THE EVOLUTION OF RANSOMWARE: FROM CRYPTOWALL TO CTBLOCKER

Christy Chung & Neo Tan
Fortinet Technology Inc., Canada

Email {cchung, ntan}@fortinet.com

ABSTRACT

The CryptoLocker ransomware was first discovered in late 2013. Millions of computers were infected, billions of files were encrypted, and millions of dollars' worth of ransom was collected within several months. It caught a lot of researchers' attention at that time, and it was finally isolated in late May 2014. After a few months' silence, a new variant, CryptoWall, appeared in late 2014. Millions of computers were infected within five months, and CryptoWall is still active now. Both CryptoLocker and CryptoWall propagated as attachments or malicious links through email messages. Once the computer was infected, the targeted files were encrypted and a payload popped up to demand a ransom. Sometimes files can be recovered after paying the ransom, but sometimes not. Compared to CryptoLocker, CryptoWall is stealthier as it uses the Tor network to host payment websites in order to avoid being tracked and discovered. It deletes the volume's shadow copies and disables the *Windows Error Recovery* screen at start-up to increase the difficulty of recovering files. It also disables *Windows Update* and error reporting in order to avoid detection.

On top of these, another piece of ransomware, called CTBLocker, was found in July 2014 right after CryptoWall was discovered. 'CTBLocker' is short for 'Curve Tor Bitcoin Locker'. The C, T and B are the three core components of the ransomware; respectively they represent Elliptical Curve (its encryption algorithm), Tor (its communication protocol) and Bitcoin (its ransom currency). It targets all current versions of *Windows*, such as *Windows XP*, *Windows Vista*, *Windows 7* and *Windows 8*. Compared with CryptoLocker and CryptoWall, CTBLocker is even harder to track, because it communicates with C&C servers through the Tor network instead of connecting directly to them.

We will compare CryptoWall and CTBLocker in terms of their communication methods, the way they select target files, and their encryption methods. Last but not least, we will demonstrate how to recover encrypted files.

INTRODUCTION

In recent years, many different kinds of ransomware have begun to spring up rapidly. Ransomware is a type of malware that encrypts target files with different encryption methods and requires a ransom to decrypt them so that they can be accessed again. From CryptoLocker to CTBLocker, the encryption methods are becoming more and more complicated. Payment schemes are also changing, from CryptoLocker asking for payment in real currency to CryptoWall and CTBLocker asking for ransom payments in bitcoins. Communication methods have also become more and more intricate. CryptoLocker used a Domain Generation Algorithm (DGA) to generate a list of C&C servers; CryptoWall uses the RC4 algorithm to encrypt all the communication between C&C servers, and requires payment of the ransom through the Tor network; CTBLocker communicates with its C&C servers through the Tor network. In this presentation we will look at the evolution of two of the currently most active pieces of ransomware, CryptoWall and CTBLocker, through their communication protocols, the way they select target files, and their encryption schemes.

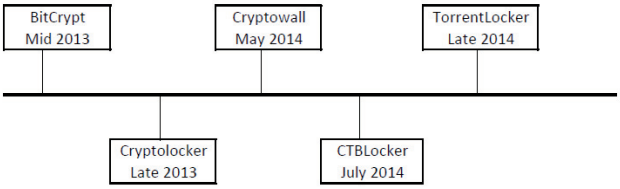

Figure 1: Timeline for crypto-ransomware.

SET-UP PHASE

CryptoWall first collects the victim's computer information, such as computer name, system drive, processor identifier, processor level and process revision. It then generates an MD5 hash from all of this computer information, which is used as an event object name to indicate whether the malware has already been running on the computer. If the hash is not found, a new explorer.exe process will be created and it will inject itself into it. Then the malware creates another process, svchost.exe, under explorer.exe and injects its payload into it. All of the communication, warning displays and encryption happens in within svchost.exe.

Meanwhile, CTBLocker injects itself into two processes: svchost (hidden) and explorer (GUI). The code injected into the svchost process is responsible for file encryption, and the code injected into the explorer process is responsible for contacting the C&C server, displaying the warning (GUI), and the decryption of the files according to the server response. The reason for making two separate processes is that, even if the user kills the warning GUI process, the encryption process will still be active – and if the user restores the files while the encryption process is active, the files will be encrypted again.

COMMUNICATION METHODS

CryptoWall creates another thread inside the svchost process which is responsible for contacting the C&C server, and for the encryption process. The malware generates an RC4 key that is sent to the C&C server as POST request. The server returns an encrypted message containing the onion address of the attackers' website, a user ID for the victim, country code and the public key that is used to encrypt the victim's files using RSA-2048.

