**Malware Analysis Report: Agent Tesla**

For this report, I chose to go with a popular strain of malware that seems to appear nearly everywhere on the MalwareBazaar Database. I suspect that this might be due to its source code either being released or leaked to the public. Either way, Agent Tesla is categorized as a RAT and since new variations of it are popping up quite fast, I decided that it would be an interesting piece of malware to learn about (abuse.ch).

The specific strain of Agent Tesla we are going to be analyzing is identified by its   
SHA-256 hash: 2639e5b110681251c4e95e28c86f93e02871e46e896a202d731c393e647b08f2

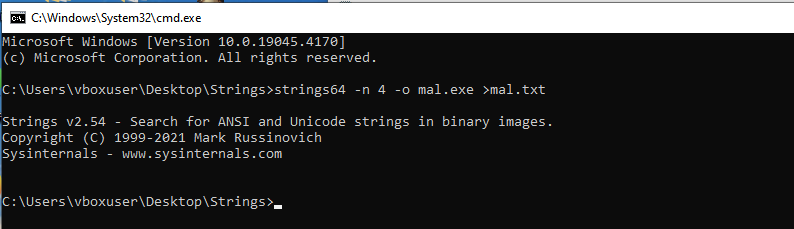
Links to references will be provided at the end of the report.  
 This report will include the usage of a Windows 10 Virtual Machine and the following tools: Strings64, UPX (did not work), PEview, VirusTotal, Regshot, Process Monitor, ApateDNS, Wireshark, x64 Debugger, and IDA pro.

Tools not used in this report: Resource Hacker (was not needed), and Dependency walker (had too many errors).

**AgentTesla: Strings**

First, we will utilize Strings to get a sense of what functions and literal strings are present in the executable.

**Figure 1**

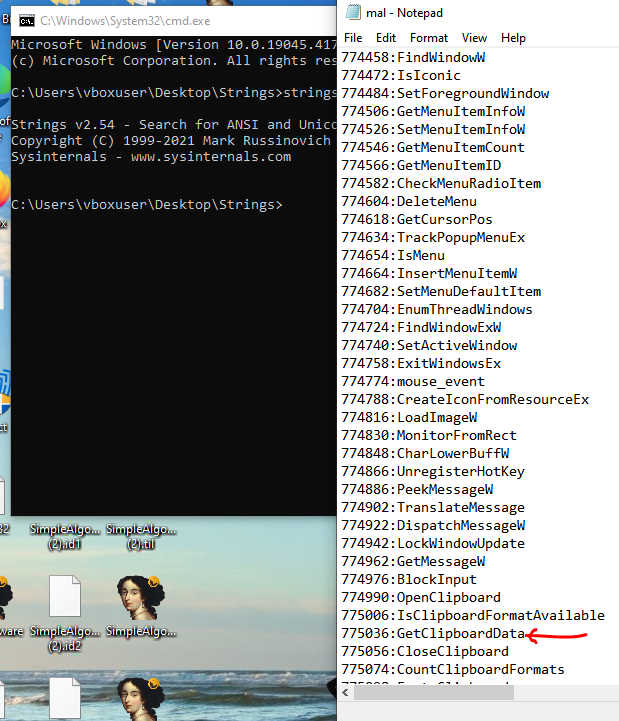


Upon successfully running Strings, we are presented with a massive text file containing around 1,000,000 lines of text. Most of which I ended up erasing because it was undecipherable. After doing so, I ended up with around a few thousand lines of text. This executable has a whole myriad of functions that expand its capabilities. While going through some of the functions I found a few to be noteworthy:

**775036: GetClipboardData**

This function is fairly self-explanatory, it allows the executable to read whatever is stored in the user's clipboard. Seeing how we are dealing with a RAT; this could be used maliciously to extract some sensitive information a user does not wish to type out. This is especially important since RATs usually come with keyloggers, making this RAT very versatile.

**Figure 2.**



**774990: OpenClipboard**

**775006: IsClipboardFormatAvailable**

**775036: GetClipboardData**

**775056: CloseClipboard**

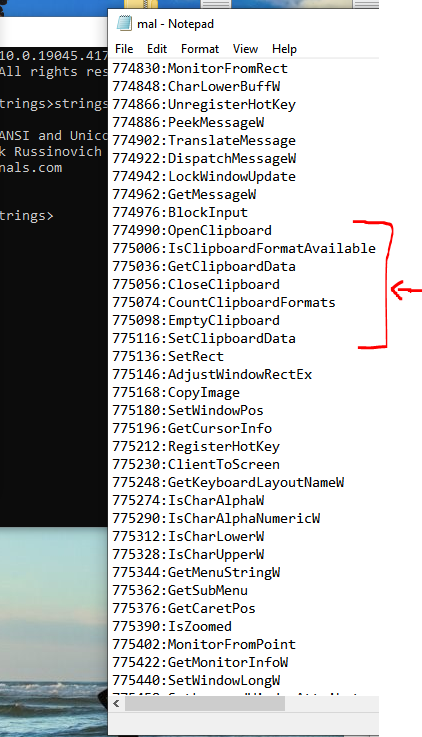
**775074: CountClipboardFormats**

**775098: EmptyClipboard**

**775116: SetClipboardData**

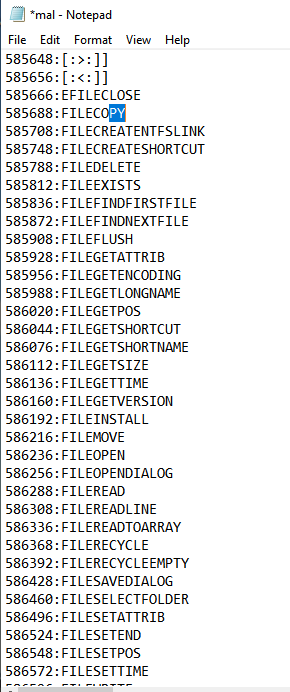
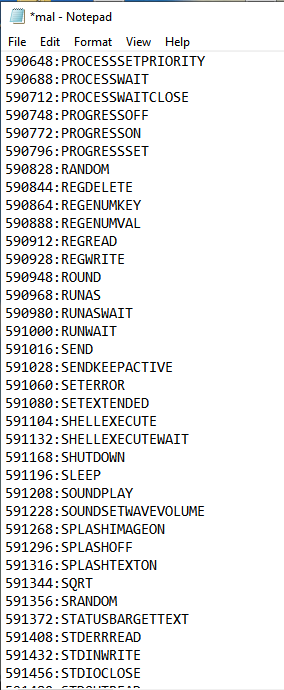
As we can see, similar to the previous figure we have a cluster of functions that all directly interact with our clipboard, even as far as emptying and setting our clipboard data, being highly invasive.

**Figure 3.**



**More Highly Invasive Functions:**

While I can’t provide the whole list in an image, I will provide an extra two photos. Keep in mind that there are thousands of functions similar to the ones in Figures 4 and 5. Indicating that this RAT is not meant to be compact and precise. If infected, it should grant the attacker full unfettered access to a victim's PC. Some dangerous functions include: FILECOPY, FILEDELETE, FILEMOVE, SHELLEXECUTE, SOUNDPLAY, SHUTDOWN, etc…

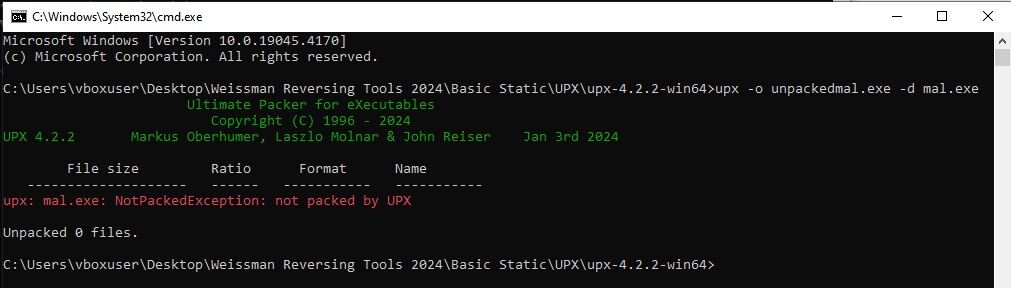
 

**Figure 4.** **Figure 5.**

**AgentTesla: UPX**

As previously stated, there were thousands of lines which were undecipherable. So, I decided to see if it was packed, and if UPX could unpack it. Unfortunately, as we can see in Figure 6, UPX could not unpack it which means it was not packed in that way.

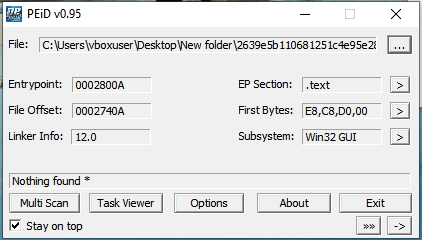
**Figure 6.**



**AgentTesla: PEiD**

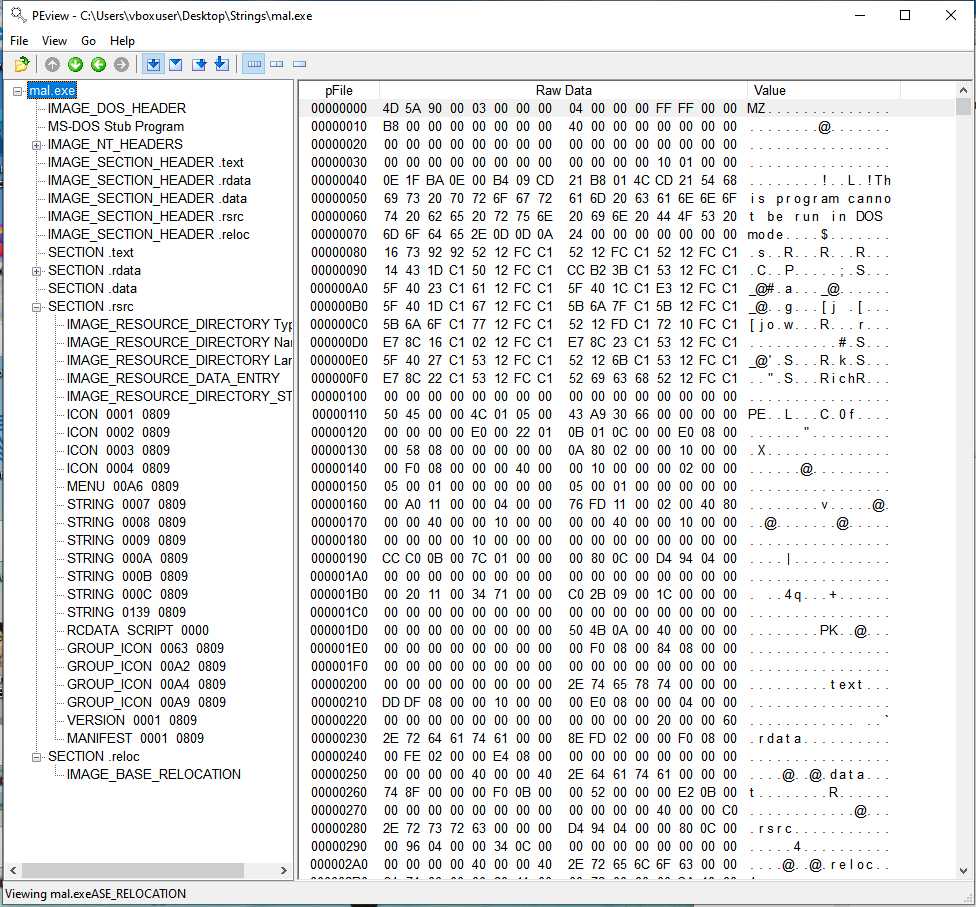
Given the results presented by UPX I decided to analyze the file using PEiD. As we can see in Figure 7, this executable does not use any kind of packer (hardcore scanning was enabled here). This means our initial results from Strings are valid, and that the undecipherable text might have been an attempt to obfuscate some data.

**Figure 7.**

****

**AgentTesla: PEview**

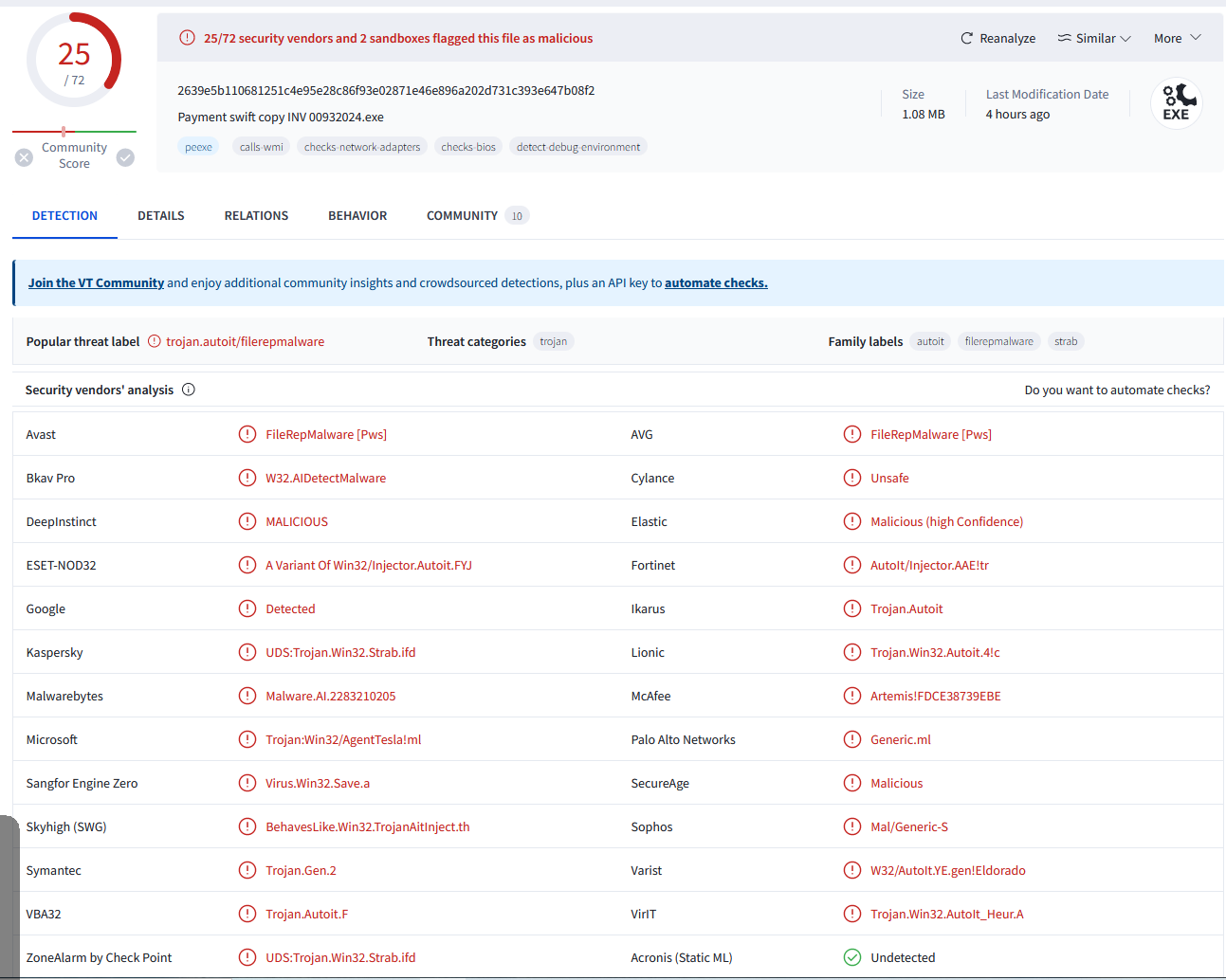
Loading the executable in PEview did not lead to any remarkable intel. However, it did tell us that our executable (a portable executable) is not hiding any other binaries from us. This means that we can somewhat safely assume that most of the functionality of this malware is done directly and solely from this portable executable. We can see this in Figure 8 where no additional binaries are present in the “.rsrc” section. Thus, “resource hacker” was not of use here.

