Adrian Necaj
CSEC 202 – Reverse Engineering Fundamentals
4/15/2022

Malware Analysis of WannaCry
Using
MalwareDB aka theZoo

Background of Malware

The name of the malware under analysis is WannaCry. This is a Ransomware that gained its popularity in May of 2017. This ransomware spread across the world, encrypting over two hundred thousand machines. This directly had caused damages costing hundreds of millions of dollars. WannaCry had encrypted files on the systems that were compromised, and demanded a certain amount of ransom, in particularly Bitcoin, to decrypt the files. The ransomware would show two countdowns, one displaying that the payment will be raised after the timer on the screen ran out, along with the other portraying that the files would be lost if the ransom was not paid within that counter's time. WannaCry used a Windows exploit, commonly known as EternalBlue, as a medium to affect as many systems as possible. The EternalBlue vulnerability has since been patched by Microsoft, but many machines had not yet updated their systems, causing them to be vulnerable and still open for a potential WannaCry attack.

The WannaCry binary was downloaded from a malware GitHub called theZoo. When looking at the screenshots, the executable file will be named:

ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe
OR
wannacry.exe

Testing Environment / System

To analyze this WannaCry binary, I had used the following machines:

Physical Machine:

Operating System – Windows 10 Enterprise
Version – 1903
OS Build – 1832.267
Processor: Intel(R) Core™ i7-6700 CPU @ 3.40GHz 3.41GHz
RAM – 16.0 GB
System Architecture – 64-bit operating system, x64-based processor

Virtual Machine (VM):

Operating System – Windows 10 Enterprise
Version – 1703
OS Build – 15063.540
Processor: Intel(R) Core™ i7-6700 CPU @ 3.40GHz 3.41GHz
RAM – 4.0 GB
System Architecture – 64-bit operating system, x64-based processor

Environment:

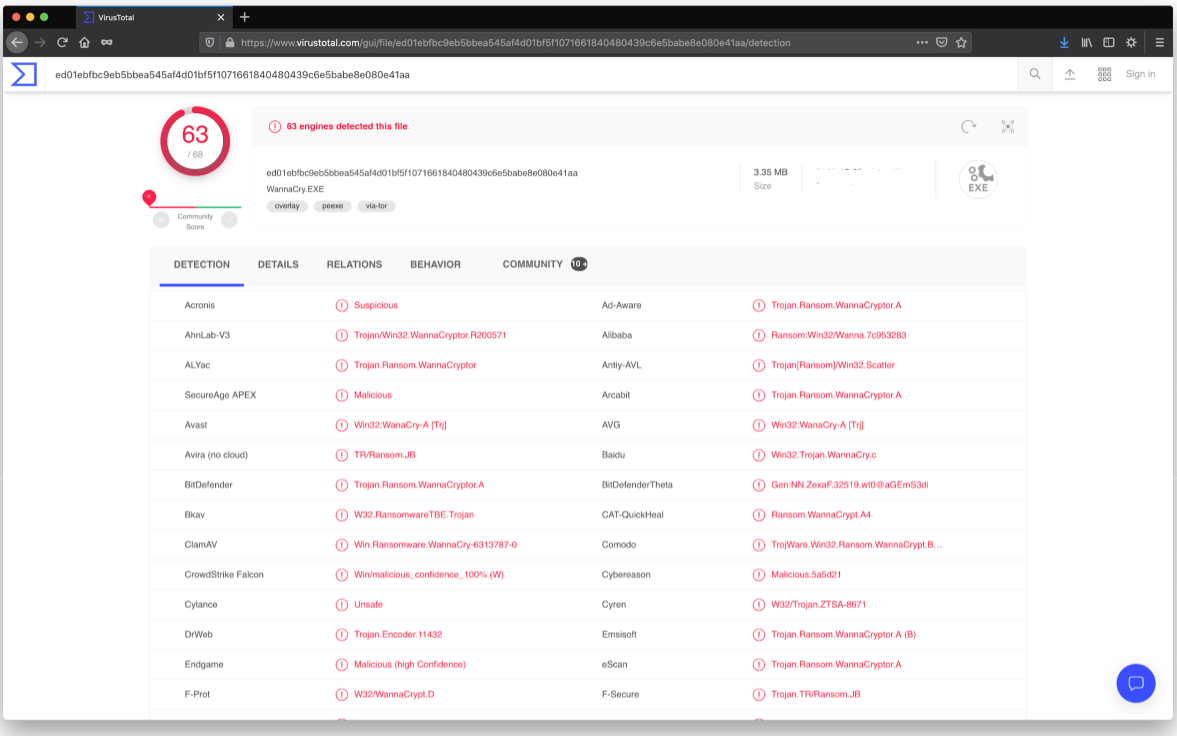
Originally, when running the tools for dynamic analysis along with the advanced dynamic analysis, I had executed and ran the WannaCry malware on a computer in my apartment complex's computer room, APEX. This was on a Microsoft machine, using a Windows 10 Virtual Machine (VM). I tried to emulate an AirGap environment where the computers within the computer room cannot access or communicate with each other or without the networks available. This would ideally prevent worm-like viruses to infect other computers on the same network. I also had installed DeepFreeze on the computer, similar to those at RIT's AirGap lab, where the system would restart the computer and revert the computer back to its original state, before the WannaCry malware was even installed. I then entered the Virtual Machine environment which gave me several advantages and pros. I was able to keep all of my screenshots, tools, and the executable all on the same machine. Once the Virtual Machine was infect, I could obviously no long do any analysis, so I would be able to just revert to a snapshot of the Virtual Machine I had taken prior to running the executable.

Basic Static Analysis

Static analysis tools are tools that are to be used to analyze potential malicious executables without actually physically having to run the executable. These tools can help us get a quick rundown about what the malware might do or act. Some malicious executables will often pack their PE file headers rendering a tool, like PEview & Resource Hacker, useless as they can not analyze anything until it is unpacked. The first step in the reverse engineering process would be to run some static analysis, because the information an analyst could obtain prior to running the executable could be very useful when detailing a report. The tools I will use for my specific static analysis of WannaCry will be VirusTotal, Strings & Floss, PEiD, Dependency Walker, PEview, and Resource Hacker.

VirusTotal

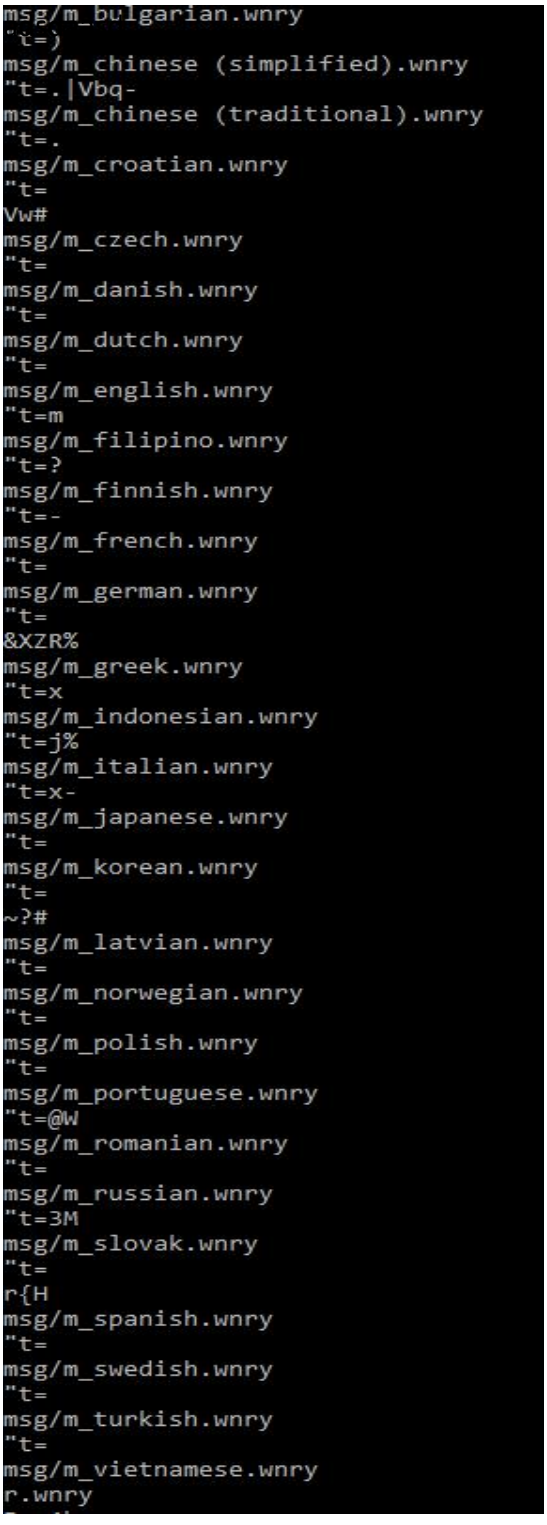
VirusTotal is a helpful static analysis tool if an analyst is unaware of what an executable could potentially do. Once the executable is uploaded, VirusTotal will automatically compare the executable's hash to hashes it has stored in its database. If there were a match, a full report on the executable will be returned along with how it will act once executed, and how dangerous it is. If there is not a match, it means that there has not been an uploaded executable like it before, and the engines will attempt to still determine how malicious the executable is. We have used this before in our labs, so I am quite familiar with it and how it works. Since WannaCry was such an impactful piece of malware, it was easily detected by multiple different detection tools. If we were to have been given just the binary with the random exe name, it still would be easy to find out what type of malware, and what specific malware it is, thanks to VirusTotal.



AegisLab	Undetected	Avast-Mobile	Undetected
CMC	Undetected	Kingsoft	Undetected
SUPERAntiSpyware	Undetected	ESET-NOD32	Timeout
FireEye	Timeout	Symantec Mobile Insight	Unable to process file type
Trustlook	Unable to process file type		

Strings & FLOSS

Strings is a static analysis tool that is ran using the command line interface (CLI). This tool will search the executable for strings, which may or may not be found useful in later stages of our analysis. FLOSS is a tool that was developed by FireEye, now Mandiant, that has the same function as strings, except it will also attempt to decode strings found in the executable.



Strings:

strings.exe
ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe

Strings returned a lot of random text when looking at the WannaCry executable. Nonetheless, I did find that this section had contained the names of different languages followed by a .wnry. I assume that this is so WannaCry can translate its message to those who speak different languages, so that there would be no discrepancy and that all of those affected could pay the ransom.

FLOSS

floss64.exe ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe

```
FLOSS static UTF-16 strings
WanaCrypt0r
Software\
.der
.pfx
.key
.crt
.csr
.p12
.pem
.odt
.ott
.sxw
.stw
.uot
.3ds
.max
.3dm
.ods
.ots
.sxc
.stc
.dif
.slk
.wb2
.odp
.otp
.sxd
.std
.uop
.odg
.otg
.sxm
.mml
.lay
.lay6
.asc
.sqlite3
.sqlitedb
.sql
.accdb
.mdb
.dbf
.odb
.frm
.myd
.myi
.ibd
.mdf
.ldf
.sln
.suo
.cpp
.pas
.asm
.cmd
.bat
.ps1
.vbs
.dip
.dch
.sch
.brd
```

As shown in the screenshot to the left, these are static UTF-16 strings which was not shown with Strings. I saw that, near the top, it read WanaCrypt0r Software. The majority of the screenshot is organized in a .x format, where the x represents file extensions like sql. I am assuming once again that these are the file extensions WannaCry or the Software will look to encrypt. There are even more extensions that are listen that are not shown in this screenshot.

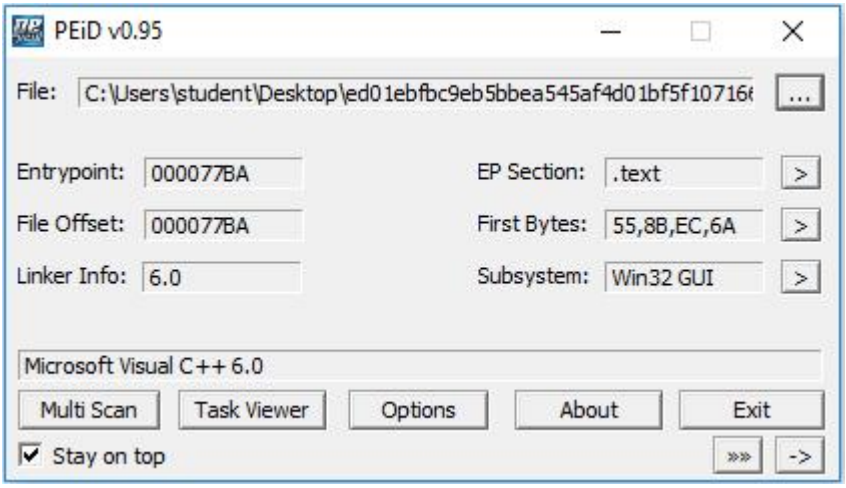
```
FLOSS decoded 3 strings
x?VA
x?VA
pVVA

FLOSS extracted 2 stackstrings
oftware\
9BAA
```

These are the strings that FLOSS decoded which again was not shown with strings. These strings provide me with no idea as to what they do however.

