Challenges in cyber security - Ransomware Phenomenon

Paşca Vlad-Raul and Simion Emil

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 [1], 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 cyber space which is called WannaCry, and according to [2], 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 U.S. National Security Agency (NSA) and released by The Shadow Brokers on 14 april 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 [3] (the campaign from 14.02.2017) and [4] (the campaign from 06.03.2017).

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, 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, 1cd, dbf, sqlite, accdb, jpg, jpeg, tiff, zip, rar, 7z, backup, sql, bak. One crucial point here is that everybody can rename the files in order to avoid such

---

Paşca Vlad-Raul
Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Splaiul Independenei 313, Bucharest, Romania, 060042, e-mail: vvladd_pasca@yahoo.com

Simion Emil
Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Splaiul Independenei 313, Bucharest, Romania, 060042, e-mail: emil.simion@upb.ro

infections, but the mandatory requirement is to back up the data.

Spora doesn't adds extensions to the encrypted files, which is really unusual in the case of ransomware, for example Locky adds .locky extension, TeslaCrypt adds .aaa extension, WannaCry appends .WNCRY extension. In this case, each file is encrypted with a separate key and it is a non deterministic 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), 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 reveals 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 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, attack in case of small difference between prime numbers p and q.

Similar situation on AES implementation: ECB attack, CBC implementation without HMAC verification, 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 figure 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. In 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.

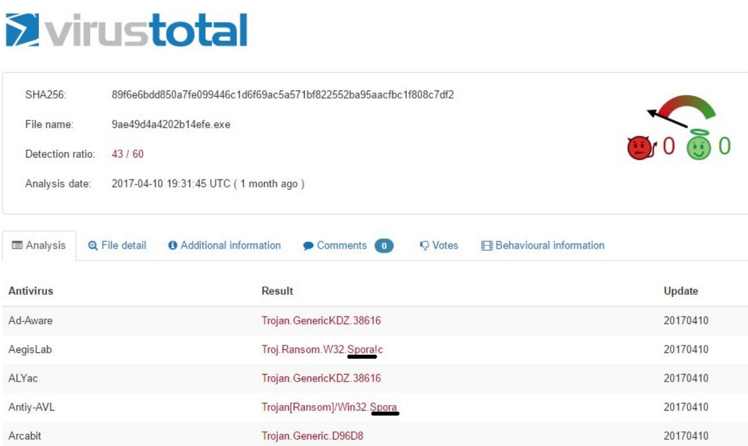


Fig. 1: VirusTotal Report

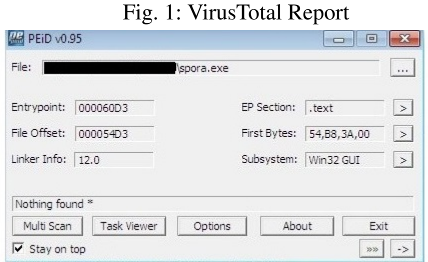


Fig. 2: PEiD Report

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

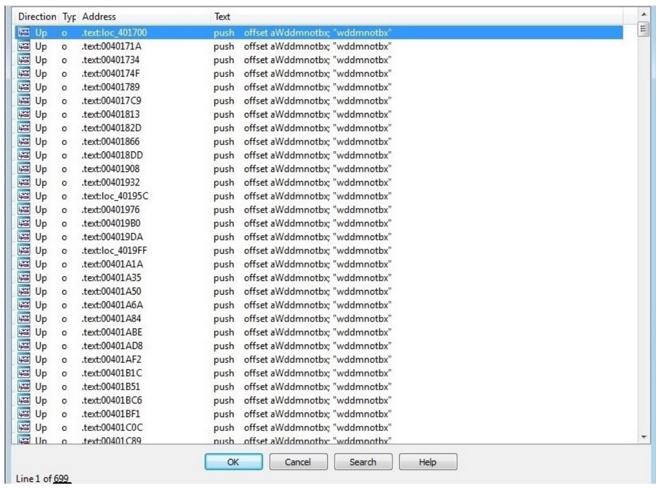


Fig. 3: IDA Pro 1

In the figure 4 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 in this case, because the malware doesn't call **CreateMutexA**, this is another trick used to complicate the analysis.

