Malware Analysis Report: “Practical3.exe”

CAP6137 Malware Reverse Engineering: P0x03

Naman Arora
naman.arora@ufl.edu

April 13, 2021

Contents

1 Executive Summary

The provided binary has been identified as a member of *Ryuk* family of *Ransomwares* [6]. This family of ransomwares gain access to victim systems via phishing [8] or social engineering attacks [9] and are known to be biased towards targeting commercial systems more as compared to personal systems. *FBI* warned against this campaign citing the malware authors as most profitable when compared to other authors of such malwares. Also, this malware family is known to attack *Windows* systems in general and *Windows 10* systems in particular.

The malware binary itself is very small in size, around *170 KB*, and has no significant obfuscation techniques built in. Nevertheless, on dynamic analysis, the malware injects itself into common *Windows* processes and installs a registry key to achieve persistance over reboots. Any new file created or drive *(for eg. USB etc.)* connected to the system is also targeted and encrypted. Before any encryption happens, the malware,

- Stops and kills multiple services and executables generally associated with commercial systems
- Enumerates all drives associated to the system and all the files within them.

The execution of malware is entirely offline and hence once infected, isolating the system from the network does not thwart its any malicious activities. This also means that leaking of sensitive information from the infected system is highly improbable as a result of this infection. After encryption, the malware installs multiple text files named *RyukReadMe.txt* acknowledging the data loss, stating the ransom and providing correspondence email and *Bitcoin Wallet* addresses. This ransomware family has, however, built up a reputation of consistently decrypting the data once ransom has been paid.

Independent analysis as well as aggregated opinion from the community very strongly suggests that decrypting the data without paying the ransom amount is not possible. The cryptographic algorithms used for encryption are industry standard and hence extensively secure. The prime recommendation would be to pay the ransom unless,

- the data lost has been backed up and is tested to be recoverable to its fullest extent.
- the lost data cumulatively provides less value than ransom itself.

2 Static Analysis

2.1 Basic Identification

Attribute	Value
Bits	64
Endianess	Little
Operating System	Microsoft Windows
Class	PE32+
Subsystem	Windows GUI
Size	175616 Bytes
Compiler Timestamp	Tue Aug 14 07:46:26 2018
Compiler	Visual Studio 2015 (Likely)
SHA256 Hash	98ece6bcafa296326654db862140520afc19cfa0b4a76a5950deedb2618097ab

2.2 Malware Sample Family Identification

The malware is identified to belong to *Ryuk* family of *ransomwares* (Fig. 1). This identification is corroborated by three observations, *viz.*,

- On submitting the sample to *Virustotal* [7], a majority of the *AV* vendors identify it as such.
- Debug file *(pdb)* name within the binary has a path that mentions *Ryuk*.
- On dynamic analysis *(next section)*, it self identifies itself by installing a *RyukReadMe.txt* file in each directory.

2.3 PE Headers

Sections within the binary show no significant deviation from the norm. All the sections exhibit correct and expected permission sets (Fig. 2).

2.4 A case against Packing

A packed *PE* executable, in the context of malicious binaries, is used to obfuscate behavior in hopes of evading static analysis and *AV* detection. The binary sample, though, attempts no such evasion. It is very unlikely that the given sample is packed or encrypted in any way given the following observations,

- The sheer quantity of imports gathered from auto-analysis within *Ghidra*, some of which are generally associated with malicious behavior (Fig 4, 3 and 5)
- Multiple strings indicating malicious behavior

2.5 Interesting Imports

Following are the interesting *DLLS* that are imported,

Advapi32.dll

The malware sample imports functions like *AdjustTokenPrivileges, GetTokenInformation, OpenProcessToken etc.* indicate some token privilege escalation (Fig. 4). Such a behavior indicate the possibility of process injection, which on further analysis, pans out.

Kernel32.dll

The functions imported from this library can be further subdivided in terms of their intended use as,

- Functions like *OpenProcess, VirtualAllocEx, VirtualFreeEx, CreateRemoteThread and WriteProcessMemory* indicate a strong possibility, in conjunction with above *token* related imports, of process injection.
- Functions like *FindFirstFileExA, FindNextFileA and SetFilePointerEx* indicate towards filesystem enumeration.
- Functions like *GetProcAddress and LoadLibraryA* indicate dynamic resolution of function references which, in conjunction with strong traits of process injection, give a bigger picture.

