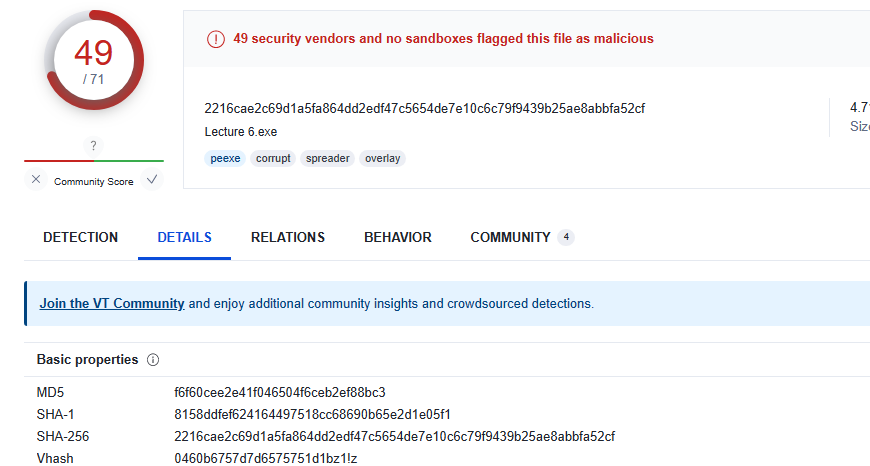
**Upload the file to**[**http://www.VirusTotal.com/**](http://www.virustotal.com/)**. Does the file match any existing antivirus signatures?**

**What is the hash of the file?**

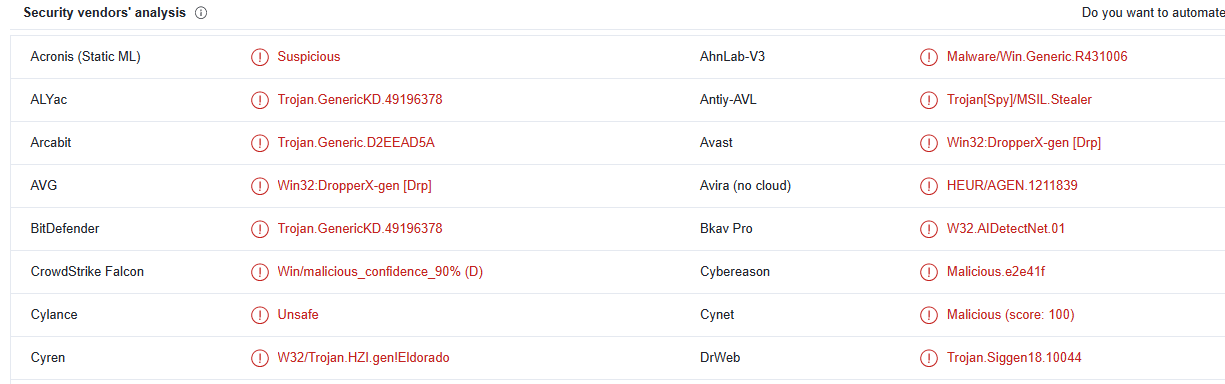
The file matches 49 of 71 existing antivirus signatures.

It has an MD5 hash of f6f60cee2e41f046504f6ceb2ef88bc3.

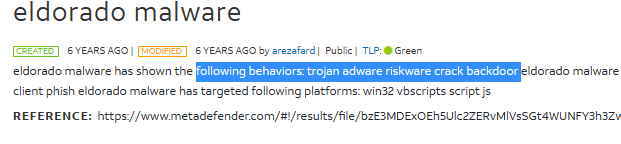


**What is this file known for?**

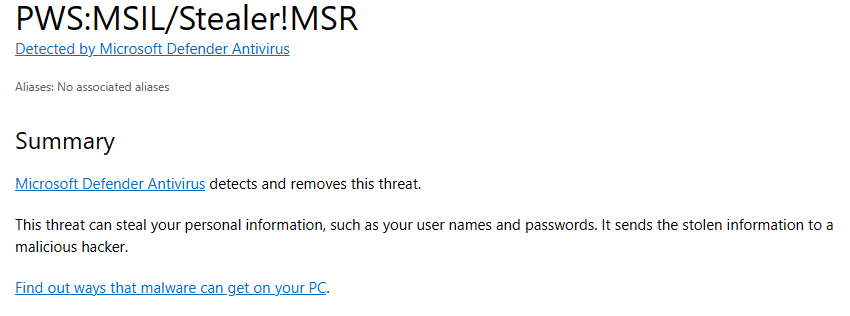
On VirusTotal, the security vendors who recognize this malware identify a few unique names, but generically the malware is a Trojan. We see multiple unique names in addition to Trojan, in particular “Eldorado,” “Siggen,” and “MSIL.Stealer.” (There is another possibility of what this malware is known for after inspecting the packing data. Scroll to “Are there any indications that this file is packed or obfuscated?” for more details).



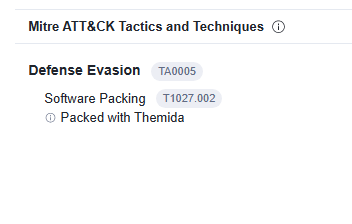
Alien Vault states that the Eldorado malware has behaviors of adware and functionality to create backdoors.



Microsoft states that MSIL.Stealer can steal personal information (like usernames and passwords) and then send the info to a malicious actor.



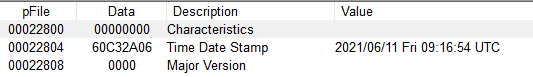
There were no detailed reports on siggen18. However, based on the naming conventions identified in VirusTotal along with the minimal behavior identified for the malware (only defense evasion via packing was stated. See screenshot below), the keywords that stand out in the security vendor naming conventions for the malware are “stealer,” “spy,” “dropper,” and “Eldorado” certainly suggest that the malware is spyware designed to steal information. The name “Eldorado” refers to the mythical city of gold, El Dorado, which suggests that the malware potentially targets financial information stored on the infected machine.



**Using the tools, we discussed in lectures 1 and 2, answer the below.**

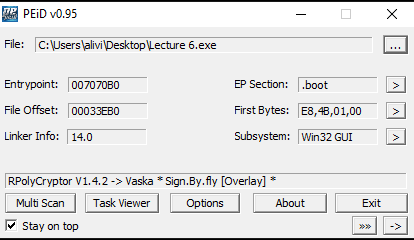
**When was this file compiled?**

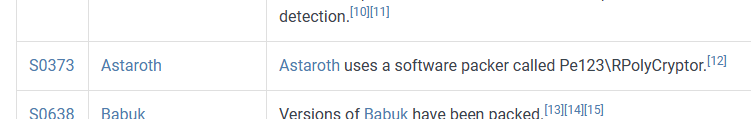
PEView states the malware was compiled on 11 June 2021 at 09:16:54 UTC.



