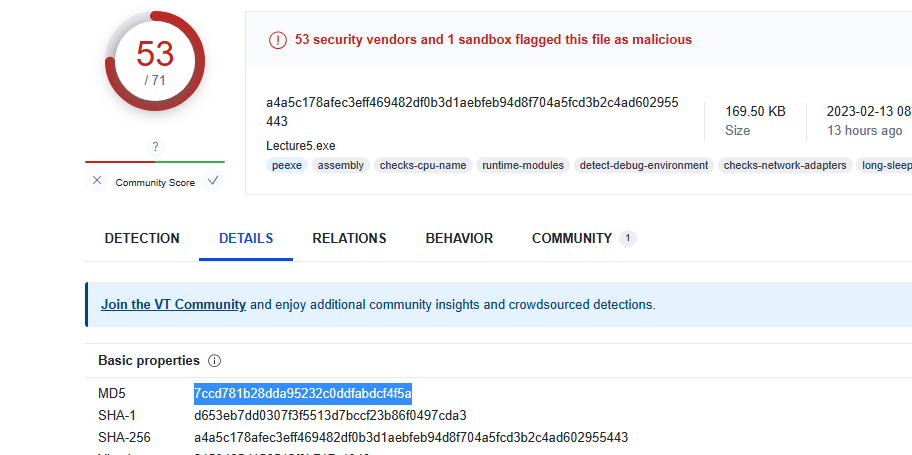
**Virus Total**

Signatures: Matches 53 of 70 existing antivirus signatures.

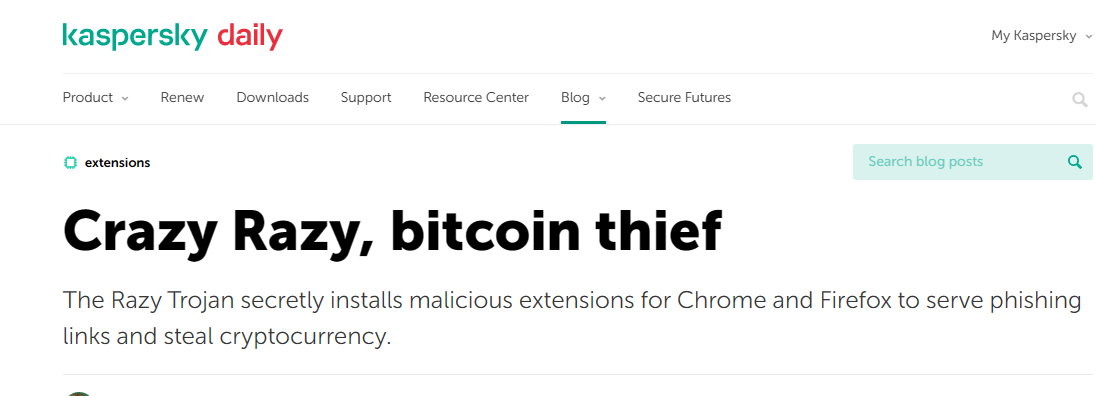
MD5 Hash: 7ccd781b28dda95232c0ddfabdcf4f5a



This malware appears to be a Remote Access Trojan that allows a user a backdoor into the machine. Some vendor analyses labels the malware signature as “Razy”.

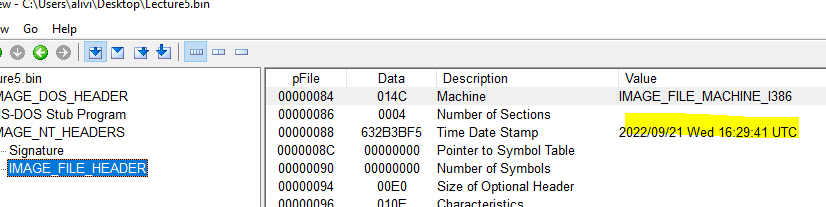


According to Kaspersky, Razy is a bitcoin/cryptocurrency thief which self-installs its own plugins into certain browsers. This plugin disables updates and malicious extension detection and then installs additional malicious extensions. The malware uses a Man-in-the-Browser (MITB) tactic to redirect or advertise to users false cryptocurrency websites in order to fleece them of their crypto (https://usa.kaspersky.com/blog/razy-trojan-cryptocurrency-stealer/17048/).