PEiD

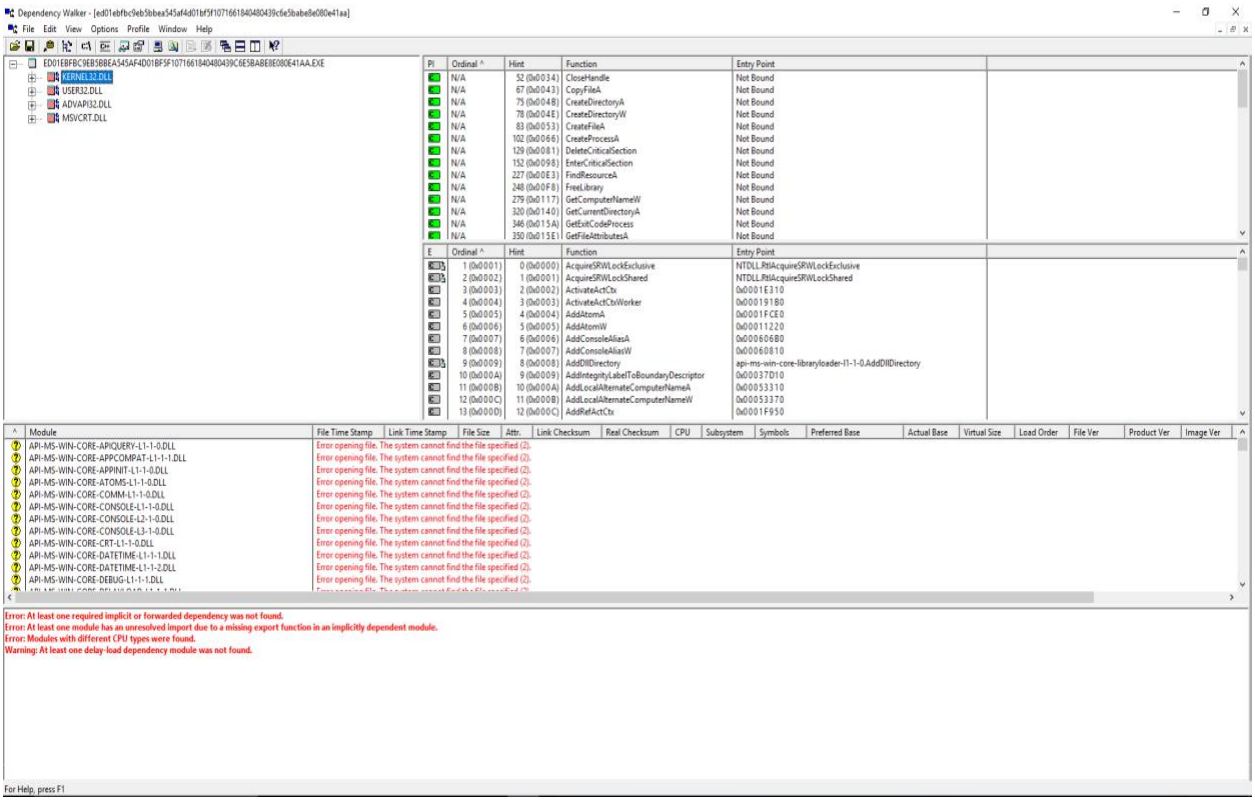
PEiD is a static analysis tool that can determine what an executable is compiled with, and whether or not the executable is packed. This program should be used prior to using PEview because if the executable were to be packed, PEview will not be able to read the headers.



After running this executable through PEiD we have learned that the WannaCry executable is not packed, and is compiled with Microsoft Visual C++ 6.0, aka C++ code. Since the executable is not packed, we will be able to look at the data, text, and resource section through PEview.

Dependency Walker

Dependency Walker is a static analysis tool used to look at the DLLs a malware will be using, and what the DLLs are doing, like system calls. An analyst can then look for any calls they deem valuable later in the advanced static/dynamic phases.

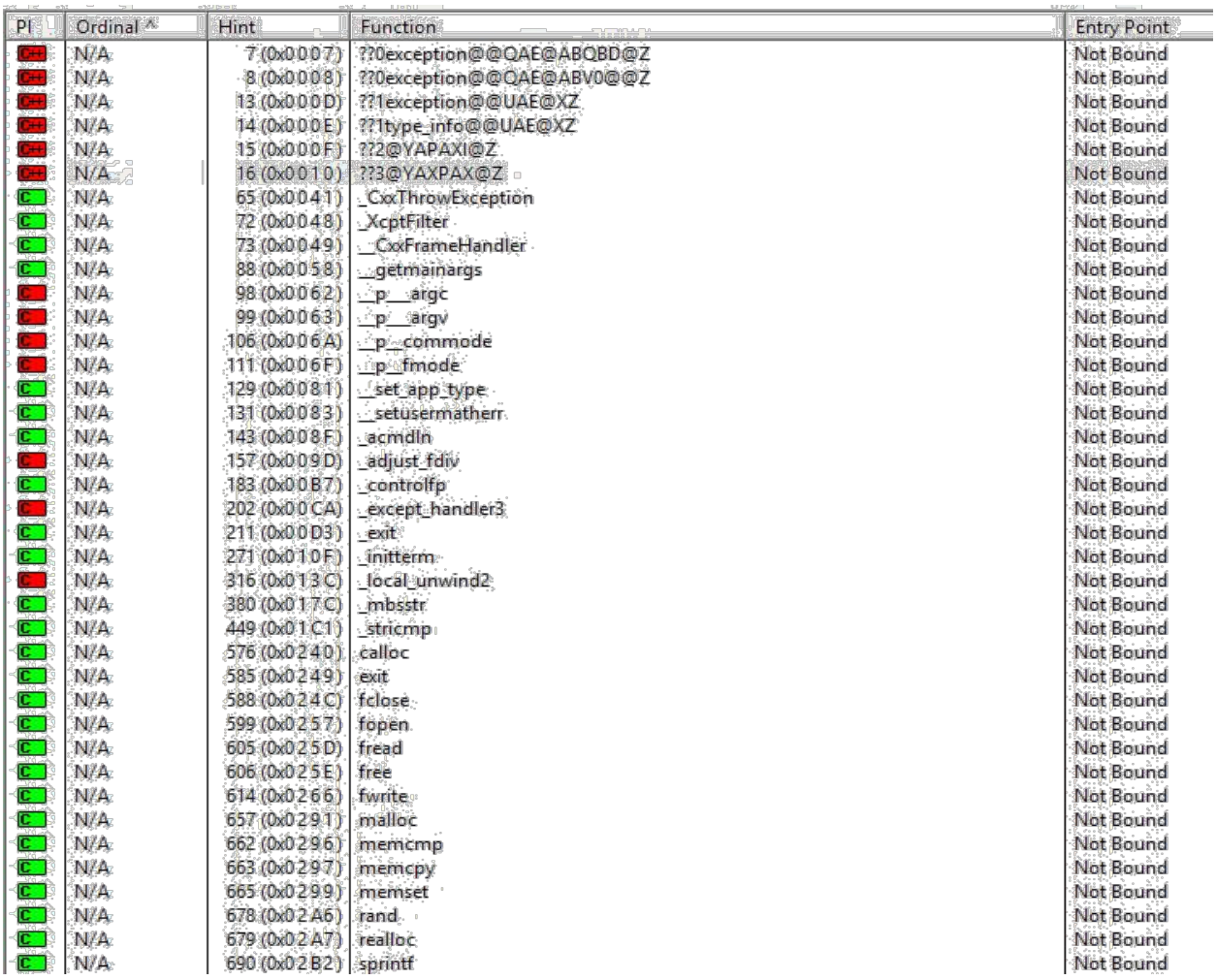


In the screenshot provided above, kernel32.dll is selected and in the righter more pane, are some of the system calls it would use during runtime. Some of the interesting calls that are worth noting are CreateFileA, CreateProcessA, and CreateDirectoryA. This shows me that WannaCry is going to be creating files, processes, and directories on the machine/system once it is executed.

- The kernel32 DLL “is a very common DLL that contains core functionality, such as access and manipulation of memory, files, and hardware.” (malwareanalysis Slides)

- The user32 DLL “contains all the user-interface components, such as buttons, scroll bars, and components for controlling and responding to user actions.” (malwareanalysis Slides)

- The advapi32 DLL “provides access to advanced core Windows components such as the Service Manager and Registry.” (malwareanalysis Slides)



PI	Ordinal ^	Hint	Function	Entry Point
C++	N/A	7 (0x0007)	??0exception@@QAE@ABQBD@Z	Not Bound
C++	N/A	8 (0x0008)	??0exception@@QAE@ABV0@@Z	Not Bound
C++	N/A	13 (0x000D)	??1exception@@UAE@XZ	Not Bound
C++	N/A	14 (0x000E)	??1type_info@@UAE@XZ	Not Bound
C++	N/A	15 (0x000F)	??2@YAPAXI@Z	Not Bound
C++	N/A	16 (0x0010)	??3@YAXPAX@Z	Not Bound
C	N/A	65 (0x0041)	_CxxThrowException	Not Bound
C	N/A	72 (0x0048)	_XcptFilter	Not Bound
C	N/A	73 (0x0049)	__CxxFrameHandler	Not Bound
C	N/A	88 (0x0058)	__getmainargs	Not Bound
C	N/A	98 (0x0062)	__p__argc	Not Bound
C	N/A	99 (0x0063)	__p__argv	Not Bound
C	N/A	106 (0x006A)	__p__commode	Not Bound
C	N/A	111 (0x006F)	__p__fmode	Not Bound
C	N/A	129 (0x0081)	__set_app_type	Not Bound
C	N/A	131 (0x0083)	__setusermatherr	Not Bound
C	N/A	143 (0x008F)	_acmdln	Not Bound
C	N/A	157 (0x009D)	_adjust_fdiv	Not Bound
C	N/A	183 (0x00B7)	_controlfp	Not Bound
C	N/A	202 (0x00CA)	_except_handler3	Not Bound
C	N/A	211 (0x00D3)	_exit	Not Bound
C	N/A	271 (0x010F)	_initterm	Not Bound
C	N/A	316 (0x013C)	_local_unwind2	Not Bound
C	N/A	380 (0x017C)	_mbsstr	Not Bound
C	N/A	449 (0x01C1)	_stricmp	Not Bound
C	N/A	576 (0x0240)	calloc	Not Bound
C	N/A	585 (0x0249)	exit	Not Bound
C	N/A	588 (0x024C)	fclose	Not Bound
C	N/A	599 (0x0257)	fopen	Not Bound
C	N/A	605 (0x025D)	fread	Not Bound
C	N/A	606 (0x025E)	free	Not Bound
C	N/A	614 (0x0266)	fwrite	Not Bound
C	N/A	657 (0x0291)	malloc	Not Bound
C	N/A	662 (0x0296)	memcmp	Not Bound
C	N/A	663 (0x0297)	memcpy	Not Bound
C	N/A	665 (0x0299)	memset	Not Bound
C	N/A	678 (0x02A6)	rand	Not Bound
C	N/A	679 (0x02A7)	realloc	Not Bound
C	N/A	690 (0x02B2)	sprintf	Not Bound

From further research online, the msvcrt DLL contains C library functions. This new screenshot above shows that the functions calls are only from msvcrt.dll. After analyzing, I still do not know what those first six or so lines of functions calls refer to or mean, as I have never seen something like that before and they do not resemble normal C library functions.

Sources:

- malwareanalysis Slides. Slide 28 - https://mycourses.rit.edu/d2l/le/content/960635/viewContent/8030806/View

PEview

PEview is a static analysis tool useful for looking at PE headers within an executable. PE headers are usually broken up into .text, .data, .rdata, .idata, .edata, .pdata, .rsrc, and .reloc. For my analysis I focused on the .rdata section, as this "holds read-only data that is globally accessible within the program," .data which "stores global data accessed throughout the program," .rsrc which "stores resources needed by the executable.