The communications between the malware and the C&C server are encrypted with RC4. The malware generates a pseudorandom string with a variable length key, first using the key scheduling algorithm (KSA) to initialize the string and then using the pseudorandom generation algorithm (PRGA) to output a secret key that encrypts the victim's computer information and sends it to the hard-coded C&C server URL. Every compromised computer has its own unique RC4 key (see Figure 2).

```
00A00000  74 3D 30 65 34 64 30 33 39 39 31 38 36 36 61 63  t=0e4d03991866ac
00A00010  62 37 66 61 34 32 61 36 36 35 39 37 34 38 39 31  b7fa42a665974891
00A00020  64 64 30 39 64 63 66 66 64 37 64 62 65 63 35 65  dd09dcffd7dbec5e
00A00030  35 31 39 62 37 38 39 65 36 61 36 39 61 65 37 31  519b789e6a69ae71
00A00040  38 30 35 36 35 33 66 63 65 32 65 31 66 37 65 30  805653fce2e1f7e0
00A00050  38 31 36 30 30 34 64 35 34 34 33 63 65 38 65 34  816004d5443ce8e4
00A00060  63 38 36 63 00 00                                c86c..
```

Figure 2: RC4 encryption key.

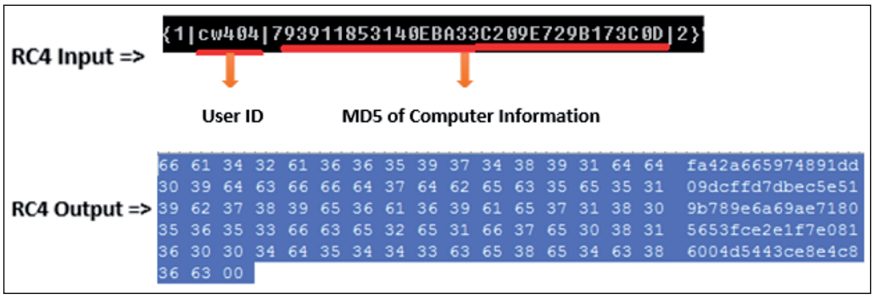

Figure 3: RC4 encryption for POST request.

```
00413938
00413938 loc_413938:
00413938 mov     eax, [ebp+lpParseOutVitmRecv]
0041393B push    eax             ; $ ==>     >00000004
0041393B                         ; $+4       >00CB0000  UNICODE "aLYk"
0041393B                         ; $+8       >00000016
0041393B                         ; $+C       >00CA0000  UNICODE "kpai7ycr7jxqkilp.onion"
0041393B                         ; $+10      >0000001B
0041393B                         ; $+14      >009C0000  UNICODE "kpai7ycr7jxqkilp.onion/aLYk"
0041393B                         ; $+18      >00000002
0041393B                         ; $+1C      >00CC0000  UNICODE "CA"
0041393B                         ; $+20      >0000002B
0041393B                         ; $+24      >009E0000  UNICODE "https://kpai7ycr7jxqkilp.enter2tor.com/aLYk"
0041393B                         ; $+28      >00000029
0041393B                         ; $+2C      >009F0000  UNICODE "https://kpai7ycr7jxqkilp.tor2web.org/aLYk"
0041393B                         ; $+30      >00000026
0041393B                         ; $+34      >00A00000  UNICODE "https://kpai7ycr7jxqkilp.onion.to/aLYk"
0041393B                         ; $+38      >00000000
0041393C call    Prep_ONION_URL
00413941 add     esp, 4
00413944 test    eax, eax
00413946 jz      loc_413A14

0041394C call    Prep_Letter_to_Victim
00413951 test    eax, eax
00413953 jz      loc_413A14
```

Figure 4: Concatenating onion addresses for paying ransom.

```
mov     ecx, [ebp+lpRecvPareseServerInformation] ;
                         ; $ ==>     >00000004
                         ; $+4       >00CB0000  UNICODE "aLYk"
                         ; $+8       >00000016
                         ; $+C       >00CA0000  UNICODE "kpai7ycr7jxqkilp.onion"
                         ; $+10      >00000000
                         ; $+14      >00000000
                         ; $+18      >00000002
                         ; $+1C      >00CC0000  UNICODE "CA"
                         ; $+20      >00000000
                         ; $+24      >00000000
mov     edx, [ecx+ServerSetInformation.StringCountryCode]
push    edx              ; edx:
                         ;       $ ==>     > 00CC0000  UNICODE "CA"
call    CRC32W
add     esp, 4
push    eax
call    FilterOutSpecialCountry ; "BY" = 0x9121D628 Belarus
                         ; "UA" = 0x87CECAE8 Ukraine
                         ; "RU" = 0xD2558852 Russia
                         ; "KZ" = 0xD9EA3CDB Kazakstan
add     esp, 4
test    eax, eax
jz      short loc_412039
call    RemoveRegEntry
call    RemoveFileAndRegEntry
```

Figure 5: Information received from the server.

