12/ 21/2023

Ramy Naiem

Reverse Engineering Cw2

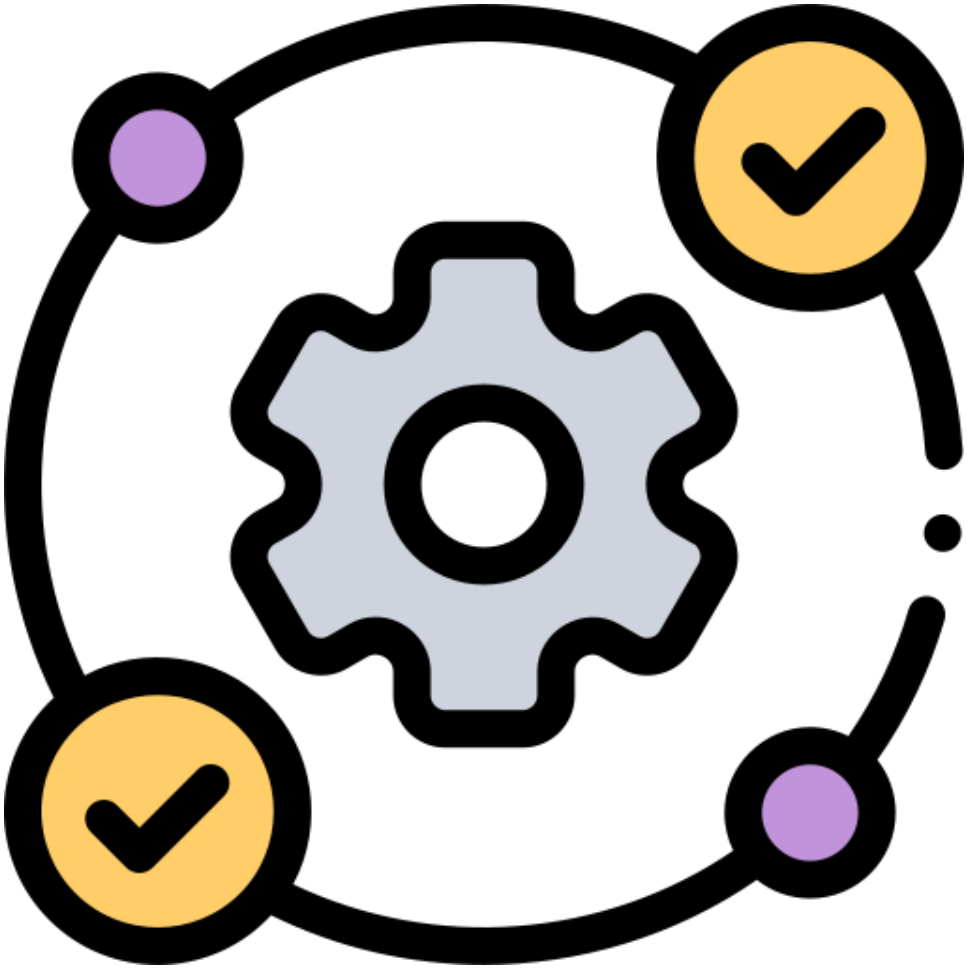
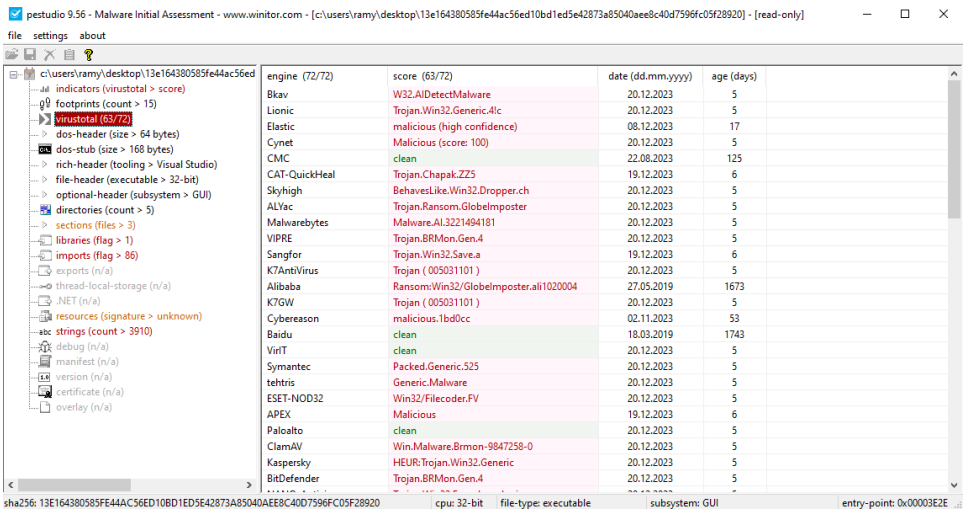


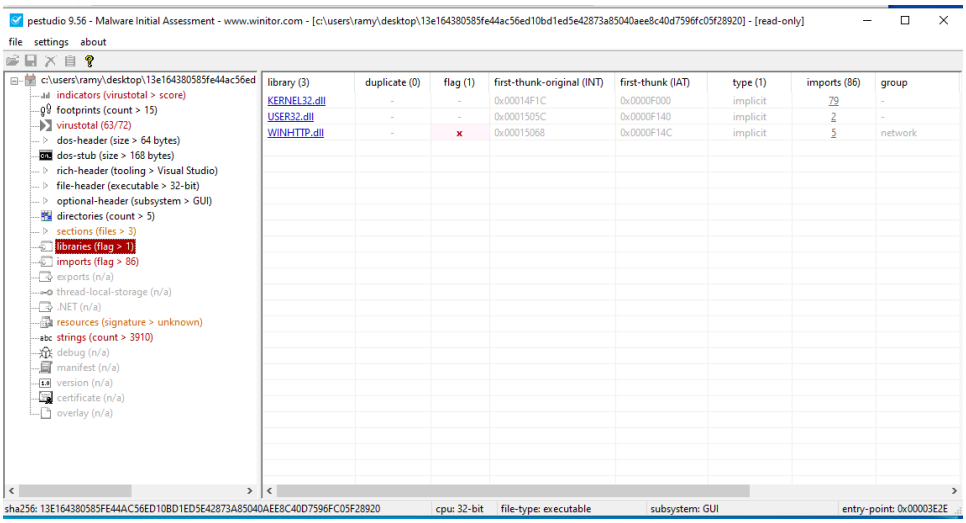
Table of Contents:

