

Ransomware data recovery techniques

Irimia Alexandru-Vasile

Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi,

Email: alexviirimia@gmail.com

Abstract

This article presents and explains methodologies that can be employed to recover information from encrypted files generated by ransomware based on cryptanalytic techniques. By using cryptanalysis and related knowledge as much as possible, the methodology's goal is to use static and dynamic analysis as little as possible. We present three case studies that illustrate different approaches that can be used to recover the encrypted data.

Keywords: cryptanalysis; ransomware; stolen information; encrypted data

1. Introduction

Nowadays, malware attacks are becoming more and more frequent to disrupt services, steal sensitive information, block access to data and much more. In this article we will be talking about ransomware, a certain type of malware that threatens its victims to destroy or block access to data or systems until a ransom is paid.

Many companies, as well as individuals, are affected annually by this type of malware. According to [1], in 2019, a new company was infected with ransomware every 14 seconds. The cost of these infections is also high, as a ransom pay for a small business is on average \$5,900 (see [3]), but it can be as high as \$50 million, the highest demand in history that Acer was requested in 2021 (see [4]). Most of the time it is less costly to not pay the ransom, as seen in [2], in 2020 the average ransom pay was \$1,450,000, while the cost to recover from the attack without paying the ransom was \$732,000. Even

by paying the ransom, victims do not get all their data back. As seen in [2], files get corrupted in the decryption process and only recover around 65 percent of data due to technical faults in the ransomware itself. On the bright side, businesses affected by ransomware retrieved their data in 57 percent of cases utilizing backups. A further 8 percent of people recovered their data using other methods. This results in a 97 percent data recovery rate when combined with the 32 percent who actually paid the ransom (see [2]).

Ransomwares take various form, the most common being:

- **Crypto ransomware or encryptors** are the most well-known and the most damaging. They encrypt files and data within a system, then ask the victim to pay a ransom to recover them.
- **Lockers** block computer functions such as being able to use your mouse and keyboard, or not being able to access the desktop, making the computer inoperable until the ransom demands are met.
- **Scareware** is fake software that claims to have detected an issue on the victim's computer or a virus and demands payment to solve those problems. Some of these malwares lock the computer or flood the screen with pop-up alerts.
- **Doxware\Leakware** threatens to distribute sensitive or personal information online.
- **RaaS (Ransomware as a service)** is a variant that is anonymously hosted by a "professional" hacker that handles all aspects of the attack in return for a cut of the ransom.

In this article we will be talking about **crypto ransomware** and methodologies used to recover the encrypted data based on cryptanalytic and reverse engineering techniques.

2. History of ransomware

The first ransomware was **AIDS Trojan** (see [5]), also known as **PC Cyborg**, created in 1989 by Joseph Popp and was distributed to 20,000 attendees at the 1989 World Health Organization AIDS conference, hence the name. The malware was distributed over a floppy disk, so it relied only on researchers' interest about what was on the disk, not on any kind of exploit. There was a questionnaire regarding AIDS on the floppy disk. The application was installed by researchers, and everything worked as intended on their computers up to the 90th restart. The ransomware would encrypt the victim's filenames, but not the contents of the files. It would also demand \$189 in licensing fees for the PC Cyborg Software, which were to be paid by cashier's check or international money order and sent to a Panama P.O. Box, but later a decryptor called **CLEARAID** was developed that would restore the files without paying the ransom.

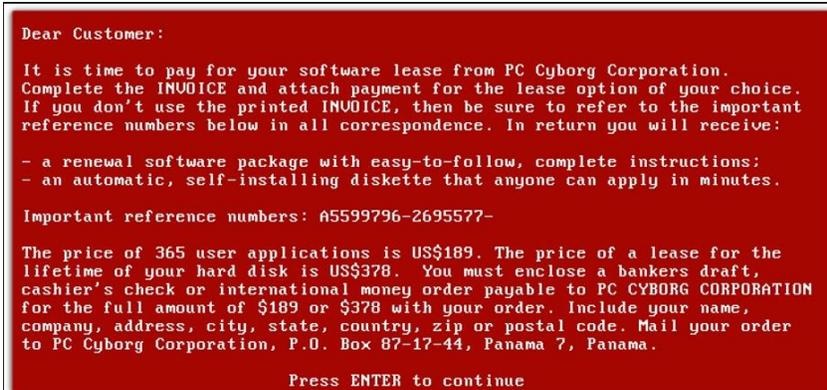


Figure 1. AIDS Trojan ransom note

More than a decade later, in 2004, wave of ransomware infections starts taking place, beginning with **GPCoder** identified by Symantec as a Trojan that in its September 2005 Internet Security Threat Report “encrypts data files such as documents, spreadsheets, and database files on the compromised computer,” although it was not labeled as ransomware. A note demanding \$200 as ransom would be left in each directory. The next year, the **Archiveus Trojan** tried a slightly different strategy in 2006. Only the files in the "My Documents" folder would be encrypted. Victims had to purchase decryption software from certain websites to access their files. It is fascinating to observe how much of the note from the Archiveus Trojan has been directly appropriated by modern ransomware, including the following passage:

“Do not try to search for a program that encrypted your information—it simply does not exist in your hard disk anymore. System backup will not help you to restore files. Reporting to police about a case will not help you, they do not know the password. Reporting somewhere about our email account will not help you to restore files. Moreover, you and other people will lose contact with us, and consequently, all the encrypted information.”

Many other ransomwares would appear in waves from that point onward, that would try different approaches to encrypt data and make its recovery more difficult, sometimes even managing to make it virtually impossible. As time passes, more and more ransomware variants are appearing and the damage costs they produce keep increasing. According to [1] and [9], in 2015 ransomware cost the world \$325 million, in 2021 around \$20 billion and it is expected to rise to \$265 billion by 2031.

3. Proposed methodology

The analysis of the file holding the encrypted stolen data is the first stage in the suggested approach, the phases of which are shown in **Figure 2**. This may be done by simply opening it in a hexadecimal editor to see if it is a text file or if there were any patterns that would suggest the file is a binary one, a method of encoding, such as Base64, Radix-64, or something else. Once this verification receives a favorable result, one should go on to the data decoding. Repeat this procedure until it is impossible to recognize an output that contains encoded data.

Compression of the data can be used to determine its amount of redundancy, which can be used to determine whether a traditional (or weak) cryptographic procedure is being employed. A good encryption system should provide random-looking results, which indicates that compression should result in a larger file, instead of a smaller one. This is because compression relies on a small number of items appearing more frequently than others, which should not happen in a random stream when considering a sample of appropriate size because each element typically appears around the same number of times. Making a histogram of the file contents and looking for an uneven distribution of the data is another method to verify this.

The cryptanalysis of a traditional algorithm may be done using a variety of methods. One may use frequency analysis, which is based on certain facts, for basic substitution ciphers: in the ciphertext, plaintext symbol frequencies are preserved, and each language has a unique symbol frequency distribution. Given these details, the concept is straightforward by exchanging similar-frequency characters from one alphabet for another. Language statistics, specifically those pertaining to the frequency of digraphs and trigraphs, can also be used to crack transposition ciphers. Another idea could be the usage of the frequency of digrams and trigrams in the given language. One can use Kasiski's [10] approach, which considers that a repeating sequence of symbols produces the same ciphertext when encrypted with the same key locations, in the case of polyalphabetic mechanisms. This determination of the key length k , which is sufficient to limit the original issue to the cryptanalysis of k mono-alphabetic ciphers, is made possible by this observation. As an alternative, the index of coincidence [10], which gauges the relative frequency of symbols in the ciphertext, can be used to determine the period of the polyalphabetic cipher.