Refer to figure 3.

Shell32.dll

The functions imported from this library *ShellExecuteA and ShellExecuteW*, on further analysis, reveal to be used in stopping system processes and executables (Fig 5).

2.6 Interesting Code Constructs

The Stopping of System Services

Before any significant exhibition of malicious behavior, the malware sample uses the shell execution commands to stop a list of predefined processes and executables. This list is contained within the binary and can be referenced here [4] and [2]. It uses *"taskkill /IM $taskName /F"* and *"net stop $serviceName /y"* template commands to achieve this. All the tasks and executables are generally associated with commercial systems which further strengthens the argument that this malware targets businesses rather than private individuals. The function responsible for this execution is at offset ***7ff686d11250***.

Process Injection

The malware sample, after stopping common services and processes, injects itself into virtual memory section of multiple *Windows* services. Primary function that implements this injection receives a list of arguments including the process name of the victim process and its *PID*. This function then calculates its own size and allocates memory within the victim process and writes itself to that memory area. Following that, it calls *CreateRemoteThread* to execute a function within its own memory area located at offset ***7ff686d119b0***. The function responsible for all this is at offset ***7ff686d126f0***.

Dynamic Function Address Resolution

Once the remote process injection is successful and the remote thread has begun the routine which the malware sample requested, a function located at offset ***7ff686d16850*** is then called. This function is responsible for dynamically resolving addresses to a series of functions that, down the line, rest of the malware uses. Some function that are resolved by the said function are absent from the list of imports statically available for investigation (talked about in the dynamic analysis section).

3 Dynamic Analysis

3.1 Interesting Features

The following interesting features were observed on dynamically analyzing the malware sample,

- As expected and previously mentioned in *static analysis* section, the malware issues multiple shell commands to stop processes and executables mentioned in [2] and [4].
- The malware injects itself into important *Windows* services like *sihost.exe, crss.exe, lsass.exe, etc.* and initiates the function at offset ***7ff686d119b0***.
- The *injection-victim* process then executes the function at offset ***7ff686d16850***. This function is, apparently, responsible for resolving multiple import addresses. Some new and previously unknown imports are revealed, most notably,
  - *CryptAcquireContext, CryptImportKey, CryptExportKey, CryptGenKey and CryptEncrypt* which are then later used for the actual cryptographic encryption of the data stored on the victim system.
  - *GetLogicalDrives and GetDriveType* which, within a loop, indicates that all the drives connected to the system, physical and virtual, are enumerated for the purpose of encrypting.

  Also, interestingly, a minority yet non-trivial quantity of imports within the resolution process are never stored anywhere and are never used either. This might indicate use of a scripted routine to actually write high level code within the malware. With that assumption at hand, a *YARA* rule can leverage such a script's repeated usage for malware author's attribution/connection with other malware samples.
- A dedicated subroutine at the offset ***7ff686d14710*** receives the file name to encrypt which then the function reads in, encrypts and writes back out.
- The *"RyukReadMe.txt"* seems to be stored within the malware binary encoded, most likely, in *Base64* encoding. This, though, could not be firmly established in this analysis. The suspected functions for this trait seem to reside at offsets ***7ff686d12890 and 7ff686d112f0***.
- Multiple references to *CryptAcquireContext* seem to suggest the following,
  - There are two types of *Cryptographic Services Providers (CSPs)* spotted as arguments to calls to this function, *viz. AES and RSA*.
  - The *RSA* key instance seems to be embedded as a *Keyblob* type within the binary's data section at offset ***7ff686d34a10***.

  The above observations lead to an educated guess that the imported key is a *RSA Public key* and another *AES* session key is generated while execution is taking place. The process of encryption then would look like,
  - Import the *RSA public key*
  - Generate an *AES* session key
  - Encrypt files with the session key
  - Generate unique ID which is directly linked with the session key and encrypt that ID with the *RSA* public key and store it on the disk.