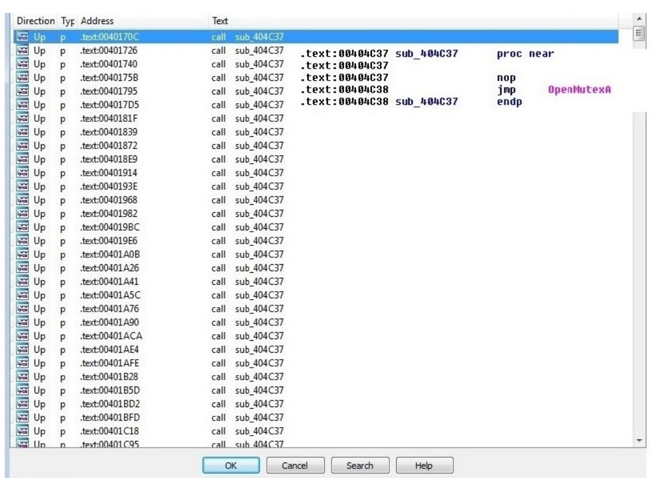


Fig. 4: IDA Pro 2

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 97e84cc8afca475d15d8c3e1f38ddeba.

The malware calls **GetVolumeInformationW** to get information about the file system and volume associated with the root directory, as shown in figure 5.
A mutex is created and it has the following format: m⟨GetVolumeInformationResult⟩ (in decimal), to ensure that the malware runs only once.

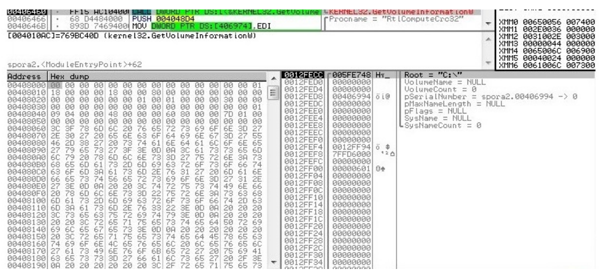


Fig. 5: GetVolumeInformationW call

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 figure 6. It represents an AES256 key, stored in a form of a blob. The 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.

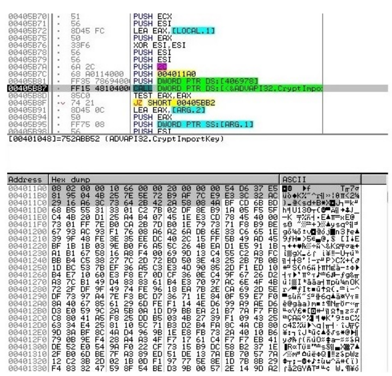


Fig. 6: CryptImportKey call

The AES Key is used to decrypt another key, which is a RSA key embedded in the binary, as shown in figure 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.

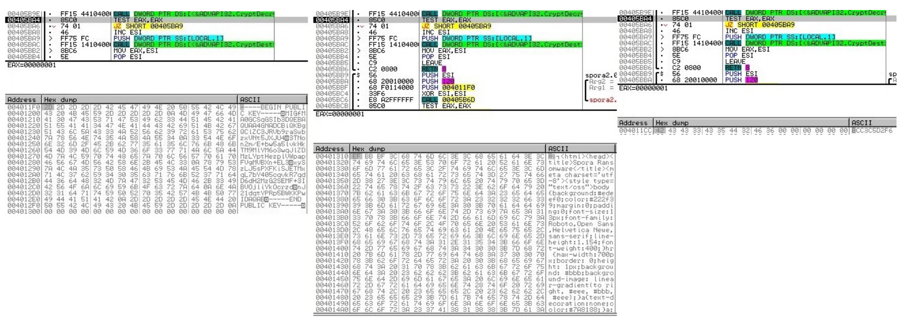


Fig. 7: CryptDecrypt calls

The attacked extensions are presented in the table below:

.xls	.doc	.xlsx	.docx	.rtf	.odt	.pdf	.ppt	.pptx
.psd	.dwg	.cdr	.cd	.mdb	.1cd	.dbf	.sqlite	.accdb
.jpg	.jpeg	.tiff	.zip	.rar	.7z	.backup	.sql	.bak

The next folders are excluded from the attack:

windows	program files	program files (x86)	games

For every victim, the malware creates a pair of RSA keys. Below, the fragment which generates the RSA key pair (1024 bit):

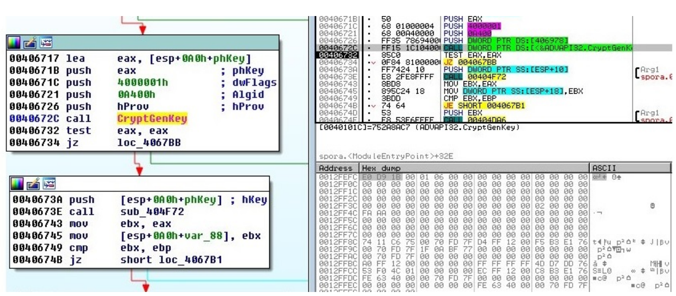


Fig. 8: CryptGenKey call

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 Figure 9.

