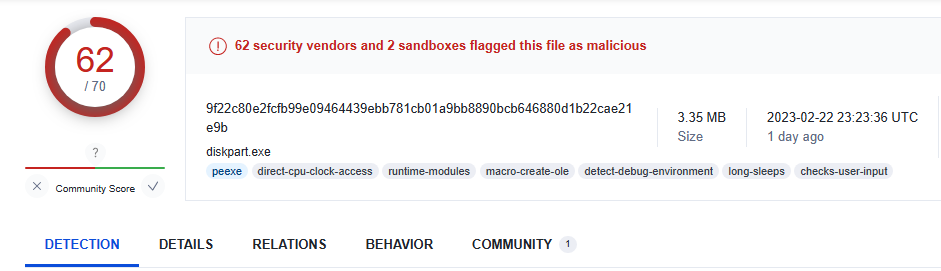
In Lecture 6, we discussed the tool IDA pro which you will now include in your malware analysis environment. This will build on your tools from Lectures 1 and 2.

Answer the following questions using Lecture7.exe:

Answer the following questions:

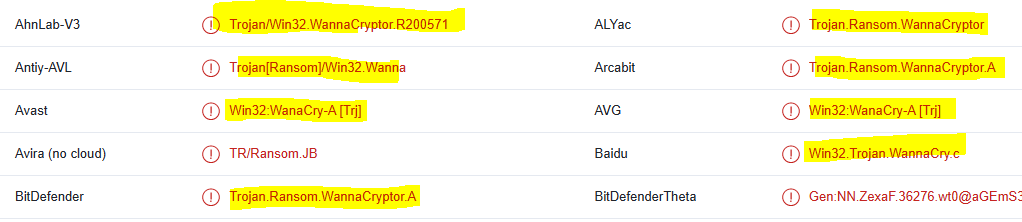
**Upload the file to**[**http://www.VirusTotal.com/**](http://www.virustotal.com/)**. Does the file match any existing antivirus signatures?**

Yes, the file matches 62 of 70 vendor signatures.



**What is this file known for?**

It appears evident that this file is ransomware, specifically the “WannaCry” malware.

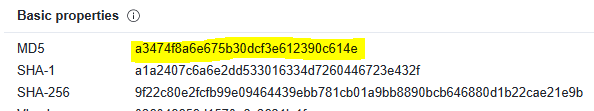


According to Wikipedia, WannaCry was an infamous ransomware attack perpetuated in May of 2017 that target Windows operating systems. It encrypted the data on the host machine and demanded bitcoin payments. It leveraged an exploit called “Eternal Blue” which was developed by the National Security Agency (NSA). Eternal Blue exploits a vulnerability in Microsoft’s implementation of the SMB protocol. This leads me to believe that when the malware runs, I can expect to see traffic over port 445 during network analysis. The WannaCry malware was perpetuated by the hacker group, “The Shadow Brokers”.



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

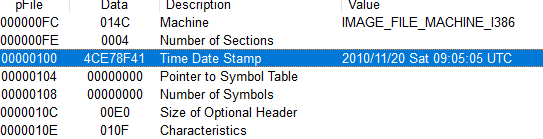
The MD5 hash of the file is a3474f8a6e675b30dcf3e612390c614e.



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

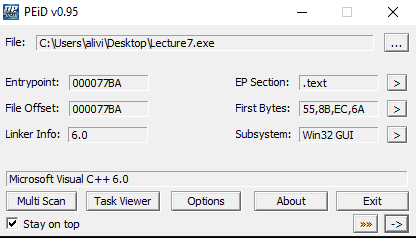
**When was this file compiled?**

The file was compiled on 20 November 2010 at 09:05:05 UTC.

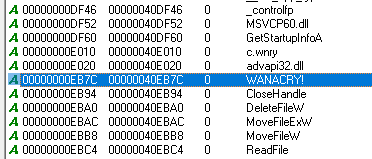


**Are there any indications that this file is packed or obfuscated? If so, what are the indicators?**

A hardcore scan run by PEiD showed no packing indicators and that the file was written in C++.

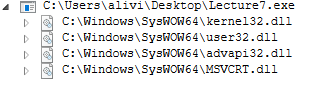


When the file strings were analyzed with BinText, there was a healthy mix of hexadecimal values as well as regular strings, but the main indicator of what the malware is are the strings that state, “WANNACRY!”, and other references to “wnry” which is the shorthand name. Therefore, there is some obfuscation but not enough to prevent deducing the intentions of the file.

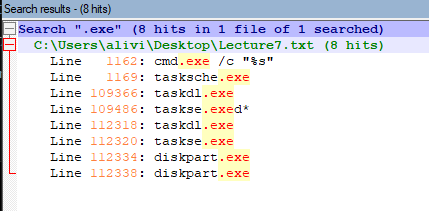


**Do any imports hint at what this malware does? If so, which imports are they?**

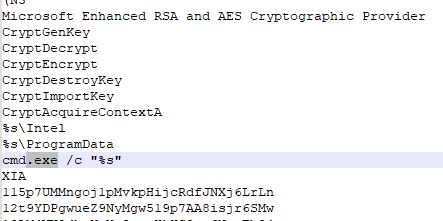
The dll imports of the file are kernel32, user32, advapi32, and MSVCRT. We know that kernel32 will give the file access to hardware functions and the kernel and user32 will give it access to the user interface. advapi32.dll indicates that core Windows components will be altered, such as the Service Manager and Registry. MSVCRT.dll is the standard library for Visual C++, which makes sense since PEiD detected that it was written in C++ and MSVCRT.dll will allow it to access string manipulation, memory allocation, and I/O calls.



