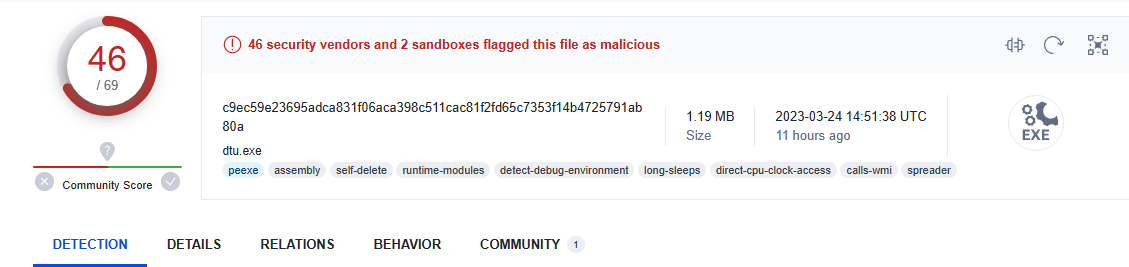
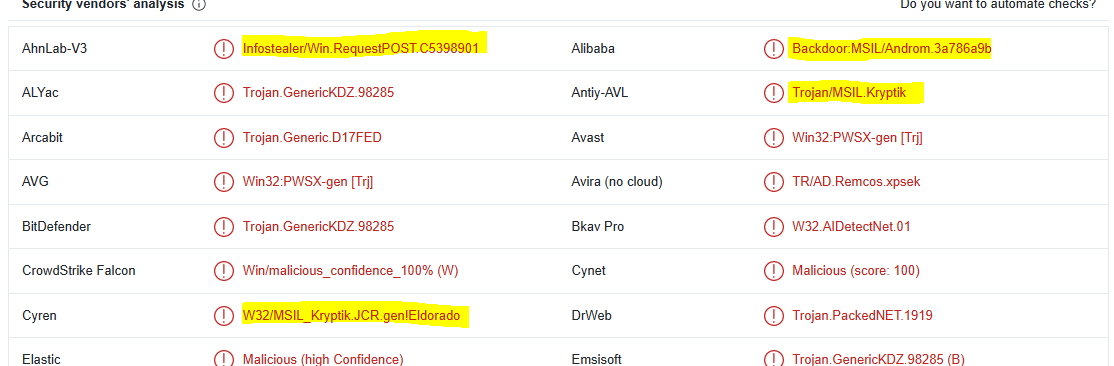
**Upload the file to http://www.VirusTotal.com/. Does the file match any existing antivirus signatures?**

Yes. It matches 46 of 69 security vendor signatures.



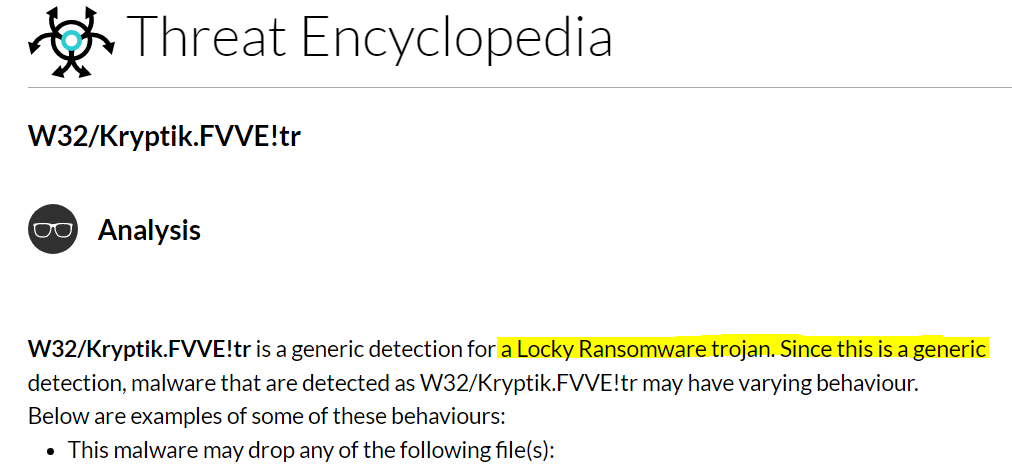
**What is this file known for?**

There are mentions that the malware is an information stealer and backdoor with a unique name of “Kryptik”



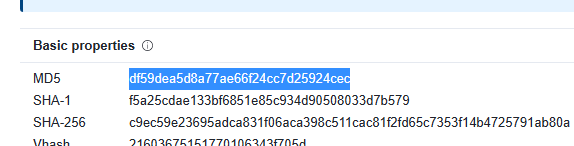
On the Threat Encyclopedia, Kryptik is the generic “Locky Ransomware Trojan”.