**Figure 8.**

**AgentTesla: VirusTotal**

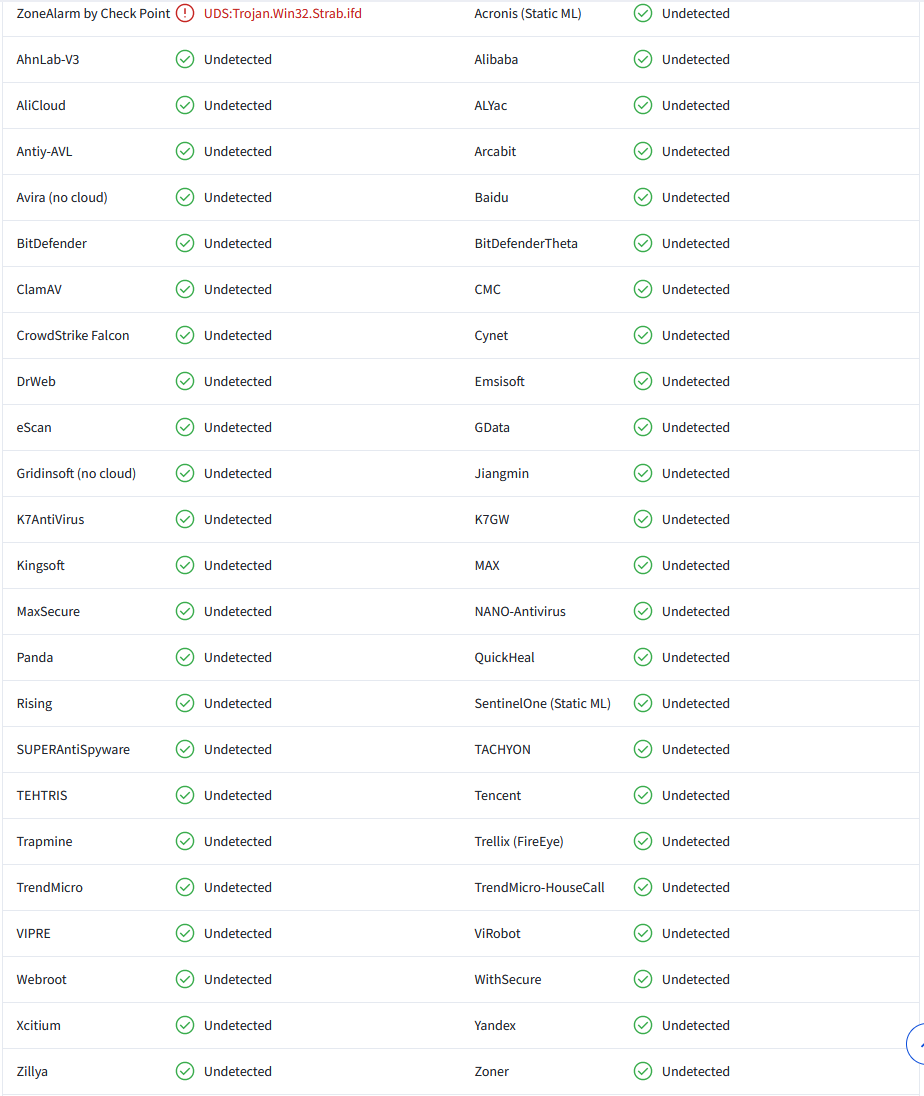
VirusTotal was a crucial tool in this analysis. However, later on in the report, we will see that we discover some things that were not yet added to VirusTotal, most likely due to this strain of malware being fairly new. Keeping this in mind, we will do a short return to VirusTotal to go over the unreported findings. Without further ado let’s dive into the initial analysis.

According to VirusTotal detection tab in Figure 9, only 25/72 vendors found this executable malicious, this could be partly because this is a newer strain of the Agent Tesla Malware. The good news is that major vendors have caught on to it.

**Figure 9.**  


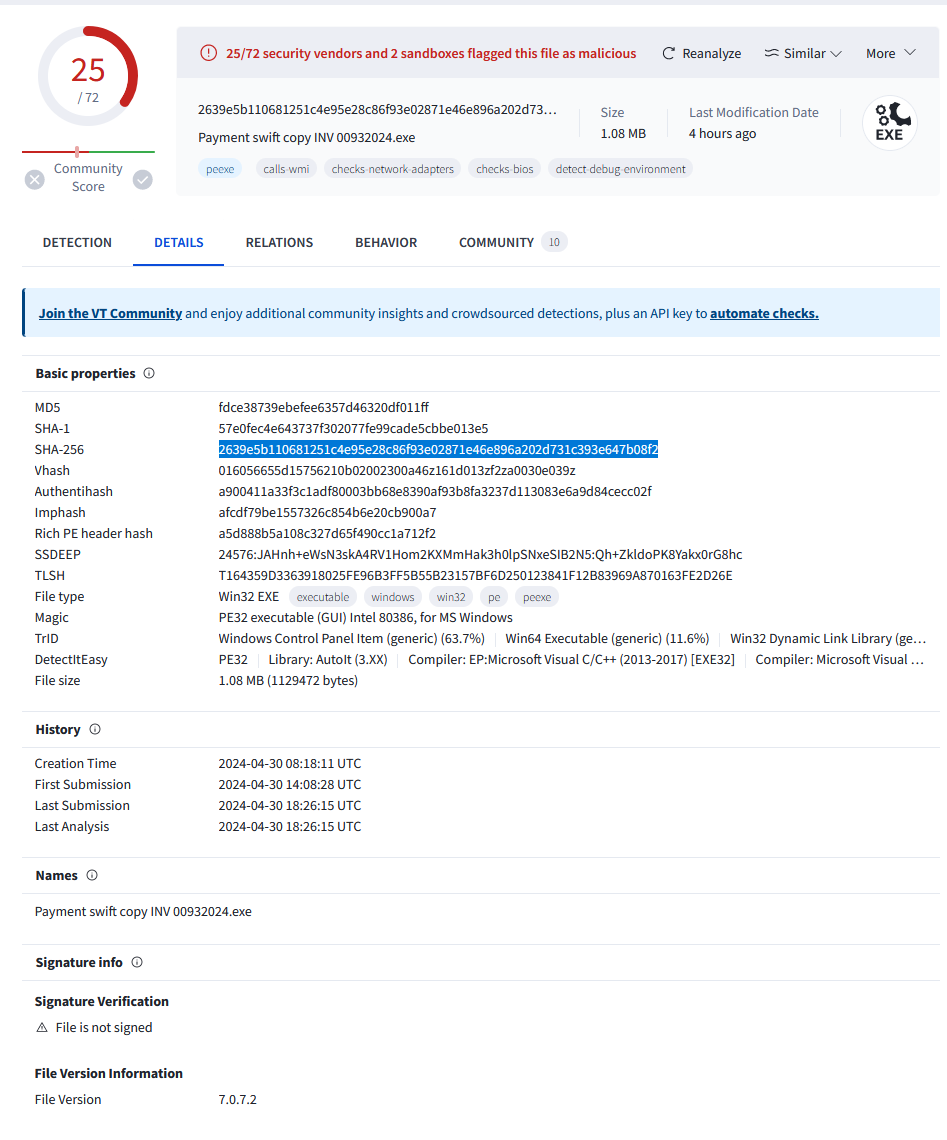
While looking at some of the vendors in Figure 10, I found a few that seem worrying as they did not detect this malware. Bitdefender, Crowdstrike Falcon, VIPRE, and SentinalOne did not detect the malware which is odd since they are major cybersecurity companies. Again, this could be partly because this strain is new.

**Figure 10.**

****

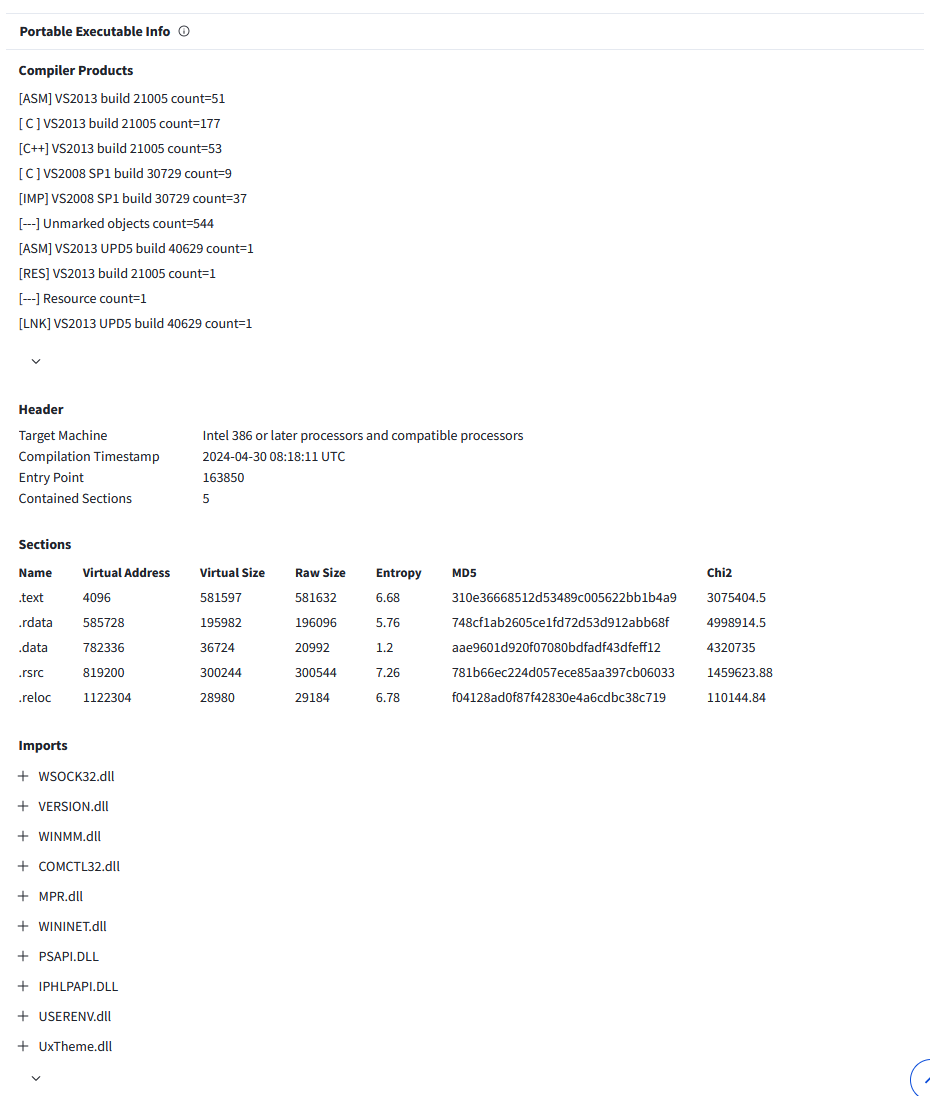
Moving onwards to the details tab in Figure 11, we can see the confirmation that this is indeed the same file we saw on MalwareBazaar as indicated by the matching SHA-256 hash. We also see that this executable was first analyzed by someone using VirusTotal only 4 hours before us! Also, we need to pay attention to the executable name on the top as it might come in handy in the future (INV 00932024.exe).

**Figure 11.**



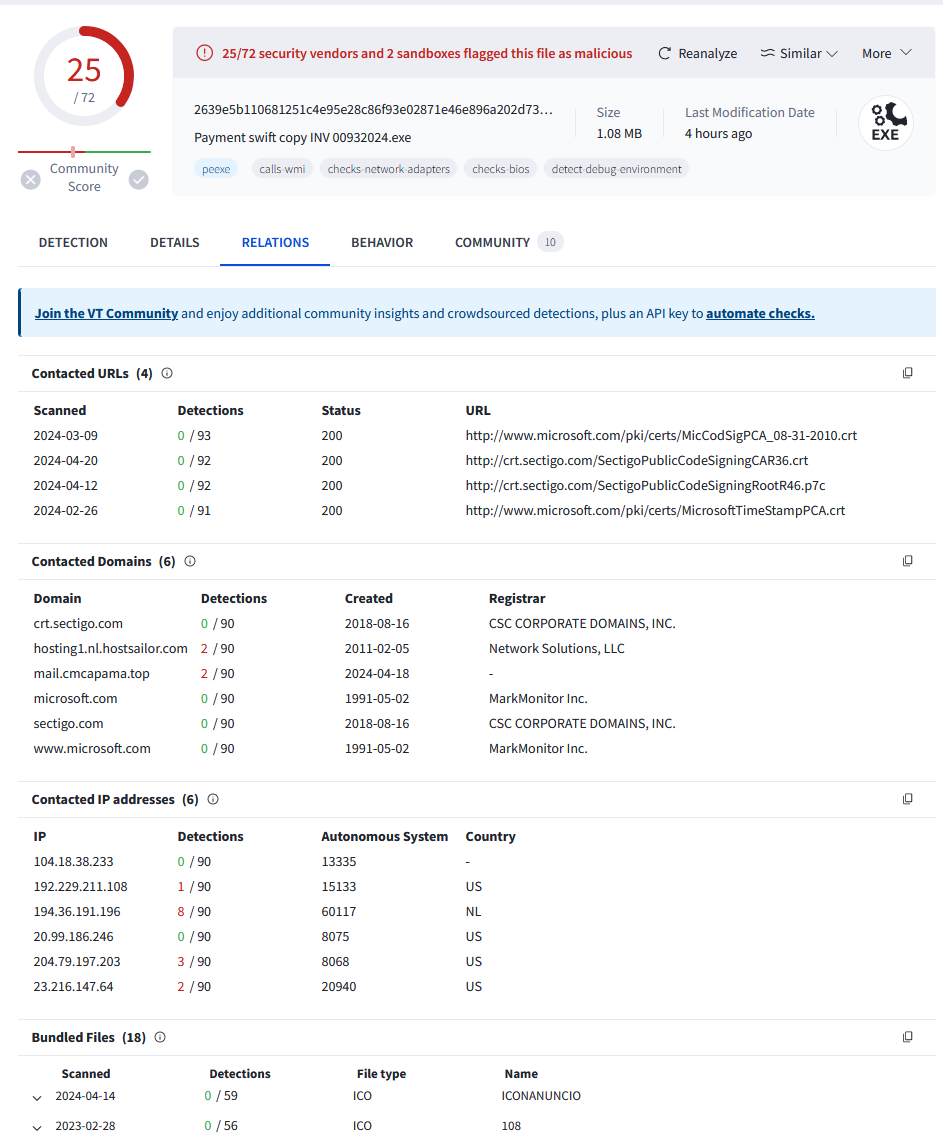
Continuing with the details tab we can see some more relevant information, as mentioned before, some of the information on VirusTotal is incomplete. As such it’s worth taking note of the Compiler Products section and its content in Figure 12 as it could have relevance in the secondary analysis.