We also see that there are a few executable files that the file contains in its strings.



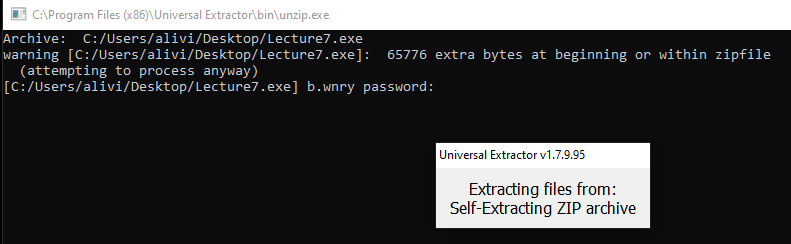
Prior to the command prompt being invoked with the flag /c followed by a %s, indicating a string will be passed into it. The command prompt is invoked after calling cryptographic commands and what appears to be getting the file paths to \Intel and \ProgramData. This suggests that those two directories will be encrypted

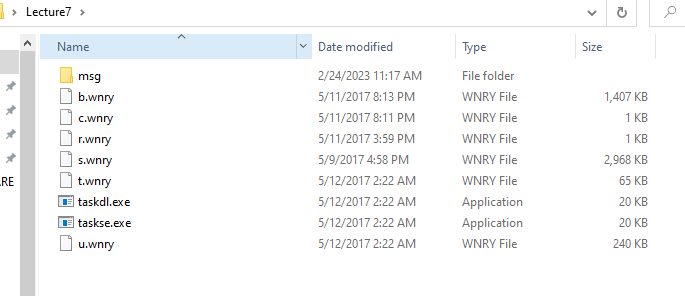


There are also imports that suggest the file will display a message either in multiple languages or in the native language of the host machine.

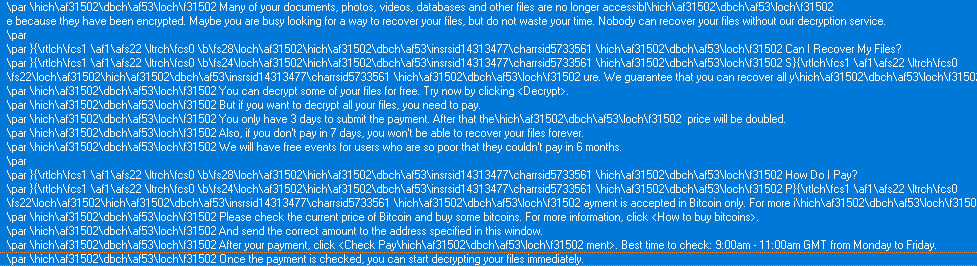


When attempting to extract the file with UniExtract, a command prompt opened up asking for a password. A quick Google search for the password for b.wnry revealed that it was “WNcry@2ol7”. After pasting the password in, the files were successfully extracted.

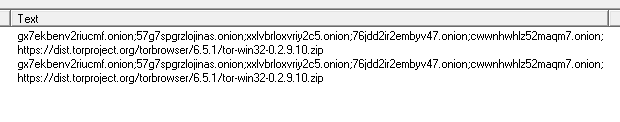




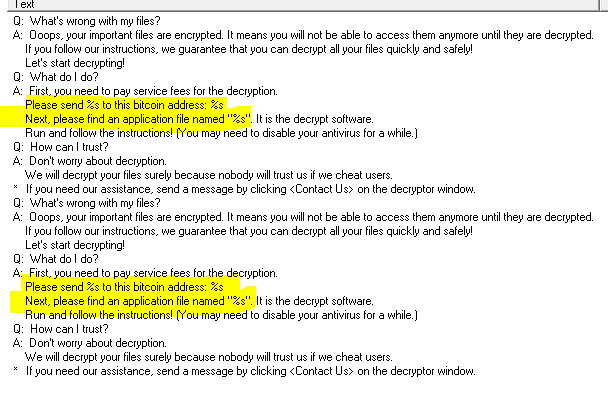
Upon examination of the folder, the “msg” directory contained all the languages. Opening the m\_english.wnry file in BinText, text was found that contains what appears to be a ransomware message. The strings within the message indicate that there will be interactive links that the victim of the ransomware can click.



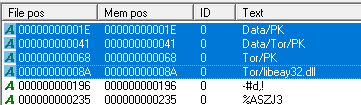
b.wnry contained gibberish, but c.wnry contained what appear to be calls to a TOR browser which is commonly used by malicious actors on the proverbial “dark web” to mask their identity.



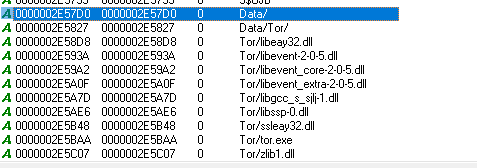
r.wnry contained a more readable version of the ransomware message. It appears that there are variables within the strings to pass values into (the “%s”). This tells me that the malware possible has a table of bitcoin addresses and amount values to randomly pass into the string variables.



