

Challenges in Cyber Security: Ransomware Phenomenon



Vlad-Raul Paşca and Emil Simion

Abstract Ransomware has become one of the major threats nowadays due to its huge impact and increased rate of infections around the world. According to <https://www.adaware.com/blog/cryptowall-ransomware-cost-users-325-million-in-2015>, just one family, CryptoWall 3, was responsible for damages of over 325 millions of dollars, since its discovery in 2015. Recently, another family of ransomware appeared in the cyberspace which is called WannaCry, and according to <https://www.cnet.com/news/wannacry-wannacrypt-uiwix-ransomware-everything-you-need-to-know>, over 230,000 computers around the world, in over 150 countries, were infected. This type of ransomware exploited a vulnerability which is present in the Microsoft Windows operating systems called EternalBlue, an exploit which was developed by the US National Security Agency (NSA) and released by The Shadow Brokers on April 14, 2017.

Spora ransomware is a major player in the field of ransomware families and is prepared by professionals. It has the ability to encrypt files offline like other families of ransomware, DMA Locker 3.0, Cerber, or some editions of Locky. Currently, there is no decryptor available in the market for the Spora ransomware.

Spora is distributed using phishing e-mails and infected websites which drops malicious payloads. There are some distribution methods which are presented in <http://malware-traffic-analysis.net/2017/02/14/index2.html> (the campaign from February 14, 2017) and <http://malware-traffic-analysis.net/2017/03/06/index.html> (the campaign from March 6, 2017).

Once the infection has begun, Spora runs silently and encrypts files with a specific extension, not all extensions are encrypted. This type of ransomware is interested in office documents, PDF documents, Corel Draw documents, database files, images, and archives and is important to present the entire list of extension in order to warn people about this type of attack: xls, doc, xlsx, docx, rtf, odt, pdf, psd, dwg, cdr, cd, mdb, lcd, dbf, sqlite, accdb, jpg, jpeg, tiff, zip, rar, 7z, backup, sql,

V.-R. Paşca (✉) · E. Simion

Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest,
Bucharest, Romania

e-mail: emil.simion@upb.ro

and bak. One crucial point here is that everybody can rename the files in order to avoid such infections, but the mandatory requirement is to back up the data.

Spora doesn't add extensions to the encrypted files, which is really unusual in the case of ransomware, for example, Locky adds .locky extension, TeslaCrypt adds .aaa extension, and WannaCry appends .WNCRY extension. In this case, each file is encrypted with a separate key, and it is a nondeterministic encryption (files with an identical content are encrypted in different ciphertexts); the content which was encrypted has a high entropy and visualization of an encrypted file, which suggests that a stream cipher or chained block was used (AES in CBC mode is suggested, because of the popularity of this mode of operation in ransomware's encryption schemes).

There are some methods which are used frequently to assure that a single copy of a malware is running, for example, the creation of a mutex, which means that the encrypted data is not encrypted again; therefore, we have a single step of encryption. Of course, there are some folders which are excluded from encryption, because the system must remain in a working state in order to make a payment, so Spora doesn't encrypt the files which are located in the following directories: windows, program files, program files (x86), and games.

Spora uses Windows Crypto API for the whole encryption process. Firstly the malware comes with a hardcoded AES 256 key, which is being imported using CryptImportKey (the parameters which are passed to this function reveal that an AES 256 key is present). The AES key is further used to decrypt another key, which is a RSA public key, using a CryptDecrypt function (a ransom note is also decrypted using the AES key, as well as a hardcoded ID of the sample).

For every computer, Spora creates a new pair of RSA keys. This process uses the function CryptGenKey with some parameters which are specific for RSA keys, after that the private key from the pair is exported using the function CryptExportKey and Base64 encoded using the function CryptBinaryToString. A new AES 256 key is generated using CryptGenKey, is exported using CryptExportKey, and is used to encrypt the generated private RSA key (finally, the key is encrypted using the hardcoded RSA public key and stored in the ransom note). For every file a new AES key is generated which is used to encrypt the file, is encrypted using the generated public RSA key, and is stored at the end of every encrypted file.

Spora is a professional product created by skilled attackers, but the code is not obfuscated or packed, which makes the analysis a little bit easier. The implementation of cryptographic algorithms uses the Windows Crypto API and seems to be consistent; nonetheless the decryption of files is not really possible without paying the ransom. The ability to handle a complex process of encryption offline makes Spora ransomware a real danger for unprepared clients.

Ransomware usually uses the RSA algorithm to protect the encryption key and AES for encrypting the files. If these algorithms are correctly implemented, then it is impossible to recover the encrypted information.

Some attacks, nonetheless, work against the implementation of RSA. These attacks are not against the basic algorithm, but against the protocol. Examples of such attacks on RSA are chosen-ciphertext attack, common modulus attack, low

encryption exponent attack, low decryption exponent attack, attack on encryption and signing with the same pair of keys, and attack in case of small difference between prime numbers p and q .

The attacks on AES implementation include ECB attack, CBC implementation without HMAC verification and oracle padding attack.

In the following sections, we present the fully analysis on three representative ransomware: Spora, DMA Locker, and WannaCry.

1 Spora Ransomware

Name: 9ae49d4a4202b14efe.exe
md5: 116d339b412cd1baf48bcc8e4124a20b
Type: encrypting ransomware

In Fig. 1 a detection report by VirusTotal scanner mechanism is presented, which shows that the malware is known and most vendors already offer a protection mechanism for it. Figure 2 shows us that the malware itself is not packed; nonetheless later results will show that the malware is obfuscated and hence the complexity of the analysis grows.

Figure 3 shows a string which is pushed on the stack 699 times; this trick is used to obfuscate the code.

In Fig. 4 it is shown that a function is called 700 times (the function calls **OpenMutexA**, which tries to open an existing mutex), which doesn't make sense

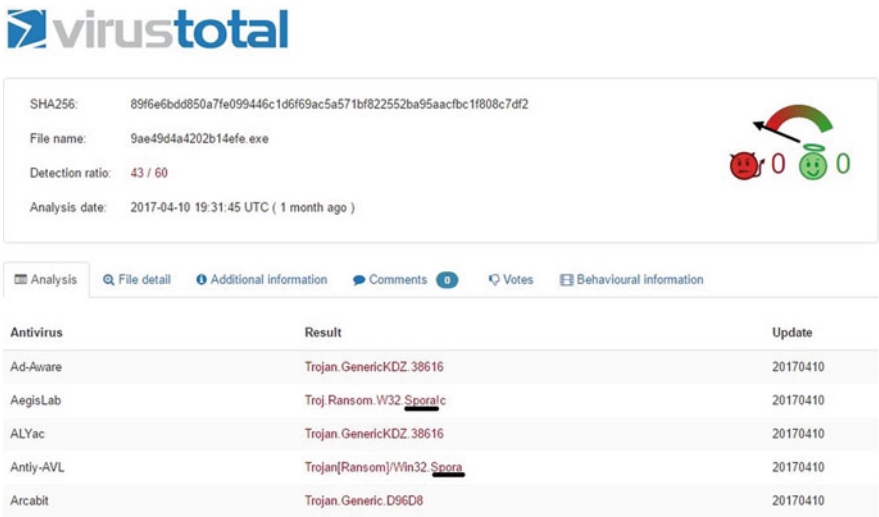


Fig. 1 VirusTotal report

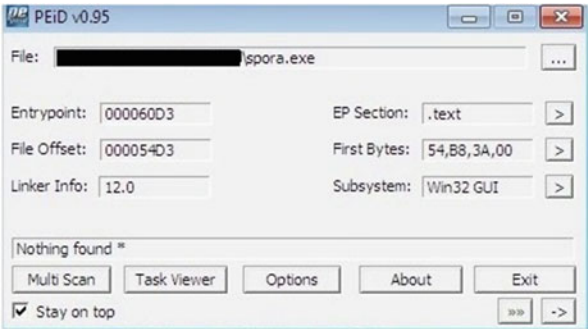


Fig. 2 PEiD report

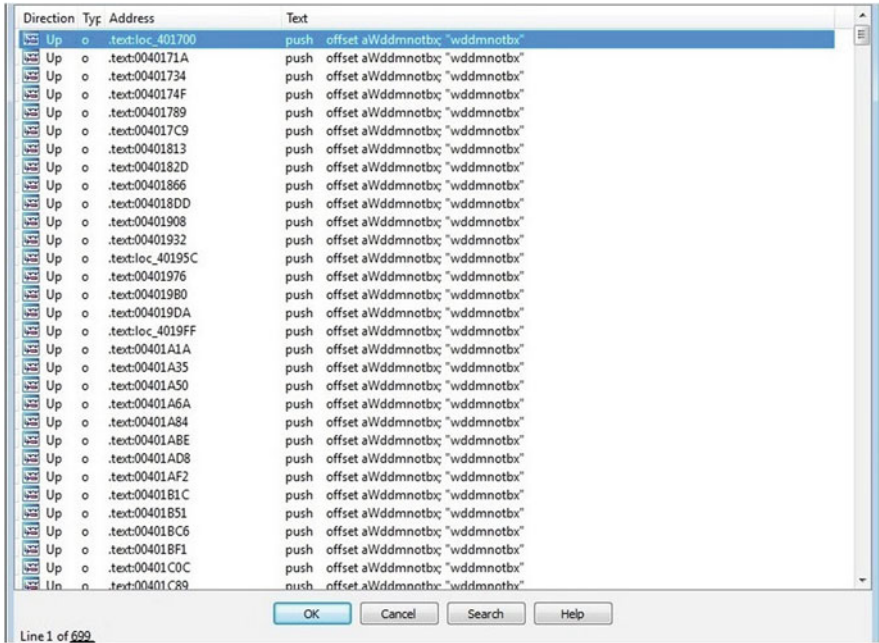


Fig. 3 IDA Pro 1

in this case, because the malware doesn't call **CreateMutexA**; this is another trick used to complicate the analysis. The malware uses the function **VirtualAlloc** to allocate space in the process address space, and then it writes the actual payload in that space. The initial conclusion is that the initial executable is just a packer and the actual malicious code is contained in the newly executable, which has the md5 97e84cc8afca475d15d8c3e1f38d deba.

The malware calls **GetVolumeInformationW** to get information about the file system and volume associated with the root directory, as shown in Fig. 5.