  The above sequence could not be verified in its entirety but serves as an educated guess.
- The malware sample never encrypts any executable or *DLL* file.

3.2 File System Interaction

The malware sample interacts with the filesystem in the following ways,

- It writes a file *"C:\Users\Public\UNIQUE_ID_DO_NOT_REMOVE"*.
- Writes a file *"RyukReadme.txt"* [3] at each node returned by *FindNextFileA* function.

- Writes a batch script *"C:\Users\Public\window.bat"* [1] which is deleted after execution
- Encrypts all data files
- Gaines persistance over reboots of system by adding itself to the *RegistryKey "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run "*. Although consistent, this cannot be an considered an indicator of compromise since the value of the executable name might vary.

3.3 Network Interaction

The malware sample exhibited no attempt to access network resources during this analysis.

4 Indicators of Compromise

4.1 Host Based

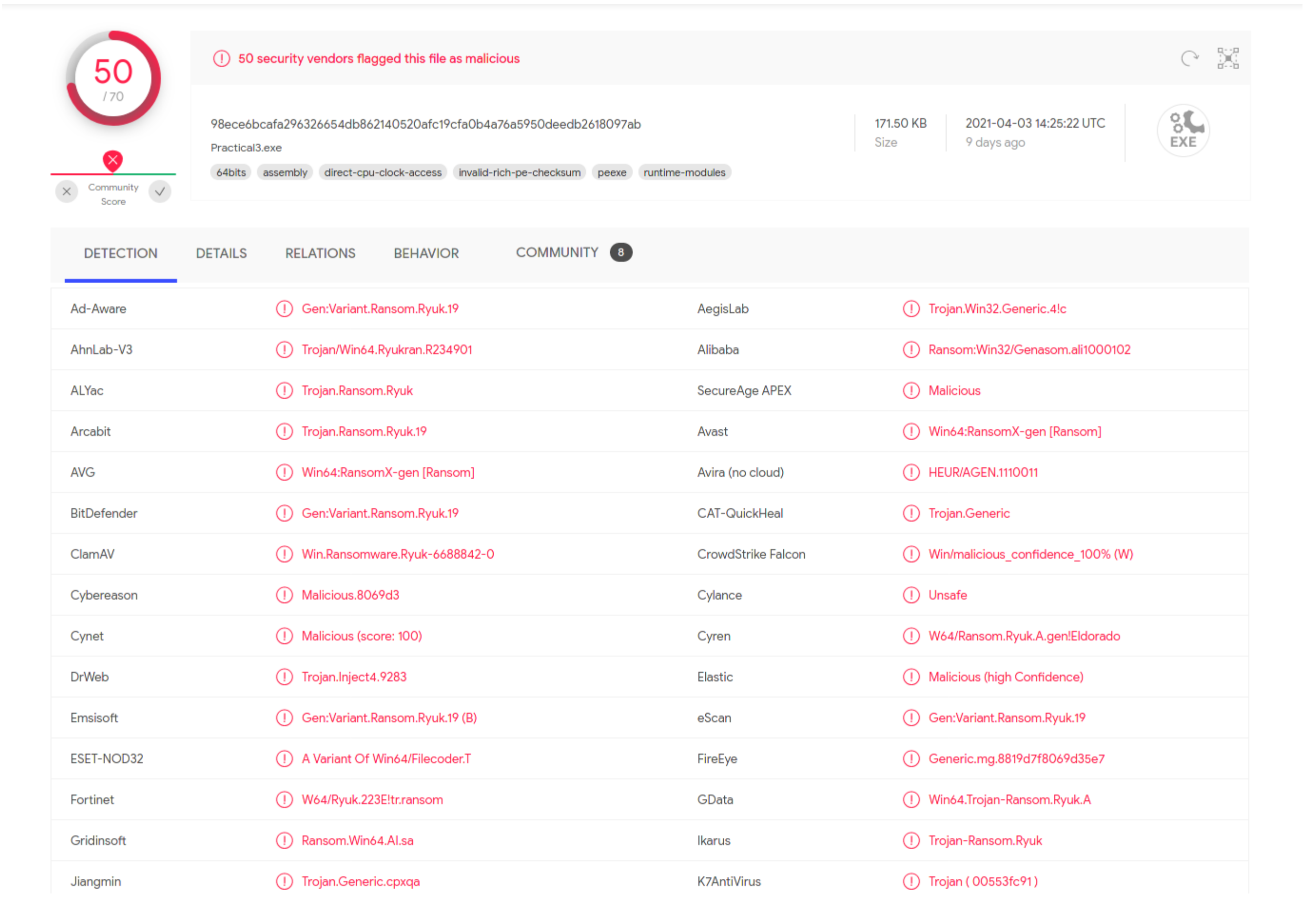
- Presence of the file *"C:\Users\Public\UNIQUE_ID_DO_NOT_REMOVE"*
- Presence of the file *"RyukReadme.txt"*

