Title: **Dissecting Zeus: Malware Analysis Report of the 2013 Zbot Malware**

Investigator 1: Samantha Henry

Investigator 2: Rayyan Baig Mirza

Date: 4/29/2025

---

1. Summary

The Zeus (Zbot) 2013 Trojan malware is a 32-bit Windows executable that is packed with UPX, Themida, and NSPack. It employs self-modifying code and dynamic API resolution to avoid static detection and reverse engineering. Anti-VM checks for VMware were discovered and suggest that it attempts to circumvent sandbox analysis.

Static and dynamic analysis found Zeus stealing keystrokes (GetAsyncKeyState), clipboard contents (GetClipboardData), writing data to files (WriteFile), anti-debugging techniques (GetTickCount), and running additional commands (WinExec). These actions allow it to steal sensitive data such as online banking credentials and tighten its hold on compromised systems.

Beyond credential theft, Zeus acts like an intruder, adapting to the environment, hiding its tracks, and configuring infected machines for long-term exploitation. Its mixture of stealth, persistence, and flexible attack features made it one of the most successful malware families in financial cybercrime history.

Malware Sample Download:
https://github.com/ytisf/theZoo/tree/master/malware/Binaries/ZeusBankingVersion_26Nov2013

Password: infected

---

2. Identification

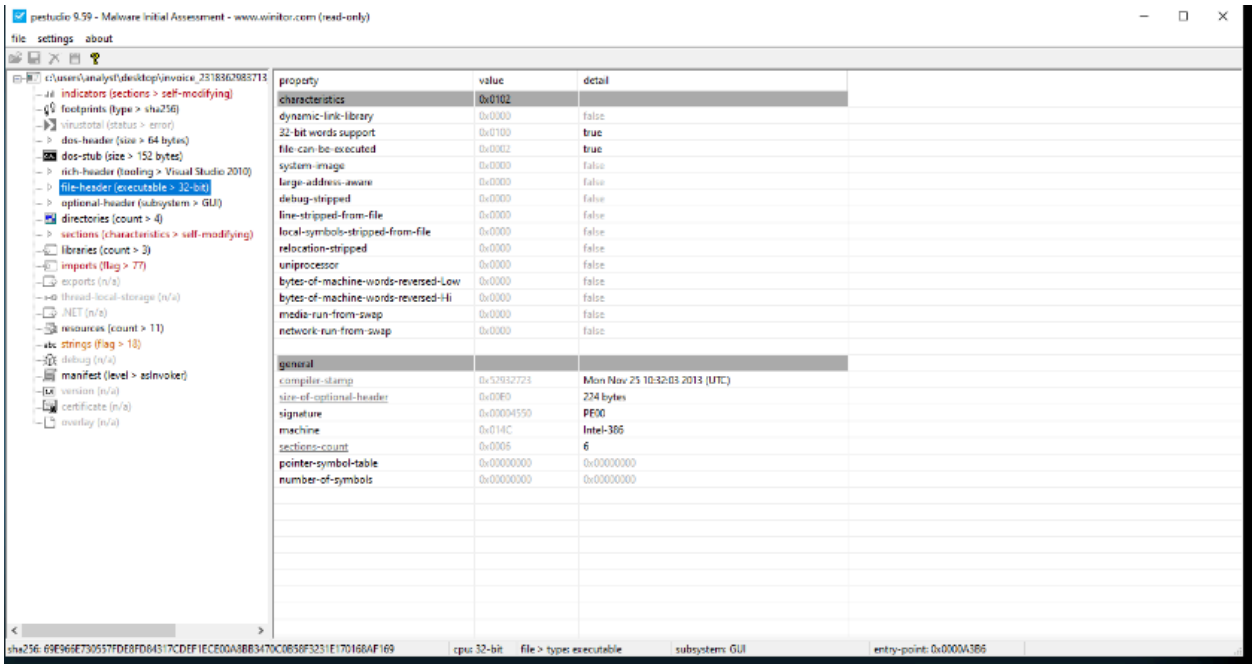
2.1) Filename: ZeusBankingVersion_26Nov2013

File size: 247.00 KB (252928 bytes)

File type: Win32 EXE, PE32 executable (GUI) Intel 80386, for MS Windows

2.2) MAC timestamps

The MAC (Modified, Accessed, Created) timestamps were extracted from the PE header of the sample.



The TimeDateStamp value 0x52932273 was manually decoded to confirm the following:

2013-11-25 13:29:23

2.3) Hashes

MD5: ea039a854d20d7734c5add48f1a51c34

SHA-1: 9615dca4c0e46b8a39de5428af7db060399230b2

SHA-256: 69e966e730557fde8fd84317cdef1ece00a8bb3470c0b58f3231e170168af169

SSDEEP:
6144:Tz/LBBTHT+7oEf2ZstxQMSGToLoOhD2saLsW8fsmFBkObjD:PLBdy7FpQMlToThD+sW8fsmP7bj fuzzy)

2.4) Signing Information (Certificates): N/A

2.5) Packer information

UPX (Ultimate Packer for Executables)
Themida
NSPack.

2.6) Aliases

AhnLab-V3 - Trojan/Win32.ZAccess.R87034
AliCloud - Backdoor: Win/ZAccess.emkb
K7AntiVirus - RootKit (004b9ee21)

Microsoft - PWS:Win32/Zbot or Trojan:Win32/Zbot

2.7) Dependencies

The ZeusBankingVersion_26Nov2013 malware sample relies on several Windows libraries (DLLs) to perform its malicious activities. Based on static analysis in PEStudio and IDA Pro, the following dependencies were identified:

GetAsyncKeyState (user32.dll): Capture key presses (keylogging)

GetClipboardData (user32.dll): Steal clipboard data

CallWindowProcW (user32.dll): Interact with GUI windows

CreateFileMappingA (kernel32.dll): Map files into memory

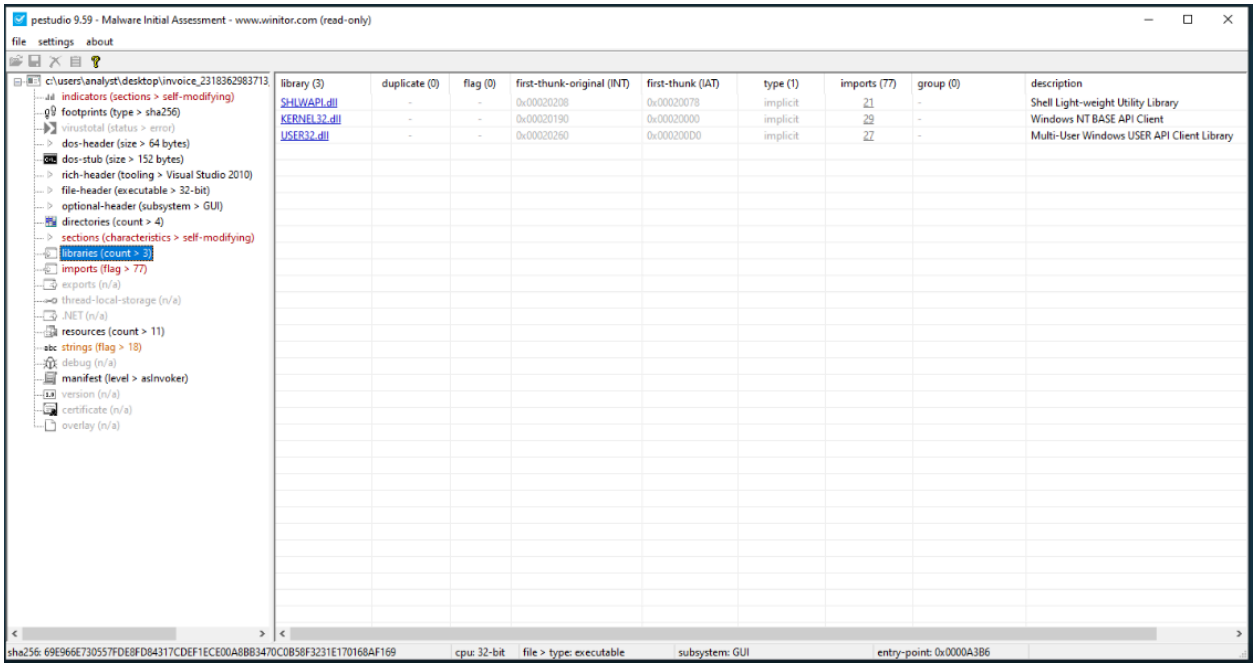
GetEnvironmentVariableA (kernel32.dll): Read system environment info

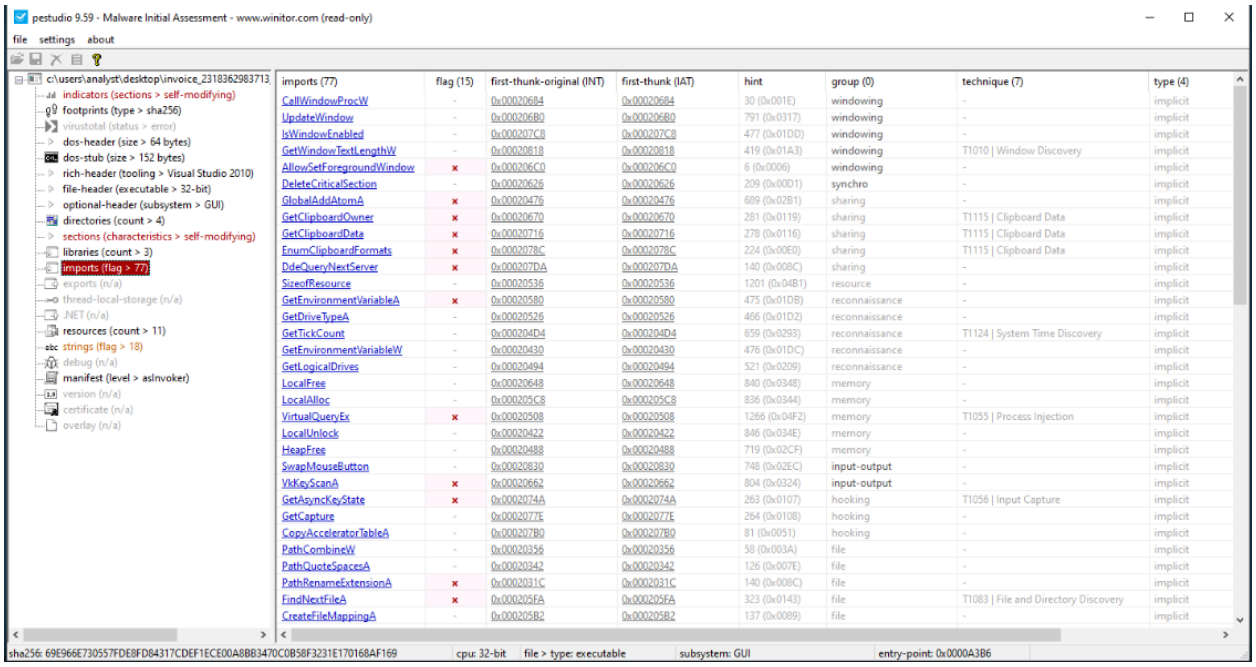
LocalAlloc / LocalFree (kernel32.dll): Allocate or free memory

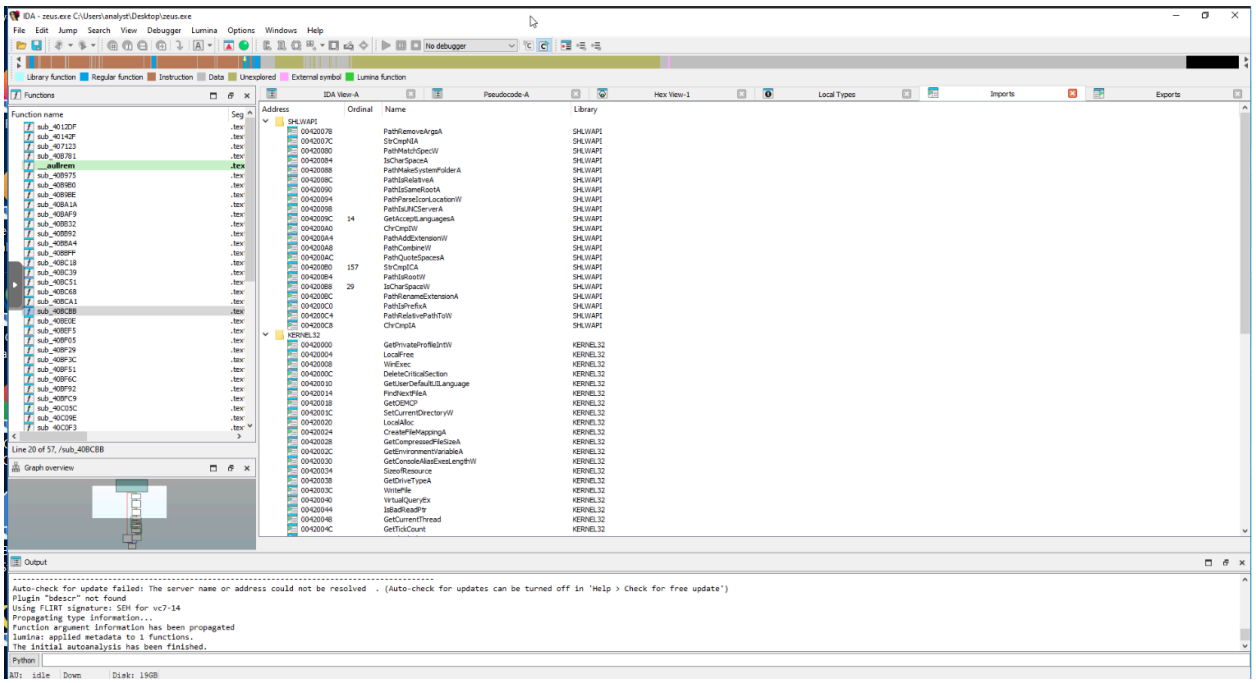
FindNextFileW (kernel32.dll): Enumerate files/folders

