WannaCry Ransomware Attack Lab

1 Lab Overview

The learning objective of this lab is to enable you to have an understanding of the WannaCry ransomware and its vulnerability. "WannaCry" is a ransomware cryptoworm which affected millions of computers on the Internet in 2017. WannaCry maliciously encrypts all the files of targeted computers running Microsoft Windows operating systems, exploiting the Server Message Block (SMB) protocol vulnerability, and it demands ransom payments from the user. Still, it is possible to perform a reverse engineering process to recover the decryption keys and therefore the encrypted files.

In this lab, you will be given a virtual environment to simulate the WannaCry ransomware attack. You will understand the various cryptographic algorithms adopted by WannaCry and be able to recover the encrypted file generated by WannaCry.

2 Background

2.1 Encryption Algorithms

WannaCry uses a combination of different encryption algorithms to encrypt files and keys. It uses both the Advanced Encryption Standard (AES) and the Rivest-Shamir-Adleman (RSA) algorithm in the attack. AES is a symmetric-key encryption algorithm, while RSA is asymmetric-key (or public key) encryption algorithm. It combines the computation efficiency in symmetric-key cryptosystem and convenience in key management in public-key cryptosystem.

The roles of AES and RSA in WannaCry are shown as follows:

AES WannaCry uses a 128-bit AES key to encrypt the files in Cipher Block Chaining (CBC) mode. Each file is encrypted by a separate random AES key. WannaCry searches for all storage devices in the targeted system and encrypts all the files with the following extension:

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

The AES key will be encrypted using RSA and is stored in the file header. The details are to be elaborated in the next section.

RSA When WannaCry is executed, the Microsoft CryptoAPI is called to generate an RSA 2048-bit key pair which included a public key (pk_c) and private key (sk_c). The key pair (pk_c, sk_c) will be stored in the infected computer. pk_c will be stored directly as file "00000000.pky" while sk_c will be encrypted by another

RSA key pair (pk_s, sk_s), then stored as file "00000000.eky". The attacker keeps sk_s secret. In other words, the infected computer is unable to know the value of sk_s and decrypt "00000000.eky".

WannaCry uses pk_c to encrypt the 128-bit AES key of each file. The following shows the structure of an encrypted file.

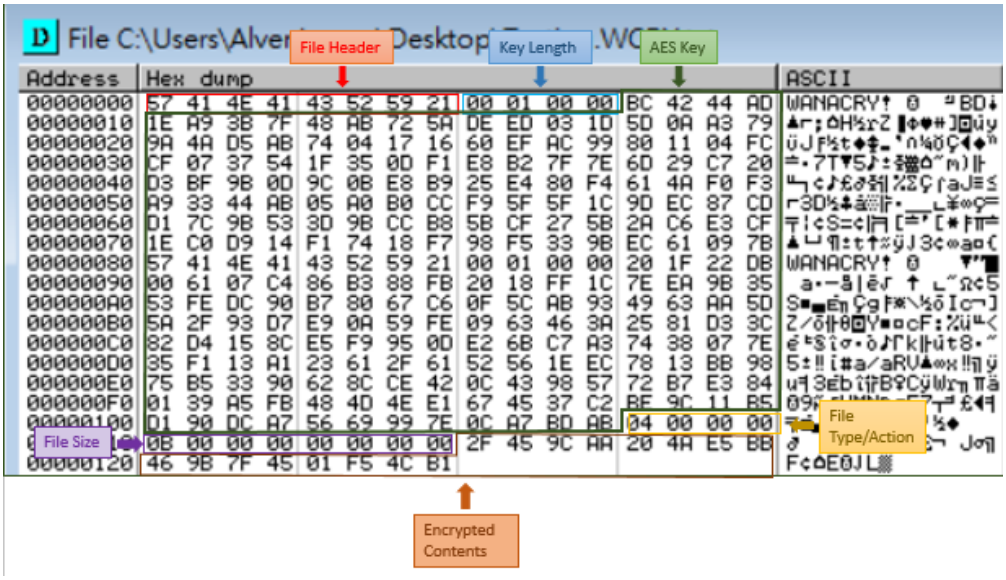

Figure 1: Encrypted File Structure

- File Header (8 bytes): ASCII codes of "WANACRY!"
- Key Length (4 bytes) : Length of RSA encrypted AES key in terms of bytes
- AES Key (256 bytes): Encrypted AES key using sk_c
- File Type/Action (4 bytes): File type internal to WannaCry
- File Size (8 bytes): Length of original file in terms of bytes
- Encrypted File Content: Encrypted file content using AES-CBC