Typically, the encryption algorithm that produced a specific ciphertext should not be identifiable. However, by looking at the encrypted data, one can at least attempt to gather some information about the type of cipher that was used. One way to do that is by searching for known structures that might be used in different ciphers. For instance, $PC1$ and $PC2$ matrices are often defined by DES [10] implementations to be used in the key scheduling procedure, or the forward or inverse S -Boxes matrices definitions, which are already in place, if we use AES [10] as another example. From here the key can be found by looking at the code referencing that data. Another way to find the key would be looking for it in the binary of the data. Hiding sensitive information in the source code is surely a vulnerability, but malware creators usually do that. If this fails, Shamir's technique [7] can be used, which considers the entropy of a securely generated key. The idea is to look through a fixed-size window for an area that has the greatest entropy by scanning the entire binary.

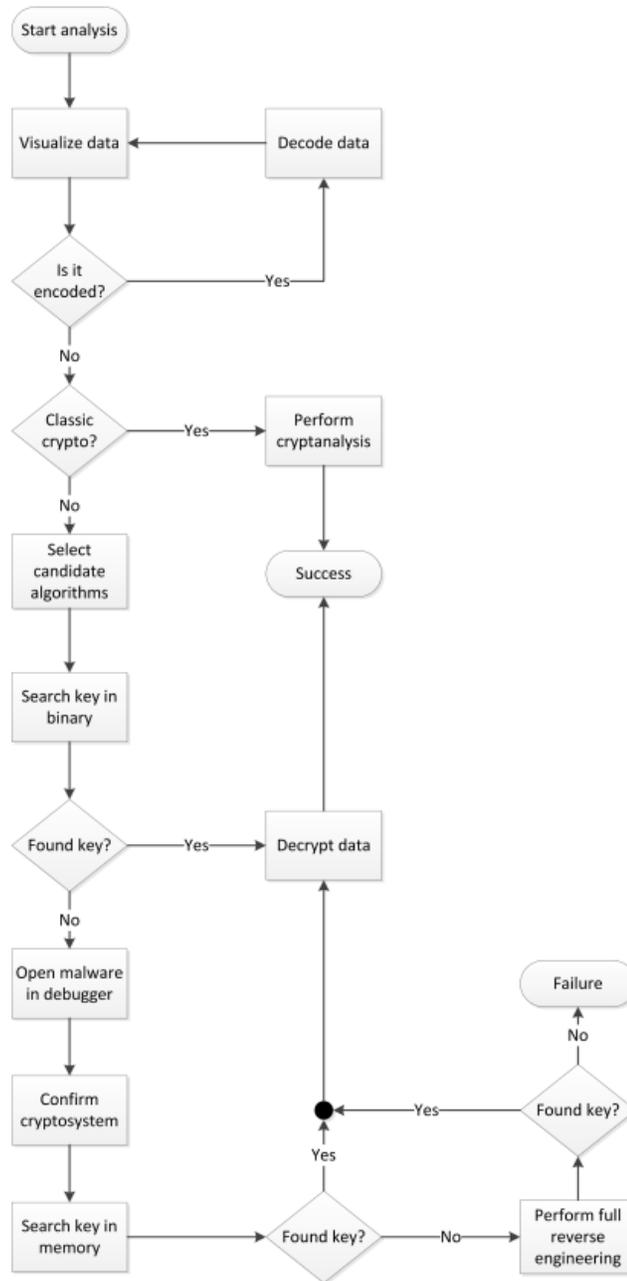


Figure 2. Proposed methodology (see [6])

4. Case studies

4.1. First Malware

The malware presented in this part only uses classical cryptography, therefore, to recover the original data we only need to examine the output file. The name of this malware is **system.exe** because this is the name of the file found on its victim's computer.

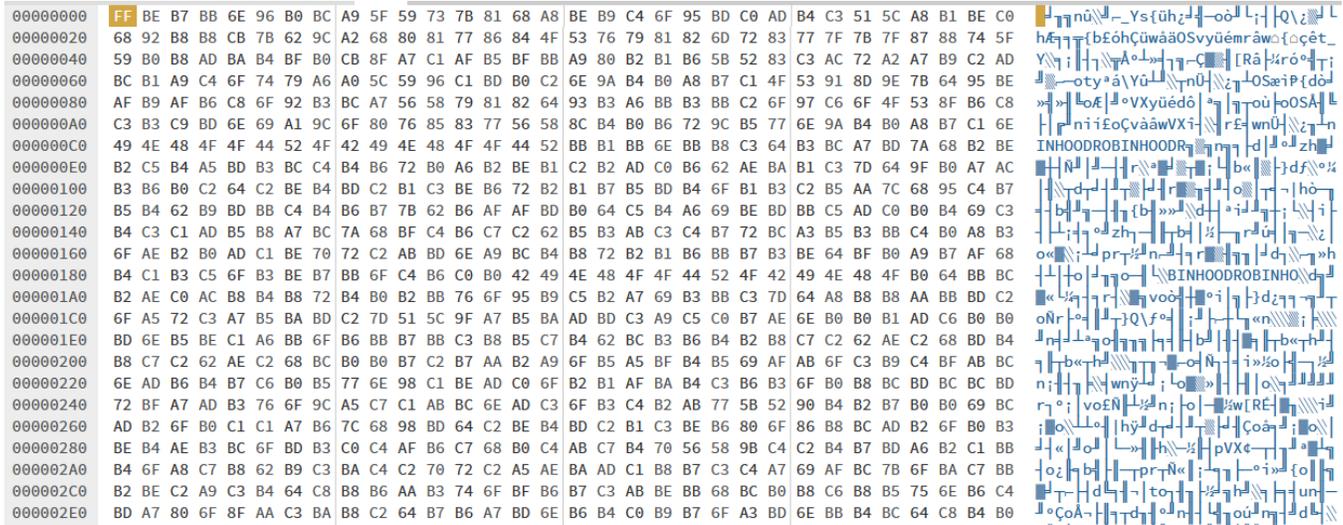


Figure 3. Encrypted sample of "system.exe"

As was already mentioned, one of the fundamental principles of a good encryption algorithm is that its output should appear random, meaning that there should be no patterns in the ciphertext. The encrypted file in this section does not comply with this requirement, which can be seen by the repeated appearances of "ROBINHOOD". The histogram shown in Figure 3 clearly demonstrates a non-uniform distribution, with values concentrated between 100 and 200. The histogram in Figure 4 may also be used to visually identify the problem.

Now that we have established that a classic cryptographic scheme is used, we need to find out which one. The first thing that might come to mind is that a constant number modulo 256 is being added to each byte. We can check this, but it will not produce any meaningful output. By looking at the repeating text "ROBINHOOD", we see that it appears 198 (0xC6) bytes from the beginning of the file and 225 bytes after the first appearance, on position 0x195. Both 198 and 225 are divisible by 9, the length of the text. This might mean that a Vigenère cipher [10] over an alphabet of 256 elements is used as the encryption algorithm. From now on we can use different methods to break this cipher such as the Kasisky method to find the key length, frequency tables or any other method described in [11]. For our case, it is enough to guess the key by looking at the ciphertext. The text "ROBINHOOD" looks like a good candidate for it since it is very possible that it was added to a series of null bytes from the original text. Testing this theory, we obtain the result shown in Figure 5.