In s.wnry, we see at the very top an additional mention of TOR as well as the libeay32.dll, which contains encryption functions to allow for coded communications over networks. This most likely is an additional layer of security to protect the malicious actor’s identity as well as to ensure confidentiality of communications.

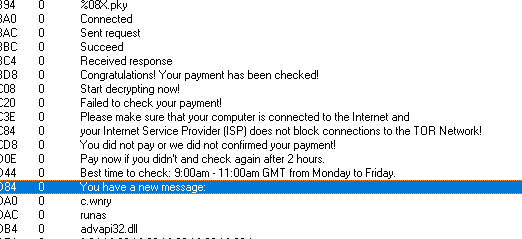


Additionally in s.wnry, there is tor.exe mentioned along with a host of other tor-related imports that potentially install the TOR browser on the machine, or potentially allows the malicious actor access into the infected machine.

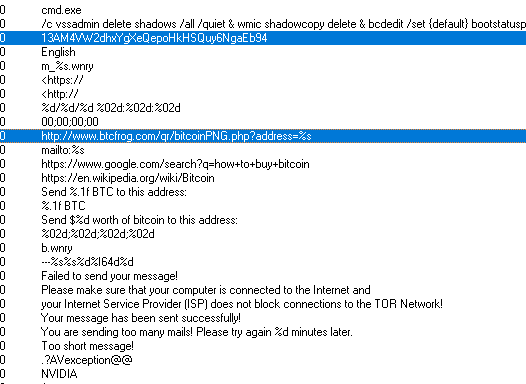


u.wnry shows strings that contain messages to the user, indicating confirmation of payment/non-payment. It also shows a variety of shell strings which is most likely the connection method used between the infected machine and the criminal organization. It also appears the malware will track the mouse, download a file with a specific URL, and set registry keys.





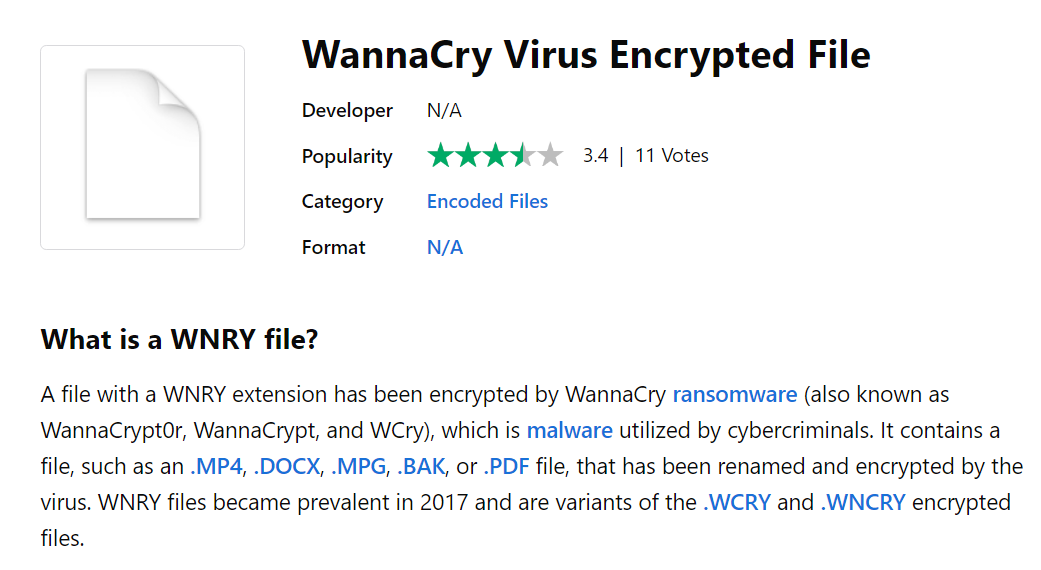
Further down in u.wnry, we see some URLs to websites. Notably, the address to btcfrog.com has a field for additional information to be passed. In the image below, it is possible that the first highlighted section is the bitcoin address to send payment to. Additionally, the phrase telling the user to ensure their ISP does not block connections to the TOR network confirms that payment will be sent over that network.



There are numerous other imports within these files and too numerous to justify time spent analyzing them since it is quite clear that this file is malware.

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

Based on the findings in the strings analyses, it is clear that this file will encrypt files on the host machine and display messages to the user demanding payment. But any file that has the extension .wnry is indicative of this malware and is definitely indicative of this malware being present on a system. I would also think that any file that originally did not have this extension but now has it is indicative of a machine being infected.