2.2 Encryption Flow

The WannaCry encryption flow is shown below:

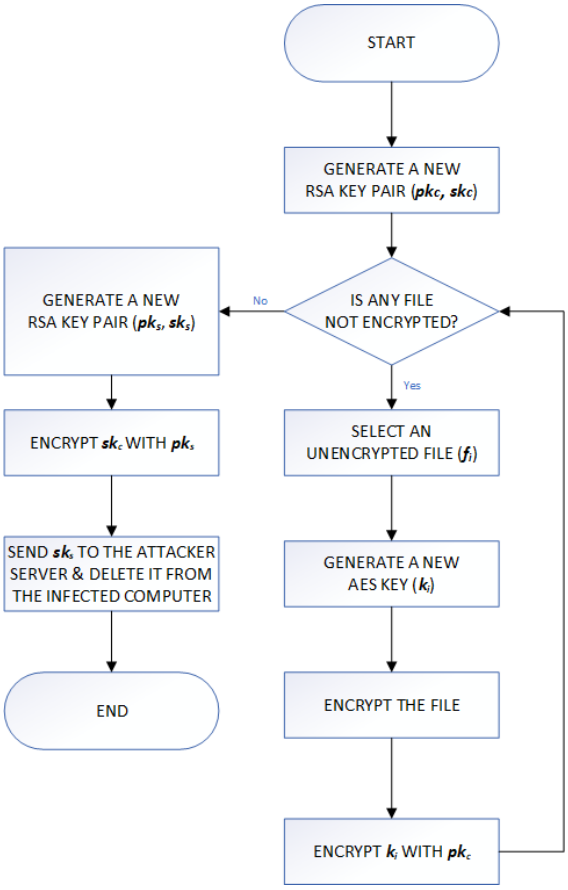

Figure 2: WannaCry Cryptography Flow Chart

WannaCry carries out the following steps in the infection process.

- Generate a 2048-bit RSA key pair (pk_c, sk_c).
- Perform the following steps for all targeted files:
  - Generate a new 128-bit AES key, k_i.
  - Encrypt file f_i using k_i in CBC mode.
  - Encrypt the k_i using pk_c.
- Generate a 2048-bit RSA key pair (pk_s, sk_s).
- Encrypt sk_c using pk_s.
- Send sk_s the attacker's server and delete it from the infected computer.

Remark: The program code of generating the AES keys and RSA keys is executed on the infected computer, which means that the intermediate values of generating the keys could be stored in the main memory of the system. By examining those memory locations, it is possible for us to retrieve those values and re-calculate the decryption key(s).

2.3 Generated Files

After WannaCry is triggered, the following files will be generated:

File Name	Content
00000000.pky	RSA Public Key
00000000.eky	RSA Private Key
00000000.res	Command & Control Communication Data
r.wnry	Additional decryption instructions used by the decryptor tool
s.wnry	ZIP archive containing Tor software executable
t.wnry	File encryption Dynamic-Link Library(DLL) for encrypting file as “WANNACRY!” encryption format
taskdl.exe	File deletion tool with temporary file cleanup program
taskse.exe	WannaCry Remote Desktop Protocol (RDP) session decryptor display tool
@Please_Read_Me@.txt	Ransom note
@WanaDecryptor@.exe	Decryptor
@WanaDecryptor@.exe.lnk	Decryptor shortcut

Figure 3: List of Exported Files

2.4 Breaking WannaCry

WannaCry encrypts all files in the system using AES. To decrypt the files, we must find out the AES keys. However, the AES keys are encrypted by an RSA public key. The only way to decrypt the AES keys is to find out the RSA private key.