**Figure 12.**

****

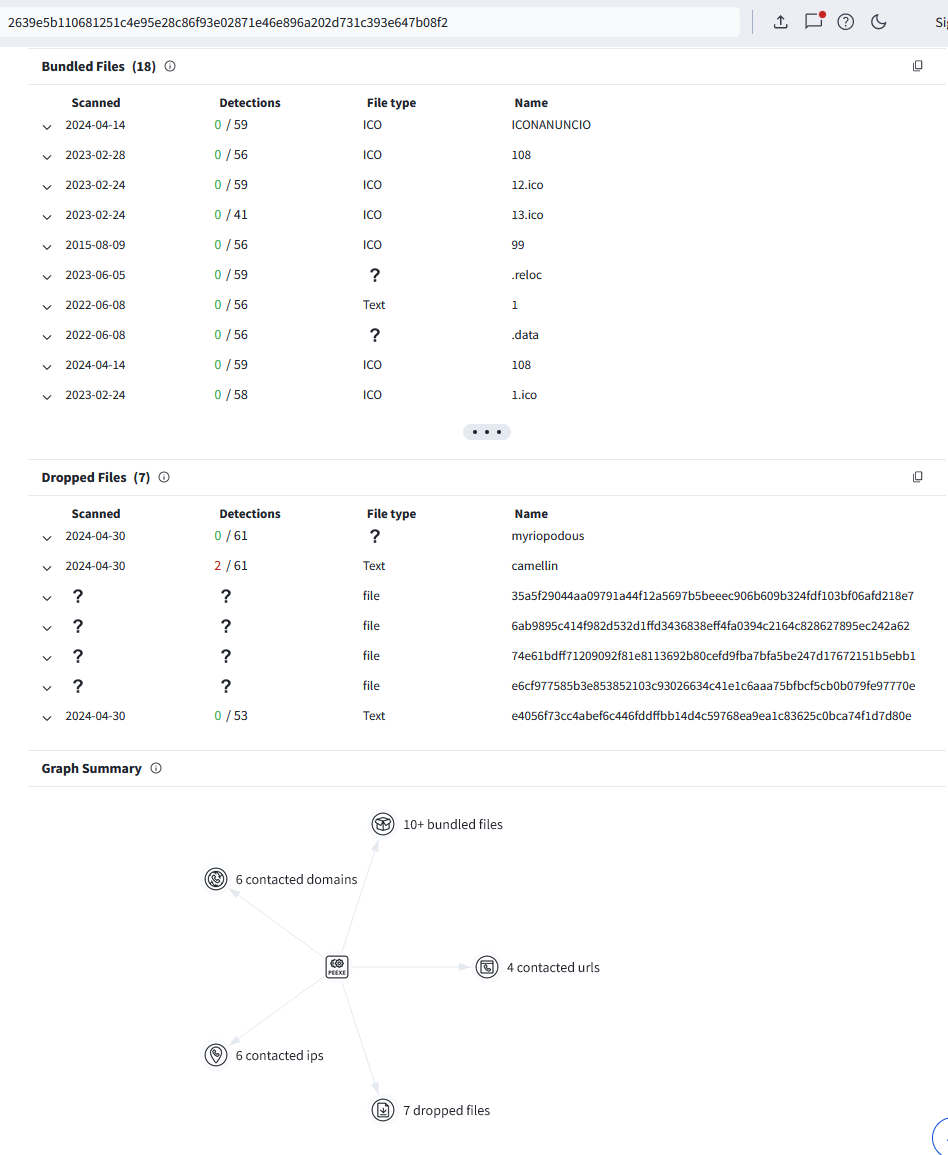
Moving into the relations tab in Figure 13, there is a plethora of information that will be relevant to the secondary analysis. Upon first analysis, we can see that not many of the related IPs or FQDNs are being marked as malicious, again most likely partly due to it being new. It almost looks as if some of these results may be benign. However, note that some of the content presented here (domains) will be referenced in subsequent sections of the report with vindication.

**Figure 13.**



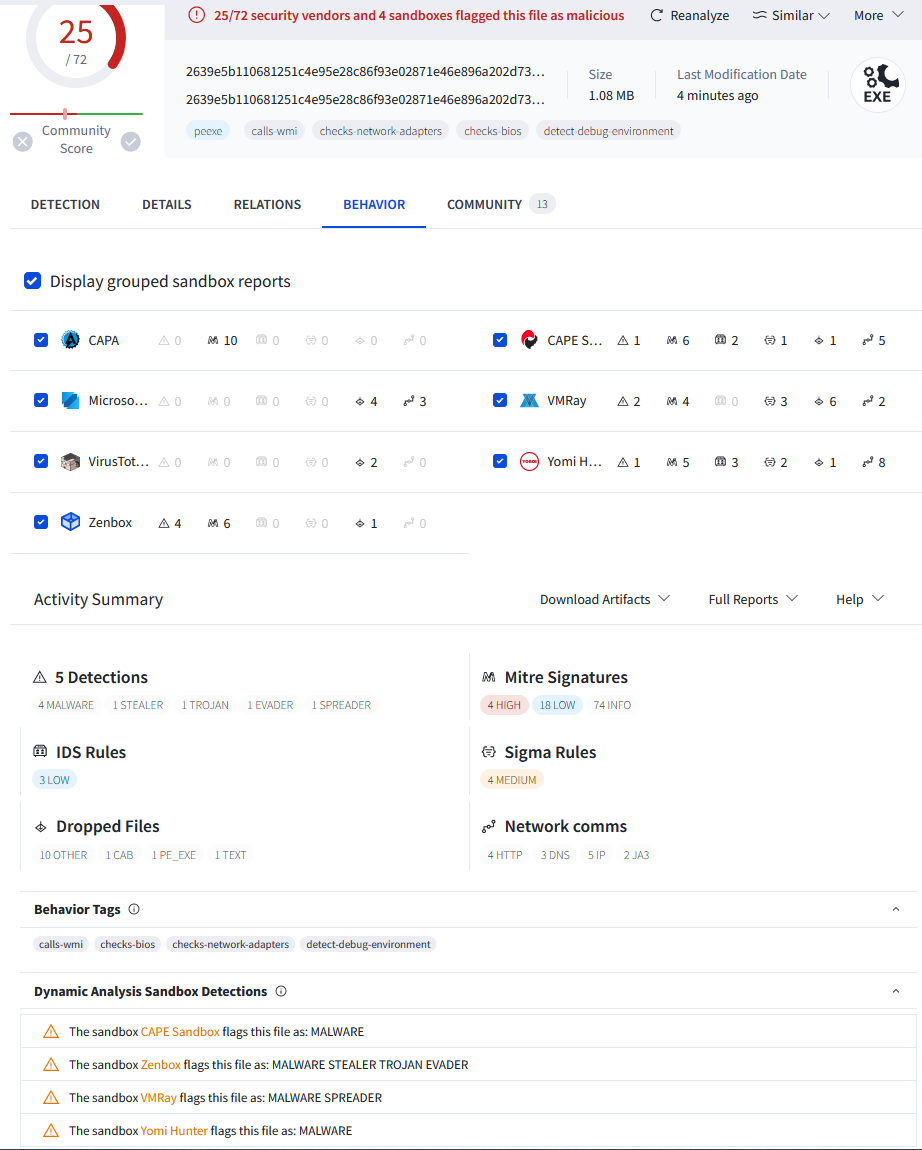
Continuing down in the relations tab in Figure 14, we see some more benign-looking results. However, pay very close attention and take note of the “Dropped Files” section. Upon first analysis, I overlooked this detail thinking it may not be too relevant. But when we dive more deeply into Basic Dynamic analysis, we will see how this becomes increasingly relevant.

**Figure 14.**

****

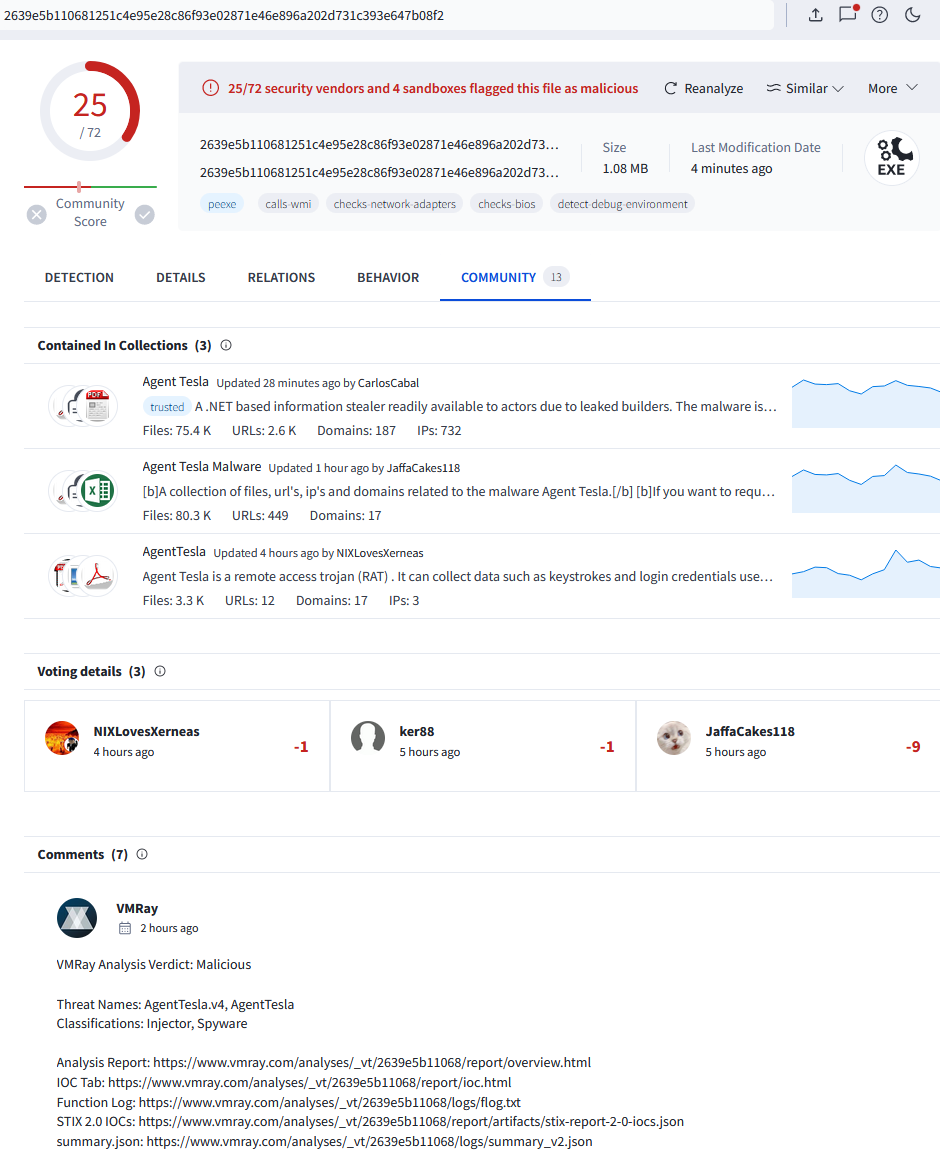
Under the behavior tab, we see additional information about what the sandbox environments found and concluded. As we can see, the dynamic analysis sandboxes in Figure 15 found that this executable is a MALWARE, and/or MALWARE STEALER TROJAN EVADER, and/or MALWARE SPREADER. Again, solidifying our assumption that this is a newer strain of malware that is difficult to detect using more methodical analysis employed by the undetected vendors.

**Figure 15.**



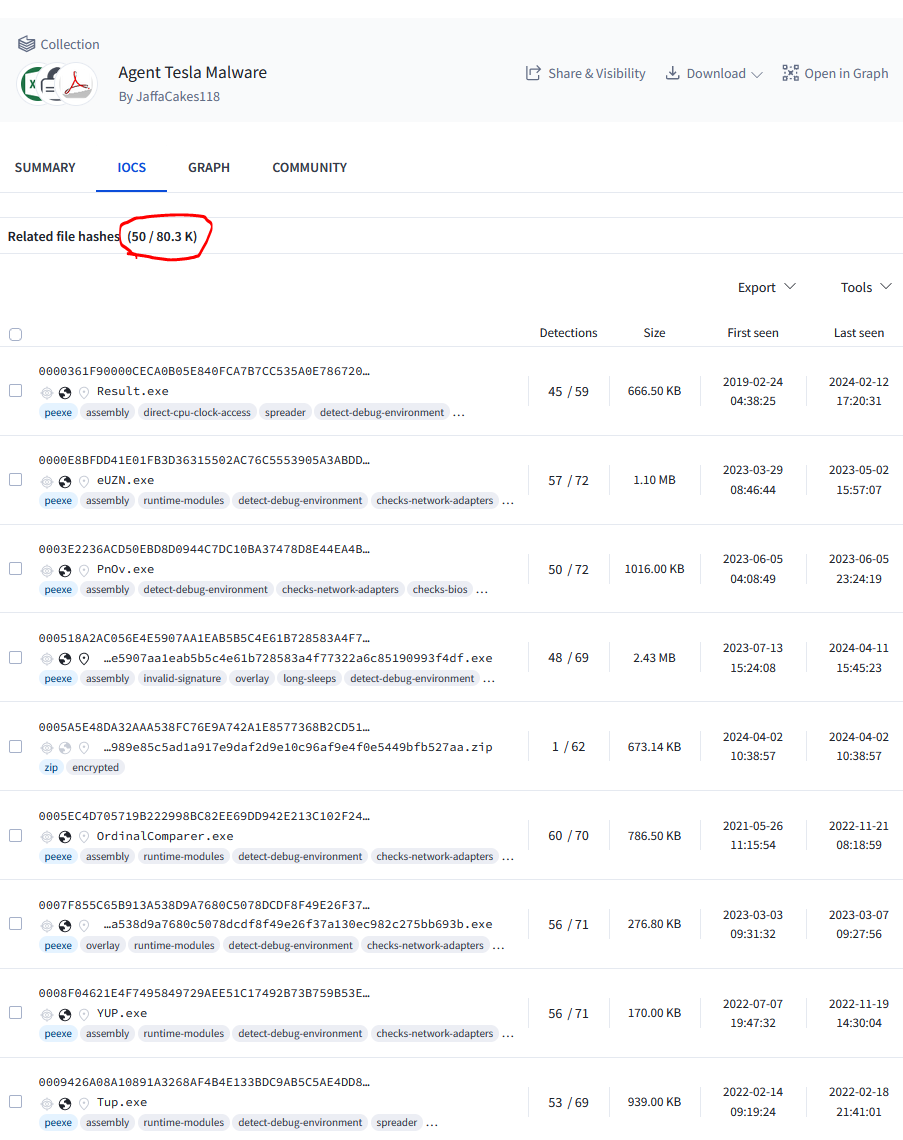
Finally, arriving at the community tab in Figure 16 shows us that while this strain of malware may be fairly new and undetected, its origin strain is very well known and documented in the “Contained in Collections” section. The middle link by JaffaCakes118 contains some interesting information as well which we will quickly look at next.

**Figure 16.**



Reading over some community information provided by JaffaCakes118 in Figure 17 we can see some related file hashes. And there are quite a lot, more than 80,000 to be precise. This shows us how this malware has been blown out of proportion.