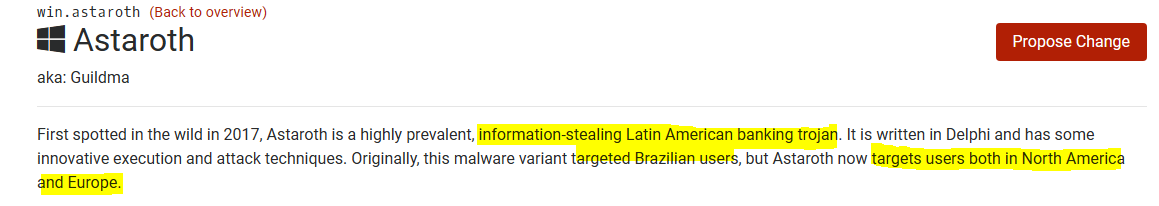
**Are there any indications that this file is packed or obfuscated? If so, what are the indicators?**

Yes. PEiD identified packing via RPolyCryptor which, according to MITRE ATT&CK, has been used by the Astaroth malware for packing. A Google Search of RPolyCryptor returned limited results, but Astaroth was almost exclusively referred to in the search results.

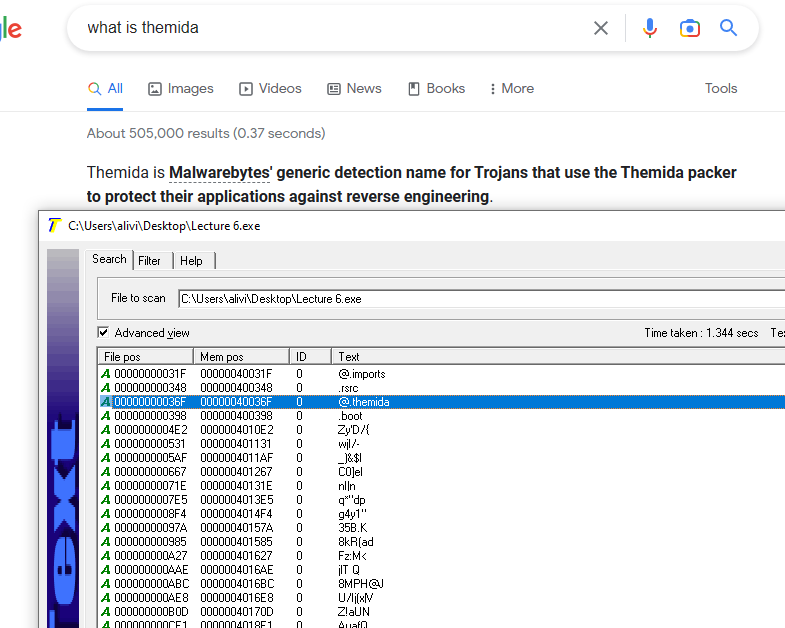




Since Astaroth was almost exclusively referred to in the search results of the packing module RPolyCryptor, Astaroth is another potential answer to the question, “What is this file known for?” The name “Astaroth” is in reference to the Great Duke of Hell in demonology. According to Malpedia, it is an “information-stealing Latin American banking trojan” that “originally…targeted Brazilian users”. This has multiple parallels to the previously-identified malware name “Eldorado” and the implications that it potentially steals banking information.

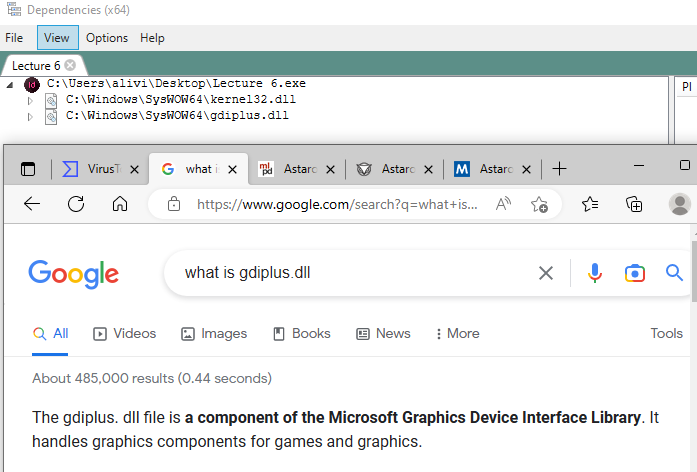


Additionally, it is clear that the malware is obfuscated based on a simple strings analysis. Save for a few lines within the output provided by BinText, most of the lines are 6-character gibberish. There is also a mention of “themida”, which is a packer used to protect applications from reverse engineering. This could be what is causing the obfuscation and was potentially used to pack the malware in addition to or in conjunction with RPolyCryptor.

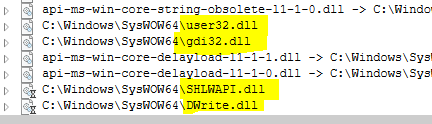


**Do any imports hint at what this malware does? If so, which imports are they?**

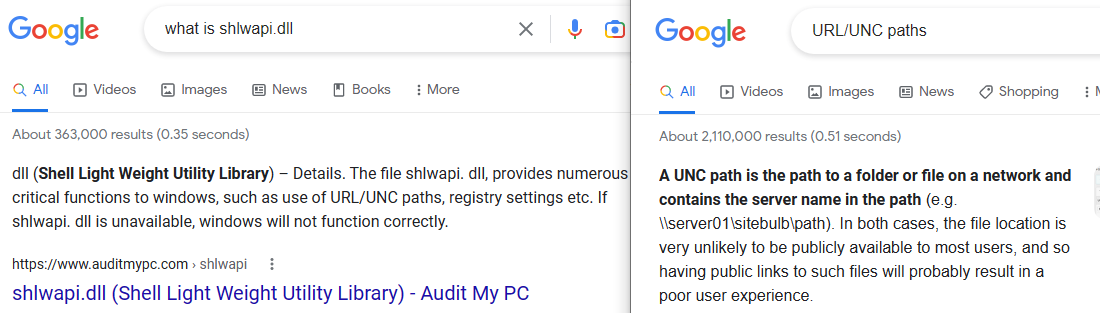
(NOTE: File imports are analyzed in the next question, “Are there other files or host-based indicators you could look for on infected systems?”) The two primary dll imports that the malware uses are kernel32 and gdiplus. Kernel32 will give the malware access to core OS functions and the ability to manipulate memory, files, and other hardware. Gdiplus “is a component of the Microsoft Graphics Device Interface Library which handles graphics components for games and graphics.” The gdiplus.dll suggests that some graphics manipulation will take place when the malware is run.



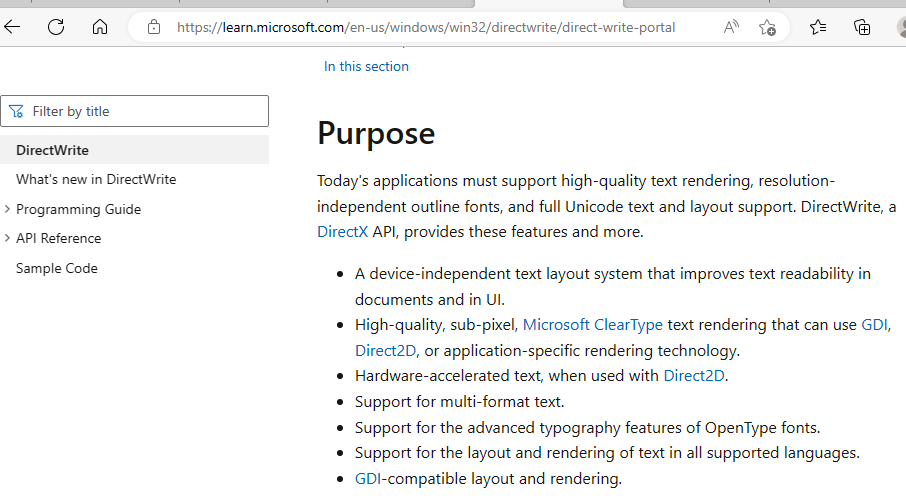
Some sub dlls within the gdiplus.dll import are user32.dll, gdi32.dll, SHLWAPI.dll, and DWrite.dll. It makes sense that user32.dll and gdi32.dll are subcategories within the gdi32.dll import because they contain user interface components and functions for manipulating graphics.