<https://www.fortiguard.com/encyclopedia/virus/7472257/w32-kryptik-fvve-tr>



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

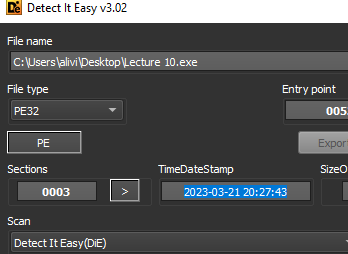
MD5: df59dea5d8a77ae66f24cc7d25924cec



**Using the tools, we discussed so far, answer the below.**

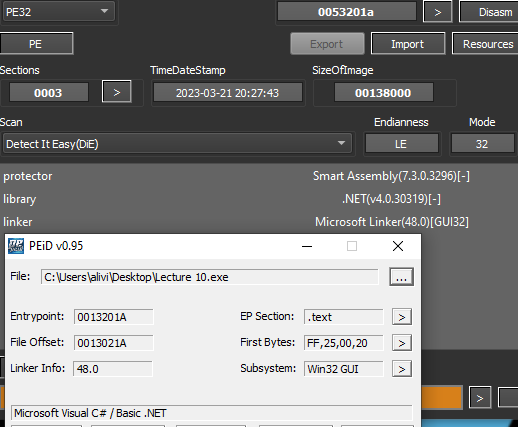
**When was this file compiled?**

2023-03-21 20:27:43



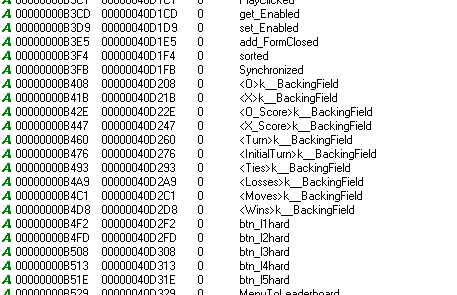
**Are there any indications that this file is packed or obfuscated? If so, what are the indicators?**

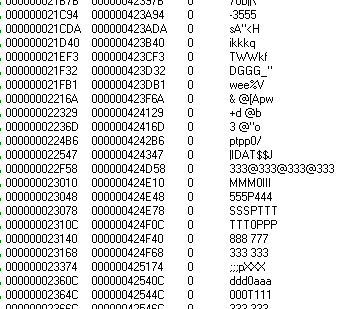
Packing: Neither PEiD nor DIE showed that the file was packed.



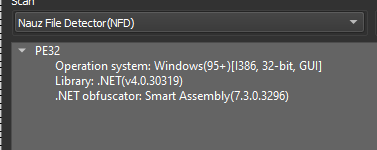
Obfuscation:

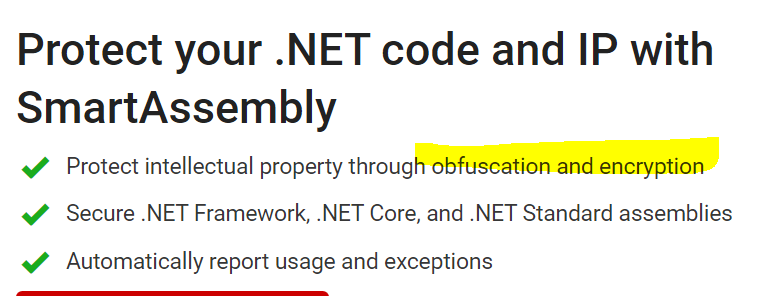
There are many lines of code that appear to be random hex values, but much of the code strings when observed in BinText show regular functions and other text strings, indicating that this file is NOT obfuscated.





However, DIE stated that there is some obfuscation by way of the .NET obfuscator: Smart Assembly. This is a framework to protect intellectual property (according to the manufacturer).





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

Dependencies shows the following .dll imports:

mscoree.dll: Essential for the execution of "managed code" applications written for use with the .NET Framework. Serves as the entry point.

mscorlib.dll: Microsoft Common Object Runtime Library. Contains the fundamental classes and types that provide the foundation for all .NET applications

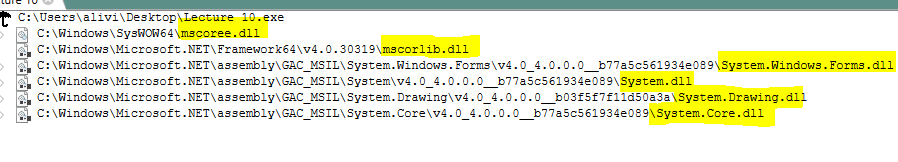
System.Windows.Forms.dll: .NET framework. Provides classes for building Windows-based UIs.

