Deep Dive Into Ryuk Ransomware

By astro

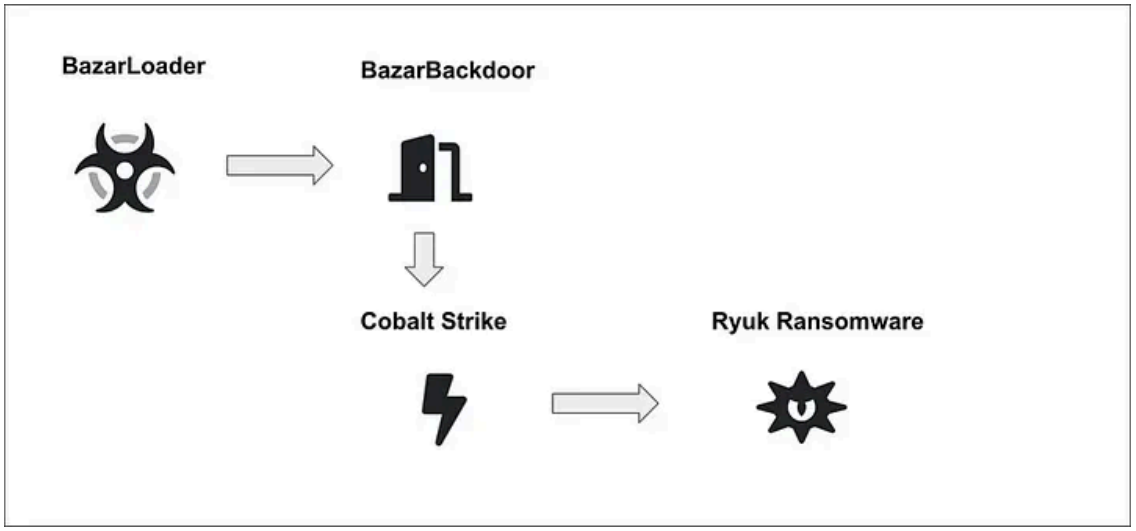
Published: 2020-11-18 · Archived: 2026-04-05 18:57:05 UTC

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 arrtributed to an APT Group Called Lazarus Group.

WANTED BY THE FBI

PARK JIN HYOK

Conspiracy to Commit Wire Fraud; Conspiracy to Commit Computer-Related Fraud (Computer Intrusion)

DESCRIPTION

Aliases: Pak Jin Hek, Jin Hyok Park	
Place of Birth: Democratic People's Republic of Korea (North Korea)	**Hair:** Black
Eyes: Brown	**Sex:** Male
Race: Asian	**Languages:** English, Korean

REMARKS

Park attended the Kim Chaek University of Technology in Pyongyang, North Korea. He is a North Korean citizen last known to be in North Korea. Park has traveled to China in the past and conducted legitimate IT work under the front company "Chosun Expo" or the Korean Expo Joint Venture in addition to activities conducted on behalf of North Korea's Reconnaissance General Bureau.

CAUTION

Park Jin Hyok is allegedly a North Korean computer programmer who is part of a state-sponsored hacking organization responsible for some of the costliest computer intrusions in history, including the cyber attack on Sony Pictures Entertainment, a series of attacks targeting banks across the world that collectively attempted to steal more than one billion dollars, and the WannaCry ransomware attack that affected tens of thousands of computer systems across the globe.

Park was alleged to be a participant in a wide-ranging criminal conspiracy undertaken by a group of hackers employed by a company that was operated by the North Korean government. The front company - Chosun Expo Joint Venture, also known as Korea Expo Joint Venture - was affiliated with Lab 110, one of the North Korean government's hacking organizations. That hacking group is what some private cybersecurity researchers have labeled the "Lazarus Group." On June 8, 2018, a federal arrest warrant was issued for Park Jin Hyok in the United States District Court, Central District of California, after he was charged with one count of conspiracy to commit wire fraud and one count of conspiracy to commit computer-related fraud (computer intrusion).

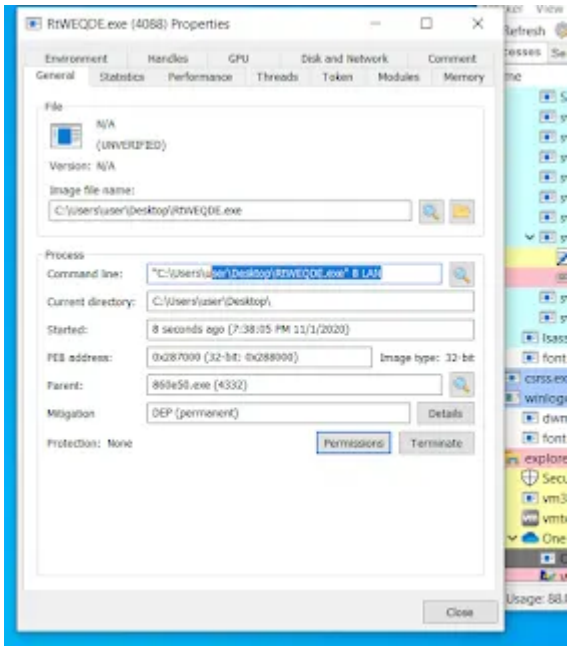
If you have any information concerning this person, please contact your local FBI office or the nearest American Embassy or Consulate.