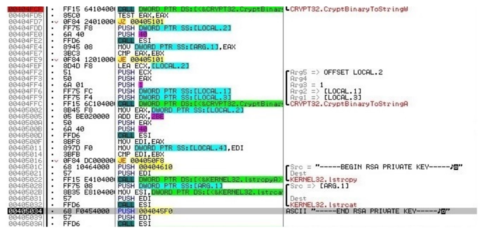


Fig. 9: RSA Key is Base64 encoded

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 Figure 10.

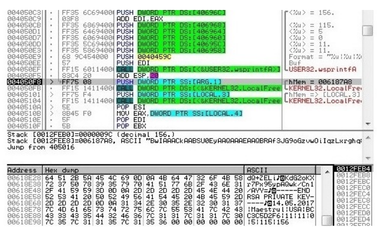


Fig. 10: Buffer contains information about system

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 Figure 11.

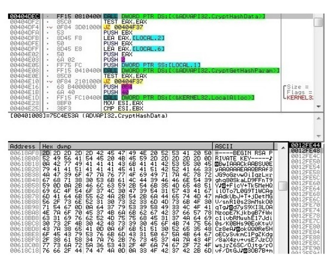


Fig. 11: MD5 Algorithm is used to hash the buffer

Another AES key is generated then it's exported and encrypted using public RSA key that was hardcoded. In figure 12 is shown this process.

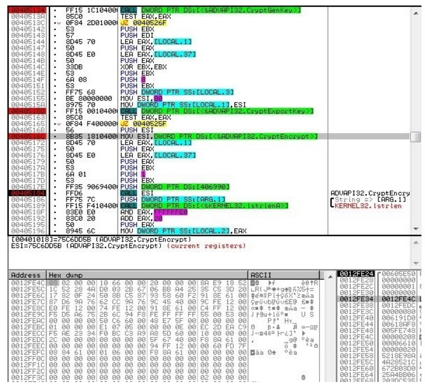


Fig. 12: Another AES key is generated, exported and encrypted using the embedded RSA key

The generated AES key is used to encrypt the data (including the RSA private key), as shown in figure 13. Finally, all encrypted data is Base64 encoded and stored in the ransom note.

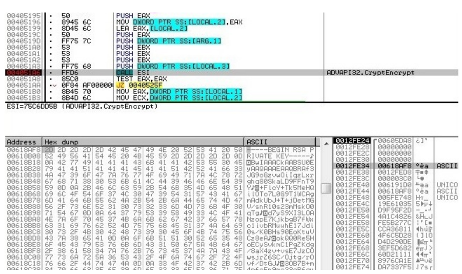


Fig. 13: The AES key, which was generated, is used to encrypt a private RSA key

For every file is generated a new AES256 key, as shown below:

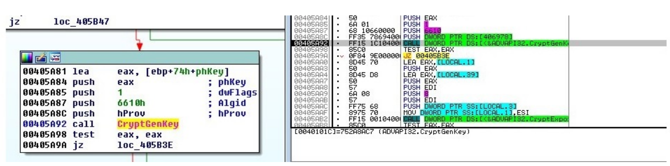


Fig. 14: Another AES256 key is generated

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.

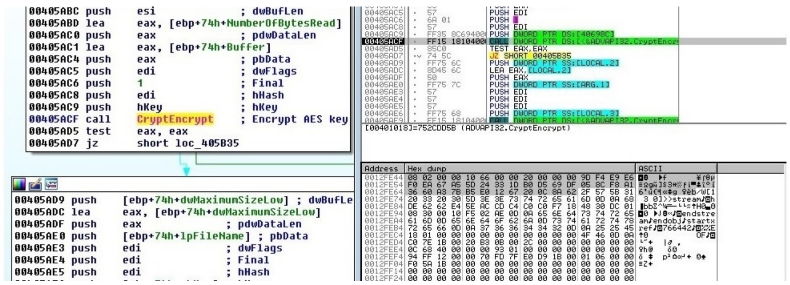


Fig. 15: The AES key is encrypted using RSA key

Each file is encrypted using the AES key, as shown in the figure.