System.dll: Contains a large number of classes and types that provide essential functionality for building .NET applications

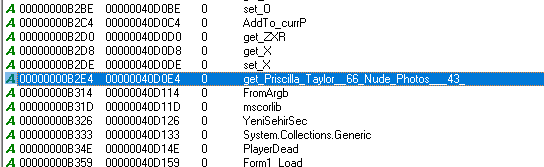
System.Drawing.dll: .NET framework. Provides classes for working with images, colors, fonts, and other graphics-related tasks

System.Core.dll: .NET framework. Provides a set of extension methods and types that enable LINQ queries to be written against various data sources

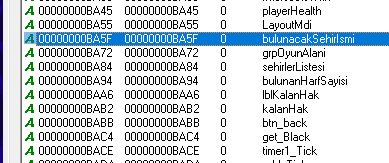
Based on these imports, the program was written in Microsoft’s .NET framework.



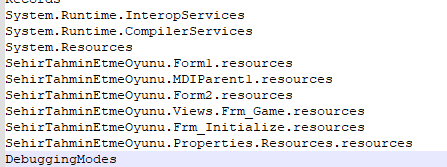
Near the top of the BinText strings, there was this interesting tidbit that is possibly a function to get nude photos of the March 1996 Playmate of the Month? There is also the string “YeniSehirSec”, but it is unknown as to what this means.



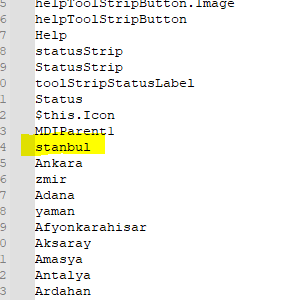
Some more phrases were found. Plugging a few of these into Chat GPT, they appear to be Turkish phrases related to cities. The rough translation of these phrases appears to be contextually game-related when interpreted together.



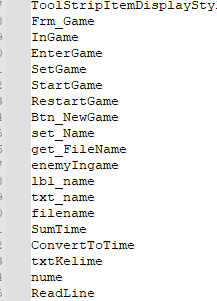
This is further supported by finding the phrase, “SehirTahminEtmeOyunu” which translates to “City Guessing Game”.



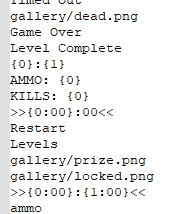
This is further supported by finding a list of what appears to be scrambled city names. The scrambled name “Istanbul” stood out. This appears to have the user guess what the city is based off of a scrambled name.



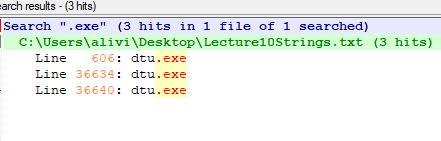
Some strings show more gaming functionality to the file.



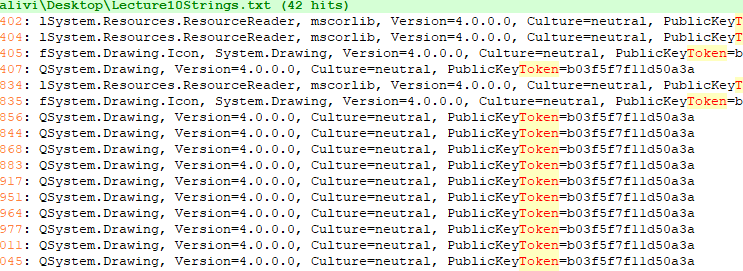
However, the inclusion of “Ammo” and “Kills” in a city guessing game is very suspicious.



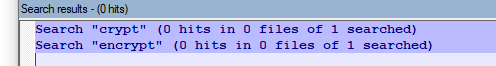
The only executable file mentioned was dtu.exe, the “Dell Data Transfer Utility”.



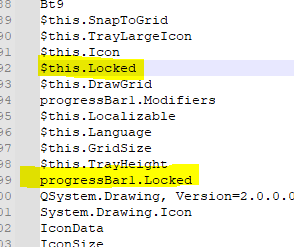
A lot of mentions of PublicKeyToken. This specific token is associated with .NET assemblies and is well-known and trusted.



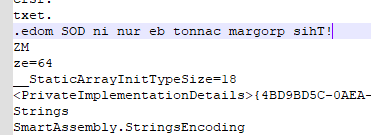
There were no mentions of encryption-based language in the strings.



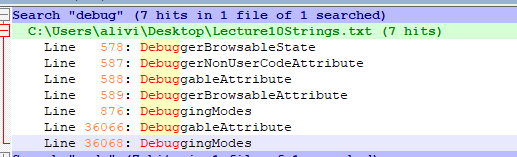
Some code was present to lock some items as part of a Windows Forms application.