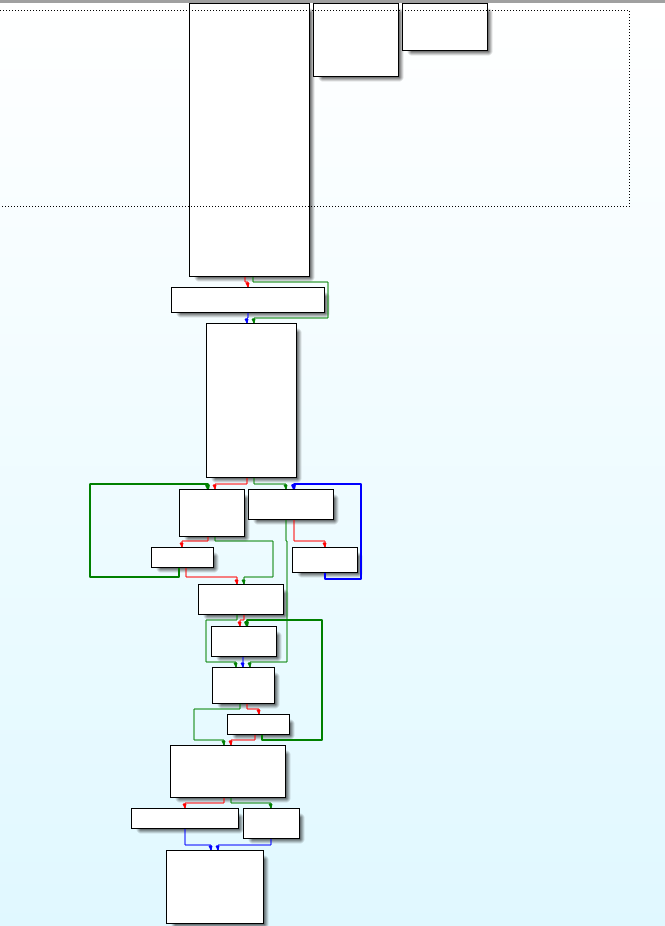
**BEFORE you run this malware, would you consider this file malware based on your findings? "Exclude your virus total findings."**

Absolutely. This is the WannaCry ransomware and all indicators point towards it being so.

\****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.**



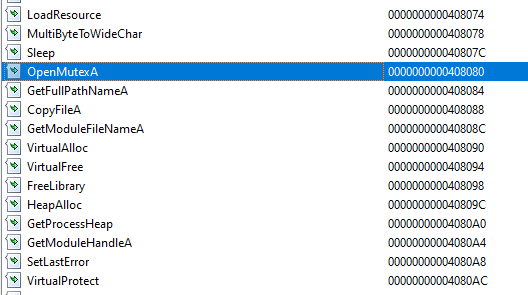
**Provide a screenshot of the navigator bar (the color-coded bar showing library functions, regular functions, code, data, etc.)**

The graph view is large. Sorry it shows up so small.

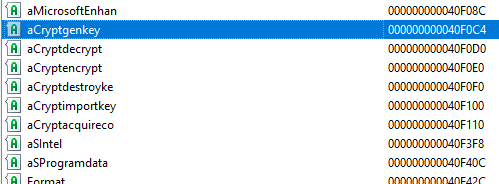


**Open the Names Window and view every function, Library, Code, String, Dara, and Linked Function. Provide an analysis of which of these are “interesting.”**

The OpenMutexA import is interesting and most likely is the import that allows the user to connect to the malicious actor for payment.

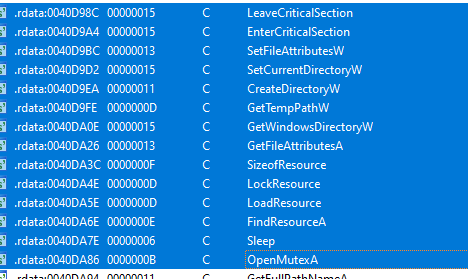


There are also strings that indicate cryptographic key generation for encryption and decryption.

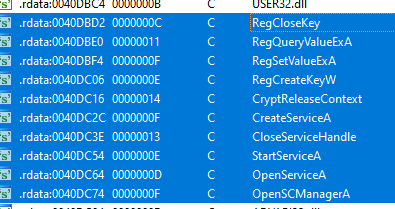


**Open the Strings window and provide an analysis of which ones are interesting and why.**

There are strings that indicate the program will enter critical sections of the machine, create directory information, and then lock resource(s). This is interesting because it indicates that the malware potentially locks critical sections and resources of the infected machine.

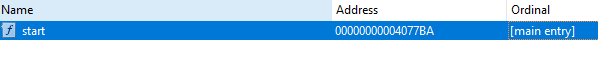


There are also trings that indicate registry keys will be created and their values set. Also, a string indicating opening of the service manager indicate that the malware will use svchost or other windows-native service managing process to run itself.

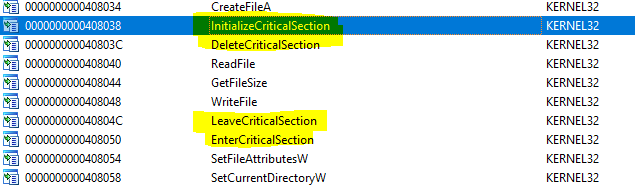


**Open the Imports and Exports windows and provide an analysis of your findings.**

The only export is the start function which will begin running the malware.