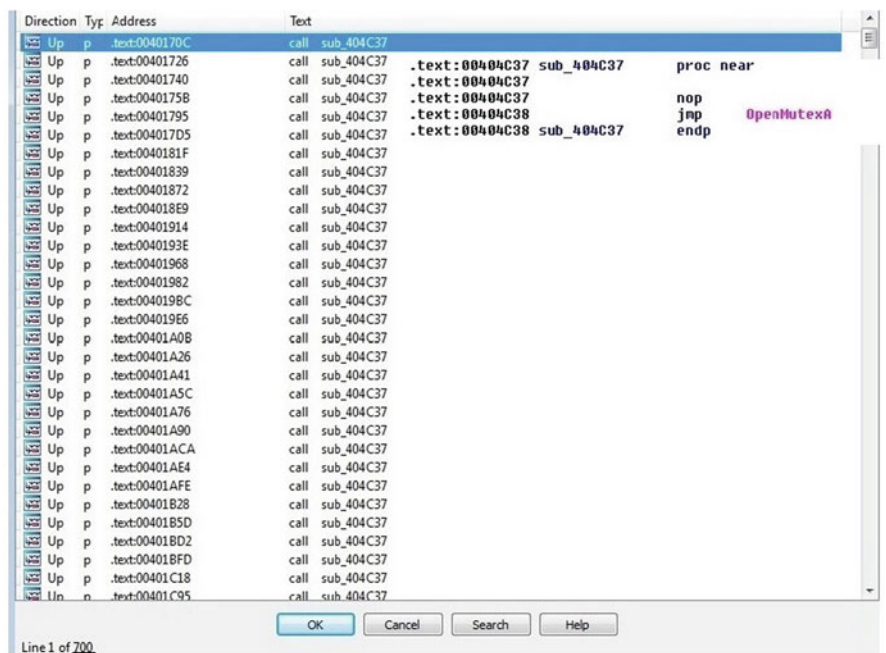


Fig. 4 IDA Pro 2

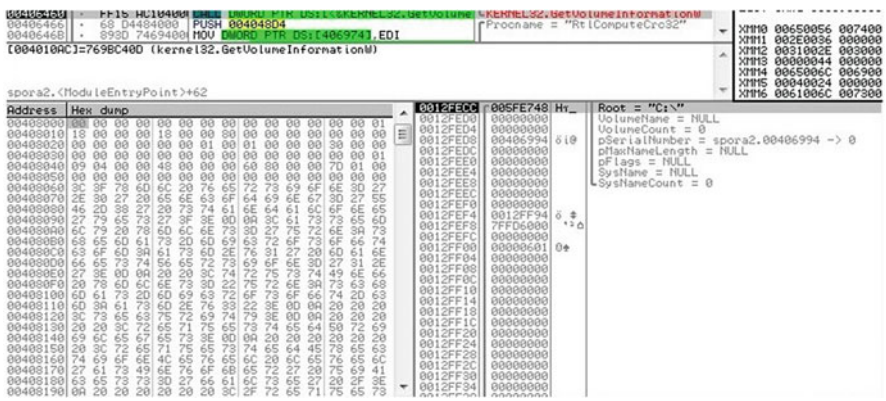


Fig. 5 GetVolumeInformationW call

A mutex is created and it has the following format: m(GetVolumeInformationResult) (in decimal), to ensure that the malware runs only once. The sample creates a file which has the following name: C:\Users\<user>\AppData\Roaming\<Mutex>. The malware comes with a hardcoded key, which is being imported using the function **CryptImportKey**, as shown in Fig. 6. It represents an AES256 key, stored in a form of a blob. The

```

00405B70 | . 51          | PUSH ECX
00405B71 | . 56          | PUSH ESI
00405B72 | . 8D45 FC     | LEA EAX, [LOCAL.1]
00405B75 | . 50          | PUSH EAX
00405B76 | . 33F6        | XOR ESI, ESI
00405B78 | . 56          | PUSH ESI
00405B79 | . 56          | PUSH ESI
00405B7A | . 6A 2C       | PUSH 2C
00405B7C | . 68 A0114000 | PUSH 004011A0
00405B81 | . FF35 7869400 | PUSH DWORD PTR DS:[406978]
00405B87 | . FF15 4810400 | CALL DWORD PTR DS:[4ADAP132.CryptImportKey]
00405B8D | . 5C          | TEST EAX, EAX
00405B8F | . 74 21       | JZ SHORT 00405BB2
00405B91 | . 8D45 0C     | LEA EAX, [ARG.2]
00405B94 | . 50          | PUSH EAX
00405B95 | . FF75 08     | PUSH DWORD PTR SS:[ARG.1]
00405B98 | . 56          | PUSH ESI
[00401048]=752ABBS2 (ADUAP132.CryptImportKey)

```

Address	Hex dump	ASCII
004011A0	08 02 00 00 10 66 00 00 20 00 00 00 54 D6 37 E5	08 02 00 00 10 66 00 00 20 00 00 00 54 D6 37 E5
004011B0	81 95 04 4B 25 7E 5E 72 B9 AF 7C E9 E3 3C 32 AC	81 95 04 4B 25 7E 5E 72 B9 AF 7C E9 E3 3C 32 AC
004011C0	29 16 A6 3C 73 64 2B 42 2A 58 08 4A BF CD 68 B0	29 16 A6 3C 73 64 2B 42 2A 58 08 4A BF CD 68 B0
004011D0	63 B5 55 31 33 01 C2 7B 02 DF 8E B9 1A 05 F5 5F	63 B5 55 31 33 01 C2 7B 02 DF 8E B9 1A 05 F5 5F
004011E0	C4 4B 20 D1 25 A4 B4 07 45 1E E3 CD 78 45 40 00	C4 4B 20 D1 25 A4 B4 07 45 1E E3 CD 78 45 40 00
004011F0	73 01 FF 7E B0 CA 2B 7D B0 1E 79 73 71 F8 B9 BE	73 01 FF 7E B0 CA 2B 7D B0 1E 79 73 71 F8 B9 BE
00401200	67 93 AC 93 F1 76 08 A6 A2 6A DB 6E 33 C6 65 1E	67 93 AC 93 F1 76 08 A6 A2 6A DB 6E 33 C6 65 1E
00401210	39 9F 48 FE 3E 35 EE DC 40 2C 15 FF 5B 49 AD 45	39 9F 48 FE 3E 35 EE DC 40 2C 15 FF 5B 49 AD 45
00401220	BF 18 1B 03 9E B0 F6 A5 5C 26 4B EA D1 E5 91 1B	BF 18 1B 03 9E B0 F6 A5 5C 26 4B EA D1 E5 91 1B
00401230	A1 B1 67 58 16 A8 F4 00 69 9D 13 C4 55 C2 A3 FC	A1 B1 67 58 16 A8 F4 00 69 9D 13 C4 55 C2 A3 FC
00401240	BB B4 C5 38 27 7C 20 72 BD 50 3E 43 25 2B 7B 0B	BB B4 C5 38 27 7C 20 72 BD 50 3E 43 25 2B 7B 0B
00401250	10 BC 53 7B EF 36 A5 C3 E3 4D 90 85 2D F1 ED 10	10 BC 53 7B EF 36 A5 C3 E3 4D 90 85 2D F1 ED 10
00401260	B4 E7 10 60 E3 F8 E7 00 CF 36 0E C4 9F 67 26 D2	B4 E7 10 60 E3 F8 E7 00 CF 36 0E C4 9F 67 26 D2
00401270	A3 7C B1 49 D4 83 83 61 B4 E3 70 97 AC 6E 4F 4B	A3 7C B1 49 D4 83 83 61 B4 E3 70 97 AC 6E 4F 4B
00401280	72 2F DF 9F 49 74 FE 96 18 EA 7B 2E CA 69 2D 5E	72 2F DF 9F 49 74 FE 96 18 EA 7B 2E CA 69 2D 5E
00401290	DF 73 97 A4 7E F3 BC D7 36 71 1E 84 0F 59 E7 F0	DF 73 97 A4 7E F3 BC D7 36 71 1E 84 0F 59 E7 F0
004012A0	8A 40 67 85 61 29 6D FE F1 14 4E D6 99 A9 AE D6	8A 40 67 85 61 29 6D FE F1 14 4E D6 99 A9 AE D6
004012B0	C3 0E 59 9C 2A 58 0A 1D 09 BB EA 21 B7 7A F7 F8	C3 0E 59 9C 2A 58 0A 1D 09 BB EA 21 B7 7A F7 F8
004012C0	D8 80 41 A5 F8 25 D0 B5 03 4B 27 39 F1 09 43 25	D8 80 41 A5 F8 25 D0 B5 03 4B 27 39 F1 09 43 25
004012D0	63 34 4E 25 81 10 5C 71 B3 D2 B4 FA 8C 4A C8 80	63 34 4E 25 81 10 5C 71 B3 D2 B4 FA 8C 4A C8 80
004012E0	90 3A BF 8C 4A D4 96 98 1E E8 FB 73 2A 40 10 B6	90 3A BF 8C 4A D4 96 98 1E E8 FB 73 2A 40 10 B6
004012F0	79 08 9E FA 28 A4 A3 4F F7 17 61 C4 F7 F7 E8 41	79 08 9E FA 28 A4 A3 4F F7 17 61 C4 F7 F7 E8 41
00401300	DE 52 E0 54 9A F0 22 CF 73 15 B9 DC 58 B2 37 1E	DE 52 E0 54 9A F0 22 CF 73 15 B9 DC 58 B2 37 1E
00401310	2F B0 6D BE 7F A8 89 ED 51 DE 13 7A EB 70 57 7A	2F B0 6D BE 7F A8 89 ED 51 DE 13 7A EB 70 57 7A
00401320	12 C2 38 20 02 1B 00 F1 97 77 5E 8E 10 7B 88 29	12 C2 38 20 02 1B 00 F1 97 77 5E 8E 10 7B 88 29
00401330	F4 83 32 47 59 8F 54 BE D3 9B 00 57 2E 14 9D A2	F4 83 32 47 59 8F 54 BE D3 9B 00 57 2E 14 9D A2

Fig. 6 CryptImportKey call

explanation of the fields is: 08 represents PLAINTEXTKEYBLOB and means that the key is a session key; 02 CUR_BLOB_VERSION, 0x00006610 which represents Alg_ID: CALG_AES_256, 0x20=32 represents key length.

The AES key is used to decrypt another key, which is a RSA key embedded in the binary, as shown in Fig. 7. The AES key is also used to decrypt the ransom note and the binary's hardcoded ID. The malware uses **GetLogicalDrives** to obtain the currently available disk drives and then a loop, which selects the files that have a specified extension which is attacked by this ransomware, is created. The malware also uses **WNetOpenEnum** and **WNetEnumResource** APIs to enumerate the network resources, and the created file is used to store temporary data, like the files which will be encrypted.



Fig. 7 CryptDecrypt calls

The attacked extensions are presented in the table below:

.xls	.doc	.xlsx	.docx	.rtf	.odt	.pdf	.ppt	.pptx
.psd	.dwg	.cdr	.cd	.mdb	.lcd	.dbf	.sqlite	.accdb
.jpg	.jpeg	.tiff	.zip	.rar	.7z	.backup	.sql	.bak

The next folders are excluded from the attack:

windows	programfiles	programfiles(x86)	games
---------	--------------	-------------------	-------

For every victim, the malware creates a pair of RSA keys. The fragment which generates the RSA key pair (1024 bits) is shown in Fig. 8.