pFile	Data	Description	Value
00008000	0000DC2A	Hint/Name RVA	0064 CreateServiceA
00008004	0000DC62	Hint/Name RVA	01AF OpenServiceA
00008008	0000DC52	Hint/Name RVA	0249 StartServiceA
0000800C	0000DC3C	Hint/Name RVA	003E CloseServiceHandle
00008010	0000DC14	Hint/Name RVA	00A0 CryptReleaseContext
00008014	0000DC04	Hint/Name RVA	01D3 RegCreateKeyW
00008018	0000DBF2	Hint/Name RVA	0204 RegSetValueExA
0000801C	0000DBDE	Hint/Name RVA	01F7 RegQueryValueExA
00008020	0000DBD0	Hint/Name RVA	01CB RegCloseKey
00008024	0000DC72	Hint/Name RVA	01AD OpenSCManagerA
00008028	00000000	End of Imports	ADVAPI32.dll
0000802C	0000D8FC	Hint/Name RVA	0161 GetFileAttributesW
00008030	0000D912	Hint/Name RVA	0164 GetFileSizeEx
00008034	0000D922	Hint/Name RVA	0053 CreateFileA
00008038	0000D930	Hint/Name RVA	0223 InitializeCriticalSection
0000803C	0000D94C	Hint/Name RVA	0081 DeleteCriticalSection
00008040	0000D964	Hint/Name RVA	02B5 ReadFile
00008044	0000D970	Hint/Name RVA	0163 GetFileSize
00008048	0000D97E	Hint/Name RVA	03A4 WriteFile
0000804C	0000D98A	Hint/Name RVA	0251 LeaveCriticalSection
00008050	0000D9A2	Hint/Name RVA	0098 EnterCriticalSection
00008054	0000D9BA	Hint/Name RVA	031A SetFileAttributesW
00008058	0000D9D0	Hint/Name RVA	030B SetCurrentDirectoryW
0000805C	0000D9E8	Hint/Name RVA	004E CreateDirectoryW
00008060	0000D9FC	Hint/Name RVA	01D6 GetTempPathW
00008064	0000DA0C	Hint/Name RVA	01F4 GetWindowsDirectoryW
00008068	0000DA24	Hint/Name RVA	015E GetFileAttributesA
0000806C	0000DA3A	Hint/Name RVA	0355 SizeofResource
00008070	0000DA4C	Hint/Name RVA	0265 LockResource
00008074	0000DA5C	Hint/Name RVA	0257 LoadResource
00008078	0000D8E6	Hint/Name RVA	0275 MultiByteToWideChar
0000807C	0000DA7C	Hint/Name RVA	0356 Sleep
00008080	0000DA84	Hint/Name RVA	0284 OpenMutexA
00008084	0000DA92	Hint/Name RVA	0169 GetFullPathNameA
00008088	0000DAA6	Hint/Name RVA	0043 CopyFileA
0000808C	0000DAB2	Hint/Name RVA	017D GetModuleFileNameA
00008090	0000DAC8	Hint/Name RVA	0381 VirtualAlloc
00008094	0000DAD8	Hint/Name RVA	0383 VirtualFree
00008098	0000DAE6	Hint/Name RVA	00F8 FreeLibrary
0000809C	0000DAF4	Hint/Name RVA	0210 HeapAlloc
000080A0	0000DB00	Hint/Name RVA	01A3 GetProcessHeap
000080A4	0000DB12	Hint/Name RVA	017F GetModuleHandleA
000080A8	0000DB26	Hint/Name RVA	0328 SetLastError
000080AC	0000DB36	Hint/Name RVA	0386 VirtualProtect
000080B0	0000DB48	Hint/Name RVA	0233 IsBadReadPtr
000080B4	0000DB58	Hint/Name RVA	0216 HeapFree
000080B8	0000DB64	Hint/Name RVA	035B SystemTimeToFileTime
000080BC	0000DB7C	Hint/Name RVA	025A LocalFileTimeToFileTime
000080C0	0000DB96	Hint/Name RVA	004B CreateDirectoryA
000080C4	0000DF5E	Hint/Name RVA	01B7 GetStartupInfoA
000080C8	0000D8D4	Hint/Name RVA	031B SetFilePointer

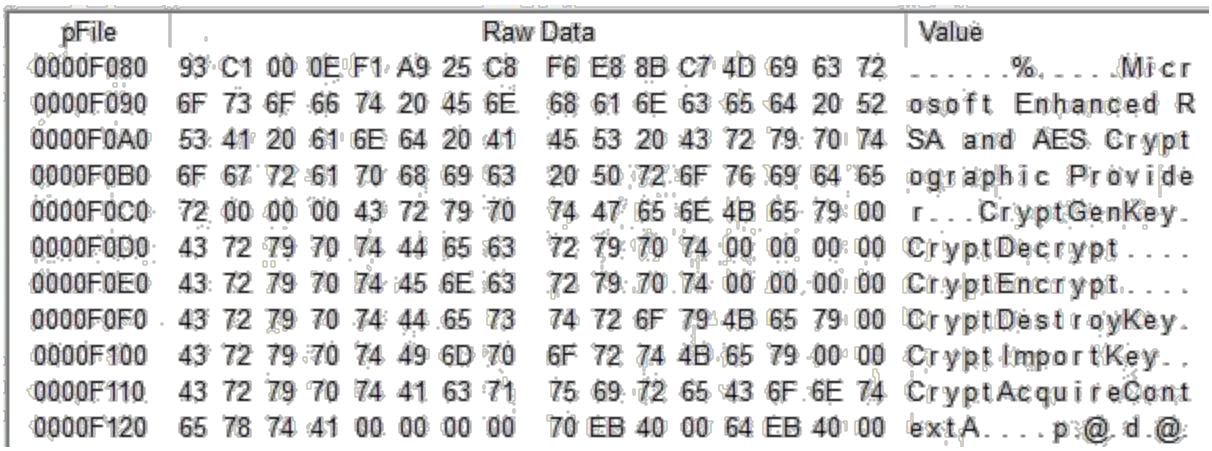
This is what PEview had returned for the "SECTION .rdata – IMPORT Address Table." In the screenshot I had noticed the various system calls WannaCry uses. Some valuable calls that are noticeable are as follows: OpenServiceA, StartServiceA, CreateServiceA, RegCreateKeyW, RegSetValueExA, and more. This again shows that WannaCry will be looking to create services, start them, create registry keys, and setting values to those very keys. There are more important calls that are listed within the screenshot and some that are not, but can be seen within PEview.

pFile	Data	Description	Value
0000D5A8	0000D638	Import Name Table RVA	
0000D5AC	00000000	Time Date Stamp	
0000D5B0	00000000	Forwarder Chain	
0000D5B4	0000DBAA	Name RVA	KERNEL32.dll
0000D5B8	0000802C	Import Address Table RVA	
0000D5BC	0000D7DC	Import Name Table RVA	
0000D5C0	00000000	Time Date Stamp	
0000D5C4	00000000	Forwarder Chain	
0000D5C8	0000DBC4	Name RVA	USER32.dll
0000D5CC	000081D0	Import Address Table RVA	
0000D5D0	0000D60C	Import Name Table RVA	
0000D5D4	00000000	Time Date Stamp	
0000D5D8	00000000	Forwarder Chain	
0000D5DC	0000DC84	Name RVA	ADVAPI32.dll
0000D5E0	00008000	Import Address Table RVA	
0000D5E4	0000D714	Import Name Table RVA	
0000D5E8	00000000	Time Date Stamp	
0000D5EC	00000000	Forwarder Chain	
0000D5F0	0000DE88	Name RVA	MSVCRT.dll
0000D5F4	00008108	Import Address Table RVA	
0000D5F8	00000000		
0000D5FC	00000000		
0000D600	00000000		
0000D604	00000000		
0000D608	00000000		

This is what PEview had returned for the “SECTION .rdata – IMPORT Directory Table.” The directory table is one of the many ways an analyst can see what DLLs a malware will be utilizing. In the case of our WannaCry executable, it will be utilizing kernel32.dll, user32.dll, advapi32.dll, and msvcrt.dll at the minimum.

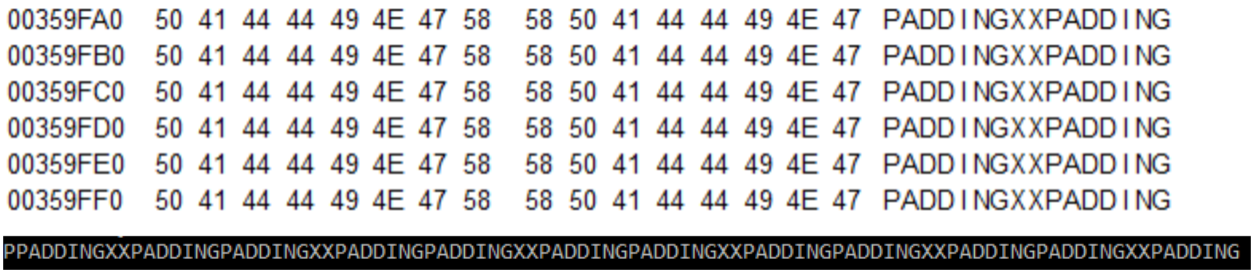
pFile	Raw Data																Value
0000EB70	2E	00	64	00	6F	00	63	00	00	00	00	00	57	41	4E	41	. . d . o . c . . . . . WANA
0000EB80	43	52	59	21	00	00	00	00	25	00	73	00	5C	00	25	00	CRY! . . . . % . s . \ . % .
0000EB90	73	00	00	00	43	6C	6F	73	65	48	61	6E	64	6C	65	00	s . . . C l o s e H a n d l e .
0000EBA0	44	65	6C	65	74	65	46	69	6C	65	57	00	4D	6F	76	65	D e l e t e F i l e W . M o v e
0000EBB0	46	69	6C	65	45	78	57	00	4D	6F	76	65	46	69	6C	65	F i l e E x W . M o v e F i l e
0000EBC0	57	00	00	00	52	65	61	64	46	69	6C	65	00	00	00	00	W . . . R e a d F i l e . . . .
0000EBD0	57	72	69	74	65	46	69	6C	65	00	00	00	43	72	65	61	W r i t e F i l e . . . C r e a
0000EBE0	74	65	46	69	6C	65	57	00	6B	65	72	6E	65	6C	33	32	t e F i l e W . k e r n e l 3 2
0000EBF0	2E	64	6C	6C	00	00	00	00	07	02	00	00	00	A4	00	00	. d l l . . . . .

This screenshot shows what PEview returned for the “SECTION .data”. This shows the name of the malware, WANACRY, the name of some system calls, along with the kernel32.dll.



pFile	Raw Data	Value
0000F080	93 C1 00 0E F1 A9 25 C8 F6 E8 8B C7 4D 69 63 72	......%.....Micr
0000F090	6F 73 6F 66 74 20 45 6E 68 61 6E 63 65 64 20 52	osoft Enhanced R
0000F0A0	53 41 20 61 6E 64 20 41 45 53 20 43 72 79 70 74	SA and AES Crypt
0000F0B0	6F 67 72 61 70 68 69 63 20 50 72 6F 76 69 64 65	ographic Provide
0000F0C0	72 00 00 00 43 72 79 70 74 47 65 6E 4B 65 79 00	r...CryptGenKey.
0000F0D0	43 72 79 70 74 44 65 63 72 79 70 74 00 00 00 00	CryptDecrypt....
0000F0E0	43 72 79 70 74 45 6E 63 72 79 70 74 00 00 00 00	CryptEncrypt....
0000F0F0	43 72 79 70 74 44 65 73 74 72 6F 79 4B 65 79 00	CryptDestroyKey.
0000F100	43 72 79 70 74 49 6D 70 6F 72 74 4B 65 79 00 00	CryptImportKey..
0000F110	43 72 79 70 74 41 63 71 75 69 72 65 43 6F 6E 74	CryptAcquireCont
0000F120	65 78 74 41 00 00 00 00 70 EB 40 00 64 EB 40 00	extA....p.@.d.@.

This screenshot is another one returned from “SECTION .data”. I wanted to show this screenshot because WannaCry is a type of ransomware that likes to encrypt files on the system and in the screenshot there are encryption calls shown on the right like, CryptGenKey, CryptEncrypt, CryptDestroyKey, and more.



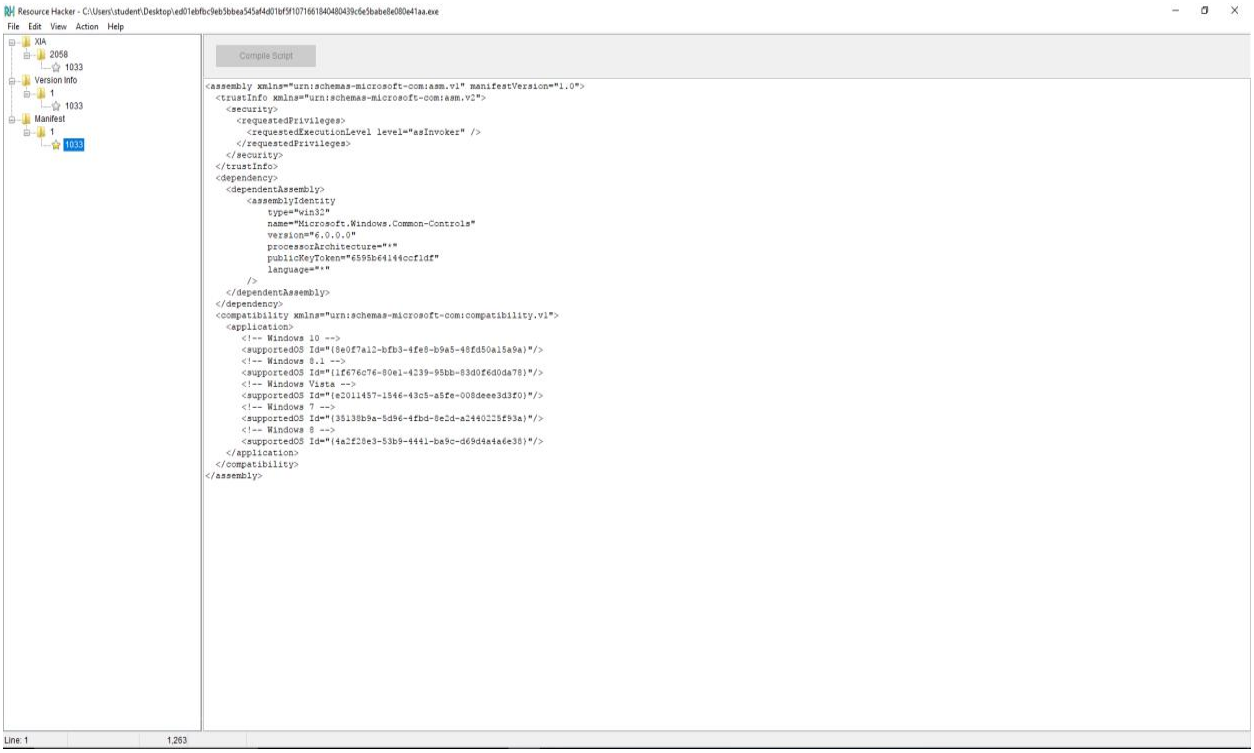
00359FA0	50 41 44 44 49 4E 47 58 58 50 41 44 44 49 4E 47	PADDINGXXPADDING
00359FB0	50 41 44 44 49 4E 47 58 58 50 41 44 44 49 4E 47	PADDINGXXPADDING
00359FC0	50 41 44 44 49 4E 47 58 58 50 41 44 44 49 4E 47	PADDINGXXPADDING
00359FD0	50 41 44 44 49 4E 47 58 58 50 41 44 44 49 4E 47	PADDINGXXPADDING
00359FE0	50 41 44 44 49 4E 47 58 58 50 41 44 44 49 4E 47	PADDINGXXPADDING
00359FF0	50 41 44 44 49 4E 47 58 58 50 41 44 44 49 4E 47	PADDINGXXPADDING

```
PPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDING
```