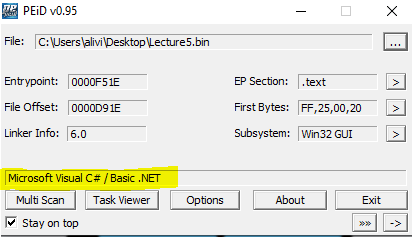
**STATIC ANALYSIS**

**Compile date**: 21 Sep 2022 at 16:29:41 UTC

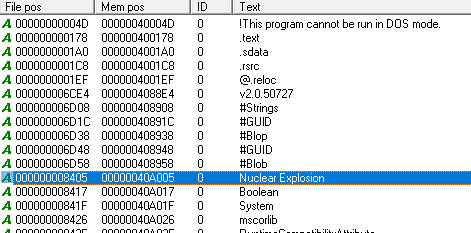


**Packing/Obfuscation:**

PEiD reported no packing by the malware and showed that the programming language it was written in is C#.

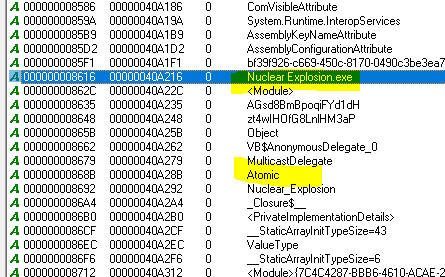


The malware is not obfuscated as there are suspicious strings that clearly do not belong in a legitimate program and indicate malicious intent. Particularly, when ran through BinText, there are numerous references to “Nuclear Explosion”.

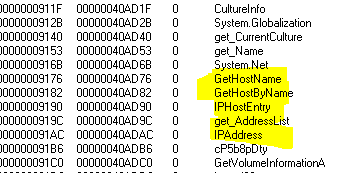


**Imports**

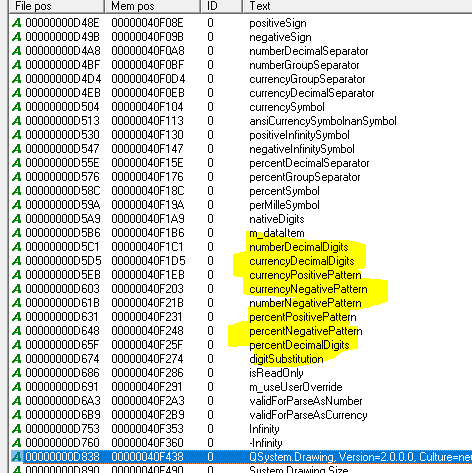
A BinText strings analysis indicated a suspicious file import of “Nuclear Explosion.exe” with other string references to Nuclear-based verbiage such as “atomic”. Unless the user intended to download an application that displays a “Nuclear Explosion,” then this executable import is very concerning.



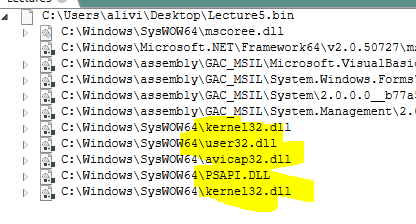
Further analysis of the malware strings indicates that it has networking capabilities. There are strings that indicate the malware gets host names and IP addresses.

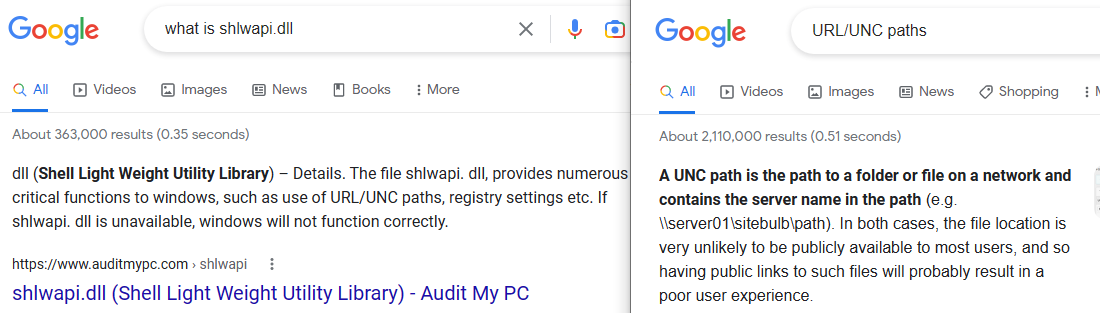


The strings analysis continues to show indications of currency and percentages, indicating a monetary aspect to this malware. There is also an import that states “QSystem.Drawing” that supersedes the currency strings, which possibly indicates a flooding of the user interface with symbols related to money.