The relevant parameters for **CryptGenKey** are 0xA400 which represents AlgId: CALG_RSA_KEYX and 0x04000001 which represents RSA1024BIT_KEY | CRYPT_EXPORTABLE. The private RSA key is exported and Base64 encoded, as shown in Fig. 9. The encryption of the private RSA key is stored into a buffer alongside the data regarding the machine and the infection, like date, username, country code, malware ID, and statistics of encrypted file types. An example is shown in Fig. 10. The malware uses a MD5 algorithm to hash the buffer which contains the private RSA key (the hash is used to create the user ID) as shown in Fig. 11. Another AES key is generated; then it's exported and encrypted using public RSA key that was hardcoded. In Fig. 12 this process is shown. The generated AES key is used to encrypt the data (including the RSA private key), as shown in Fig. 13. Finally, all encrypted data is Base64 encoded and stored in the ransom note.

```

00406717 lea    eax, [esp+0A0h+phKey]
00406718 push   eax                ; phKey
0040671C push   4000001h          ; dwFlags
00406721 push   0A400h            ; Algid
00406726 push   hProv            ; hProv
0040672C call  CryptGenKey
00406732 test   eax, eax
0040673A jz     loc_4067B8

0040673A push   [esp+0A0h+phKey] ; hKey
0040673E call  sub_404F72
00406743 mov   ebx, eax
00406745 mov   [esp+0A0h+var_88], ebx
00406749 cmp   ebx, ebp
0040674B jz     short loc_4067B1
  
```

Hex dump (Address 00406718):

Address	Hex	dump	ASCII
0012FFC0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFC1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFC2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFC3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFC4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFC5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFC6	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFC7	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFC8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFC9	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFCA	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFCB	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFCC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFCD	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFCE	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFCF	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFD0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFD1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFD2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFD3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFD4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFD5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFD6	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFD7	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFD8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFD9	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFDA	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFDB	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFDC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFDD	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFDE	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFDF	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFE0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFE1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFE2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFE3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFE4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFE5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFE6	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFE7	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFE8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFE9	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFEA	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFEB	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFEC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFED	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFEE	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFEF	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFF0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFF1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFF2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFF3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFF4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFF5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFF6	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFF7	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFF8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFF9	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFFA	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFFB	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFFC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFFD	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFFE	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0012FFFF	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		

Fig. 8 CryptGenKey call

```

00404FD5 . FF15 64104000 . DWORD PTR DS:[7528B4C7,CryptBinaryToSt
00404FD6 . 8510 . TEST EAX,EAX
00404FD7 . 0F84 24010000 . JZ 00405101
00404FD8 . FF75 F8 . PUSH DWORD PTR SS:[LOCAL.2]
00404FD9 . 6A 40 . PUSH 40
00404FE0 . FFD6 . PUSH ESI
00404FE1 . 3945 08 . MOV DWORD PTR SS:[IARG.1],EAX
00404FE2 . 3BC3 . CMP EAX,EBX
00404FE3 . 0F84 12010000 . JE 00405101
00404FE4 . 3D40 F8 . LEA ECX,[LOCAL.2]
00404FE5 . 51 . PUSH ECX
00404FE6 . 50 . PUSH EAX
00404FE7 . 50 . PUSH EAX
00404FE8 . 6A 01 . PUSH 1
00404FE9 . FF75 FC . PUSH DWORD PTR SS:[LOCAL.1]
00404FEA . FF75 F4 . PUSH DWORD PTR SS:[LOCAL.3]
00404FEB . FF15 6C104000 . DWORD PTR DS:[7528B4C7,CryptBinaryToSt
00404FEC . 8B45 F8 . MOV EAX,DWORD PTR SS:[LOCAL.2]
00404FED . 05 BE020000 . ADD EAX,0
00404FEE . 50 . PUSH EAX
00404FEF . 6A 40 . PUSH 40
00404FF0 . FFD6 . PUSH ESI
00404FF1 . 3BF8 . MOV EDI,EAX
00404FF2 . 8970 F0 . MOV DWORD PTR SS:[LOCAL.4],EDI
00404FF3 . 3BF8 . CMP EDI,EBX
00404FF4 . 0F84 DC000000 . JE 004050F8
00404FF5 . 68 10464000 . PUSH 00404610
00404FF6 . 57 . PUSH EDI
00404FF7 . FF15 E4104000 . DWORD PTR DS:[7528B4C7,CryptBinaryToSt
00404FF8 . FF75 08 . PUSH DWORD PTR SS:[IARG.1]
00404FF9 . 8B35 E8104000 . MOV ESI,DWORD PTR DS:[7528B4C7,CryptBinaryToSt
00404FFA . 57 . PUSH EDI
00404FFB . FFD6 . PUSH ESI
00404FFC . 68 F0454000 . PUSH 004045F0
00404FFD . 57 . PUSH EDI
00404FFE . FFD6 . PUSH ESI
  
```

ArgS => OFFSET LOCAL.2
Arg4
Arg3 => 1
Arg2 => [LOCAL.1]
Arg1 => [LOCAL.3]
CRYPT32.CryptBinaryToStringA
Src = "-----BEGIN RSA PRIVATE KEY-----J8"
Dest
KERNEL32.istrncpy
Src => [ARG.1]
Dest
KERNEL32.istrcat
ASCII "-----END RSA PRIVATE KEY-----J8"

Fig. 9 RSA key is Base64 encoded

For every file a new AES256 key is generated, as shown in Fig. 14. The AES key is encrypted using the generated public RSA key, and it is appended to the encrypted file; also the CRC32 is being computed and stored in the file (Fig. 15). Each file is encrypted using the AES key, as shown in Fig. 16.

In order to decrypt a file, a ransom note is uploaded to the server giving the attacker access to all information needed. He uses the private RSA key corresponding to the hardcoded public RSA key to decrypt the first AES key, and then the key is used to decrypt the generated private RSA key. Because of the fact that each AES256 key is encrypted using the corresponding public RSA key and stored at the end of each file, it is possible to decrypt each key and then decrypt each file individually.