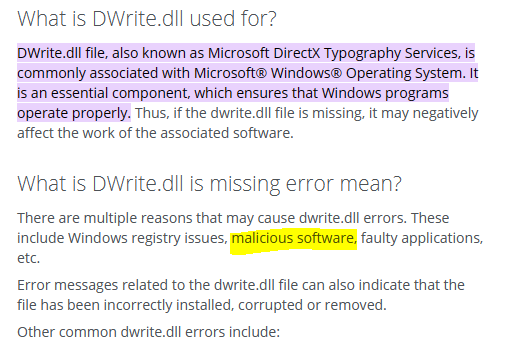


SHLWAPI.dll is an acronym for “Shell Light Weight Utility Library” that manages URL/UNC paths and registry settings (“a UNC path is the path to a folder or file on a network and contains the server name in the path”). This suggests that the malware potentially drops public links within folder for access, most likely for the attacker.



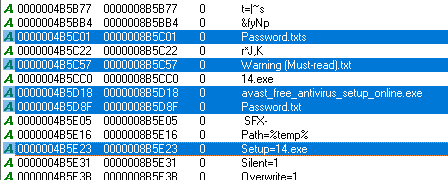
DWrite.dll stands for “Microsoft DirectX Typography Services” which is a critical service for Windows systems and essentially provides text-readability in documents and the user interface. Since the malware calls this .dll as an import, it most likely has the purpose to negate the usability of this DWrite.dll and make it difficult or impossible to use regular applications. Issues with DWrite.dll is potentially an indicator of malware compromise.



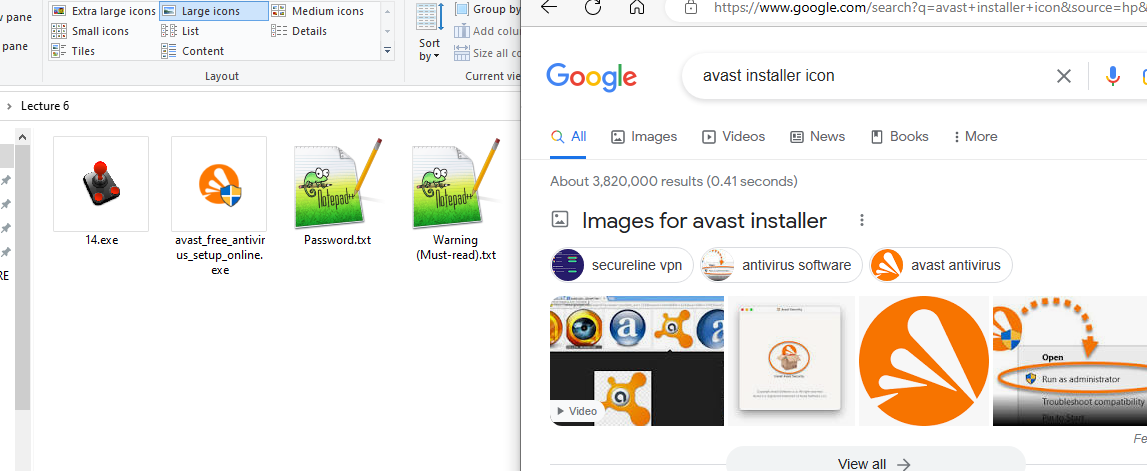


**Are there other files or host-based indicators you could look for on infected systems?**

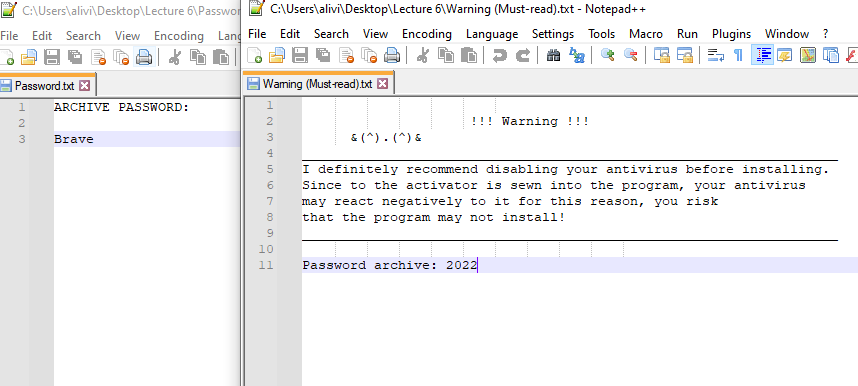
Although most of the strings outputted by BinText were obfuscated, there are a few file imports that potentially suggest what the malware does and what to look for on an infected system. We see two executables of “Setup=14.exe” and “avast\_free\_antivirus\_setup\_online.exe.” The setup executable most likely is an installer that delivers an additional malicious payload. In particular, the antivirus executable suggests that the malware’s effects will be obvious to the user which would cause them to click on a tempting solution and, as a result, download more malicious software. The “Setup=14.exe” file will most likely be stored in the user’s temp folder as indicated above the line where it is declared.



After running UniExtract on the malware, there were four files that were extracted identified in the screenshot above, but not Password.txts. The file that was not extracted does not have a valid file extension and contains an extra “s” at the end of the valid extension, “.txt”. This could either be a typo by the malware creator or a hidden file to store stolen passwords for exfiltration. The antivirus executable has a legitimate-looking icon and matches the installer icon for Avast Antivirus. The “Setup=14.exe” identified in the screenshot above simply changed to “14.exe” and has an icon of a joystick peripheral, tempting the user to click on a “game” and causing further infection.



The two .txt documents give us a password of “Brave,” but it is unknown what it could be used for. The “Warning (Must-Read).txt” document is most likely a faulty read-me created by the malware author so the malware can effectively deliver the payload without interference.



Based on these findings, the file-based indicators would be 14.exe, Password.txt, Warning (Must-Read).txt, and avast\_free\_antivirus\_setup\_online.exe. Notably, the user’s temp folder would be the likely place to look for 14.exe. It is unknown where the other files will place themselves until the malware is run. It is also unclear as to where the Password.txts document will be placed (if at all), but it would most likely be in a discreet location on the machine.