Static analysis

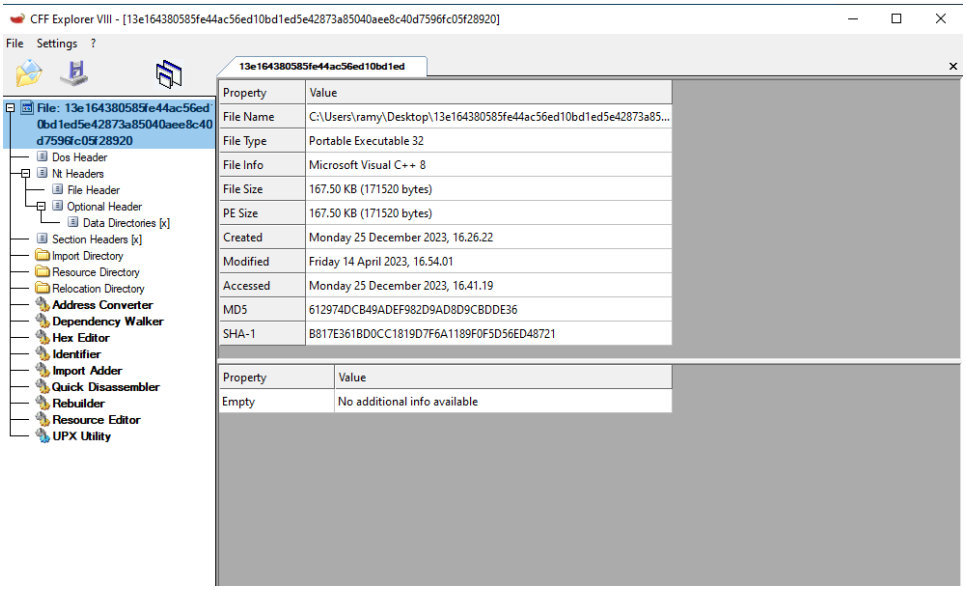
When opening the Malware in Pe studio we can directly see that the file is malware because Virust total flags it 63/72 as malware.



Libraries:



Creation date: 25 December (When the file is extracted from the winrar folder)



Modified Friday 14 April 2023

Synchronization Functions: Functions like DeleteCriticalSection, InitializeCriticalSectionAndSpinCount, EnterCriticalSection, and LeaveCriticalSection suggest the program manages concurrent access to

resources. This is common in multi-threaded applications where data integrity during concurrent operations is crucial.

File Operations: Functions such as CreateFileW, WriteFile, GetFileSize, and SetFilePointerEx indicate interaction with the file system, which could be for legitimate data handling or possibly for accessing and manipulating files without user knowledge.

Memory Management: The presence of GlobalMemoryStatus, HeapAlloc, HeapFree, etc., signifies dynamic memory allocation and management within the program. This is typical in applications that handle varying data sizes or require efficient memory usage.

Network Communication: Functions like WinHttpOpen, WinHttpConnect, WinHttpReadData highlight the program's capability to establish network connections and transfer data. This could be for routine data exchange over the internet or potentially for covert communications in the case of malware.

System Information and Reconnaissance: Functions such as GetCurrentProcessId, GetTickCount, IsDebuggerPresent are indicative of the program gathering system information or monitoring its environment, which could be used for adapting its behavior or for anti-debugging purposes in malicious software.

Exception Handling and Diagnostics: RaiseException, UnhandledExceptionFilter, and OutputDebugStringW are used for exception handling and debugging. This could be part of robust error handling in legitimate software or used in sophisticated malware to evade analysis.

Dynamic Library and Execution Functions: The use of GetProcAddress, GetModuleHandleA, and ExitProcess reflects dynamic library loading and process control, which is common in modular applications but can also be a technique in malware for runtime resolution of dependencies or terminating processes.

Console and User Interface Interactions: Functions like WriteConsoleW and GetConsoleMode suggest interactions with the console, which could be part of a command-line interface or for other user interaction purposes.

File Paths and Names: Strings indicating file paths or names, such as "C:\Windows\System32\" or "settings.conf", can unveil the files the program interacts with, suggesting its operational scope.

URLs and Network Resources: Discovering strings that are formatted as URLs, domain names, or IP addresses, like "https://www.example.com/update", hints at the program's network communication capabilities or external dependencies.

User Interface Text: Strings resembling dialog messages, button labels, or menu items, for instance, "Failed to load module" or "Enter password:", are indicative of the application’s interactive aspects and user interface design.

Registry Keys and System Resources: References to specific registry keys or system resources in strings, such as "HKEY_LOCAL_MACHINE\Software\Example\", suggest interactions with the Windows Registry for configuration or state management.

Command-Line Commands or Scripts: The presence of command-line instructions or script contents within strings, like "netstat -an" or "echo %username%", implies that the program may execute these commands, potentially for configuration or system operations.