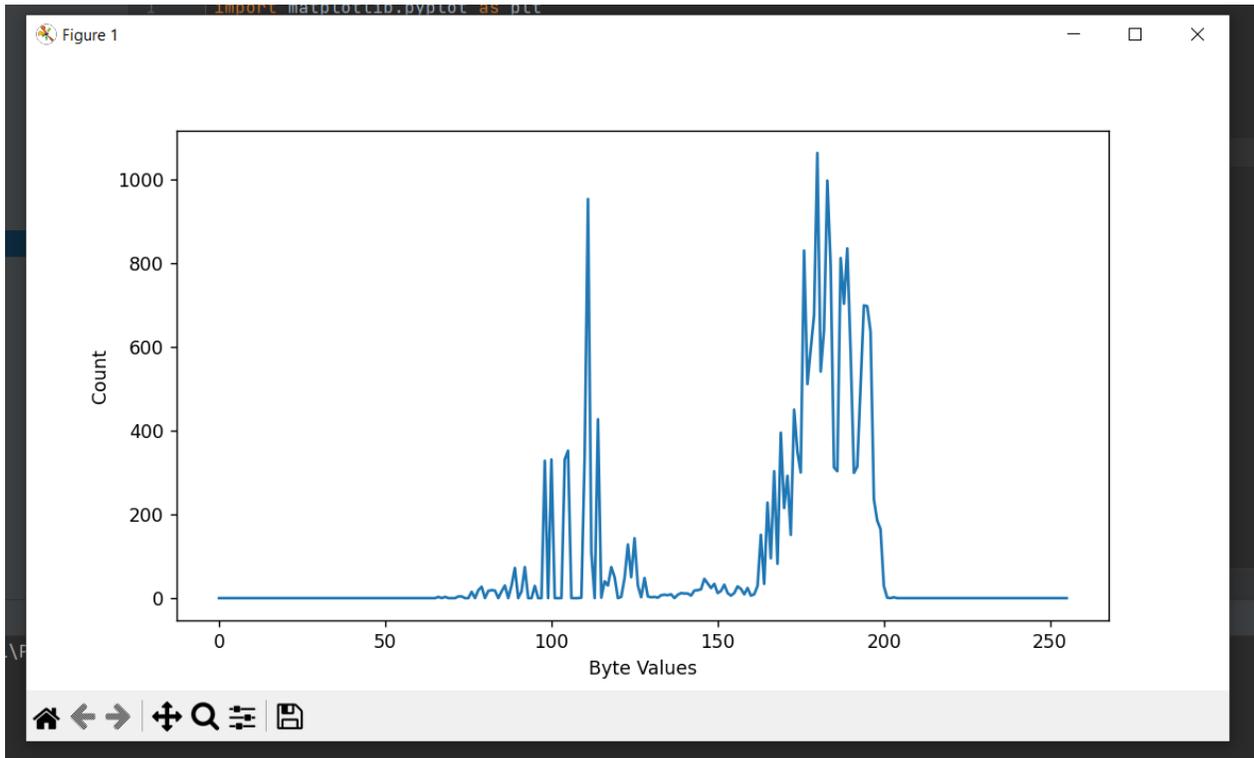


Figure 4. Histogram of byte values for the encrypted file

```

00000000 59 6F 75 72 20 4E 61 6D 65 0D 0A 31 32 33 20 59 6F 75 72 20 53 74 72 65 65 74 0D 0A 59 6F 75 72 Your Name..123 Your Street..Your
00000020 20 43 69 74 79 2C 20 53 54 20 31 32 33 34 35 0D 0A 28 31 32 33 29 20 34 35 36 20 37 38 39 30 0D City, ST 12345..(123) 456-7890.
00000040 0A 6E 6F 5F 72 65 70 6C 79 40 65 78 61 6D 70 6C 65 2E 63 6F 6D 0D 0A 34 74 68 20 53 65 70 74 65 .no_reply@example.com..4th Septe
00000060 6D 62 65 72 20 32 30 58 58 0D 0A 52 6F 6E 6E 79 20 52 65 61 64 65 72 0D 0A 43 45 4F 2C 20 43 6F mber 20XX..Ronny Reader..CEO, Co
00000080 6D 70 61 6E 79 20 4E 61 6D 65 0D 0A 31 32 33 20 41 64 64 72 65 73 73 20 53 74 20 0D 0A 41 6E 79 mpany Name..123 Address St ..Any
000000A0 74 6F 77 6E 2C 20 53 54 20 31 32 33 34 35 0D 0A 44 65 61 72 20 4D 73 2E 20 52 65 61 64 65 72 2C town, ST 12345..Dear Ms. Reader,
000000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6C 6F 72 20 73 69 74 20 61 6D 65 74 2C 20 63 6F .....lor sit amet, co
000000E0 6E 73 65 63 74 65 74 75 65 72 20 61 64 69 70 69 73 63 69 6E 67 20 65 6C 69 74 2E 20 4D 61 65 63 nsectetur adipiscing elit. Maec
00000100 65 6E 61 73 20 70 6F 72 74 74 69 74 6F 72 20 63 6F 6E 67 75 65 20 6D 61 73 73 61 2E 20 46 75 73 enas porttitor congue massa. Fus
00000120 63 65 20 70 6F 73 75 65 72 65 2C 20 6D 61 67 6E 61 20 73 65 64 20 70 75 6C 76 69 6E 61 72 20 75 ce posuere, magna sed pulvinar u
00000140 6C 74 72 69 63 69 65 73 2C 20 70 75 72 75 73 20 6C 65 63 74 75 73 20 6D 61 6C 65 73 75 61 64 61 ltricies, purus lectus malesuada
00000160 20 6C 69 62 65 72 6F 2C 20 73 69 74 20 61 6D 65 74 20 63 6F 6D 6D 6F 64 6F 20 6D 61 67 6E 61 20 libero, sit amet commodo magna
00000180 65 72 6F 73 20 71 75 69 73 20 75 72 6E 61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 61 20 69 6D eros quis urna.....a im
000001A0 70 65 72 64 69 65 74 20 65 6E 69 6D 2E 20 46 75 73 63 65 20 65 73 74 2E 20 56 69 76 61 6D 75 73 perdiet enim. Fusce est. Vivamus
000001C0 20 61 20 74 65 6C 6C 75 73 2E 0D 0A 50 65 6C 6C 65 6E 74 65 73 71 75 65 20 68 61 62 69 74 61 6E a tellus..Pellentesque habitan
000001E0 74 20 6D 6F 72 62 69 20 74 72 69 73 74 69 71 75 65 20 73 65 6E 65 63 74 75 73 20 65 74 20 6E 65 t morbi tristique senectus et ne
00000200 74 75 73 20 65 74 20 6D 61 6C 65 73 75 61 64 61 20 66 61 6D 65 73 20 61 63 20 74 75 72 70 69 73 tus et malesuada fames ac turpis
00000220 20 65 67 65 73 74 61 73 2E 20 50 72 6F 69 6E 20 70 68 61 72 65 74 72 61 20 6E 6F 6E 75 6D 6D 79 egestas. Proin pharetra nonummy
00000240 20 70 65 64 65 2E 20 4D 61 75 72 69 73 20 65 74 20 6F 72 63 69 2E 0D 0A 41 65 6E 65 61 6E 20 6E pede. Mauris et orci...Aenean n
00000260 65 63 20 6C 6F 72 65 6D 2E 20 49 6E 20 70 6F 72 74 74 69 74 6F 72 2E 20 44 6F 6E 65 63 20 6C 61 ec lorem. In porttitor. Donec la
00000280 6F 72 65 65 74 20 6E 6F 6E 75 6D 6D 79 20 61 75 67 75 65 2E 0D 0A 53 75 73 70 65 6E 64 69 73 73 oreet nonummy augue...Suspendiss
000002A0 70 20 64 75 69 20 70 75 72 75 73 2C 20 73 63 65 6C 65 72 69 73 71 75 65 20 61 74 2C 20 76 75 6C e dui purus, scelerisque at, vul