**BEFORE you run this malware, would you consider this file malware based on your findings? Exclude your virus total findings.**

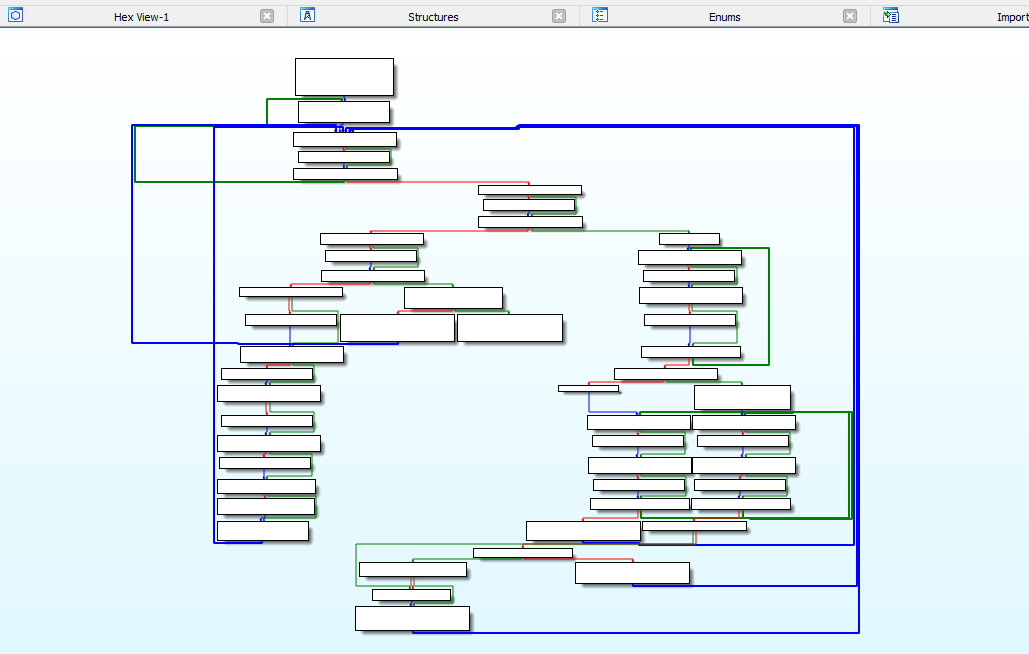
Based on these findings, I would definitely consider this file to be malware of a certain variety of the Eldorado or Azeroth malware. The giveaway is the packing tools used and the Warning (Must-Read).txt document telling the user to disable their antivirus. The vast majority of legitimate programs that are well-vetted would not tell users to take such action. Although obfuscation is standard practice for large companies in order to protect source-code intellectual property, the packing algorithms and association with malicious software is extremely concerning and point towards malware.

\****ONLY IN THE VM\****

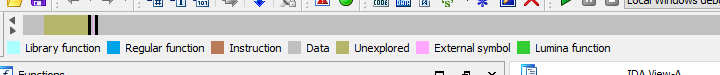
**TURN OFF NETWORKING!!!!!**

Using the tools, we discussed in lecture 6, answer the questions below and provide screenshots.

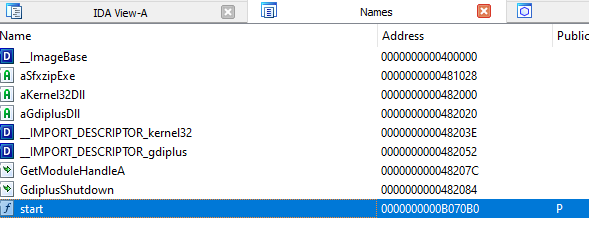
**Provide a screenshot of the graph view of the program.**



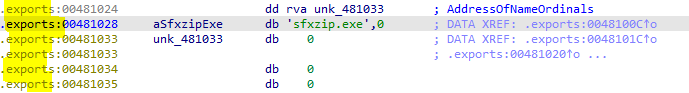
**Provide a screenshot of the navigator bar (the color-coded bar showing library functions, regular functions, code, data, etc.)**



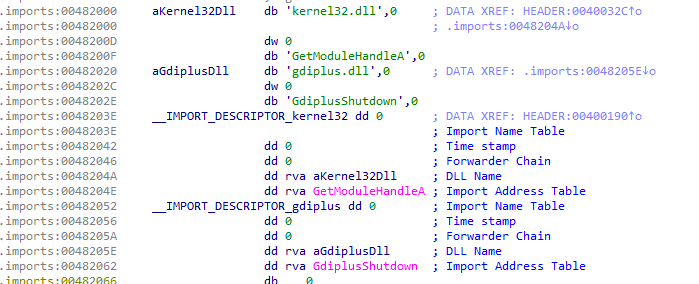
**Open the Names Window and view every function, Library, Code, String, Dara, and Linked Function. Provide an analysis of which of these are “interesting.”**



The first name that stood out was “aSfxzipExe”. When I double-clicked on it, there wasn’t anything too interesting except that the section name was “exports”. No functions called sfxzip.exe. The most interesting part of this were the characters “SFX”, a common abbreviation to “special effects”.

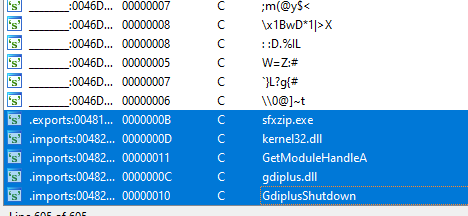


The other four interesting names are the two main .dll imports that were garnered from the static analysis in Discrepancies: gdiplus and kernel32. But these two are formatted as: \_\_IMPORT\_DESCRIPTOR\_<dllName>. When navigating to both of these functions by double-clicking on them, they perform the same commands by declaring variables of type double that serve the purposes of importing the name table, storing the time stamp, the forwarder chain, and the dll name, and importing the address table. There weren’t any functions that called these variables.

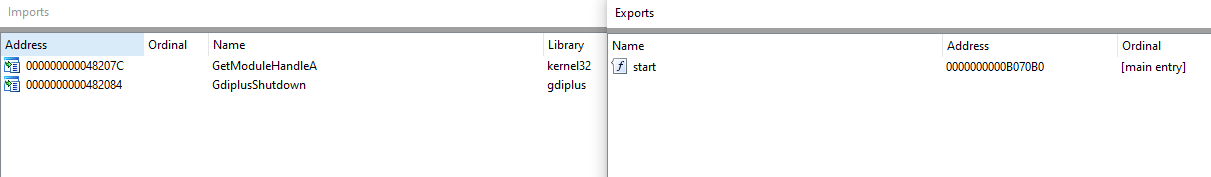


**Open the Strings window and provide an analysis of which ones are interesting and why.**

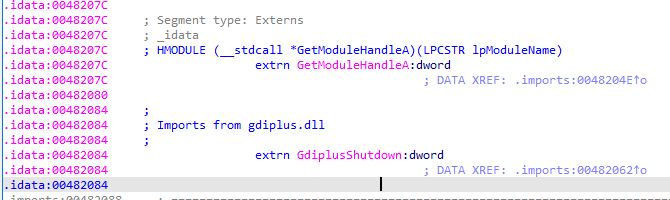
The strings window provided much the same as BinText, mostly gibberish except for the inclusion of the items from the Names window at the bottom of the strings window with their appropriately-mentioned import or export functionality.