The malware authors use the campaign ID to track the spreading method of the sample. For example, the sample we analysed has ID 'cw404', which indicates that the sample was spread by Cutwail. The ID, along with the MD5 hash of computer information that was generated earlier and the generated unsorted RC4 key, will be used for the POST request to the C&C server to indicate the infection status of the computer. Figure 3 shows that message before and after encrypting with RC4.

CryptoWall uses static onion addresses for payment of ransoms. After finishing preparing the onion URLs, the malware starts preparing the three different types of file which are 'DECRYPT_INSTRUCTIONS' that direct victims to pay the ransom once the target files and folders have been encrypted. Those file types are with extension 'TXT', 'HTML' and 'URL', respectively (see Figure 4).

With the information received from the server (Figure 5), the malware generates a CRC32 hash for the victim's country code and compares it against a list of specific country codes: BY, UA, RU and KZ, which correspond to Belarus, Ukraine, Russia and Kazakhstan, respectively. If the victim's country code matches one of the codes in the list, the malware proceeds to remove all the dropped files and registry entries that it has created. Otherwise, it starts to import the RSA key and encrypt the victim's files.

Figure 6 shows some of the C&C server URLs that are hard coded in the malware.

During its communication with the C&C server, CryptoWall sends back a public key which is encrypted with RC4. Figure 7 shows the decrypted public key in PEM format, which will be used for RSA-2048 encryption for all the targeted files and folders.

CTBLocker, meanwhile, injects two processes: svchost (hidden) and explorer (GUI). The code injected into the explorer process is responsible for contacting the C&C server, displaying the warning (GUI), and decryption of the files according to the server response.

```
loc_404F91:
mov     ecx, [ebp+idx]
push    ecx
call    Init_URL            ; $ ==>  _downloa.004201F8
                            ; $+4    ASCII "vivatsaultppc.com"
                            ; $+8    ASCII "milimalipali.com"
                            ; $+C    ASCII "torichipinis.com"
                            ; $+10   ASCII "covermontislol.com"
                            ; $+14   ASCII "bolizarsospos.com"
                            ;
                            ;
add     esp, 4
mov     [ebp+lpURL], eax
cmp     [ebp+lpURL], 0
jz      short loc_404FE5
```

Figure 6: Hard-coded C&C server URLs.

```
; lpOut:
$ ==>    >{280|kpai7ycr7jxqkilp.onion|aLYk|CA|-----BEGIN PUBLIC KEY-----..
$+40     >MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAr8xEaYLbb0HPudCYGoaz
$+80     >..i+KCUp/YxZKtO/zLomc8I6o+581JAp3x2MwmJfoqMHhI5NjmfC79TIHIhiqjag
$+C0     >zg..QvrdI+yURChoE5COcsLqVIip84QrrTw2cSeUyPur5TGrBjjS98fyhaP4NILt
$+100    >WRwC..1JXAmRNXj7Drzi0hEP0YCwBZ412PPjuWvB9qyJf3n3siPV+Qo7lpndRwXB
$+140    >M/nChQ..RZDuaZOAORLdPY6QR1FJGlm3pJrrT0H4FPmmcKuncqMGi4o/FkX/gIM/
$+180    >TwxqhC/H..k+0C7WeuQw76u09PiaoW2w5DqUuViAWsTa+OpttzX1DLYnqJv+kTHz
$+1C0     >6azgFQVGP9..8QIDAQAB..-----END PUBLIC KEY-----}..
```

Figure 7: RSA public key in PEM format.

```
00750837 xor     eax, eax

00750839
00750839 loc_750839:                ; "jssestaew3e7ao3q.onion"
00750839 mov     cl, byte ptr ds:aJssestaew3e7ao[eax]
0075083F mov     [esp+eax+2B70h+var_2B2F], cl
00750843 inc     eax
00750844 test    cl, cl
00750846 jnz     short loc_750839

00750848 push    edi
00750849 push    20h
0075084B lea     eax, [esp+2B78h+var_2B38]
0075084F push    eax
00750850 push    esi
00750851 call    ds:send
00750857 push    edi
00750858 push    8
```

Figure 8: Hard-coded onion URL.