Encoded or Obfuscated Strings: Encoded strings (in formats like Base64 or Hexadecimal) or seemingly obfuscated text could signal efforts to conceal certain program functionalities or data, a common practice in malware.

Debugging or Log Messages: Strings that look like debug messages or logging information, such as "Connection established" or "Error logged in file", offer insights into the program's internal processes and error-handling routines.

Cryptographic Keys and Certificates: Strings resembling cryptographic keys or certificates, perhaps in the format of long alphanumeric sequences, indicate encryption-related operations, secure communication, or data integrity verification processes.

API Names and External References: Identifying strings that are names of APIs or references to external libraries, like "OpenSSL_encrypt" or "kernel32.dll", sheds light on the program's use of external functionalities and system interactions.

Usernames, Passwords, or Tokens: The presence of strings that appear to be hardcoded credentials, such as specific usernames, passwords, or security tokens, is a critical security concern, often indicating authentication mechanisms or access control methods.

Brand Names or Product References: Strings containing specific brand names, product names, or even developer comments can sometimes provide clues about the software’s origin, purpose, or affiliations.

ADV STATIC

FUN_00401a30: This function exhibits a high level of complexity, with overlapping instructions and intricate pointer arithmetic. Such a structure is indicative of obfuscation techniques, often employed in malware to evade detection by analysis tools. The use of multiple, nested conditional statements and arithmetic operations suggests that this function could be involved in data encoding, decoding, or performing cryptographic operations. Its complexity is likely intended to complicate reverse engineering efforts.

FUN_004041c0: This function appears to be involved in string manipulation and numerical conversion, likely a part of the malware's data parsing or processing mechanism. It conditionally handles different types of data based on the parameters passed, suggesting versatility in handling various data formats. This can be a critical component in malware, especially for processing command and control communications or extracting information from a compromised system.

FUN_00405962: This function includes multiple mathematical and bitwise operations, which could be integral to complex data manipulation tasks, such as encryption or hashing. The use of local function calls and conditional checks points towards a sophisticated data processing routine. It might be involved in generating, modifying, or verifying data in a way that is crucial to the malware's operation, like maintaining the integrity of data or securing communications.

FUN_00408808: This is a more straightforward function focused on copying data between structures. Such a functionality is essential in many software contexts, including malware, for tasks like replicating data, setting up necessary data structures, or moving data into specific formats required for further processing. Its simpler nature compared to other functions doesn't diminish its potential importance in the overall operation of the malware.

FUN_0040e5a9: This function involves repetitive modification of a character string. The loop structure and the nature of operations suggest a potential role in data encoding or performing a simple obfuscation

technique. Such a function can be part of a mechanism to alter data to avoid detection or prepare it for transmission or storage.

Icon and Initial Appearance: The ransomware loader is accompanied by the following icon, and it was compiled in 2016 but first detected in-the-wild on December 4, 2017.

Installation Process: To ensure persistence after a system reboot, it creates a registry key in [HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce], which points to an executable file in the user's AppData Roaming directory.

File Creation and Decryption: GlobeImposter generates a file with a 256-bit RC4 key in %All Users%, which is essential for decrypting the ransomware's configuration. The malware decrypts itself in blocks, storing the decrypted content in a temporary file in the %Temp% folder. It also extracts components like System.dll from the .NET framework and drops an encrypted file in the Temp folder.

Configuration Decryption: A SHA256 hash is used as a key to decrypt the ransomware's configuration. This decryption process involves the use of the RC4 cipher with a 256-bit key.

Detailed Configuration Contents: The configuration includes a list of folder and file extension exclusions, a specific string for appending to encrypted files, the ransom note filename ('Read___ME.html'), and additional data, primarily consisting of zeros.

Process Termination List: The ransomware maintains a list of processes to terminate, which is stored outside of the encrypted config, in the payload body.

Encryption Key Details: GlobeImposter loads a hard-coded 256-bit key (HCK265) used to generate the AES key and IV for file encryption. A key file with session keys is created, containing auxiliary data encrypted with AES-256-CBC.

File Encryption Mechanism: The ransomware encrypts files using the AES-256-CBC algorithm. It calculates the IV and AES key from the hardcoded key and applies specific conditions before encrypting each file.

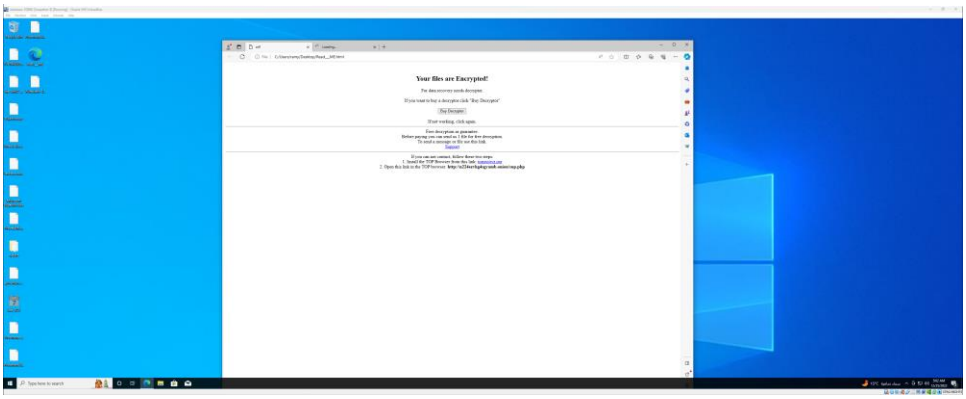
Encryption Footer: The encryption footer includes 32 bytes of the encrypted AES-256 key, 16 bytes of IV, and 768 bytes of encrypted auxiliary data.