GetDriveTypeA (kernel32.dll): Detect connected drives

These dependencies highlight the malware's focus on stealing information through keylogging and clipboard capturing, file system interaction, and system reconnaissance.




pestudio 9.59 - Malware Initial Assessment - www.winitor.com (read-only)
file settings about
c:\users\analyst\desktop\invoice_2318362983713
indicators (sections > self-modifying)
footprints (type > sha256)
virustotal (status > error)
dos-header (size > 64 bytes)
dos-stub (size > 152 bytes)
rich-header (tooling > Visual Studio 2010)
file-header (executable > 32-bit)
optional-header (subsystem > GUI)
directories (count > 4)
sections (characteristics > self-modifying)
libraries (count > 3)
imports (flag > 77)
exports (n/a)
thread-local-storage (n/a)
.NET (n/a)
resources (count > 11)
strings (flag > 18)
debug (n/a)
manifest (level > asInvoker)
version (n/a)
certificate (n/a)
overlay (n/a)
imports (77) | flag (15) | first-thunk-original (INT) | first-thunk (IAT) | hint | group (0) | technique (7) | type (4)
CallWindowProcW | 0x00020684 | 0x00020684 | 30 (0x001E) | windowing | implicit
UpdateWindow | 0x000206B0 | 0x000206B0 | 791 (0x0317) | windowing | implicit
IsWindowEnabled | 0x000207C8 | 0x000207C8 | 477 (0x01DD) | windowing | implicit
GetWindowTextLengthW | 0x00020818 | 0x00020818 | 419 (0x01A3) | windowing | T1010 | Window Discovery | implicit
AllowSetForegroundWindow | 0x000206C0 | 0x000206C0 | 6 (0x0006) | windowing | implicit
DeleteCriticalSection | 0x00020626 | 0x00020626 | 209 (0x00D1) | synchro | implicit
GlobalAddAtomA | 0x00020476 | 0x00020476 | 689 (0x02B1) | sharing | implicit
GetClipboardOwner | 0x00020670 | 0x00020670 | 281 (0x0119) | sharing | T1115 | Clipboard Data | implicit
GetClipboardData | 0x00020716 | 0x00020716 | 278 (0x0116) | sharing | T1115 | Clipboard Data | implicit
EnumClipboardFormats | 0x0002078C | 0x0002078C | 224 (0x00E0) | sharing | T1115 | Clipboard Data | implicit
DdeQueryNextServer | 0x000207DA | 0x000207DA | 140 (0x008C) | sharing | implicit
SizeofResource | 0x00020536 | 0x00020536 | 1201 (0x04B1) | resource | implicit
GetEnvironmentVariableA | 0x00020580 | 0x00020580 | 475 (0x01DB) | reconnaissance | implicit
GetDriveTypeA | 0x00020526 | 0x00020526 | 466 (0x01D2) | reconnaissance | implicit
GetTickCount | 0x000204D4 | 0x000204D4 | 659 (0x0293) | reconnaissance | T1124 | System Time Discovery | implicit
GetEnvironmentVariableW | 0x00020430 | 0x00020430 | 476 (0x01DC) | reconnaissance | implicit
GetLogicalDrives | 0x00020494 | 0x00020494 | 521 (0x0209) | reconnaissance | implicit
LocalFree | 0x00020648 | 0x00020648 | 840 (0x0348) | memory | implicit
LocalAlloc | 0x000205C8 | 0x000205C8 | 836 (0x0344) | memory | implicit
VirtualQueryEx | 0x00020508 | 0x00020508 | 1266 (0x04F2) | memory | T1055 | Process Injection | implicit
LocalUnlock | 0x00020422 | 0x00020422 | 846 (0x034E) | memory | implicit
HeapFree | 0x00020488 | 0x00020488 | 719 (0x02CF) | memory | implicit
SwapMouseButton | 0x00020830 | 0x00020830 | 748 (0x02EC) | input-output | implicit
VkKeyScanA | 0x00020662 | 0x00020662 | 804 (0x0324) | input-output | implicit
GetAsyncKeyState | 0x0002074A | 0x0002074A | 263 (0x0107) | hooking | T1056 | Input Capture | implicit
GetCapture | 0x0002077E | 0x0002077E | 264 (0x0108) | hooking | implicit
CopyAcceleratorTableA | 0x000207B0 | 0x000207B0 | 81 (0x0051) | hooking | implicit
PathCombineW | 0x00020356 | 0x00020356 | 58 (0x003A) | file | implicit
PathQuoteSpacesA | 0x00020342 | 0x00020342 | 126 (0x007E) | file | implicit
PathRenameExtensionA | 0x0002031C | 0x0002031C | 140 (0x008C) | file | implicit
FindNextFileA | 0x000205FA | 0x000205FA | 323 (0x0143) | file | T1083 | File and Directory Discovery | implicit
CreateFileMappingA | 0x000205B2 | 0x000205B2 | 137 (0x0089) | file | implicit
sha256: 69E966E730557FDE8FD84317CDEF1ECE00A8BB3470C0B58F3231E170168AF169
cpu: 32-bit
file > type: executable
subsystem: GUI
entry-point: 0x0000A3B6


IDA - zeus.exe C:\Users\analyst\Desktop\zeus.exe
File Edit Jump Search View Debugger Lumina Options Windows Help
Functions
IDA View-A
Pseudocode-A
Hex View-1
Local Types
Imports
Exports
Graph overview
Output
Auto-check for update failed: The server name or address could not be resolved . (Auto-check for updates can be turned off in 'Help > Check for free update')
Plugin "bdescr" not found
Using FLIRT signature: SEH for vc7-14
Propagating type information...
Function argument information has been propagated
lumina: applied metadata to 1 functions.
The initial autoanalysis has been finished.
Python
AU: idle Down Disk: 19GB

3. Capabilities & Characteristics

Escalate User Privilege [SO]
Zeus displays a fake Adobe Flash Player UAC prompt to trick users into granting administrative privileges.

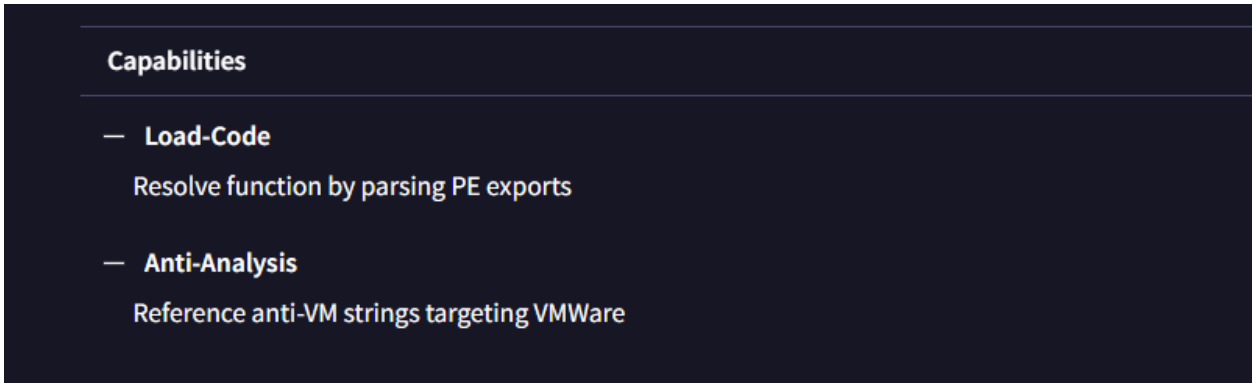
Capture Keyboard Input [TO]
Zeus captures keystrokes using GetAsyncKeyState to monitor user activity and steal confidential information.

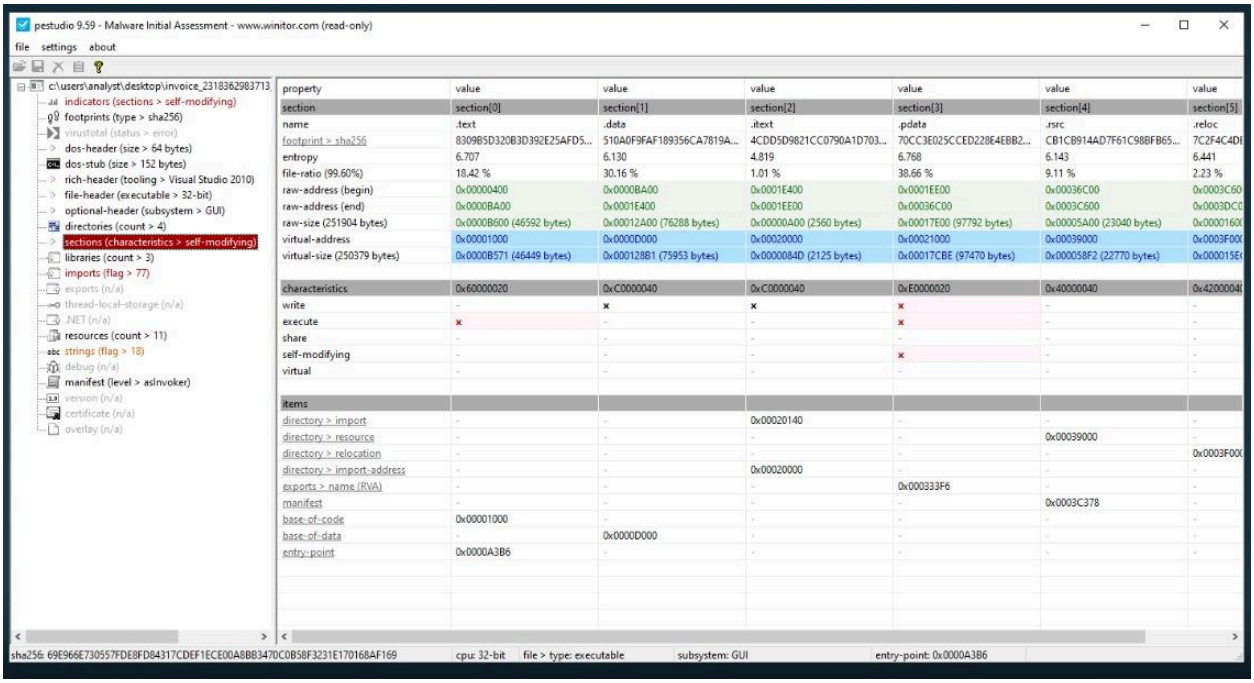
Self-Modification [SO]
Zeus modifies its own code at runtime via writable and executable sections to avoid detection.

Obfuscate Imports [TO]
Zeus dynamically resolves APIs (e.g., GetProcAddress) instead of using static imports to obstruct static analysis.

Detect VM Environment [TO]
Zeus checks for virtualized environments such as VMware and avoids execution in sandboxes.

Persist After System Reboot [TO]
Zeus creates a registry key under HKCU\Software\Microsoft\Windows\CurrentVersion\Run to automatically re-execute after reboot.