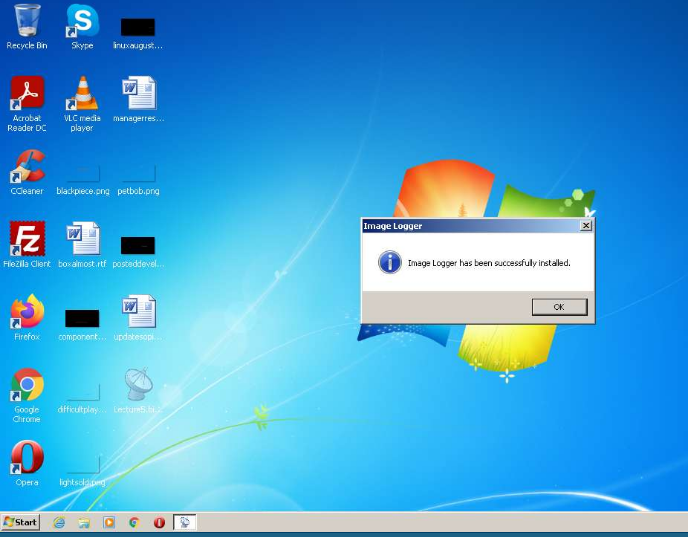
The dll imports that this malware calls for are kernel32, user32, avicap32, and PSAPI. Kernel32 will give the malware access to core OS functions and the ability to manipulate memory, files, and other hardware. User32 will give it the ability to control the user interface. Avicap32 will give it the ability to capture video files from camera hardware on the computer and store them in a .avi file extension. PSAPI provides support for process status and is needed for the machine to run properly. 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).





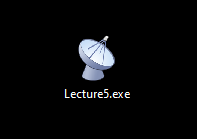
**Other Files/Host-Based Indicators**

The file that primarily stands out is “Nuclear Explosion.exe” which would indicate infection on a system. Additionally, there is an application window titled, “Image Logger” that pops up on the user’s screen when the malware was ran in app.any.run. This potentially confirms the malware’s ability to capture video from the host machine by using the avicap32.dll import. App.any.run did not show indications of file modification or importation to the system. This tells me that the “Nuclear Explosion.exe” found in the strings analysis is automatically run by the malware as the malicious delivery package.