A backwards string was found that, when reversed, spells “!This program cannot be run in DOS mode.” This potentially indicates that much of the strings within this file need to be XOR-ed with a value to get the true meaning or are all stored in little-endian (or big endian…I can never keep track which is which).

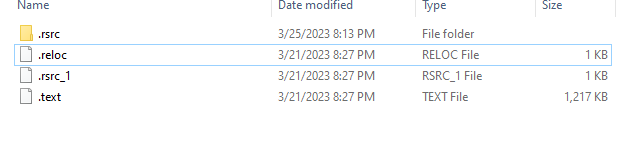


Some “debugging” functions were present, but nothing to indicate that the file attempts to detect a debugger, such as “IsDebuggerPresent()” or, in the .NET framework, “System.Diagnostics.Debugger.IsAttached”.

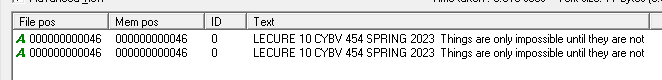


**Are there other files or host-based indicators you could look for on infected systems?**

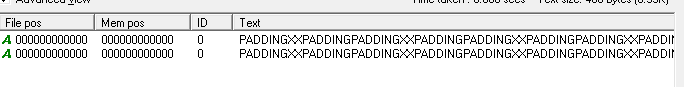
When unpacked with UniExtract, there were some different files that were detected. There was a large “.text” file, a .rsrc\_1 file, a .reloc file, and a folder that contained some uninteresting files. The only file of note was a .ico file that is the image of the umbrella icon for the .exe.



A RELOC file contains information about the relocation of code or data in a compiled executable file or library. It contains the relocation table. But this file only included a message from Prof. Galde.



The RSRC\_1 file is a common name for a resource file on windows that usually contains data resources such as images, sounds, or other data. This file only contained padding strings.



Other than that, there were no other host-based indicators found within the static analysis.

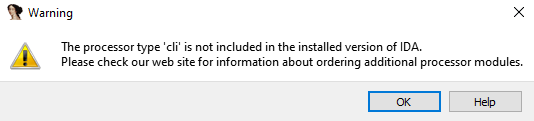
**BEFORE you run this malware, would you consider this file malware based on your findings? "Exclude your virus total findings."**

If it were not for the mentions of getting Priscilla Taylor’s nude photos, then I would not consider this file malware. But those strings are very suspicious and point to some hidden agenda within this file.

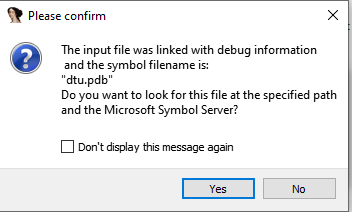
**Using the tools, we discussed in lecture 6, answer the questions below and provide screenshots from IDA Pro.**

**Provide a screenshot of any error messages this malware produces.**

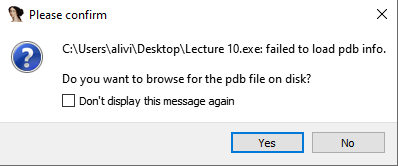
I got this when loading the file into IDA. The .NET processor module is only available in the Pro version.



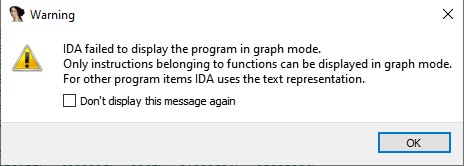
I then selected the “Portable Executable” option when loading into IDA and got this message, clicking “yes”.



Then this:

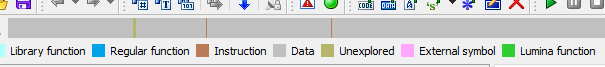


Trying to look at graph view of \_CorExeMain:

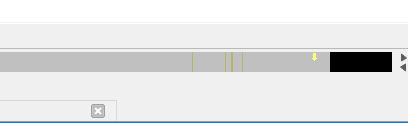


**Provide a screenshot of the navigator bar (the color-coded bar showing library functions, regular functions, code, data, etc.)**

First portion of the navigator bar:

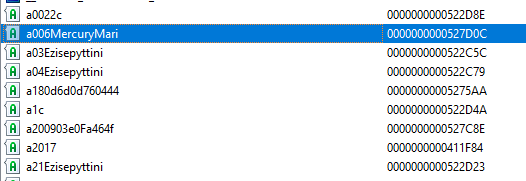


This portion was followed by a LOT of grey. Then at the end, there was this portion:



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

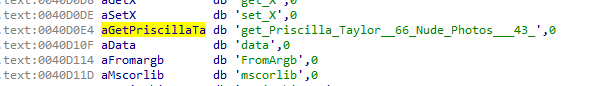
There are a lot of options in the Names window (958 to be exact). What is interesting is the backwards type as noticed in the static analysis.