**Figure 17.**



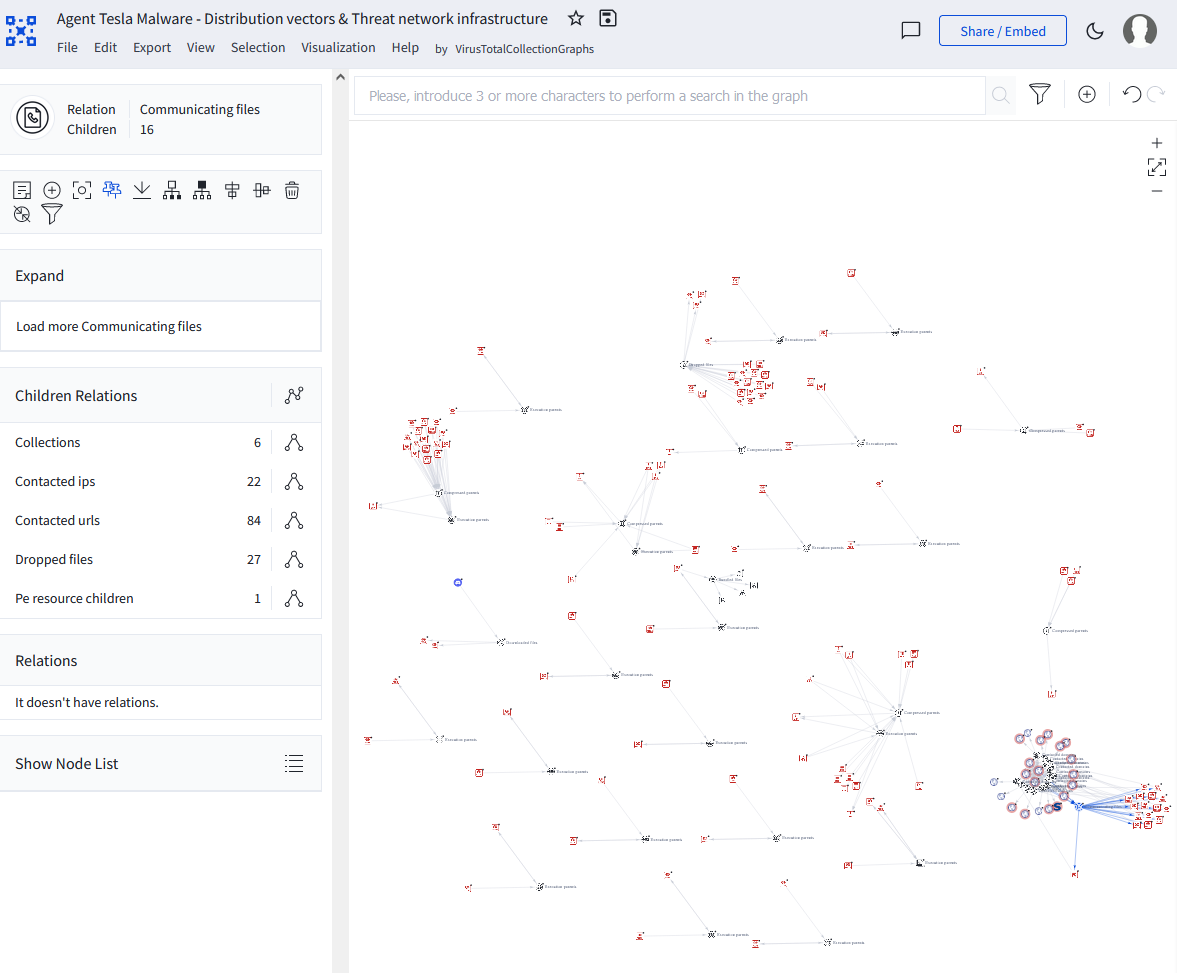
Additional information was provided as well in Figure 18, including Related domains that could be spreading the malware. I noticed Discord to be on the list as well which is interesting since Discord has been increasingly used by hackers to host “Webhooks” which are used almost like a Dropbox, where stolen user data gets sent to.

**Figure 18.**



Finally, our last tab in Figure 19 contains a visual graph mapping of the malware. It provides us with a nice top-down view of how this malware is behaving. And again, as we see, it has many isolated instances of distribution vectors that correspond with our assumptions.

**Figure 19.**

****

**AgentTesla: Reghot**

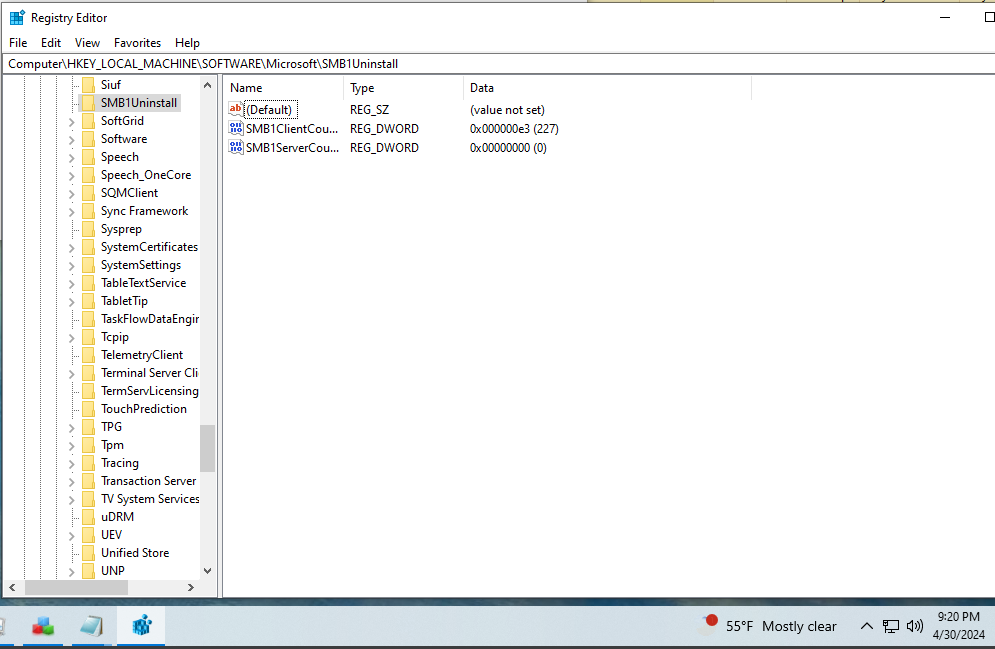
Regshot allowed us to gain some insight as to how the malware operates, keep note that I ran Regshot twice to confirm my suspicions while also comparing some values to my host PC. It was the first tool that started allowing me to connect some dots and gain information about the logical behavior of the malware. While sifting through some of the entries in Figure 20 I noticed a reference to SMB1. Usually, SMB1 is disabled in modern Operating Systems by SMB2 and SMB3 since it is very vulnerable. However here we see that SMB1 usage is being tracked, and modified moments after the execution of the executable. Possibly indicating that there is an exploit being used to spread the malware throughout networks. The usage (counter) in our case has a hex value of E1 (225) jumping to a value of E2 (226) as highlighted below.

**Figure 20.**

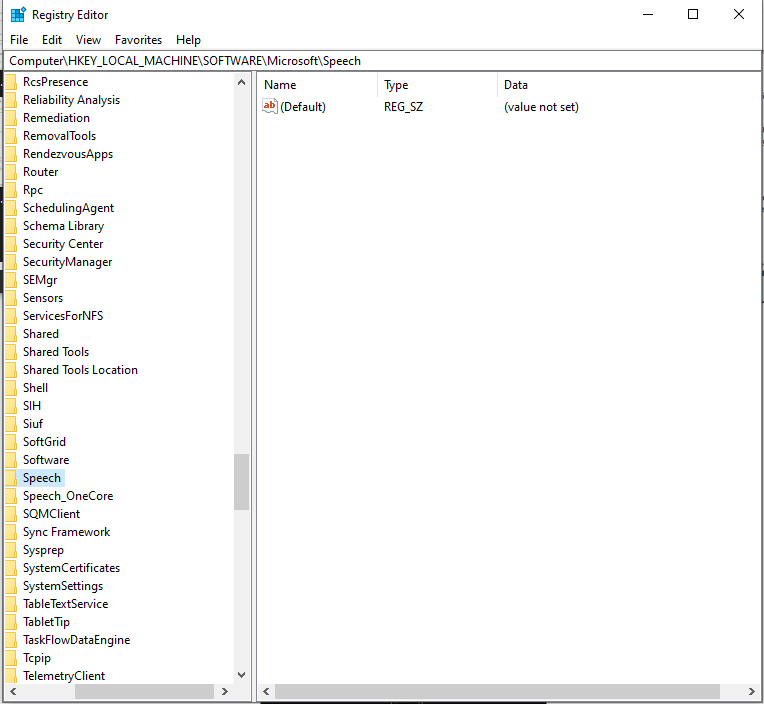


Interestingly enough, a few minutes after the Regshot was done. I decided to observe the registry value in Registry Editor, that’s where I noticed the value increment in Figure 21. Indicating that a process is repeatedly trying to access SMB1 and incrementing it to E3 (227).

**Figure 21.**



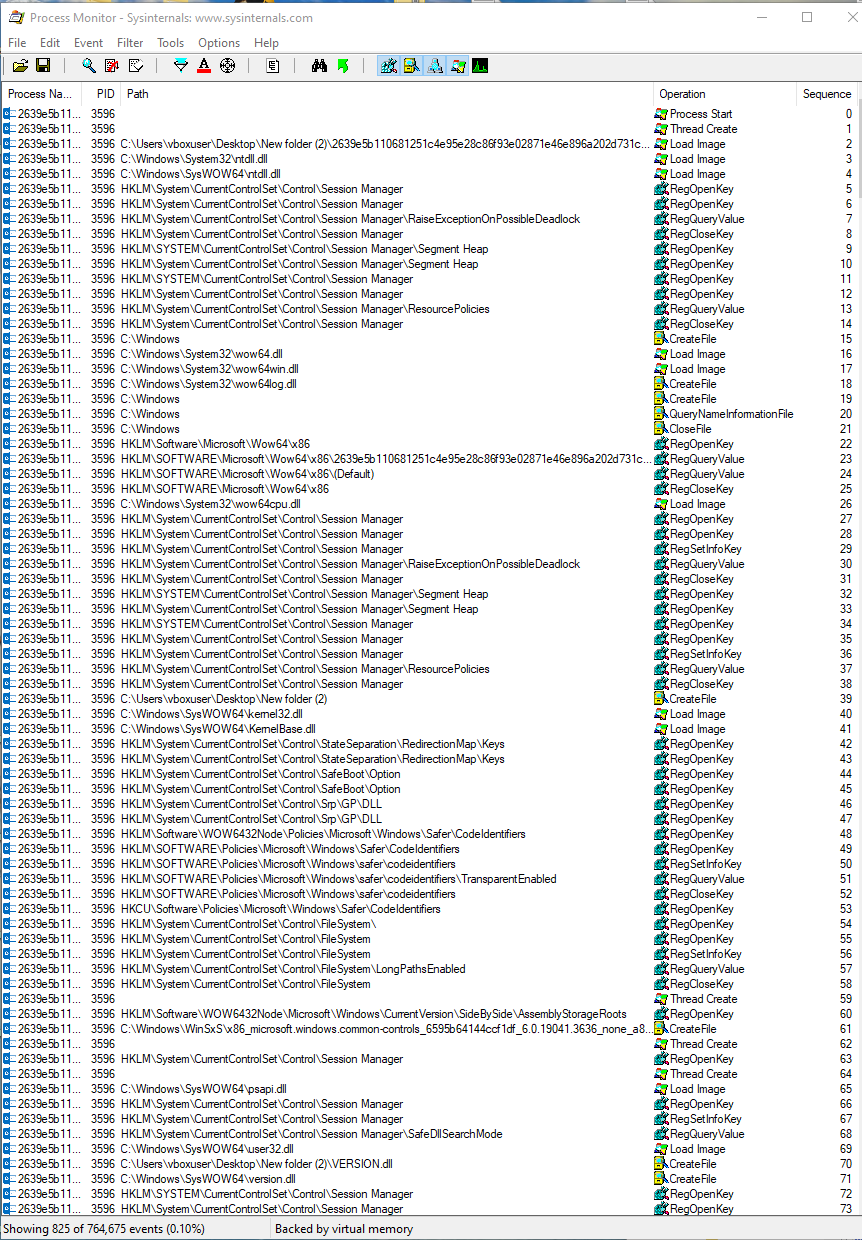
To investigate the SMB1 registry entry, I compared it and its values between my host PC and the virtual machine. This comparison was important because all of the programs installed on the virtual machine (except the malware) were also installed on my host PC. Thus, any registry values present on the virtual machine but absent from the host PC would suggest that these were introduced by the malware, as it is the only differing factor. And as we can see in Figure 22, this is exactly the case. The host PC does not even contain the path for SMB1, indicating that the malware is accessing this outdated vulnerable protocol.

**Figure 22.**

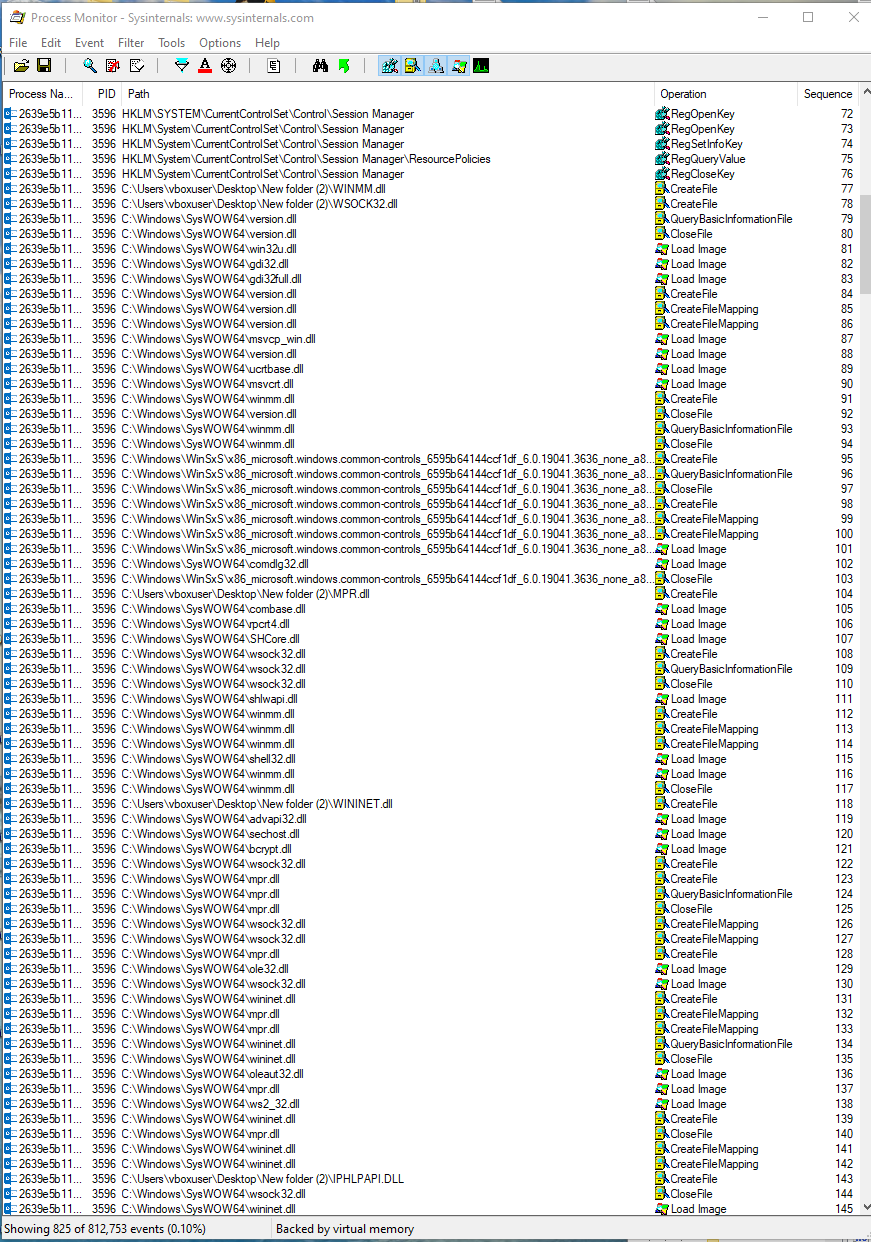
**AgentTesla: Process Monitor**

Process Monitor provided a goldmine of information. I did my best to omit and condense as much as I could. However, upon execution of the malware, a long list of operations was executed. The reason why I won't omit them is because I am certain they all compile and write their information in two separate files which also appear in the list of operations. The next Figures will illustrate this. The configuration for the filters will be provided at the end as well.