---

4. Dependencies

GetAsyncKeyState (user32.dll): Capture key presses (keylogging)

GetClipboardData (user32.dll): Steal clipboard data

CallWindowProcW (user32.dll): Interact with GUI windows

CreateFileMappingA (kernel32.dll): Map files into memory

GetEnvironmentVariableA (kernel32.dll): Read system environment info

LocalAlloc / LocalFree (kernel32.dll): Allocate or free memory

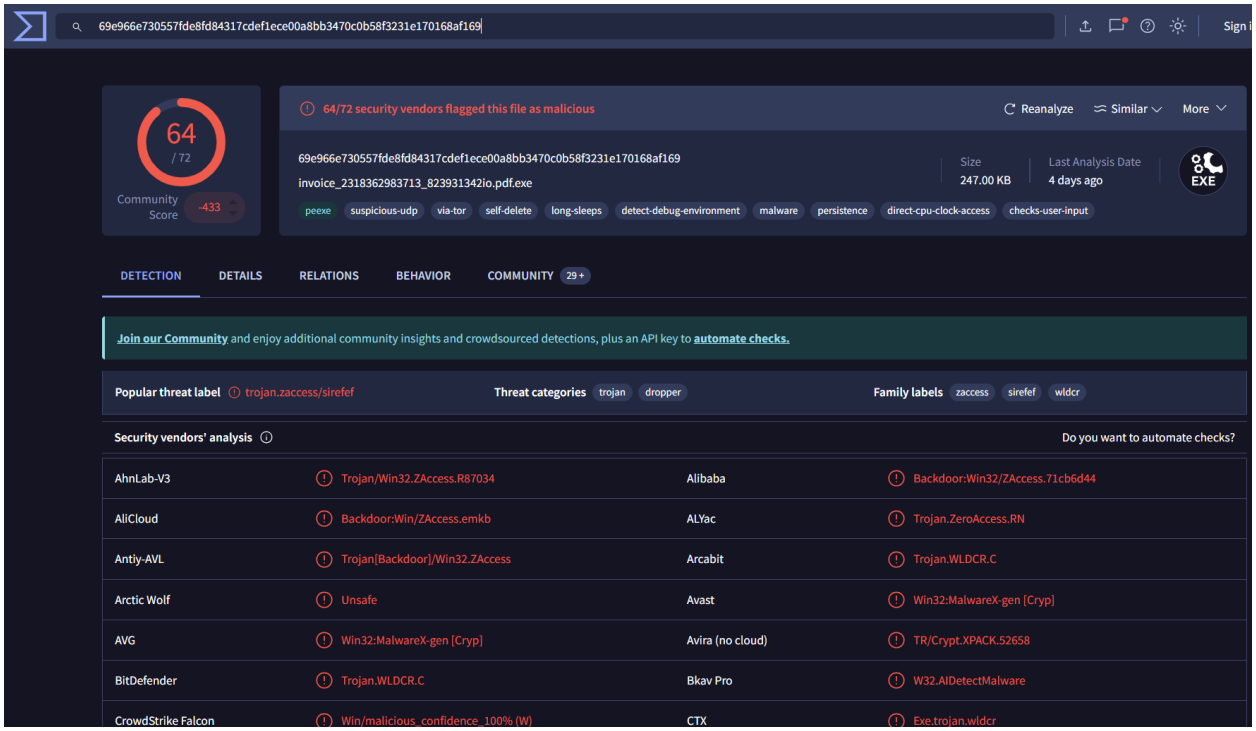
FindNextFileW (kernel32.dll): Enumerate files/folders

GetDriveTypeA (kernel32.dll): Detect connected drives

---

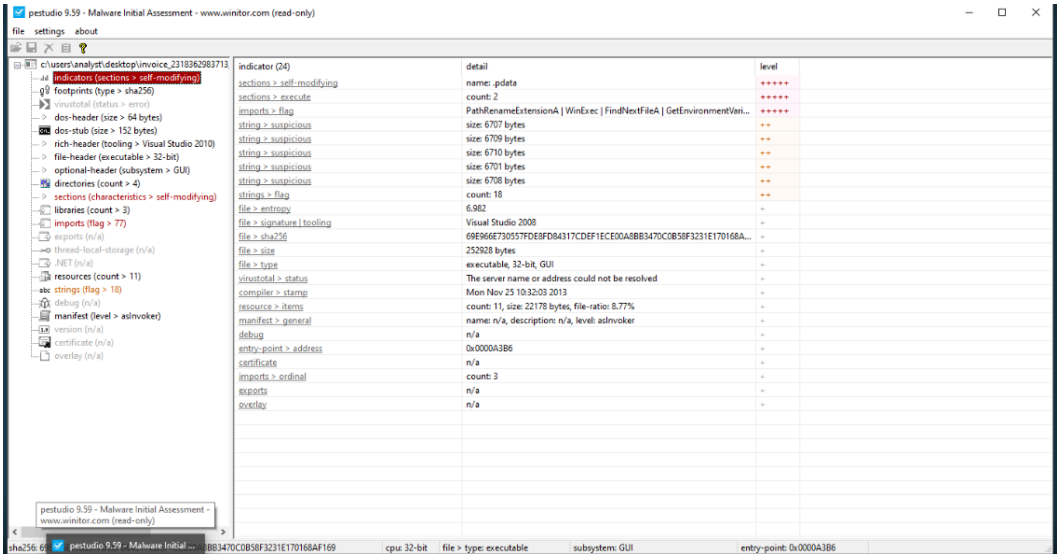
5. Static Analysis

This screenshot shows the VirusTotal scan results for the Zeus sample. Out of 72 antivirus engines, 64 detected the file as malicious, labeling it with names like Trojan.ZAccess and Trojan.WLDCR. Based on the threat categories listed (Trojan, dropper), it confirms that the sample is likely used for stealing data and possibly dropping malware.



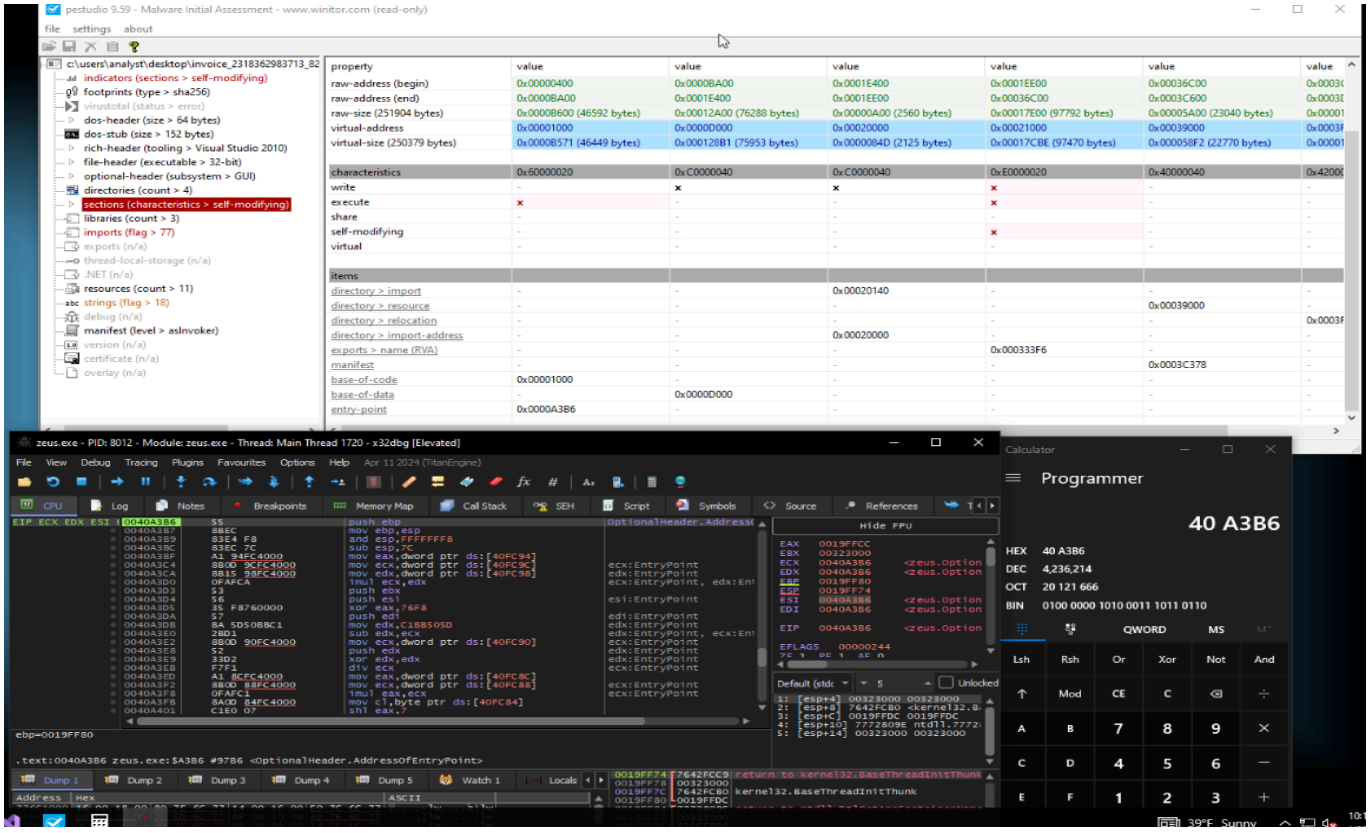
5.1) Top-level components

The sample is a 32-bit Windows PE32 Intel 80386 targeted executable. The main sections, such as .text, .data, .rsrc, .pdata, and .reloc were discovered. The .itext section contains write and execute permissions, which is a sign of self-modifying code behavior. Sections such as .pdata and .rsrc contained high entropy values (~6.7), which is an indication of compression or encryption.



5.2) Execution points of entry

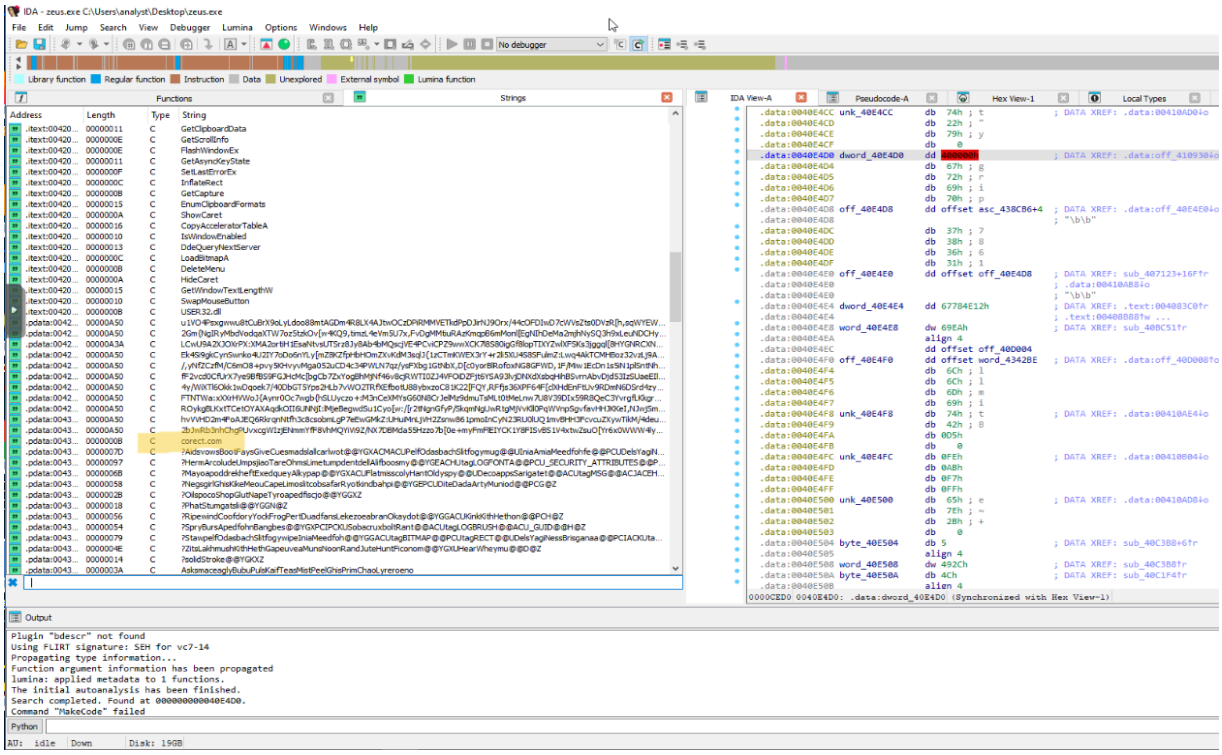
The sample entry point is seen at 0x0000A3B6 in both IDA Pro and PEStudio. There are instructions at the start that push and prepare registers, setting up stack frames, with multiple calls to dynamically resolve API functions, suggesting obfuscation at runtime.