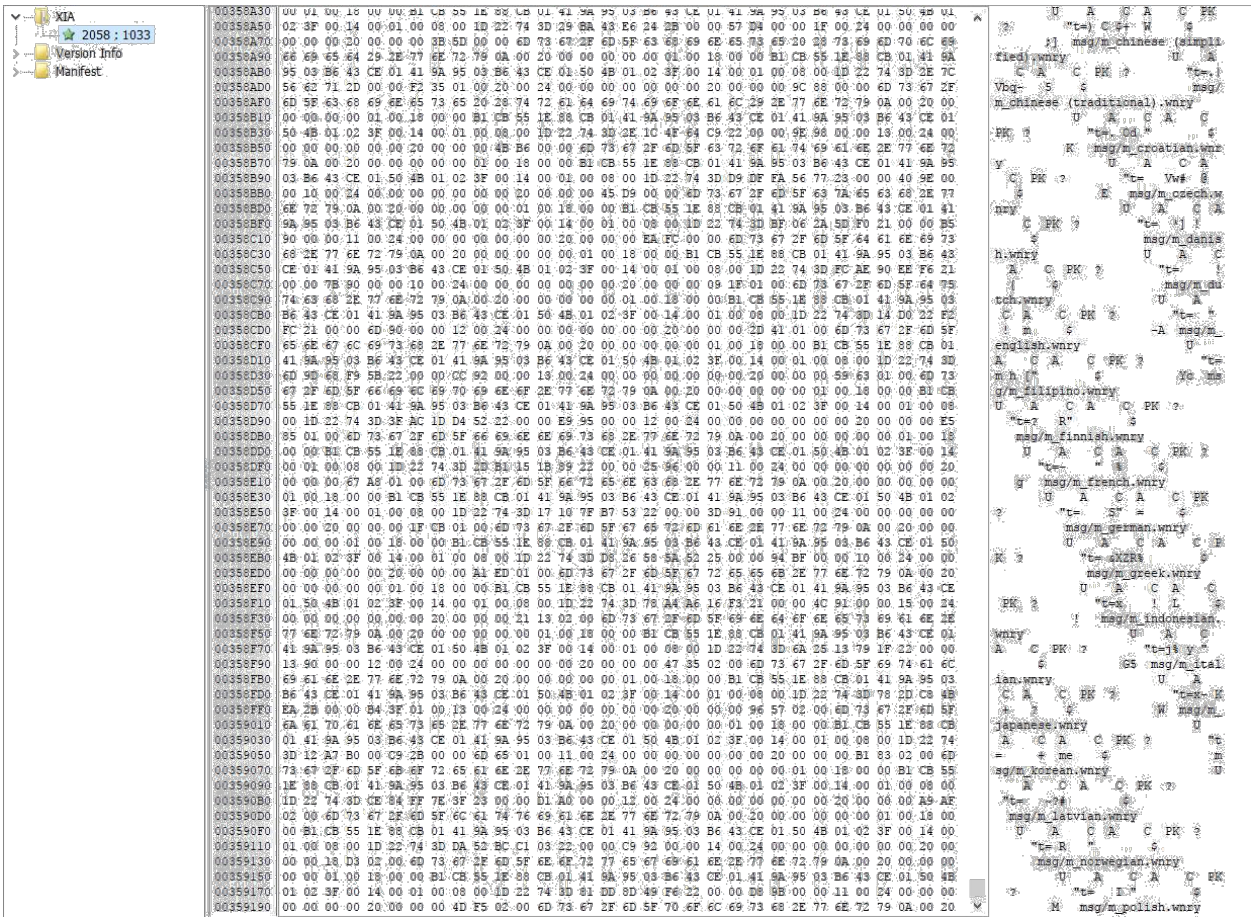
In this screenshot we can see the pattern of “PADDINGXXPADDING” at the bottom of the resource section in PEview. I had seen this at the end of the strings command line tool as well, which is why I chose to include it here as well, although I am not sure as to what it means exactly.

Resource Hacker

Resource Hacker is a static analysis tool that will help us look at the resource section of PE headers. An analyst can review these files and find something useful about the malware. The thing is, that if the executable is packed a tool like Resource Hacker is useless due to the fact it will have no resource section until it is unpacked.



Resource Hacker had shown us three other files WannaCry has within it, including, “XIA”, “VERSION”, and “MANIFEST”. I had also noticed these earlier in PEview. They were contained within the resource section, .rsrc. In the Manifest section listed above, there were tags setting its name and type to make it seem camouflaged, and look like a Windows product.



The XIA resource file, towards the end, show the languages we saw earlier with FLOSS, just confirming what we already know which is always good!

Dynamic Analysis

The dynamic analysis phase entails an analyst running malware and reporting on how it acts. This phase makes it vital that an analyst is in a safe environment, such as an AirGap, where the malware will not affect their physical machine, or any other machine/computer using the same network. If an analyst were to not be careful in this phase, the malware could escape the virtual machine and harm their computer and spread to other computers on the network. For this particular analysis of WannaCry I will be using several tools including: Regshot, Procman, Process Explorer, Regshot, ApateDNS, Netcat, and Wireshark.

Regshot

Regshot is a dynamic analysis tool that is useful for looking at registry changes within the “two shots.” Prior to running the malware an analyst should take the first shot of the system to capture what is already on the system, in terms of registry keys. An analyst should then run the malware on the system and anytime after that take the second shot. Regshot will then compare the two shots and save the output to a plain text file or a HTML document. The output will show any keys deleted or added and values deleted, added, or modified from registry keys. Regshot shows edits that occur to the Windows Registry. In some instances that should be pointed out are the addition of the WanaCrypt0r key, the addition of certain values like the malware executable, and the modification of the background image.

Keys added: 15

HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:0000000000030336
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Microsoft\Windows\CurrentVersion\Search\RecentApps\{FACC4B16-2298-4A1F-8D72-1B4D9B450124}
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Classes\Local Settings\MrtCache\C:%5CProgram Files%5CWindowsApps%5CMicrosoft.Windows.Photos_2017.35071.13510.0_x64__8wekyb3d8bbwe%5CMicrosoft.System.Package.Metadata%5CS-1-5-21-834678334-2673773515-2042175648-1001-Me
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Classes\VirtualStore
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Classes\VirtualStore\MACHINE
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Classes\VirtualStore\MACHINE\SOFTWARE
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Classes\VirtualStore\MACHINE\SOFTWARE\WOW6432Node
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Classes\VirtualStore\MACHINE\SOFTWARE\WOW6432Node\WanaCrypt0r
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\WanaCrypt0r
HKU\S-1-5-21-834678334-2673773515-2042175648-1001_Classes\Local Settings\MrtCache\C:%5CProgram Files%5CWindowsApps%5CMicrosoft.Windows.Photos_2017.35071.13510.0_x64__8wekyb3d8bbwe%5CMicrosoft.System.Package.Metadata%5CS-1-5-21-834678334-2673773515-2042175648-1001-MergedResc
HKU\S-1-5-21-834678334-2673773515-2042175648-1001_Classes\VirtualStore
HKU\S-1-5-21-834678334-2673773515-2042175648-1001_Classes\VirtualStore\MACHINE
HKU\S-1-5-21-834678334-2673773515-2042175648-1001_Classes\VirtualStore\MACHINE\SOFTWARE
HKU\S-1-5-21-834678334-2673773515-2042175648-1001_Classes\VirtualStore\MACHINE\SOFTWARE\WOW6432Node
HKU\S-1-5-21-834678334-2673773515-2042175648-1001_Classes\VirtualStore\MACHINE\SOFTWARE\WOW6432Node\WanaCrypt0r

Values added: 13

HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\fghqrag\Qrfxgbc\rq01rosop9ro5oorn545ns4q01os5s1071661840480439p6r5onor8r080r41nn.rkr: 00 00 00 00 01 00 00 00 00 00 00 0
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:0000000000030336\VirtualDesktop: 10 00 00 00 30 30 44 56 C1 9D E3 19 3D 96 CE 4A A1 22 9F D3 9D 75 88 E9
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Microsoft\Windows\CurrentVersion\Run\pmcuwdekduxkp076: ""C:\Users\student\Desktop\tasksche.exe""
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Microsoft\Windows\CurrentVersion\Search\RecentApps\{FACC4B16-2298-4A1F-8D72-1B4D9B450124}\LastAccessedTime: 70 6D 14 28 C0 B2 D5 01
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Microsoft\Windows\CurrentVersion\Search\RecentApps\{FACC4B16-2298-4A1F-8D72-1B4D9B450124}\AppId: "C:\Users\student\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe"
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Microsoft\Windows\CurrentVersion\Search\RecentApps\{FACC4B16-2298-4A1F-8D72-1B4D9B450124}\LaunchCount: 0x00000001
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Microsoft\Windows\CurrentVersion\Search\RecentApps\{FACC4B16-2298-4A1F-8D72-1B4D9B450124}\AppPath: ""
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store\C:\Users\student\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe: 53 41 43 50 01 00 00 00 00 00 00 00 07 00 00 00 28 0
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Classes\Local Settings\MrtCache\C:%5CProgram Files%5CWindowsApps%5CMicrosoft.Windows.Photos_2017.35071.13510.0_x64__8wekyb3d8bbwe%5CMicrosoft.System.Package.Metadata%5CS-1-5-21-834678334-2673773515-2042175648-1001-Me
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\Classes\Local Settings\MrtCache\C:%5CProgram Files%5CWindowsApps%5CMicrosoft.Windows.Photos_2017.35071.13510.0_x64__8wekyb3d8bbwe%5CMicrosoft.System.Package.Metadata%5CS-1-5-21-834678334-2673773515-2042175648-1001-Me
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Software\WanaCrypt0r\wd: "C:\Users\student\Desktop"
HKU\S-1-5-21-834678334-2673773515-2042175648-1001_Classes\Local Settings\MrtCache\C:%5CProgram Files%5CWindowsApps%5CMicrosoft.Windows.Photos_2017.35071.13510.0_x64__8wekyb3d8bbwe%5CMicrosoft.System.Package.Metadata%5CS-1-5-21-834678334-2673773515-2042175648-1001-MergedResc
HKU\S-1-5-21-834678334-2673773515-2042175648-1001_Classes\Local Settings\MrtCache\C:%5CProgram Files%5CWindowsApps%5CMicrosoft.Windows.Photos_2017.35071.13510.0_x64__8wekyb3d8bbwe%5CMicrosoft.System.Package.Metadata%5CS-1-5-21-834678334-2673773515-2042175648-1001-MergedResc