Field Office: Los Angeles

In Depth Reversing:

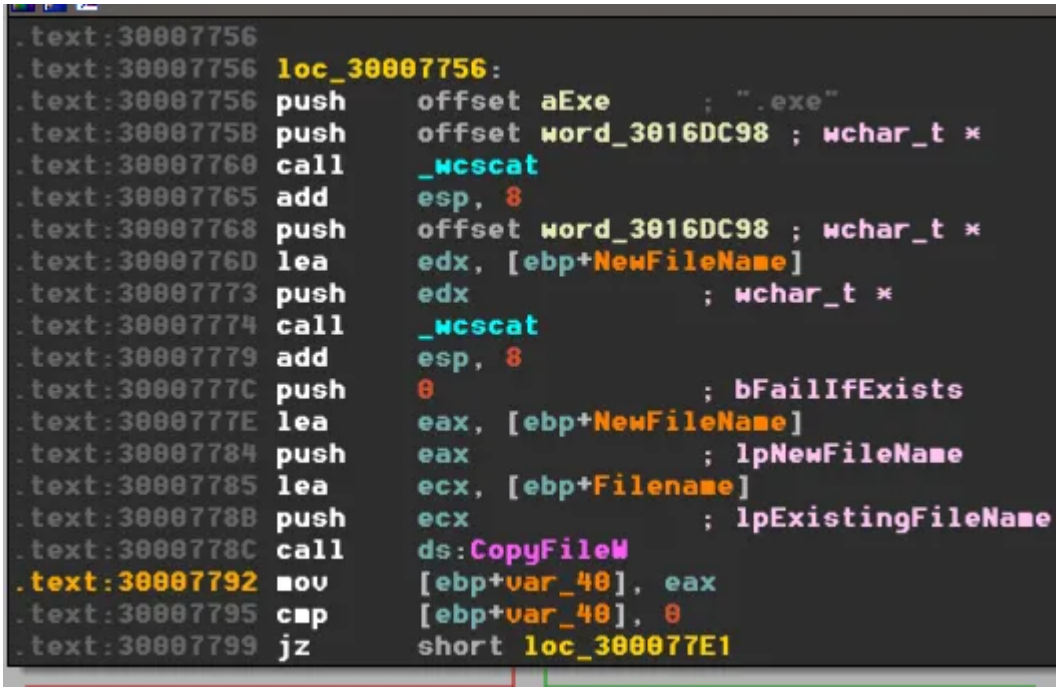
- I Used a Combination of Cutter, IDA and x64 dbg to reverse this malware so nvm xD
- When Executing the Sample It Drops a Copy From its Self and Execute it using “8 lan” Command.

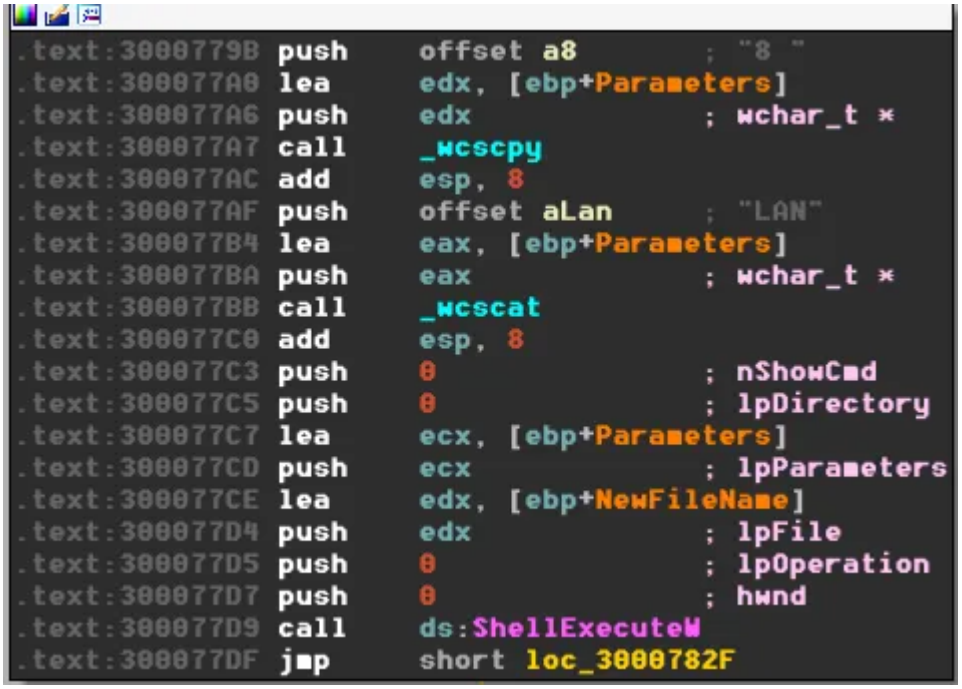
Press enter or click to view image in full size



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 feauture 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.

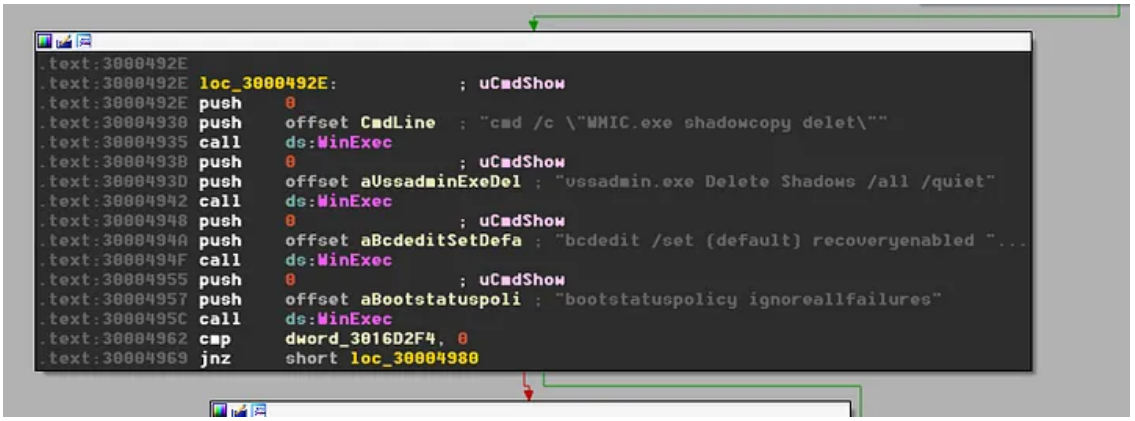




- The Name of the Dropped Exe are Seven Random Characters.
- The Malware Injects Into 4 Process taskeng.exe, host.exe, dwm.exe, ctfmon.exe
- It Encrypts the Files using “RYK” Extension
- The Malware Deletes Shadow Copies Using:

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

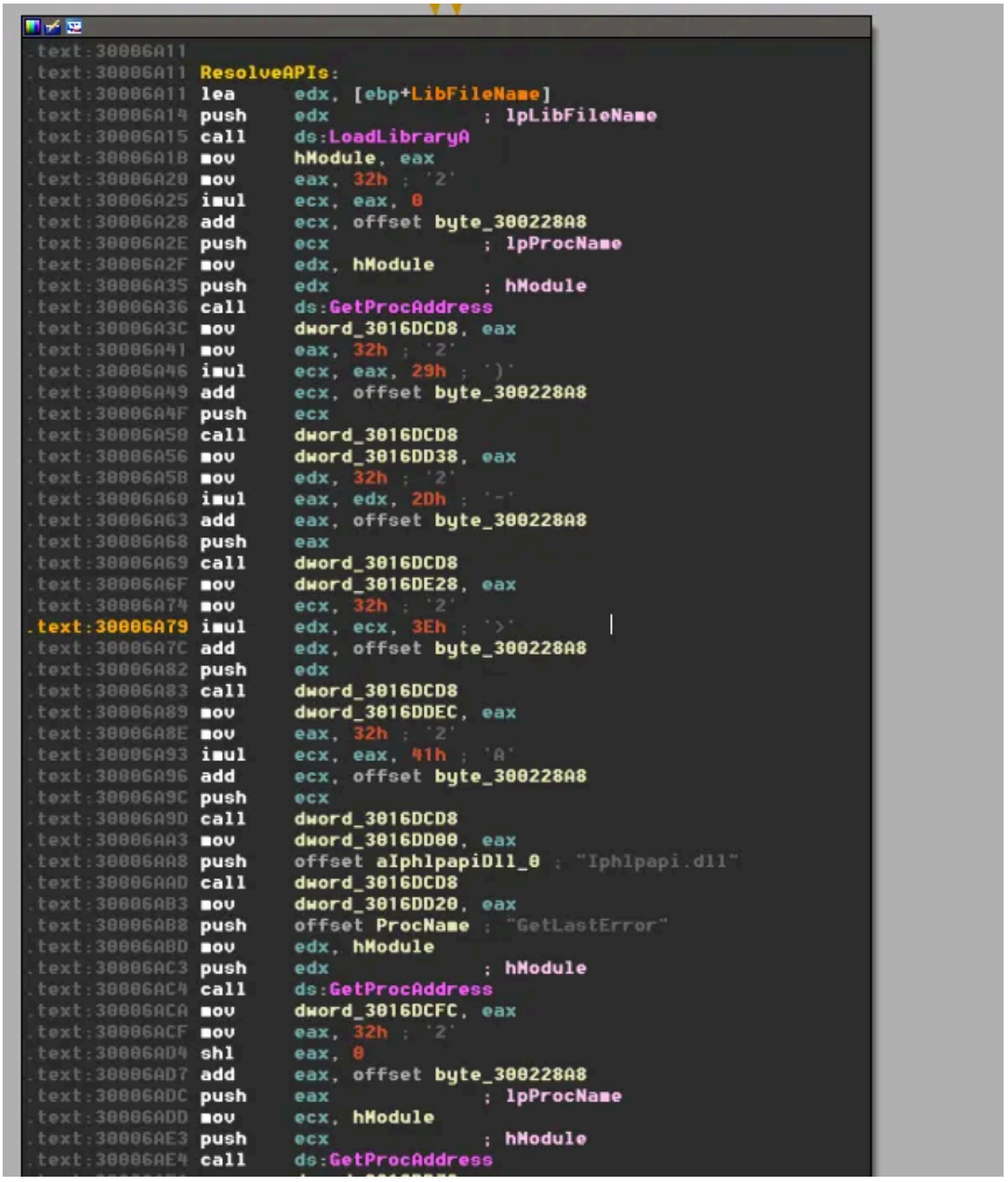
Press enter or click to view image in full size



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

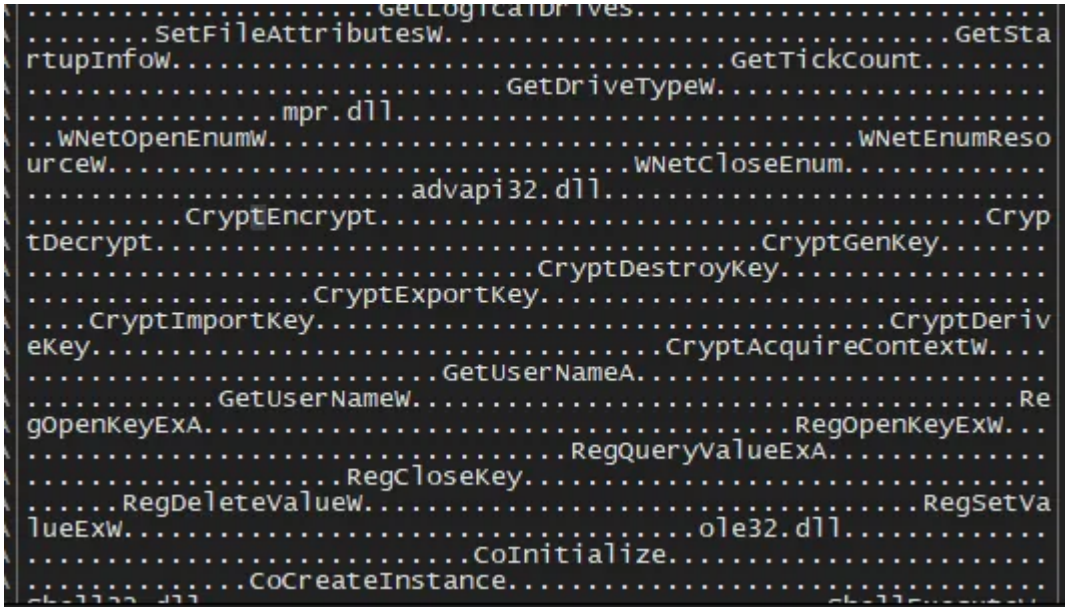
API Resolving:

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_30022BA8
.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 ; ')'
.text:30006A49 add     ecx, offset byte_30022BA8
.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, 2Dh ; '-'
.text:30006A63 add     eax, offset byte_30022BA8
.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_30022BA8
.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_30022BA8
.text:30006A9C push    ecx
.text:30006A9D call    dword_3016DCD8
.text:30006AA3 mov     dword_3016DD00, eax
.text:30006AA8 push    offset aIphlpapiDll_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_30022BA8
.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:



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:

Privilege Escalation:

Ryuk Escalates Privilege by Modifying the Access Token

Press enter or click to view image in full size

```
undefined4 __cdecl PrivliageEscalation(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 PrivliageEscalation
add esp, 0xc
push 0xba090
push 0                          ; int32_t arg_ch
lea ecx, [var_bb3e4h]
push ecx                        ; int32_t arg_8h
call fcn.300074b0
add esp, 0xc
lea edx, [var_bb3e4h]
push edx                        ; int32_t arg_8h
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, LPWST...
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:

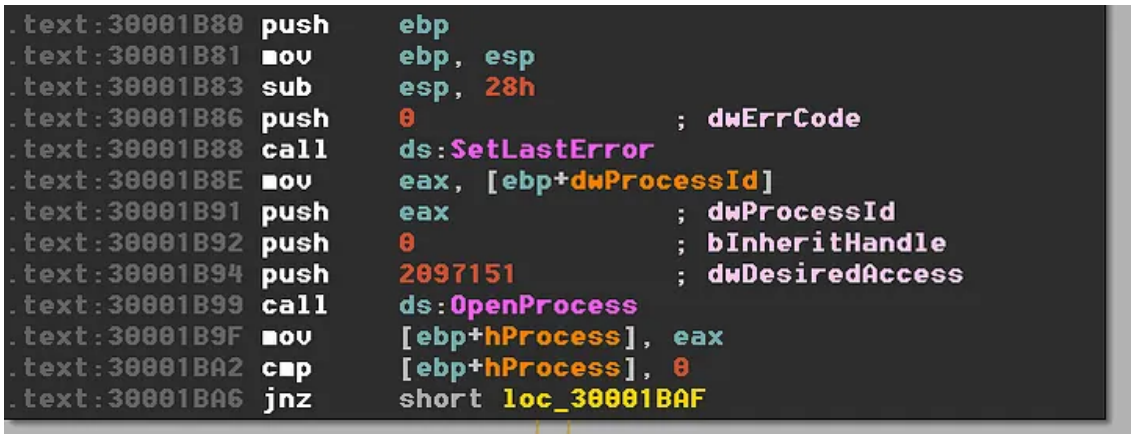
Ryuk Acheives Persistence by Adding the Path of the malware under /Run Key In the Registry Makes it Run Every Time the User logs In It uses this Command

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

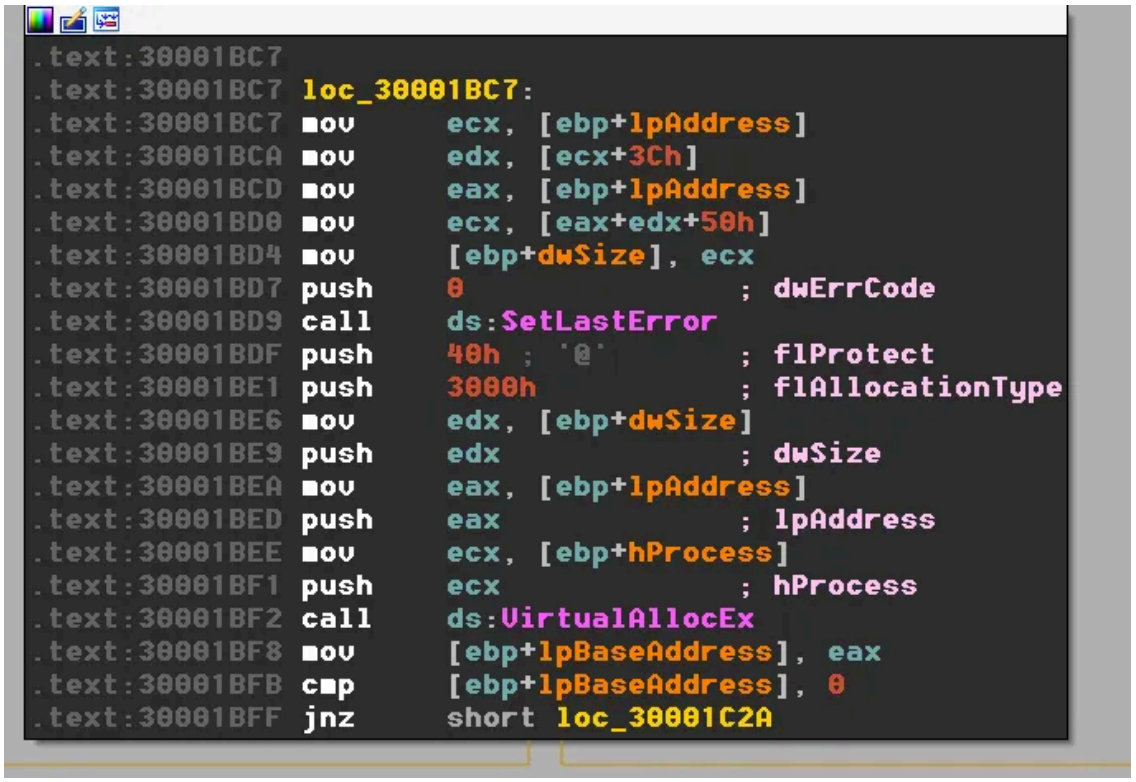
Process Injection:

First It Opens a Process

Press enter or click to view image in full size

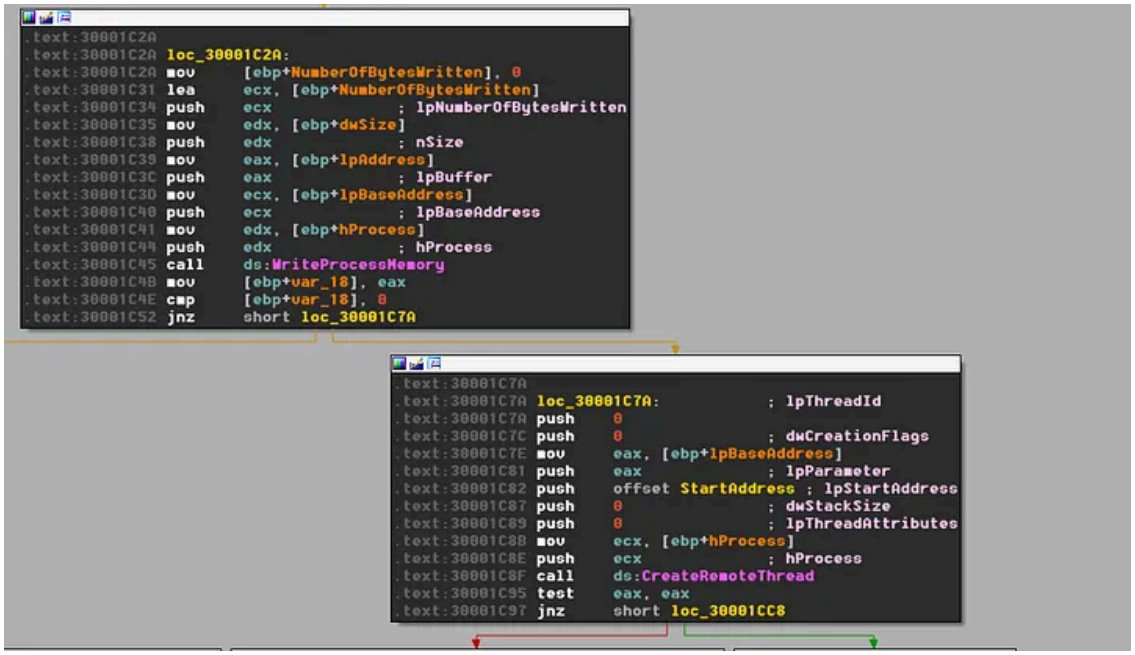


Then It Allocates Memory in the Target Process

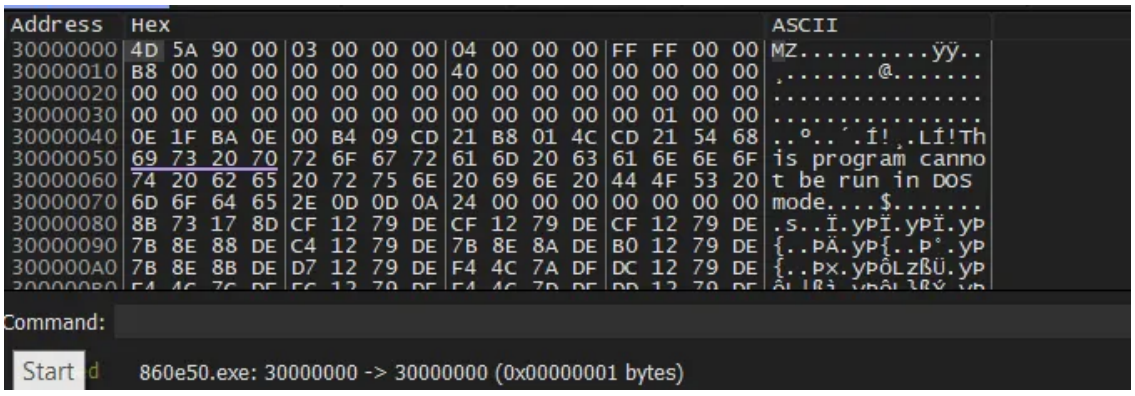


Next it Writes injects its Self using WriteProcessMemory and Creates a Thread to run the injected code

Press enter or click to view image in full size



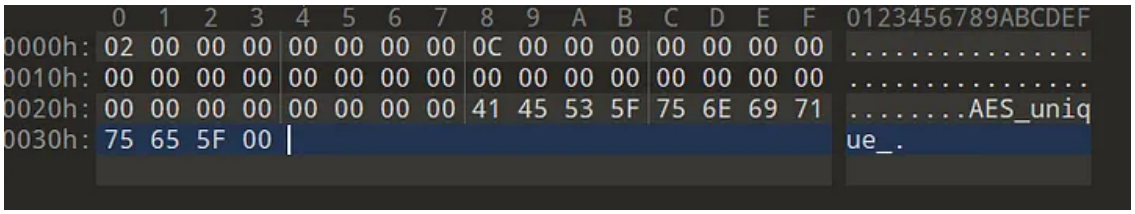
Following with the debugger we can see the exectuable in the memory dump



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

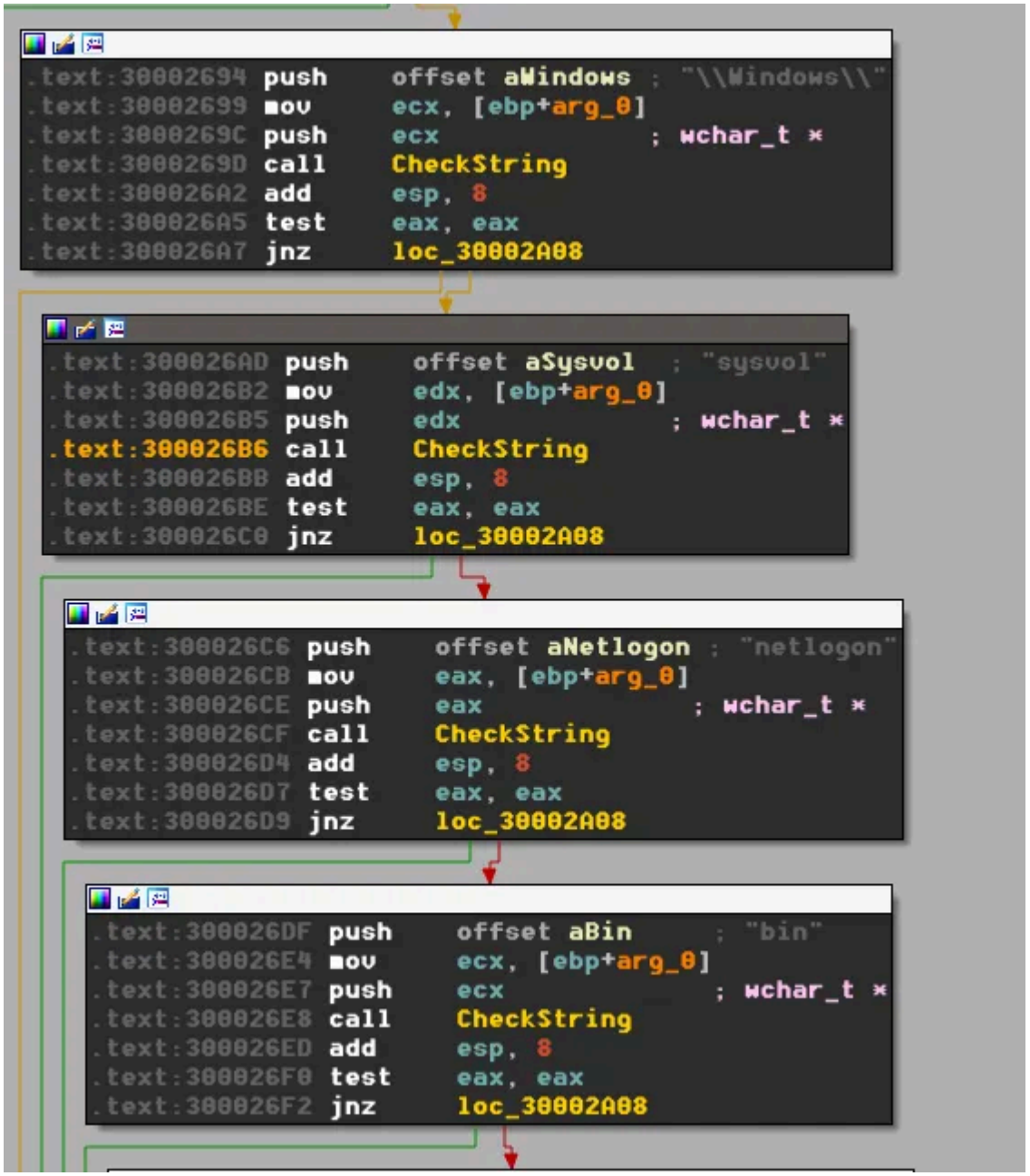


It uses a Marker “HERMES” to identify if the file is encrypted or not.

Its Uses:

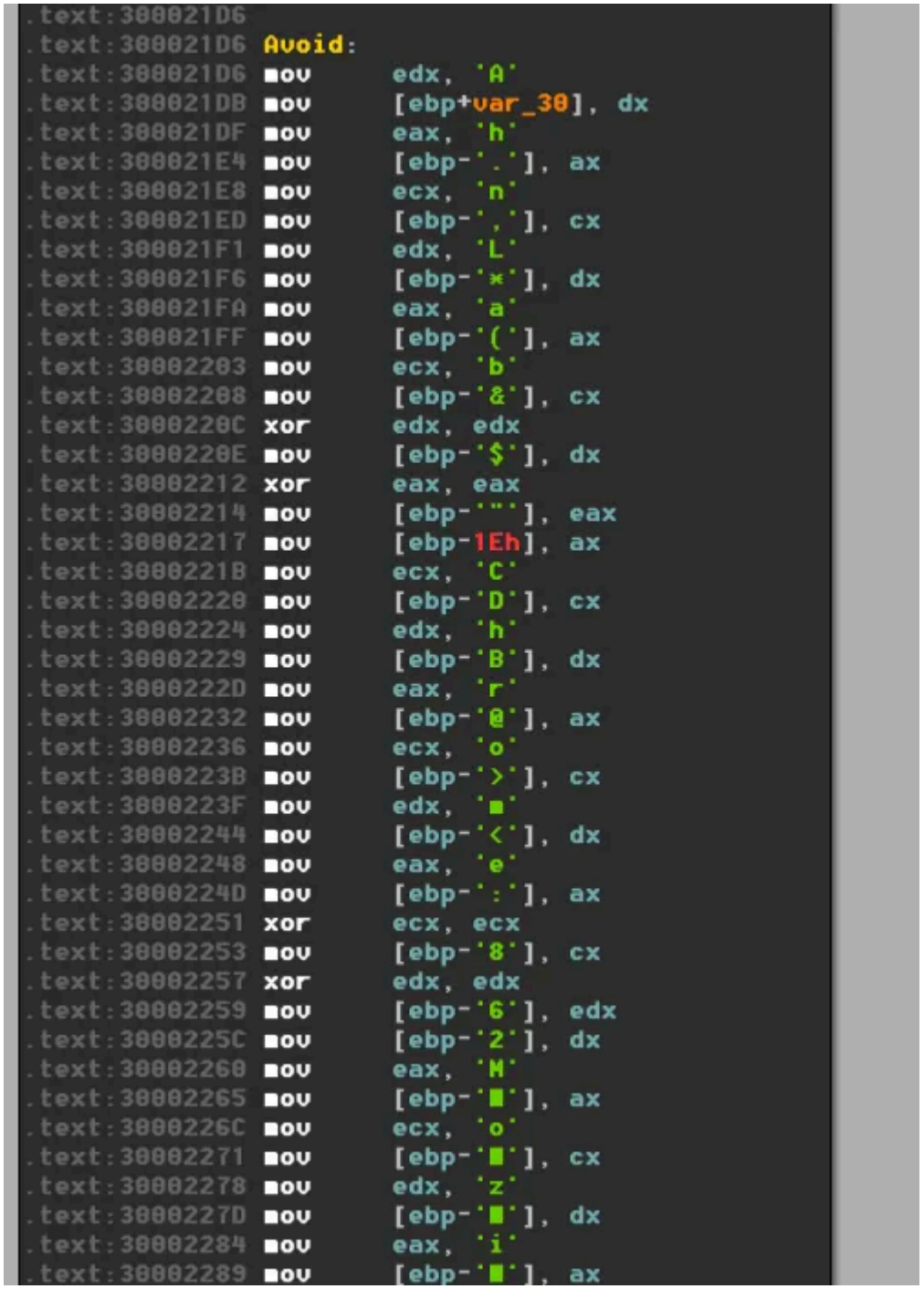
```
[+] CryptEncrypt
[+] CryptGenKey
[+] CryptDecrypt
[+] CryptAquireContextW
[+] CryptDestroyKey
[+] CryptDeriveKey
[+] CryptImportKey
```

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
[+] hrmlog
[+] ini
[+] exe
```

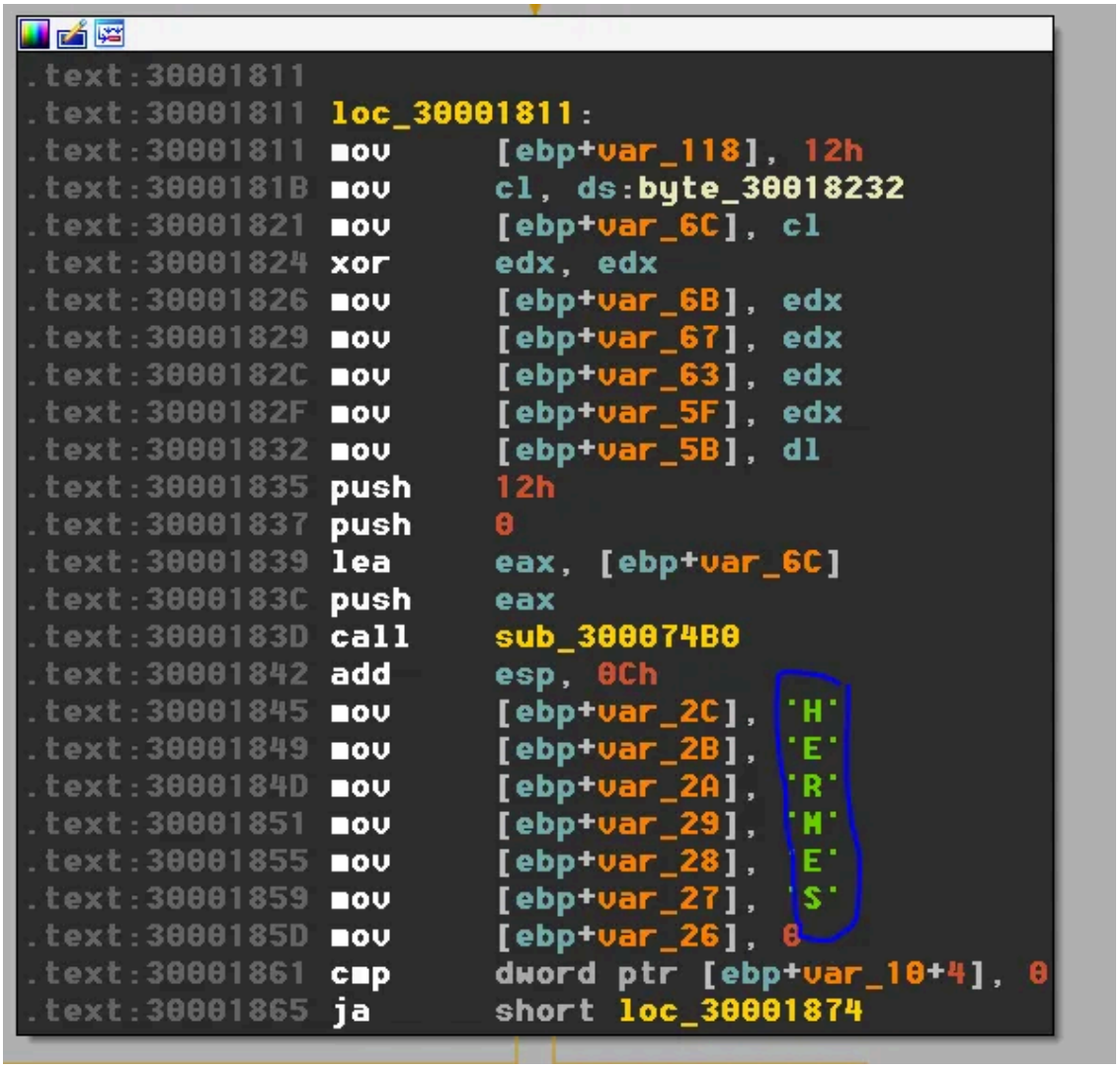


Here it builds Strings on stack for folders to avoid encrypting its files or skipping it

```
[+] 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.



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

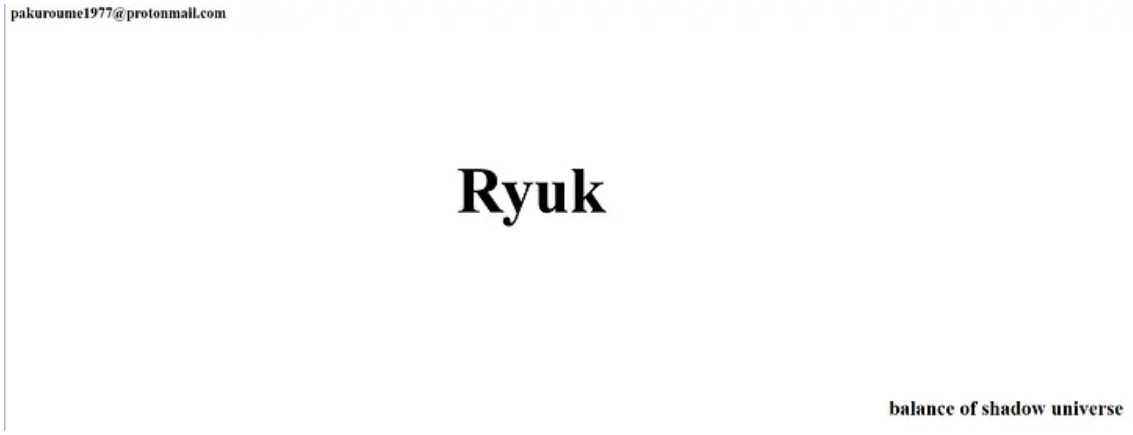
[+] Data Encrypted for Impact T1486

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



Refrences:

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

---

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