**Open the Imports and Exports windows and provide an analysis of your findings.**



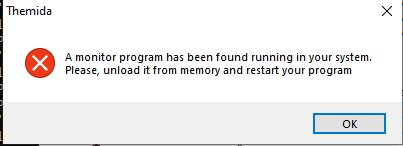
The imports window shows GetModuleHandleA and GdiplusShutdown with respective libraries of kernel32 and gdiplus. Exports shows the start function. This makes sense since the Names window showed both of these as imports with the symbol of a right arrow. When double clicking on them, they navigate to the same segment location and show that they call an external module or function (the segment type is “Externs”).



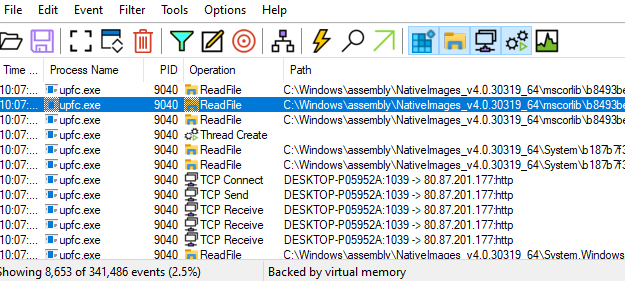
It is interesting that the “start” function is labeled as an export when it wasn’t identified as one in the Strings window. It’s also interesting that sfxzip.exe is not in the exports window when it WAS identified as an export in the Strings window. However, it makes a bit of sense because sfxzip.exe is a string type (based on the Names window).

**Run the file and record your observations. This malware may require you to reset your box a few times, this is nasty malware, don’t enable networking, and run fakenet.**

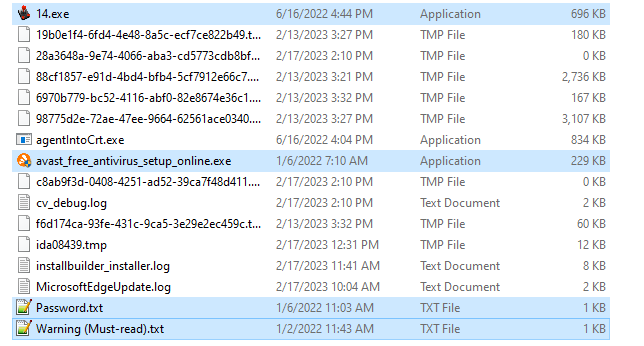
When the malware was initially run with procmon, process explorer, and Fakenet running, an error window popped up saying that there was an error. I closed all the programs and still received the error message. I then restarted the machine and ran the file without any monitoring software up and I did not receive an error message. After some trial and error, it appears that procmon makes it so that the program cannot run as a way to mask its true intentions. Even after closing procmon and running the malware, it would still show the error window. After finding this out and being unable to capture data using procmon, the file was run with Fakenet and Process Manager actively monitoring as well as regshot.



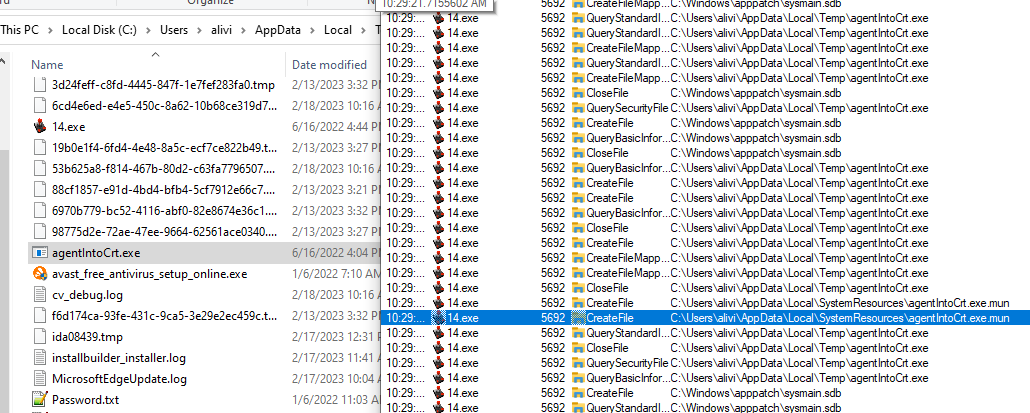
After running the malware again without procmon and then quickly initiating a capture with procmon, the application ran without issue and Fakenet showed an attempted connection. Looking in procmon, the IP address identified by Fakenet showed a new process name titled “upfc.exe” and showed over 8,000 registry, file, network, and process activity events. Upfc.exe is short for the “Updateability From SCM” process and is a native Microsoft Windows process.



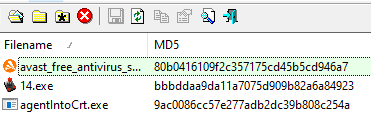
I went to the Users/<username>/AppData/Local/Temp folder and noticed the four files identified earlier when the malware was unpacked with UniExtract. This location was identified during the strings analysis.



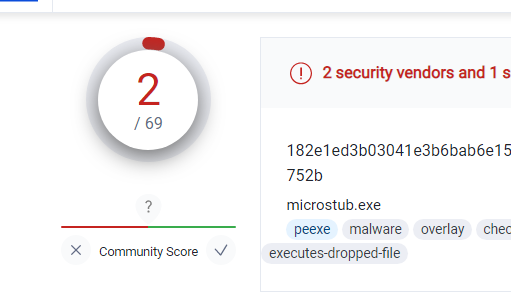
After running 14.exe, it created a new .exe file titled “agentintoCrt.exe” but did not perform any network connections. The contents of the .txt files created by Lecture6.exe did not change.



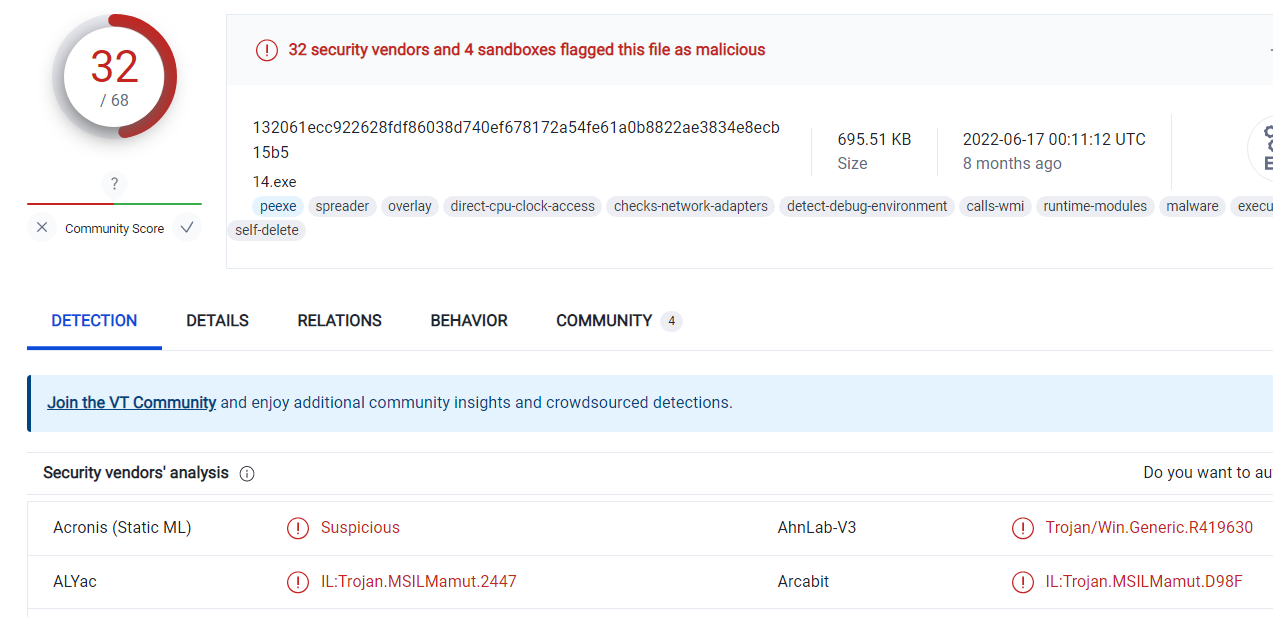
I gathered hashes from all three executables and searched VirusTotal.com for any other matches.