Backup Removal and Ransom Note: GlobeImposter removes shadow copies, disables remote desktop capabilities, and clears Windows event logs by executing a batch file. It also creates a ransom note file named 'Read___ME.html.'

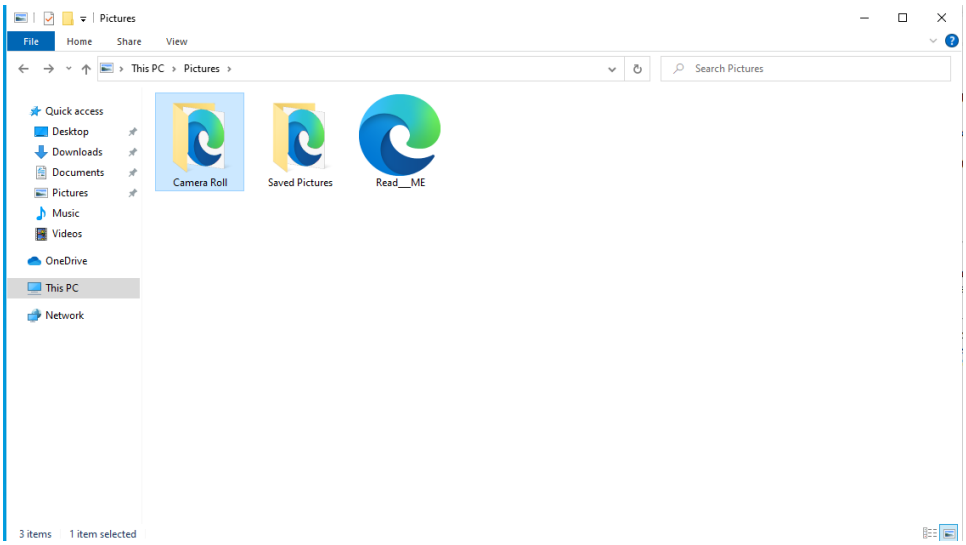
Communication with C&C Servers: The ransomware communicates with specific IP addresses and provides a decryption service through an onion link.

Simple dynamic

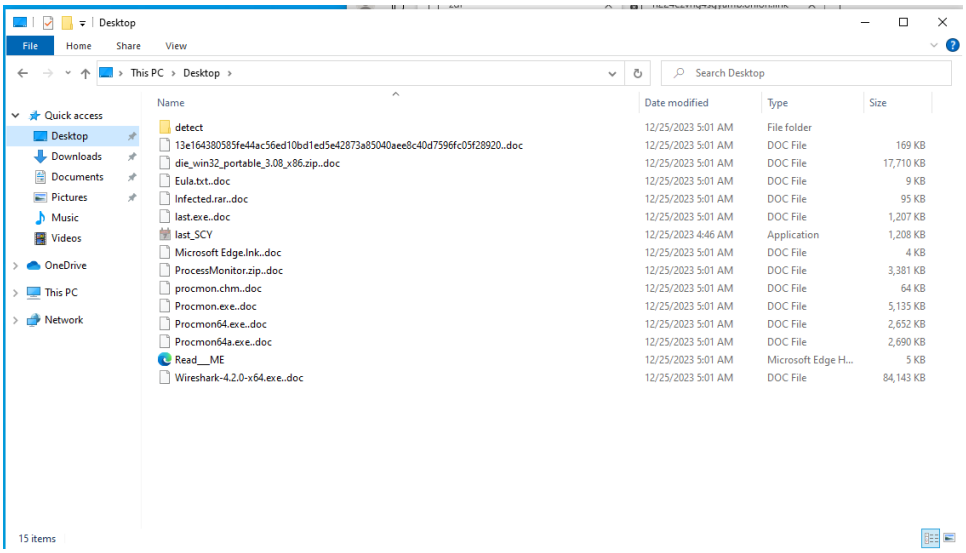
As soon as the malware runs it decrypts all the files on the target system. and a Readme file gets created on the Dekstop. When the file is opend it goes to a page that has a message “Your files are Encrypted” with a link to buy a Decryptor where the Url doesn’t exist anymore.



In every directory on the pc the readme file gets created.



All the files that are encrypted are ending now with .doc.



ADV Dynamic

System and Configuration File Modification: Files like AUTOEXEC.BAT, CONFIG.SYS, IO.SYS, MSDOS.SYS, and NTDETECT.COM are crucial for system operation. Malware may modify these to ensure persistence, launch malicious processes at startup, or compromise system integrity.

Executable File Replacement or Mimicry: The presence of executables in common directories (e.g., AcroRd32.exe, firefox.exe, wmplayer.exe) might indicate that malware is either replacing legitimate files with malicious ones or disguising itself as common applications to avoid detection.

Scripts and Python Files: Numerous Python scripts and related files could be tools used by the malware for various purposes such as spreading to other systems, downloading additional payloads, or executing specific tasks.

Unusual File Names and Locations: Executables and files with long alphanumeric names in temporary or user directories (e.g., files in C:\Documents and Settings\Administrator\Local Settings\Temp) could be directly related to the malware. These might be payloads, data exfiltration tools, or part of a command and control structure.

Modification of User and System Data: Files in user directories and system recycle bins may indicate that the malware is attempting to access and possibly exfiltrate user data, or conceal its activities by hiding in less commonly scrutinized locations.

Presence in Critical System Folders: Malware files located in system32, Windows, and other critical system directories can indicate attempts to interfere with system processes, monitor user activities, or gain elevated privileges.

Signs of Rootkit or Bootkit: The presence of files like ntldr, boot.ini, and bootfont.bin can be indicative of rootkit or bootkit behavior, where the malware attempts to infect the system at a very deep level, often before the full operating system loads.