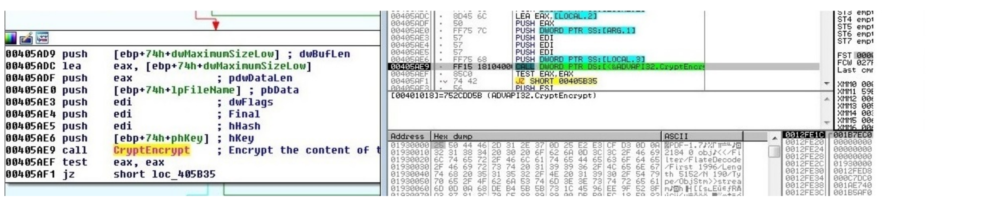


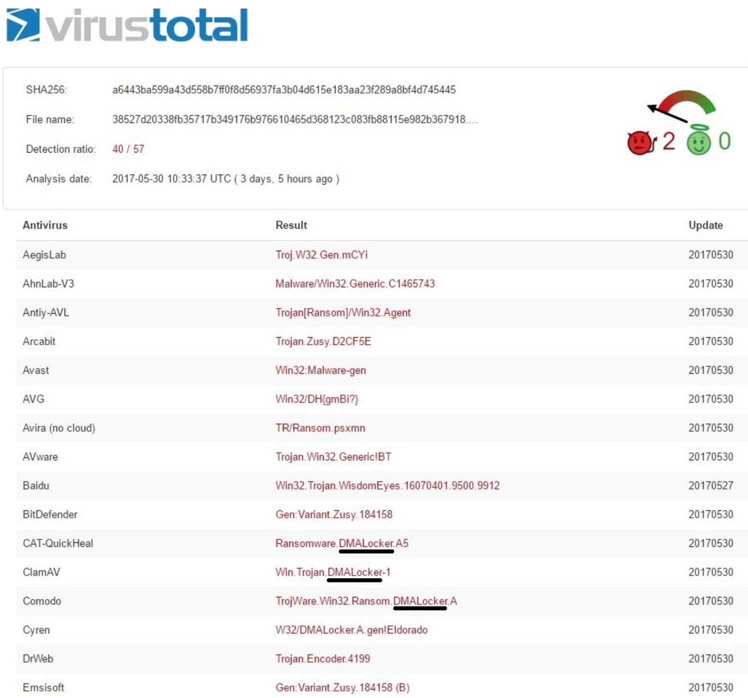
Fig. 16: The file is encrypted using the AES key

In order to decrypt a file, a ransom note is uploaded to the server giving the attacker access to all information needed. He use 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.

2 DMA Locker ransomware

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

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



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{gmBi?}	20170530
Avira (no cloud)	TR/Ransom.psxmn	20170530
AVware	Trojan.Win32.Generic!BT	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

According to the Figure 18, the ransomware isn't packed, if this is obfuscated it is then necessary to reveal it.

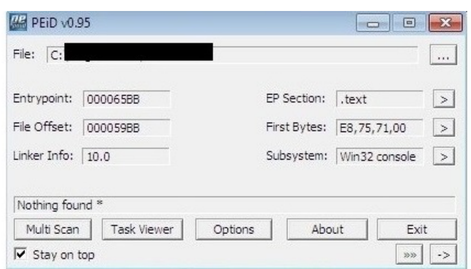


Fig. 18: PEiD Report DMA Locker

As shown in Figure 19, the malware moves the original file to C:\ProgramData and renames the file svchosd.exe (the author of ransomware is trying to hide the malicious purposes, in order to look like the Service Host Process svchost.exe).

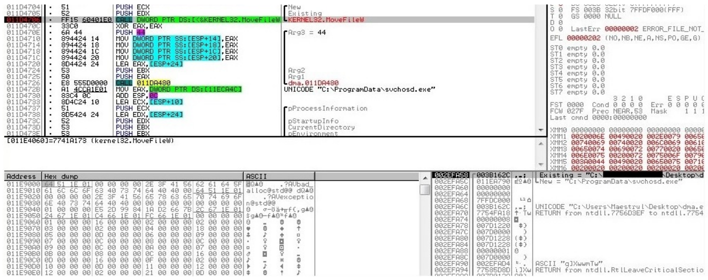


Fig. 19: The malware moves the original file to another location

Once the file is copied, the malware starts svchosd.exe process (which obviously is a copy of the original process) and then exits. As shown below, the function **CreateProcessW** is used.