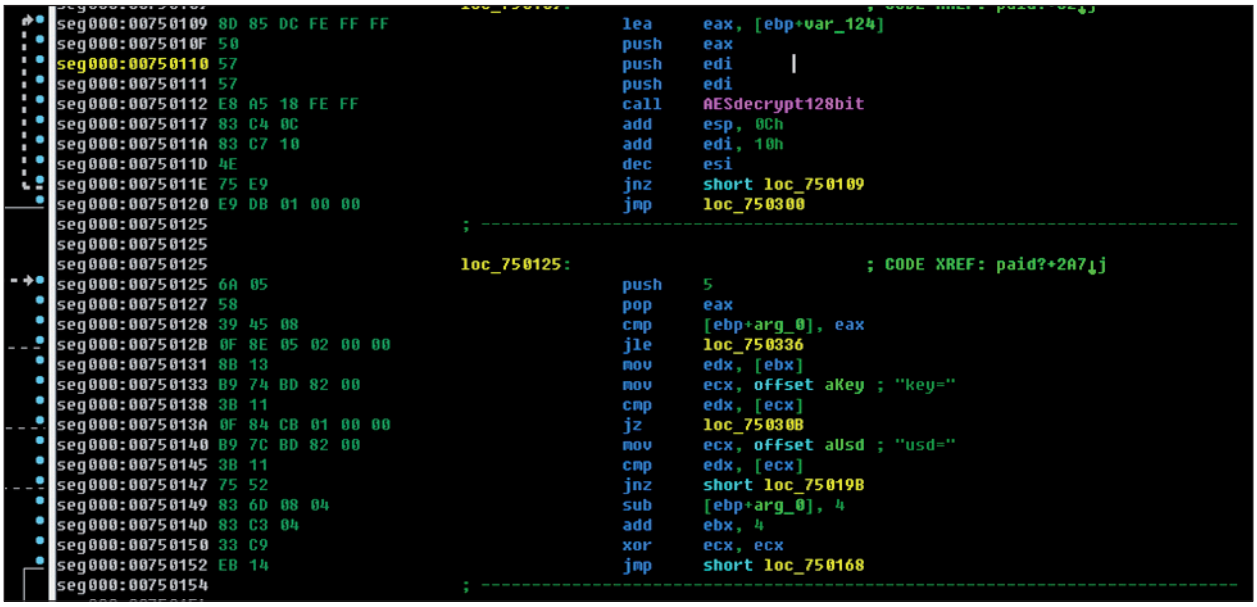

Figure 9: Parsing 'key' and 'usd' value from received data.

The C&C server list is static in the malware binary; they are onion domains. This makes it harder, but not impossible to trace the origin of the C&C server location. The malware links statically to the Tor library for the Tor procedures. As can be seen in Figure 8, it tries to contact the hard-coded onion URLs.

The communication uses both Tor and SSL protocols. On top of that, the message body is encrypted using AES and the ECDH (Elliptical Curve Diffie-Hellman) algorithm. ECDH is used to generate the AES key (session key). The encryption scheme is similar to that of the file encryption (which will be analysed in detail in a later section), except the server's public keys for traffic encryption and file encryption are different. The malware sends the generated client public key to the server to initiate the communication and in return receives in XML format the address of the bitcoin wallet, the price in bitcoins, as well as the price in the currency of the local machine. If the user pays the ransom, the server may send back the private key for the decryption of the files (we have seen the parsing and decryption process, which is able to use the key, but we cannot confirm whether this will actually happen). The screenshot in Figure 9 depicts the process of parsing the 'key' (decryption key) and 'usd' (ransom in USD) values returned from the server.

CryptoWall requires more of the victim's unique information than CTBLocker does during the C&C server communication – for example, user ID, computer information including Volume Serial Number, processor model, and computer name. Both CTBLocker and CryptoWall have chosen to store hard-coded C&C server/proxy URLs in the binary instead of using DGA or other dynamic techniques. CTBLocker even uses Tor as its client-server communication protocol. Newer versions may come with updated server lists. By doing this, they can avoid future C&C servers being sinkholed or blocked. Besides, they do not rely on constant communication with the C&C servers in the same way as normal botnets anyway.

SELECTING FILES AND FILE ENCRYPTION

CryptoWall looks for special folders which will be skipped. These include 'WINDOWS', 'Program Files' and 'Temp'. It then scans the system for all the mounted drives except CD-ROM drives. The purpose of skipping these folders and drives is to maintain the infected computer's normal operation. The malware also filters out files based on file name, and then filters files based on extension name. When a file is found, the file name is hashed into CRC32, which is then used to compare against a hash list. The hash list includes all the files which should not be encrypted.