Part 2:

System File Interference: Files like AUTOEXEC.BAT, CONFIG.SYS, IO.SYS, MSDOS.SYS, and NTDETECT.COM are critical system files. The malware may have modified or replaced them to gain persistence, disrupt normal system operations, or hide its activity.

Executable File Creation: The presence of .exe files in user directories, such as 996E.exe and the long alphanumeric named file in Documents and Settings, suggests the malware is deploying executables for malicious purposes, possibly as part of its payload or to further spread the infection.

Python Environment Manipulation: The extensive list of Python-related files (Python27 directory and its subdirectories) indicates that the malware could be using Python scripts for automation of tasks, spreading itself, or even for complex operations like data exfiltration or encryption (in the case of ransomware).

Recycler Directory Usage: Files in C:\RECYCLER suggest that the malware might be trying to hide its components or data in areas of the system that are less likely to be scrutinized.

User Data Targeting: The presence of files in C:\Users\Public and C:\Users\user directories can indicate attempts to access user data, possibly for theft or encryption.

Suspicious File Creation in App Data: Files in C:\Users\user\AppData\LocalLow and C:\Users\user\AppData\Roaming indicate that the malware may be using application data folders to store its components or configuration data, which is common in persistent threats.

Modification of Boot and System Configuration Files: The presence of boot.ini, ntldr, and similar files suggests that the malware may be attempting to modify boot processes or system configurations for persistence or to cause system instability.

Creation of Readme Files: Numerous Read___ME.html files across different directories might be indicators of ransomware, which often leaves such files with ransom notes or instructions for the victim.

Potential Web and Browser Interference: Files in directories related to Mozilla Firefox (C:\Users\user\AppData\Roaming\Mozilla\Firefox) could imply attempts to steal browser data, manipulate web sessions, or even inject malicious content into web pages.

Executable and Configuration Files in Root Directory: The presence of install.exe, vcredist.bmp, tools.rar, and similar files in the root directory (C:) might be related to the initial infection vector or secondary payload deployment.

Processes Created

Google Chrome Process with Specific Argument: The chrome.exe process initiated with the --start-maximized --single-argument flags and pointing to a Read___ME.html file in the Startup folder suggests an attempt to automatically display a specific webpage or message upon system startup. This could be a decoy, distraction, or part of a phishing attempt.

Chrome Utility Process: Another instance of chrome.exe with various flags like --type=utility, --utility-sub-type=network.mojom.NetworkService, and several --disable-features options indicate the malware may be trying to use Chrome's utility processes for its operations, possibly for network communication or to blend in with normal traffic.

Executable in AppData Roaming: The process C:\Users\user\AppData\Roaming\executable.exe suggests the malware has placed an executable in the user's AppData Roaming directory, a common tactic for ensuring persistence, as files in this location are typically executed on startup.

Suspicious Executable Path: The process with the path %SAMPLEPATH%\13e164380585fe44ac56ed10bd1ed5e42873a85040aee8c40d7596fc05f28920.exe is notable for its complex name, possibly indicative of a unique identifier or hash value. This could be a dropped payload or a component of the malware.

Desktop Executable: C:\Users\user\Desktop\executable.exe could be another payload or a tool used by the malware, placed on the desktop for execution.

OpenWith Process: The OpenWith.exe process with -Embedding flag is curious. This executable is typically associated with the Windows Open With dialog, but its use here could be an attempt to masquerade as a legitimate system process.

Windows Update API Host: wuapihost.exe is a legitimate Windows process related to the Windows Update API. However, its execution in this context might indicate an attempt by the malware to disguise its activity as a legitimate update process or to interfere with Windows updates.

Processes Terminated

Termination of OpenWith.exe: This might be an attempt to shut down a legitimate system process to replace it with a malicious one, or it could be part of a cleanup routine post-execution of a malicious task.

Termination of wuapihost.exe: Similar to OpenWith.exe, this could be an attempt to interfere with Windows Update processes, possibly to prevent the system from receiving updates that could detect or interfere with the malware.

User Preferences and System Settings: Changes to keys like HKEY_CURRENT_USER\Control Panel\International\Geo and various HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer subkeys suggest the malware may be altering user settings and system preferences, potentially to suit its operational needs or to disrupt normal usage.

File Associations and Handlers: Modifications in HKEY_CURRENT_USER\Software\Classes and its subkeys, including those related to file extensions (e.g., .doc), indicate that the malware might be hijacking file associations. This can redirect the opening of certain files to malicious executables or scripts.

Startup and Persistence Mechanisms: The usage of HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce and similar keys suggests that the malware is establishing persistence mechanisms to ensure it's executed every time the system starts.

Disabling Security Measures and Policies: Changes to HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Explorer and HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Safer\CodeIdentifiers hint at attempts to weaken or bypass security policies and controls, potentially to allow unrestricted operation of the malware or to disable warnings and restrictions that might alert users to its presence.

Manipulation of CLSID and InProcServer32 Keys: The creation and modification of various CLSID and InProcServer32 keys under both HKEY_CURRENT_USER_Classes and HKEY_LOCAL_MACHINE\SOFTWARE\Classes are indicative of attempts to register new COM objects or hijack existing ones. This can be used for executing arbitrary code, intercepting system calls, or creating backdoors.

Interference with System Components and Libraries: The listing of Image File Execution Options for various critical system libraries and executables (like KERNEL32.dll, USER32.dll, ntdll.dll) points towards attempts to interfere with or monitor the execution of these components. This could be for injecting code, monitoring system activity, or hijacking system processes.