000002C0 70 75 74 61 74 65 20 76 69 74 61 65 2C 20 70 72 65 74 69 75 6D 20 6D 61 74 74 69 73 2C 20 6E 75 putate vitae, pretium mattis, nu
000002E0 6E 63 2E 20 4D 61 75 72 69 73 20 65 67 65 74 20 6E 65 71 75 65 20 61 74 20 73 65 6D 20 76 65 6E nc. Mauris eget neque at sem ven
00000300 65 6E 61 74 69 73 20 65 6C 65 69 66 65 6E 64 2E 20 55 74 20 6E 6F 6E 75 6D 6D 79 2E 0D 0A 46 75 enatis eleifend. Ut nonummy...Fu
00000320 73 63 65 20 61 6C 69 71 75 65 74 20 70 65 64 65 20 6E 6F 6E 20 70 65 64 65 2E 20 53 75 73 70 65 sce aliquet pede non pede. Suspe
00000340 6E 64 69 73 73 65 20 64 61 70 69 62 75 73 20 6C 6F 72 65 6D 20 70 65 6C 6C 65 6E 74 65 73 71 75 ndisse dapibus lorem pellentesqu
00000360 65 20 6D 61 67 6E 61 2E 20 49 6E 74 65 67 65 72 20 6E 75 6C 61 2E 0D 0A 44 6F 6E 65 63 20 62 e magna. Integer nulla...Donec b
00000380 6C 61 6E 64 69 74 20 66 65 75 67 69 61 74 20 6C 69 67 75 6C 61 2E 20 44 6F 6E 65 63 20 68 65 6E landit feugiat ligula. Donec hen
000003A0 64 72 65 72 69 74 2C 20 66 65 6C 69 73 20 65 74 20 69 6D 70 65 72 64 69 65 74 20 65 75 69 73 6D drerit, felis et imperdiet euism
000003C0 6F 64 2C 20 70 75 72 75 73 20 69 70 73 75 6D 20 70 72 65 74 69 75 6D 20 6D 65 74 75 73 2C 20 69 od, purus ipsum pretium metus, i
000003E0 6E 20 6C 61 63 69 6E 69 61 20 6E 75 6C 6C 61 20 6E 69 73 6C 20 65 67 65 74 20 73 61 70 69 65 6E n lacinia nulla nisl eget sapien
00000400 2E 20 44 6F 6E 65 63 20 75 74 20 65 73 74 20 69 6E 20 6C 65 63 74 75 73 20 63 6F 6E 73 65 71 75 . Donec ut est in lectus consequ
00000420 61 74 20 63 6F 6E 73 65 71 75 61 74 2E 0D 0A 45 74 69 61 6D 20 65 67 65 74 20 64 75 69 2E 20 41 at consequat...Etiam eget dui. A
00000440 6C 69 71 75 61 6D 20 65 72 61 74 20 76 6F 6C 75 74 70 61 74 2E 20 53 65 64 20 61 74 20 6C 6F 72 liquam erat volutpat. Sed at lor
00000460 65 6D 20 69 6E 20 6E 75 6E 63 20 70 6F 72 74 61 20 74 72 69 73 74 69 71 75 65 2E 0D 0A 50 72 6F em in nunc porta tristique...Pro
00000480 69 6E 20 6E 65 63 20 61 75 67 75 65 2E 20 51 75 69 73 71 75 65 20 61 6C 69 71 75 61 6D 20 74 65 in nec augue. Quisque aliquam te
000004A0 6D 70 6F 72 20 6D 61 67 6E 61 2E 20 50 65 6C 6C 65 6E 74 65 73 71 75 65 20 68 61 62 69 74 61 6E mpor magna. Pellentesque habitan

```

Figure 5. Decrypting the sample with key "ROBINHOOD"

4.2 Second Malware

The second malware is called **TorrentLocker**, also known as **Crypt0Locker** [13]. It is a ransomware tool that encrypts files and targets all versions of Windows, including Windows XP, Windows Vista, Windows 7, and Windows 8 and it was released towards the end of August 2014. After encryption, a ransom note like the one in Figure 6 will pop up on the victim's machine. Starting at roughly \$550 USD, the ransom increases after about three days. Each infected user has a different bitcoin address to which the ransom must be paid.

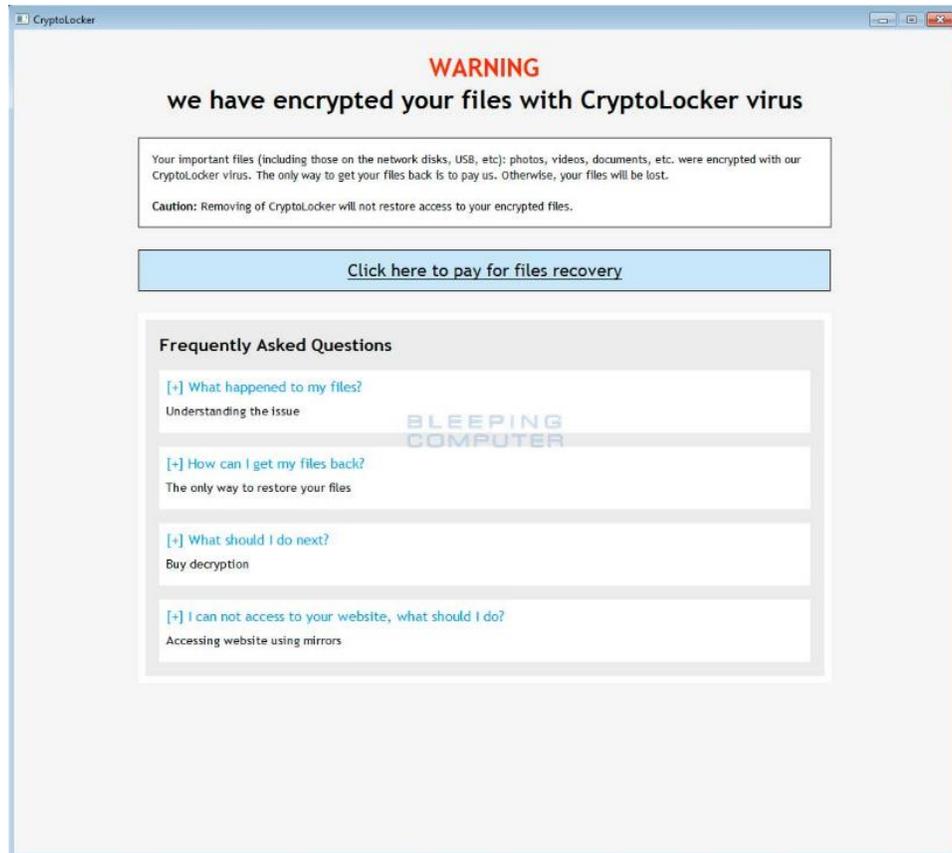


Figure 6. TorrentLocker ransom note

By just looking at the note we are not able to tell how the files are encrypted since it does not tell us anything about it. The encrypted files suggest that a strong encryption algorithm is being used such as AES, DES etc., because of the randomness of the text (Figure 7).