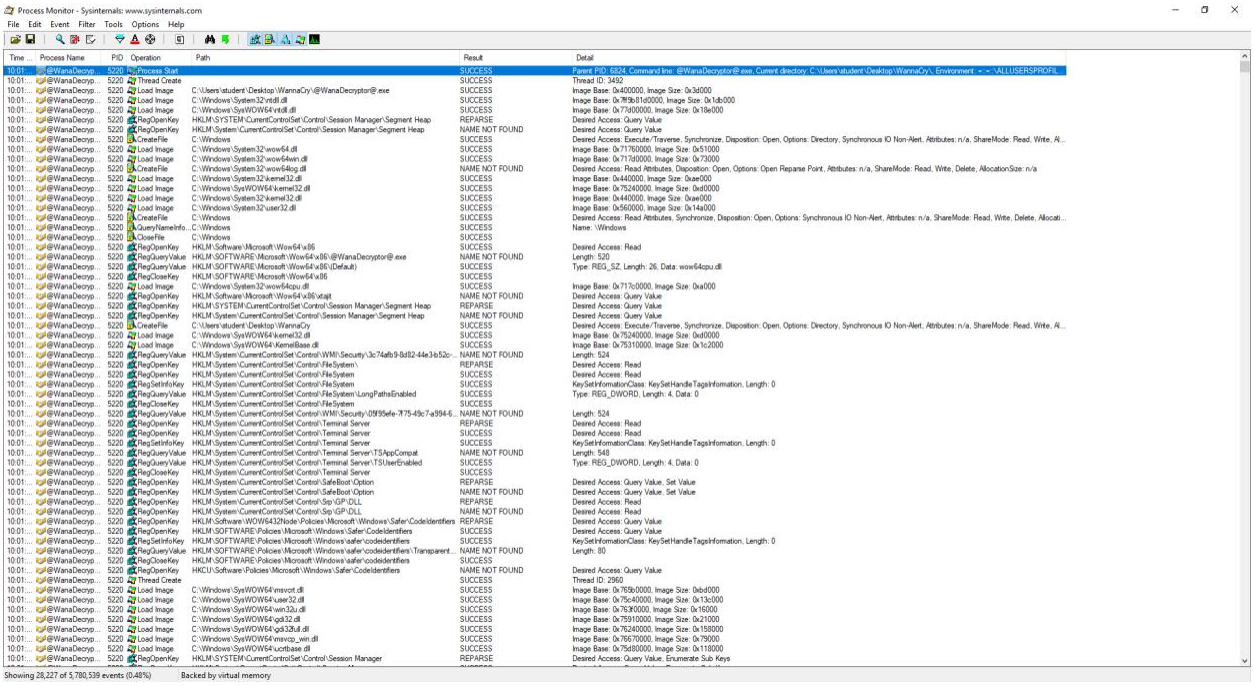
Values modified: 19

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Notifications\Data\418A073AA3BC3475: BF 00 00 00 00 00 00 00 04 00 04 00 01 02 08 00 00 00 00 00 0C 00 00 00 0D 78 79 00 01 00 00 00 4A AA 81 00 05 00 00 00 6B 50 7E 00 02 00 00 00 81 06 95 00 B8 01 00 00 A1 9F 5E 00 05 00 00 00 E6 C5 31 00 A2
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Notifications\Data\418A073AA3BC3475: C0 00 00 00 00 00 00 00 04 00 04 00 01 02 08 00 00 00 00 00 0C 00 00 00 0D 78 79 00 01 00 00 00 4A AA 81 00 05 00 00 00 6B 50 7E 00 02 00 00 00 81 06 95 00 B8 01 00 00 A1 9F 5E 00 05 00 00 00 E6 C5 31 00 A2
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks\{A48ED662-E9D0-4763-A3E4-8F78FC3CB5EF}\Hash: B5 52 D6 11 44 AF D8 45 55 83 A3 7D 3C CD 63 AD 22 AC D1 FA BC 5D 6D E0 AA 9D 3E E0 EF 03 C5 F4
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks\{A48ED662-E9D0-4763-A3E4-8F78FC3CB5EF}\Hash: CA AB 64 72 41 BE 09 99 59 40 66 D5 10 7A AE 89 22 5E F8 4D 5F 02 66 F7 F4 A1 E5 55 25 F5 31 C5
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks\{A48ED662-E9D0-4763-A3E4-8F78FC3CB5EF}\Triggers: 17 00 00 00 00 00 00 00 07 0C 00 00 00 0F 00 00 49 1B 7B 7C B3 D5 01 00 07 0C 00 00 00 0F 00 FF FF FF FF FF FF FF 48 21 C2 02 48 48 48 48 0C FE 48 44 48 48 48 48 4
00 00 00 00 00 01 05 D2 01 00 00 00 00 00 00 00 F1 EF 00 00 00 00 00 00 48 48 48 48
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks\{A48ED662-E9D0-4763-A3E4-8F78FC3CB5EF}\Triggers: 17 00 00 00 00 00 00 00 07 0C 00 00 00 0F 00 00 A4 F1 2D 89 B3 D5 01 00 07 0C 00 00 00 0F 00 FF FF FF FF FF FF FF 48 21 C2 02 48 48 48 48 1D BD D0 75 48 48 48 4
00 00 00 00 00 01 05 D2 01 00 00 00 00 00 00 00 F1 EF 00 00 00 00 00 00 48 48 48 48
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileService\References\S-1-5-21-834678334-2673773515-2042175648-1001\RefCount: 03 00 00 00
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileService\References\S-1-5-21-834678334-2673773515-2042175648-1001\RefCount: 04 00 00 00
HKLM\SOFTWARE\Microsoft\Windows Search\UsnNotifier\Windows\Catalogs\SystemIndex\{37145A3B-0000-0000-0000-501F00000000}: "143743512"
HKLM\SOFTWARE\Microsoft\Windows Search\UsnNotifier\Windows\Catalogs\SystemIndex\{37145A3B-0000-0000-0000-501F00000000}: "146604472"
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Control Panel\Desktop\WallPaper: "C:\Windows\web\wallpaper\Windows\img0.jpg"
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Control Panel\Desktop\WallPaper: "C:\Users\student\Desktop\@WanaDecryptor@.bmp"
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Control Panel\Desktop\TranscodedImageCache: 7A C3 01 00 0A 48 01 00 00 04 00 00 00 03 00 00 08 25 8A 2D 2A A0 D2 01 43 00 3A 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00 5C 00 77 00 65 00 62 00 5C 00 77 00 61 00 6C 00 6C 00 70 00 61 00
HKU\S-1-5-21-834678334-2673773515-2042175648-1001\Control Panel\Desktop\TranscodedImageCache: 7A C3 01 00 36 F9 15 00 20 03 00 00 58 02 00 00 00 F0 B5 16 D6 CA D2 01 43 00 3A 00 5C 00 55 00 73 00 65 00 72 00 73 00 5C 00 73 00 74 00 75 00 64 00 65 00 6E 00 74 00 5C 00 44 00 65 00 73 00 6B 00

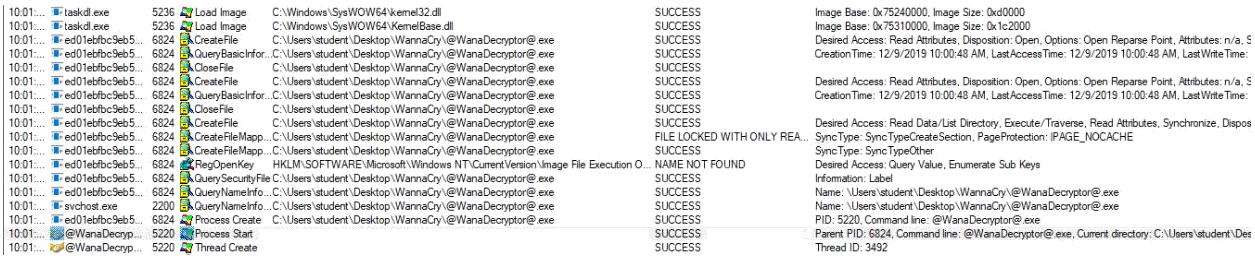
Procman

Procman, otherwise known as Process Monitor, is a very useful dynamic analysis tool that allows an analyst to look at all processes on the system and how they are interacting with the system. It also contains a filter option which we can utilize to search for certain interactions, like only searching for files that were created or processed by a company name.

I decided to use the filter “Operation is “CreateFile” then Include” for the above screenshot. This helped portray all the files WannaCry had attempted to create and whether or not they were successful.



For this next screenshot I used the filter [“Command Line is “@WanaDecryptor@.exe](#)” then Include”. This screenshot is bigger than the last, as it contains most of what WannaCry is doing / trying to do behind the scenes. There are images loaded, registry keys opened, processes started, setting values to those registry keys and more.

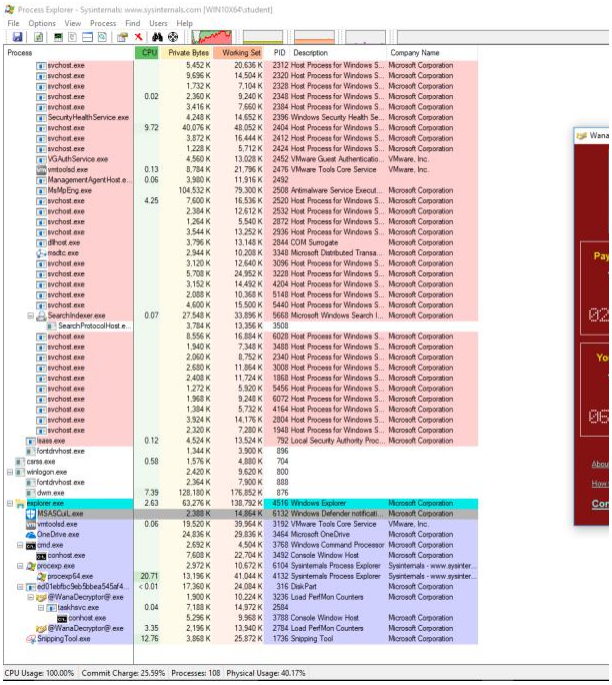


10:01:...	taskdl.exe	5236	Load Image	C:\Windows\SysWOW64\kernel32.dll	SUCCESS	Image Base: 0x75240000, Image Size: 0xd0000
10:01:...	taskdl.exe	5236	Load Image	C:\Windows\SysWOW64\KernelBase.dll	SUCCESS	Image Base: 0x75310000, Image Size: 0x1c2000
10:01:...	ed01ebfbc9eb5...	6824	CreateFile	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, S
10:01:...	ed01ebfbc9eb5...	6824	QueryBasicInfor...	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	CreationTime: 12/9/2019 10:00:48 AM, LastAccessTime: 12/9/2019 10:00:48 AM, LastWriteTime:
10:01:...	ed01ebfbc9eb5...	6824	CloseFile	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	
10:01:...	ed01ebfbc9eb5...	6824	CreateFile	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, S
10:01:...	ed01ebfbc9eb5...	6824	QueryBasicInfor...	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	CreationTime: 12/9/2019 10:00:48 AM, LastAccessTime: 12/9/2019 10:00:48 AM, LastWriteTime:
10:01:...	ed01ebfbc9eb5...	6824	CloseFile	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	
10:01:...	ed01ebfbc9eb5...	6824	CreateFile	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	Desired Access: Read Data/List Directory, Execute/Traverse, Read Attributes, Synchronize, Dispos
10:01:...	ed01ebfbc9eb5...	6824	CreateFileMapp...	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	FILE LOCKED WITH ONLY REA...	SyncType: SyncTypeCreateSection, PageProtection: IPAGE_NOCACHE
10:01:...	ed01ebfbc9eb5...	6824	CreateFileMapp...	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	SyncType: SyncTypeOther
10:01:...	ed01ebfbc9eb5...	6824	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution O...	NAME NOT FOUND	Desired Access: Query Value, Enumerate Sub Keys
10:01:...	ed01ebfbc9eb5...	6824	QuerySecurityFile	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	Information: Label
10:01:...	ed01ebfbc9eb5...	6824	QueryNameInfo...	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	Name: \Users\student\Desktop\WannaCry\@WanaDecryptor@.exe
10:01:...	svchost.exe	2200	QueryNameInfo...	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	Name: \Users\student\Desktop\WannaCry\@WanaDecryptor@.exe
10:01:...	ed01ebfbc9eb5...	6824	Process Create	C:\Users\student\Desktop\WannaCry\@WanaDecryptor@.exe	SUCCESS	PID: 5220, Command line: @WanaDecryptor@.exe
10:01:...	@WanaDecryp...	5220	Process Start		SUCCESS	Parent PID: 6824, Command line: @WanaDecryptor@.exe, Current directory: C:\Users\student\Des
10:01:...	@WanaDecryp...	5220	Thread Create		SUCCESS	Thread ID: 3492

For this last screenshot I decided to use the filter “Company is “Microsoft Corporation” then Include”. This screenshot is like the one above, except this shows WannaCry hiding itself by claiming to be a product made by the Microsoft Corporation.

Process Explorer

Process Explorer is another dynamic analysis tool that will look at processes that are currently running on the system. The difference between Process Explorer and Procman is that Procman will give us an analysis of exactly what the process is doing whereas Process Explorer is just returning an analysis CPU time, working set, and private bytes. Process Explorer does have an option to show VirusTotal results of the process which can be helpful in some scenarios.



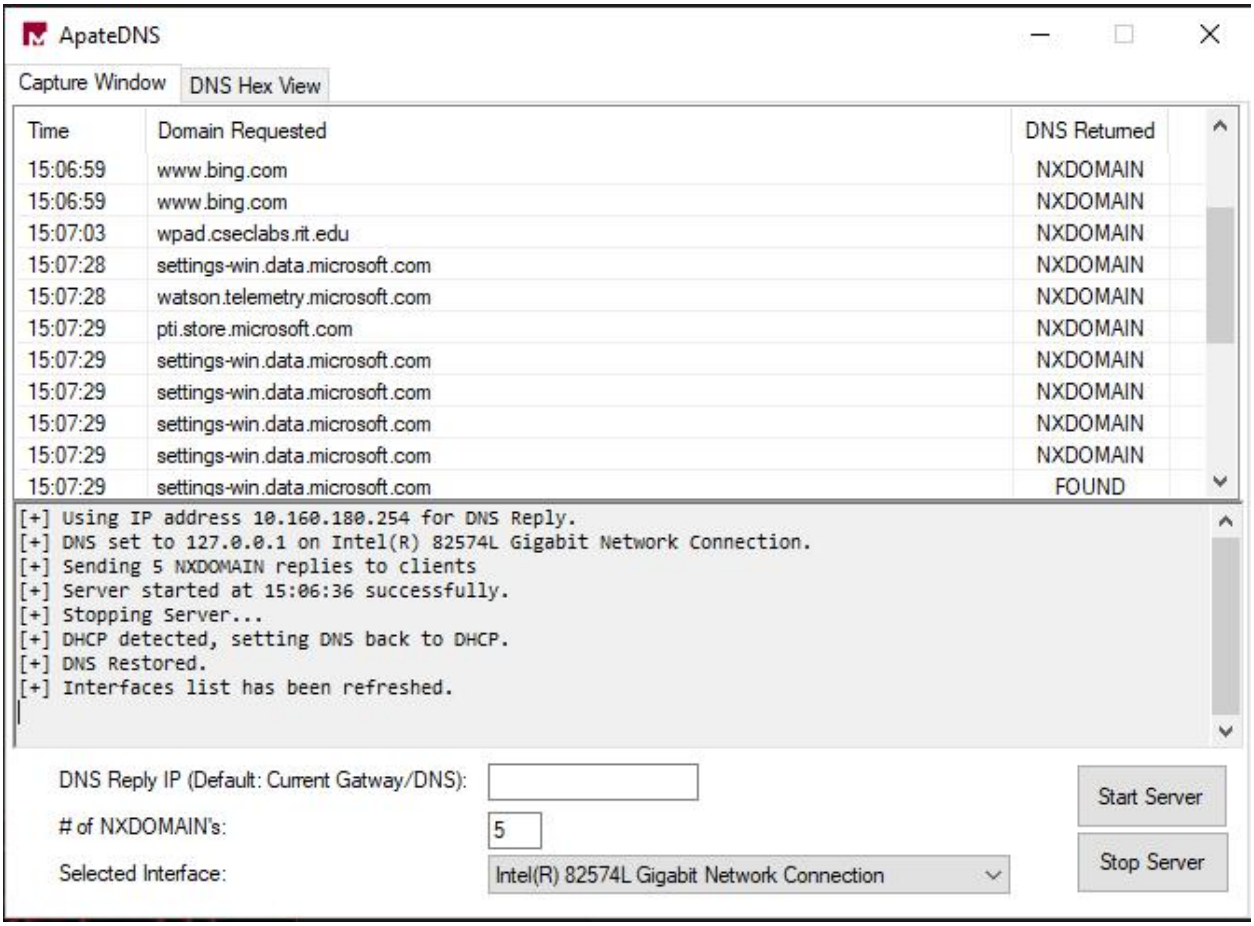
In this screenshot the executable, ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe, is running under the process 316. The executable also had spawned some child processes like @WanaDecryptor@.exe, taskhavc.exe, and conhost.exe. @WanaDecryptor@.exe is the little red message box to the right. The taskhsvc.exe and conhost.exe are unknown to me as I can only assume that conhost.exe is a command prompt icon that is executing some commands behind the scenes. The WannaCry processes also have the company name “Microsoft Corporation” again. I believe this has some sort of correlation to the <assembly> tags that I had seen previously in Resource Hacker. These however, are most likely not all of the processes WannaCry is currently running. It could be hiding some of the processes to make it harder to find / reverse, and harder for analysts to understand everything that is happening. It is also possible that WannaCry ran a few quick processes that had been executed and terminated before I was able to open and capture the screen of the processes.