5.3) Embedded strings

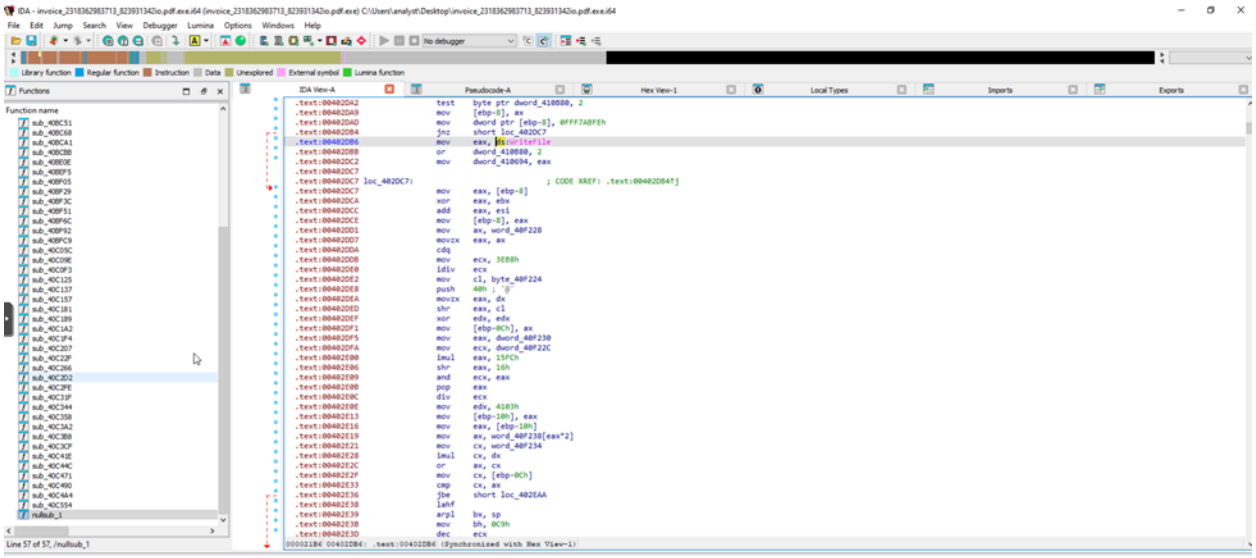
Numerous embedded strings were identified, where a few of them mapped directly to Windows API calls usually employed for malicious purposes. A few of the notable extracted strings included GetClipboardData, GetAsyncKeyState, CreateFileMappingA, GetEnvironmentVariableA, and EnumClipboardFormats, which point towards keylogging, spying on the clipboard, and file functions.

Additionally, the string corect.com was discovered within the sample. This could be a potential Command and Control (C2) domain.

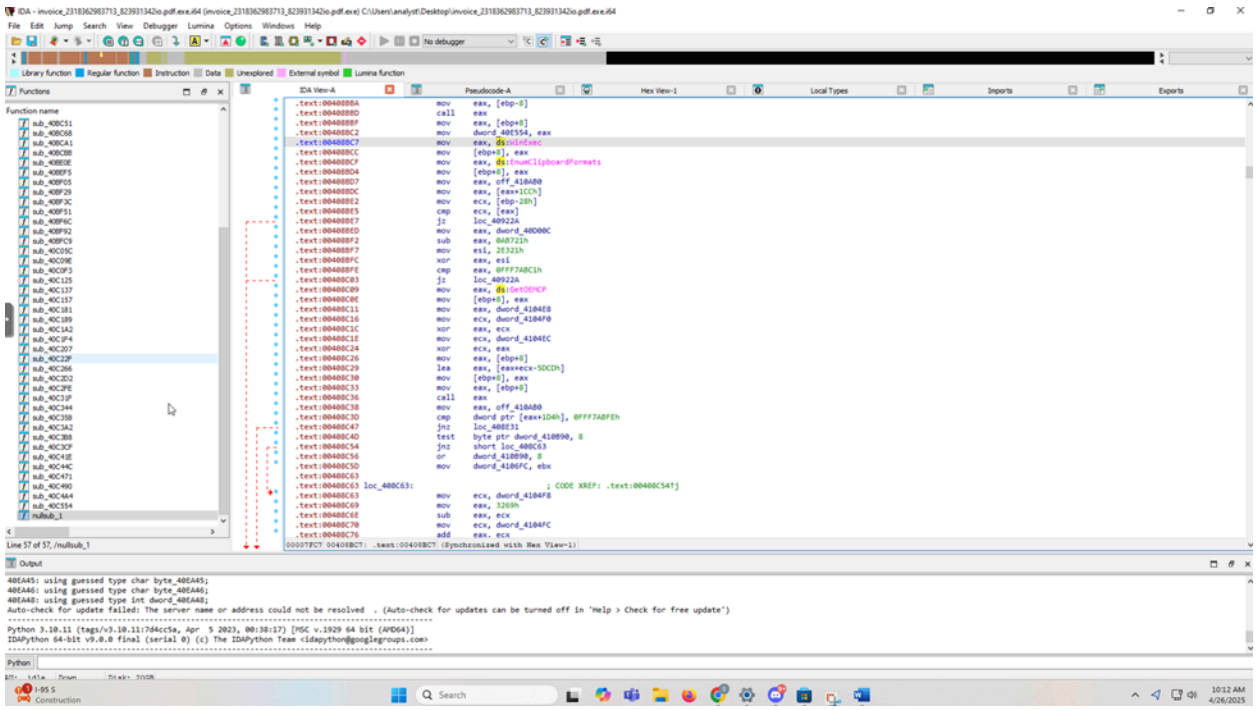


IDA Pro Assembly and Pseudocode

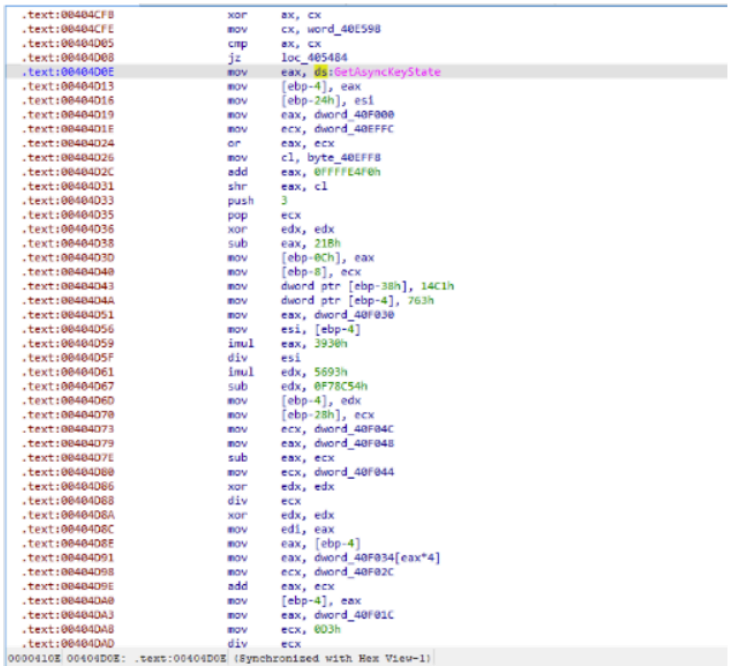
API Function: WriteFile
Description: Writes data to a file or input/output (I/O) device. Commonly used in malware to save stolen data or drop payloads.



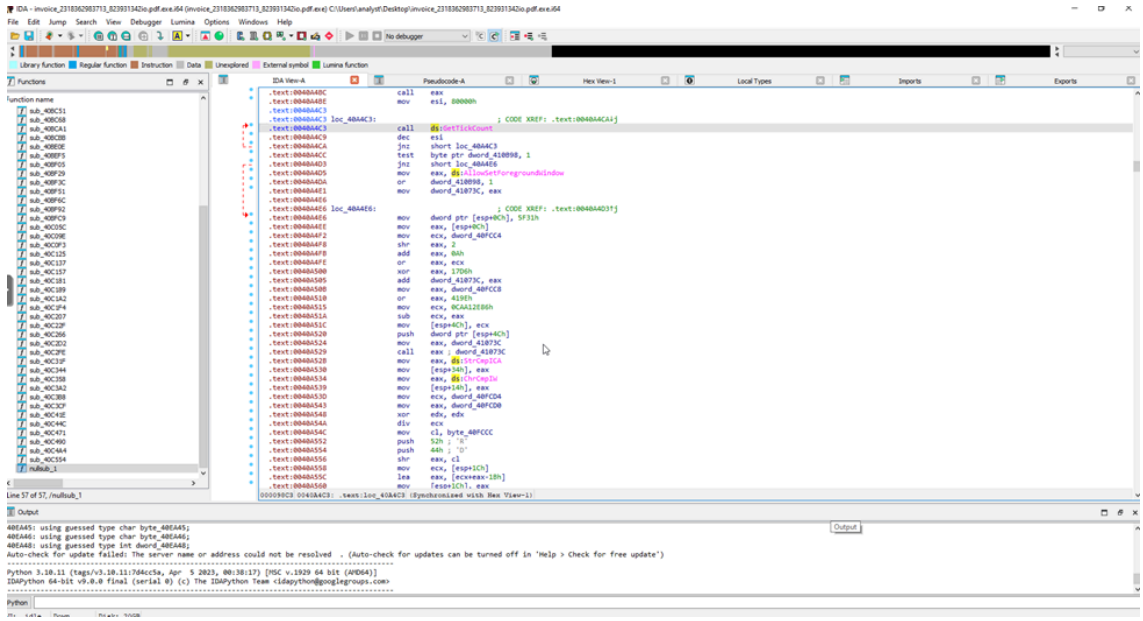
API Function: WinExec
Description: Allows execution of an application or a command. Commonly used in malware to silently launch programs or scripts.



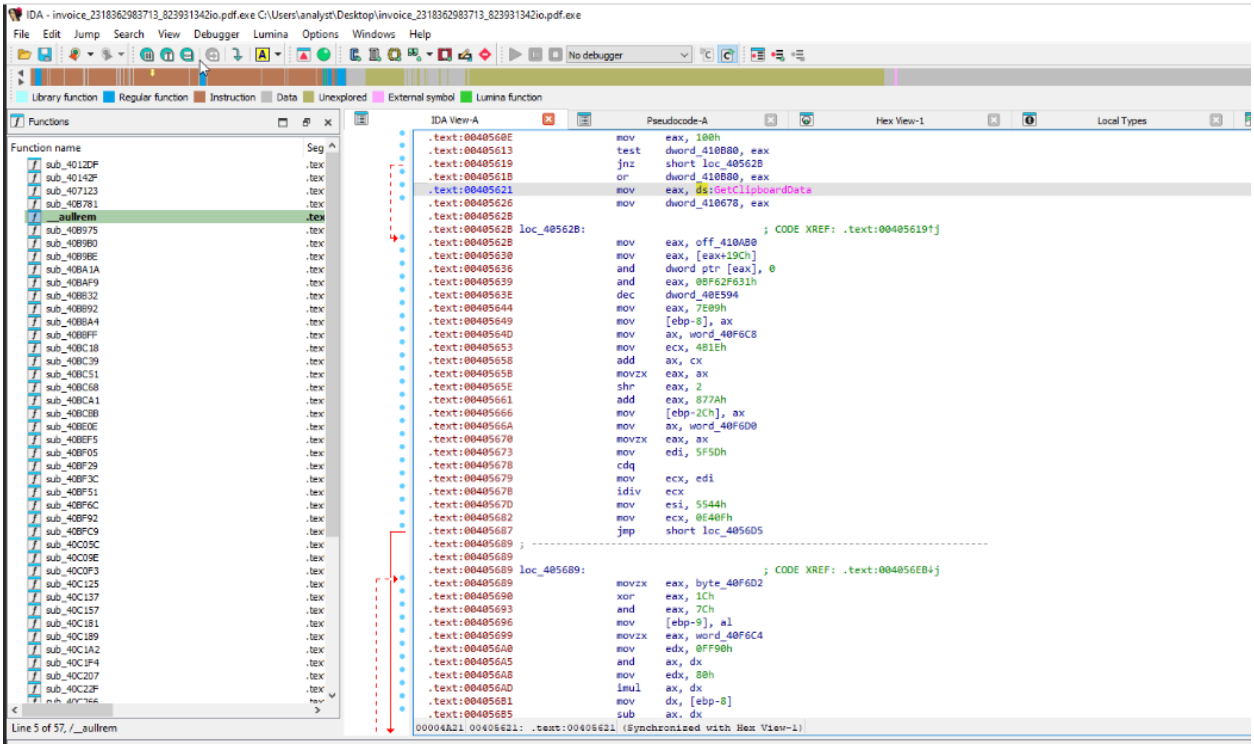
API Function: GetAsyncKeyState
Description: Checks if specific keys are being pressed. Commonly used in malware to capture keyboard input for a keylogger.



API Function: GetTickCount
Description: Retrieves the number of milliseconds since the system has started. Commonly used for anti-debugging to detect if the malware code is stepping through instructions, if the code is being analyzed slowly, or if there are abnormal delays in sandboxes or debuggers.