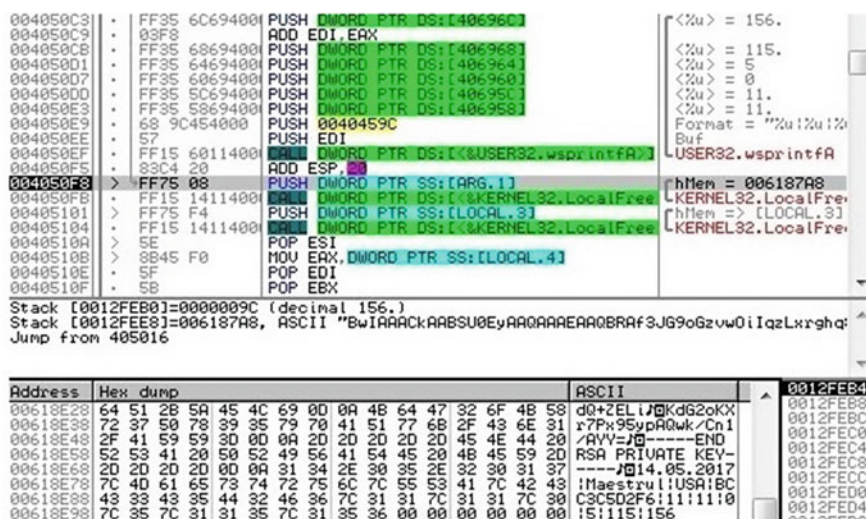


Fig. 10 Buffer contains information about the system

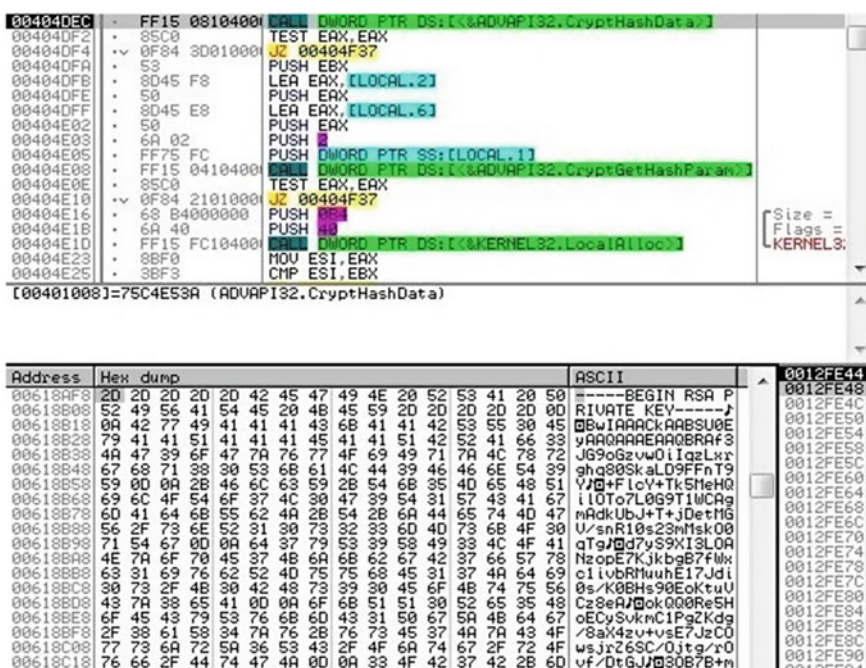


Fig. 11 MD5 Algorithm is used to hash the buffer

00405130	FF15 1C104000	CALL DWORD PTR DS:[<ADUAPI32.CryptGenKey>]	
0040513C	85C0	TEST EAX,EAX	
00405142	0F84 2D010000	JZ 0040526F	
00405148	53	PUSH EBX	
0040514C	57	PUSH EDI	
00405144	8D45 70	LEA EAX,[LOCAL.1]	
00405147	50	PUSH EAX	
00405148	8D45 E0	LEA EAX,[LOCAL.37]	
0040514B	50	PUSH EAX	
0040514C	33D8	XOR EBX,EBX	
0040514E	53	PUSH EBX	
0040514F	6A 08	PUSH 8	
00405151	53	PUSH EBX	
00405152	FF75 68	PUSH DWORD PTR SS:[LOCAL.3]	
00405155	BE 00000000	MOV ESI,0	
00405158	8975 70	MOV DWORD PTR SS:[LOCAL.1],ESI	
00405159	FF15 00104000	CALL DWORD PTR DS:[<ADUAPI32.CryptExportKey>]	
00405163	85C0	TEST EAX,EAX	
00405165	0F84 F4000000	JZ 0040526F	
00405168	56	PUSH ESI	
00405172	8B35 18104000	MOV ESI,DWORD PTR DS:[<ADUAPI32.CryptEncrypt>]	
00405175	8D45 70	LEA EAX,[LOCAL.1]	
00405176	50	PUSH EAX	
00405176	8D45 E0	LEA EAX,[LOCAL.37]	
00405179	50	PUSH EAX	
0040517A	53	PUSH EBX	
0040517B	6A 01	PUSH 1	
0040517D	01 00 00 00	PUSH 0	
0040517E	FF35 90694000	PUSH DWORD PTR DS:[406990]	
00405180	FFD6	CALL ESI	
00405186	FF75 7C	PUSH DWORD PTR SS:[ARG.1]	
00405189	FF15 F4104000	CALL DWORD PTR DS:[<KERNEL32.IsntUnicodeString>]	
0040518F	83E0 E0	AND EAX,EAX	
00405192	83C0 20	AND ECX,ECX	
00405195	50	PUSH EAX	
00405196	8945 6C	MOV DWORD PTR SS:[LOCAL.2],EAX	
[00401018]=75C6D05B (ADUAPI32.CryptEncrypt)			
ESI=75C6D05B (ADUAPI32.CryptEncrypt) (current registers)			
Address	Hex dump	ASCII	0012FE24
0012FE4C	00 02 00 00 00 66 00 00 20 00 00 00 0A E9 18 52	00000000	00000000
0012FE50	1C 52 2B 4A 00 03 2B 67 0E B8 4A 25 35 C5 30 20	00000001	00000000
0012FE56	17 52 0F 24 50 38 05 37 93 58 60 F2 91 8E 61 00	00000000	00000000
0012FE7C	37 D6 9A 76 62 CC 9A 76 9C 45 40 9C FE 12 00	00000000	00000000
0012FE8C	E0 FE 12 00 74 FE 12 00 91 8E 61 00 C4 FF 12 00	00000000	00000000
0012FE9C	F5 D5 A6 75 2B 6C 94 F8 FE FF FF FF 55 00 53 00	00000000	00000000
0012FEAC	01 00 00 00 00 C5 60 00 48 E7 5F 00 00 00 00	00000000	00000000
0012FEB0	01 00 00 00 00 E1 07 05 00 00 00 00 EC 2D EA C9	00000000	00000000
0012FEC0	F5 AE 23 34 F0 BC C3 A9 8A 50 60 00 10 00 00 00	00000000	00000000
0012FEDC	2C 00 00 00 00 00 00 00 5F 67 40 00 F8 8A 61 00	00000000	00000000
0012FEFC	00 00 00 00 00 00 00 00 94 FF 12 00 00 60 FD 7F	00000000	00000000
0012FEFF	03 84 61 00 01 06 00 00 F8 8A 61 00 00 00 00	00000000	00000000
0012FF0C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00000000	00000000
0012FF1C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00000000	00000000
0012FF2C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00000000	00000000
0012FF3C	00 00 00 00 00 00 00 00 00 00 00 00 02 00 00	00000000	00000000

Fig. 12 Another AES key is generated, exported, and encrypted using the embedded RSA key

00405195	50	PUSH EAX	
00405196	8945 6C	MOV DWORD PTR SS:[LOCAL.2],EAX	
00405199	8D45 6C	LEA EAX,[LOCAL.2]	
0040519C	53	PUSH EBX	
0040519D	FF75 7C	PUSH DWORD PTR SS:[ARG.1]	
004051A0	53	PUSH EBX	
004051A1	53	PUSH EBX	
004051A2	53	PUSH EBX	
004051A3	FF75 68	PUSH DWORD PTR SS:[LOCAL.3]	
004051A6	FFD6	CALL ESI	
004051A8	85C0	TEST EAX,EAX	
004051AB	0F84 AF000000	JZ 0040526F	
004051B0	8945 70	MOV EAX,DWORD PTR SS:[LOCAL.1]	
004051B3	8B40 6C	MOV ECX,DWORD PTR SS:[LOCAL.2]	
ESI=75C6D05B (ADUAPI32.CryptEncrypt)			
Address	Hex dump	ASCII	0012FE24
00618AF8	20 20 20 20 20 42 45 47 49 4E 20 52 53 41 20 50	-----BEGIN RSA P	00000000
00618B08	52 49 56 41 54 25 20 48 45 59 20 20 20 20 20 0D	RIVATE KEY-----	00000000
00618B18	0A 42 77 49 41 41 41 43 68 41 41 42 53 55 30 45	0BwIAARACKARBSU0E	00000000
00618B28	73 41 41 41 51 41 41 41 41 41 51 42 52 41 66 33	YAR0ARARARARAR3F	00000000
00618B38	40 47 71 68 71 68 63 68 61 4C 44 39 46 46 65 54	JS9P0zuw01IqLwT9	00000000
00618B48	67 68 71 39 30 53 68 61 4C 44 39 46 46 65 54 39	ghg80SkalD9FFnT9	00000000
00618B58	59 00 0A 20 46 6C 63 63 59 28 44 68 35 40 65 48	VJ00+F1cV+TtStHeH0	00000000
00618B68	69 6C 64 0B 37 40 48 30 47 39 54 31 55 43 41 67	l10T07L099T1WcRg	00000000
00618B78	60 41 64 68 55 62 40 73 32 33 60 40 73 68 4F 30	mAdkUbuJ+TJ0etH6	00000000
00618B88	56 2F 73 6E 52 31 30 73 32 33 60 40 73 68 4F 30	U/snR10s23NnSk00	00000000
00618B98	71 54 67 00 0A 37 73 79 53 39 58 49 33 40 4F 41	qTgR0d7yS9X13L0A	00000000
00618BA8	4E 7A 6F 70 45 37 48 68 68 62 67 37 66 57 78	HzoeE7KJkb87fWk	00000000
00618BB8	63 31 69 76 62 70 75 7F 69 45 31 37 40 74 63	9c1lv0RfhuuE17Jd1	00000000
00618BC8	30 73 48 40 42 48 73 39 30 45 6F 48 74 75 56	0s/K0Bh590EkTuU	00000000
00618BD8	43 7A 38 65 41 00 0A 6F 68 51 51 30 52 65 35 48	Cz8eRj0k000Re5H	00000000
00618BE8	6F 45 73 73 53 75 56 50 43 31 50 67 5A 4B 64 67	0eCySvknC1PgKdg	00000000
00618BF8	2F 38 61 34 7A 75 28 73 45 37 40 74 43 4F	F-SAkgzuvsEJz00	00000000
00618C08	77 43 6A 72 5A 56 53 43 4F 6A 74 67 2F 72 4F	wsJr26SC0JtgAr0	00000000
00618C18	76 66 2A 44 74 47 4A 00 0A 33 4F 42 37 42 2B	VF/DtGJJB30B7+h	00000000
00618C28	54 2F 22 23 0E 2E 30 2A 0E 39 25 25 0E 31 7E	h4c30rE30d00	00000000

Fig. 13 The AES key, which was generated, is used to encrypt a private RSA key



Fig. 14 Another AES256 key is generated

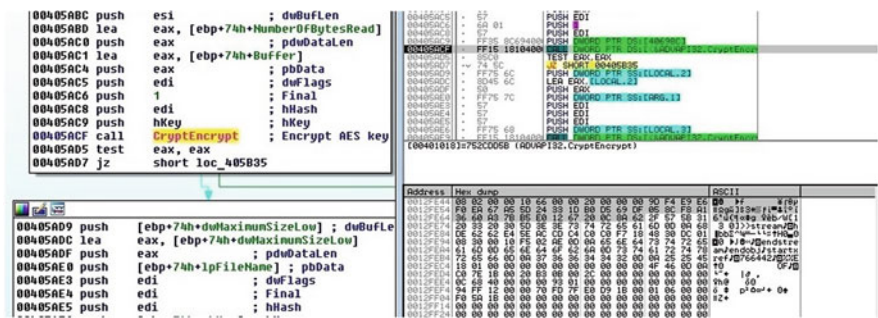


Fig. 15 The AES key is encrypted using RSA key

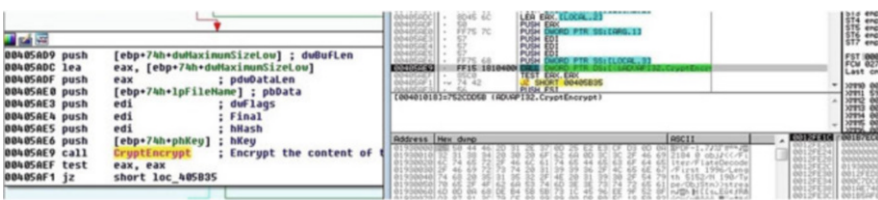



Fig. 16 The file is encrypted using the AES key

2 DMA Locker Ransomware

Name: dma.exe
md5: FDECD41824E51F79DE6A25CDF62A04B5
Type: encrypting ransomware

In Fig. 17 a report by VirusTotal, which shows that the malware is known to most vendors, is presented.

According to Fig. 18, the ransomware isn't packed; if this is obfuscated, it is then necessary to reveal it. As shown in Fig. 19, the malware moves the original file to C:\ProgramData and renames the file svchost.exe (the author of ransomware is trying to hide the malicious purposes, in order to look like the Service Host Process svchost.exe). Once the file is copied, the malware




SHA256: a6443ba599a43d558b7f0f8d56937fa3b04d615e183aa23f289a8bf4d745445

File name: 38527d20338fb35717b349176b976610465d368123c083fb88115e982b367918....

Detection ratio: 40 / 57

Analysis date: 2017-05-30 10:33:37 UTC (3 days, 5 hours ago)



Antivirus	Result	Update
AegisLab	Troj.W32.Gen.mCYi	20170530
AhnLab-V3	Malware/Win32.Generic.C1465743	20170530
Antiy-AVL	Trojan[Ransom]/Win32.Agent	20170530
Arcabit	Trojan.Zusy.D2CF5E	20170530
Avast	Win32:Malware-gen	20170530
AVG	Win32/DH(gmBI7)	20170530
Avira (no cloud)	TR/Ransom.psxmn	20170530
AVware	Trojan.Win32.GenericIBT	20170530
Baidu	Win32.Trojan.WisdomEyes.16070401.9500.9912	20170527
BitDefender	Gen.Variant.Zusy.184158	20170530
CAT-QuickHeal	Ransomware.DMALocker.A5	20170530
ClamAV	Win.Trojan.DMALocker-1	20170530
Comodo	TrojWare.Win32.Ransom.DMALocker.A	20170530
Cyren	W32/DMALocker.A.gen/Eldorado	20170530
DrWeb	Trojan.Encoder.4199	20170530
Emsisoft	Gen.Variant.Zusy.184158 (B)	20170530

Fig. 17 VirusTotal report DMA Locker

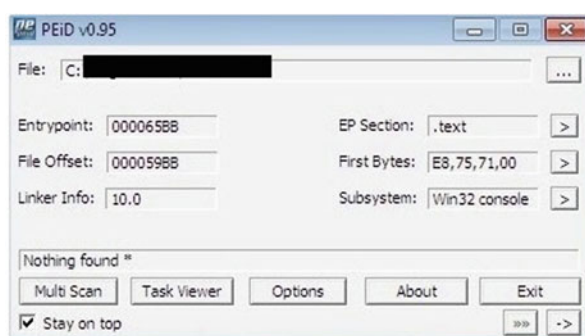


Fig. 18 PEiD report DMA Locker



Fig. 19 The malware moves the original file to another location

starts `svchost.exe` process (which obviously is a copy of the original process) and then exits. As shown in Fig. 20, the function `CreateProcessW` is used. The original process creates two keys in registry for persistence: `HKLM\Software\Microsoft\Windows\CurrentVersion\Run\Windows Firewall` which has the value `C:\ProgramData\svchost.exe` and `HKLM\Software\Microsoft\Windows\CurrentVersion\Run\Windows Update`, which has the value `notepad C:\ProgramData\cryptinfo.txt` (at every reboot the ransom note is shown). The DMA Locker deletes backups and shadow copies, using the native Windows utility `VSSAdmin`, as shown in Fig. 21. A `start.txt` file is created to show that the encryption has begun (and there is no need to restart it again). Logical disks and network shares are attacked, and checks against the Floppy and CD using `QueryDosDeviceA` (Floppy and CD are skipped) are made, as shown in Fig. 22. The sample uses a hardcoded public RSA key, stored in a form of BLOB, as shown in Fig. 23. Some directories are excluded from the encryption process; this entire list is in Fig. 24. A list of skipped extensions is presented in Fig. 25. A unique AES256 key is generated for every file using the API `CryptGenRandom`, as shown in Fig. 26. The AES key is used to encrypt 16-byte-long data with AES ECB mode, as shown in Fig. 27. Once used, the AES key is encrypted using the hardcoded RSA key (Fig. 28). The structure of the encrypted file is the prefix which is added, the encrypted AES key, and the encrypted original content (Fig. 29). Once the encryption process is complete, a message alert is presented (Fig. 30). The malware may be fooled in order to avoid the encryption through the creation of the files `start.txt` and `cryptinfo.txt` in `ProgramData` directory. If these two files are present, the encryption cannot start and only the ransom message is displayed. However, if the algorithms, which are used in the encryption process, are consistent, the decryption without the RSA private key which is kept secret will not be possible.