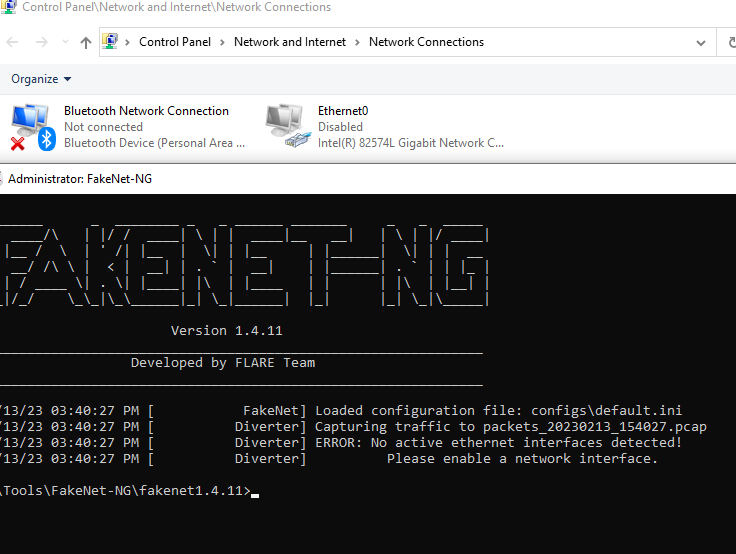
**Hypothesis of File:**

I would certainly consider this file malware, especially with the “Nuclear Explosion.exe” and other atomic-based references found throughout the strings. Nuclear explosions invoke fear and images of devastation, which therefore indicates malicious intent. I would be curious to know what this file was labeled as when the user downloaded it in an attempt to mask its intentions to the layman. If it was marketed as an app that had anything to do with nuclear explosions, then potentially not. The main question would then be as to why such an application would need access to avicap32.dll, which to me is the clear giveaway in conjunction with the networking-based strings that were extracted. We can therefore deduce that this file most likely allows someone access to the machine’s camera to check if the user is present. If the user is not present, then the malicious actor can use that connection to exfiltrate information or spy on the user. There is further indication that this file has networking capabilities implied by the radar icon that emerges when it is changed to the .exe file extension.

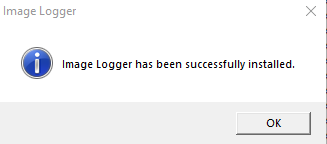


**RUNNING THE MALWARE**

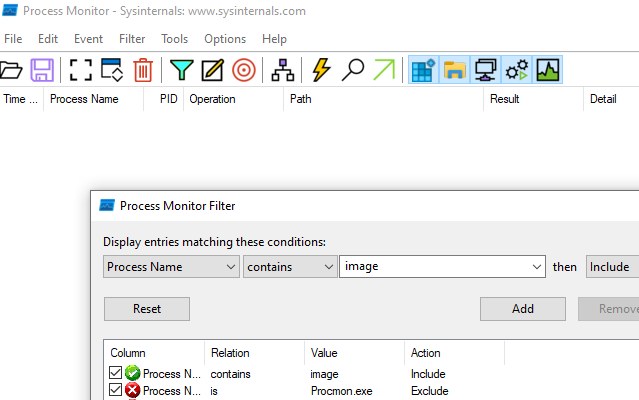
When running the malware, procmon and Process Explorer were used. Fakenet-NG could not be used due to the instructions stating to turn off networking. To enable the safest-possible environment, I disabled the ethernet connection on my VM which disallowed use of Fakenet-NG.



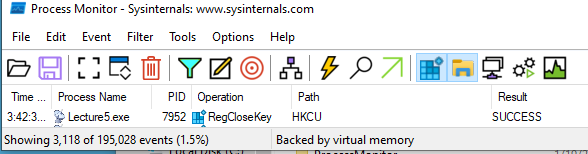
Immediately after running the file, the same window that appeared on app.any.run appeared, indicating that “image logger” was successfully installed.



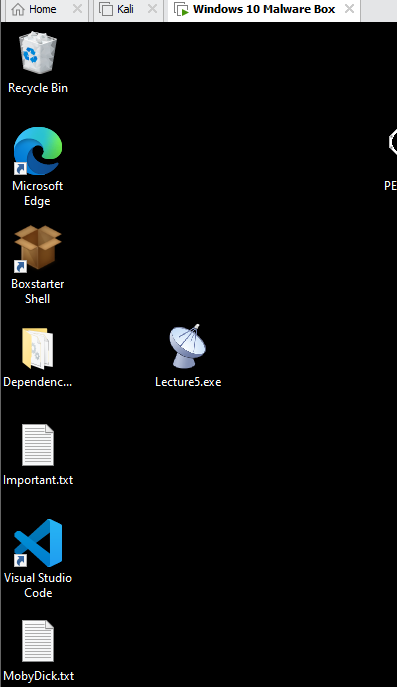
A process name that contains the word “image” was not found when applying filters to procmon in an attempt to search for this “Image Logger”.