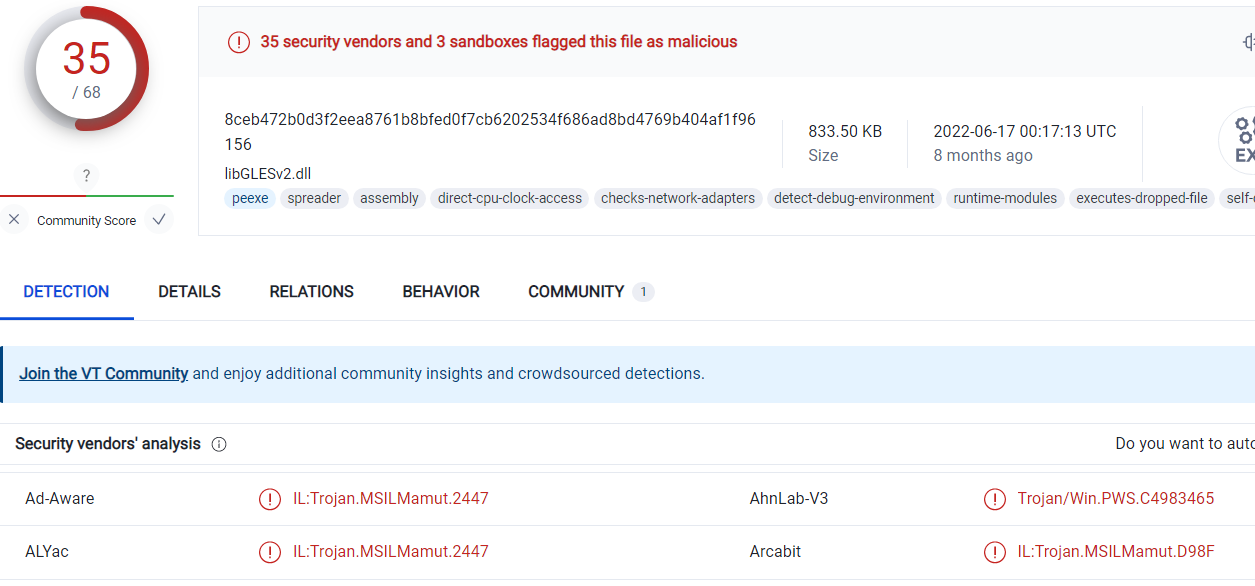


The avast executable found 2 of 69 matches for malware and had a matching signature for a differently-named executable of microstub.exe. MD5 hash: 80b0416109f2c357175cd45b5cd946a7. Running the avast executable didn’t do anything other than bring up a window with a progress bar that never moved, most likely due to it requiring a valid internet connection.



14.exe found 32 of 68 matches for malware. MD5 hash: bbbddaa9da11a7075d909b82a6a84923.

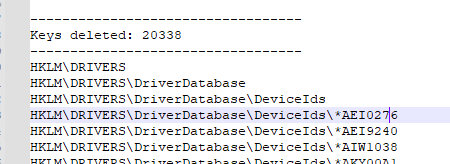


agentintoCrt.exe had 35 of 68 matches for malware but the hash matched the file name libGLESv2.dll. MD5 hash: 9ac0086cc57e277adb2dc39b808c254a. Notably, 14.exe and agentintoCrt.exe returned the same malware types/names of Trojan.MSIL.Mamut.2447 from the security vendor ALYac. 

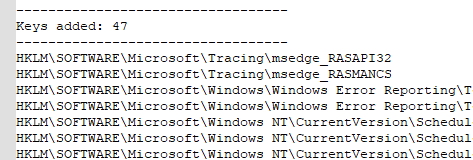
**Registry Comparisons**

**Lecture6.exe Regshot**

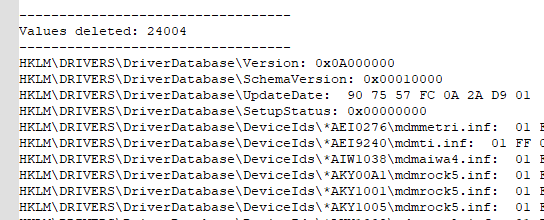
The regshot comparison from running only Lecture6.exe showed 20,338 deleted registry keys from the Local Machine hive in the \DRIVERS\DriverDatabase directory. Deleted keys of note include numerous keys in the \DeviceIds\PMCIA (Personal Computer Memory Card International Association) directory. Other deleted keys were in subdirectors that relate to Smart Card Filters, Device ID changers, Secure Digital device Identifiers, and other network interface keys.



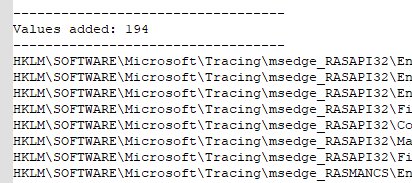
47 Keys were added, 42 of which were located in the TaskCache.



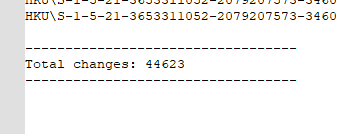
24,004 key values were deleted, the overwhelming majority in the DRIVERS\DriverDatabase.



194 values were added. Notably, values were added in SOFTWARE\Microsoft\Tracing\msedge which enables file tracing, a feature normally disabled by default.

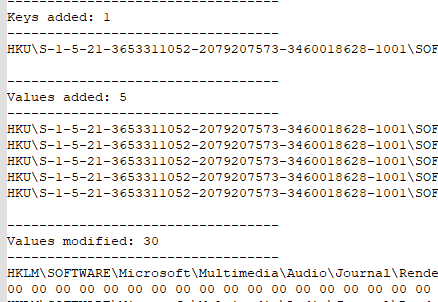


In total, Lecture6.exe made 44,623 changes to the registry.



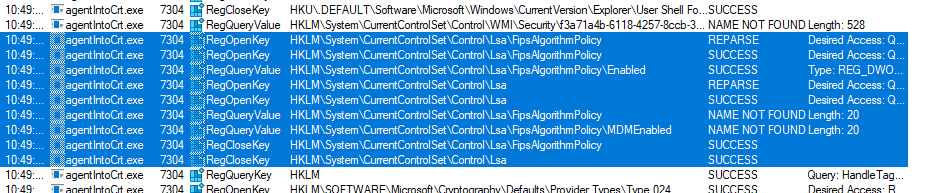
**14.exe Regshot**

14.exe added 1 key, 5 values, and modified 30 values in the registry, but nothing that stood out.