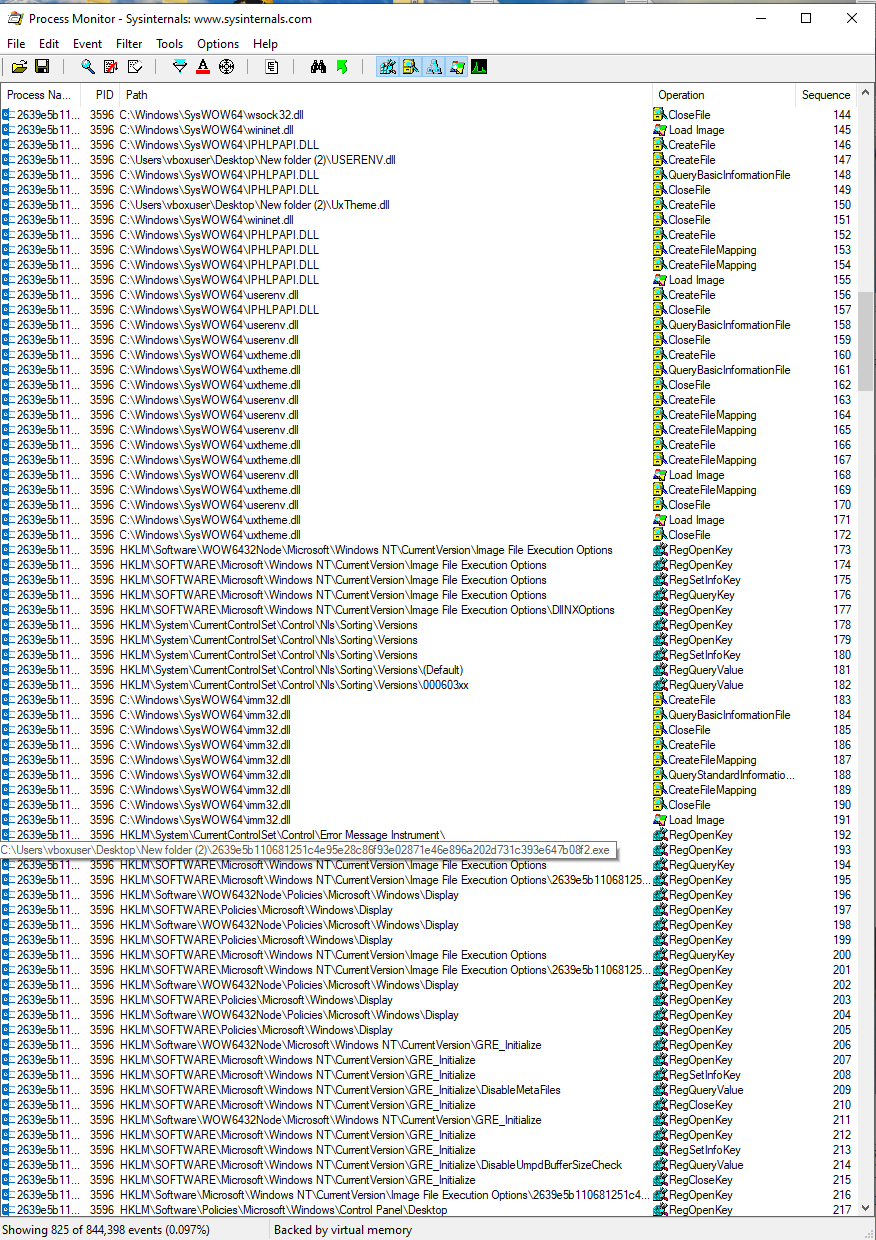
**Figure 23.**



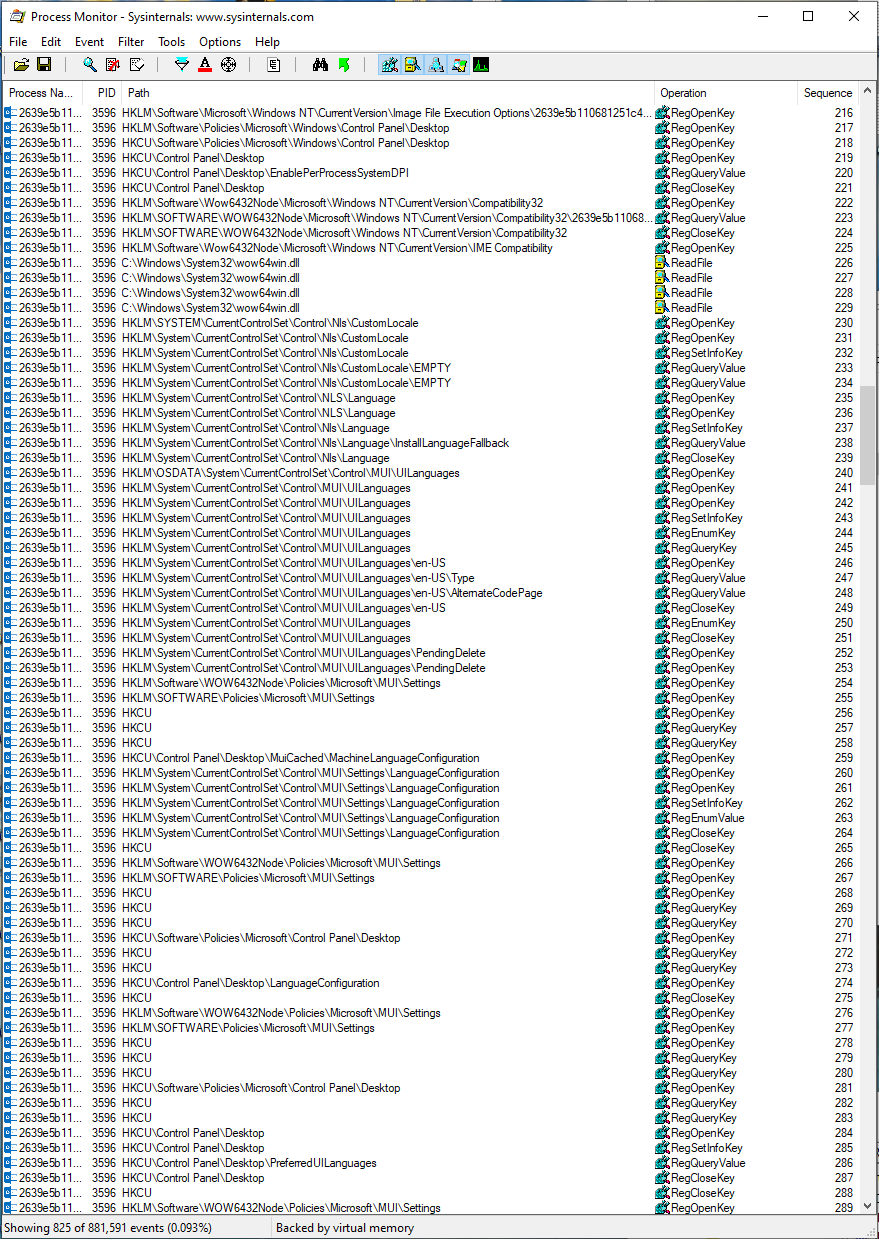
**Figure 24.**



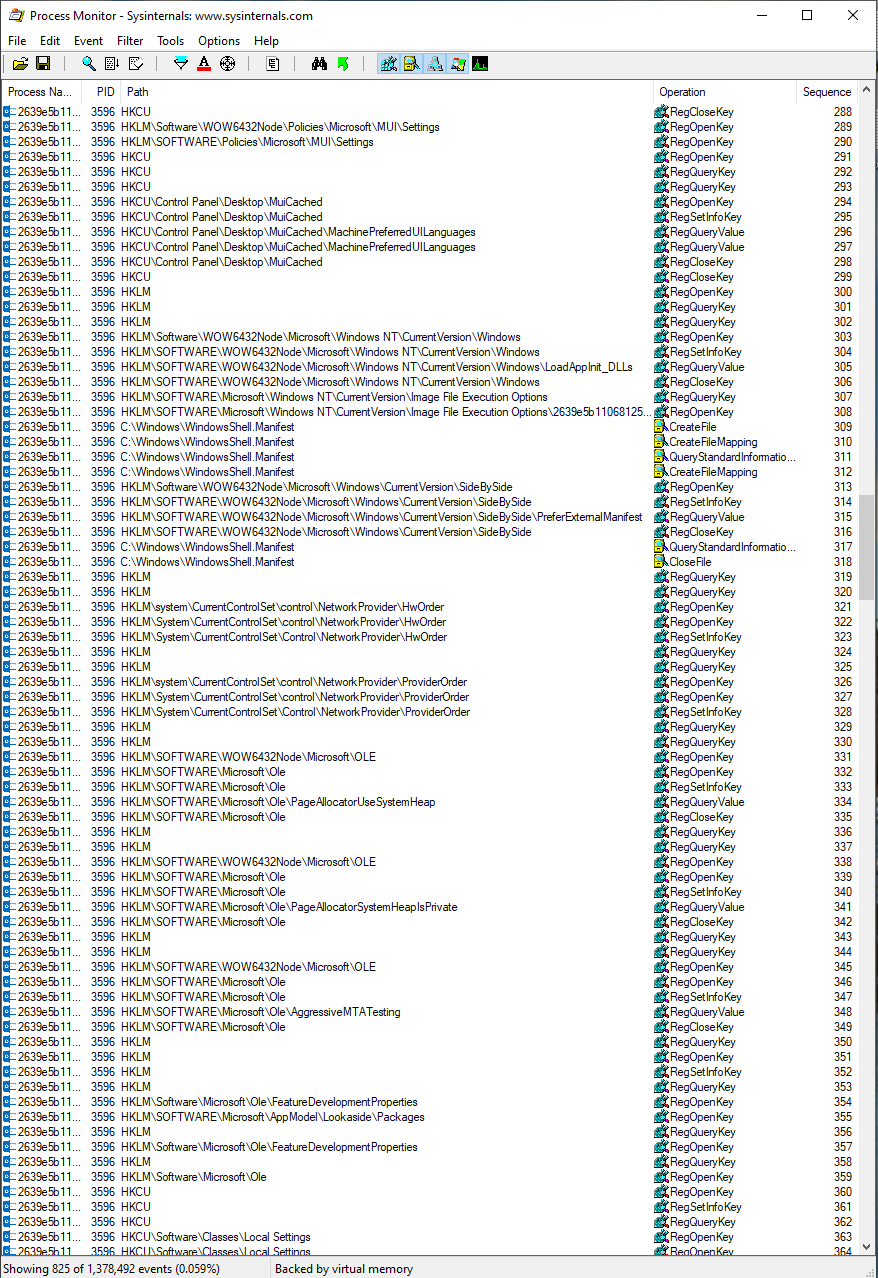
**Figure 25.**



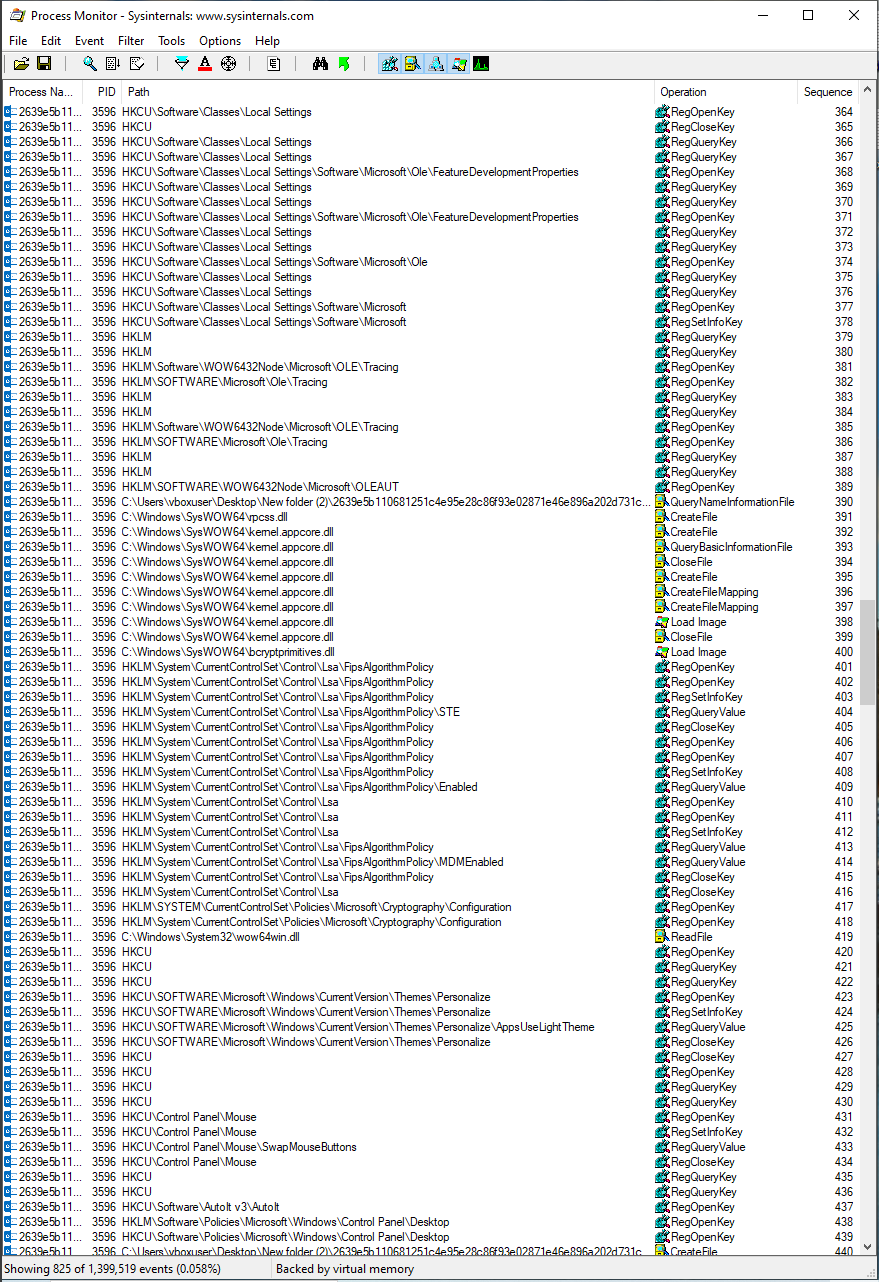
**Figure 26.**



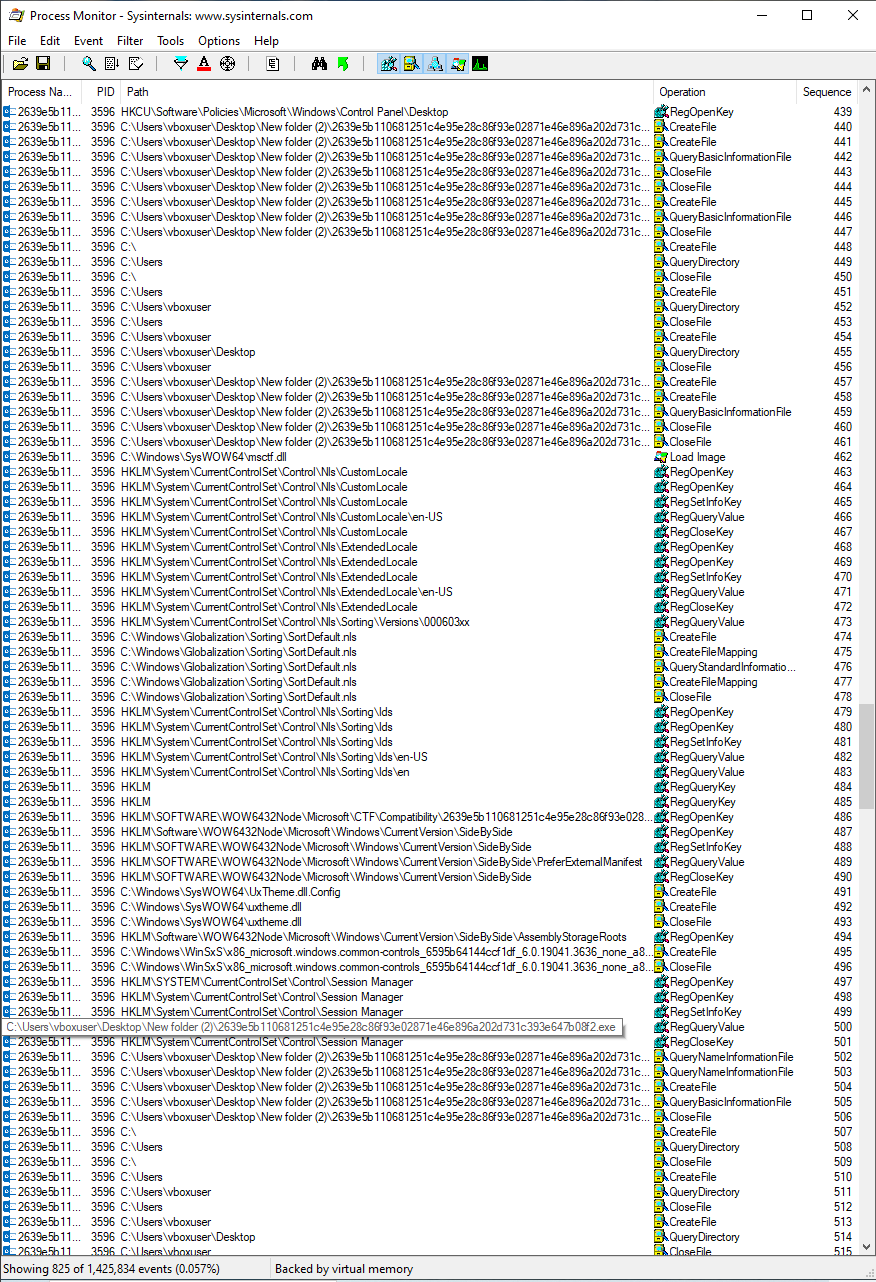
**Figure 27.**

**.**

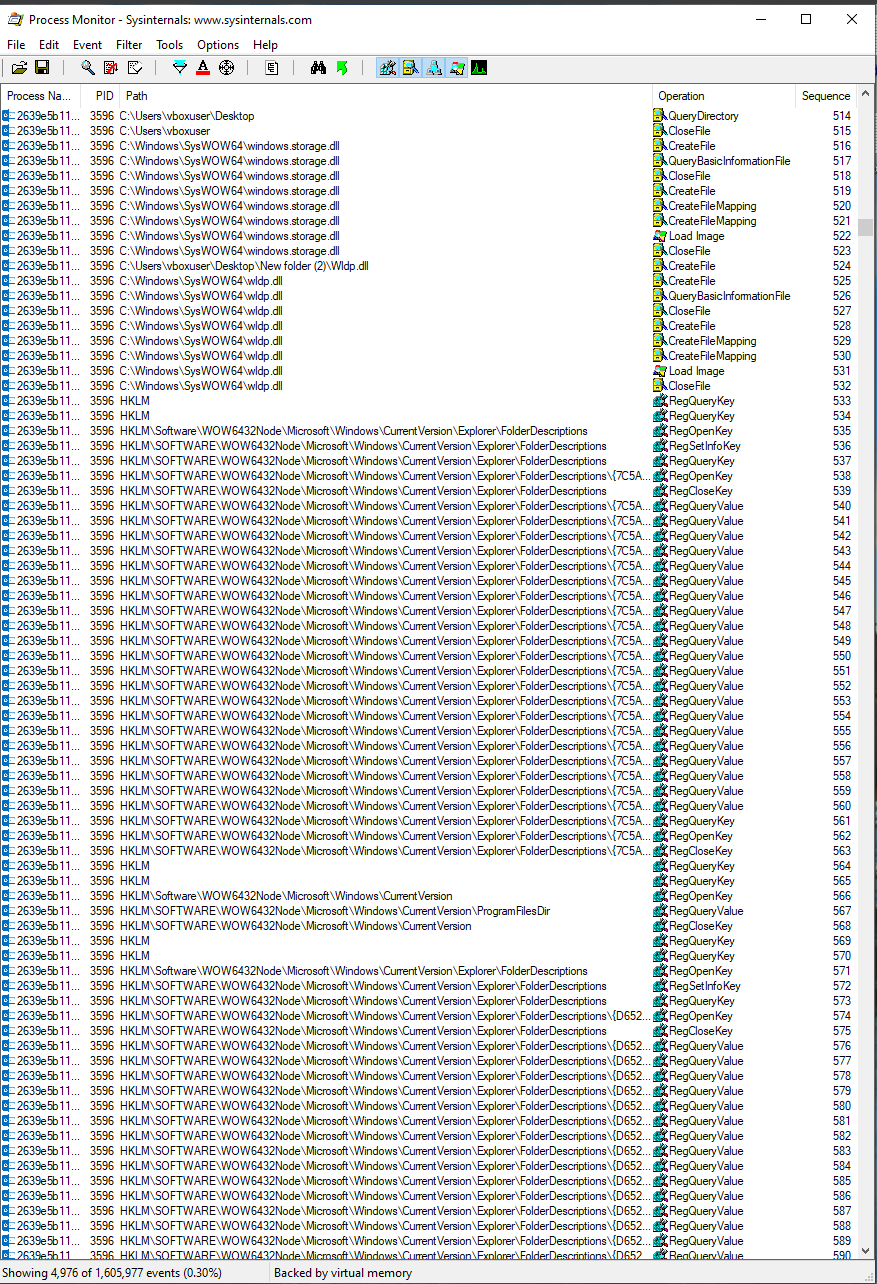