There are two methods to find out the RSA private key:

Method 1. Access to (or hack into) the WannaCry server to get the RSA private key, sk_s. Then, use it to decrypt the client-side RSA private key sk_c. After that, locate the encrypted AES key in file, f_i, and recover the AES key, k_i, using sk_c.

Method 2. Re-calculate sk_c and recover all k_i. Let us review how the RSA keys are derived:

RSA Public Key: $pk_c = (e, n)$

RSA Private Key: $sk_c = d$

Relationship between e and d:

$$ed \ mod \ \phi(n) \equiv 1 \tag{1}$$

Euler’s Totient Function:

$$\phi(n) = (p-1)(q-1) \tag{2}$$

where p and q are two primes.

In fact, we can ignore the server side RSA private key, sk_s, and calculate the client side RSA private key, sk_c, by ourselves. We can easily find out n and e since the values are stored in file ”00000000.pky”. To calculate d, we have to know the factorization of n, such that $n = pq$. However, due to the difficulty of factorization of large integers, it is not easy to find out p and q from n, such that $n = pq$. However, the program code of generating RSA key pairs is running on the infected computer, and the program is using the infected computer’s main memory to store the variable values, so the user may be able to find out p and q by examining the main memory. Then we are able to calculate d for generating sk_c to decrypt the AES keys, k_i, to recover all the encrypted files.

3 Lab Tasks

In this lab, we will experience a simplified version of WannaCry. You are going to simulate a WannaCry attack and examine its behaviour. After completing the tasks, you will understand the various cryptographic algorithms adopted by WannaCry and be able to recover the encrypted file generated by WannaCry.

Remark: This simplified version of WannaCry is used for ACADEMIC PURPOSE ONLY.

PLEASE DO NOT modify the program and use it for malicious purposes.

Our simplified version of WannaCry has three major differences from the original one:

1. All the files are encrypted using the same 128-bit AES key.
2. A 32-bit RSA key is used to encrypt the 128-bit AES key. The 128-bit AES key will be divided into sixteen bytes. The 32-bit RSA key will encrypt the byte one-by-one and finally sixteen ciphertexts will be generated.
3. The encrypted AES key will be stored in a separate file named, "AES.WNCRY", which includes the file header, key length and the encrypted AES key. For the encrypted file such as "Testing.WNCRY", it will include the encrypted contents of the original file.

3.1 Initial Setup

- Open "Oracle VM VirtualBox".
- On the "VirtualBox Manager Window", select the "Wannacry_Lab" and click "Start".

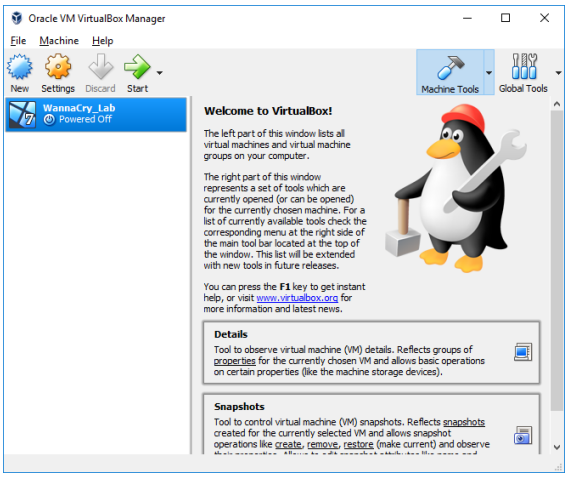

Figure 4: VirtualBox Manager Window

- A Windows 7 image should have booted up. You should see on the Desktop two executable files, "WannaCry.exe", and "DecryptFile.exe", and also a file folder called "UserFile" that includes two files, "TestingMe.txt" and "Music01.mp3".

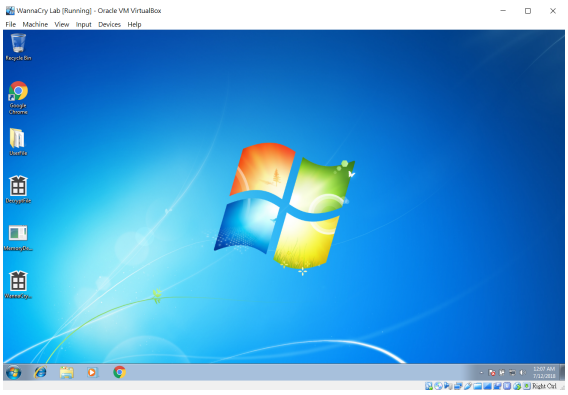

Figure 5: Image WannaCry Lab" interface

3.2 Encryption

Now, we will simulate a WannaCry ransomware attack. Make sure that "TestingMe.txt" and "Music01.mp3" are inside file folder "UserFile".