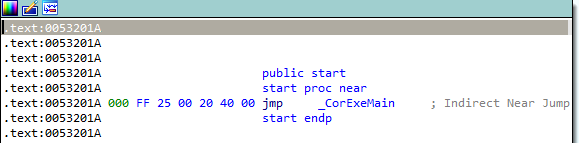
Examining some of the “game” strings, they are simply variables and not functions that would typically be associated with a game.



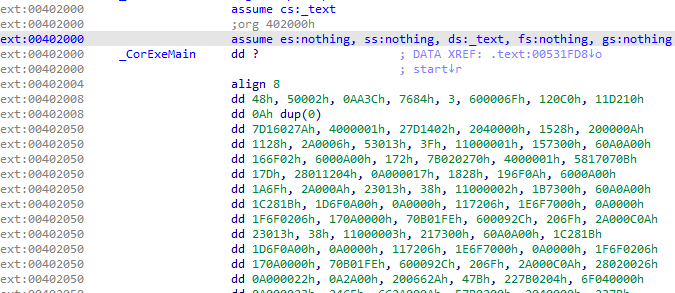
And we find our Playmate friend again.



Wow. What a start function!

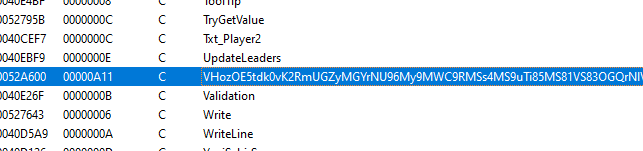


The jump instruction to \_CorExeMain is another name that was in the Names window. This is where I believe the true intentions of this program resides as it is all in hex. Interestingly, it cannot be viewed in graph view.

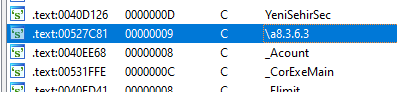


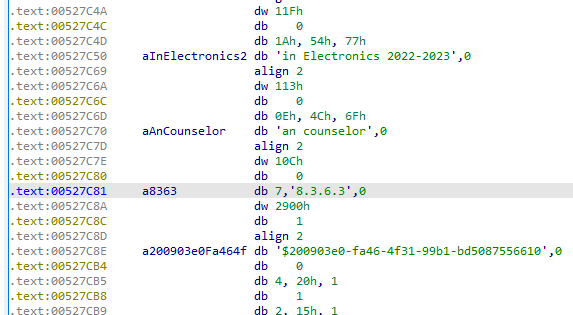
**Open the Strings window and provide an analysis of which ones are interesting and why.**

Once again, many of the strings found are simply variables within the code. There were a couple of strings that were interesting, such as this long string of random characters. Perhaps this is an encryption key?

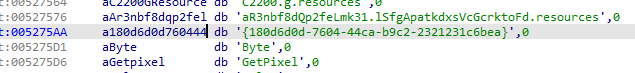


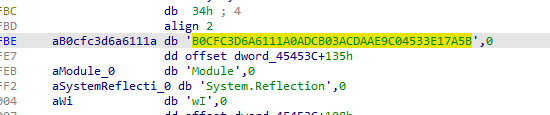
Another string that caught my eye was “\a8.3.6.3”. This showed a lot more complex variables than the majority of the strings observed.

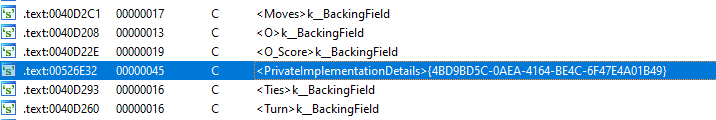




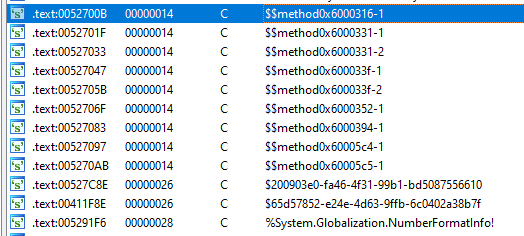
Another string that did not make sense can be seen below at its memory address. There is another string below it that looks like a software key, or perhaps a decryption/encryption key. There were a few of these kinds of strings found and the data was saved for later use.