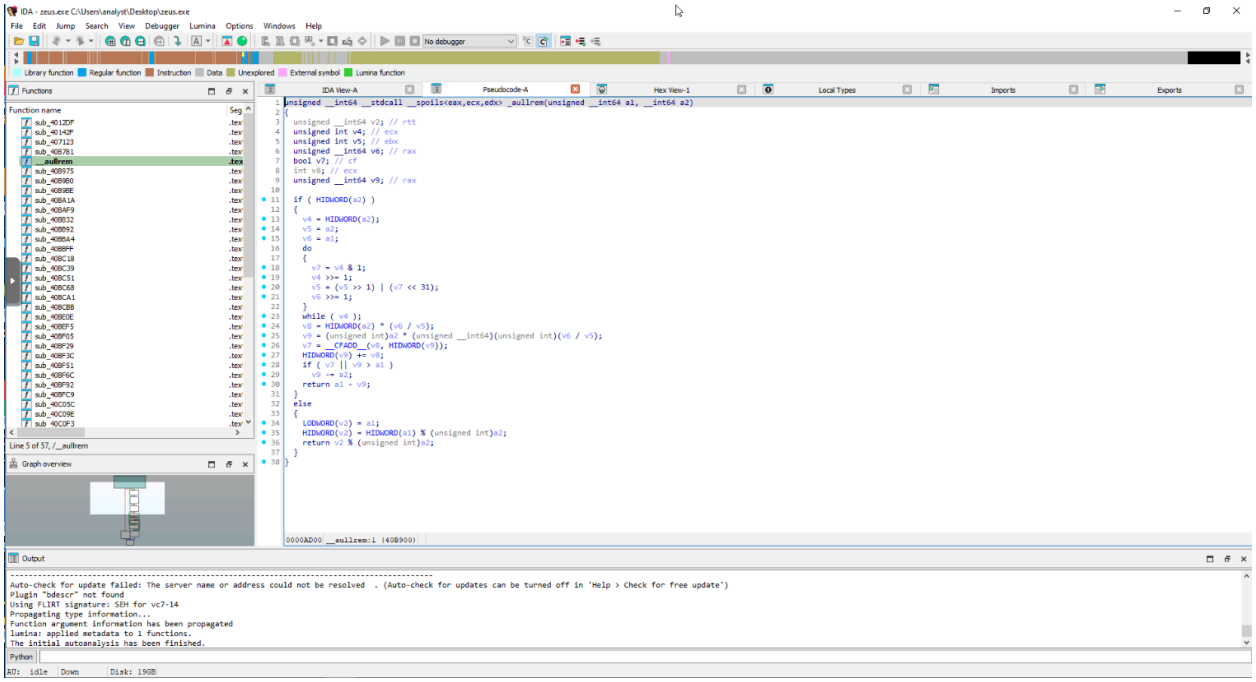


API Function: GetClipBoardData
Description: Retrieves data from the clipboard. Commonly used in malware to capture sensitive content.



IDA Pro Pseudocode

During the static analysis in IDA Pro, no decompiled pseudocode directly referenced the Windows APIs: WriteFile, WinExec, GetTickCount, GetAsyncKeyState, and GetClipboardData. These API functions were mainly visible in the disassembly view or the Imports table. However, the function _aulllen was observed performing bitwise operations (>>, <<, &) and XORs, which are likely associated with hashing, decryption, or obfuscation. This suggests the malware may be decrypting API names, payloads, strings, configuration data.



5.4) Code-related observations (Reflection, Obfuscation, Encryption, Native code, etc)

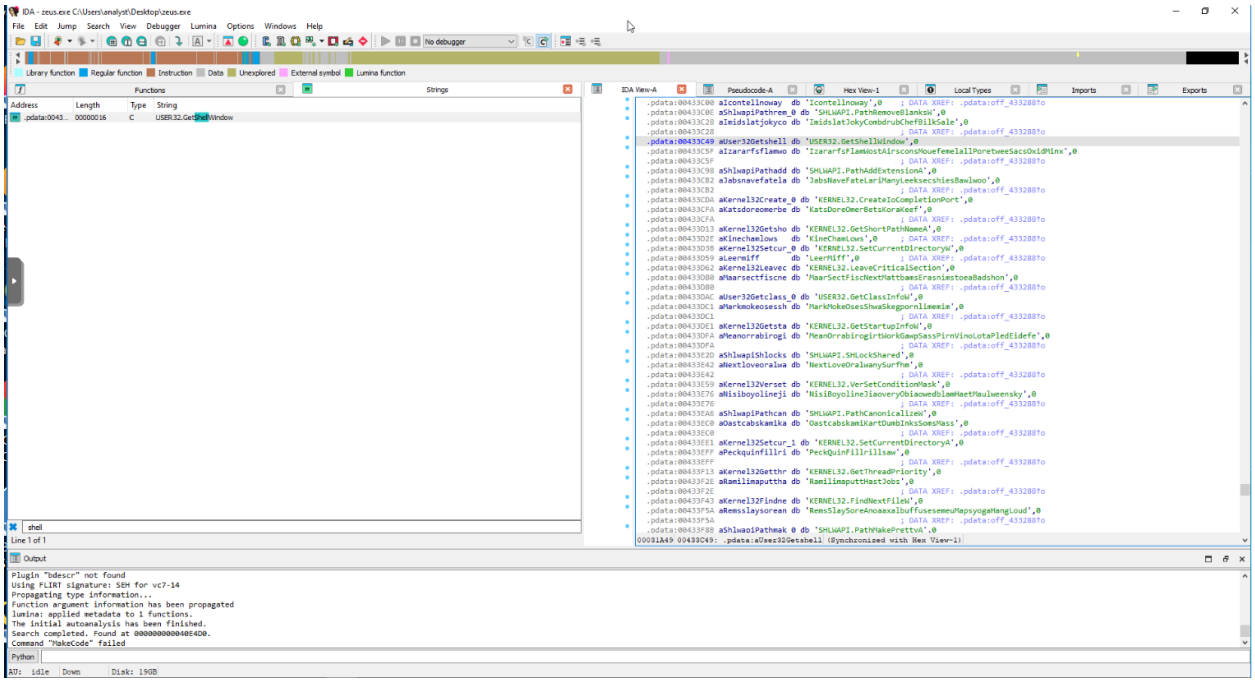
Dynamic API Resolution:
Functions are resolved at runtime through techniques like manually parsing PE exports instead of static imports (LoadLibrary, GetProcAddress style obfuscation).

Self-Modifying Code:
The .itext section is marked as writable and executable, enabling the malware to alter its code during execution to evade static analysis.

Anti-VM Techniques
The malware attempts to detect VMware environments, checking for specific registry keys and process names.

Entropy Analysis
Sections like .pdata and .rsrc have elevated entropy, suggesting packing, encryption, or data compression to obfuscate content.

Native Code Execution
Malware does not utilize a managed runtime (such as.NET); instead, it executes native 32-bit Windows instructions.



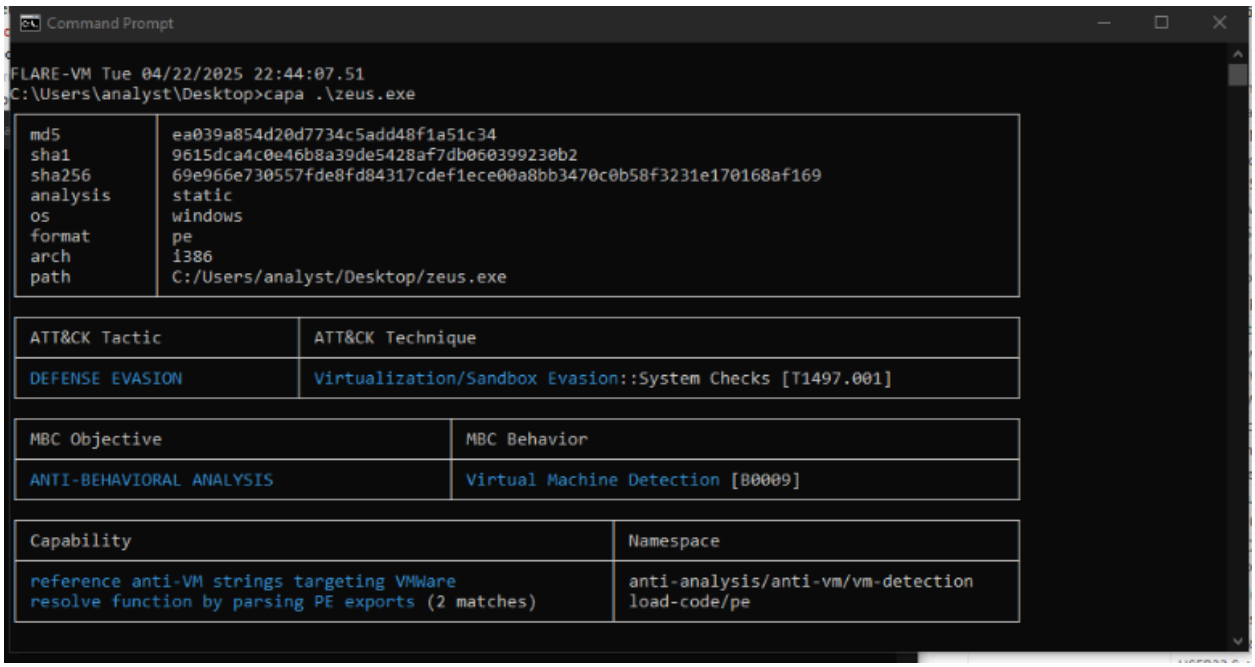
Command-line tools

The malware sample was analyzed using the capa tool on the FLARE-VM environment to extract behavioral capabilities and tactics quickly. The capa scan confirmed the following:

Anti-Analysis Behavior:
The sample contains functionality to detect if it is running inside a virtual machine environment, particularly VMware. This supports defense evasion and anti-behavioral analysis tactics.

Dynamic Code Loading:
Capa detected behavior associated with resolving function addresses at runtime by parsing PE exports. This matches manual findings from static analysis in IDA Pro and supports the malware's obfuscation and anti-static analysis capabilities.

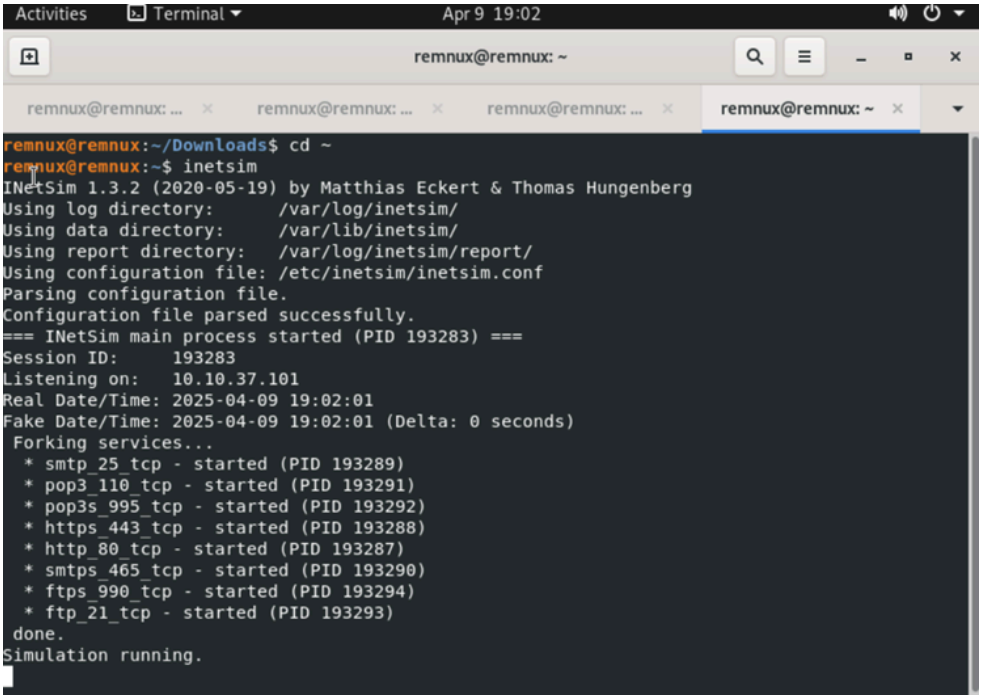
Mapping to MITRE ATT&CK and MBC:

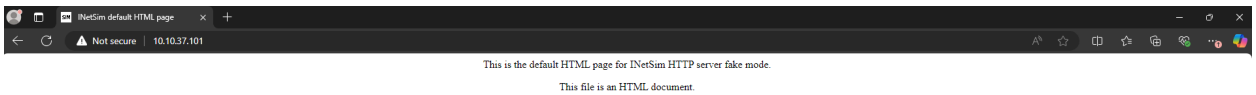


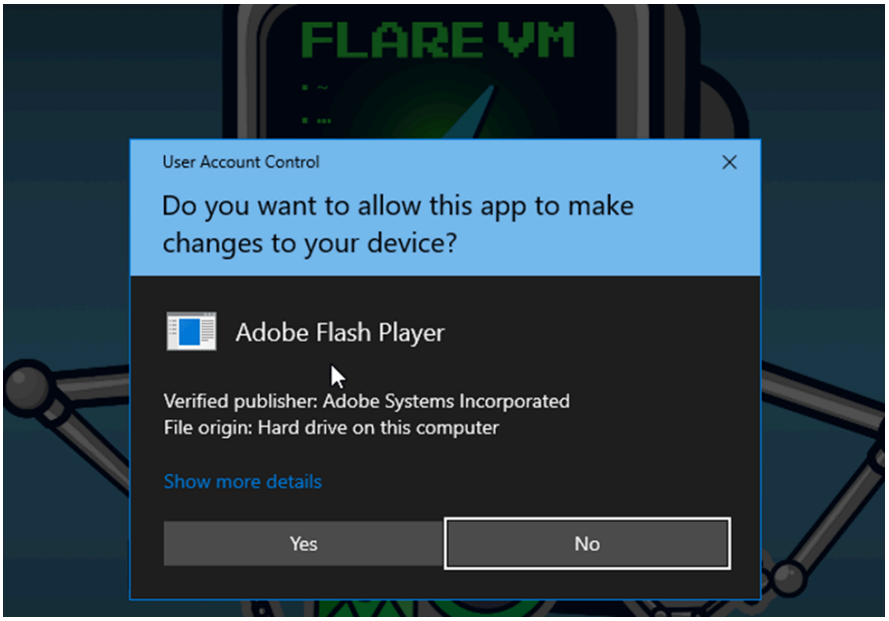
---

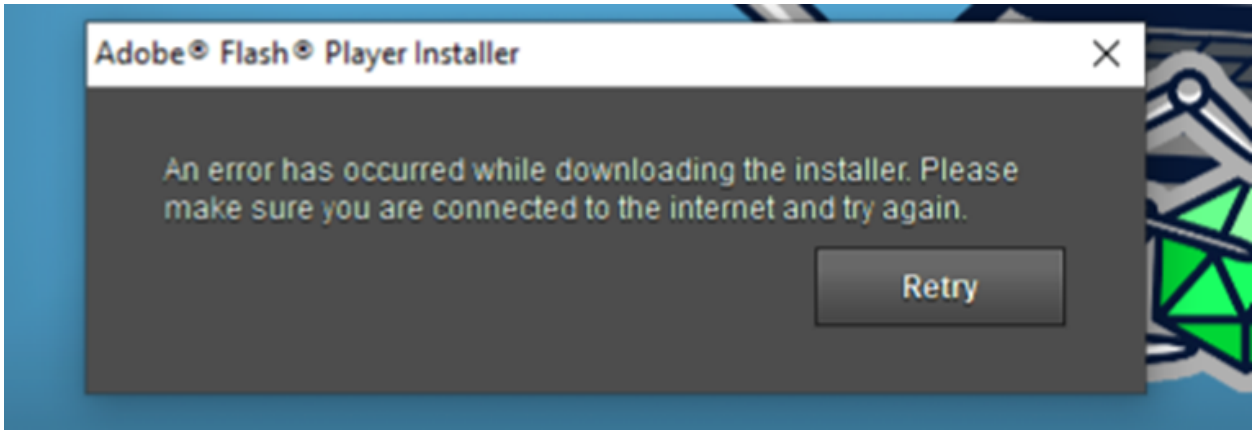
6. Dynamic Analysis

Before the execution of the Zeus Banking Trojan, InetSim was established on the REMnux virtual environment. In the Flare virtual environment, the Remnux IP address, ‘10.10.37.101,’ was entered into the Firefox browser to retrieve the INetSim default HTML page. Upon execution of the malware, a User Account Control (UAC) window for Adobe Flash Player appears on the screen. The malware prevents the analyst from selecting “No” or the exit button on the top right corner of the window, as the UAC prompt reappears. This suggests that the malware manipulated the UAC prompt to force the user to select the “Yes" button. Once the “Yes” button has been chosen, the malware is executed, and immediately an Adobe Flash Player Installer window appears with an error message stating, “An error has occurred while downloading the installer. Please make sure you are connected to the internet and try again.” There is a Retry button, but clicking it makes the prompt reappear. However, despite exiting the error message, the malware still executes in the background and deletes itself from the desktop.

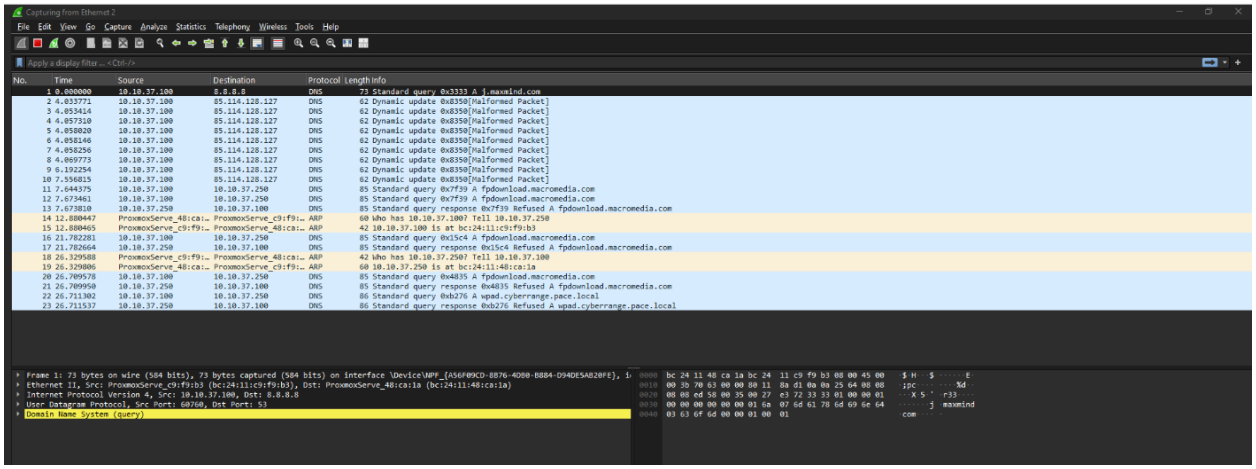

Activities
Terminal
Apr 9 19:02
remnux@remnux: ~
remnux@remnux: ...
remnux@remnux: ...
remnux@remnux: ...
remnux@remnux: ~
remnux@remnux:~/Downloads$ cd ~
remnux@remnux:~$ inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 193283) ===
Session ID: 193283
Listening on: 10.10.37.101
Real Date/Time: 2025-04-09 19:02:01
Fake Date/Time: 2025-04-09 19:02:01 (Delta: 0 seconds)
Forking services...
* smtp_25_tcp - started (PID 193289)
* pop3_110_tcp - started (PID 193291)
* pop3s_995_tcp - started (PID 193292)
* https_443_tcp - started (PID 193288)
* http_80_tcp - started (PID 193287)
* smtps_465_tcp - started (PID 193290)
* ftps_990_tcp - started (PID 193294)
* ftp_21_tcp - started (PID 193293)
done.
Simulation running.


INetSim default HTML page
Not secure
10.10.37.101
This is the default HTML page for INetSim HTTP server fake mode.
This file is an HTML document.


FLARE VM
User Account Control
Do you want to allow this app to make changes to your device?
Adobe Flash Player
Verified publisher: Adobe Systems Incorporated
File origin: Hard drive on this computer
Show more details
Yes
No



6.1) Network Traffic Analysis



6.1.1) DNS Queries

To determine if the malware communicates with command-and-control (C2) servers or if there is any suspicious network activity, the network traffic is examined in Wireshark. The captured traffic shows that there were a few DNS requests where the malware attempted to resolve to the domain fpdownload.macromedia.com. Through further research, it has been discovered that this domain is associated with legitimate Adobe Flash Player updates. This could have been the malware's attempt to test if the system has internet access.

6.1.2) HTTP Conversations

No HTTP conversations were established or captured during the malware's execution.

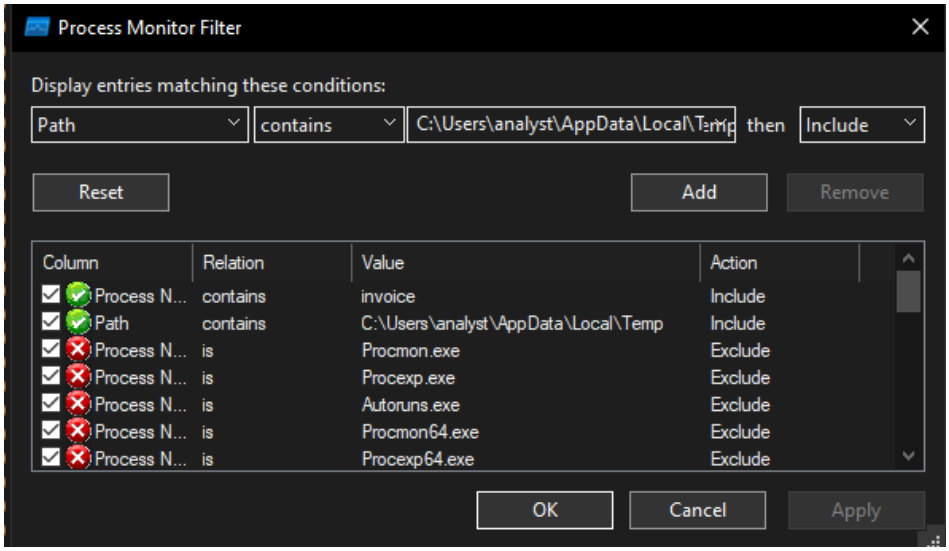
6.1.3) TCP/UDP Communication

There is no presence of TCP or UDP communication initiated by the malware. However, as expected, the DNS queries were transmitted over UDP port 53.

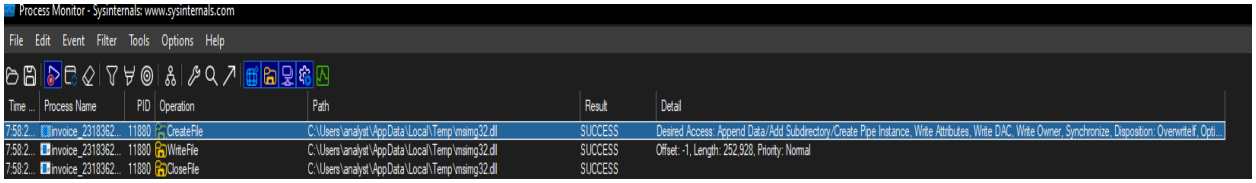
6.2) File operations (files read and written)

Using Process Monitor (Procmon), a set of filters is applied to narrow down the results:

- **Process Name contains 'invoice' then Include** - This ensures that operations performed by invoice_2318362983713_823931342io.pdf.exe are only logged.
- **Path contains C:\Users\analyst\AppData\Local\Temp then Include** - This captures if any operations were performed in the specified file path.

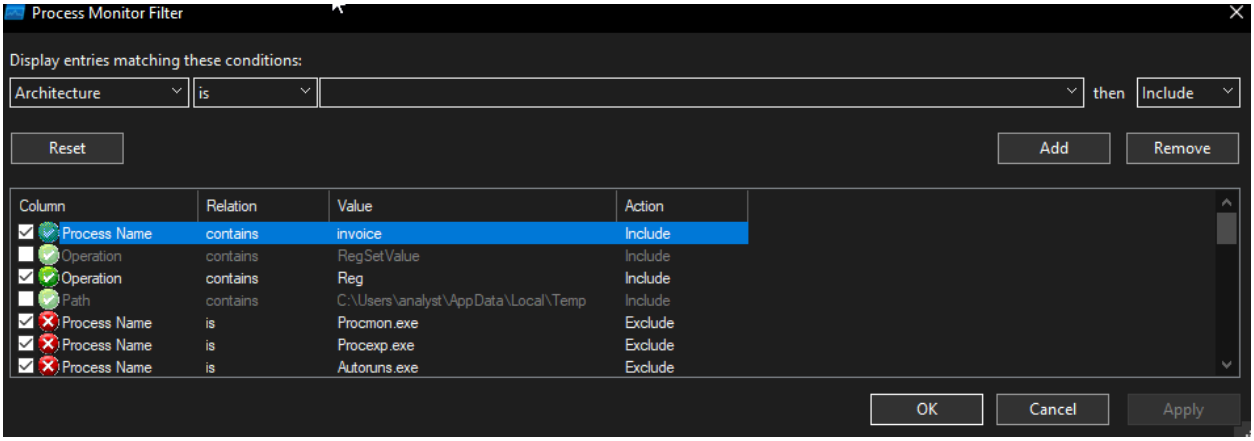


It was observed that the malware had successfully created and written to a DLL file named msimg32.dll in the file path C:\Users\analyst\AppData\Local\Temp. Three operations were conducted in this file path: CreateFile, WriteFile, and CloseFile. The malware may have dropped a suspicious payload in the Temp directory, as it is common for applications to store temporary files, making it difficult to distinguish between malicious and legitimate files.
In the Detail column of the CreateFile operation, it says: Desired Access: Append Data/Add Subdirectory/Create Pipe Instance, Write Attributes, Write DAC, Write Owner, Synchronize, Disposition: OverwriteIf, Options: . It is suspected that the malware intended to create the DLL file to append data, add a subdirectory, and set file attributes and permissions. If the file already existed, it should be overwritten.