- Execute "WannaCry.exe". The following window should appear. DO NOT turn off the WannaCry simulation window!

Figure 6: WannaCry Simulation Window

- Open the folder "UserFile". What do you see?

We should not close the program because we have to find out the two distinct primes p and q from the main memory, to calculate the decryption key. If we close the program, p and q may be overwritten by other processes.

3.3 Memory Analysis

As mentioned in Sec. 2.4, we can search the main memory and locate the values of p and q and re-calculate d based on the public key (e, n). To find out (e, n), it is necessary to understand the structure of

"00000000.pky" which is a file generated by the WannaCry program and stored the content of pk_c. Indeed, file "00000000.pky" is a kind of PUBLICKEYBLOB-type file. The following is the structure of this file:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x06								0x02								0x00								0x00							
0x00								0xA4								0x00								0x00							
0x52								0x53								0x41								0x31							
bitLen																															
pubExp																															
modulus (variable)																															
...																															

Figure 7: Structure of PUBLICKEYBLOB file

The first 8 bytes are BLOB file header. The next 4 bytes are the RSA1 magic signature. It is followed by the bit length of the public key in 4 bytes. The next 4 bytes present the exponent e, and the remaining bits are the value of modulus n:

BLOB Structure Details:

- BLOB File Header (8 bytes)
  - Public Key Flag (1 byte) :
    The BLOB type of key (e.g., PUBLICKEYBLOB = 0x6, PRIVATEKEYBLOB =0x7)
  - Version Number (2 bytes) :
    Version of BLOB format
  - Reserved Flag (1 byte) :
    For future use
  - Key Exchange algorithm (4-bytes) :
    Structure of `ALG_ID`, included the message of Key Exchange algorithm
- Key Magic (4 bytes) :
  Algorithm identifier (e.g., RSA1 = 0x31415352, RSA2 = 0x32415352)
- Length of bits (4 bytes) :
  Length of modulus, which is a 32-bit unsigned integer in little-endian format
- Public Exponent (4 bytes) :
  Exponent e of public key, which is a 32-bit unsigned integer in little-endian format
- Modulus (Remaining bytes) :
  Modulus n of public key in little-endian format

[Task 1] Find out the values of exponent (e) and modulus (n) from "00000000.pky" and the two distinct prime numbers (p, q) in the main memory.

Step 1: Find e, n. Follow the steps below to find out the values of e and n:

- Open "OllyDbg".
- Click "File" and then "Open".
- Select "00000000.pky" from Desktop.
- Click "Open". Then, the following screen will be displayed.