**Figure 28.**

****

**Figure 29.**

**.**

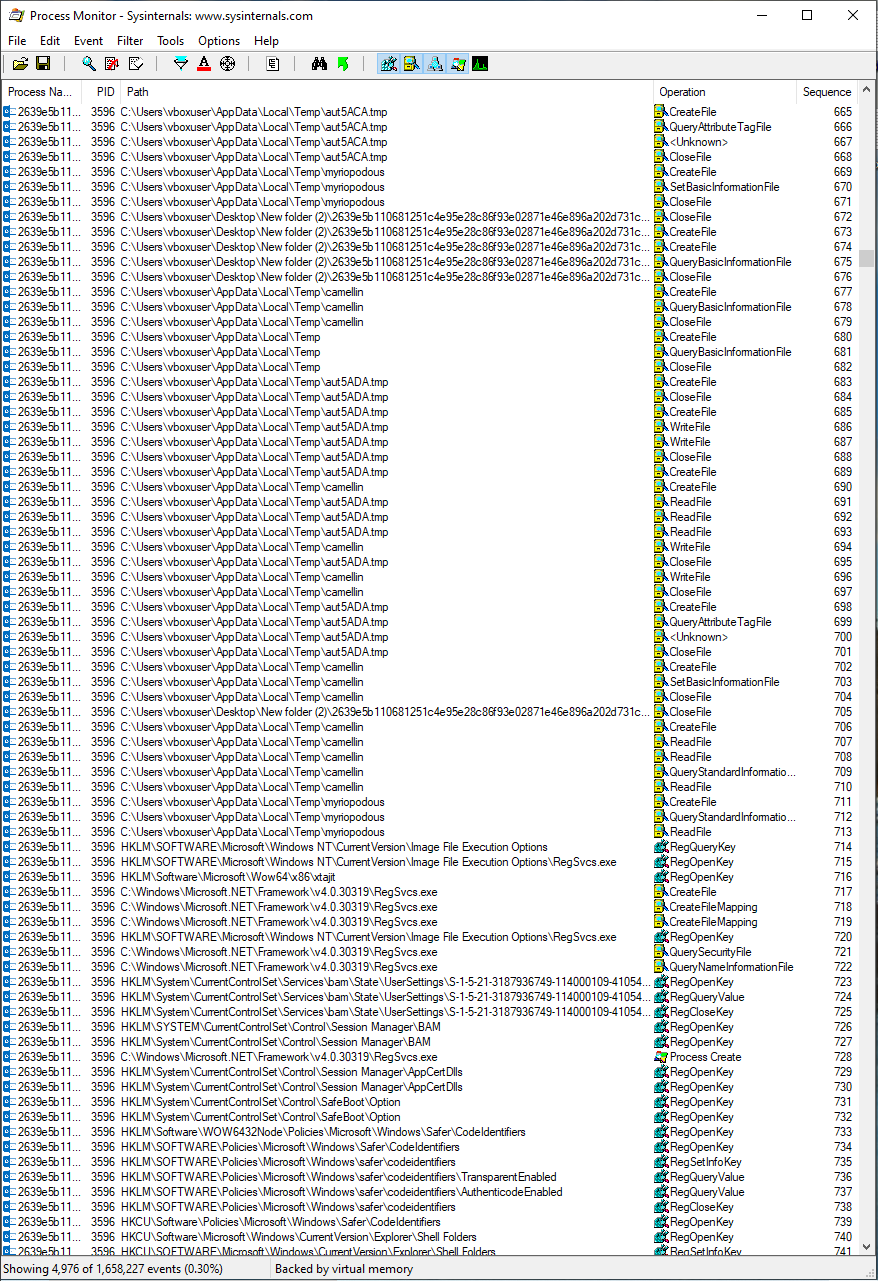
**Figure 30.**



**Figure 31.**

****

**Figure 32.**



**Figure 33.**



**Figure 34.**

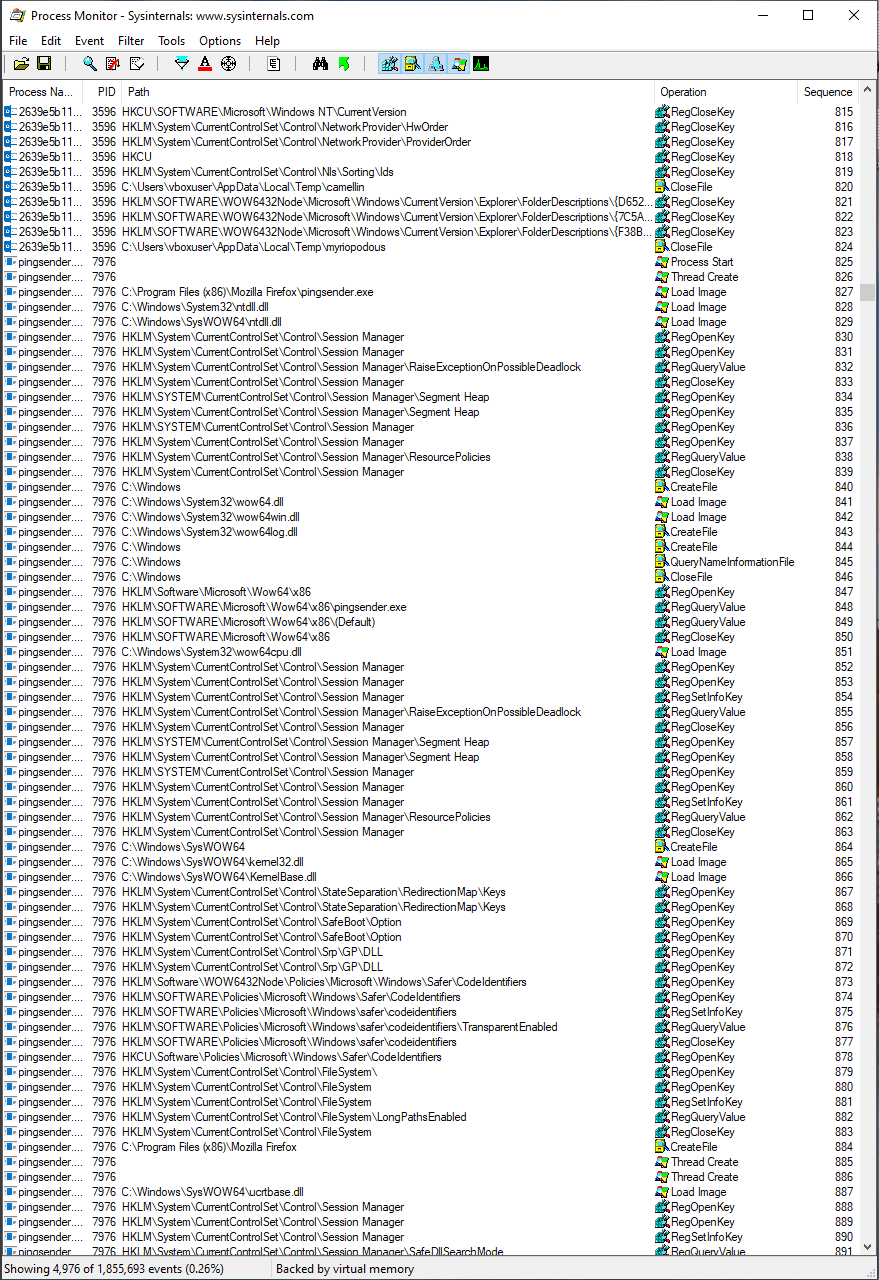
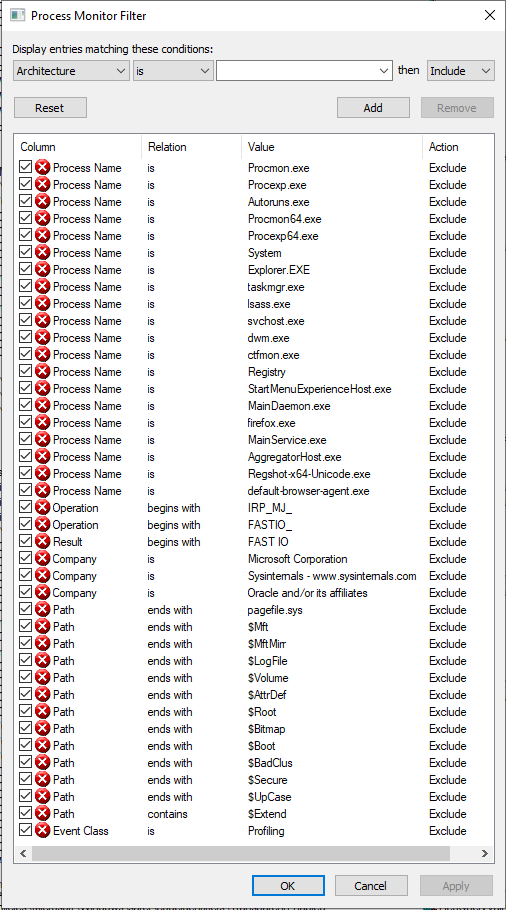


Figure 35 contains the filters used to effectively block out all unnecessary processes which allowed us to generate figures 23 to 34.