ApateDNS

ApateDNS is a dynamic analysis tool used for controlling DNS response through an-easy-to use GUI. As a phony DNS server, ApateDNS spoofs DNS responses to a user-specified IP address by listening on UDP port 53 on the local machine.

15:06:48	www.bing.com	NXDOMAIN
15:06:49	www.bing.com	NXDOMAIN
15:06:49	www.bing.com	NXDOMAIN
15:06:59	www.bing.com	NXDOMAIN
15:06:59	www.bing.com	NXDOMAIN

After starting up the apateDNS server, and configuring it properly, setting the "# of NXDOMAIN's" to 5, and executing the malware I got exactly 5 rows of WannaCry trying to go out to www.bing.com. It also tries to reach out to Bing many other times throughout the course of the server's uptime.



Above is my entire screen after the three www.bing.com outreaches. Besides trying to reach out to the CSEC server, it attempts to reach out to settings-win.data.microsoft.com very frequently.

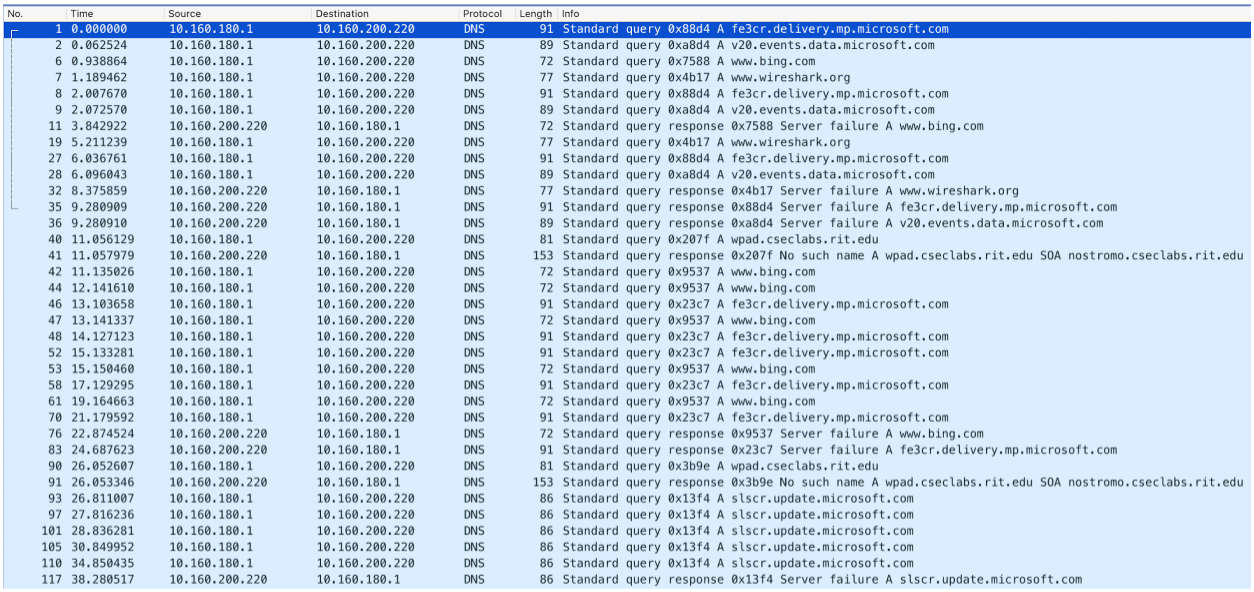
Netcat

I attempted to use Netcat but realized it was not possible. I could not use netcat for this malware because it was not using HyperText Transfer Protocol (HTTP). This could be due to the environment I am using, as the malware could have potentially detected that there was no internet connection and did not attempt to connect to any services using HTTP. Since the HTTP traffic was nonexistent, netcat was rendered useless since throughout the course we would mainly use netcat to read HTTP headers.

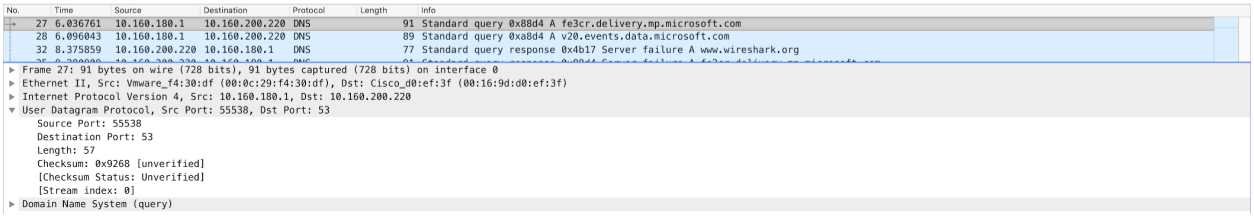
If there were to be HTTP traffic from WannaCry, netcat would have been very useful though, for looking at the HTTP header, like mentioned above. This is useful for an analyst to determine what websites a malware is trying to connect to and whether or not the request is a GET or a POST.

Wireshark

Wireshark is a packet sniffing tool that shows an analyst packets that are either entering or leaving the system. Wireshark is very useful to show an analyst what protocols the malware is using and what other servers or websites the malware might be interacting with.



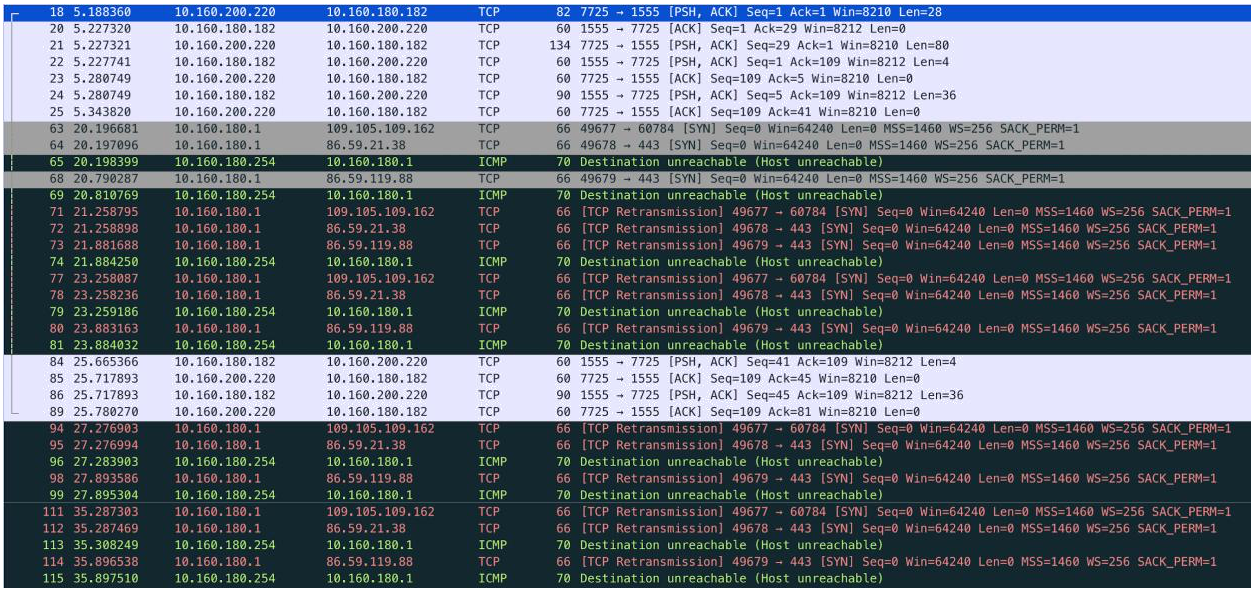
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.160.180.1	10.160.200.220	DNS	91	Standard query 0x88d4 A fe3cr.delivery.mp.microsoft.com
2	0.062524	10.160.180.1	10.160.200.220	DNS	89	Standard query 0xa8d4 A v20.events.data.microsoft.com
6	0.938864	10.160.180.1	10.160.200.220	DNS	72	Standard query 0x7588 A www.bing.com
7	1.189462	10.160.180.1	10.160.200.220	DNS	77	Standard query 0x4b17 A www.wireshark.org
8	2.007670	10.160.180.1	10.160.200.220	DNS	91	Standard query 0x88d4 A fe3cr.delivery.mp.microsoft.com
9	2.072570	10.160.180.1	10.160.200.220	DNS	89	Standard query 0xa8d4 A v20.events.data.microsoft.com
11	3.842922	10.160.200.220	10.160.180.1	DNS	72	Standard query response 0x7588 Server failure A www.bing.com
19	5.211239	10.160.180.1	10.160.200.220	DNS	77	Standard query 0x4b17 A www.wireshark.org
27	6.036761	10.160.180.1	10.160.200.220	DNS	91	Standard query 0x88d4 A fe3cr.delivery.mp.microsoft.com
28	6.096043	10.160.180.1	10.160.200.220	DNS	89	Standard query 0xa8d4 A v20.events.data.microsoft.com
32	8.375859	10.160.200.220	10.160.180.1	DNS	77	Standard query response 0x4b17 Server failure A www.wireshark.org
35	9.280909	10.160.200.220	10.160.180.1	DNS	91	Standard query response 0x88d4 Server failure A fe3cr.delivery.mp.microsoft.com
36	9.280910	10.160.200.220	10.160.180.1	DNS	89	Standard query response 0xa8d4 Server failure A v20.events.data.microsoft.com
40	11.056129	10.160.180.1	10.160.200.220	DNS	81	Standard query 0x207f A wpad.cseclabs.rit.edu
41	11.057979	10.160.200.220	10.160.180.1	DNS	153	Standard query response 0x207f No such name A wpad.cseclabs.rit.edu SOA nostromo.cseclabs.rit.edu
42	11.135026	10.160.180.1	10.160.200.220	DNS	72	Standard query 0x9537 A www.bing.com
44	12.141610	10.160.180.1	10.160.200.220	DNS	72	Standard query 0x9537 A www.bing.com
46	13.103658	10.160.180.1	10.160.200.220	DNS	91	Standard query 0x23c7 A fe3cr.delivery.mp.microsoft.com
47	13.141337	10.160.180.1	10.160.200.220	DNS	72	Standard query 0x9537 A www.bing.com
48	14.127123	10.160.180.1	10.160.200.220	DNS	91	Standard query 0x23c7 A fe3cr.delivery.mp.microsoft.com
52	15.133281	10.160.180.1	10.160.200.220	DNS	91	Standard query 0x23c7 A fe3cr.delivery.mp.microsoft.com
53	15.150460	10.160.180.1	10.160.200.220	DNS	72	Standard query 0x9537 A www.bing.com
58	17.129295	10.160.180.1	10.160.200.220	DNS	91	Standard query 0x23c7 A fe3cr.delivery.mp.microsoft.com
61	19.164663	10.160.180.1	10.160.200.220	DNS	72	Standard query 0x9537 A www.bing.com
70	21.179592	10.160.180.1	10.160.200.220	DNS	91	Standard query 0x23c7 A fe3cr.delivery.mp.microsoft.com
76	22.874524	10.160.200.220	10.160.180.1	DNS	72	Standard query response 0x9537 Server failure A www.bing.com
83	24.687623	10.160.200.220	10.160.180.1	DNS	91	Standard query response 0x23c7 Server failure A fe3cr.delivery.mp.microsoft.com
90	26.052607	10.160.180.1	10.160.200.220	DNS	81	Standard query 0x3b9e A wpad.cseclabs.rit.edu
91	26.053346	10.160.200.220	10.160.180.1	DNS	153	Standard query response 0x3b9e No such name A wpad.cseclabs.rit.edu SOA nostromo.cseclabs.rit.edu
93	26.811007	10.160.180.1	10.160.200.220	DNS	86	Standard query 0x13f4 A slscr.update.microsoft.com
97	27.816236	10.160.180.1	10.160.200.220	DNS	86	Standard query 0x13f4 A slscr.update.microsoft.com
101	28.836281	10.160.180.1	10.160.200.220	DNS	86	Standard query 0x13f4 A slscr.update.microsoft.com
105	30.849952	10.160.180.1	10.160.200.220	DNS	86	Standard query 0x13f4 A slscr.update.microsoft.com
110	34.850435	10.160.180.1	10.160.200.220	DNS	86	Standard query 0x13f4 A slscr.update.microsoft.com
117	38.280517	10.160.200.220	10.160.180.1	DNS	86	Standard query response 0x13f4 Server failure A slscr.update.microsoft.com