4.2 *YARA* Rule

In case of unsuccessful copy from text below, use [5].

```
rule practical3_Ryuk {
        meta:
                description = "Detect Practical3.exe Ryuk Ransomware"
                author = "Naman Arora"
                date = "2021-04-13"
                hash = "98ece6bcafa296326654db862140520afc19cfa0b4a76a5950deedb2618097ab"
        strings:
                $pdb = "C:\\Users\\Admin\\Documents\\Visual Studio 2015\\Projects From Ryuk\\ConsoleApplication54\\x64\\Release\\ConsoleApplication54.pdb" fullword ascii
        condition:
                uint16(0) == 0x5a4d and filesize < 180KB and $pdb
}
```

5 Appendix A: Screenshots


50
/70

Community Score

50 security vendors flagged this file as malicious

98ece6bcafa296326654db862140520afc19cfa0b4a76a5950deedb2618097ab
Practical3.exe

64bits assembly direct-cpu-clock-access invalid-rich-pe-checksum peexe runtime-modules

171.50 KB
Size

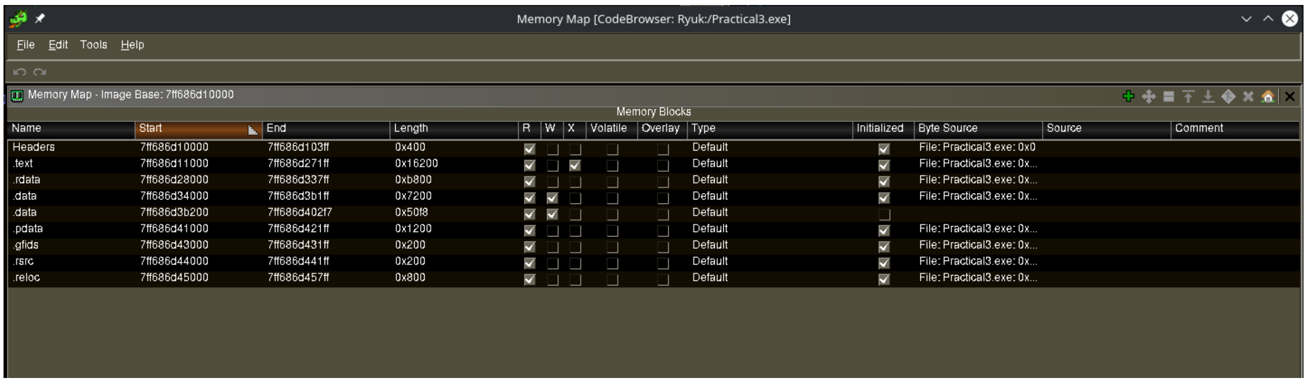
2021-04-03 14:25:22 UTC
9 days ago

EXE

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 8

Ad-Aware	Gen:Variant.Ransom.Ryuk.19	AegisLab	Trojan.Win32.Generic.4!c
AhnLab-V3	Trojan/Win64.Ryukran.R234901	Alibaba	Ransom:Win32/Genasom.ali1000102
ALYac	Trojan.Ransom.Ryuk	SecureAge APEX	Malicious
Arcabit	Trojan.Ransom.Ryuk.19	Avast	Win64:RansomX-gen [Ransom]
AVG	Win64:RansomX-gen [Ransom]	Avira (no cloud)	HEUR/AGEN.1110011
BitDefender	Gen:Variant.Ransom.Ryuk.19	CAT-QuickHeal	Trojan.Generic
ClamAV	Win.Ransomware.Ryuk-6688842-0	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cybereason	Malicious.8069d3	Cylance	Unsafe
Cynet	Malicious (score: 100)	Cyren	W64/Ransom.Ryuk.A.gen!Eldorado
DrWeb	Trojan.Inject4.9283	Elastic	Malicious (high Confidence)
Emsisoft	Gen:Variant.Ransom.Ryuk.19 (B)	eScan	Gen:Variant.Ransom.Ryuk.19
ESET-NOD32	A Variant Of Win64/Filecoder.T	FireEye	Generic.mg.8819d7f8069d35e7
Fortinet	W64/Ryuk.223E!tr.ransom	GData	Win64.Trojan-Ransom.Ryuk.A
Gridinsoft	Ransom.Win64.AI.sa	Ikarus	Trojan-Ransom.Ryuk
Jiangmin	Trojan.Generic.cpxqa	K7AntiVirus	Trojan (00553fc91)

Figure 1: VirusTotal: VirusTotal Scan


Memory Map [CodeBrowser: Ryuk:/Practical3.exe]