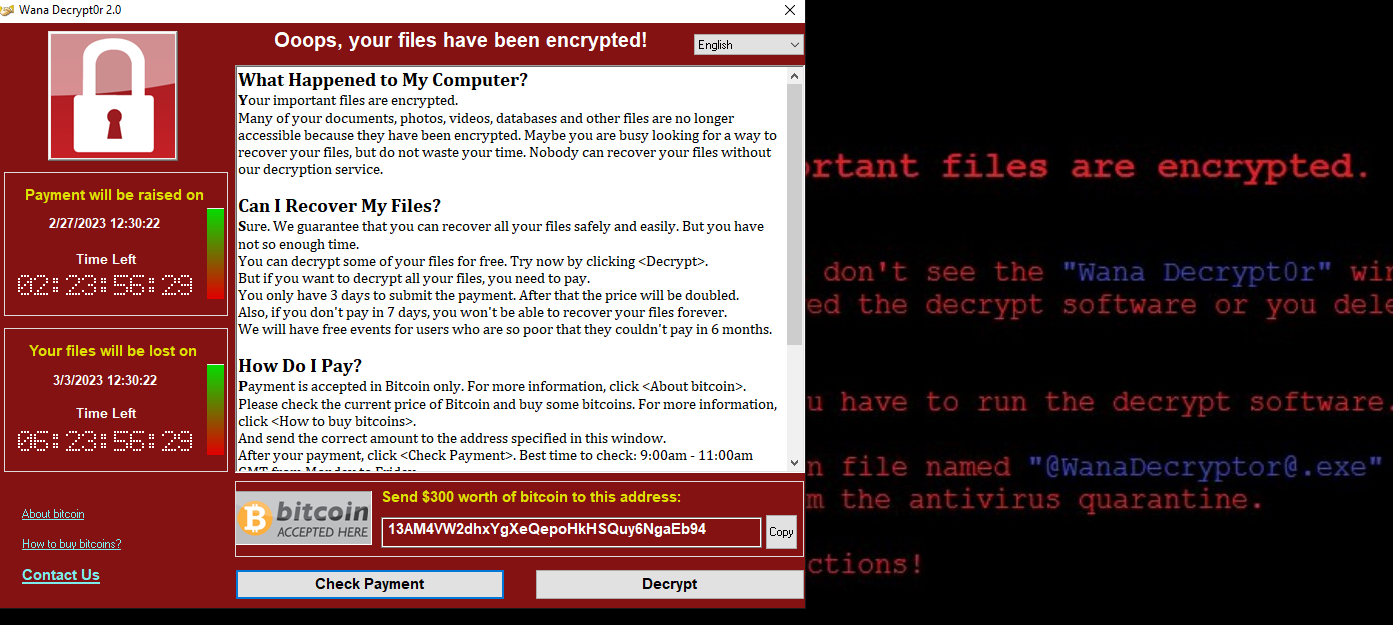


There are 114 total imports, many of which are native C++ functions. There are also numerous imports from the kernel32 and advapi32 libraries, many of which have already been analyzed above. Notably, the kernel32 library is what is responsible for initializing, deleting, entering, and leaving critical sections.

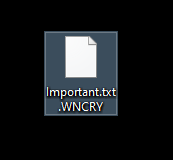


**Run the file and record your observations. This malware may require you to reset your box a few times, this is nasty malware, don’t enable networking, and run fakenet.**

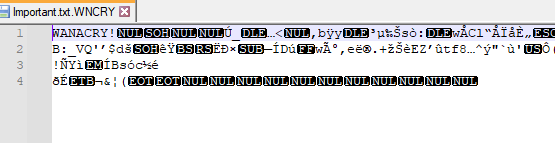
After running the malware, the desktop background changed and an application named “Wana Decrypt0r 2.0” appeared demanding payment in bitcoin. The message was the same as gathered in the strings analysis. There are also two timers on the left side of the application, giving the user a countdown of when payment cost will be raised and when the files will be deleted. There are also numerous links to click on, indicating that the malware does not restrict internet access. The bitcoin address is identical to the string identified in in the “imports” section of this analysis. All of the changes 30 seconds to complete. The application window was annoying because it would place itself in front of all applications and would reopen when closed.



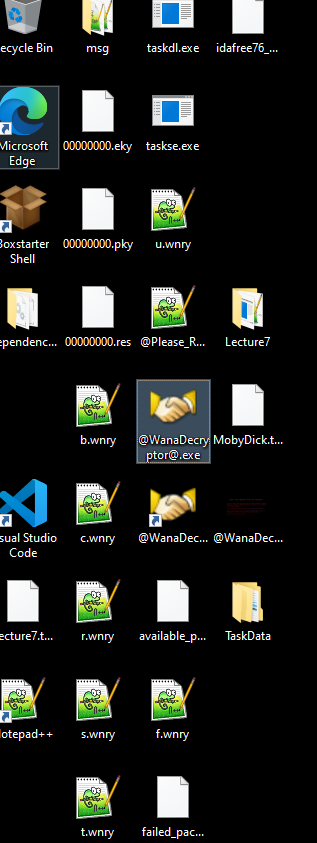
Since I don’t have Microsoft Office on this machine, I had placed some .txt files on the desktop to see how malware interacts with them. The .txt files had additional extensions appended to them, the extension of “.WNCRY”.



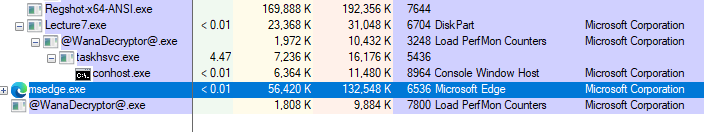
When this file was opened in Notepad++, the file was clearly encrypted.



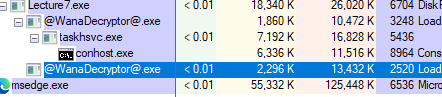
There were numerous files that were placed onto the desktop that were previously extracted using UniExtract. Notably, there are two executables titled “@WanaDecryptor@.exe”. This is also the title of the application window that popped up.