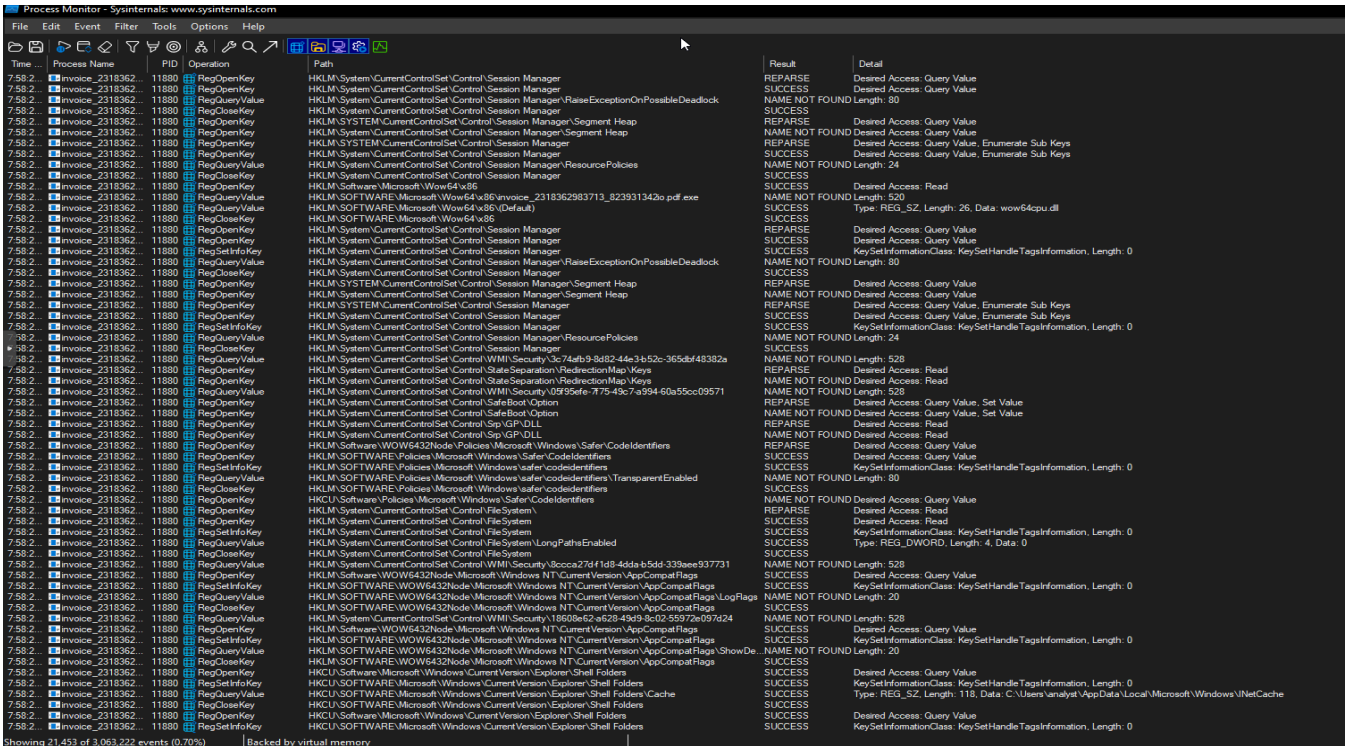


An additional filter was configured to inspect registry activity:

Operation contains Reg then Include - Provide the events that contain the word “Reg”.

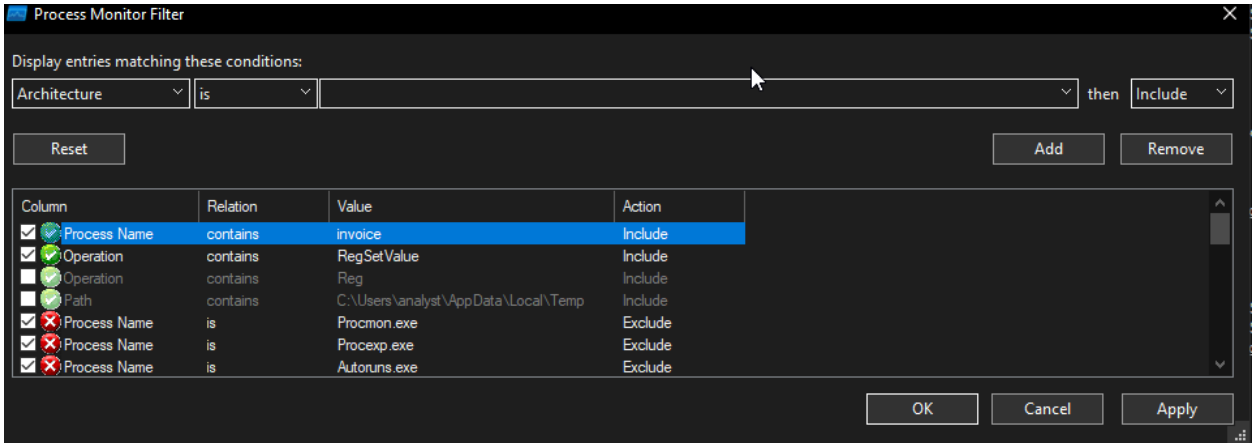


Procmon captured the registry-related events with operations RegOpenKey, RegSetValue, RegQueryValue, RegSetInfoKey, and RegCloseKey. The malware interacted with multiple registry keys, primarily under the HKLM\SYSTEM and the HKCU\SOFTWARE hives.

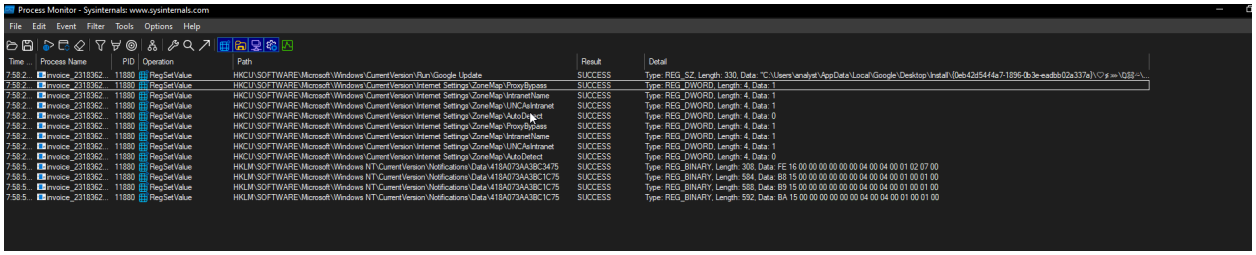


An additional filter was applied to narrow the search even further:

Operation contains RegSetValue, then Include - Show events of any registry activity, including creating or changing a new registry value.

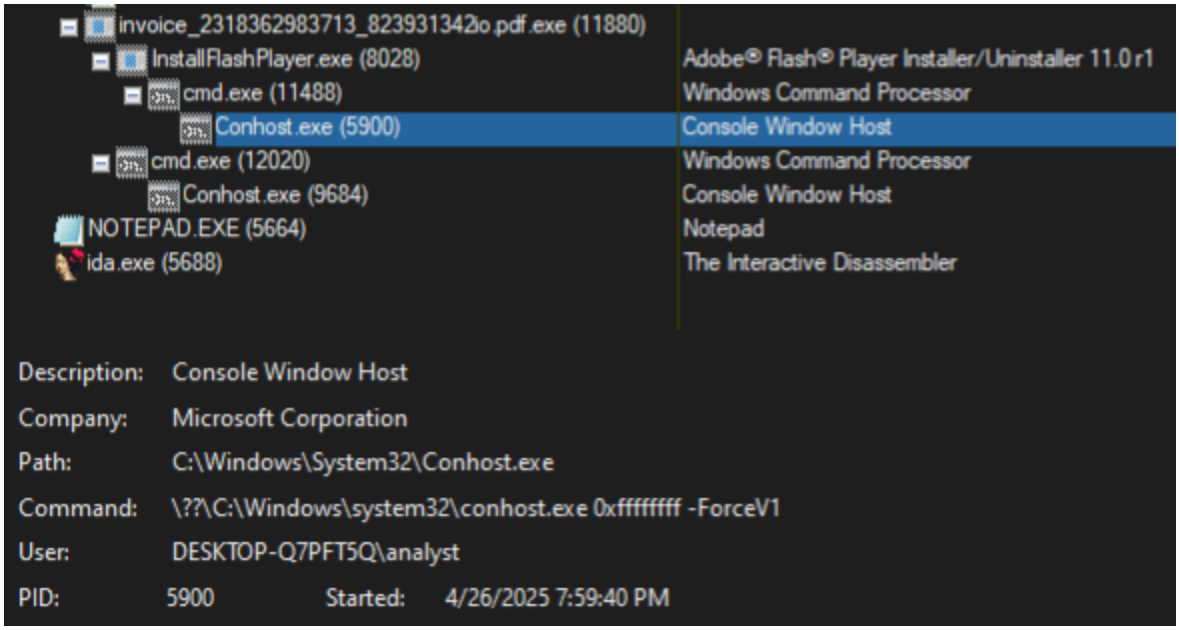


The output from the applied filter shows that the malware wrote specific values to several keys in the Registry. An event that was of interest was the malware setting a value under HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Google Update. This suggests that the malware is intended to create a persistence mechanism, such that if a value is set under Run, then the malware will automatically launch the malicious payload every time the user logs into the system. This malicious payload could be disguised as a Google update.

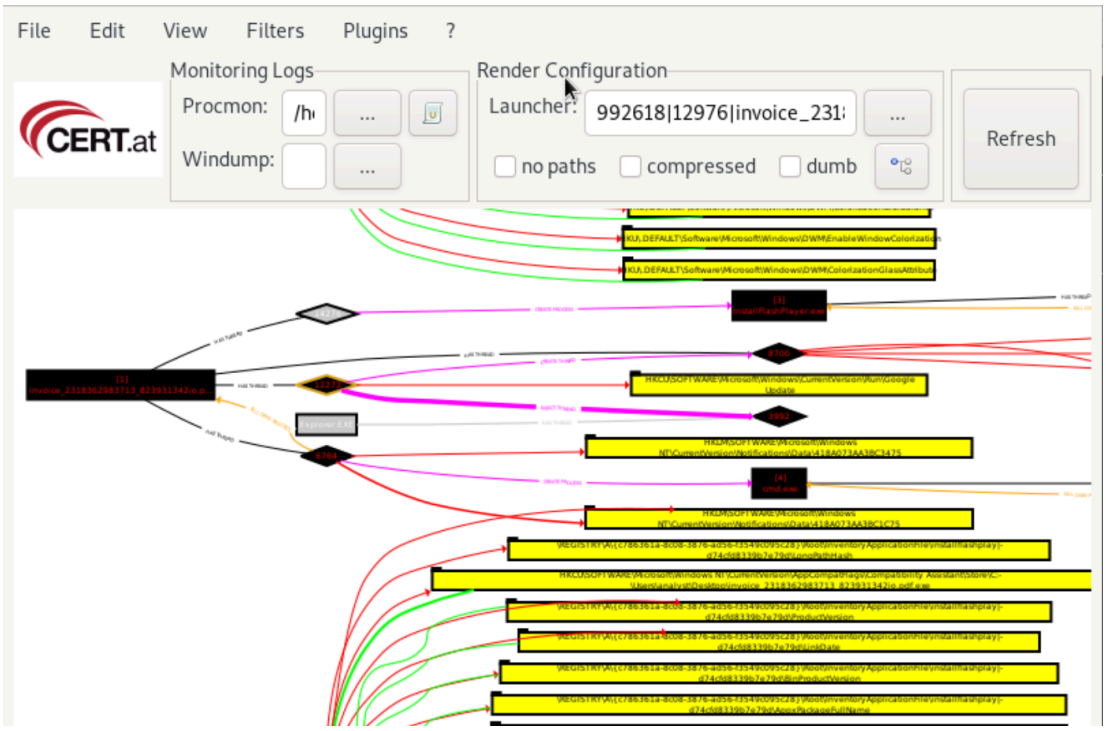


6.3) Services/Processes started

By analyzing Procmon's Process Tree feature, it was seen that the malware had executed the payload InstallFlashPlayer.exe, which subsequently launched cmd.exe and Conhost.exe**.** No command prompt window appeared on the screen, suggesting that hidden command-line instructions or operations were being performed in the background.