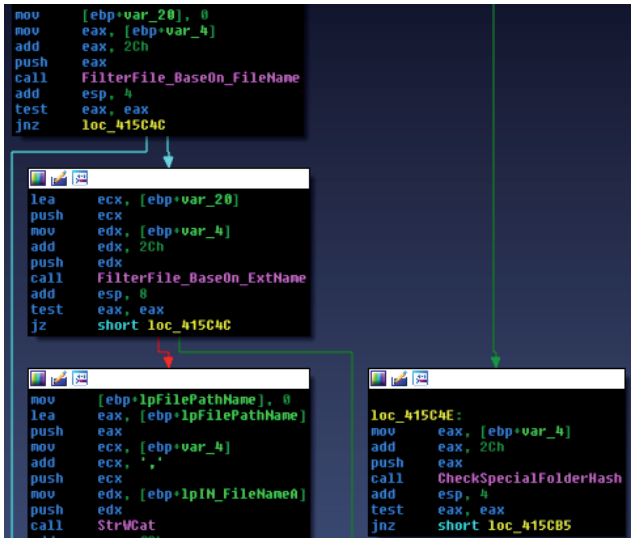

Figure 10: Choosing files/folders.

The target files are document files, graphics files, source files and video/audio files. Table 1 shows the full list of targeted file extensions and the difference between CryptoWall and CTBLocker.

Once the target files are found, they will be encrypted with RSA using the Wincrypt library, with the public key that is received from the C&C server. The malware filters out the encrypted files based on the first magic DWORD of the header. It saves the original file size and the encrypted file size into the header for later decryption purposes. It saves the

Common						Difference					
Doc/Text			Graphics		Images	Doc/Text		Graphics	Audio/Video		backup
7z	ods	wb2	3fr	mef	jpe	ach	pps	3dm	3g2	wav	back
abu	odt	wpd	ai	mrw	jpeg	db	prf	3ds	3gp	wmv	bak
accdb	p12	wps	arw	nef	jpg	des	qba	max	asf		**Source**
cer	p7b	xlk	bay	nrw	**Source**	eml	qbb	obj	asx		class
crt	p7c	xls	cdr	orf	c	key	qbm	ps	avi		dtd
dbf	pdd	xlsb	cr2	pef	cpp	lif	qbr	pspimage	flac		fla
der	pdf	xlsm	crw	psd	cs	msg	qbw	sr2	flv		h
doc	pem	xlsx	dcr	ptx	pas	nd	qby	svg	m4v		hpp
docm	pfx		dds	r3d	pl	nk2	tex	tga	mov		java
docx	ppt		dng	raf	py	oab	thm	x3f	mp3		lua
indd	pptm		dwg	raw	**system**	odc	tig	yuv	mp4		m
mdb	pptx		dxf	rw2	config	ost	wallet		mpg		
mdf	pst		dxg	rwl		pab	xlr		rm		
odb	rtf		eps	srf		pages			srt		
odm	sql		erf	srw		pct			swf		
odp	txt		kdc			pdb			vob		

Table 1: Targeted file extensions.

```
loc_413D01:
mov     [ebp+var_8], 0
lea     eax, [ebp+var_8]
push    eax
call    PrepareSoftwareString ; 009A0000  UNICODE "software\793911853140EBA33C209E729B173C0D"
add     esp, 4
test    eax, eax
jz      loc_413E18
```

Figure 11: Created registry key.

encrypted file with a random extension appended to the original, preserving the original file information, and then deletes the original file. The malware also creates a full path of the encrypted file to the machine user registry key, which is under 'software\[MD5 hash of computer information]\CRYPTLIST' (see Figure 11).

The encrypted file structure is as shown in Figure 12.

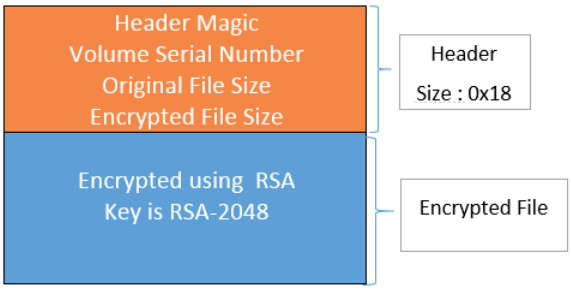

Figure 12: Encrypted file structure for CryptoWall.