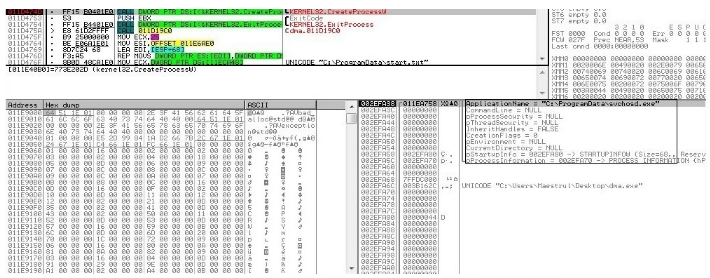


Fig. 20: The malware starts a copy of the original process

The original process creates two keys in registry for persistence: **HKLM\Software\Microsoft\Windows\CurrentVersion\Run\Windows Firewall** which has the value C:\ProgramData\svchosd.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 the following figure.

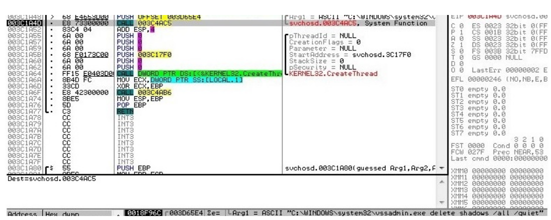


Fig. 21: DMA Locker deletes backups and shadow copies

A start.text 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 below:

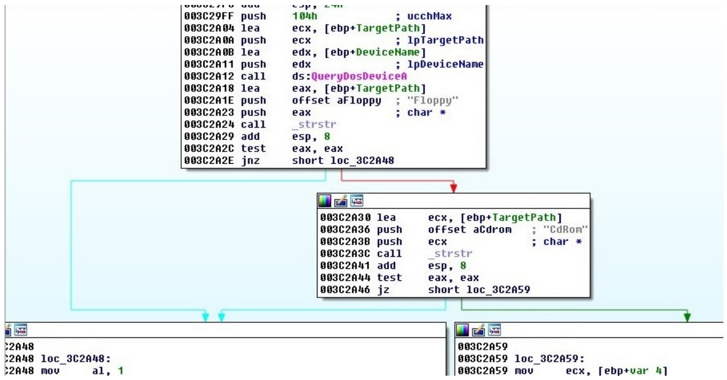


Fig. 22: Floppy and CD are skipped

The sample uses a hardcoded public RSA key, stored in a form of BLOB, as shown in figure below:

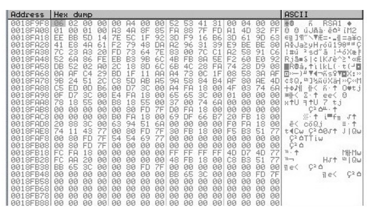


Fig. 23: Hardcoded RSA key

Some directories which are excluded from the encryption process, this entire list is shown below:

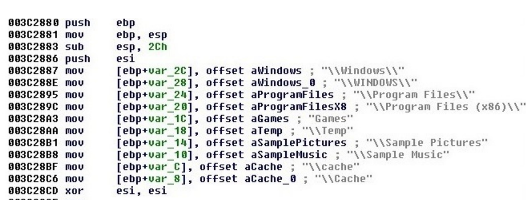


Fig. 24: The directories which are excluded from the encryption

A list of skipped extensions is presented in the next figure:

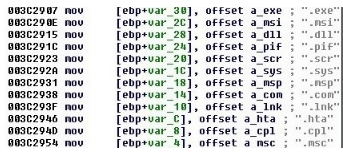


Fig. 25: Skipped extensions

A unique AES256 key is generated for every file using the API **CryptGen-Random**, as shown in the figure below:

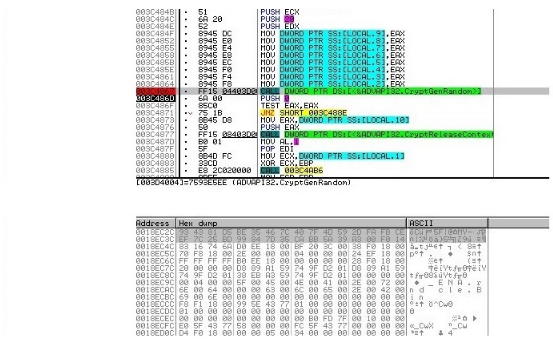


Fig. 26: A unique AES key is generated

The AES key is used to encrypt 16 byte long data with AES ECB mode, as shown in the figure below:

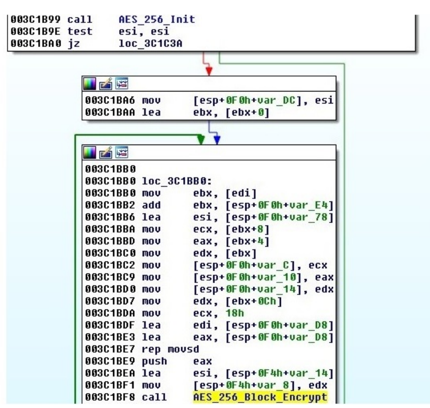


Fig. 27: The data is split in chunks of 16 bytes and encrypted

Once used, the AES key is encrypted using the hardcoded RSA key:

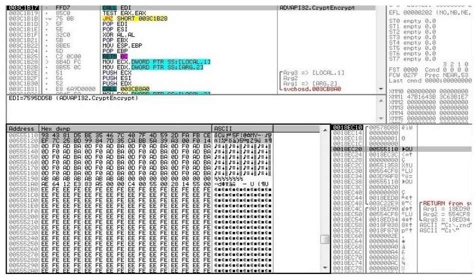


Fig. 28: The AES key is encrypted using the hardcoded RSA key

The structure of the encrypted file is: the prefix which is added, encrypted AES key and the encrypted original content:

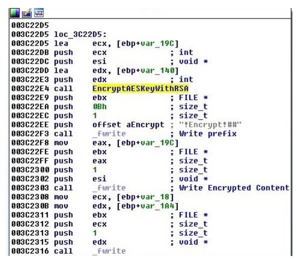


Fig. 29: A prefix is added to each file

Once the encryption process is complete, a message alert is presented:

Fig. 30: DMA Locker Message

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.

3 WannaCry ransomware

Name: diskpart.exe
md5: 84c82835a5d21bbcf75a61706d8ab549
Type: encrypting ransomware

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

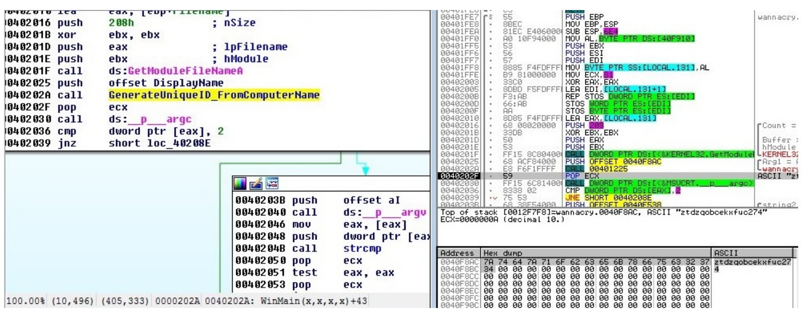


Fig. 31: A unique identifier is generated for every victim

Run with /i argument

The malware copies the binary to C:\ProgramData\ $\langle GeneratedID \rangle$ \ tasksche.exe if the directory exists, otherwise it is copied to C:\Intel\ $\langle GeneratedID \rangle$ \tasksche.exe and updates the current directory to the new directory. The binary tries to open the service named $\langle GeneratedID \rangle$. If it doesn't exists, the malware creates one with DisplayName $\langle GeneratedID \rangle$, the BinaryPath of cmd \c $\langle PathOftasksche.exe \rangle$ and starts the service. It attempts to open the mutex Global\MsWinZonesCacheCounterMutexA0, if it isn't created within 60 seconds, 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:

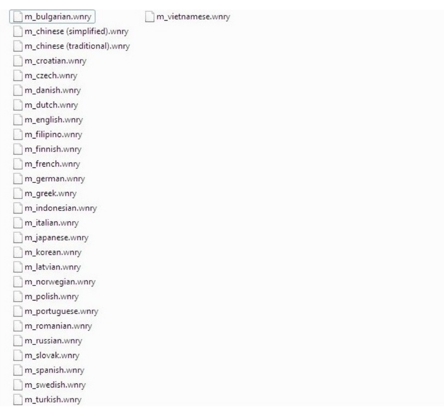


Fig. 32: Ransom notes

The ransomware opens c.wnry (configuration data) and loads it into memory. The malware chooses between 3 bitcoin addresses, 13AM4VW2dhxYgXeQepoHkH SQuy6NgaE b94, 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw, 115p7UMMngoj1 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:

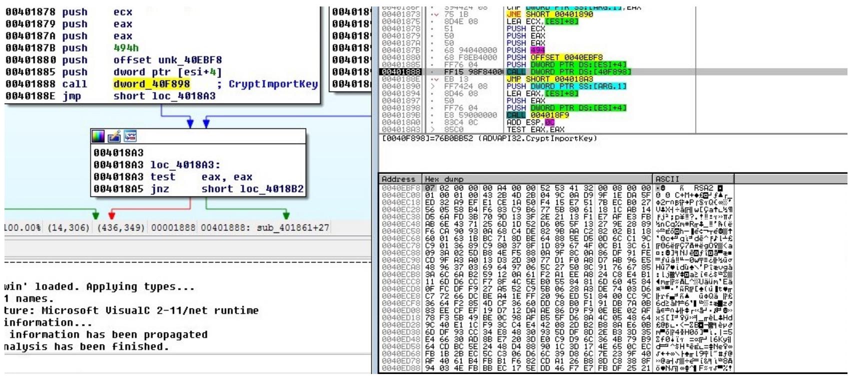


Fig. 33: private RSA key is being imported

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 the figure below:

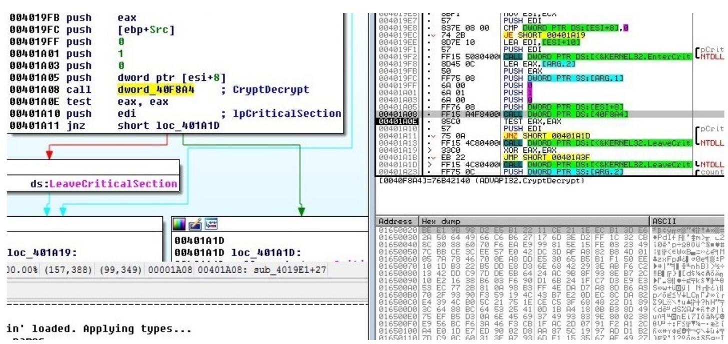


Fig. 34: The encrypted key is decrypted using private RSA key

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 below:

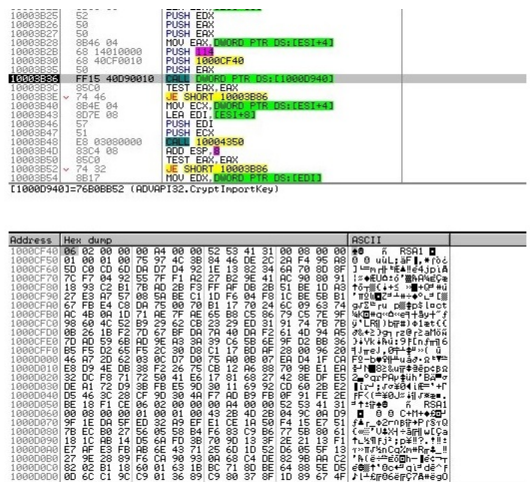


Fig. 35: public RSA key is being imported

A new pair of RSA2048 keys is generated and the public key is saved to 00000000.pky, as shown below:

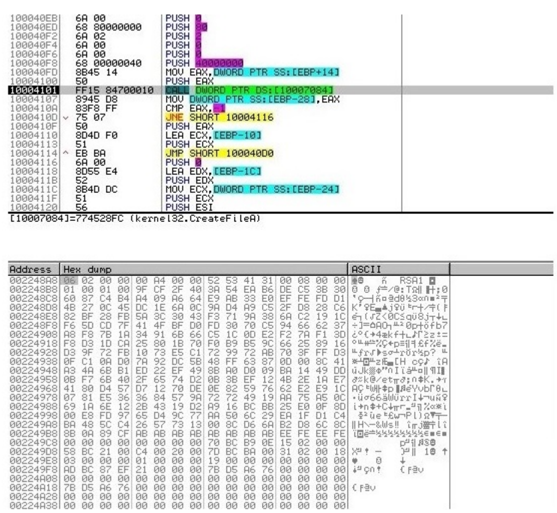


Fig. 36: The public key is exported and saved to 00000000.pky

The malware uses the hardcoded RSA key to encrypt the generated private key and saves the result to 00000000.eky:

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

A thread that writes 136 bytes to 00000000.key is created every 25 seconds (if it exists, otherwise it is created). Initially, as shown below, 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.