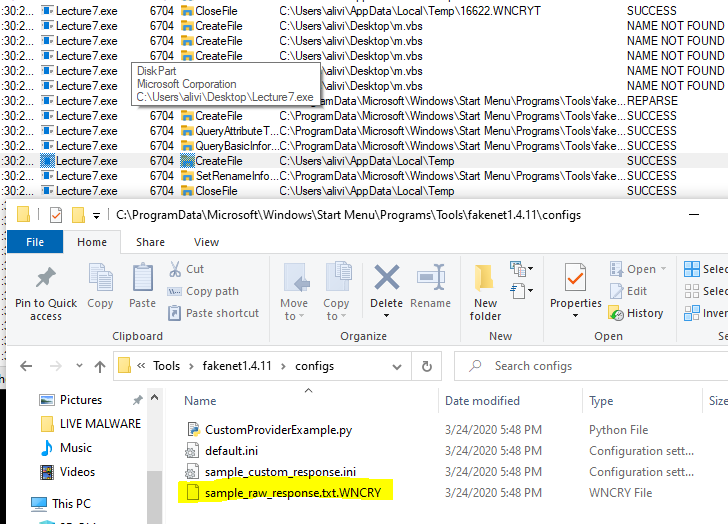
When the malware was run, the root program identified in process explorer was Lecture7.exe with three children: @WanaDecryptor@.exe, taskhsvc.exe, and conhost.exe.



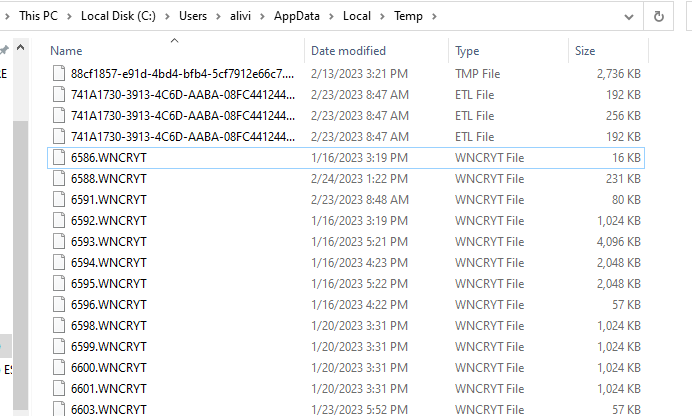
However, when the WanaDecrypt0r 2.0 application window was closed, it reappeared but also created a new process with the previous instance still running.



In procmon, it appears that the Lecture7.exe file is the program that actually encrypts the files. After following a file path that Lecture7.exe called, I found that the .txt file in the Fakenet folder was encrypted. This demonstrates the file walking capability of the ransomware throughout the entire local machine.



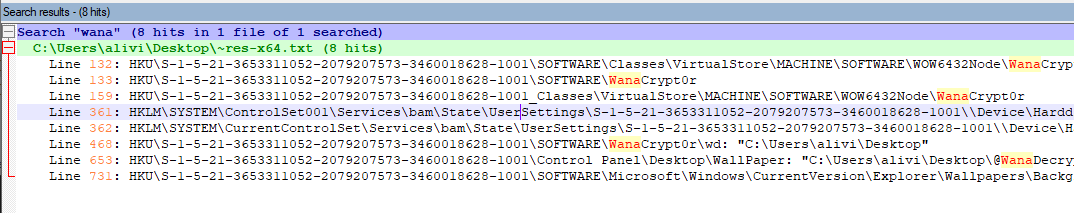
Also noting in the above screenshot is Lecture7.exe accessing the temp folder. In here, there was a new file extension type discovered. Instead of .wnry, it was .wncryt. This would be an additional host-based indicator of infection.



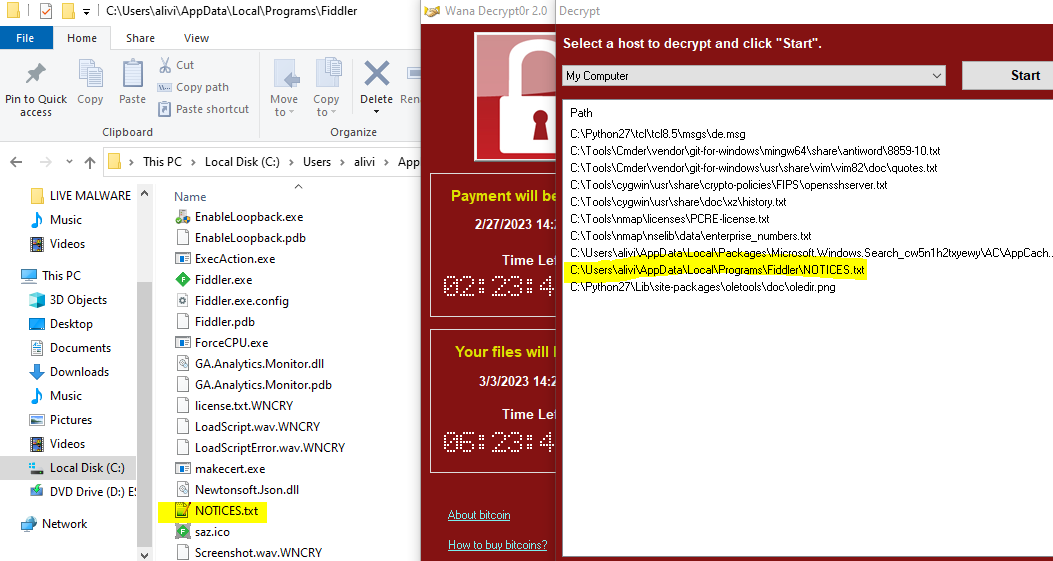
After a brief Google search, these are created as part of the ransomware’s encryption process.