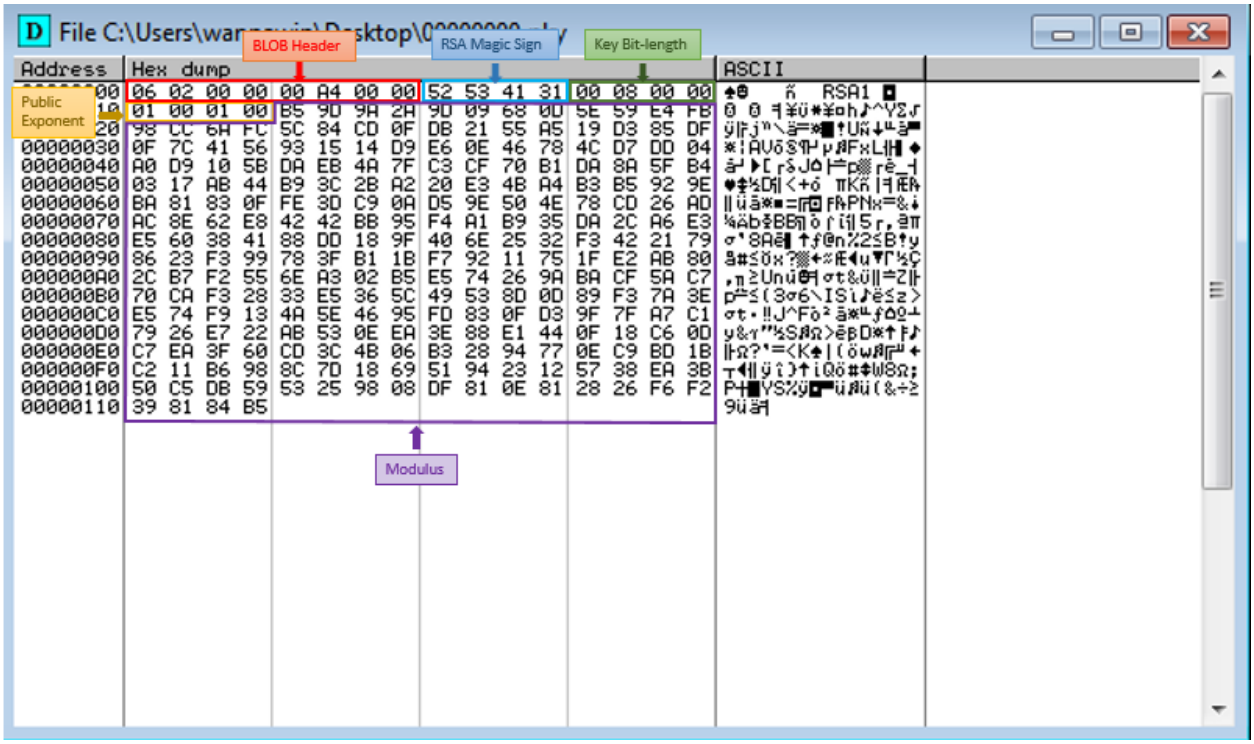

Figure 8: Sample content in "00000000.pky"

Remarks: The bit-length and exponent e are in little-endian format.

Step 2: Find p and q in the main memory According to Eq. 2, we know that:

1. p and q are prime numbers.
2. $p \neq q$.
3. $p \times q = n$.

[Task 2] WannaCry generates two random prime numbers to generate the client-side RSA public key (pk_c) and private key (sk_c). These two random prime numbers are stored temporarily in the main memory.

Complete the following C++ program to search the main memory for the values of p and q. [1 mark]

```
#include "stdafx.h"
#include <iostream>
#include <Windows.h>
#include <TlHelp32.h>
#include <string>
#include <cstdint>
#include <fstream>

using namespace std;
DWORD pid;
DWORD Temp = (int)0;
int NumValue;

int main() {
    HWND hWnd = FindWindowA(0, ("WannaCry V1.0"));
    GetWindowThreadProcessId(hWnd, &pid);
    HANDLE pHandle = OpenProcess(PROCESS_VM_READ, FALSE, pid);

    /* Prompt the user to enter the value of n, obtained in Step 1 */

    /* Assign n to a variable */

    //"140737488355327" is the end of memory address.
    while (Temp != 140737488355327){
     ReadProcessMemory(pHandle, (LPVOID)Temp, &NumValue, sizeof(NumValue), 0);

     /* NumValue is the value read from a memory address */

     /* Check and print out the values that can be p or q such that n = pq */

     Temp += 1;
    }
    system("pause");
}
```

Question: What are the values of e, n, p and q? [1 mark]

3.4 Decryption

[Task 3] After we have found out the values of p and q, we can calculate d which is the modular multiplicative inverse of e. The Extended Euclidean Algorithm can be used for the calculation. Write a C++/Java/Python program for this algorithm based on the pseudocode as shown below: [1 mark]

```
/* Pseudocode */
Specification:
Input: public exponent (e), modulus (phi_n)
Output: modular multiplicative inverse of e

\BEGIN

1. (A1, A2, A3) = (1, 0, phi_n);
   (B1, B2, B3) = (0, 1, e);
2. if B3 = 0
    return A3 which is GCD(phi_n, e) and there is no inverse;
3. if B3 = 1
    return B3 which is GCD(phi_n, e) and B2 which is the inverse of e;
4. Q = A3 div B3;
5. (T1, T2, T3) = (A1 - Q * B1, A2 - Q * B2, A3 - Q * B3);
6. (A1, A2, A3) = (B1, B2, B3);
7. (B1, B2, B3) = (T1, T2, T3);
8. goto 2;

\END
```

[Task 4] The modular multiplicative inverse d is used to decrypt the encrypted AES keys, k_i. A program has been written for you to decrypt the files. Now, follow the steps to recover the files.