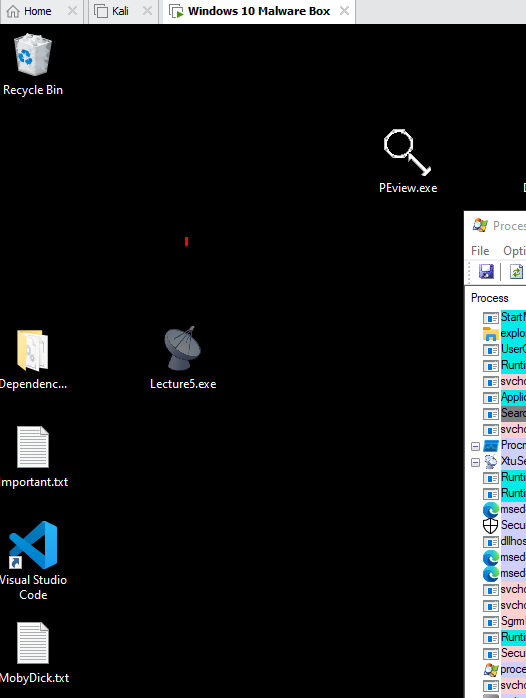


In total, there were 3,118 file system and registry events associated with Lecture5.exe.

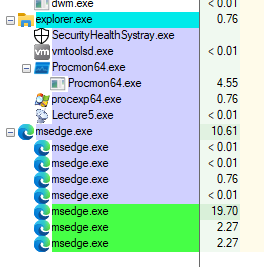


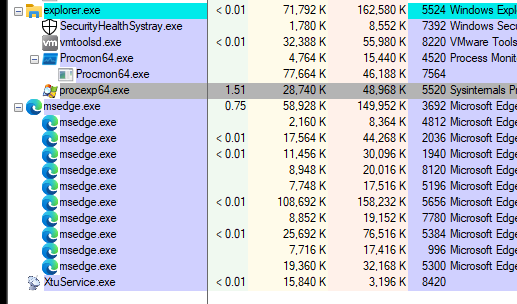
After closing the window indicating a successful install of “Image Logger,” there were two icons that were missing from the desktop: Microsoft Edge and Boxstarter Shell. This was not immediately noticed as it took some time for the malware to delete them. Another item of note is that the Lecture5.exe icon became greyed-out, or somewhat transparent-looking. Below is a before picture followed by an after.



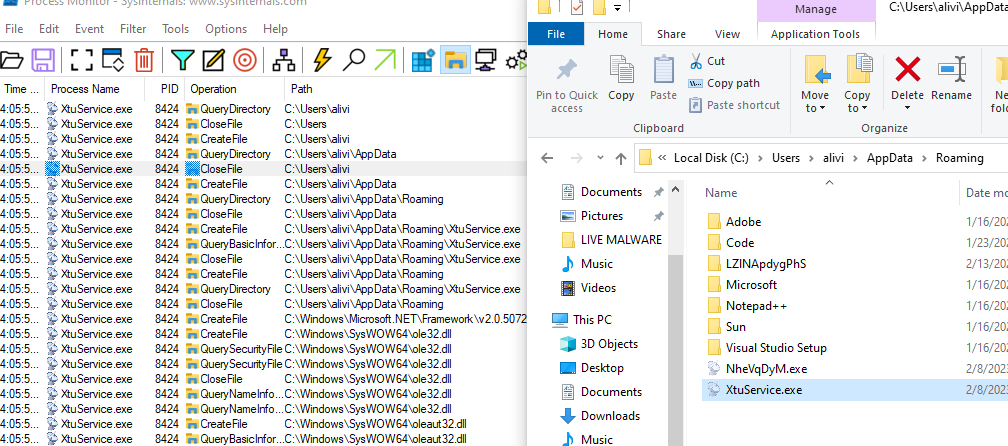


However, Microsoft Edge was still in the toolbar and when I clicked on it then closed it, another popup appeared indicating a successful install of “Image Logger.” Additionally, the Lecture5.exe process in Process Explorer closed and was replaced with a process containing an identical icon and was titled “XtuService.exe.” See below for before and after pictures, respectively.