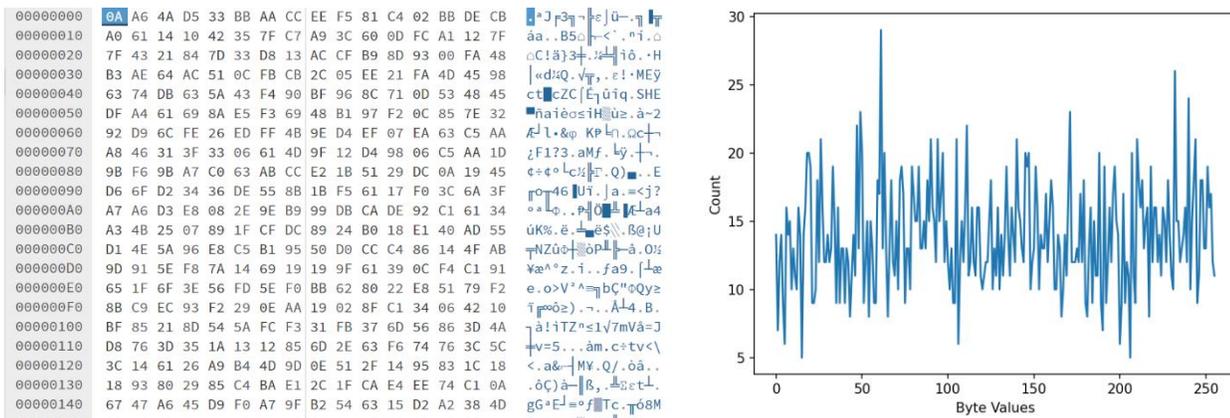


Figure 7. TorrentLocker encrypted file sample and byte distribution

The only option we have now is to look through the binaries of the malware to try to find out how the encryption is done. If we look at the strings found throughout the binary, we can see references to the encryption process. The most valuable information we can extract from this is "tomcrypt\nodes\ctr\ctr_encrypt.c" which is a cryptographic toolkit. The name of the path suggests the use of AES in CTR mode.

```

611839296A789A38C0045C8A5FB42C7D18D998F54449579B446817AF8D17273E662C97EE72995EF42640C550B9013FAD0761353C7086A272C240888E94769FD16650
PrimaryEmail
DisplayName
PStoreCreateInstance
pstorec.dll
stamp
LibTomMath
_on_before_encryption_1_work
error
can't set work type
tomcrypt\nodes\ctr\ctr_encrypt.c
_on_before_encryption_2_work
can't set number of enc files
_on_encryption_work
warning
stamp (%d)
rack_check_and_set_instance
_set_encrypted_file_name

```

Figure 8. Strings found in the malware binaries

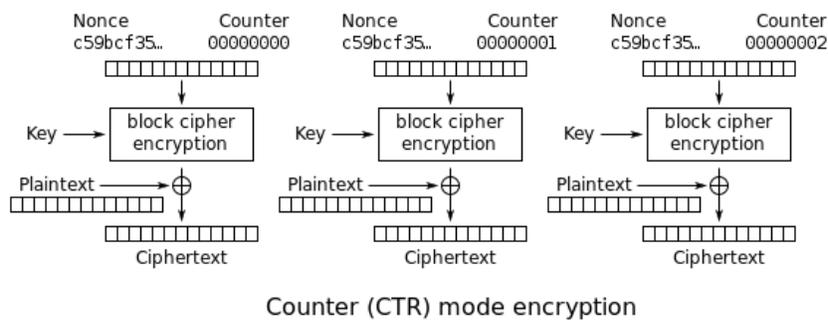


Figure 10. AES CTR mode encryption (see [14])

We can notice another important thing if we look at other encrypted files. As seen in Figure 9, multiple files seem to start with the same sequence of bytes. This might be because the encryption uses the same key and nonce to encrypt all the files and the files shown in Figure 10 might be a certain type of files such as executables, images, or anything else that has a certain format.

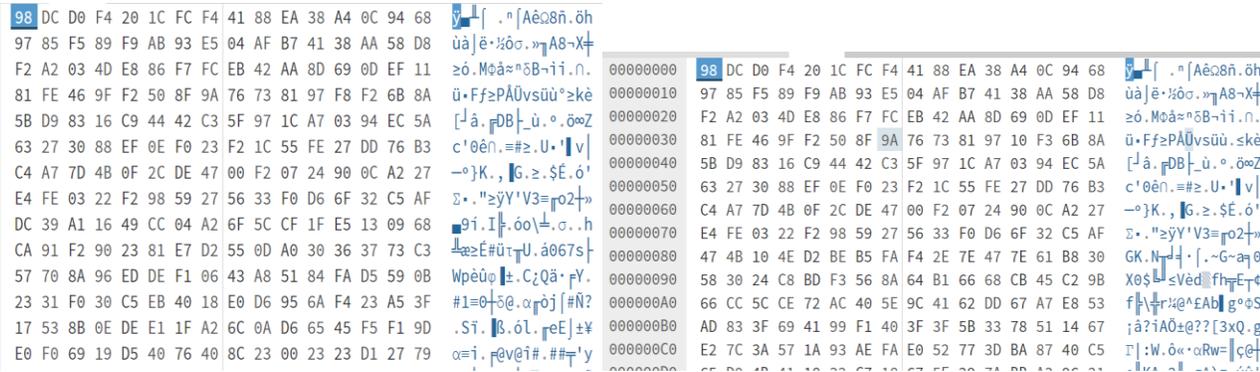


Figure 10. TorrentLocker encrypted samples

If we look at the encryption algorithm described in Figure 9, we can see a vulnerability in the implementation of the ransomware: since the same key and nonce is being used, the algorithm will output the same keystream at every encryption, therefore, we can use the following algorithm to decrypt out files without even knowing the key:

```

A' = ENCRYPT(A) //the encrypted file
B = large plaintext of NULL bytes (size(B)>size(A))
B' = ENCRYPT(B)

KEYSTREAM = B XOR B'

return A' XOR KEYSTREAM

```

If we follow the described algorithm, we can encrypt out plaintext full of null bytes using the ransomware itself and obtain the following result:



Figure 11. Decrypted file

4.3 Third malware

The third malware is called **portsys.exe** because this is the name of the file this malware is found on the victim's device. In Figure 12 we have a sample encrypted by this ransomware. At first glance it seems that the file is Base64 encoded. If we decode it, we obtain the result from Figure 13 which seems to be encrypted using a strong encryption algorithm.

```
00000000 57 44 4A 77 61 42 36 46 72 64 51 65 45 41 59 4B 6E 43 4C 4D 4B 51 47 35 78 42 39 72 WDJwaB6FrdQeEAYkNCLMKQG5x89r
0000001C 49 31 62 58 55 5A 64 6D 46 49 4D 5A 4C 64 49 64 62 37 36 61 6A 34 71 32 65 38 47 67 I1bXUZdmFIMZLdIdb76aj4q2e8Gg
00000038 51 6E 38 55 63 56 48 36 7A 2B 62 30 35 50 75 59 42 62 70 38 32 55 47 61 37 4E 5A 56 Qn8UcVH6z+b05PuYbBp2UGa7NZv
00000054 74 54 76 69 52 54 79 53 77 6B 6D 57 31 52 59 77 77 52 74 4E 66 32 33 52 35 43 49 67 tTviRTySwkmW1RYwrtNf23R5CIg
00000070 6D 53 71 4D 6D 65 79 33 4F 47 5A 69 68 33 5A 4C 49 42 61 43 6B 6F 31 5A 37 4E 55 73 mSqMmey30GZ2ih3ZLIbACoLZ7NU5
0000008C 51 42 4F 50 53 39 38 56 4D 4B 67 56 35 59 44 36 65 49 2F 5A 41 57 30 47 6C 57 77 77 QBOP598VMKGV5VDGeI/ZAW0GLwww
000000A8 64 75 32 2F 56 57 32 48 4E 45 76 41 4E 51 43 43 55 53 62 44 53 38 74 72 2B 66 63 66 du2/Vw2HNEvANQCCUSbDS8tr+fcf
000000C4 75 6A 61 4F 55 43 69 48 63 51 68 68 2B 35 54 43 39 6B 72 31 4F 36 72 67 36 64 7A 65 uja0UCiHcQhh+5TC9krI06rg6dze
000000E0 70 57 54 32 51 72 5A 31 64 49 63 61 34 37 68 77 61 67 55 62 68 4F 6C 65 58 78 6D 32 pWT2QrZ1dIca47hwaqUbb0LeXm2
000000FC 65 66 47 47 2B 67 6B 7A 58 33 30 45 41 74 52 73 77 69 6E 55 67 73 5A 69 53 53 77 77 efGG+gkzX30EAtRswinUgsZiSSww
00000118 54 62 74 4F 62 55 35 62 33 4E 6F 73 52 4A 2F 4B 53 2B 44 35 6E 73 77 71 64 42 59 71 Tbt0bU5b3NosR3/KS+D5nswqdBYq
00000134 6B 78 54 31 32 42 62 66 55 34 72 74 73 69 52 62 64 75 6B 49 70 35 7A 75 4F 30 38 6D kxTl2BbFu4rtsiRbTukIpa5zu008M
00000150 6A 6C 55 58 30 6E 75 56 58 4F 37 79 33 66 68 56 75 32 4D 2F 79 41 51 72 74 6C 56 52 jLUX0nuVX07y3FhVu2M/yAQrtLVR
0000016C 38 50 42 5A 35 51 4C 38 75 38 4C 7A 51 61 5A 61 4A 4F 38 6A 45 30 5A 73 58 43 42 30 8PBZ5QL8u8LzQaZaJ08jE0ZsXCBO
00000188 53 4F 30 34 4E 57 33 71 39 45 32 73 43 74 54 74 75 6D 54 41 34 58 70 46 51 4C 35 55 S004Nw3q9E2sCtTumTA4XpFL5U
000001A4 48 56 4F 41 53 72 47 4C 57 36 31 6C 63 6B 36 30 35 6F 7A 78 71 41 75 4E 6F 4E 37 49 HVOASrGLW6l1ck605ozxqAuNoN7I
000001C0 53 2F 56 75 6F 57 34 6B 66 59 77 73 48 6F 65 51 42 77 2F 4B 4D 7A 32 73 32 34 75 75 S/VuoW4kfYwshHoeQBw/KMz2s24uu
000001DC 39 61 73 64 61 79 56 43 59 45 62 46 65 6D 5A 72 6D 46 74 69 46 66 4D 54 4F 54 33 6F 9asdayVCYEBFemZrmFtiFfMTOT3o
000001F8 76 34 5A 34 79 32 51 57 65 47 53 57 44 78 68 5A 68 68 78 61 4C 59 6D 6F 58 41 6A 6C v4Z4y2QWeGSWdxhZhxalYmoXAJc
00000214 52 68 6F 45 51 4B 50 33 41 48 41 6E 4C 48 51 6B 56 76 4D 6D 78 78 58 58 48 79 63 67 RhoEQKP3AHANLHQkVvMmxXXHYcg
00000230 6B 52 41 52 6F 66 54 4C 4A 38 47 50 2F 67 43 6C 61 62 67 78 44 42 36 67 34 51 37 49 kRARofTLJ8PG/gCtLabgxDB6g4Q7I
0000024C 71 34 48 54 38 68 66 4E 6F 47 47 6F 64 62 6C 63 64 4D 33 6A 56 42 44 73 6F 35 75 73 q4HT8hfNoGGodblcdM3jVBDso5us
00000268 7A 34 49 34 69 32 41 73 52 49 53 63 36 48 50 32 54 79 68 37 71 73 67 4E 36 52 4A 77 z4I4i2AsRiSc6HP27yh7qsgN6R3w
00000284 32 5A 4F 4F 59 57 6E 4D 54 46 33 59 2F 45 34 46 64 79 6C 43 62 32 70 76 58 2B 44 56 2Z00YwnMTF3Y/E4FdyLcb2pvX+DV
000002A0 2B 55 31 38 33 34 47 48 45 76 72 65 38 52 65 71 70 59 6E 4E 34 76 74 37 43 7A 46 63 +U1834GHEvre8ReapYnN4vt7CzFc
000002BC 38 6C 73 51 77 63 36 53 6C 37 47 5A 75 6D 73 37 49 70 41 2B 46 56 6D 59 46 71 37 6D 8lsQwc6S17GzumC7IpA+FVnYFq7m
000002D8 58 69 46 2B 31 49 75 4E 49 61 51 75 6D 75 6B 53 68 35 4D 39 78 58 2F 33 57 51 75 6B XiF+1IuNiAqumukSh5M9Xx/3WQuk
000002F4 35 77 46 36 57 2B 42 36 69 53 34 73 50 6D 38 52 54 52 58 33 72 70 53 65 4E 47 66 5A 5wF6w+B6iS4sPm8RTRX3rpSeNGfZ
00000310 5A 2F 56 77 31 69 6B 66 59 34 48 55 4C 4E 62 6B 57 50 38 54 57 42 31 38 55 72 71 30 Z/Vw1ikfY4HULNbkWP8TWB18Urq0
0000032C 45 31 71 4F 72 6A 44 79 6D 59 4E 45 69 42 2F 75 41 6F 44 67 67 41 68 55 6A 51 49 34 ElqOrjDymYNEiB/uAoDggAhUjQI4
00000348 63 47 64 62 69 48 64 47 71 59 36 2B 4E 4F 64 4F 6F 2B 46 6E 68 38 75 39 37 6D 52 4F cGdbiHdGqY6+N0dOo+Fnh8u97mRo
00000364 4F 41 66 59 34 56 2B 41 36 51 68 55 66 35 50 54 34 4B 49 6E 48 64 6A 4A 70 4E 7A 61 OAFY4V+AQ0uF5PT4KInHdjPnZa
```