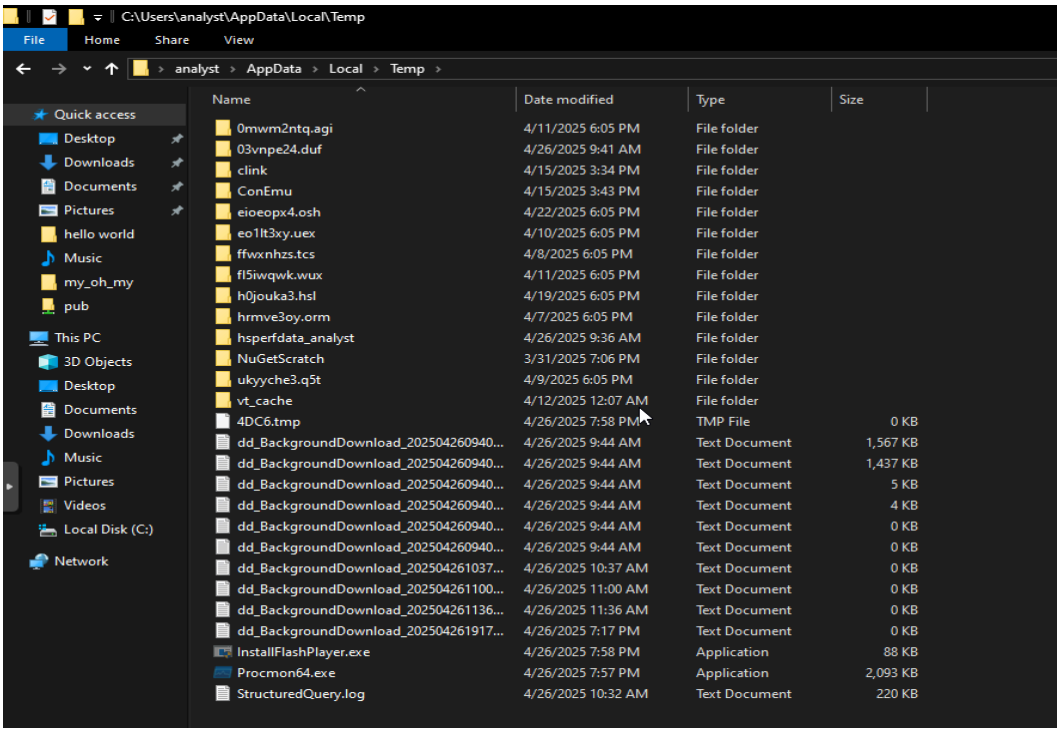
The Procmon log data was saved as a CSV file and was uploaded to procdot on the Remnux virtual environment. The procdot graph shows the initial malware execution of invoice_2318362983713_823931342io.pdf.exe and the malware spawning multiple child processes, which initiated various registry activities. InstalllashPlayer.exe was launched, which potentially acted as a fake Flash installer to deceive users while malicious activity was being performed in the background. The cmd.exe process was also launched, as the window was not displayed, which suggests that command-line activity was potentially being performed discreetly. The registry key HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Google Update is also identified in the graph, where the malware attempted to maintain persistence by relaunching itself when the user logged in to their system.



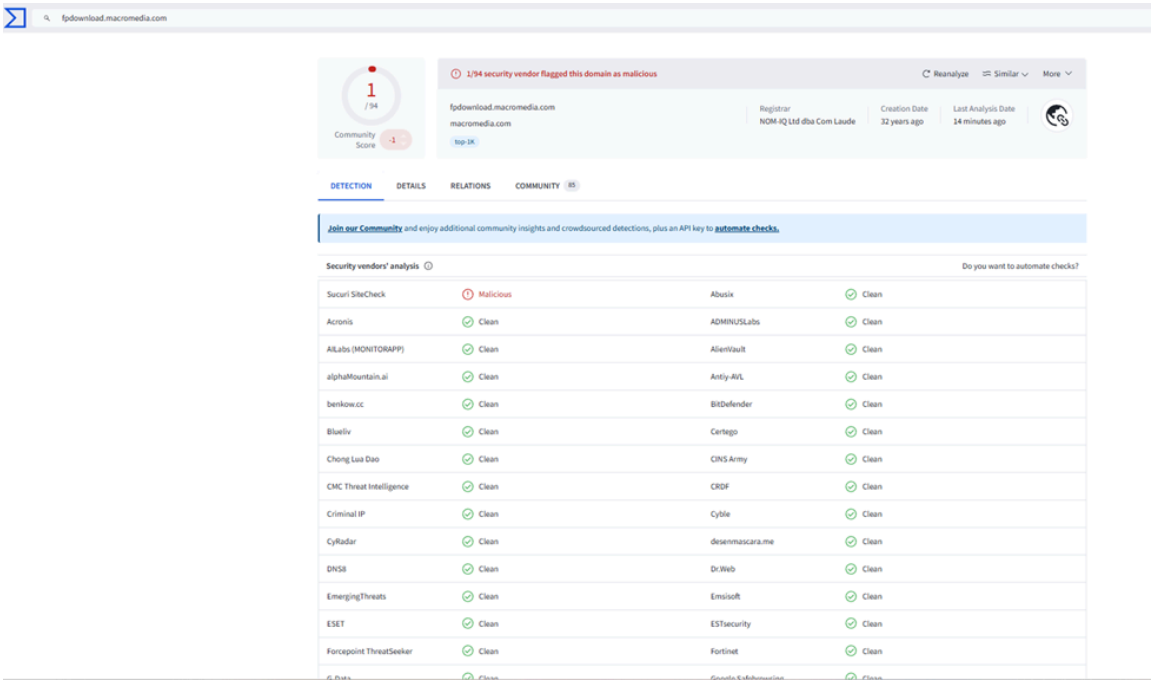
---

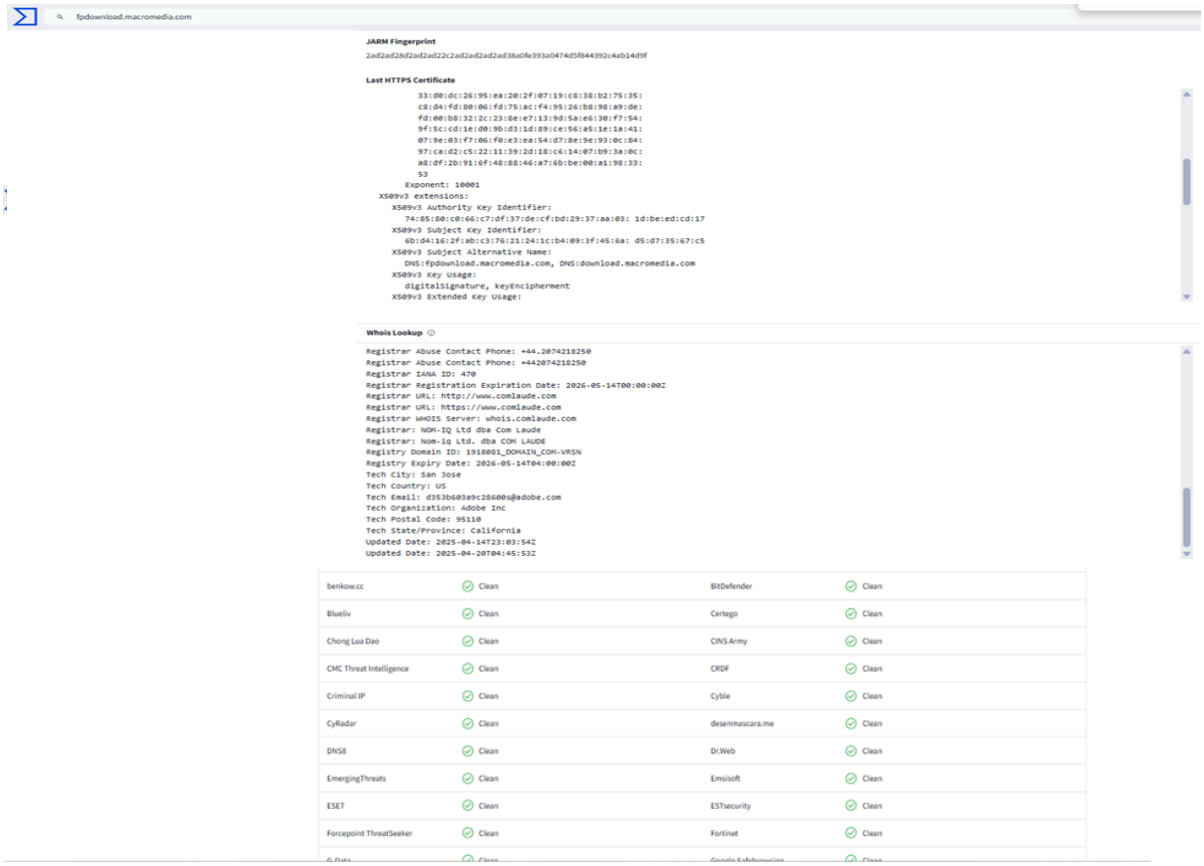
7. Supporting Data

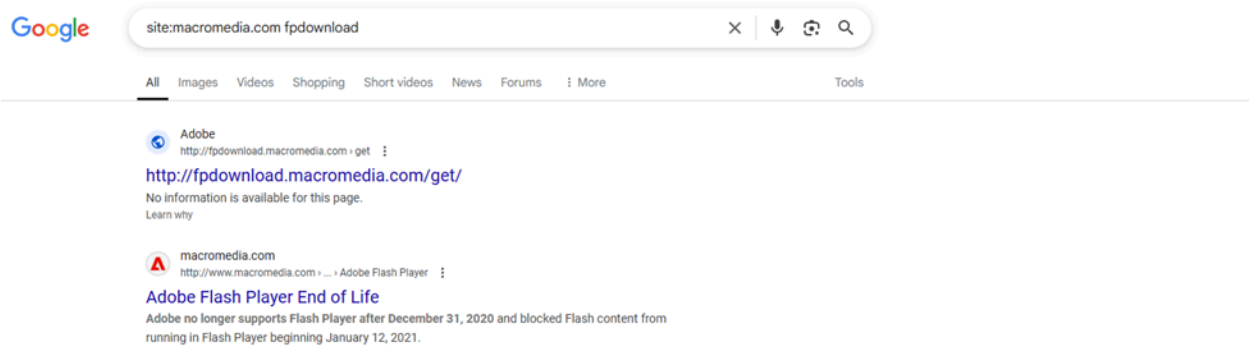
In the file path C:\Users\analyst\AppData\Local\Temp\, the InstallFlashPlayer.exe executable is present and greyed out.



VirusTotal analysis of the domain fpdownload.macromedia.com shows it has been flagged suspicious for the engine Sucuri SiteCheck, while the other 93 vendors consider it clean. This was an old, legitimate domain that was created 32 years ago and is tied to Adobe. However, it's appearance in the malware may indicate an attempt to blend in with the network traffic to check internet connectivity.







---

8. Indicators of Compromise

These indicators of compromise were extracted from the analysis of the Zeus Banking Trojan malware:

- invoice_2318362983713_823931342io.pdf.exe
- msimg32.dll
- C:\Users\analyst\AppData\Local\Temp\msimg32.dll
- HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Google Update
- HKLM\System\CurrentControlSet\Control\Session Manager
- fpdownload.macromedia.com
- GetTickCount()
- WriteFile()
- WinExec()
- 85.114.128.127
- 10.10.37.250
- MD5: ea039a854d20d7734c5add48f1a51c34
- SHA-1: 9615dca4c0e46b8a39de5428af7db060399230b2
- SHA-256: 69e966e730557fde8fd84317cdef1ece00a8bb3470c0b58f3231e170168af169
- SSDEEP: 6144:Tz/LBBTHT+7oEf2ZstxQMSGToLoOhD2saLsW8fsmFBkObjD:PLBdy7FpQMlToThD+sW8fsmP7bj fuzzy)

---

9. Recommendations

It is recommended that if a system is suspected or confirmed to be infected with the Zeus banking trojan, then users, administrators, and organizations should consider implementing these best practices to strengthen future defenses.

- Isolate affected systems from the network to prevent lateral movement, C2 communication, and credential theft.
- Scan for additional payloads and persistence mechanisms by employing endpoint detection tools.
- Assume all user and domain credentials are compromised and enforce immediate password changes and a strong password policy.
- Reimage the system from a trusted baseline.
- Update antivirus and EDR solutions to ensure definitions and signatures are current and enabled.
- Keep operating system patches up-to-date.
- Enable multi-factor authentication (MFA) on sensitive systems used for online banking.
- Apply the principle of least privilege and restrict administrative privileges.
- Update firewalls, apply DNS filters, and configure proxies to block known IOCs: IPs, hashes, and domains.
- Deploy EDR and SIEM to detect suspicious process creation and autorun registry keys.
- Conduct user awareness training to educate employees on phishing, malicious attachments, and fake update scams that could be disguised as a legitimate update.
- Segment the network using VLANs and firewall rules to separate workstations and servers from financial systems.
- Apply application whitelisting to allow only trusted applications to run.
- Conduct regular backups for critical systems.