After using regshot, there were 627 total changes detected in the system registry. After scrolling through the changes, I found the suspicious name of “WanaCrypt0r”, for which there were 8 total references to in the regshot output. Two changes were made in the Local Machine hive and six were made to the Users hive.

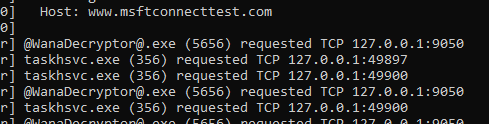


I also clicked on “Decrypt” and the malware outputted some file paths that stated it decrypted them for free. I confirmed this by checking the NOTICES.txt file specified.

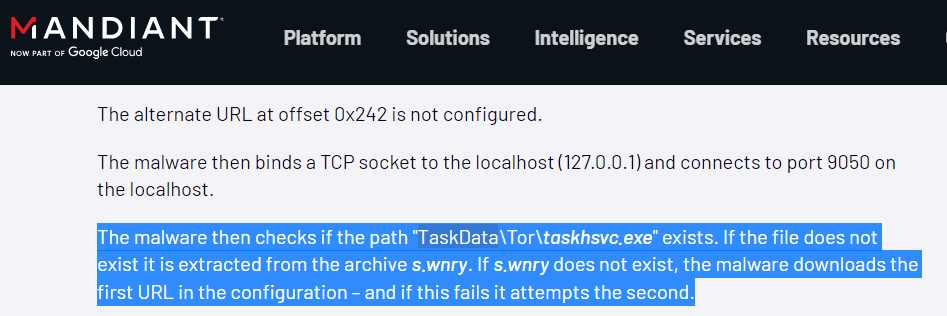


**What network-based indicators could be used to find this malware on infected machines?**

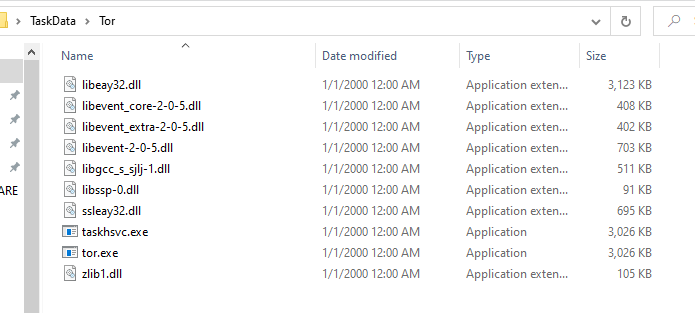
“@WanaDecryptor@.exe” wanted to connect to 127.0.0.1 which is a localhost loopback address. This means that it was simply attempting to establish a TCP connection to the local machine but not an outbound connection.



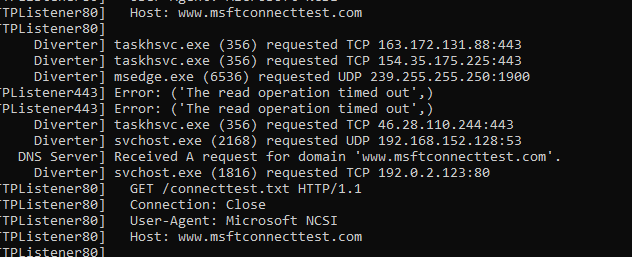
But this was actually expected as it is binds to that socket in order to check if the path Taskdata\Tor\taskhsvc.exe exists (according to Mandiant).



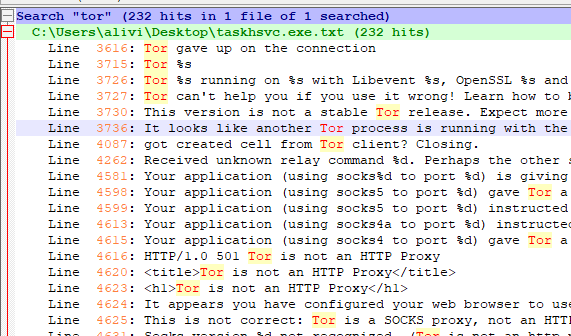
This path was confirmed to exist based on the TaskData folder that was created on the desktop and this folder contained multiple dll files as well as the tor browser executable.



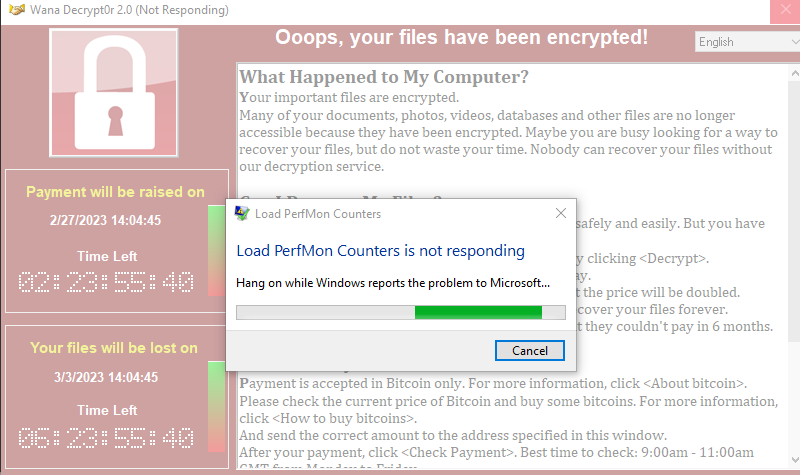
There was another process titled “taskhsvc.exe” that attempted to make multiple outbound TCP connections to various IP addresses. This was located in the folder that was created on the desktop and screenshotted above.