Address	Hex dump	ASCII
0018F9F8	06 02 00 00 00 A4 00 00 52 53 41 31 00 04 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0018FA08	01 00 01 00 A3 4A 8F 85 FA 88 7F FD A1 40 32 FF	0 0 0 JA 8 8 5 IM2
0018FA18	EE 88 5D 14 7E 5C 1F 92 3D F9 16 B6 3D 61 9D 63	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FA28	41 E8 4A 61 F2 79 48 DA A2 96 31 39 E9 BE BE 80	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FA38	7C 23 A3 20 FD 73 64 7E 83 00 7C C1 A2 58 91 C6	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FA48	52 6A 86 FE EB B3 98 6C 4B FB 8A 5E F2 60 E0 92	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FA58	DB 52 02 A0 2C 18 8D 6C 6B 4C 28 FA 74 28 09 08	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FA68	0A AF C4 29 8D 1F 11 AA A4 73 0C 1F 08 58 3A AF	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FA78	9B 24 51 2C C8 5D A8 A5 9A 58 84 B4 AF 80 AE 4D	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FA88	C5 ED 0D B6 0D 73 00 00 A4 FA 18 00 4F 03 74 6A	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FA98	0F D7 3C 00 E4 FA 18 00 65 65 3C 00 01 00 00 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FAA8	78 18 55 00 B8 18 55 00 37 00 74 6A 00 00 00 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FAB8	00 00 00 00 80 80 FD 7F D0 FA 18 00 00 00 00 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FAC8	00 00 00 00 80 FA 18 00 69 DF 66 B7 20 FB 18 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FAD8	20 88 3C 00 63 94 51 6A 00 00 00 00 F0 FA 18 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FAE8	74 11 43 77 00 80 FD 7F 30 FB 18 00 F5 B3 51 77	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FAF8	00 80 FD 7F 54 54 69 77 00 00 00 00 00 00 00 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FB08	00 80 FD 7F 00 00 00 00 00 00 00 00 00 00 00 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FB18	FC FA 18 00 00 00 00 00 FF FF FF FF 40 D7 4D 77	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FB28	FC AA 20 00 00 00 00 00 48 FB 18 00 C8 B3 51 77	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FB38	BB 65 3C 00 80 FD 7F 00 00 00 00 00 00 00 00 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FB48	00 00 00 00 00 00 00 00 BB 65 3C 00 80 FD 7F	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FB58	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FB68	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FB78	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0018FB88	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Fig. 23 Hardcoded RSA key

```

003C2880 push    ebp
003C2881 mov     ebp, esp
003C2883 sub     esp, 2Ch
003C2886 push    esi
003C2887 mov     [ebp+var_2C], offset aWindows ; "\\Windows\\"
003C288E mov     [ebp+var_28], offset aWindows_0 ; "\\WINDOWS\\"
003C2895 mov     [ebp+var_24], offset aProgramFiles ; "\\Program Files\\"
003C289C mov     [ebp+var_20], offset aProgramFilesX86 ; "\\Program Files (x86)\\"
003C28A3 mov     [ebp+var_1C], offset aGames ; "Games"
003C28AA mov     [ebp+var_18], offset aTemp ; "\\Temp"
003C28B1 mov     [ebp+var_14], offset aSamplePictures ; "\\Sample Pictures"
003C28B8 mov     [ebp+var_10], offset aSampleMusic ; "\\Sample Music"
003C28BF mov     [ebp+var_C], offset aCache ; "\\cache"
003C28C6 mov     [ebp+var_8], offset aCache_0 ; "\\Cache"
003C28CD xor     esi, esi
003C28CF nop

```

Fig. 24 The directories which are excluded from the encryption

```

003C2907 mov     [ebp+var_30], offset a_exe ; ".exe"
003C290E mov     [ebp+var_2C], offset a_msi ; ".msi"
003C2915 mov     [ebp+var_28], offset a_dll ; ".dll"
003C291C mov     [ebp+var_24], offset a_pif ; ".pif"
003C2923 mov     [ebp+var_20], offset a_scr ; ".scr"
003C292A mov     [ebp+var_1C], offset a_sys ; ".sys"
003C2931 mov     [ebp+var_18], offset a_msp ; ".msp"
003C2938 mov     [ebp+var_14], offset a_com ; ".com"
003C293F mov     [ebp+var_10], offset a_lnk ; ".lnk"
003C2946 mov     [ebp+var_C], offset a_hta ; ".hta"
003C294D mov     [ebp+var_8], offset a_cpl ; ".cpl"
003C2954 mov     [ebp+var_4], offset a_msc ; ".msc"

```

Fig. 25 Skipped extensions

003C484B	• 51	PUSH ECX
003C484C	• 6A 20	PUSH
003C484E	• 52	PUSH EDX
003C484F	• 8945 DC	MOV DWORD PTR SS:[LOCAL.9],EAX
003C4852	• 8945 E0	MOV DWORD PTR SS:[LOCAL.8],EAX
003C4855	• 8945 E4	MOV DWORD PTR SS:[LOCAL.7],EAX
003C4858	• 8945 E8	MOV DWORD PTR SS:[LOCAL.6],EAX
003C485B	• 8945 EC	MOV DWORD PTR SS:[LOCAL.5],EAX
003C485E	• 8945 F0	MOV DWORD PTR SS:[LOCAL.4],EAX
003C4861	• 8945 F4	MOV DWORD PTR SS:[LOCAL.3],EAX
003C4864	• 8945 F8	MOV DWORD PTR SS:[LOCAL.2],EAX
003C486D	• FF15 04403D00	JMP DWORD PTR DS:[I:&ADUAPI32.CryptGenRandom]
003C486F	• 6A 00	PUSH
003C4871	• 85C0	TEST EAX,EAX
003C4873	• 75 1B	JNZ SHORT 003C488E
003C4876	• 8B45 D8	MOV EAX,DWORD PTR SS:[LOCAL.10]
003C4877	• 50	PUSH EAX
003C487D	• FF15 04403D00	JMP DWORD PTR DS:[I:&ADUAPI32.CryptReleaseContext]
003C487F	• B0 01	MOV AL,1
003C4880	• 5F	POP EDI
003C4883	• 8B4D FC	MOV ECX,DWORD PTR SS:[LOCAL.1]
003C4885	• 33CD	XOR ECX,EBP
003C4886	• E8 2C020000	CALL 003C4AB6
003C4887	• 8B4D FC	MOV ECX,DWORD PTR SS:[LOCAL.1]

[003D4004]=7593E5EE (ADUAPI32.CryptGenRandom)

Address	Hex dump	ASCII
0018EC2C	93 43 81 05 BE 35 46 7C 40 7F 4D 59 2D FA FB CE	6CuP5F1@0Nv- jf
0018EC3C	EF 7C 25 BD 99 84 7D 35 CA BB 5A 39 A3 00 F0 14	n1220a)5mZ9d 89
0018EC4C	83 16 74 6A D0 EE 18 00 BF 20 3C 00 38 F0 18 00	ä.tj4et 7 < 8=
0018EC5C	70 F8 18 00 2E 00 00 00 04 00 00 00 24 EF 18 00	p0† ♦ \$n†
0018EC6C	FF FF FF FF B0 EE 18 00 00 00 00 00 28 F0 18 00	set (set
0018EC7C	20 00 00 00 D8 39 A1 59 74 9F D2 01 D8 89 A1 59	teivtft0teiv
0018EC8C	74 9F D2 01 38 EB A3 59 74 9F D2 01 00 00 00 00	tft08\$uVtft0
0018EC9C	00 04 00 00 5F 00 45 00 4E 00 41 00 2E 00 72 00	♦ - E N A . r
0018ECAC	6E 00 64 00 00 00 63 00 6C 00 65 00 2E 00 42 00	n d c l e . B
0018ECBC	69 00 6E 00 00 00 00 00 00 00 00 00 00 00 00	i n
0018ECC0	F8 F1 13 00 99 5E 43 77 01 00 00 00 00 00 00	0zt ü^Cw0
0018ECD0	01 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0
0018ECE0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0018ECF0	E0 5F 43 77 58 00 00 00 FC 5F 43 77 00 00 00 00	α_CwX n_Cw
0018ED00	D4 F0 18 00 00 00 05 00 34 00 00 00 00 00 00 00	=st ± 4