In the header, the first four bytes are the magic number, which is chosen randomly from one of the four static magic DWORDs. The next DWORD is the CRC32 hash of the Volume Serial Number of the computer; the following two DWORDs are the original file size and encrypted file size; and the last eight bytes are reserved.

```
struct Header
{
        Int32    magic;   //"0x0c70defe","0x00defece"
,"0x0c0d3fc3","0x00f3d3f4"
        int32    crc32Hash;
        int32    originalFileSize;
        int32    encryptedFileSize;
}
```

In addition to encrypting the files, the malware deletes the volume's shadow copies and disables the *Windows Error Recovery* screen at start-up in order to increase the difficulty of recovering files. The malware also attempts to disable the wscsvc, WinDefend, wuauserv, BITS, ERSvc and WerSvc services in order to reduce the computer's security level. Furthermore, the malware disables *Windows* updates and error reporting in order to avoid detection.

For CTBLocker, the encryption process will start regardless of whether the other process can successfully contact the C&C server.

CTBLocker gets the current *Windows* serial number from the registry: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\MachineGuid and then hashes it into SHA256 form. This is the key used to encrypt and decrypt local configuration files. The encryption algorithm is AES with a 128-bit key. The local configuration file is saved under %AppData%\[randomfilename] (in the analysed example it is %AppData%\vbrxptc). The configuration file contains a flag indicating whether the system is already infected, a flag for decryption, a holder structure for shared keys for the decryption of the five demo files, a holder structure for the server's private key, a C&C server communication shared key and the C&C server's onion URL (i.e. jssestaew3e7ao3q.onion). The holders are initially empty. There are threads that check the flags and the holders periodically.

The target files are mostly text, photo or data storage files; a full list of targeted file extensions is shown in Table 2.

The malware has one thread which is responsible for encrypting all the targeted files, and another thread which drops the 'helper' files, indicating that the files are encrypted

Documents/Text related				Graphics			Source	system	misc		Audio/Video	database
7z	abu	accdb	blend	3fr	dwg	mrw	bas	config	bdcu	gsd	arp	gdb
cer	crt	dbf	der	ai	dxf	nef	bdcr		bdd	ims	gsf	
doc	docm	docx	indd	arw	dxg	nrw	c		bdp	iss		
md	mdb	mdf	odb	bay	eps	orf	cpp		bds	kwm		
odm	odp	ods	odt	cdr	erf	pef	cs		bpdr	pwm		
p12	p7b	p7c	pdd	cr2	jpe	psd	js		bpdu	rgx		
pdf	pem	pfx	php	crw	jpeg	ptx	pas		bsdr	rik		
ppt	pptm	pptx	pst	dcr	jpg	r3d	pl		bsdu	safe		
rar	rtf	sql	txt	dds	kdc	raf	py		dbx	vsd		
wb2	wpd	wps	xlk	dng	mef	raw			dd			
xls	xlsb	xlsm	xlsx	rw2	srf				fdb			
zip				rwl	srw				groups			

Table 2: CTBLocker targeted file extensions.

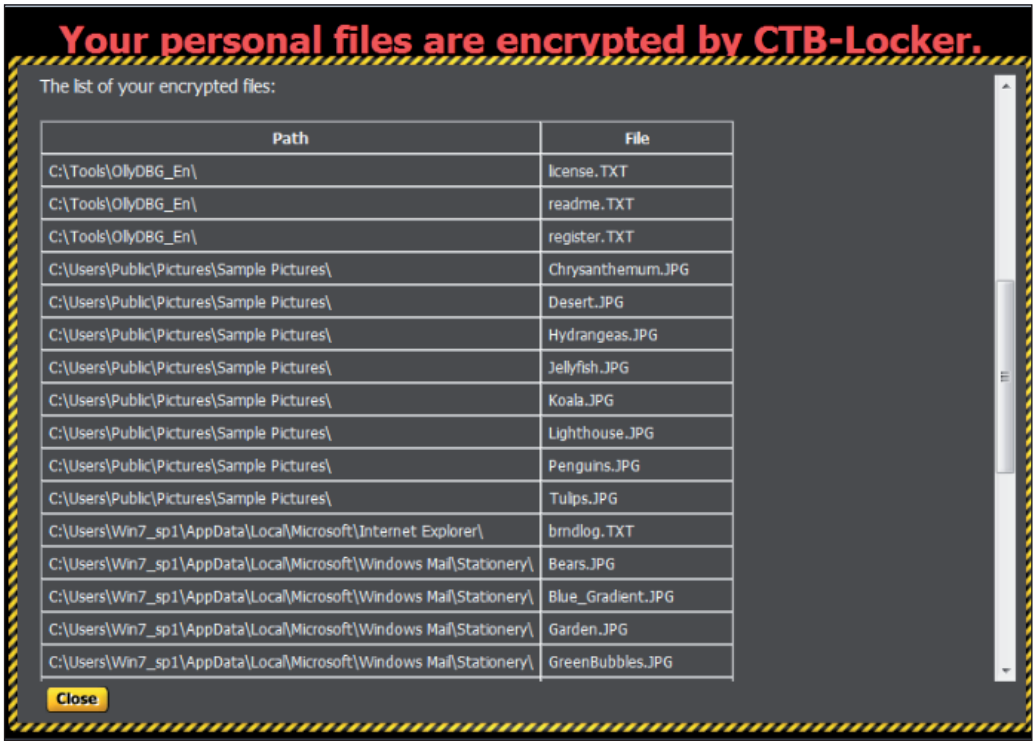

Figure 13: Table of encrypted files.