No.	Time	Source	Destination	Protocol	Length	Info
27	6.036761	10.160.180.1	10.160.200.220	DNS	91	Standard query 0x88d4 A fe3cr.delivery.mp.microsoft.com
28	6.096043	10.160.180.1	10.160.200.220	DNS	89	Standard query 0xa8d4 A v20.events.data.microsoft.com
32	8.375859	10.160.200.220	10.160.180.1	DNS	77	Standard query response 0x4b17 Server failure A www.wireshark.org

```
▶ Frame 27: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface 0
▶ Ethernet II, Src: Vmware_f4:30:df (00:0c:29:f4:30:df), Dst: Cisco_d0:ef:3f (00:16:9d:d0:ef:3f)
▶ Internet Protocol Version 4, Src: 10.160.180.1, Dst: 10.160.200.220
▼ User Datagram Protocol, Src Port: 55538, Dst Port: 53
    Source Port: 55538
    Destination Port: 53
    Length: 57
    Checksum: 0x9268 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
▶ Domain Name System (query)
```

This is all the DNS traffic Wireshark sniffed while WannaCry was executing. After I was not getting anything to analyze from netcat, I decided to switch over to the trusty Wireshark to see if HTTP traffic was even showing up in the first place. Evidently, there was no HTTP traffic and this led me to think that most of the traffic on the system was using UDP ports, which is what this second screenshot is showing.



18	5.188360	10.160.200.220	10.160.180.182	TCP	82	7725 → 1555 [PSH, ACK] Seq=1 Ack=1 Win=8210 Len=28
20	5.227320	10.160.180.182	10.160.200.220	TCP	60	1555 → 7725 [ACK] Seq=1 Ack=29 Win=8212 Len=0
21	5.227321	10.160.200.220	10.160.180.182	TCP	134	7725 → 1555 [PSH, ACK] Seq=29 Ack=1 Win=8210 Len=80
22	5.227741	10.160.180.182	10.160.200.220	TCP	60	1555 → 7725 [PSH, ACK] Seq=1 Ack=109 Win=8212 Len=4
23	5.280749	10.160.200.220	10.160.180.182	TCP	60	7725 → 1555 [ACK] Seq=109 Ack=5 Win=8210 Len=0
24	5.280749	10.160.180.182	10.160.200.220	TCP	90	1555 → 7725 [PSH, ACK] Seq=5 Ack=109 Win=8212 Len=36
25	5.343820	10.160.200.220	10.160.180.182	TCP	60	7725 → 1555 [ACK] Seq=109 Ack=41 Win=8210 Len=0
63	20.196681	10.160.180.1	109.105.109.162	TCP	66	49677 → 60784 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
64	20.197096	10.160.180.1	86.59.21.38	TCP	66	49678 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
65	20.198399	10.160.180.254	10.160.180.1	ICMP	70	Destination unreachable (Host unreachable)
68	20.790287	10.160.180.1	86.59.119.88	TCP	66	49679 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
69	20.810769	10.160.180.254	10.160.180.1	ICMP	70	Destination unreachable (Host unreachable)
71	21.258795	10.160.180.1	109.105.109.162	TCP	66	[TCP Retransmission] 49677 → 60784 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
72	21.258898	10.160.180.1	86.59.21.38	TCP	66	[TCP Retransmission] 49678 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
73	21.881688	10.160.180.1	86.59.119.88	TCP	66	[TCP Retransmission] 49679 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
74	21.884250	10.160.180.254	10.160.180.1	ICMP	70	Destination unreachable (Host unreachable)
77	23.258087	10.160.180.1	109.105.109.162	TCP	66	[TCP Retransmission] 49677 → 60784 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
78	23.258236	10.160.180.1	86.59.21.38	TCP	66	[TCP Retransmission] 49678 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
79	23.259186	10.160.180.254	10.160.180.1	ICMP	70	Destination unreachable (Host unreachable)
80	23.883163	10.160.180.1	86.59.119.88	TCP	66	[TCP Retransmission] 49679 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
81	23.884032	10.160.180.254	10.160.180.1	ICMP	70	Destination unreachable (Host unreachable)
84	25.665366	10.160.180.182	10.160.200.220	TCP	60	1555 → 7725 [PSH, ACK] Seq=41 Ack=109 Win=8212 Len=4
85	25.717893	10.160.200.220	10.160.180.182	TCP	60	7725 → 1555 [ACK] Seq=109 Ack=45 Win=8210 Len=0
86	25.717893	10.160.180.182	10.160.200.220	TCP	90	1555 → 7725 [PSH, ACK] Seq=45 Ack=109 Win=8212 Len=36
89	25.780270	10.160.200.220	10.160.180.182	TCP	60	7725 → 1555 [ACK] Seq=109 Ack=81 Win=8210 Len=0
94	27.276903	10.160.180.1	109.105.109.162	TCP	66	[TCP Retransmission] 49677 → 60784 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
95	27.276994	10.160.180.1	86.59.21.38	TCP	66	[TCP Retransmission] 49678 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
96	27.283903	10.160.180.254	10.160.180.1	ICMP	70	Destination unreachable (Host unreachable)
98	27.893586	10.160.180.1	86.59.119.88	TCP	66	[TCP Retransmission] 49679 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
99	27.895304	10.160.180.254	10.160.180.1	ICMP	70	Destination unreachable (Host unreachable)
111	35.287303	10.160.180.1	109.105.109.162	TCP	66	[TCP Retransmission] 49677 → 60784 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
112	35.287469	10.160.180.1	86.59.21.38	TCP	66	[TCP Retransmission] 49678 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
113	35.308249	10.160.180.254	10.160.180.1	ICMP	70	Destination unreachable (Host unreachable)
114	35.896538	10.160.180.1	86.59.119.88	TCP	66	[TCP Retransmission] 49679 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
115	35.897510	10.160.180.254	10.160.180.1	ICMP	70	Destination unreachable (Host unreachable)

This is all the TCP traffic that Wiresharked sniffed out while WannaCry was executing. There is also ICMP traffic Wireshark returned telling me that the destination was unreachable, which makes sense because of the AirGap type environment I created.

Advanced Static Analysis

Advanced static analysis is the third phase of our analysis and involves the use of a disassembler. While using the disassembler an analyst isn't actually running the executable, rather they're just looking at the assembly code. As of now IDA is still the most popular disassembler available. However on April 4, 2019, the National Security Agency (NSA) released their disassembler, Ghidra, which they developed. Ghidra might find itself becoming a very popular disassembler due to the fact that the NSA relied on it for disassembly purposes. The tool I'll be using for my advanced static analysis of WannaCry is IDA Pro, since we are familiar with it throughout the use of it in the semester.

IDA Pro

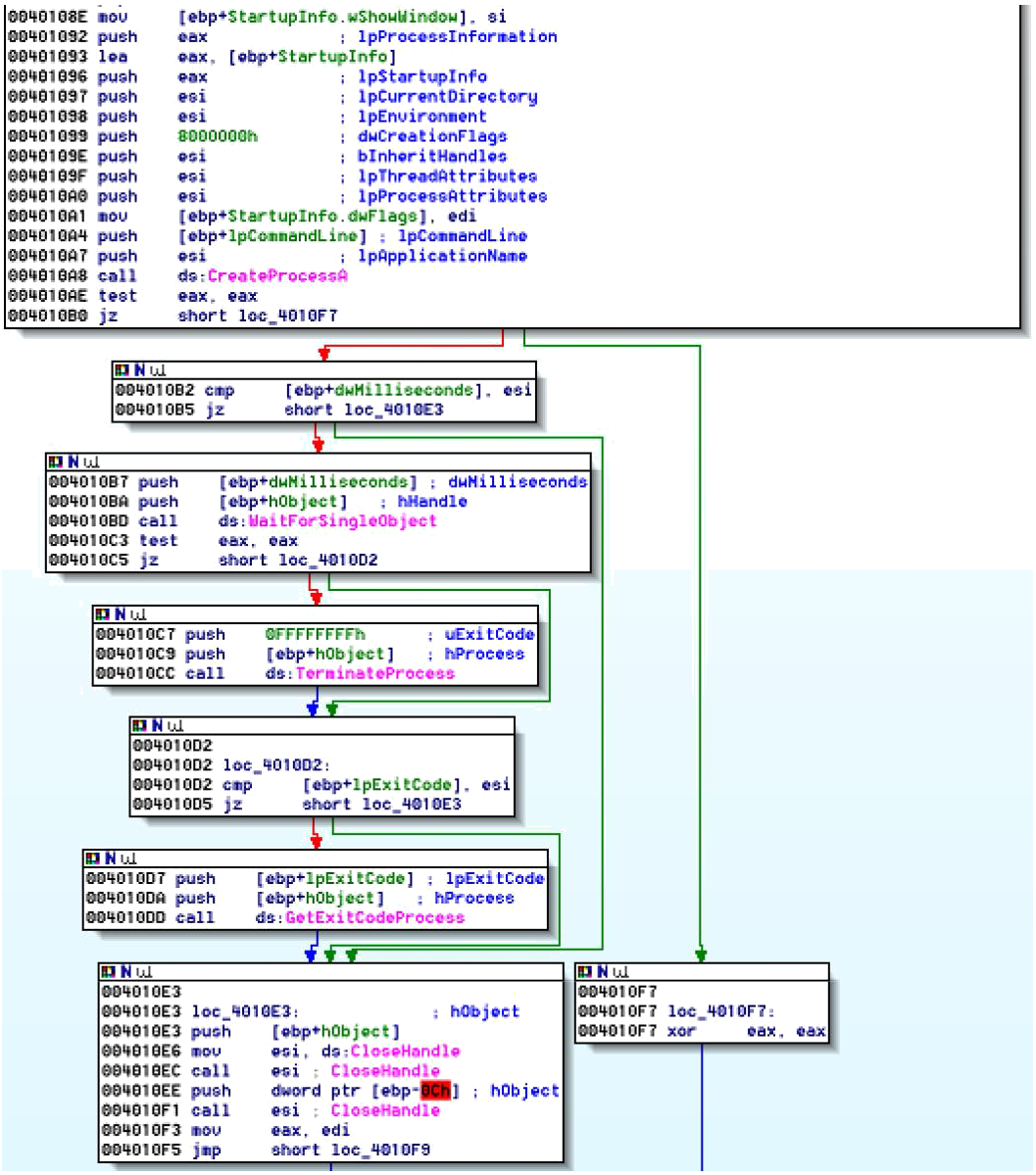
IDA Pro is a disassembler used to look at the assembly code of a malware, without running it. A disassembler is very useful when looking at things before a debugger, due to the fact that it is an advanced static tool, the malware is not running, and an analyst could find key memory locations to later view in x32dbg.

```
00000000004020C8 mov     [esp+6F4h+var_6F4], offset aWncry2ol7 ; "WNcry@2ol7"
00000000004020CF push    ebx             ; hModule
00000000004020D0 call    sub_401DAB
```