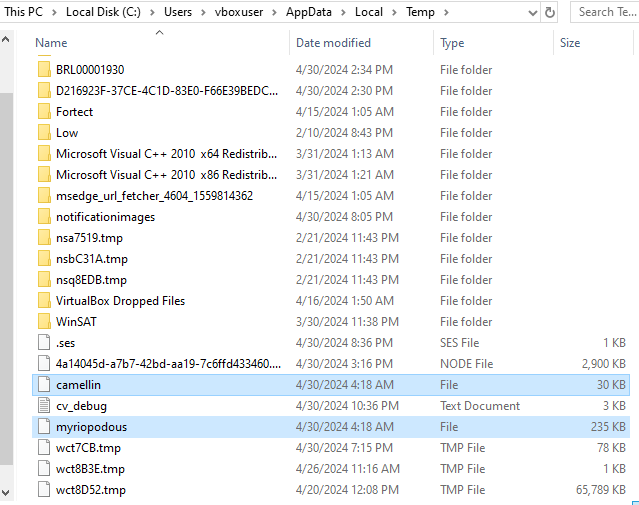
**Figure 35.**



Figures 23 to 31 show us the chain of operations executed after running the executable. In Figures 31 and 32 we finally see a reference to files. Specifically, two files called “myriopodous” and “camellin” located in “\AppData\Local\Temp”. The assumption is that the long list of executed operations is connected to these files which were created and written to. These files are likely to contain some sensitive user data or further instructions for the malware.

Now remember, we saw these files mentioned in VirusTotal in Figure 14 in the “Dropped Files” section. With only “camellin” being considered malicious, further solidifying the legitimacy of our findings. As we can see in Figure 36, the two files are present but in an unknown format. Furthermore, if we open them, they are in undecipherable encrypted format.

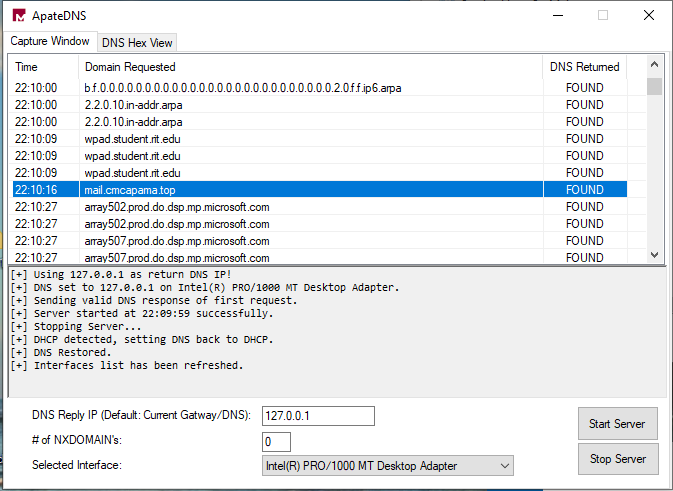
**Figure 36.**



**AgentTesla: ApateDNS & Wireshark**

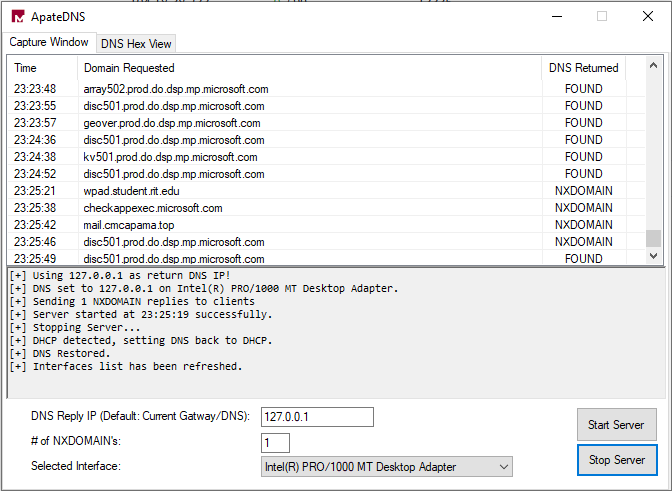
ApateDNS enabled us to learn how the malware interacts in terms of networking. When the malware executable is run it makes a DNS call to an FQDN. As we can see in Figure 37, we see that the call is made to “mail.cmcapama.top”. Now if we recall Figure 13, we can see that this FQDN is present in VirusTotals’ Contacted Domains section.

**Figure 37.**



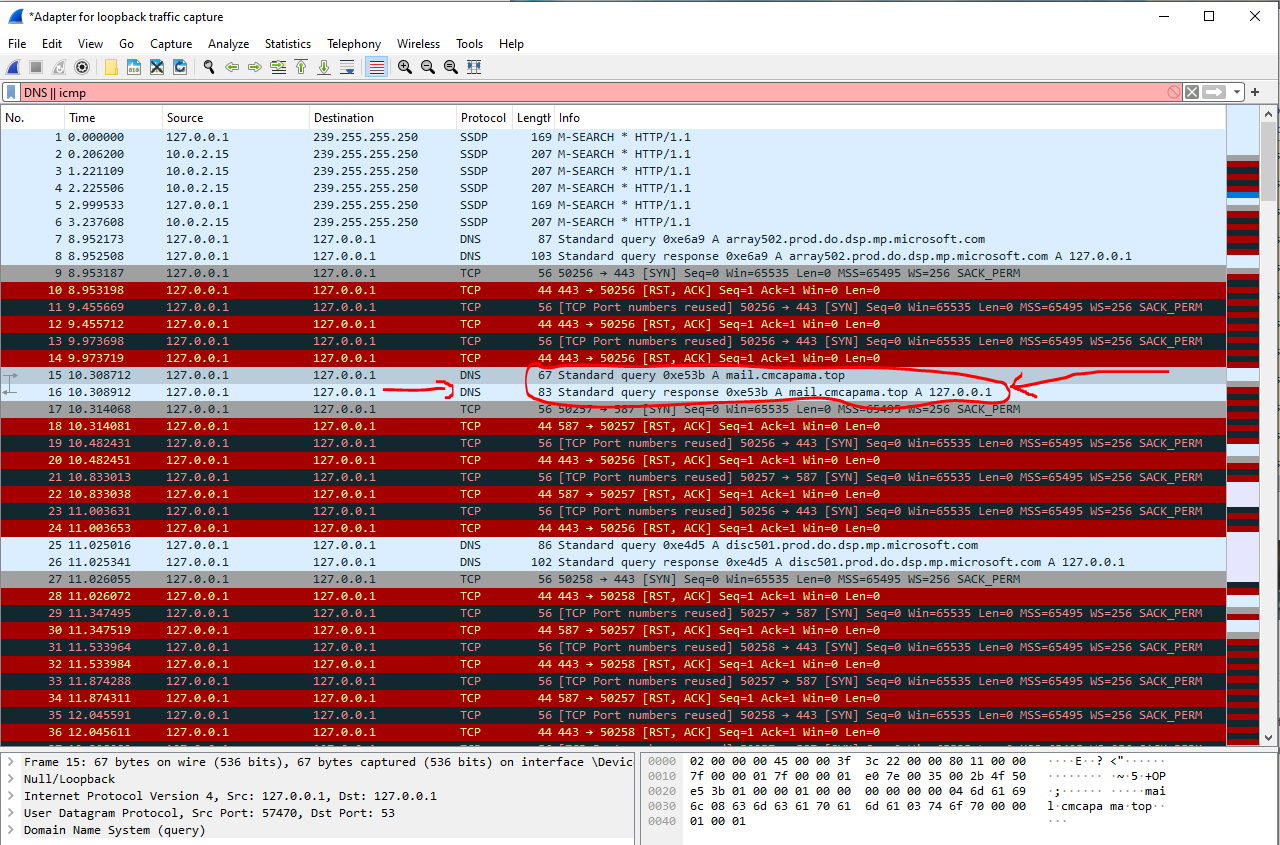
We also need to check if a secondary (backup) FQDN is present in the mechanics of the malware. To do this we’ll set the # of NXDOMAINs to 1. As we can see in Figure 38, there isn’t any secondary FQDN. Something fun to notice is a Windows mechanism is a call being made to “check.appexe.microsoft.com” right after the malware is executed as seen in Figure 38.

**Figure 38.**



Additionally, after we set our ApateDNS configuration, we can see these results in Wireshark when we observe DNS and ICMP packets as seen in Figure 39 with red arrows.

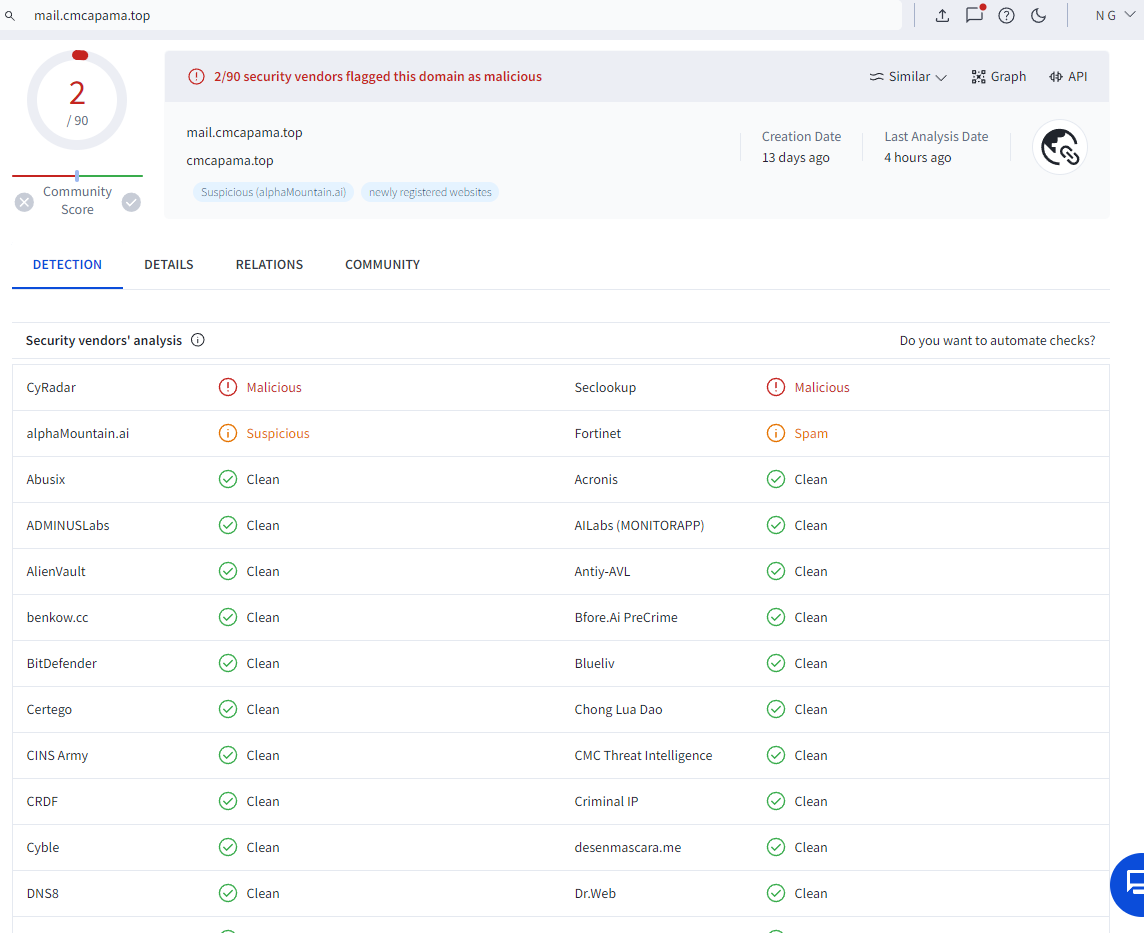
**Figure 39.**



**AgentTesla: Quickly revisiting VirusTotal to check our results.**

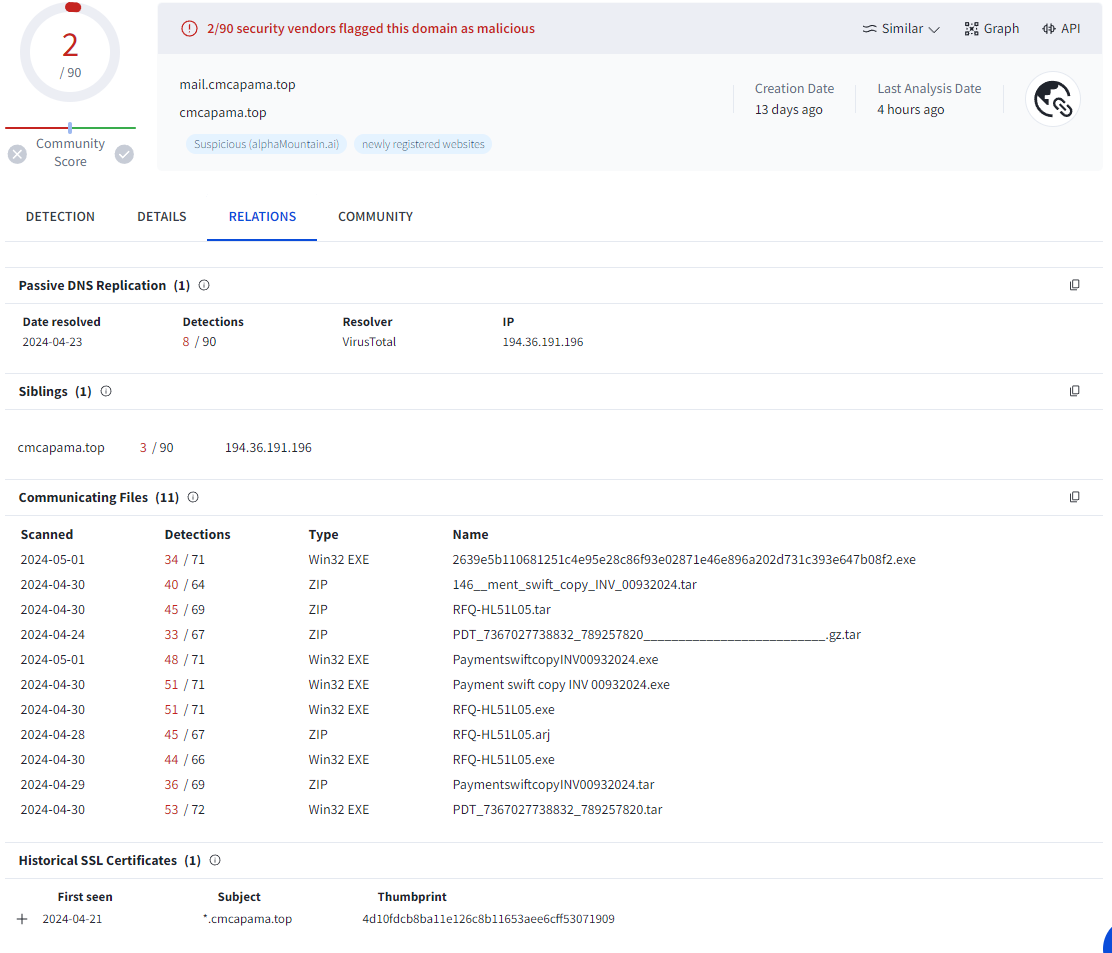
This section will serve to analyze the FQDN, its associated IP address, and the other malicious executables that connect to the same addresses (likely other variations of this malware strain derived from Agent Tesla). Figure 40 shows us all the other executables connecting to this FQDN. Amusingly, all of the executables are designated across the board as malware. However, the FQDN that this malware connects to, is only designated as malicious by two vendors. Again, indicative of a new strain of malware with a new Command and Control server not yet fully discovered.