and demanding a ransom for them, in the same folder as the targeted files. The 'helper' file claims that the malware is using the 'strongest encryption and unique key, generated from this computer', as well as providing instructions as to how to pay the ransom. The process involves accessing an onion URL, copying and pasting some public key to that site, and paying bitcoins to a certain address.

For the encryption, it moves the original file to a temporary folder or the current folder under the name [randomString].tmp (the string is generated using the same algorithm as used for the configuration file name but with a different seed). The files (system protected files or files that are being used) that the malware doesn't have permission to move will be passed over. The file creation time is set to the original, then the file is moved back to its original place with an extension name appended to it. The extension (i.e. didocgl) is generated using the same algorithm as the configuration file name but with a different static seed. This is also a marker for the malware to find and decrypt them later if needed.

```
0263486C call    sha256_0
02634871 push    7
02634873 xor     eax, eax
02634875 pop     ecx
02634876 mov     [ebp+50h+basePoint], 9
0263487D lea     edi, [ebp+50h+var_103]
02634883 rep stosd
02634885 stosw
02634887 stosb
02634888 lea     eax, [ebp+50h+basePoint]
0263488E push    eax
0263488F lea     eax, [ebp+50h+clientSecret]
02634895 push    eax
02634896 lea     eax, [ebp+50h+clientPublic]
02634899 push    eax
0263489A call    curve
0263489F push    offset serverPublic
026348A4 lea     eax, [ebp+50h+clientSecret]
026348AA push    eax
026348AB lea     eax, [ebp+50h+sharedKey]
026348B1 push    eax
026348B2 call    curve
026348B7 lea     eax, [ebp+50h+AESkey] ; output
```

Figure 14: Generating the client's public key.

The file is deflated using zlib then encrypted using AES. It appends a 0x30 byte size header to the encrypted file. AES is symmetric encryption, however the algorithm used to obtain the AES key is not. The key the AES encryption uses is calculated using the ECDH (Elliptic Curve Diffie-Hellman) function. In the code shown in Figure 14 you can see it uses the constant basepoint 9 followed by all zeros to generate the client's public key.

Suppose we call the AES key the sharedKey, it behaves like a session key and is generated using the server's public key and the client's private key. The client's public-private key pair is generated randomly for each file. After each encryption the sharedKey is discarded and only the generated client's public key is saved into the header. Therefore, at this moment, it has to use the server's private key to re-generate the shared key for the file decryption. The encrypted file structure is shown in Figure 15.

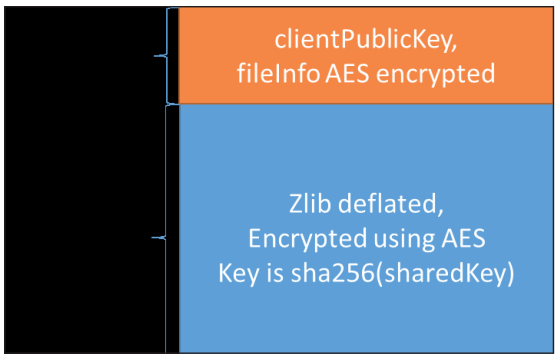

Figure 15: Encrypted file structure for CTBLocker.

In the header, the first 0x20 bytes are the client's public key, and the next 0x10 bytes structure looks like the following:

```
struct fileInfo
{
      Int32    magic;      //"CTB1"
      int32    originalFileSize;
      int32    moddedFileSize;
      int32    aFlag;      //always 1 for non-demo files
}
```

The fileInfo structure is also encrypted using AES with the same sharedKey.

The GUI process has a thread that keeps reading the local configuration file every 1,000 milliseconds. Once the private key is received and the decryption flag is set in the configuration file, the GUI process will start the decryption. As already described, in order to obtain the shared key for AES decryption, it needs the client's public key, which is stored in each file's header, as well as the server's private key.