Fig. 26 A unique AES key is generated

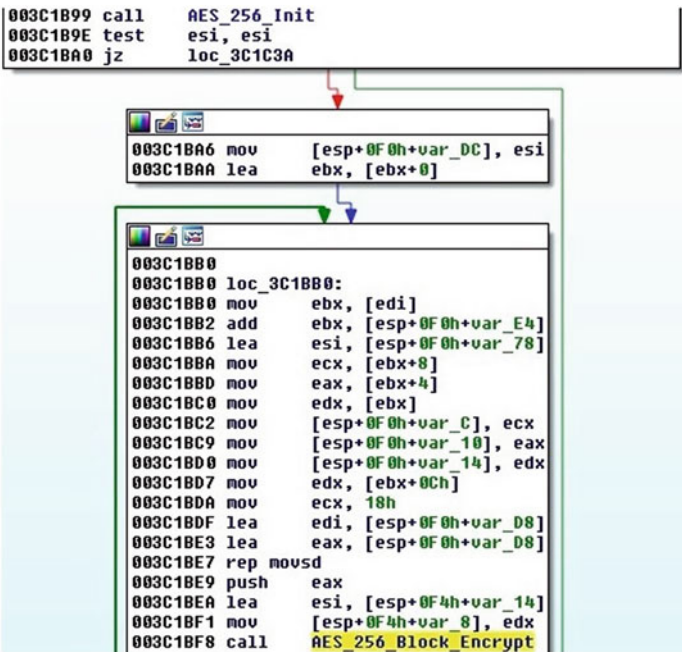


Fig. 27 The data is split in chunks of 16 bytes and encrypted

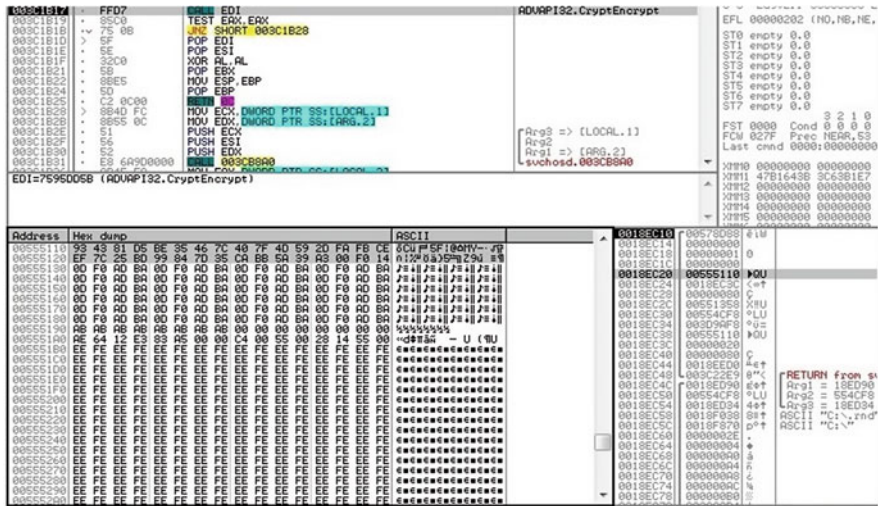


Fig. 28 The AES key is encrypted using the hardcoded RSA key

```

003C22D5
003C22D5 loc_3C22D5:
003C22D5 lea     ecx, [ebp+var_19C]
003C22D8 push    ecx           ; int
003C22DC push    esi           ; void *
003C22DD lea     edx, [ebp+var_140]
003C22E3 push    edx           ; int
003C22E4 call    EncryptAESKeyWithRSA
003C22E9 push    ebx           ; FILE *
003C22EA push    0Bh           ; size_t
003C22EC push    1             ; size_t
003C22EE push    offset aEncrypt ; "!Encrypt!##"
003C22F3 call    _fwrite    ; Write prefix
003C22F8 mov     eax, [ebp+var_19C]
003C22FE push    ebx           ; FILE *
003C22FF push    eax           ; size_t
003C2300 push    1             ; size_t
003C2302 push    esi           ; void *
003C2303 call    _fwrite    ; Write Encrypted Content
003C2308 mov     ecx, [ebp+var_18]
003C230B mov     edx, [ebp+var_1A4]
003C2311 push    ebx           ; FILE *
003C2312 push    ecx           ; size_t
003C2313 push    1             ; size_t
003C2315 push    edx           ; void *
003C2316 call    _fwrite

```

Fig. 29 A prefix is added to each file



All your personal files are LOCKED!

WHAT'S HAPPENED?

- * All your important files (including hard disks, network disks, flash, USB) are encrypted.
- * All of files are locked with asymmetric algorithm using AES-256 and then RSA-2048 cipher.
- * You are not possible to unlock your files because all your backups are removed.
- * Only way to unlock your files is to pay us 1500 GBP in Bitcoin currency (1.0 BTC).
After payment we will send you decryption key automatically, which allow you to unlock files .

HOW TO PAY US AND UNLOCK YOUR FILES?

1. Please read the steps carefully.
2. To pay us, you have to use **Bitcoin** currency. You can easily buy Bitcoins at following sites:
 - * <https://www.coinfloor.co.uk/>
 - * <https://localbitcoins.com/>
 - * <https://www.coinbase.com/>
3. If you already have Bitcoins, pay us 1.0 BTC (1500 GBP) on following Bitcoin address:

1EEHF6uCK2UNtbwX1yAzZ74wNudApYWQm
4. After payment, necessarily contact with us to get your decryption key:
data0001@tuta.io In mail title write your unique ID:

01:07:91:50:32:25:30:07
5. We will automatically send you decryption key file after bitcoin transfer .
When you receive your decryption key file, press "OPEN" button and choose your received decryption key file.
Then, press the "UNLOCK FILES" button and it will start unlocking all your files.

* You have 96 hours to pay us!

* After this time ransom will grow to 200 percent

* Ransom grow time:
7/6/2017 13:29

Fig. 30 DMA Locker Message

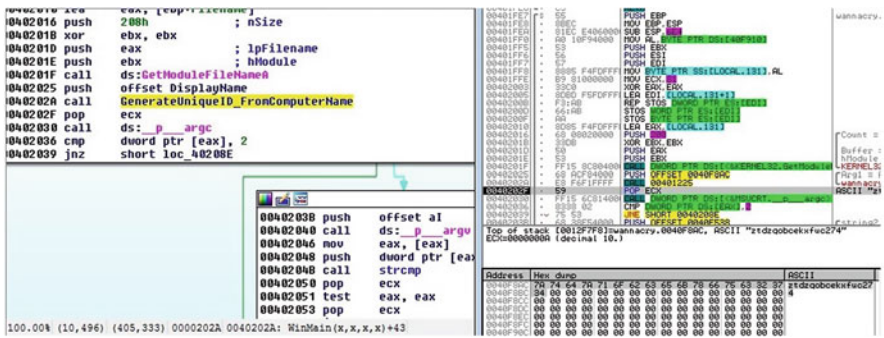


Fig. 31 A unique identifier is generated for every victim



Fig. 32 Ransom notes

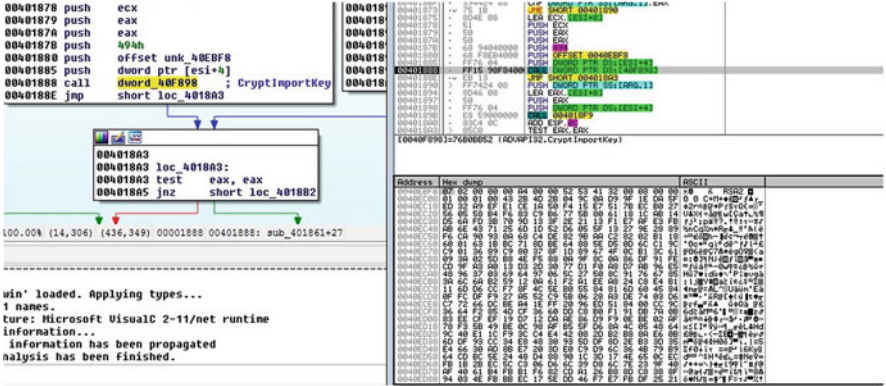


Fig. 33 Private RSA key is being imported



Fig. 34 The encrypted key is decrypted using private RSA key

10003B25	52	PUSH EDX	
10003B26	50	PUSH EAX	
10003B27	50	PUSH EAX	
10003B28	8B46 04	MOV EAX, DWORD PTR DS:[ESI+4]	
10003B2B	68 14010000	PUSH 14	
10003B30	68 40CF0010	PUSH 1000CF40	
10003B35	50	PUSH EAX	
10003B36	FF15 40D90010	JMP DWORD PTR DS:[1000D940]	
10003B3C	85C0	TEST EAX, EAX	
10003B3E	74 46	JE SHORT 10003B86	
10003B40	8B4E 04	MOV ECX, DWORD PTR DS:[ESI+4]	
10003B43	8D7E 08	LEA EDI, [ESI+8]	
10003B46	57	PUSH EDI	
10003B47	51	PUSH ECX	
10003B48	E8 03000000	JMP 10004350	
10003B4D	53C4 08	ADD ESP, 8	
10003B50	85C0	TEST EAX, EAX	
10003B52	74 32	JE SHORT 10003B86	
10003B54	8B17	MOV EDI, DWORD PTR DS:[EDI]	
[1000D940]=76B0B852 (ADVAPI32.CryptImportKey)			

Address	Hex dump	ASCII
1000CF40	06 02 00 00 00 A4 00 00 52 53 41 31 00 08 00 00	*0 n RSA1
1000CF50	01 00 01 00 75 97 4C 38 84 46 DE 2C 2A F4 95 A8	0 0 uLL; aF, *r0d
1000CF60	50 C0 CD 60 DA D7 D4 92 1E 13 82 34 6A 70 8D 8F] =nrt *E!!e4jplA
1000CF70	7C F7 04 92 55 7F F1 A2 27 B2 9E 41 AC 90 80 91	! :oEU0:0' HhA4eQe
1000CF80	18 93 C2 B1 7B AD 2B F3 FF AF 0B 2B 51 BE 1D A3	707f(i+s)#*Q*W
1000CF90	27 E3 A7 57 08 5A BE C1 1D F6 04 F8 1C BE 5B B1	'T9[2#-#*L= [
1000CFA0	67 FB E4 C8 DA 75 00 70 B1 17 70 24 6C 09 63 74	grZ=ru p[3p5 loct
1000CFB0	AC 4B 0A 10 71 AE 7F AE 65 B8 C5 86 79 C5 7E 9F	%k0=q<0«e7+3y+ f
1000CFC0	98 60 4C 52 B9 29 62 CB 23 29 ED 31 91 74 7B 7B	5'LRQ)b[ra]0lat((
1000CFD0	08 26 18 F2 7D 67 BF DA 7A 40 DA F2 61 4D 94 A5	0%+2)gr rz0r2aM0N
1000CFE0	7D AD 59 68 AD 9E A3 3A 39 C6 5B 6E 9F D2 BB 36)iVkiHu:9F[nfm]6
1000CFF0	B5 F5 D2 65 F5 2C 30 D8 C1 17 BD AF 28 00 96 20	qJreJ,07+3*))(u
1000D000	46 A7 2D 62 03 0C D7 D0 75 A0 0B 07 EA D4 1F CA	F0-b09[+u00.0 *w
1000D010	E8 D9 4E D8 38 F2 26 75 CB 12 A6 88 70 9B E1 EA	3r[82&u7f3ep<B0
1000D020	32 DC F8 71 72 50 41 E6 17 81 68 27 42 8E DF E5	2m°qrPAPu0h'BA#o
1000D030	DE A1 72 D9 38 FB E5 90 30 11 69 92 CD 60 2B E2	llr';r0#04LE=*+f
1000D040	D5 46 3C 23 CF 9D 30 4A F7 AD B9 FB 0F 91 FE 2E	rF<(=0J0 i]Jw00.
1000D050	BE 18 F1 CE 06 02 00 00 A4 00 00 52 53 41 31	*+r*0 n RSA1
1000D060	00 03 00 00 01 00 01 00 43 2B 4D 2B 04 9C 0A D9] 0 0 C+H+00
1000D070	9F 1E DA 5F ED 32 A9 EF E1 CE 1A 50 F4 15 E7 E1	fA_r02r087+P'S:Q
1000D080	7B EC 80 27 56 05 5B 84 F6 83 C9 B6 77 5B 80 61	<007U4>H+3[ll u[C0
1000D090	18 1C AB 14 05 6A FD 3B 70 9D 13 3F 2E 21 13 F1	tL30fj2;p0!!?.!H:
1000D0A0	E7 AF E3 FB AB 6E 43 71 25 6D 10 52 D6 05 5F 13	r))Tf4nCo%h#Rr4.!!
1000D0B0	27 9E 28 89 F6 CA 90 93 0A 68 C4 DE 82 9B AA C2	'h(e+000h- eC^T
1000D0C0	82 02 B1 18 60 01 63 1B BC 71 8D BE 64 88 5E D5	e00f'0c+qi=d0^f
1000D0D0	0D 6C C1 9C C9 01 36 89 C9 80 37 8F 1D 89 67 4F	Jl+Lff060f7A#0g0