**Figure 40.**



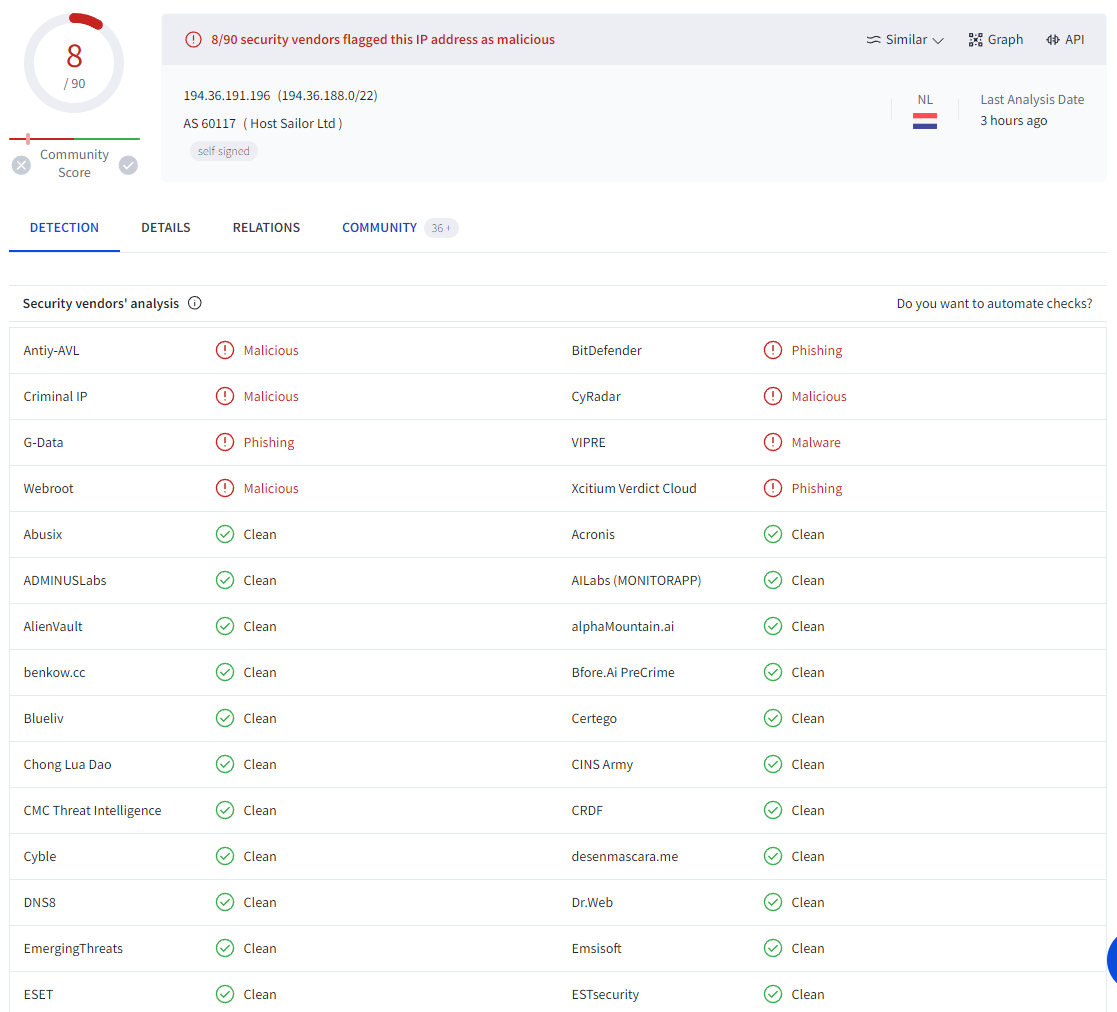
In Figure 41 we see some of the executable names that connect to this FQDN, one of them being INV 00932024.exe, if we recall this appeared in Figure 11 as an alias to our malware.

**Figure 41.**

****

Finally, we can also see that the associated IP address is flagged in Figure 42. However, again we see only a small number of vendors designating this address as malicious.

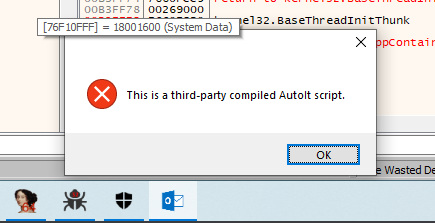
**Figure 42.**



**AgentTesla: x64 Debugger & IDA.**

During disassembly and debugging I started to have a hard time. I noticed some weird things occurring. During debugging, I received a weird error that never occurred when simply running the malware as shown is Figure 43.

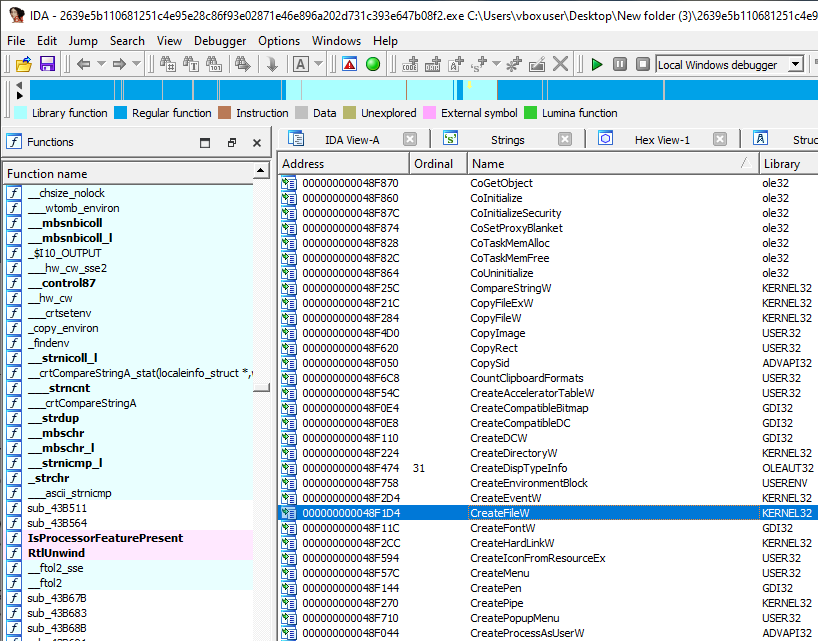
**Figure 43.**



This got me thinking that maybe I had some internal Windows safeguard mechanism enabled. So, I disabled every single option, and sub-option in Virus and threat protection, App & browser control, and Device Security. None of which changed the behavior of the debugged malware.

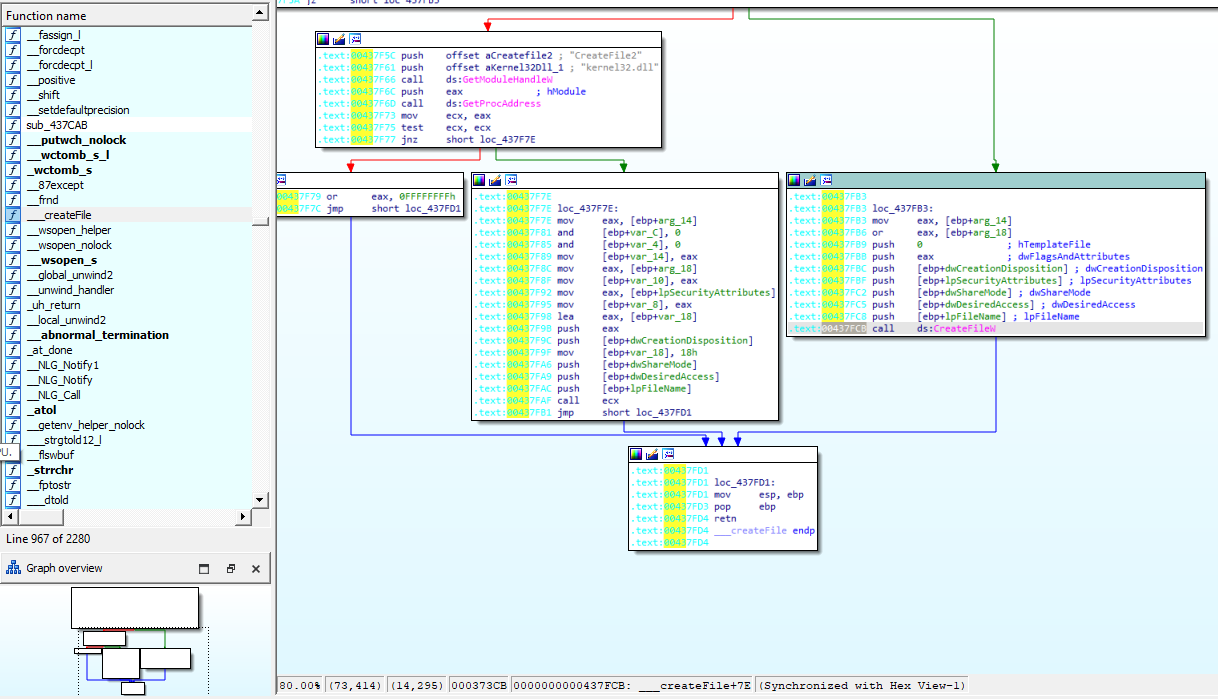
I also noticed that I would not be able to reach certain functions found during disassembly while debugging, such as the “CreateFileW” function as shown in Figure 44. This function was chosen based on the fact that this malware is a portable executable, the first action that it should logically attempt is to create another file or alter registry values possibly for persistence or redundancy.

**Figure 44.**

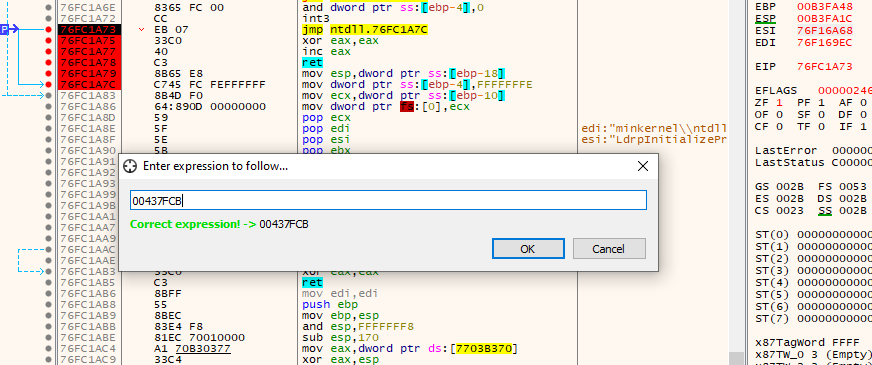


When we cross-reference this built-in Windows function, we can see it being used in another function called “\_\_\_createFile” as seen in Figure 45. This is where some of the issues started appearing. Upon investigation, we found that the address responsible for calling this function is “00437FCB”, as highlighted in gray in Figure 45 as well.

**Figure 45.**

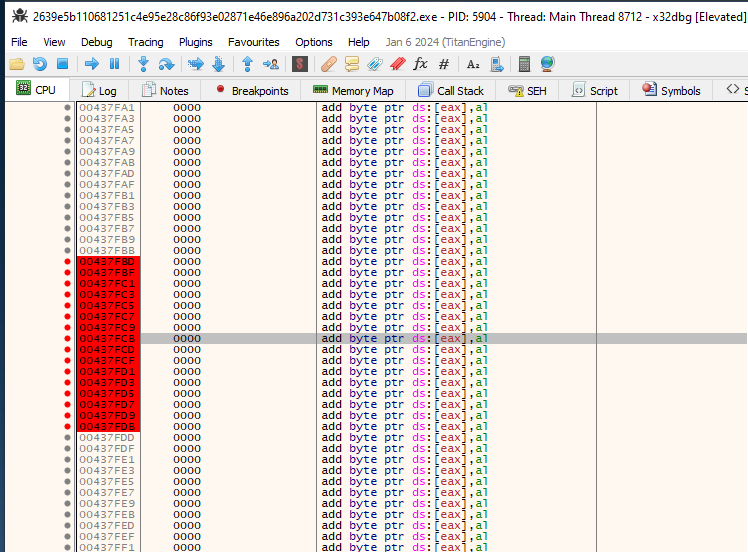
****

Upon attempting to reach the file in x64 Debugger, I noticed the address existed as shown in Figure 46, which was a good sign.



However, when we examine the instructions, we see some very unusual entries as seen in Figure 47. Not only are ALL of the instructions related to the “CreateFileW” function the same “add byte ptr ds:[eax],al”. The debugger never even reaches these instructions anyway!

**Figure 48.**



This got me thinking that maybe there are some safeguards built into the malware to protect itself from being disassembled or altering its behavior in a Virtual Machine as we learned in our course. And unfortunately, after some digging around on Google searching for “agent tesla in virtual machine”, I found two reasons why the malware was behaving irregularly.

As pointed out by Acronis Detection and Response, the malware checks whether it is running on a Virtual Machine or not by querying registry values (Acronis, Inc.). This is exactly what we saw happening in Figures 23 through 34, now making more sense. Secondly, the malware has safeguards built into it so that it doesn’t allow for clean debugging. It employs a “crypter” called AspireCrypt, which disrupts our process of debugging (Acronis, Inc.).

Regardless, some of our findings seem consistent with other vendors. The Agent Tesla variation mentioned by Acronis, seems to also point to a mail server just like ours did, just with a different address. This shows us that the source code for this malware must have been leaked, made public, or preconfigured for customers since the behavior is identical with only changes being made to addresses.

**Summary**

Agent Tesla is an extremely powerful tool as analyzed in this report. It is a Remote Access Trojan (RAT) that has many capabilities. Some of these include but are not limited to keylogging, stealing clipboard data, screenshot capturing, and webcam access. It encrypts its stolen data so that there will be no way for users to decipher what was stolen. The malware uses a mail server as its Command and Control, however, these are different from other strains of Agent Tesla.

Both variations that we saw of Agent Tesla (Acronis and ours) employ techniques for identifying whether Agent Tesla is being run on a Virtual Machine by querying registry data. As reported by Acronis, we saw that a Crypter is also utilized in this malware, making it highly difficult to detect and analyze.

**Removal Recommendations:**

If the malware was not detected by your anti-virus or the built-in Windows Defender and you suspect you have been infected with AgentTesla then I would highly suggest going with the zero-trust approach. With this approach, a clean install of windows should suffice. Additionally, it is safe to assume any passwords stored or typed recently on the machine are compromised (safe to assume browser passwords as well). Thus, rotating passwords for all major accounts will be crucial for security. However, should this not be a valid option, here are some steps you could use to potentially remove the malware:

1. Isolate the Infected System: Disconnect the PC from the internet to prevent its spread and communication to its command server.
2. Use good Antivirus Software: Using MalwareBytes Antimalware should be a good pairing to use with the built-in Windows Defender antivirus.
3. System Restore: If a restore point was made before running the malware, then consider reverting to that point.
4. Update: Using Windows Update in addition to any other anti-virus updates should improve detection and removal of the malware.
5. Change Passwords: Change all system, application, and website-related passwords while also enabling two-factor authentication (2FA).
6. Monitor System Traffic: As we saw, should a connection be made to a similar mail FQDN, then suffice to say that the malware is still present on the PC.

**References**

**MalwareBazaar:**

abuse.ch. "Malware Sample Entry for SHA256: 2639e5b110681251c4e95e28c86f93e02871e46e896a202d731c393e647b08f2." *MalwareBazaar*, 2024, <https://bazaar.abuse.ch/sample/2639e5b110681251c4e95e28c86f93e02871e46e896a202d731c393e647b08f2/> . Accessed 30 Apr. 2024.

**Acronis:**

Acronis, Inc. "68881: Acronis Detection and Response: Agent Tesla: Acronis Incident Report" *Acronis Cyber Protection*, 2024, <https://kb.acronis.com/content/68881> . Accessed 30 Apr. 2024.