Figure 12. Sample encrypted by portsys.exe

```
00000000 58 32 70 68 1E 85 AD D4 1E 10 06 0A 9C 22 CC 29 01 B9 C4 1F 6B 23 56 D7 51 97 66 14 K2ph.ä; k...E"}.-k#VQüf.
0000001C 83 19 2D D2 1D 6F BE 9A 8F 8A 86 7B C1 A0 42 7F 14 71 51 FA CF E6 F4 E4 FB 98 05 BA ä..T..oUAë{t-äBo..qQ-4uizvY.
00000038 7C D9 41 9A EC D6 55 B5 3B E2 45 3C 92 C2 49 96 D5 16 30 C1 1B 4D 7F 6D D1 E4 22 20 jAU00uH;reC<E7iU..oL..Mmmpo"
00000054 99 2A 8C 99 EC B7 38 66 62 87 76 4B 20 16 82 92 80 59 EC D5 2C 40 13 8F 4B DF 15 30 Ö+i0008fbçvk.ëëiYop.ëAK.0
00000070 A8 15 E5 80 FA 78 8F D9 81 6D 06 96 95 6C 30 76 ED BF 55 6D 87 34 4B C0 35 00 82 51 26 ç..c<x&L.m.öLövqUmç4K.5.ëQ&
0000008C C3 4B CB 68 F9 F7 1F BA 36 8E 50 28 87 71 08 61 FB 94 C2 F6 4A F5 3B AA E0 E9 DC DE HkK..GAP(cq..avö+2);-o00
000000A8 A5 64 F6 42 B6 75 74 87 1A E3 B8 79 6A 05 1B 84 E9 5E 5F 19 B6 79 F1 86 FA 09 33 5F Nd=H|utç.mpj..äö_..jy+ä..3
000000C4 7D 04 02 D4 6C C2 29 D4 82 C6 62 49 2C 30 4D BB 4E 6D 4E 5B DC 2A 2C 44 9F CA 4B E0 }...k|T)ë|H.I.0MqNm|Fp.Df&Kç
000000E0 F9 9E CC 2A 74 16 2A 93 14 F5 08 16 DF 53 8A ED B2 24 5B 4E E9 08 A7 9C EE 3B 4F 26 +P|tç..ö..j+..M$ë0$|Në..ëe;0&
000000FC 8E 55 17 D2 7B 95 5C EE F2 D0 F8 55 BB 63 3F C8 04 2B B6 55 51 F0 F0 59 E5 02 FC BB ÄU..T(iö;ë..UqçL+||UQ==Yç..q
00000118 C2 F3 41 A6 5A 24 EF 23 13 46 6C 5C 20 74 48 ED 38 35 6D EA F4 4D AC 0A D4 ED BA 64 T=A^ZSn#.Fl\ th085mz|MK..k|d
00000134 C0 E1 7A 45 40 8E 54 10 53 80 4A B1 8B 58 AD 65 72 4E B4 E6 8C F1 A8 0B 8D A0 DE C8 &Zë@T..SÇ||i|jerN|uiz..iä|
00000150 4B F5 6E A1 6E 24 7D 8C 2C 1E 87 90 07 0F CA 33 3D AC DB 8B AE F5 AB 1D 6B 25 42 60 K|nin$}i..çE..Ä3=||i};k&B'
0000016C 46 C5 7A 66 6B 98 5B 62 15 F3 13 39 3D E8 BF 86 78 CB 64 16 78 64 96 0F 18 59 86 1C F-zfkY|b..s=9=äqpd..xdu..Yä.
00000188 5A 2D 89 A8 5C 08 E5 46 1A 04 40 A3 F7 00 70 27 2C 74 24 56 F3 26 C7 15 D7 1F 27 20 Z-ëZ\..oF..öu=..p'..tSVs&|..|'
000001A4 91 10 11 A1 F4 CB 27 C1 8F FE 00 A5 69 38 31 9C 1E A0 E1 EC 8A AB 81 D3 F2 17 CD A0 æ..i|T'ÄA..Niq'..äb..Müë..-ä
000001C0 61 A8 75 89 5C 74 CD E3 54 10 EC A3 9B AC CF 82 38 8B 60 2C 44 84 9C E8 73 F6 4F 28 a2uH|tmtT..óu4/é8i'.Däëes+O(
000001DC 7B AA C8 0D E9 12 70 D9 93 8E 61 69 CC 4C 5D D8 FC 4E 05 77 29 42 6F 6A 6F 5F E0 D5 (-k..ë..p'öAai|L|+N..W)Bojo..oF
000001F8 F9 4D 7C 0F 81 87 12 FA DE F1 17 AA A5 89 CD E2 FB 7B 0B 31 5C F2 5B 10 C1 CE 92 97 -M|üç..|..Në=V(|..l|z|..Hëü
00000214 81 99 BA 60 8B 22 90 3E 15 59 98 16 AE E6 5E 21 7E 04 8B 8D 21 A4 2E 9A E9 12 87 89 ||0|q"ë>..Yy..u|'..k|iñ..Üe..çö
00000230 3D C5 7F 77 59 0B A4 E7 01 7A 5B 0E 7A 89 2E 2C 3E 6F 11 4D 15 F7 AE 94 9E 14 67 D9 +ç=V..ñç..z|azë..>..M..=öP4gl
0000024C 67 F5 70 06 29 1F 63 81 04 2C 06 4E 58 F7 13 58 10 7C 52 BA B4 13 5A 8E AE 30 F2 99 g|p|)..cü..pçX..X..|R|)..ZÄwë0
00000268 83 44 88 1F EE 02 80 E0 00 08 54 8D 02 38 70 67 5B 88 77 46 A9 8E BE 34 E7 4E A3 E1 äDë..e..ÇçC..Ti..8pg|ëwF-Ä4tNuB
```

Figure 13. File after Base64 decoding

If we inspect the binaries of the malware, we cannot see anything useful regarding how the file was encrypted. We could try to reverse engineer the file, but this will take some time. What we can do is try and find different constants strong encryption algorithms use such as S-box matrices from AES.