Registry-Based Network and Security Configuration Changes: The presence of keys under HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Cryptography and similar paths suggests that the malware may be trying to alter network and security configurations, potentially to facilitate data exfiltration, create covert channels, or compromise cryptographic functions.

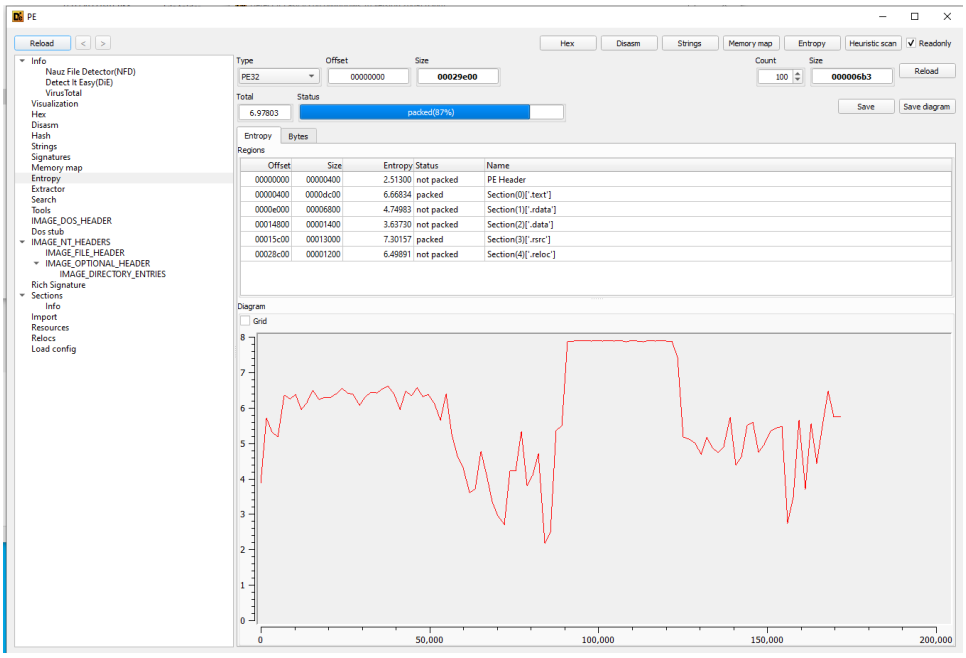
Creation of Proxy and Handler Entries: The presence of numerous ProxyStubClsid32 and InprocHandler entries under various Interface keys in HKEY_CURRENT_USER_Classes could indicate the malware's attempt to intercept or manipulate COM interface calls, which are often used for inter-process communication in Windows.

Elevation of Privilege Indicators: Multiple entries related to Elevation under CLSID keys imply attempts to gain higher privileges, possibly to bypass User Account Control (UAC) or to execute actions that normally require administrative privileges.

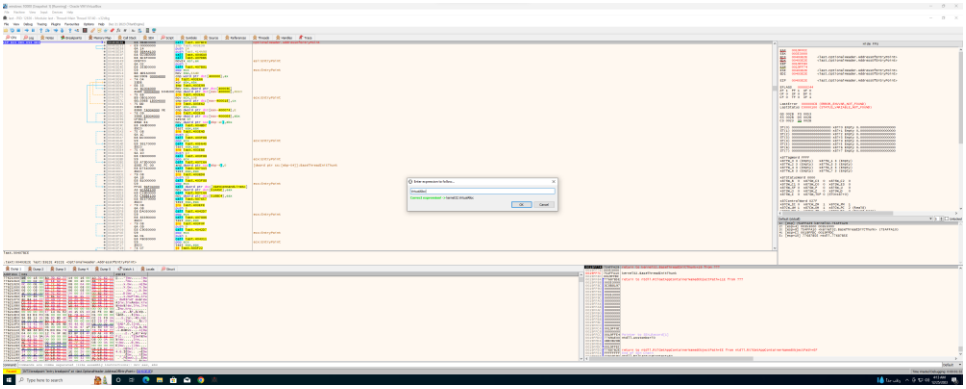
Mount Points and Drive Settings: The use of HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2 suggests manipulation of drive mount points, which could be used to execute malware when certain drives or removable media are accessed.

Unpacking the Malware:

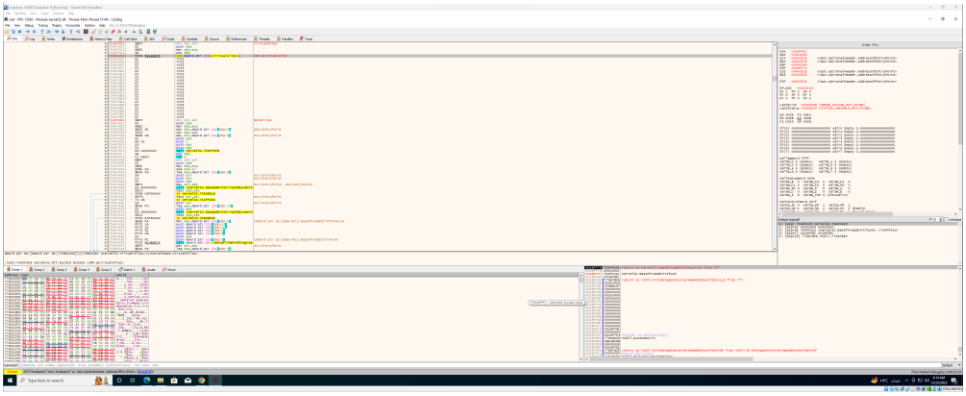
This picture shows how the packed virus looks like in DIE.