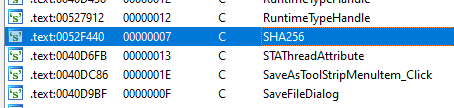




There were also “method” strings followed by memory addresses.

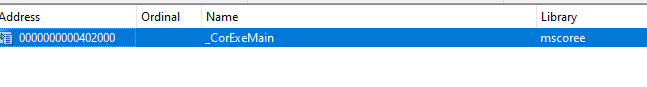


Another string was the mention of SHA256.

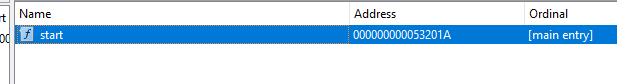


**Open the Imports and Exports windows and provide an analysis of your findings.**

Imports was only \_CorExeMain.



Exports was only the start function.

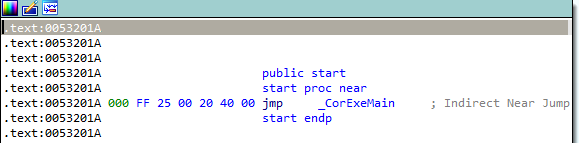


These were already examined from the Names window and were remarkably unremarkable.

**Open the file in IDA pro and identify the following:**

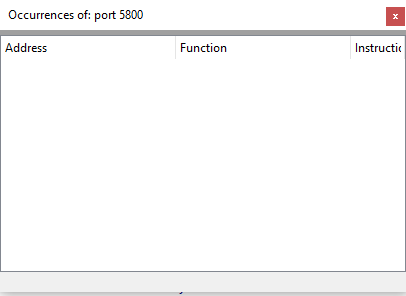
**What does IDA identify as the START of the program?**

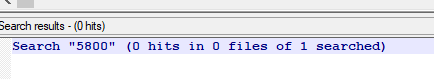
The function “start” was only 1 line and only included a jump instruction.



**What is port 5800 may be referring to?**

A text search of all occurrences of “port 5800” in IDA and in the entire Strings.exe output yielded no results.

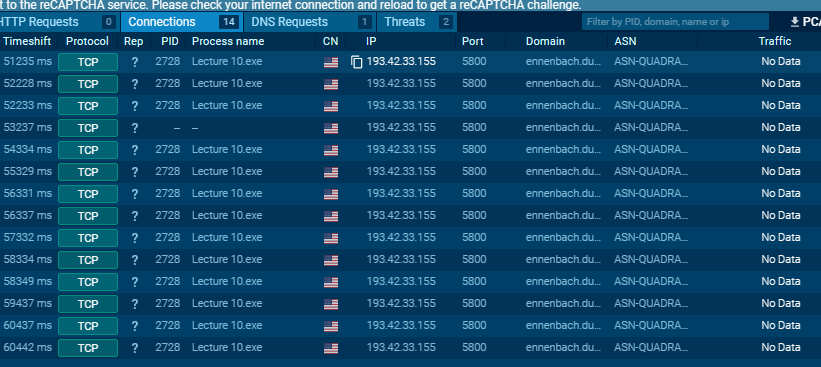




However, port 5800 is used for remote desktop access over TCP. If the program connects over port 5800, then it is likely to be a backdoor for someone to access and manipulate the host machine.

**When the file is running, a connection is attempted over port 5800 but to what address, provide an analysis of why.**

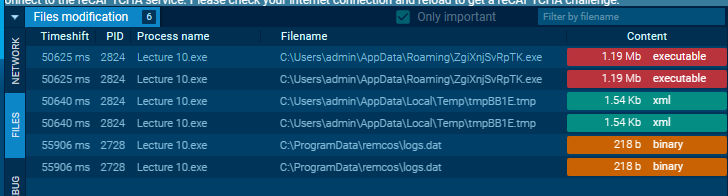
When the file was placed into app.any.run, there were 14 attempted connections over port 5800 to the same IP address of 193.42.33.155. The domain was ennenbach.duckdns.org is a subdomain of duckdns.org. This is a free Dynamic DNS service that allows users to assign a domain name to their dynamic IP address. This IP address can change often. It allows users to access their home network remotely using that domain name. Therefore, it is reasonable to assume that over port 5800 over this domain name, the owner of this file set code to establish a remote connection to the host machine.