However, the malware provides a demo decryption of five files for the victim in order to prove that it can decrypt the files. This demo can work offline without contacting the C&C server, which can give the victim false hope that the other files can also be decrypted without the server's private key. In fact, this is because the five shared keys were saved into the local configuration file. The malware has a method for finding those five specially encrypted files. Figure 16 is a screenshot of the five saved shared keys and the list of the five files loaded in memory.

According to Tables 1 and 2, there are differences in the files targeted by CryptoWall and CTBLocker, but they both mainly target text, photo or data storage files. Although CryptoWall makes more of an effort after the file encryption to ensure the original files are truly un-recoverable, compared to CryptoWall using RSA to encrypt the entire file, CTBLocker's method for encrypting the files provides the

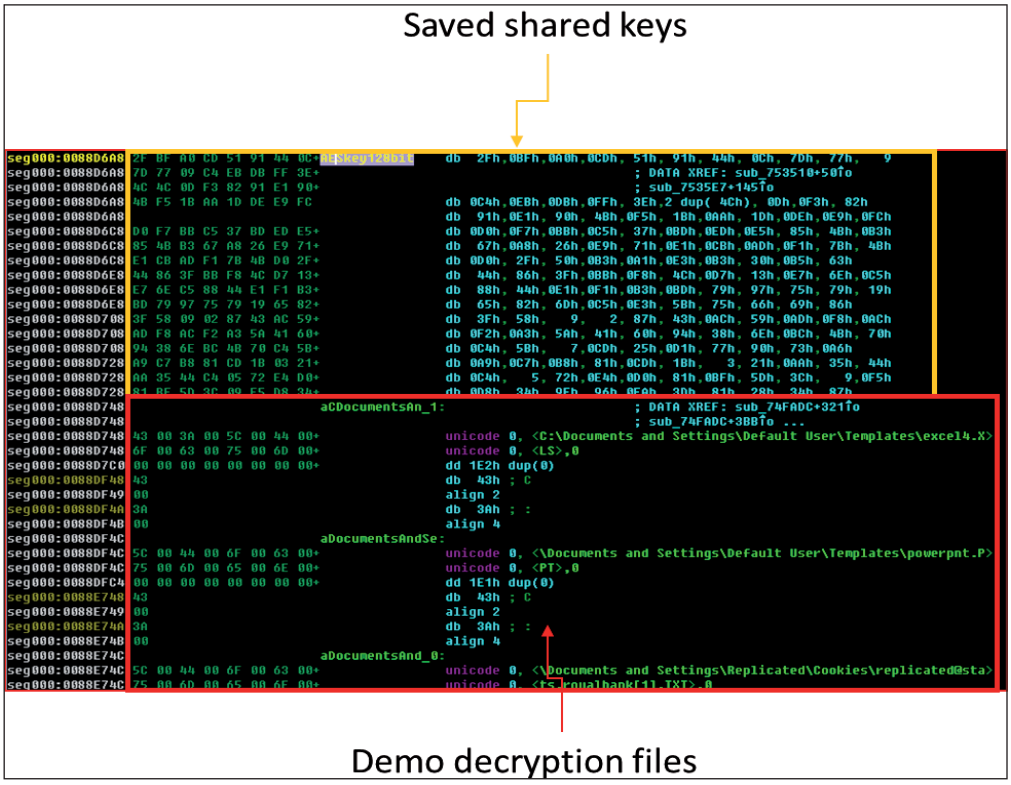

Figure 16: Shared keys for five encrypted files.

same level of crypto security (they both require the server's private key for decryption), yet better encryption and decryption efficiency since the file encryption is actually using AES. Moreover, CTBLocker's encryption process cannot be prevented by blocking communication with the C&C servers, whereas CryptoWall's can, because it requires the initial contact with the server in order to get the encryption key to start the encryption process. We have seen some of the latest variants of CTBLocker starting to employ functions to delete volume shadow copies and backups. All of these characteristics make CTBLocker a nastier proposition.

CONCLUSION

In this paper we have analysed the differences between two pieces of ransomware in terms of their encryption and communication. We believe that more ransomware variants will appear and that they will not just target European and North American countries, since this is a quite a profitable 'business model'. So far, for both pieces of ransomware analysed, once the files have been encrypted, there is no way to decrypt them unless the private key is retrieved from the server. In order to protect our computers from these pieces of ransomware, and should an infection occur, there are several things that we can try:

1. Back up the computer more often: once the files are encrypted, try to restore the computer from the most recent back-up.
2. Restore the encrypted files via Volume Shadow Copies (VSC): this method requires the VSC service to be active before the infection and may work for CryptoLocker and CTBLocker, but not for CryptoWall as the malware removes the Volume Shadow Copies quietly at the end.
3. Try file recovery software, since the malware calls ZwDeleteFile eventually to remove the original files.