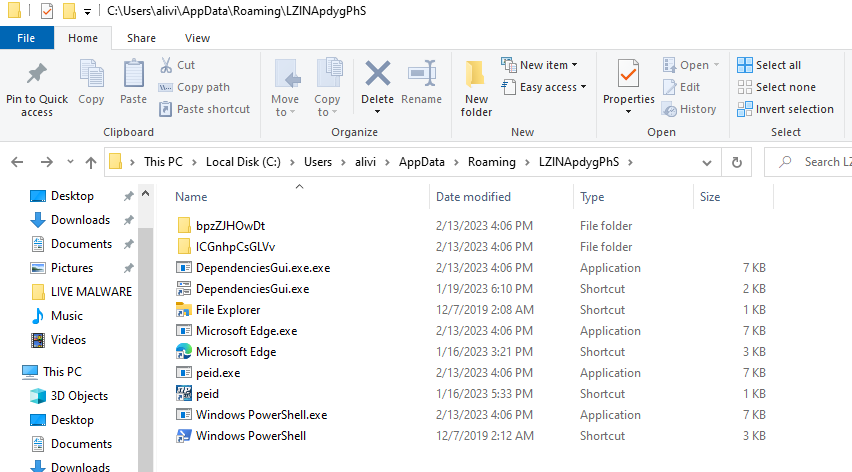




XtuService.exe was not present on a clean Windows 10 machine but was located in the C:\users\<name>\AppData\Roaming folder, found by using the capture by procmon.



In the image above, there are a couple of suspicious files. One file is “LZlNApdygPhS” and the other appears to be a similar application to XtuService.exe titled, “NheVqDyM.exe.” The file LZlNApdygPhS contained copies of my desktop and toolbar. However, it only contained Applications, file folders, and shortcuts. It did not contain documents.

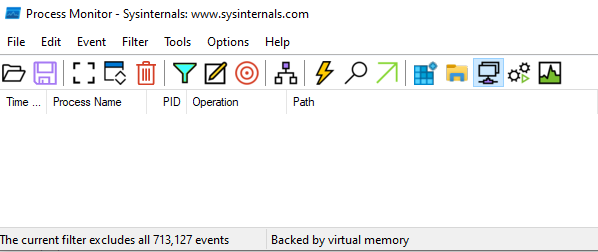


Of note, the executable file NheVqDyM.exe was searched in both procmon and Process Explorer, but no instances of it were found. However, when a browser window was opened, the same Image Logger successful install window appeared and NheVqDyM.exe appeared in Process Explorer. But NheVqDyM.exe disappeared from process explorer when “ok” was pressed.

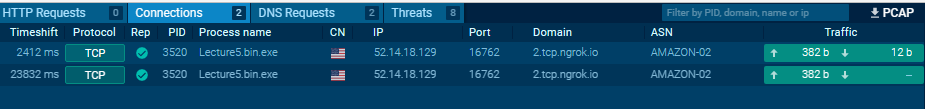


**Network-Based Indicators**

After following the instructions to disable networking and running a lengthy procmon capture, out of over 700,000 events, there were no network-based indicators that this malware provided. Since the instructions indicate to disable networking, this directly implicates the inability to run fake net. I assume that this malware does something with networking based off of the static analysis, but constraints placed on this analysis are not conducive to network analysis.



However, when the file was run on app.any.run, there were two TCP connections makde to the domain of 2.tcp.ngrok.io with an IP address of 52.14.18.129 over port 16762 (an unassigned port number). The IP address belongs to Amazon Web Services.



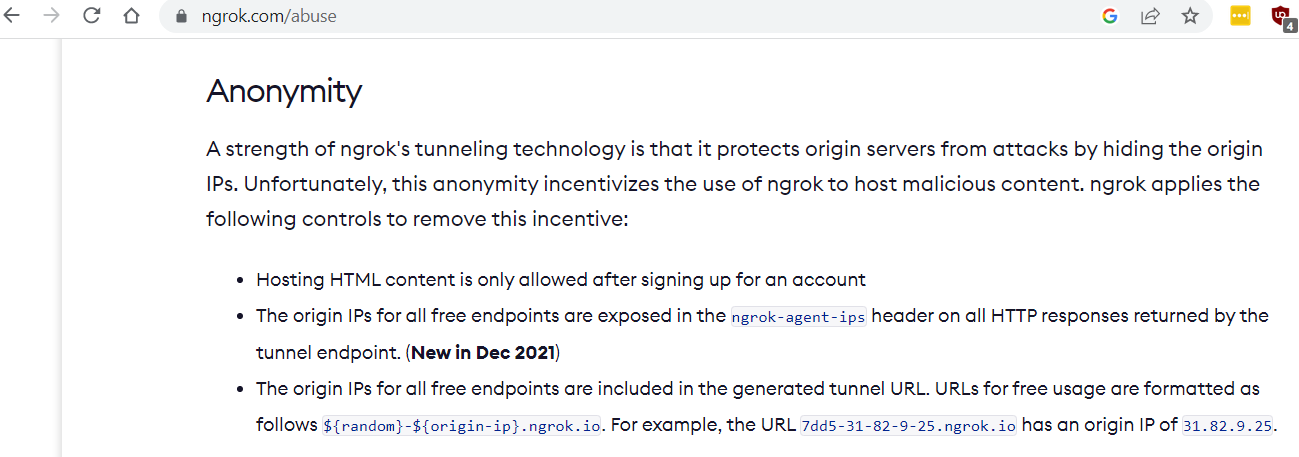
The Ngrok website describes itself as “a simplified API-first ingress-as-a-service that adds connectivity,