Fig. 35 Public RSA key is being imported

```

100040EB 6A 00 PUSH
100040ED 68 80000000 PUSH
100040F2 6A 02 PUSH
100040F4 6A 00 PUSH
100040F6 6A 00 PUSH
100040F8 68 80000040 PUSH
100040FD 8B45 14 MOV EAX, DWORD PTR SS:[EBP+14]
10004100 5B PUSH EAX
10004101 FF15 84700010 JMP DWORD PTR DS:[10007084]
10004107 8945 D8 MOV DWORD PTR SS:[EBP-28], EAX
1000410A 83F8 FF CMP EAX, 0
1000410D 75 07 JNE SHORT 10004116
1000410F 50 PUSH EAX
10004110 8D4D F0 LEA ECX, [EBP-10]
10004113 51 PUSH ECX
10004114 EB BA JMP SHORT 100040D0
10004116 6A 00 PUSH
10004118 8D55 E4 LEA EDI, [EBP-1C]
1000411B 52 PUSH EDI
1000411C 8B4D DC MOV ECX, DWORD PTR SS:[EBP-24]
1000411F 51 PUSH ECX
10004120 56 PUSH ESI

```

[10007084]=774528FC (kernel32.CreateFileA)

Address	Hex dump	ASCII
002248A8	06 02 00 00 00 A4 00 00 52 53 41 31 00 08 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002248B8	01 00 01 00 9F CF 2F 40 3A 54 EA B6 DE C5 3B 30	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002248C8	60 87 C4 B4 A4 09 A6 64 E9 AB 33 E0 EF FE FD D1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002248D8	4B 27 0C 45 DC 1E 6A 0C 9A D4 A9 C5 2F D8 28 C6	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002248E8	52 BF 28 FB 5A 3C 30 43 F3 71 9A 38 6A C2 19 1C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002248F8	F6 5D CD 7F 41 4F 6F 00 FD 38 70 C5 94 66 62 37	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224908	A8 F8 7B 1A 34 91 6B 66 C5 1C 00 E2 F2 7A F1 3D	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224918	F8 D3 1D CA 25 80 1B 70 F0 89 B5 9C 66 25 89 16	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224928	D3 9F 72 FB 10 73 E5 C1 72 99 72 AB 70 3F FF D3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224938	0F C1 0A D0 7A 92 DC 5B 48 FF 63 87 0D 00 8C 41	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224948	A3 4A 6B B1 ED 22 EF 49 8B A0 D0 09 BA 14 49 D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224958	0B F7 6B 40 2F 65 74 D2 0B 3B EF 12 4B 2E 1A E7	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224968	41 80 D4 57 D7 12 70 DE 0E 82 59 76 62 E2 E9 1C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224978	07 81 E5 36 36 84 57 BA 72 72 49 19 AA 75 A5 0C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224988	69 1A 6E 12 2B 43 19 D2 A9 16 BC 8B 25 E0 0F 8D	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224998	00 E8 FD 97 65 D4 9C 77 AA 50 6C 29 EA 1F D1 C4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002249A8	BA 48 5C C4 26 57 73 13 00 8C D6 6A B2 D8 6C 8C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002249B8	8B 0A 39 CF AB AB AB AB AB AB AB AB EE FE EE FE	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002249C8	00 00 00 00 00 00 00 00 70 BC B9 0E 15 02 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002249D8	58 BC 21 00 C4 00 20 00 7D BC BA 00 31 02 00 18	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002249E8	03 00 00 00 01 00 00 00 19 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002249F8	AD BC 87 EF 21 00 00 00 7B D5 A6 76 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224A08	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224A18	7B D5 A6 76 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224A28	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00224A38	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Fig. 36 The public key is exported and saved to 00000000.pkx

```

00004289 59      MUSH EBX
1000428A F3:R4   REP MOUS BYTE PTR ES:[EDI], BYTE PTR DS:
1000428B 808C24 2C1000 LEA ECX, [ESP+102C]
1000428C 51      PUSH ECX
1000428D 6A 00   PUSH 0
1000428E 6A 01   PUSH 1
1000428F 6A 00   PUSH 0
10004290 52      PUSH EDX
10004291 FF15 48D90010 XOR EDWORD PTR DS:[10000948]
10004292 85C0    TEST EAX, EAX
10004293 74 57   JE SHORT 1000431B
10004294 8B4C24 10 MOV ECX, DWORD PTR SS:[ESP+10]
10004295 8B4424 18 MOV EAX, DWORD PTR SS:[ESP+18]
10004296 8BD1    MOV EDI, ECX
10004297 8B4C24 241000 LEA EDI, [ESP+1024]
10004298 8BF8    MOV EDI, EAX
10004299 C1E9 02 SHR ECX, 2
100042DA F3:R5   REP MOUS DWORD PTR ES:[EDI], DWORD PTR D
100042DB 8BCA    MOV ECX, EDX
100042DC AND ECX, 00000000
100042DD REP MOUS BYTE PTR ES:[EDI], BYTE PTR DS:
100042DE F3:R4   REP MOUS DWORD PTR SS:[ESP+10]
100042DF 8B4424 18 MOV EAX, DWORD PTR SS:[ESP+18]
100042E0 8B4C24 14 MOV ECX, DWORD PTR SS:[ESP+14]
[10000948]=76820D5B (ADVAPI32.CryptEncrypt)

```

[illegible]

Fig. 37 The private key is encrypted using hardcoded RSA key

10004758	loc_10004758:	lea	eax, [esp+8+NumberOfBytesWritten]	10004758	59	POP ESI	
1000475B	push	0	lpOverlapped	10004759	5E	POP ECX	
1000475F	push	eax	lpNumberOfBytesWritten	1000475A	C3	LEA ECX, [ESP+8]	
10004761	push	88h	lpBuffer	1000475B	6A 80	PUSH EBX	
10004762	push	offset pbBuffer	lpBuffer	1000475C	68 00	PUSH EAX	
10004766	push	esi	hFile	1000475D	67 80000000	PUSH EDI	
10004767	call	esi	[esp+1Ch+NumberOfBytesWritten], 0	1000475E	67 80000000	PUSH EDI	
1000476B	call	esi	WriteFile	1000475F	67 80000000	PUSH EDI	
10004775	call	esi	CloseHandle	10004760	67 80000000	PUSH EDI	
1000477C	call	esi	hObject	10004761	67 80000000	PUSH EDI	
10004782	mov	eax, 88h		10004762	67 80000000	PUSH EDI	

Fig. 38 Firstly, the 8 generated bytes and 128 zero bytes are written to the file

10003AEB	50	PUSH EAX
10003AEC	53	PUSH EBX
10003AED	53	PUSH EBX
10003AEE	68 14010000	PUSH 14010000
10003AF3	68 54000010	PUSH 54000010
10003AF8	51	PUSH ECX
10003AF9	FF15 40D90010	CALL DWORD PTR DS:[1000D940]
10003AFF	85C0	TEST EAX,EAX
10003B01	0F85 9C000000	JNE 10003BA3
10003B07	8BCE	MOV ECX,ESI
10003B09	E8 A2000000	JMP 10003BB0
10003B0E	5F	POP EDI
10003B0F	5E	POP ESI
10003B10	33C0	XOR EAX,EAX
10003B12	5B	POP EBX
10003B13	C2 0800	RETN 8
10003B16	53	PUSH EBX
10003B17	8BCE	MOV ECX,ESI
10003B19	E8 F2000000	JMP 10003C00

[1000D940]=752ABB52 (ADVAPI32.CryptImportKey)

Address	Hex dump	ASCII
1000D054	06 02 00 00 00 A4 00 00 52 53 41 31 00 08 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D064	01 00 01 00 43 2B 4D 28 04 9C 0A D9 9F 1E DA 5F	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D074	ED 32 A9 EF E1 CE 1A 50 F4 15 E7 51 7B EC B0 27	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D084	56 05 58 B4 F6 83 C9 B6 77 5B 80 61 18 1C AB 14	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D094	D5 6A FD 3B 70 9D 13 3F 2E 21 13 F1 E7 AF E3 FB	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D0A4	AB 6E 43 71 25 6D 10 52 D6 05 5F 13 27 9E 28 89	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D0B4	F6 CA 90 93 0A 68 C4 DE 82 9B AA C2 82 02 B1 18	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D0C4	C0 01 63 1B BC 71 8D BE 64 88 5E D5 0D 6C C1 9C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D0D4	C9 01 36 89 C9 80 37 8F 1D 89 67 4F 0C B1 3C 61	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D0E4	09 3A 02 5D B8 4E F5 88 0A 9F 8C 0A 86 DF 91 FE	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D0F4	CD 9F A3 A0 13 D3 2D 30 77 D1 F0 A8 D7 AB 96 E5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D104	48 96 37 03 69 64 97 06 5C 27 50 8C 91 76 67 85	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D114	3A 6C 6A B2 59 12 0A 61 F2 A1 EE A8 24 C8 E4 B1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D124	11 6D D6 CC F7 8F 4C 5E B0 55 84 81 6D 60 45 84	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D134	0F FC DF F9 27 A5 52 C9 5B 06 28 A3 DE 74 03 D6	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D144	C7 72 66 DC BE A4 1E FF 20 96 ED 51 84 00 CC 9C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D154	36 64 F2 85 4D CF 36 60 D0 C8 B0 F1 91 DB 7A 0B	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D164	83 EE CF EF 4D 69 63 72 6F 73 6F 66 74 20 45 84	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D174	68 61 6E 63 65 64 20 52 53 41 20 61 6E 64 20 41	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D184	45 53 20 43 72 79 70 74 6F 67 72 61 70 68 69 63	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D194	20 50 72 6F 76 69 64 65 72 00 00 00 54 45 53 54	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D1A4	44 41 54 41 00 00 00 00 43 72 79 70 74 47 65 6E	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D1B4	4B 65 79 00 43 72 79 70 74 44 65 63 72 79 70 74	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D1C4	00 00 00 00 43 72 79 70 74 45 6E 63 72 79 70 74	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1000D1D4	00 00 00 00 43 72 79 70 74 44 65 73 74 72 6F 79	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Fig. 39 Another RSA key is being imported

```

Becho off
echo SET ow = wscript.CreateObject("wscript.shell")> m.vbs
echo SET om = ow.CreateShortcut("C:\[redacted]\@WanaDecryptor@.exe.lnk")>> m.vbs
echo om.TargetPath = "C:\[redacted]\@WanaDecryptor@.exe">> m.vbs
echo om.Save>> m.vbs
csript.exe //nologo m.vbs
del m.vbs

del /a %0