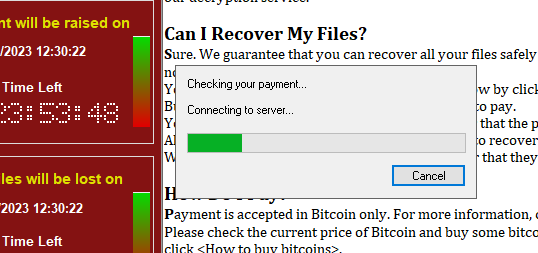
taskhsvc.exe was previously confirmed to be running in Process Explorer and I found 232 mentions to Tor within the strings it produced.



When I attempted to send a message over “Contact Us”, the program decided to not respond.



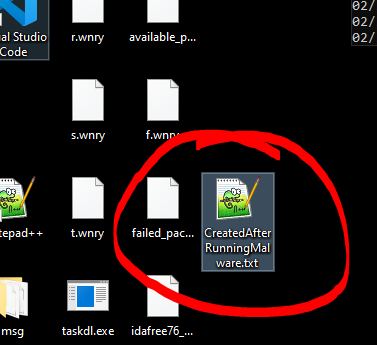
I also clicked on “Check Payment” to prompt more connections. But I believe that the disabling of networking and/or enabling Fakenet did not allow the malware to proceed and I was stuck in the perpetual loading screen.



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

It is clear that this is WannaCry ransomware. It is designed to encrypt any and all files that are .doc, .ppt, .txt, etc. file extensions and demand payment in cryptocurrency over a secure shell connection via the TOR network.

However, after examining the .txt file output from the regshot compare, it occurred to me that the malware did not encrypt it. I created another .txt file on the desktop and let it sit for a while. It did not end up becoming encrypted. Therefore, the ransomware only encrypts files that already existed on the machine before it was ran and does not necessarily render the machine unusable.



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

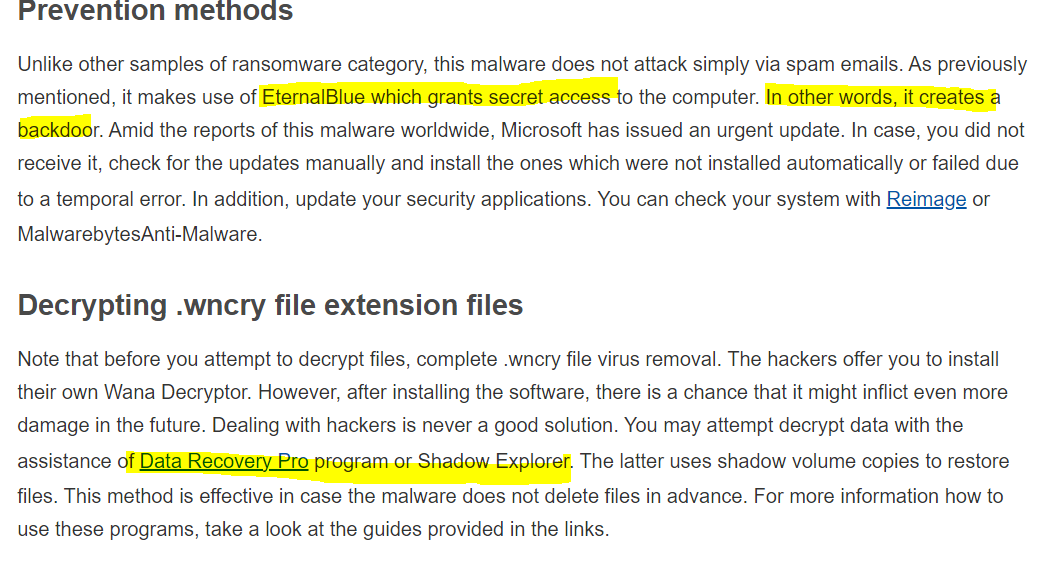
Refer back to the observations portion of this analysis.

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

I could still run it, but it would be very annoying. I deleted the files that the malware created and killed the processes in Process Explorer and the decryption/warning window did not pop up again. The applications still work. I would be very sad if this happened on my personal 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?**

First, I would recommend to NOT pay the ransom. This is highly inadvisable. I would let them know that the WannaCry malware is not installed via clicking on a link in an email, but is installed using the EternalBlue exploit which in essence, creates a backdoor. To decrypt the files, I would use some sort of data recovery software that we might have on-site or to restore from a known good backup. I would then ensure that all antivirus software and other system updates are current. This would take a bit of time, but losing all the data caused by the encryption is much more costly.



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

I could possibly clean the system using some data recovery software, but I could not do it on my own. Luckily, the files that were encrypted weren’t of any value and simply put on the machine for testing purposes. However, some .txt files might serve some important functions for other applications.

* 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: Strings Lecture7.exe | findstr /I “\.exe”

Matthew: