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

By Goutam Tripathy

Published: 2020-02-13 · Archived: 2026-04-05 22:03:29 UTC

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.

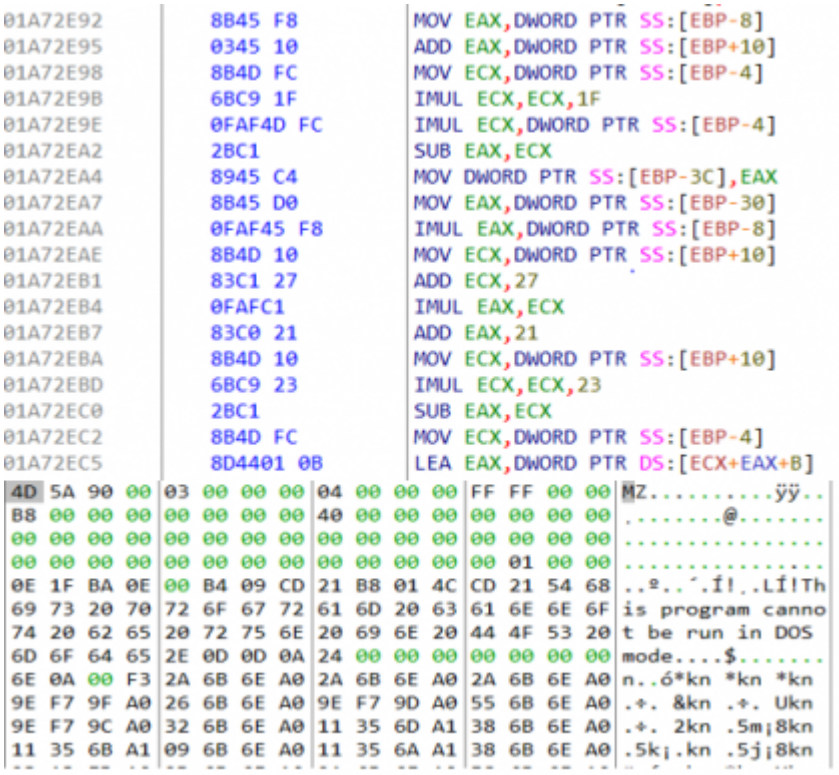


Fig 1:Final Unpack Routine

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.

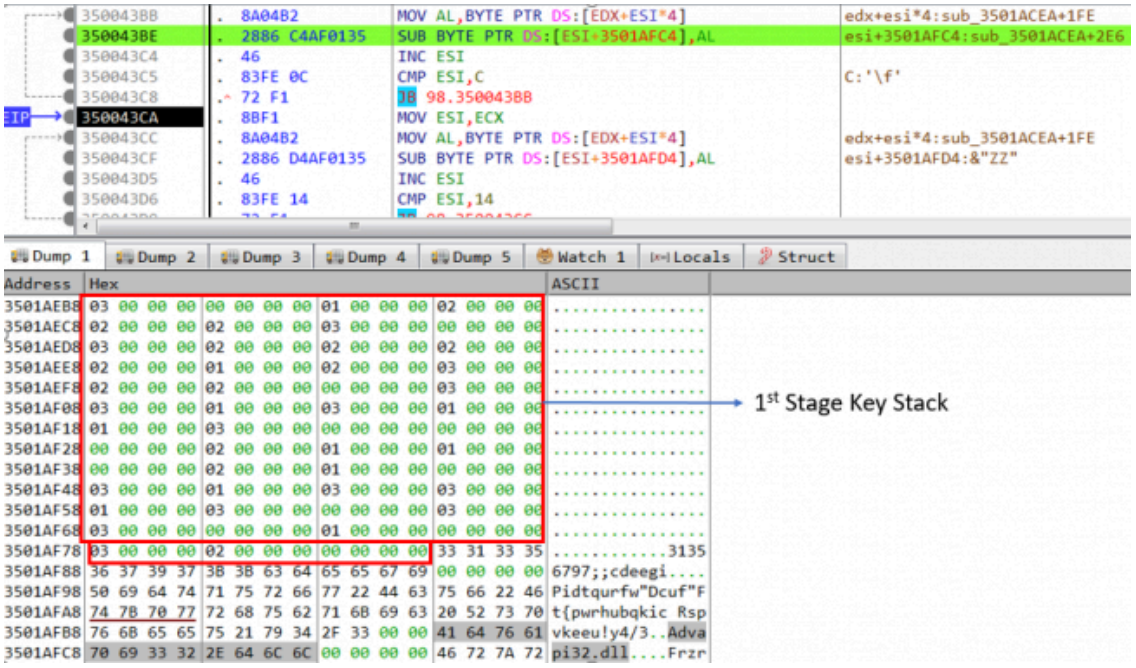


Fig 2:De-obfuscation of 1st stage obfuscated string

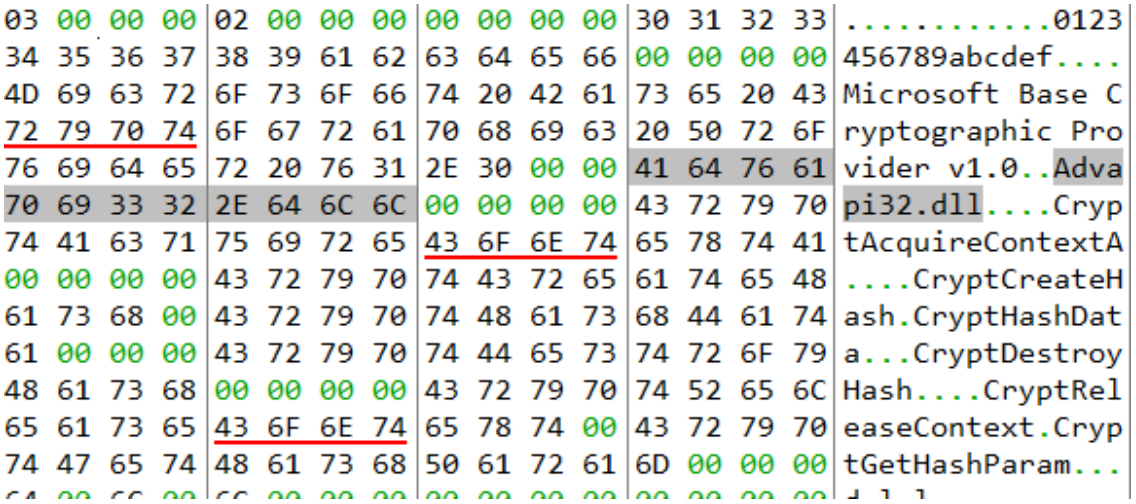


Fig 3:After de-obfuscation

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.

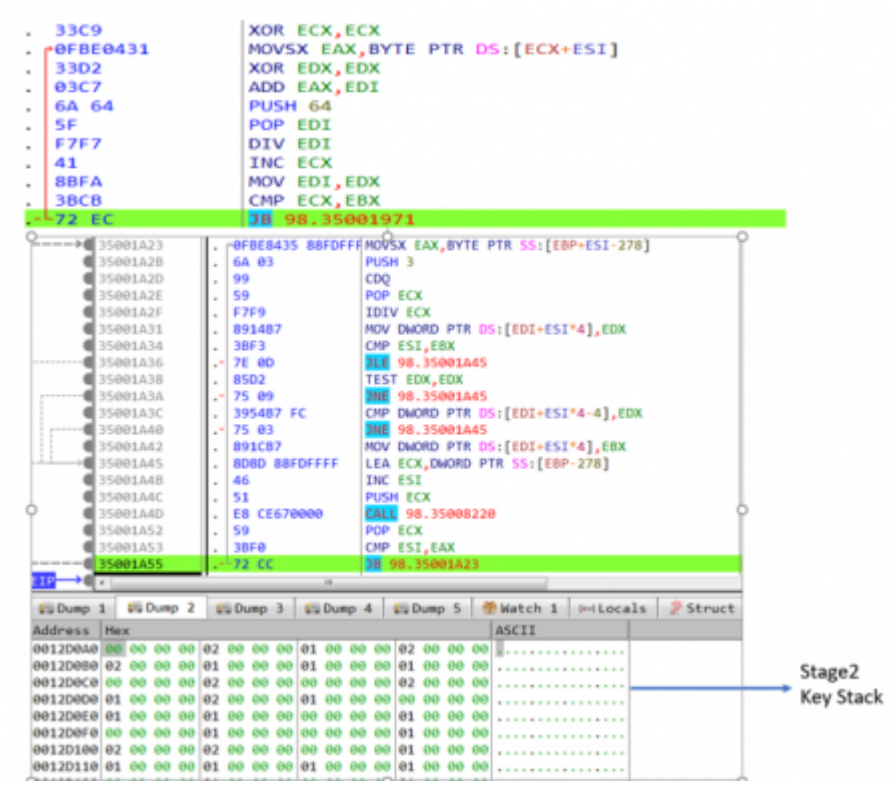


Fig 4:Generation of Stage-2 key stack

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.

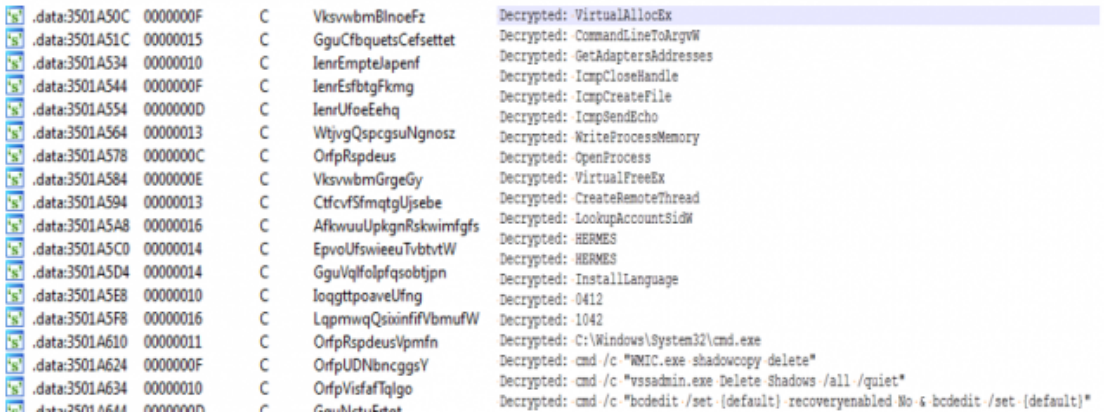


Fig 5:Part of Obfuscated and De-Obfuscate strings

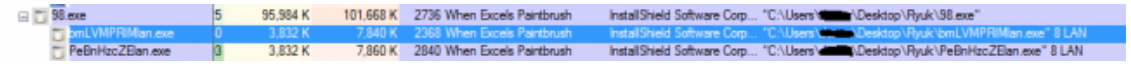
```
push    esi
call    sub_35002A43
mov     esi, ds:LoadLibraryA
push    offset aKgspgm43_fln ; "Kgspgm43.fln"
call    esi ; LoadLibraryA
push    offset aMrs0fmm ; "mrs0fmm"
mov     hModule, eax
call    esi ; LoadLibraryA
push    offset aAfwcrj43_fln ; "afwcrj43.fln"
mov     dword_3501BF88, eax
call    esi ; LoadLibraryA
push    offset aOnf54Eml ; "onf54/eml"
mov     dword_3501BF08, eax
call    esi ; LoadLibraryA
push    offset aSjfnn43Dnl ; "Sjfnn43/dnl"
mov     dword_35157D90, eax
call    esi ; LoadLibraryA
push    offset aIrinrbqj_fln ; "Irinrbqj.fln"
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
mov     dword_3501BFC0, eax
call    esi ; GetProcAddress
push    offset aDgmgvfgjlgw ; "DgmgvfGjlgW"
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\[redacted]\Desktop\Ryuk\98.exe"
bmLVMPFIMan.exe	0	3,832 K	7,840 K	2368 When Excels Paintbrush	InstallShield Software Corp...	"C:\Users\[redacted]\Desktop\Ryuk\bmLVMPFIMan.exe" 8 LAN
PeBnHzcZElan.exe	3	3,832 K	7,860 K	2840 When Excels Paintbrush	InstallShield Software Corp...	"C:\Users\[redacted]\Desktop\Ryuk\PeBnHzcZElan.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

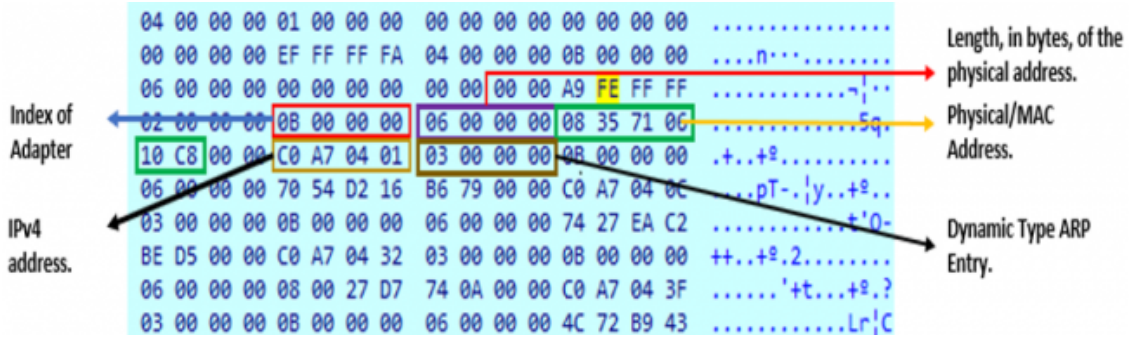


Fig 9:Structure Of ARP Table

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.

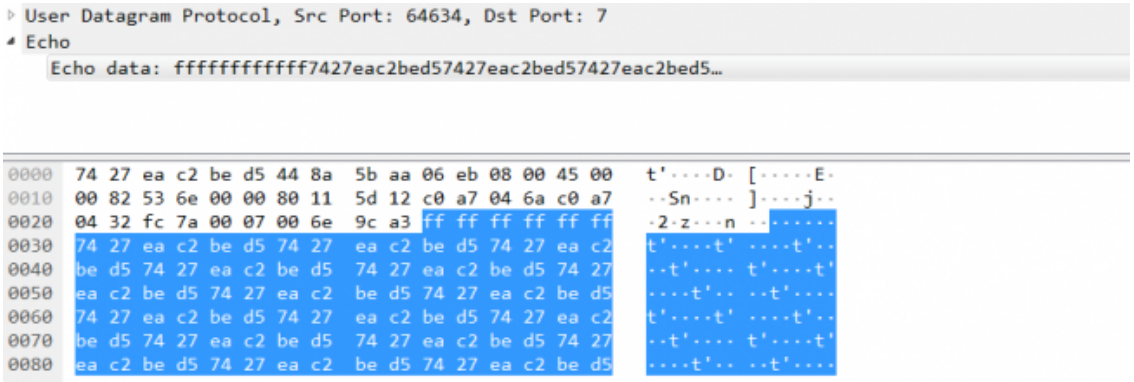


Fig 10:Magic Packet for WoL

```
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↓o ...
                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.

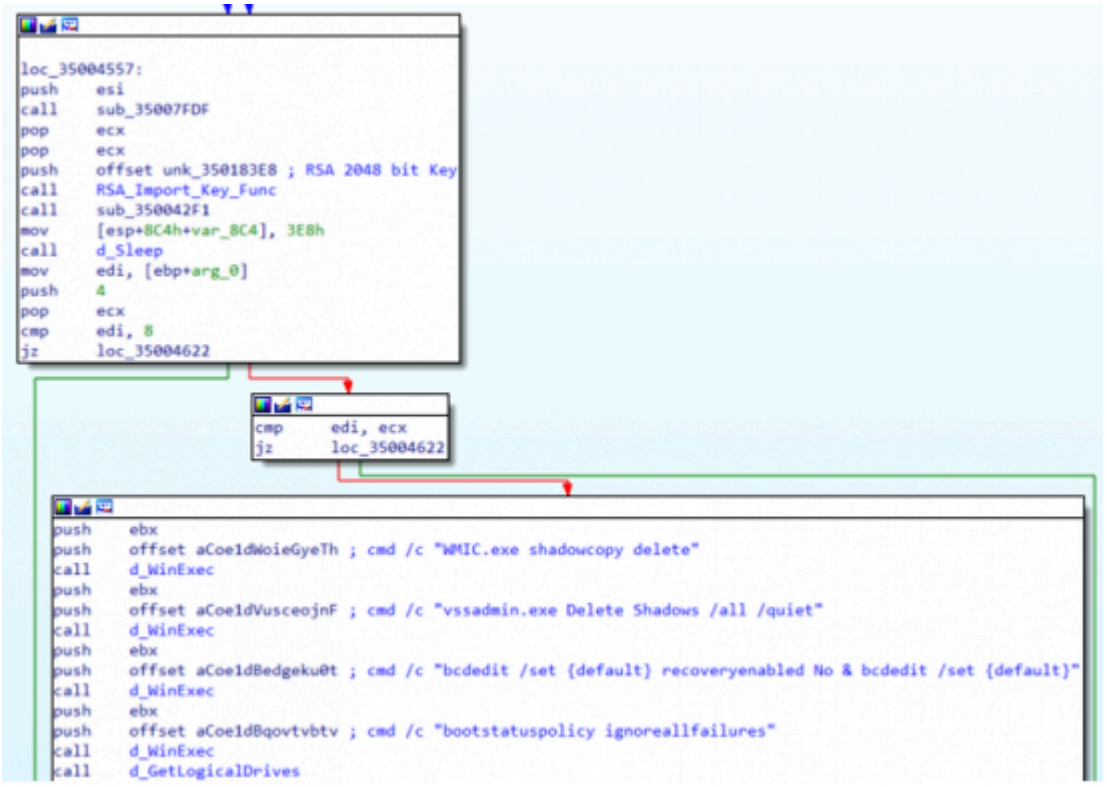


Fig 13:Importing RSA Public Key and Deleting Shadow Copy

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.

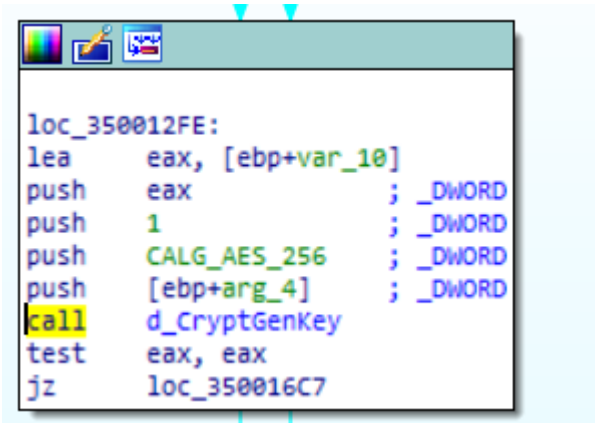


Fig 14:Generating AES 256 bit Using CryptGenKey

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.

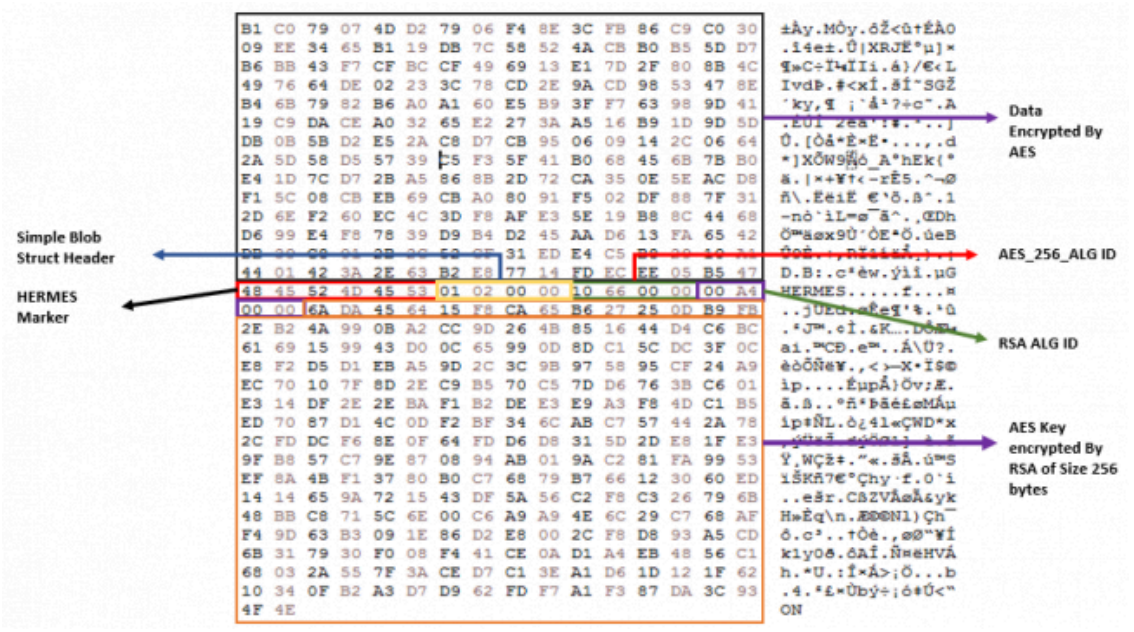


Fig 15:Encrypted File Structure

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.HermezRI.S10666632