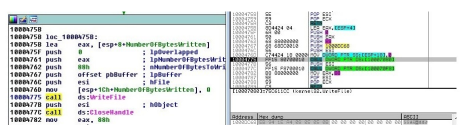


Fig. 38: Firstly, the 8 generated bytes and 128 zero bytes are written to the file

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 seconds, 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 figure below.

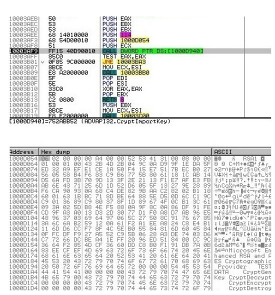


Fig. 39: Another RSA key is being imported

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.wrny 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.

```
@echo 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
cscript.exe //nologo m.vbs
del m.vbs

del /a %0
```

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

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 below:

```
.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.odg.otg.sxm.mml.lay.lay6.asc.sqlite3
.sqlitedb.sql.accdb.mdb.db.dbf.odb.frm.myd.myi.ibd.mdf.ldf.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.fla.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.xltm
.xltx.xlc.xlm.xlt.xlw.xlsb.xlsm.dotx.dotm.dot.docm.docb.jpg.jpeg.snt.onetoc2
.dwg.pdf.wk1.wks.123.rtf.csv.txt.vsdx.vsd.eml.msg.ost.pst.pptx.ppt.xlsx.xls.docx
.doc
```

Fig. 41: Targeted extensions by malware

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 below. 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.

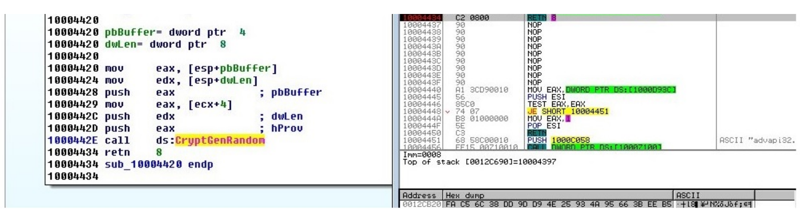


Fig. 42: A new AES key is generated for every file

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 the figure:

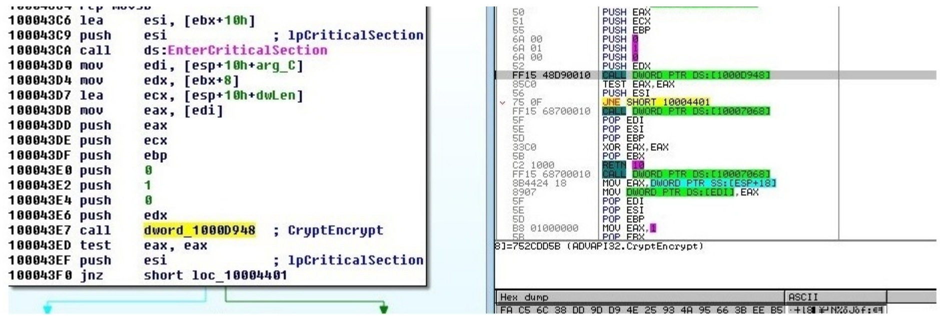


Fig. 43: The AES key is encrypted using RSA key

The ransomware executes the following commands after the encryption is finished:

Fig. 44: Executed commands after the encryption is over

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, Windows Server 2003 and 2008.

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

References

1. https://www.adaware.com/blog/cryptowall-ransomware-cost-users-325-million-in-2015.
2. https://www.cnet.com/news/wannacry-wannacrypt-uiwix-ransomware-everything-you-need-to-know.
3. http://malware-traffic-analysis.net/2017/02/14/index2.html.
4. http://malware-traffic-analysis.net/2017/03/06/index.html.