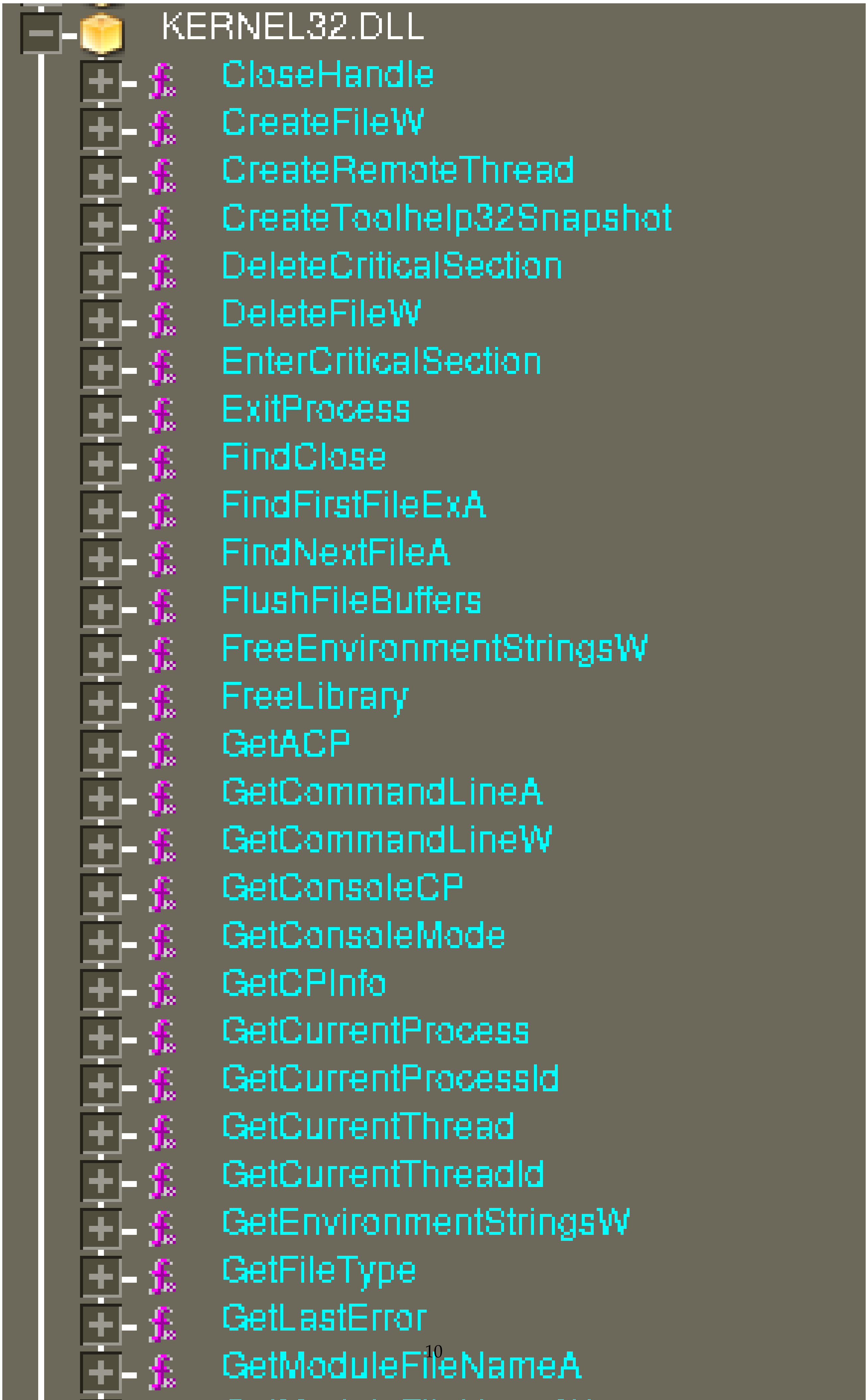
File Edit Tools Help

Memory Map - Image Base: 7ff686d10000

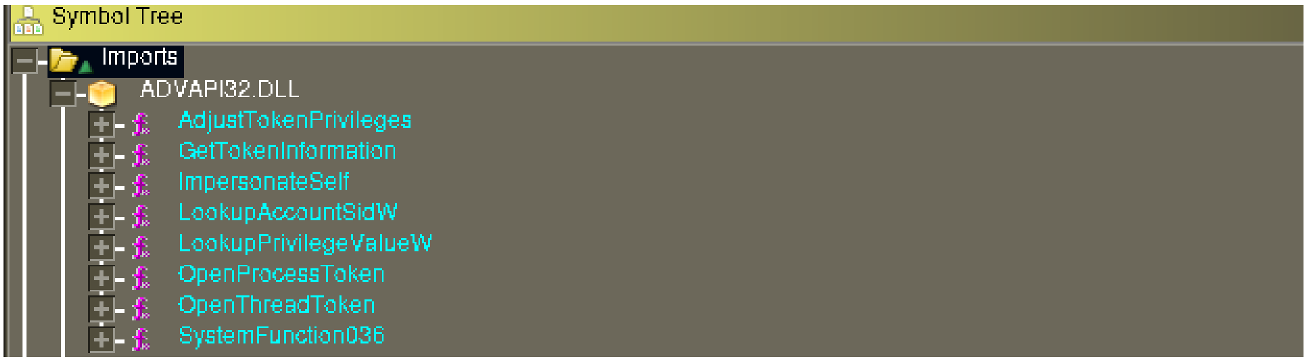
Memory Blocks

Name	Start	End	Length	R	W	X	Volatile	Overlay	Type	Initialized	Byte Source	Source	Comment
Headers	7ff686d10000	7ff686d103ff	0x400	✓					Default	✓	File: Practical3.exe: 0x0		
.text	7ff686d11000	7ff686d271ff	0x16200	✓		✓			Default	✓	File: Practical3.exe: 0x...		
.rdata	7ff686d28000	7ff686d337ff	0xb800	✓					Default	✓	File: Practical3.exe: 0x...		
.data	7ff686d34000	7ff686d3b1ff	0x7200	✓	✓				Default	✓	File: Practical3.exe: 0x...		
.data	7ff686d3b200	7ff686d402f7	0x50f8	✓	✓				Default				
.pdata	7ff686d41000	7ff686d421ff	0x1200	✓					Default	✓	File: Practical3.exe: 0x...		
.gfids	7ff686d43000	7ff686d431ff	0x200	✓					Default	✓	File: Practical3.exe: 0x...		
.rsrc	7ff686d44000	7ff686d441ff	0x200	✓					Default	✓	File: Practical3.exe: 0x...		
.reloc	7ff686d45000	7ff686d457ff	0x800	✓					Default	✓	File: Practical3.exe: 0x...		