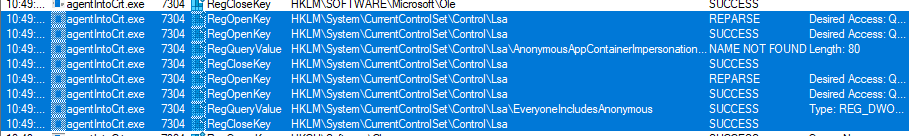


**agentCrt.exe procmon and Regshot**

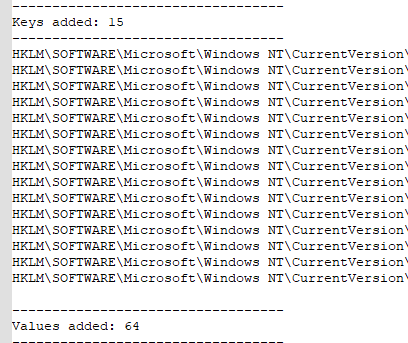
Running into agentCrt.exe displayed a lot of file-searching and mapping behaviors as well as lots of registry queries. Notably in the registry, it attempted to access the LSA (Local Security Authority) registry. Most crucially was it was attempting to modify the Federal Information Processing Standards (FIPS) Algorithm Policy which “is a set of government standards that define how certain things are used in the government” (https://knowledge.civilgeo.com/knowledge-base/disabling-fips-complaint-encryption-on-windows/#:~:text=FIPS%20stands%20for%20%E2%80%9CFederal%20Information,Home%20version%20of%20Microsoft%20Windows).



In the same LSA registry, it attempted to query the “EveryoneIncludesAnonymous” key which is normally configured to prevent anonymous users from having the same rights as the “everyone” group.

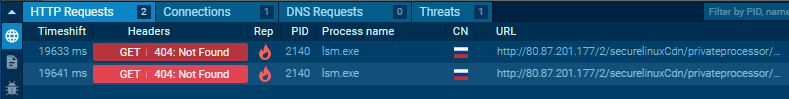


Regshot showed 15 added keys, 64 added values, and 29 modified values, but no modifications were of note.

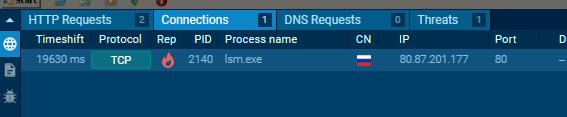


**What network-based indicators could be used to find this malware on infected machines?**

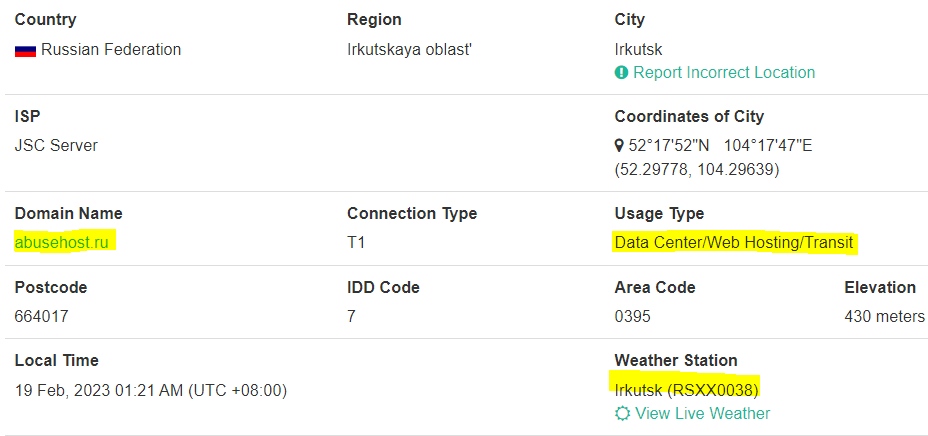
When the malware was placed into a sandbox on app.any.run, there were two HTTP requests to IP address 80.87.201.177 by the process “lsm.exe”. The country these IP addresses reside in is Russia but appear to be inactive.



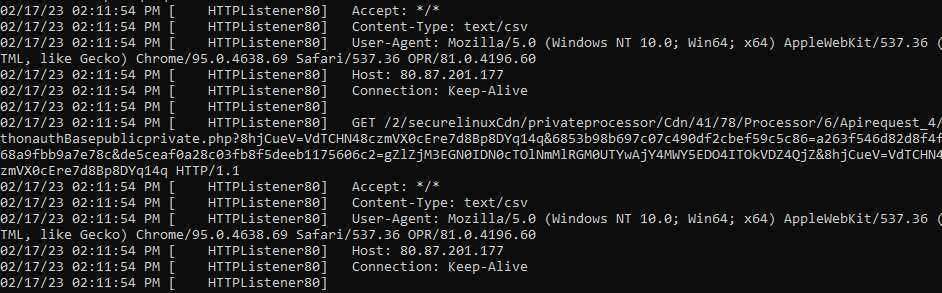
However, it appears that a TCP connection was made or attempted by lsm.exe to the same IP address over port 80.



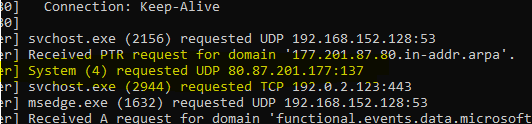
A quick search showed that the IP address of 80.87.201.177 located in the Russian city of Irkutsk and the IP is a Data Center.



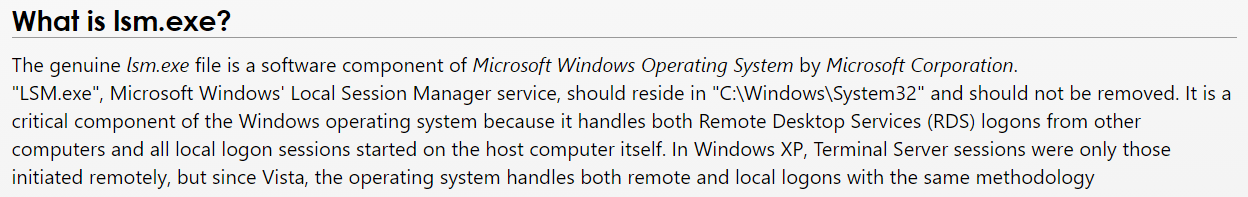
The same results were detected in Fakenet with the connections being kept alive. These connections sporadically reappeared throughout a lengthy Fakenet capture and content types were either text/html, text/csv, text/plain, application/json, text/css



The same IP address also requested UDP.



Moving back to lsm.exe identified by app.any.run, it is a piece of Microsoft software that is an acronym for “Local Session Manager” and handles remote desktop logons and “and all local logon sessions started on the host computer itself”.