security, and observability to your apps with no code changes.” It appears to be a third-rate version of github.

**Purpose of the File:**

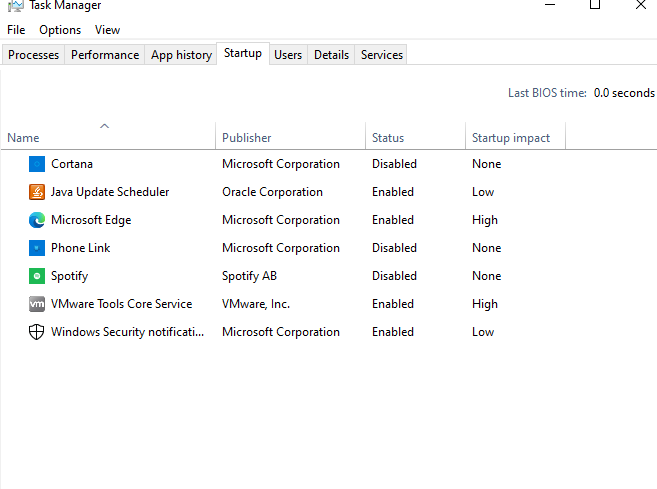
I assume that the file makes copies of the user’s desktop applications, shortcuts, and folders and places them in the AppData\Roaming folder for future access by the malware creator without authorization. This folder is intended for data synchronization between computers (like your Google bookmarks). Then, whenever a browser is opened on the host machine, NheVqDyM.exe (aka the Image Logger Window) makes a copy of the desktop and places it in the folder. Additionally, NheVqDyM.exe knows when to run due to XtuService.exe constantly running in the background. These files are exported over TCP to an AWS repository by using the ngrok product to steal information from the user. The official ngrok website even espouses anonymity of IP origin which incentivizes its use for malicious purposes and abuse.

The exfiltration of desktop and toolbar configurations that could potentially be used to identify the purpose of the host machine, indicating the malware is a profiling tool where all data is stored and parsed in a central repository on AWS.



**How Often I Could Run this VM**

I feel as if I could run this VM again after running this malware, albeit I would not feel comfortable with doing so without a snapshot as it is difficult to thoroughly ascertain the malware’s intent. The XtuService.exe process running in the background was easily closed by using task manager and did not appear as a startup process. After restarting the machine, there were no further indications of the malware running.



**Recommendations to Management**

I would recommend to management a full network sweep of machines for instances of XtuService.exe and NheVqDyM.exe and adding them to a block list. Furthermore, I would add ngrok to blacklisted applications and domains due to its use for malicious purposes. Since the malware did not appear to copy any sensitive files (only applications and folders), I would assume that there was not any exfiltration of sensitive data from the infected host machine. Most likely, the exfiltration of desktop and toolbar configurations that could potentially be used to identify the purpose of the host machine, indicating the malware is a profiling tool where all data is stored and parsed in a central repository on AWS.

We could most likely clean the system by simply deleting the applications as well as ending them through the taskbar. I would recommend setting up a sandbox environment with network connectivity for long-term evaluation. I would incorporate a DNS sinkhole to monitor potential data exfiltration. For the long-term evaluation, I would let the malware persist to see exactly what conditions trigger data exfiltration or webcam access by an unauthorized user.

* Go to two other students’ 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?