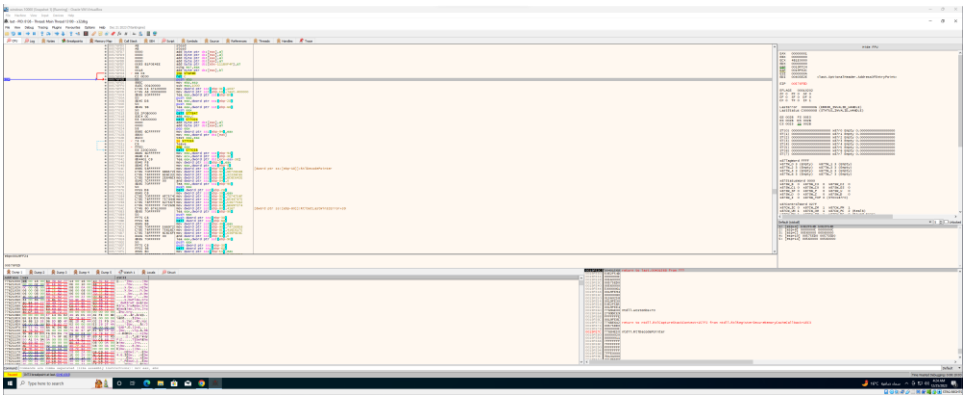
First of all we run the malware once in x32dbg and we find the first entry point and stop. After this we Search for the Virtual Alloc



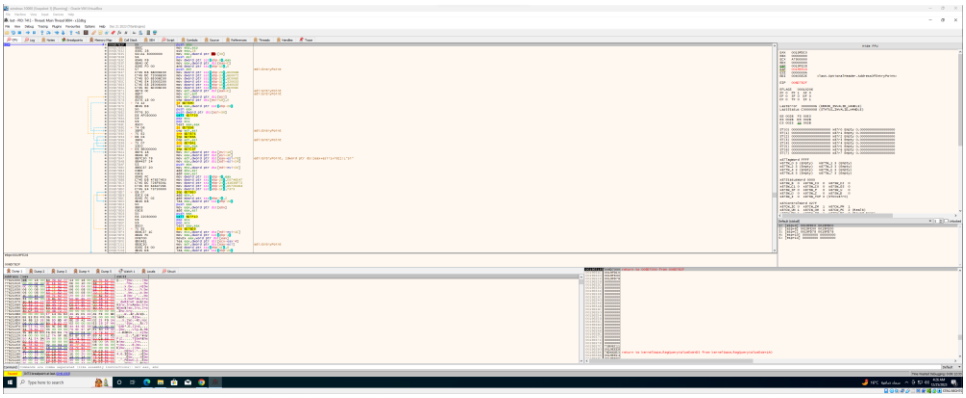
Now inside the Virtuall alloc we go to the first jump and follow it:



The jump leads us to another jmp and we enter the push ebp

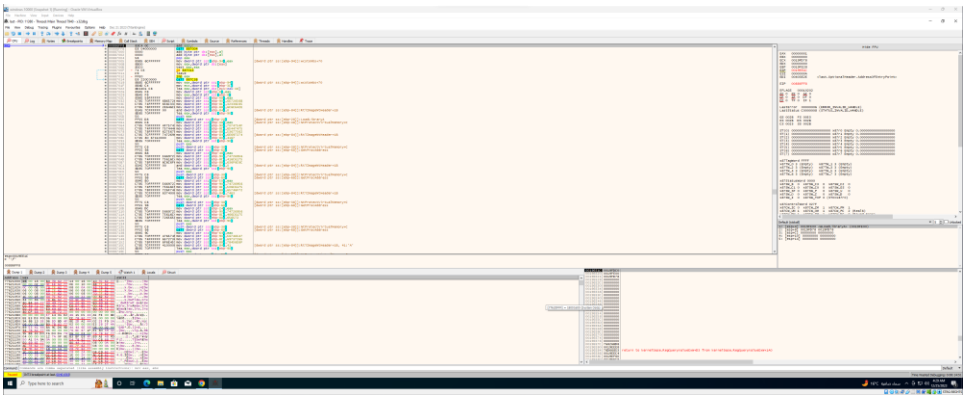


I explored the push ebp a bit but didn’t find anything interesting so I pressed on the button run till return

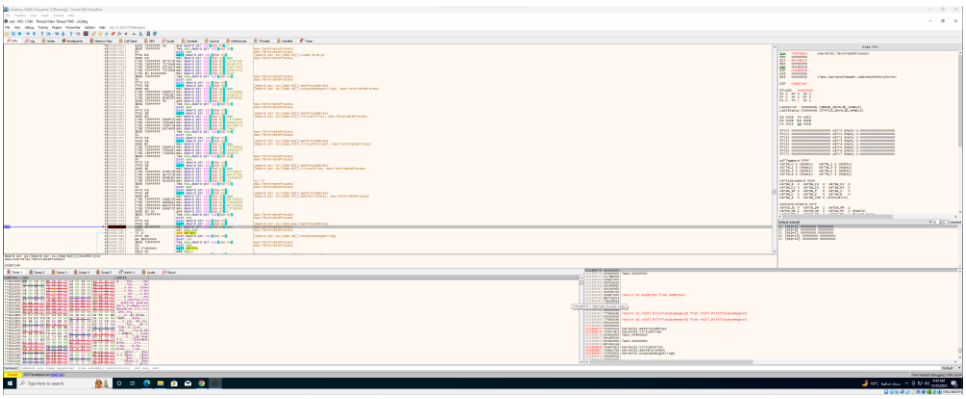


After going to the return function we jump into it.