1. Execute "DecryptFile.exe" on Desktop.
2. Enter the values of p, q and d.
3. If the entered values are correct, the file will be recovered successfully.

In the folder "UserFile", "TestingMe.WNCRY" and "Music01.WNCRY" will be changed back to "TestingMe.txt" and "Music01.mp3".

4 Submission

Complete Tasks 1, 2 and 3 [Total 3 marks]. You should submit the followings to Blackboard:

1. The source file of Task 2. The file must be named as "Task2_<your_name>_<student_no>.cpp". E.g., Task2_CHANTaiMan_12345678d.cpp.
2. The values of e, n, p and q. Provide screen shots to prove your claim. Save your work as "Task2_<your_name>_<student_no>.pdf"

3. The source file of Task 3. The file must be named as "Task3_<your_name>_<student_no>.<file_extension>". Note that the file extension depends on your chosen programming language.

Zip all files in a single file and name it as "Lab1_<your_name>_<student_no>.zip". Submit your work by $\mathbf{14^{th}\ Oct\ 2018, 23:59:00}$.

5 Challenging Questions [Optional]

[Challenge 1] Implement a program that decrypts the AES keys by using the value of n and d. Please be reminded that the simplified version of WannaCry is different from the real WannaCry program (Point 2 of Sec. 3). The following pseudocode may serve a starting point of your implementation:

```
/* Pseudocode */
Specification:
Input: An file of encrypted AES Key
Output: A recovered AES key

\BEGIN

myfile <- Encrypted AES key file;
Strcount <- 0;
count <- 0;
tempStr <- "";
AESStr[] <- "";

/* Step 1: Read encrypted AES key from file */

while (!myfile.eof()) {
    /* Read 4 bytes each time and assign it to a slot of a string array */
    if (count < 32){
        tempStr += read character;
        count += 1;
    } else {
        AESStr[Strcount] <- tempStr;
        Strcount++;
        count <- 0;
        tempStr <- "";
    }
}

/* Step 2: Decrypt the encrypted AES key using the exponent d and modulus n */

i <- 0;
AESKey <- "";

while (i < Strcount){
```

```
    AESKey += DecryptAES(AESStr[i]);
    i++;
}
return AESKey;

\END
```

[Challenge 2] Implement a program that recovers the files by using decrypted AES keys. Note that the mode of operation is CBC.

References

[1] Berry, A., Homan J., Eitzman R.. (2017, May 23). WannaCry Malware Profile. Retrieved from https://www.fireeye.com/blog/threat-research/2017/05/wannacry-malware-profile.html

[2] Blueliv. WannaCrypt Malware Analysis. Retrieved from http://www.blueliv.com/blog-news/research/wannacrypt-malware-analysis/

[3] Microsoft. [MS-MQQB]: PUBLICKEYBLOB. Retrieved from https://msdn.microsoft.com/en-us/library/ee442238.aspx

[4] Labs, L. (2017, May 16). A Technical Analysis of WannaCry Ransomware. LogRhythm. Retrieved from http://logrhythm.com/blog/a-technical-analysis-of-wannacry-ransomware/

[5] Security Response. (2017, May 15). [MS-MQQB]: Can files locked by WannaCry be decrypted: A technical analysis. Retrieved from http://medium.com/threat-intel/wannacry-ransomware-decryption-821c7e3f0a2b

[6] South West ComputAble. WannaCry?. Retrieved from http://computable.com.au/archives/1587

[7] Symantec. Ransom.Wannacry - SUMMARY. Retrieved from http://www.symantec.com/security-center/writeup/2017-051310-3522-99