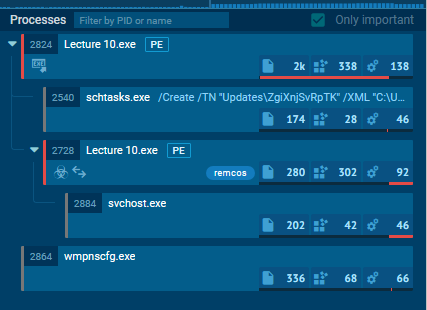
**Run the file and record your observations. This malware may require you to reset your box a few times, this is nasty malware, don't enable networking, and run fakenet.**

App.any.run:

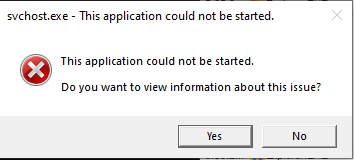
Files dropped:



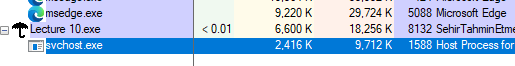
Processes run: Looks like the dropped file into the AppData\Roaming was scheduled to run by task scheduler. The file also spawned svchost.exe as a child process. There is also a windows media player config executable that was run as well.



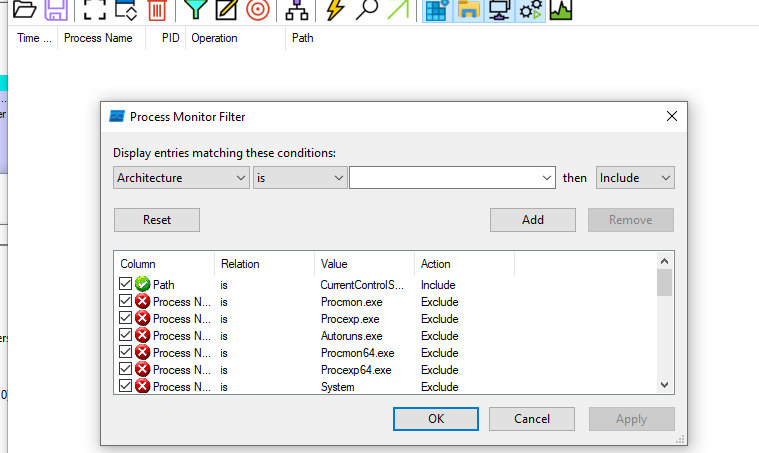
When the program was run on the VM, an error box popped up stating an error with svchost.exe. This kept popping up no matter if yes or no was clicked.



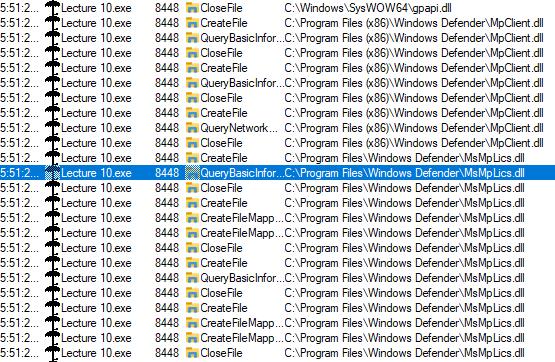
As predicted from app.any.run, Lecture10.exe spawned a svchost child process, most likely related to the error window above.



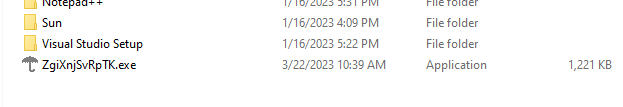
No modifications were found in CurrentControlSet\Services with procmon.



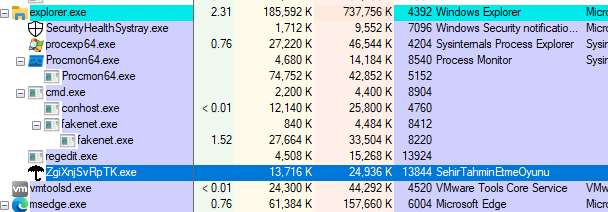
Lecture10.exe queried the Windows Defender dll file.



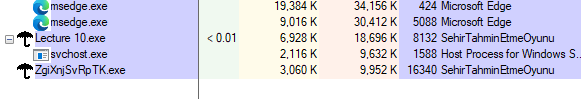
A file was found in the AppData\Roaming folder with a similar icon to the PE file run.



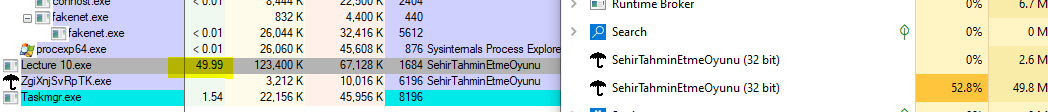
When this file was run, it was a child process of explorer.exe.