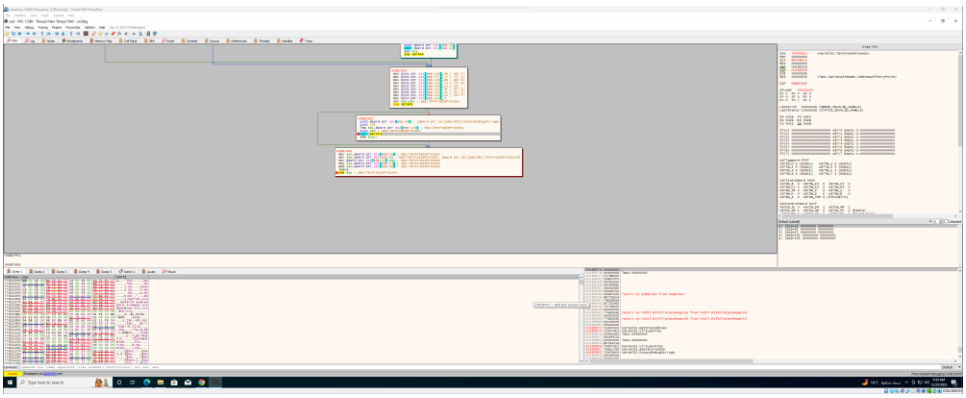
Now inside the return function we can explore the dynamic apis. So we move further down using the step into function to load the api's



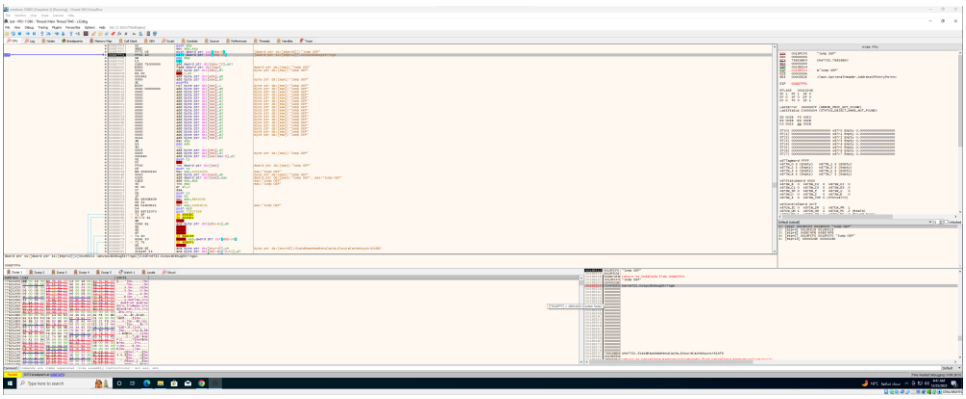
Now seeing the relabeled apis with their proper names.



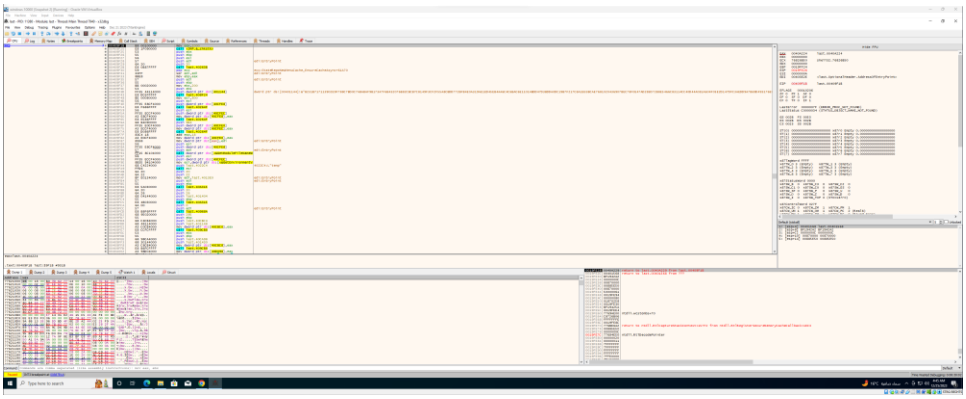
After seeing the APIs I jumped into the graph view of the push ebp function and went down to the bottom and found the Jmp eax and a call.



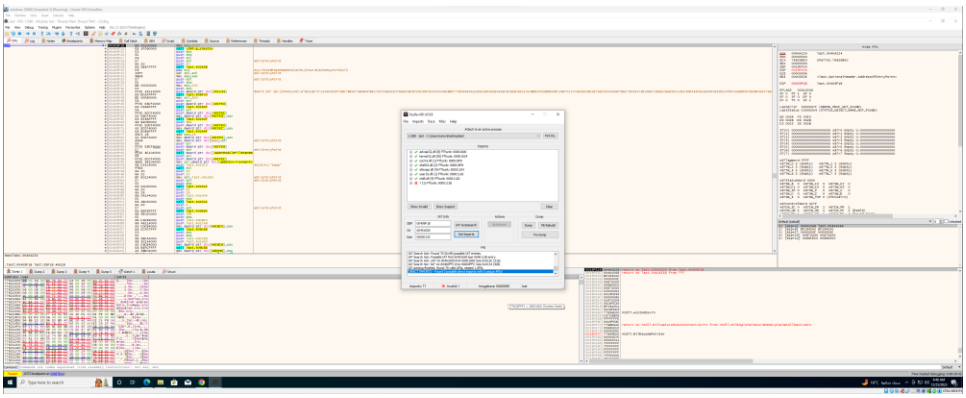
Now when we run the code we can see debug messages appear when entering the call function further.



Now when we jump into it we can find the original entry point of the malware. So, I opened Scylla and inputed the entry point.



After inputing the entry point I removed the Fthunk and dumped the file and fixed it again using Scylla.



And here is the result of the unpacked Malware. All the functionalities still persist after the unpacking so I ensured that the unpacking was successful