Figure 2: Ghidra: *PE* headers

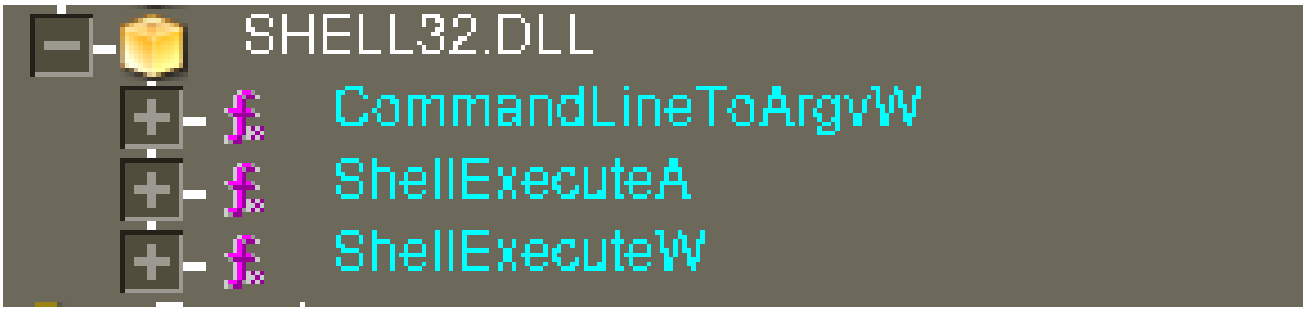


KERNEL32.DLL
CloseHandle
CreateFileW
CreateRemoteThread
CreateToolhelp32Snapshot
DeleteCriticalSection
DeleteFileW
EnterCriticalSection
ExitProcess
FindClose
FindFirstFileExA
FindNextFileA
FlushFileBuffers
FreeEnvironmentStringsW
FreeLibrary
GetACP
GetCommandLineA
GetCommandLineW
GetConsoleCP
GetConsoleMode
GetCPInfo
GetCurrentProcess
GetCurrentProcessId
GetCurrentThread
GetCurrentThreadId
GetEnvironmentStringsW
GetFileType
GetLastError
GetModuleFileNameA



```
Symbol Tree
Imports
  ADVAPI32.DLL
    AdjustTokenPrivileges
    GetTokenInformation
    ImpersonateSelf
    LookupAccountSidW
    LookupPrivilegeValueW
    OpenProcessToken
    OpenThreadToken
    SystemFunction036
```

Figure 4: Ghidra: *Advapi32.dll* Imports



```
SHELL32.DLL
  CommandLineToArgvW
  ShellExecuteA
  ShellExecuteW
```

Figure 5: Ghidra: *Shell32.dll* Imports

References

[1] Naman Arora. *Ryuk Anti-Backup Script*. `https://gist.github.com/r0ck3r008/29be0fbe8cdb0b51fa884340` [Online; accessed 13-April-2021].

[2] Naman Arora. *Ryuk Processes Stopped*. `https://gist.github.com/r0ck3r008/1271cd18978a5419b99e19b19` [Online; accessed 13-April-2021].

[3] Naman Arora. *Ryuk ReadMe.txt*. `https://gist.github.com/r0ck3r008/ab3016169b42e331b9bfe25163348` [Online; accessed 13-April-2021].

[4] Naman Arora. *Ryuk Services Stopped*. `https://gist.github.com/r0ck3r008/3f9196e4810174011db41708c0` [Online; accessed 13-April-2021].

[5] Naman Arora. *Ryuk YARA Rule*. `https://gist.github.com/r0ck3r008/64343f9ebe4dadeef8f5ad05ebc5b1` [Online; accessed 13-April-2021].

[6] Federal Bureau of Investigation. *Ransomware*. `https://www.fbi.gov/scams-and-safety/common-scams-and-crimes/ransomware`. [Online; accessed 11-April-2021].

[7] VirusTotal. *Ryuk Family Identification*. `https://www.virustotal.com/gui/file/98ece6bcafa296326654db86 detection`. [Online; accessed 12-April-2021].

[8] WikiMedia. *Phishing*. `https://en.wikipedia.org/wiki/Phishing`. [Online; accessed 11-April-2021].

[9] WikiMedia. *Social Engineering*. `https://en.wikipedia.org/wiki/Social_engineering_security`. [Online; accessed 11-April-2021].