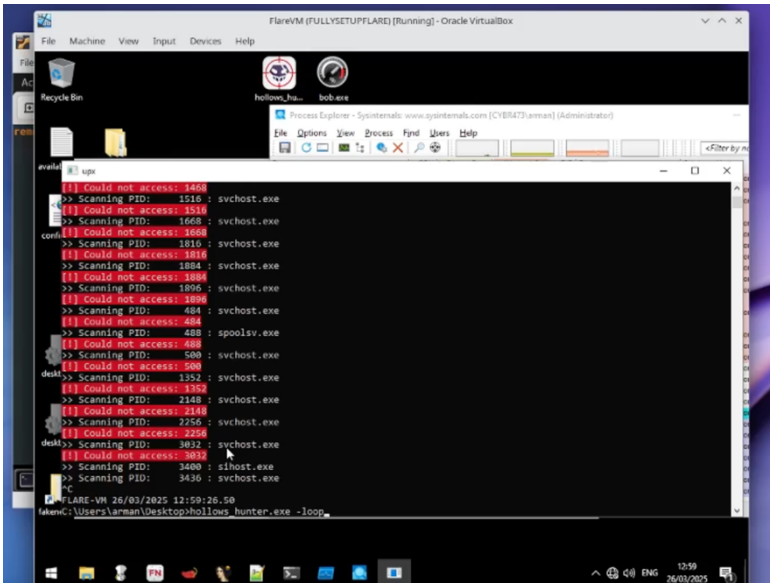
I can see FakeDNS making DNS requests for FlareVM and possibly the malware. Domains are being queried; however, no prior evidence was identified on these requests being made by the malware.

Wireshark

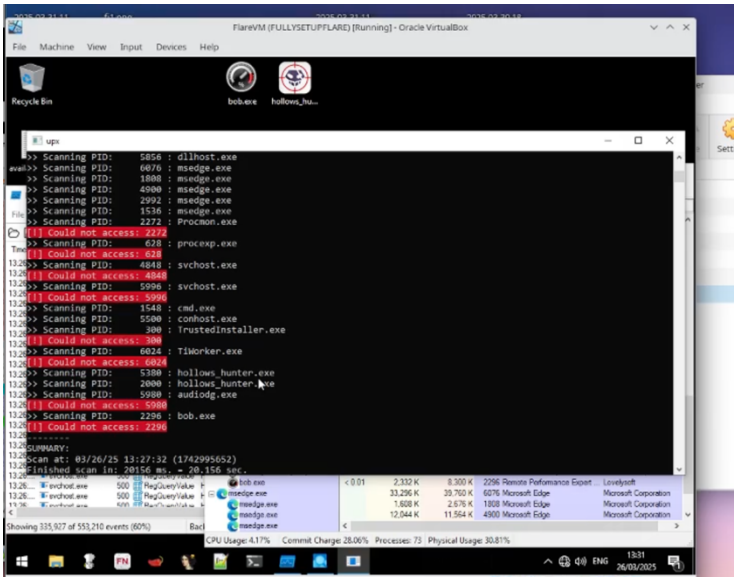


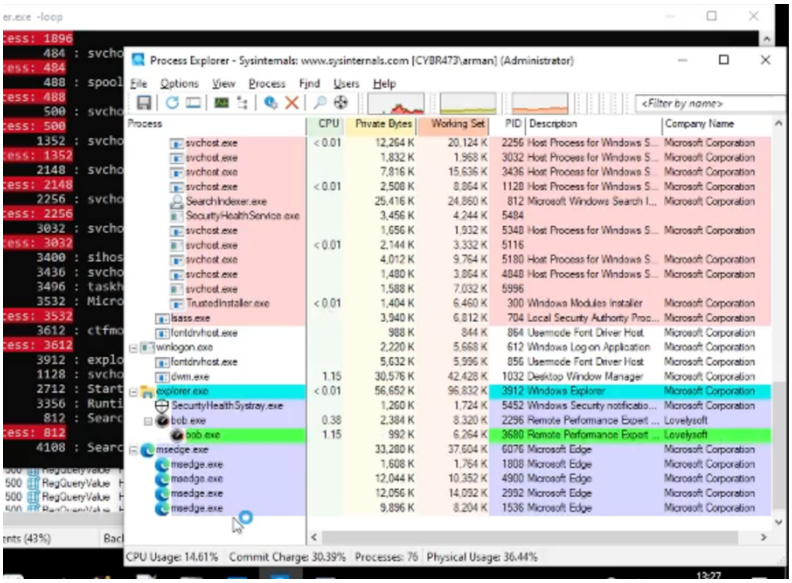
A successful connection didn't appear after the DNS request, as no follow-up HTTP request or data transfer was visible. Unclear whether request was from normal processes or malware due to lack of network-based identifiers analyzing the malware. There are limitations regarding observing/understanding network activities and malware processes using only basic dynamic analysis techniques. Applying advanced dynamic analysis techniques such as OllyDbg to observe TLS connections would help understand malware processes further.

Hollows Hunter



Ran hollows hunter on a loop, as there is no indication when malware has finished running, using command line: hollows-Hunter.exe -loop.





Hollows Hunter failed to access parent process (PID: 2296) and pick up child process (PID: 3680), suggesting the malware could possibly be using evasion techniques like handle manipulation to avoid detection during process scanning.

4. Lab Submissions

- Basic Static Analysis: https://vstream.au.panopto.com/Panopto/Pages/Viewer.aspx?id=adf7b2a9-e11a-4338-98cb-b2b101697306
- Basic Dynamic Analysis: https://vstream.au.panopto.com/Panopto/Pages/Viewer.aspx?id=a4755c3c-3d6e-4b02-bc04-b2b1016d133d

5. Word Count Clarification

Word count is 2982 for sections 1 – 3, which is the main report on the malware, excluding title and sections 4 – 6.

6. References and Tools

6.1. Tools Used For Static Analysis

- VirusTotal (https://www.virustotal.com)
- CFF Explorer (https://cff-explorer.com)
- PEiD (https://github.com/wolfram77web/app-peid)
- Detect it Easy (https://github.com/horsicq/Detect-It-Easy)
- 7-Zip (https://www.7-zip.org)
- PE-Bear (https://github.com/hasherezade/pe-bear)

- Dependency Walker (https://www.dependencywalker.com)
- Notepad (https://apps.microsoft.com/detail/9msmlrh6lzf3?hl=en-US&gl=US)

6.2. Tools Used For Dynamic Analysis

- Process Explorer (https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer)
- Process Monitor (https://learn.microsoft.com/en-us/sysinternals/downloads/procmon)
- FakeDNS (https://github.com/SocialExploits/fakedns/tree/main)
- Wireshark (https://www.wireshark.org)
- Hollows Hunter (https://github.com/hasherezade/hollows_hunter)

6.3. References/Websites

- ChatGPT (Helped in understanding and clarification of malware analysis behaviour, processes and patterns)
- https://www.youtube.com/watch?v=1Kb6tee2eJI&ab_channel=screeck (Used to help build lab environment setup for live malware analysis).
- https://www.cisa.gov/news-events/alerts/2017/05/12/indicators-associated-wannacry-ransomware (Used to understand malware HTTP kill chain process)
- https://learn.microsoft.com/en-us/windows/win32/dlls/dynamic-link-libraries (Microsoft documentation to understand processes and DLL purposes)