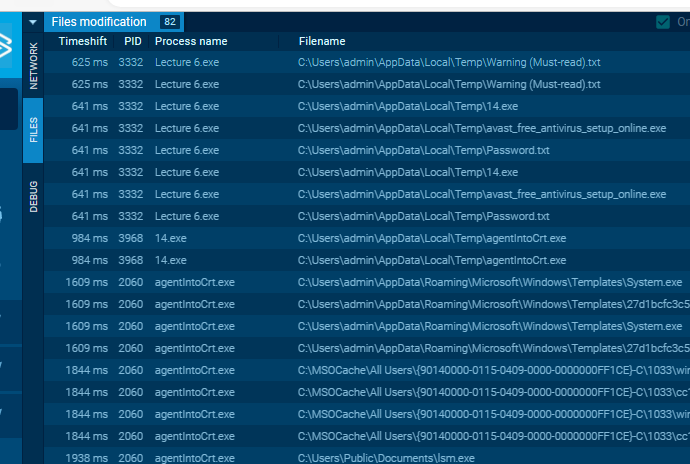


**What would you guess is the purpose of this file?**

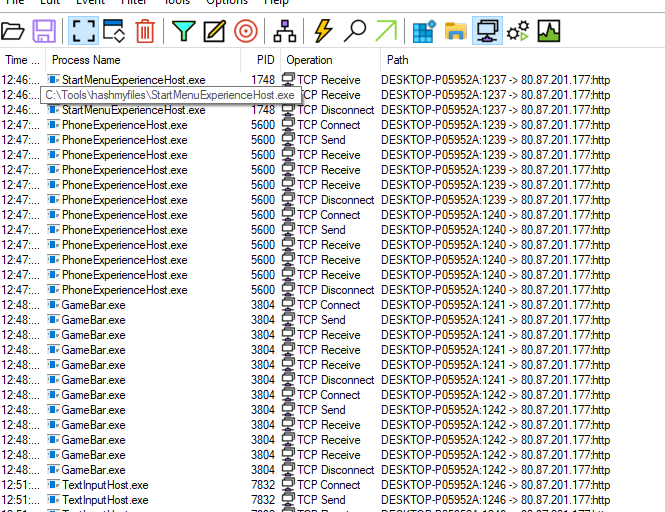
First, a breakdown of my observations: We have a Russian IP address, attempted modification of security settings (notably the Federal Information Processing Standards), constant attempts to connect over TCP and UDP to a Russian host, deletion of numerous device drivers registry keys, and use of Local Session Manager which handles remote desktop logons and “and all local logon sessions started on the host computer itself”. This tells me that this software was most likely developed by Russian state actors to infect government computers and exfiltrate data to the Russian government. It is a tool to conduct cyber espionage/warfare against those who are unlucky enough to download it.

**What programs do you see open, what is the malware trying to do?**

app.any.run detected 82 file modifications and as previously anticipated based on the static analysis, files were created and placed in the temp folder.



Additionally, procmon identified PhoneExperienceHost.exe, Gamebar.exe, StartMenuExperienceHost.exe, TextInputHost.exe, ctfmon.exe, sgrmbroker.exe, and dllhost.exe as attempting to send and receive to the IP address of 80.87.201.177. PhoneExperienceHost is used for VOIP connections. GameBar.exe is used for gaming, but has screen capture and screen/audio recording capabilities.



Again, based on the analysis conducted of the massive amounts of registry edits and constant pulse or “heartbeat” of attempted connections to a Russian host, I believe that this malware is most likely developed by Russian state actors to infect government computers and exfiltrate data to the Russian government. The fact that it is using legitimate programs tells me that it is cycling through them in an attempt to find a vulnerable vector to exfiltrate information. Or, because it is attempting to also access programs that provide audio capture, it is trying to find an open microphone to eavesdrop. It is a tool to conduct cyber espionage/warfare against those who are unlucky enough to download it.

**Now, how often would you be able to run this VM, excluding snapshots?**

I would still be able to run the VM, but would not feel comfortable doing so (especially with an active network connection). The malware didn’t affect any usability, but the likelihood of data exfiltration and eavesdropping exceeds my risk appetite.

**What would your recommendation be to Management? Do we need to stop generating revenue and cleaning, or can we go on and clean as we go?**

There are two possible scenarios of what kind of management I would make my recommendation to: I work for a private company, or I work for a government agency.

For a private company, my recommendation to management would be to add the IP address of 80.87.201.177 to the inbound and outbound block list in our enterprise firewall. Furthermore, I would also recommend blocking inbound and outbound connections from Russian domains due to the ongoing conflict in Ukraine and Russia’s “displeasure” with the US supplying arms to Ukraine. To caveat this blanket-blocking of Russian domains, we would need to consult with our legal department to see what restrictions apply to us based off of sanctions the US Government imposed on Russia and how it would affect any business partners we have in Russia. I would scan the network for any instances of the identified hash values and immediately remove those files from any and all systems. I would add those signatures to our IPS-es and IDS-es. I would also report this to intelligence agencies as potential state-sponsored malware.

For a government agency, I would immediately report this malware to my superiors as potential state-sponsored malware. This malware would be especially concerning if it was found on a SIPRNet as that network protocol is used to exchange information classified as SECRET, but still a concern if found on a NIPRNet (protocol used for Unclassified information). I would consult with our incident response manual and immediate supervisor to determine the best course of action to contain and eradicate the malware, as well as performing threat attribution. An additional concern is the potential that eavesdropping occurred using computer system microphone hardware and the potential security breach this entails.

**Can you clean the system, and if so, how would you do it?**

I don’t believe that I could clean this system. There were simply too many registry keys deleted and a full wipe and reinstall of the OS would be necessary.

* Go to two other student’s posts and observe their findings. Post if you agree or disagree with the results.
  + Did they find something using a new technique, and if so, would you use this next time?
  + From a business perspective, if you were both being paid at the same rate. Would your analysis be more cost-productive and achieve the same results?
    - Suppose you were working on this malware to see if this could be allowed in your organization. Did your analysis provide enough detail to make this determination?

Chris,

I am surprised that PEiD on your system didn’t pick up RPolyCryptor as a packing utility as mine did, but like you I also found Themedia as a packer. I am unsure what hypervisor you’re using for your VM, but in VMWare Workstation, you can go to “edit virtual machine settings”, click on “network adapter”, and then change select the bubble that says “Host-Only”. This solved my issues with running Fakenet. Overall, great analysis! I went down a pretty deep rabbit hole with this one after realizing it connected to a Russian IP. I recommend reading my analysis…found some interesting stuff (I think).

Adam,

Like Chris, I saw that when you placed the malware in PEiD did not show any output but mine did. I tried putting it through again to recreate the results, and also found that nothing showed up. However, if you click on the arrow in the very bottom-right hand corner and select “hardcore scan”, you can see the RPolyCryptor packing that I found. Additionally, I am unsure what hypervisor you’re using for your VM, but in VMWare Workstation, you can go to “edit virtual machine settings”, click on “network adapter”, and then change select the bubble that says “Host-Only”. This might solve your issues with Fakenet. If you continue to have problems with Fakenet for your network analysis, you can try putting the malware into an online sandbox like app.any.run (I really like this one).

Overall, I agree with your analysis and your formatting made it very easy to read. You clearly put a lot of effort into it. Although I am sure that you reverted to a previous snapshot on your VM after running the malware, I agree with your analysis that you could continue to run the machine…but don’t think that you could do so safely. There were a metric crap-ton of registry keys that got deleted and there were indications that the malware was attempting to use software that have access to your PC’s microphone.