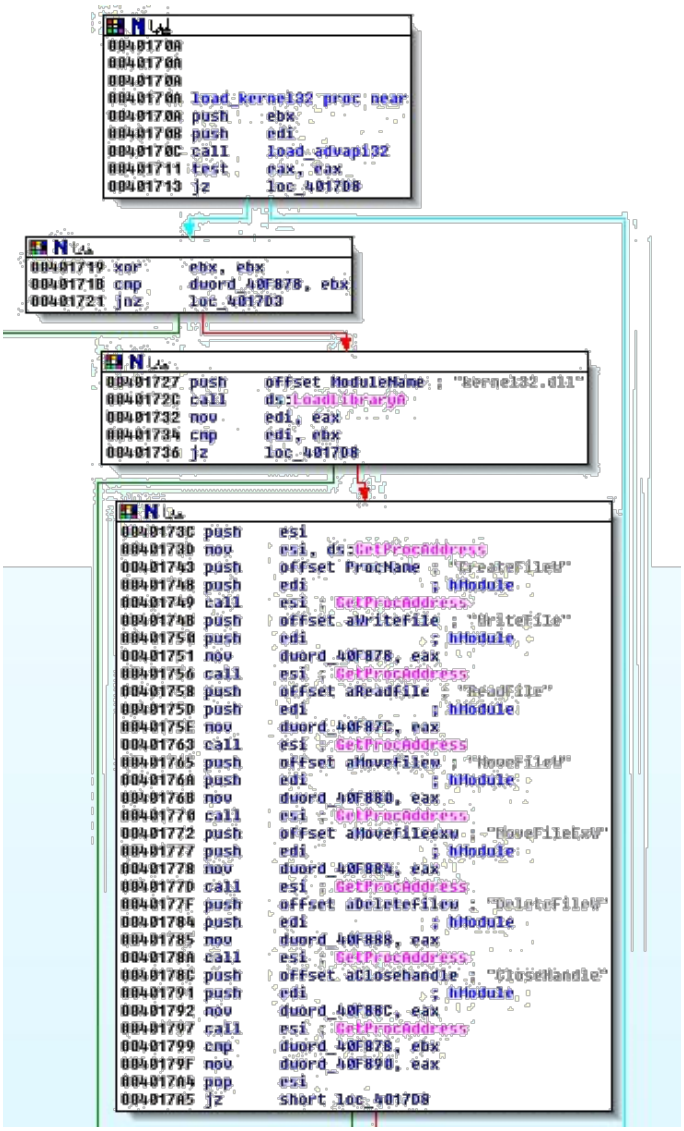
As seen in the screenshot, I found the string “WNcry@2o17” within the main method of the executable and it gets passed into a subroutine, called sub_401DAB. This subroutine opens the XIA resources and extracts something from the file. I will further analyze this later with x32dbg.

```
004020DA push    ebx             ; lpExitCode
004020DB push    ebx             ; dwMilliseconds
004020DC push    offset CommandLine ; "attrib +h ."
004020E1 call    modify_file_perms
004020E6 push    ebx             ; lpExitCode
004020E7 push    ebx             ; dwMilliseconds
004020E8 push    offset aIcacls_GrantEv ; "icacls . /grant Everyone:F /T /C /Q"
004020ED call    modify_file_perms
004020F2 add     esp, 20h
```

After further analysis, I had noticed that the sub_401064 was modifying the permissions on all of the files in the system. This explains why I changed the name of the subroutine to modify_file_perms. Referring to documentation however, attrib “displays, sets, or removes attributes assigned to files or directories”. The +h after the attrib sets the Hidden file attribute, which is again referring to documentation, icacis “displays or modifies discretionary access control lists (DACLs) on specified files, and applies stored DACLs to files in specified directories”. The F here means full control, the /T “performs the operation on all specified files in the current directory and its subdirectories”, the /C “continues the operation despite any file errors. Error messages will still be displayed”, and the /Q “suppresses success messages.” In short when the icacis command is executed with these specific parameters, it is going to apply those same permissions to all folders, subfolders, and files on the system. (MSDN Doc).



The screenshot above, represents almost the entire code execution for the modify_file_perms subroutine. This is the very subroutine that “attrib + h” and “icacis . /grant Everyone:F /T /C /Q” are passed to. This routine starts by creating a process and then proceeds to call the WaitForSingleObject function which will “wait until the specified object is in the signaled state or the time-out interval elapse.” (MSDN Doc).



This subroutine in the screenshot above is used to load in the kernel32.dll and the system calls it will be using. This includes CreateFileW, DeleteFileW, WriteFile, and others. In the first code block it is visible that there is another call to another subroutine, renamed load_advapi32, which is shown in the next screenshot.

This subroutine shown in the screenshot is used to load in the advapi32.dll and the system calls it will be using, such as CryptEncrypt, CryptDecrypt, CryptGenKey, and more.
We will now see the memory addresses we should remember to locate key subroutines/functions to observe further in x32dbg.

- Main – 401FE7
- Modify_File_Perms – 401064
- Load_kernel32 – 4020F5
- Load_advapi32 – 40170C
- Opens XIA resource and writes to c.wnry – 401DAB

MSDN Documentation icacls - https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/ica cls
MSDN Documentation attrib - https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/attr ib
MSDN Documentation WaitForSingleObject - https://docs.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-waitforsingleo bject

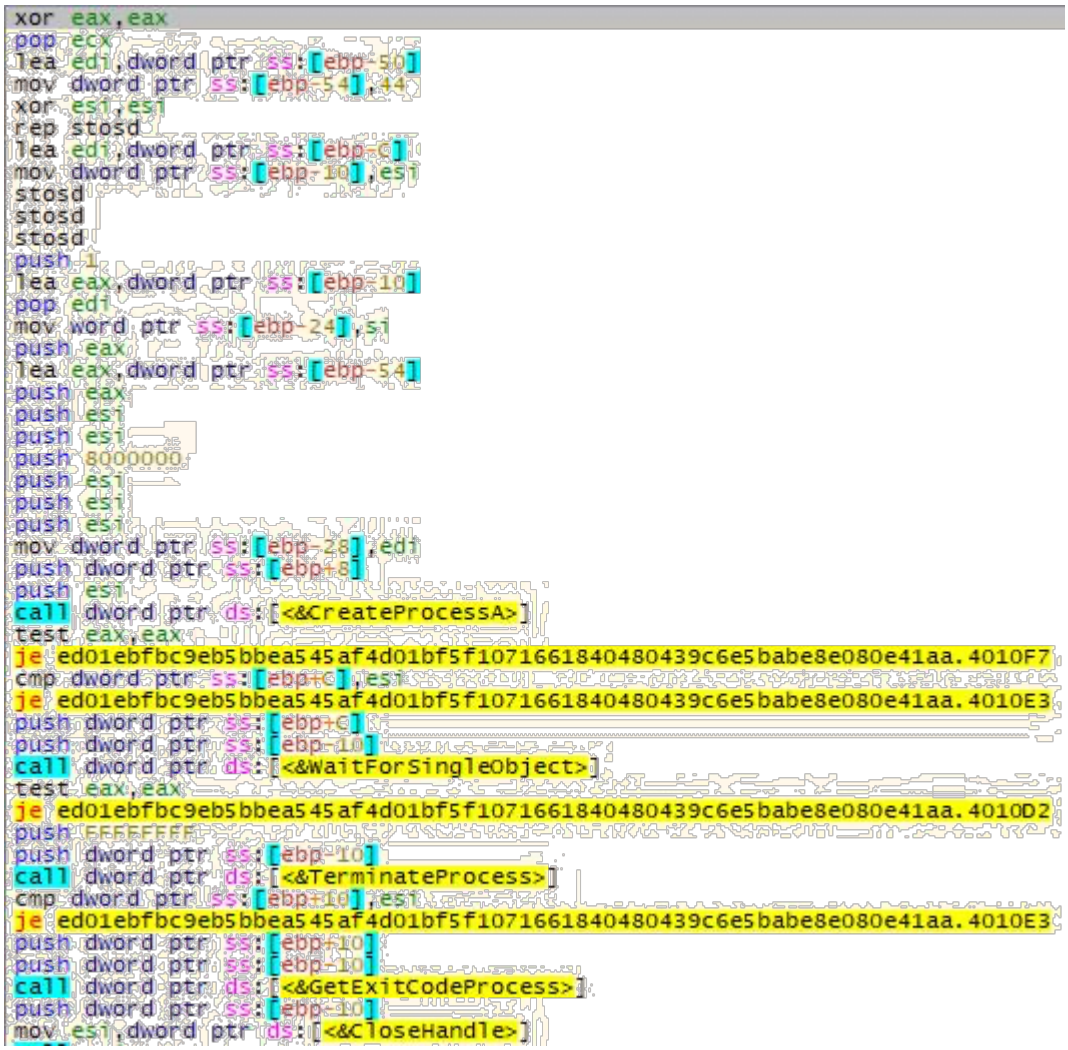
Advanced Dynamic Analysis

Advanced dynamic analysis is another phase in the analysis where an analyst is running the malware on their system. As always, it is vital that when dealing with live malware like this, to operate in a safe environment where the malware cannot affect the physical machine or other computers on the network. The tool used for the advanced dynamic analysis of WannaCry is one we've used throughout the semester, x32dbg.

x32dbg

x32dbg is a debugging tool that can help an analyst run the malware and see what happens in the backend, every step of the way via breakpoints. A debugger is useful for an analyst if they are not too sure what a certain function does specifically, allowing them to step through step by step and determine what it does.

The code section shown in the screenshot above shows the main method. Some important things that are noticeable off the bat, is the development of the /I command which could be used to build the icacis commands and the building of tasksche.exe, which is an executable that will be installed on the system.

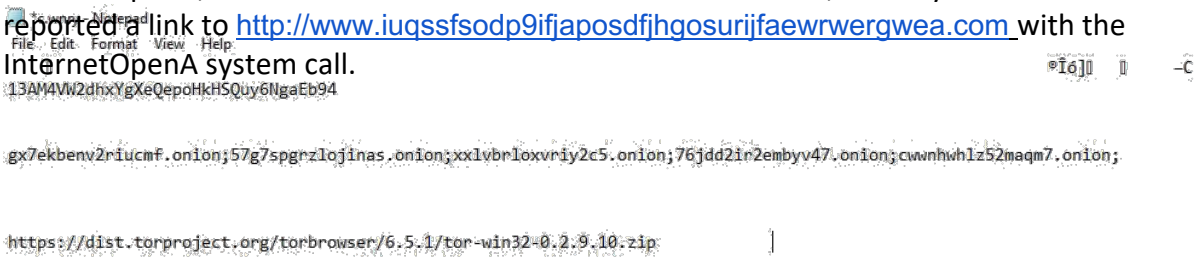


Besides the encryption methods here, I would argue that this function is very, very important to how WannaCry works. I discovered this function through IDA Pro, named modify_file_perms, and what the function actually does is that it takes the command that is passed into it, and applies that command to every file on the system. Throughout my analysis so far I have only seen “attrib +h” and “icacis . /grant Everyone:F /T /C /Q” as seen prior.

This screenshot above is the function that again loads kernel32.dll. The reason I wanted to include this screenshot in particular is because it shows how the system calls are actually loaded in, by getting the process address first through GetProcAddress. The same thing happens when the advapi32.dll is loaded in just before kernel32.dll.

This screenshot shows a function that is called to open the XIA resource, seen in PEview prior, with the password “WNcry@2o17”. The password can be seen being passed to this function in main. Referring back to the first screenshot in the IDA Pro, to see the password and the calling of this subroutine, this just confirms things. The functions procedure starts by unlocking the XIA resource and taking resources from it. After obtaining all the desired resources it wants, it locks the XIA resource and writes the resources to a c.wnry file. With some more online research, I was able to figure out that this file is a configuration file that holds important information

WannaCry refers to as overtime. This is a way to connect to the server, five .onion sites, and a download for the Tor browser. I have searched for these onion links and a system call to InternetOpenA, but I could not find a result. There are however, security firms that have reported a link to http://www.iuqssfsodp9ifjaposdfjhgosurijfaewrwergwea.com with the InternetOpenA system call.



I edited the contents of this file to be more visible as everything was jumbled up originally.

I could not find the onion websites through Ida Pro or Resource Hacker, but I did find them looking through the c.wrny file. There is also a down for the Tor browser as I said before.

Potential Dangers of WannaCry

The WannaCry ransomware is definitely a dangerous malware that you do not want on your system. It starts by looping through every file on your system, giving all users access to them, then encrypts each file with a different 128-bit AES key. FireEye/Mandiant has a report on it that I linked below, explaining more about the encryption. WannaCry encrypts mostly every file on your computer, only avoiding those files that are necessary for the computer to actually startup and run. In my specific case, I do not know if it is due to the fact that I used a virtual machine, but after running the malware my computer would be more susceptible to slowing down at times. Most antiviruses cannot detect the WannaCry malware, after doing some research, so it is crucial to always having an antivirus running on your system in case the executable finds its way onto your machine.

"The files are encrypted with a randomly generated 128-bit AES key in CBC mode with a NULL initialization vector. The key is generated per file, is encrypted with the generated RSA public key, and included in the encrypted file header. Each file encrypted by the malware starts with the string *WANACRY!* and has the *WNCRY* extension. Depending on the file properties, the malware may also stage files in a *WNCRYT* extension." - FireEye

Sources:

- FireEye's Report - https://www.fireeye.com/blog/threat-research/2017/05/wannacry-malware-profile.html

Malware Removal

After looking online, I have found mixed reviews on how to remove this malware and I am not sure as to which actually work etc. Malwarebytes, Avast, and Symantec are some companies that claim their antivirus software actually has the capability to remove the WannaCry ransomware, but I am not actually sure if this is the case and that it actually reverses the effect of the malware, or if it just gets rid of the executable. WannaCry also creates a file on the desktop telling the infected user that their AV software could potentially remove the decrypting software. If the AV software removes the executable after execution, it is not doing much for the user. It also is not good that they are not helping the user if they're debating on paying the ransom, to receive the files back, and their AV removes the decrypting software. In terms of first steps, more people need to be educated on how to secure their machiens from malware, through potential social engineering attempts like phishing emails, and how to distinguish real emails vs fake ones.

The WannaCry ransomware had also contained a worm like feature to it as said before. Reverse Engineers found that it was an unregistered domain that was helping the virus spread through different computers. The domain was then quickly registered, and WannaCry could no long spread. Although, this may sound like good news it did not matter as many different versions of WannaCry have come about, using different domains.

Final Thoughts / What I Learned

Throughout the course I have learned many methods and techniques regarding reverse engineering practices. Applying these skills and methods to this project was very fun and helped me learn a lot. I learned how to safely reverse the ransomware WannaCry through the use of all the tools mentioned above, including ones we practiced with in class. I also learned that some tools can be very useful to ones later on. Like, using the IDA Pro key memory attribute to jump to and follow in later in x32dbg. Or, using VirusTotal's report then looking for their findings in other static analysis tools or dynamic analysis tools and comparing them.