From Infection to Encryption: Tracing the Impact of RYUK Ransomware

medium.com/@shaddy43/from-infection-to-encryption-tracing-the-impact-of-ryuk-ransomware-64bd8656781c

Shayan Ahmed Khan April 20, 2024

Shayan Ahmed Khan

--

Ryuk ransomware is a very famous and deadly piece of malware that was first discovered in mid 2018 and has been active since. There are multiple variants of Ryuk that keeps surfacing again on different platforms and sandboxes. Ryuk focuses on targeting critical organizations like healthcare and finance.

Ryuk name likely originates from Popular anime show “Death Note”

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.

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.

Check out my Github Repo for Malware Analysis Series!!!

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:

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.

-
-
-

-

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:

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:

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:

1. VirtualAllocEx
2. WriteProcessMemory
3. 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 .

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.

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:

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

- CryptExportKey
- DeleteFileW
- GetDriveTypeW
- GetCommandLineW
- GetStartupInfoW
- FindNextFileW
- VirtualAlloc
- GetUserNameA
- ExitProcess
- CreateProcessA
- GetIpNetTable
- ReadFile
- RegQueryValueExA
- RegSetValueExW
- CopyFileA
- SetFileAttributesW
- WinExec
- CryptDeriveKey

- CryptGenKey
- Sleep
- GetCurrentProcess
- ShellExecuteW
- GetFileSize
- GetModuleFileNameA
- CreateFileA
- GetFileSizeEx
- WriteFile
- GetLogicalDrives
- WNetEnumResourceW
- RegOpenKeyExW
- WNetCloseEnum
- GetWindowsDirectoryW
- GetTickCount
- FindFirstFileW
- CryptAcquireContextW
- MoveFileExW
- CryptDecrypt
- CryptImportKey
- CreateProcessW
- CreateThread
- CryptDestroyKey
- CoCreateInstance
- CryptEncrypt
- RegDeleteValueW

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:

-
-
-
-
-

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:

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:

These are only a few of services and processes listed here above.

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 )}
```