```

Fig. 40 The malware creates a LNK which points to @WanaDecryptor@.exe

.der.pfx.key.crt.csr.p12.pem.odt.ott.sxw.stw.uot.3ds.max.3dm.ods.ots.sxc
.stc.dif.slk.wb2.odp.otp.sxd.std.uop.otg.sxm.mml.lay.lay6.asc.sqlite3
.sqlitedb.sql.accdb.mdb.db.dbf.odb.frm.myd.myi.ibd.mdf.lbf.sln.suo.cs
.c.cpp.pas.h.asm.js.cmd.bat.ps1.vbs.vb.pl.dip.dch.sch.brd.jsp.php.asp.rb
.java.jar.class.sh.mp3.wav.swf.flv.wmv.mpg.vob.mpeg.asf.avi.mov.mp4.3gp.mkv
.3g2.flv.wma.mid.m3u.m4u.djvu.svg.ai.psd.nef.tiff.tif.cgm.raw.gif.png.bmp
.vcd.iso.backup.zip.rar.7z.gz.tgz.tar.bak.tbk.bz2.PAQ.ARC.aes.gpg.vmx.vmdk
.vdi.sldm.sldx.sti.sxi.602.hwp.edb.potm.potx.ppam.ppsx.ppsm.pps.pot.pptm.xlsm
.xltx.xlcl.xlsm.xltx.xlwb.xlsm.dotm.dot.docm.docb.jpg.jpeg.snt.onetoc2
.dwg.pdf.wk1.wks.123.rtf.csv.txt.vsd.vsd.eml.msg.ost.pst.pptx.ppt.xlsm.xls.docx
.doc

Fig. 41 Targeted extensions by malware

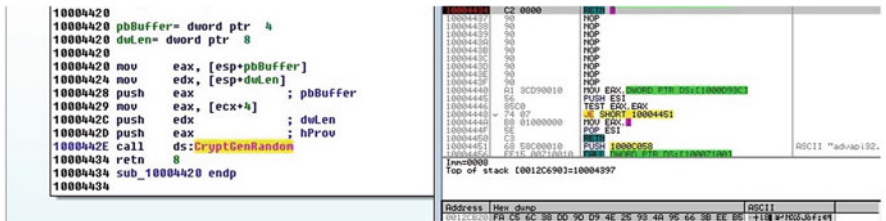


Fig. 42 A new AES key is generated for every file

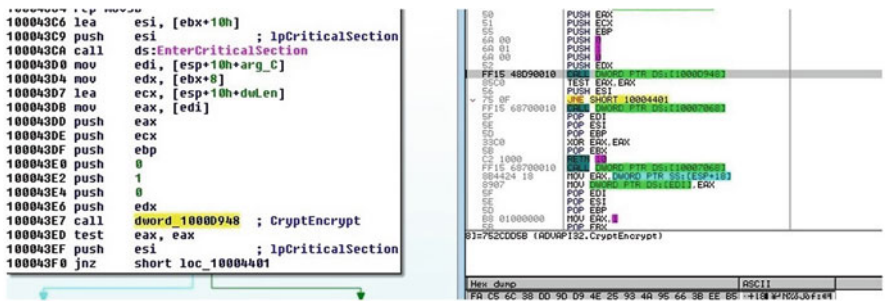


Fig. 43 The AES key is encrypted using RSA key

```

100058D3 push    0                ; lpExitCode
100058D5 push    0                ; dwMilliseconds
100058D7 push    offset aTaskkill_exeF1 ; "taskkill.exe /f /im Microsoft.Exchange.".
100058DC call     sub_10001080
100058E1 push    0                ; lpExitCode
100058E3 push    0                ; dwMilliseconds
100058E5 push    offset aTaskkill_exe_0 ; "taskkill.exe /f /im MSExchange*"
100058EA call     sub_10001080
100058EF push    0                ; lpExitCode
100058F1 push    0                ; dwMilliseconds
100058F3 push    offset aTaskkill_exe_1 ; "taskkill.exe /f /im sqlserver.exe"
100058F8 call     sub_10001080
100058FD push    0                ; lpExitCode
100058FF push    0                ; dwMilliseconds
10005901 push    offset aTaskkill_exe_2 ; "taskkill.exe /f /im sqlwriter.exe"
10005906 call     sub_10001080
10005908 push    0                ; lpExitCode
1000590D push    0                ; dwMilliseconds
1000590F push    offset aTaskkill_exe_3 ; "taskkill.exe /f /im mysqld.exe"
10005914 call     sub_10001080
10005919 add     esp, 3Ch

```

Fig. 44 Executed commands after the encryption is over

3 WannaCry Ransomware

Name: diskpart.exe

md5: 84c82835a5d21bbcf75a61706d8ab549

Type: encrypting ransomware

The malware generates a unique identifier based on the computer name, as shown in Fig. 31. A check is made to see if the malware was started with /i argument.

Run with /i Argument

The malware copies the binary to C:\ProgramData\<GeneratedID>\tasksche.exe if the directory exists; otherwise it is copied to C:\Intel\<GeneratedID>\tasksche.exe and updates the current directory to the new directory. The binary tries to open the service named <GeneratedID>. If it doesn't exist, the malware creates one with DisplayName <GeneratedID>, the BinaryPath of cmd \c <PathOftasksche.exe>, and starts the service. It attempts to open the mutex Global\MsWinZonesCacheCounterMutexA0; if it isn't created within 60 s, the malware starts itself with no arguments.

Run Without /i Argument

The binary updates the current directory to the path of the module and creates a new registry key HKLM\Software\WanaCrypt0r\wd which is set to the CD. The

malware then loads the XIA resource and extracts multiple files to the current directory; the complete list is shown below:

Filename	MD5 hash
b.wnry	c17170262312f3be7027bc2ca825bf0c
c.wnry	ae08f79a0d800b82fcbe1b43cdbdbefc
r.wnry	3e0020fc529b1c2a061016dd2469ba96
s.wnry	ad4c9de7c8c40813f200ba1c2fa33083
t.wnry	5dcaac857e695a65f5c3ef1441a73a8f
u.wnry	7bf2b57f2a205768755c07f238fb32cc
taskdl.exe	4fef5e34143e646dbf9907c4374276f5
taskse.exe	8495400f199ac77853c53b5a3f278f3e

The msg directory is created with different ransom notes in multiple languages (Fig. 32). The ransomware opens c.wnry (configuration data) and loads it into memory. The malware chooses between three bitcoin addresses, 13AM4VW2dhxYgXeQepoHkH SQuy6NgaE b94, 12t9YDPgwueZ9NyMgw519-p7AA8isjr6SMw, and 115p7UMMngojl pMvkpHijcRdfJNXj 6LrLn, writes it to offset 0xB2 in the config data, and writes the updates back to c.wnry. The binary sets a hidden attribute to the current directory using CreateProcessA API with **attrib +h** and executes the command **icacls ./grant Everyone:F /T /C /Q** in order to grant all users permissions to the current directory.

The malware uses **CryptImportKey** to import the hardcoded private RSA key (Fig. 33). The file t.wnry is then opened and the first 8 bytes are compared with the magic value “WANACRY!”; the next 4 bytes need to be 0x100; then the next 256 bytes are written in memory. The encrypted key decrypts to the AES key BEE19B98D2E5B12 211CE211EECB13DE6, as shown in Fig. 34. The AES key is used to decrypt the encrypted data, which was read from t.wnry and saves the result as a DLL. The TaskStart export function of the DLL is called, and it deals with the encryption of the files. It creates a mutex which is called **MsWinZonesCacheCounterMutexA** and reads the configuration file c.wnry. A new mutex is then created by the ransomware, **Global\MsWinZonesCacheCounterMutexA0**.

The binary will try to open a file 00000000.dky file, which at this point doesn’t exist, and it will then try to load a 00000000.pky file. If this one doesn’t exist, the ransomware will then import a public RSA key, as shown in Fig. 35. A new pair of RSA2048 keys is generated and the public key is saved to 00000000.pky, as shown in Fig. 36. The malware uses the hardcoded RSA key to encrypt the generated private key and saves the result to 00000000.eky (Fig. 37). A thread that writes 136 bytes to 00000000.key is created every 25 s (if it exists, otherwise it is created). Initially, as shown Fig. 38, 8 generated bytes and 128 zero bytes are written to the file, and after that it is written to a buffer, which contains the current time of the system. A thread that launched taskdl.exe, which is used to delete encrypted files, is created (which has that specific extension). Another thread is created that scans for new drives every 3 s; if it finds a new drive and this isn’t a CDROM drive, it

encrypts the drive. The sample imports another RSA key, as shown in Fig. 39. The process @WanaDecryptor@.exe with the “fi” argument is created, and this one can communicate with the server in order to obtain an updated bitcoin address. The file u.wnry is copied and saved as @WanaDecryptor@.exe; a script file is created and executed with the content shown below. The ransomware reads the content of r.wnry, updates the content with a ransom amount and bitcoin address, and writes the content to @Please_Read_Me@.txt (Fig. 40). The process starts scanning a directory, creates a hidden file with the prefix “~SD,” and then deletes it. The files which have the .exe, .dll, and .WNCRY extensions as well as the files which were created by the malware are not encrypted. The list of attacked extensions is presented in Fig. 41. Each file is encrypted using AES-128 algorithm in CBC mode with NULL IV. For every file a unique AES key is generated, as is shown in Fig. 42. The structure of an encrypted file is WANACRY!, length of RSA encrypted data, RSA encrypted AES key, file type, original file size, and AES encrypted content. The AES key is encrypted using the embedded RSA key or generated RSA key depending on a number which is generated (if it is a multiple of 100, the AES key is encrypted using the embedded RSA key; otherwise it is encrypted using the generated RSA public key), as shown in Fig. 43. The ransomware executes the following commands after the encryption is finished (Fig. 44). The process is trying to encrypt the logical drives that aren’t of DRIVE_CD

ROM type; it executes the commands @WanaDecryptor@.exe co and cmd.exe /c start /b @WanaDecryptor@.exe vs and copies the b.wnry to every folder on the desktop (it is saved as @WanaDecryptor@.bmp). The encryption algorithms are consistent and it is not possible to restore the files without paying the ransom; however there are some decryptors that work for Windows XP, Windows 7, Windows Vista, and Windows Servers 2003 and 2008.

Acknowledgment The authors would like to thank University Politehnica of Bucharest for the financial support.