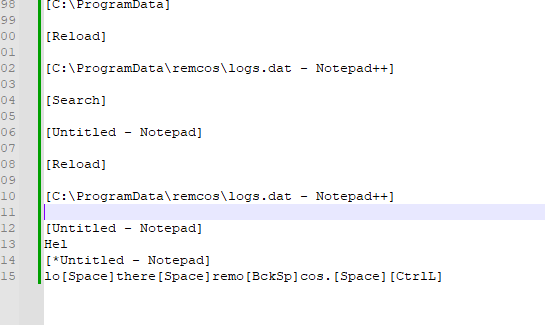
It then became a child process of Lecture10.exe. Notably, it has a string observed in the static analysis, “Sahir TahminEtmeOyunu”.



When task manager was opened, two instances of Sahir TahminEtmeOyunu were running, with one taking up 50% of the CPU. This was cross-referenced with Process Explorer and was Lecture10.exe.

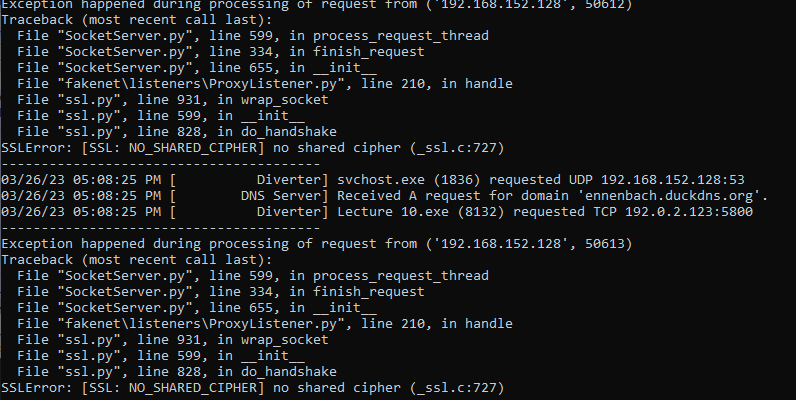


App.any.run detected a file drop in ProgramData\remcos. This file was a .dat file of logs. When reviewing the logs, it was clear that this was a log of user actions taken. It tracked files accessed and also displayed keylogger behavior.



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

Fakenet saw a lot of back-and-forth where Lecture10.exe had been requesting connection requests from 192.0.2.123 over port 5800. The DNS request was the same identified in app.any.run. There were also ssl requests but a message kept appearing stating that there wasn’t a shared cipher.



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

A search of “remcos” online showed that it is a remote access trojan. It also has behaviors of capturing login credentials and other financial data (hence the keylogging aspect). Since port 5800 was used to attempt a connection, it is very clear that this is the intent and purpose of the file.

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

See above.

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

All the time. It didn’t encrypt any of my data or programs. But I still wouldn’t feel comfortable having the potential of someone accessing my machine.

**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?**

We need to stop generating revenue and clean. Any instances of this file or file descriptions of Sahir TahminEtmeOyunu to be blacklisted from the network. The logs would need to be reviewed to see if any sensitive data was stolen. Any unauthorized connections from duckdns or over port 5800 should also be blacklisted.

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

Simply closing the program through process explorer was enough to stop the logging aspect of the malware. There were also no more requests over port 5800 or any other network-based activity captured on Fakenet. Just deleting the program after closing it should be enough and no persistence indicators were found.

**Go to two other students' posts and observe their findings. Post if you agree or disagree with the results.**

<https://d2l.arizona.edu/d2l/le/1243099/discussions/threads/9740564/View>

<https://d2l.arizona.edu/d2l/le/1243099/discussions/threads/9741681/View>

**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?**

Yes. I provided enough information and evidence to conclude that this was indeed malware. They keylogging/spyware aspects were especially troubling and would not give management any hesitation to not allow this file in the organization.

**Did they find something using a new technique, and if so, would you use this next time?**

No. We used similar techniques.

**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?**

I think my analysis provided the key piece of keylogger evidence that the posts I reviewed didn’t. But either way, we all reached the same conclusion.

aR3nbf8dQp2feLmk31.lSfgApatkdxsVcGcrktoFd.resources

*a180d6d0d760444* {180d6d0d-7604-44ca-b9c2-2321231c6bea}

<PrivateImplementationDetails>{4BD9BD5C-0AEA-4164-BE4C-6F47E4A01B49}

B0CFC3D6A6111A0ADCB03ACDAAE9C04533E17A5B

{180d6d0d-7604-44ca-b9c2-2321231c6bea}



$200903e0-fa46-4f31-99b1-bd5087556610

$65d57852-e24e-4d63-9ffb-6c0402a38b7f

}C4D9DE65B253-38F9-B2B4-331F-041B3F7B{>eludoM<

*The above backwards:*

<Module>{B7F3B140-3FB1-F133-4B2B-9F83-356B56ED9D4C}