After many searches, we can see the PC1 and PC2 matrices used in DES encryption algorithm as seen in Figure 14.

```

00001540 15 E6 F7 F5 01 58 8D 0E 34 F6 7D DA B4 2F 23 BF 4A C7 D9 82 8B E9 C9 22 3F 3D 95 9E .µ=|.Xñ.4+}r|/#J|éTøP"?=òP
00001568 FF 54 C3 0D 46 3D 8F 4A C7 BB 3A A6 57 74 F0 8D ED 42 C7 96 40 20 42 EF 4F 6B 2C 9A T|.F=ÁJ|ñ:~Wt=içB|ú@ Bn0k,0
00001596 0F 09 B4 12 A6 2E 31 EE 35 71 AE 46 6C 1E F2 9A 14 3E 50 07 75 08 B0 8E 68 B8 43 F9 ..|.~.1e5q«FL.≥0.>P.u.\\Áñ.C.
00001624 7C 4E 9A C7 3F DC 98 12 47 1C 05 9B 76 F4 37 B4 93 A8 7E 49 D4 4D 9E 6F 7D 5C 31 81 |NÜ|?~ÿ.G..cv[7]ôz-I4MPo}\1ü
00001652 40 1A A4 97 AA 24 5A 20 08 0B 9F 2D DD 24 EF 33 2C B3 AC F5 4E 7D 0D D1 B3 FF 71 70 @.nú-$Z ..f-|$n3,|ñ|N|.T| qp
00001680 66 24 39 A0 84 B6 42 77 C6 56 9C 2F E5 29 9C B6 5E 54 A4 F1 90 DA 37 AD 85 47 2C 0F f$9áb|Bw|VE/c)E|ÁTñ±E|7i|äg,.
00001708 7C 06 C7 90 33 E7 26 A2 3E 31 E8 37 8C 8C 9E D3 87 85 59 7B B3 ED 4D 57 90 DC 69 90 |.|É3-&ó>1e7i|PçàY{|øMWEñiÉ
00001736 30 80 BD 4D 40 9E 57 49 41 33 25 17 09 01 58 50 42 34 26 18 10 02 59 43 35 27 19 11 0Ç|MøPñIA3%...XPB4&...YC5'...
00001764 03 60 52 44 36 52 74 86 48 B8 AC 66 4A 65 6B 7D 73 66 13 C3 ED 9C DA B3 E1 B6 0A BF .RD6RtâH|fJek}sf. |çE|B|B|.ç
00001792 67 D0 7F DB 20 F5 79 E9 52 90 AF 0C 8A B5 56 A7 44 52 61 1B 90 01 1A 5B 33 9F AB 36 gL- |yøRE».è|V°DRa.E...[3f;6
  
```

Figure 14. PC1 matrix found in malware binaries

PC1						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
above for C_i ; below for D_i						
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Figure 15. DES matrices extracted from [10]

The next step would be to find the encryption key. Since the PC1 matrix was found hardcoded in the source code of the malware, we might believe that the key is also hardcoded somewhere inside the binaries. As already mentioned, the key should be randomly generated. Starting from this information, an idea that comes to mind is to search for blocks of memory with high entropy. We can identify the locations of the keys by just looking at the data in some suitable way because most of the data in programs have some organization, whereas we anticipate seeing very little structure in key data.

We can find a key by breaking the data into smaller portions, measuring the entropy of each portion, and displaying the areas with very high entropy because we know that key data has more entropy than non-key data. Although obtaining a true measure of entropy is a difficult task, most program code has an entropy level that makes counting the unique bytes in each block a particularly good and simple measurement for entropy. After making some tests, we found that the entropy values for the majority of programs is similar to the one in Figure 16 when taking block of 64 bytes.

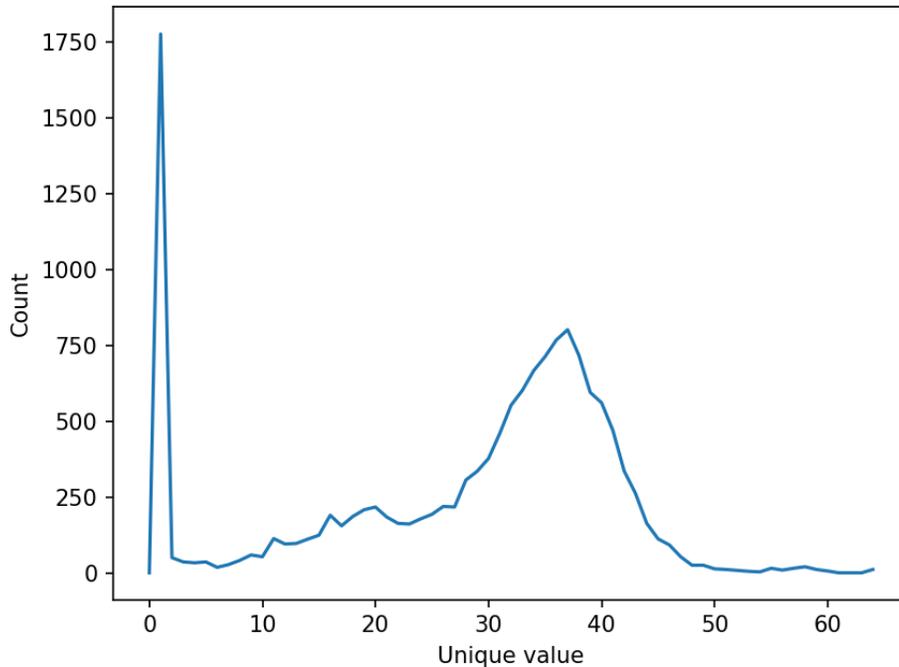


Figure 16. Entropy of the majority of binaries

As we can see, most blocks have around 30-40 unique byte values. By applying this method on our malware sample, we found 23 high entropy blocks, some of them being the blocks that contain PC1 and PC2 matrix, the other containing other forms of random data. By trying to decrypt the sample using different possible keys in these blocks, we found out key that correctly decrypts our file: 0xb34aa010811eb173.

5. Conclusions

In this post, we discussed a method for extracting data from malware-generated encrypted files with the least amount of work necessary. Most of the methods employed to achieve that goal rely on cryptanalysis rather than static and dynamic reverse engineering.

However, it should be noted that if cryptography is employed correctly in situations like the ones described above, it is impossible to succeed without having access to a memory dump of the real-world setting. A malware program that creates session keys to encrypt the stolen data and transfers it all together while being secured by a public key cryptosystem is one example. There isn't much that can be done to recover the original data if the criminal is the only one who has the private key. That would require the task of obtaining the data encryption key via cracking a well-known asymmetric cryptosystem.

References

- [1] [CV-HG-2019-Official-Annual-Cybercrime-Report.pdf \(herjavecgroup.com\)](#)
- [2] [sophos-state-of-ransomware-2021-wp.pdf](#)
- [3] [Datto2019_StateOfTheChannel_RansomwareReport.pdf](#)
- [4] [The State of Ransomware in 2021 | BlackFog](#)
- [5] [The History of Ransomware? Understand | Prevent | Recover](#)
- [6] "A Methodology for Retrieving Information from Malware Encrypted Output Files: Brazilian Case Studies", Future Internet 2013, 5, 140-167; doi:10.3390/fi5020140, Rua Dr. Ricardo Benetton Martins
- [7] "Playing hide and seek with stored keys", Adi Shamir and Nicko van Someren September 22, 1998
- [8] [Preparation Instruction \(mecs-press.org\)](#)
- [9] [Global Ransomware Damage Costs Predicted To Exceed \\$265 Billion By 2031 \(cybersecurityventures.com\)](#)
- [10] Menezes, A.; van Oorschot, P.; Vanstone, S. Handbook of Applied Cryptography, 5th ed.; CRC Press: Boca Raton, FL, USA, 2001
- [11] [Five-ways-to-crack-a-Vigenere-cipher.pdf \(cipherchallenge.org\)](#)
- [12] <https://www.virustotal.com/gui/>
- [13] https://www.welivesecurity.com/wp-content/uploads/2014/12/torrent_locker.pdf
- [14] [https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Counter_\(CTR\)](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Counter_(CTR))