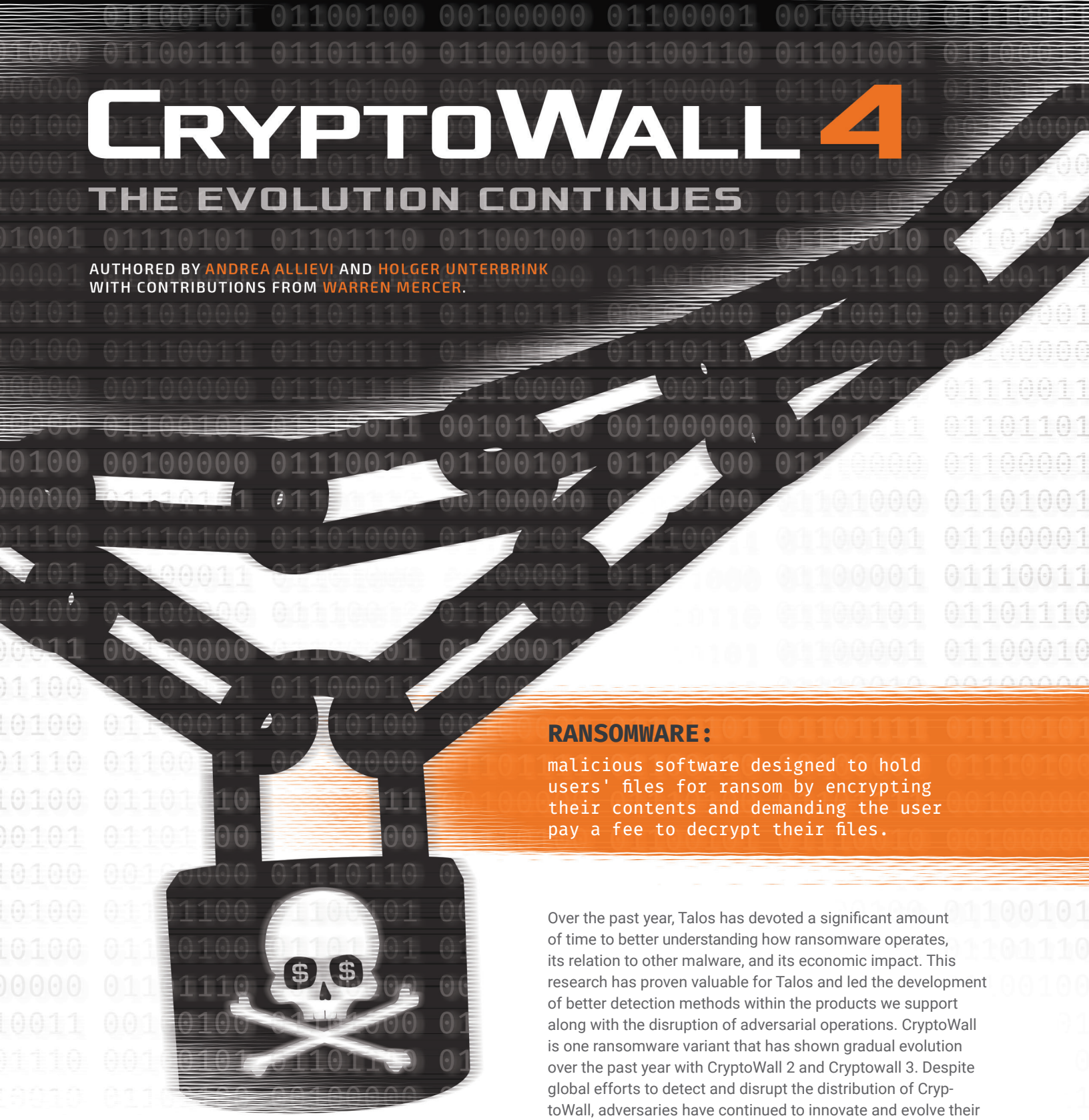
CryptoWall 4

THE EVOLUTION CONTINUES

AUTHORED BY ANDREA ALLIEVI AND HOLGER UNTERBRINK
WITH CONTRIBUTIONS FROM WARREN MERCER.



RANSOMWARE:

malicious software designed to hold users' files for ransom by encrypting their contents and demanding the user pay a fee to decrypt their files.

Over the past year, Talos has devoted a significant amount of time to better understanding how ransomware operates, its relation to other malware, and its economic impact. This research has proven valuable for Talos and led the development of better detection methods within the products we support along with the disruption of adversarial operations. CryptoWall is one ransomware variant that has shown gradual evolution over the past year with CryptoWall 2 and Cryptowall 3. Despite global efforts to detect and disrupt the distribution of CryptoWall, adversaries have continued to innovate and evolve their craft, leading to the release of CryptoWall 4. In order to ensure we have the most effective detection possible, Talos reverse engineered CryptoWall 4 to better understand its execution, behavior, deltas from previous versions and share our research and findings with the community.

DESIGN & ILLUSTRATIONS BY MELISSA TAYLOR

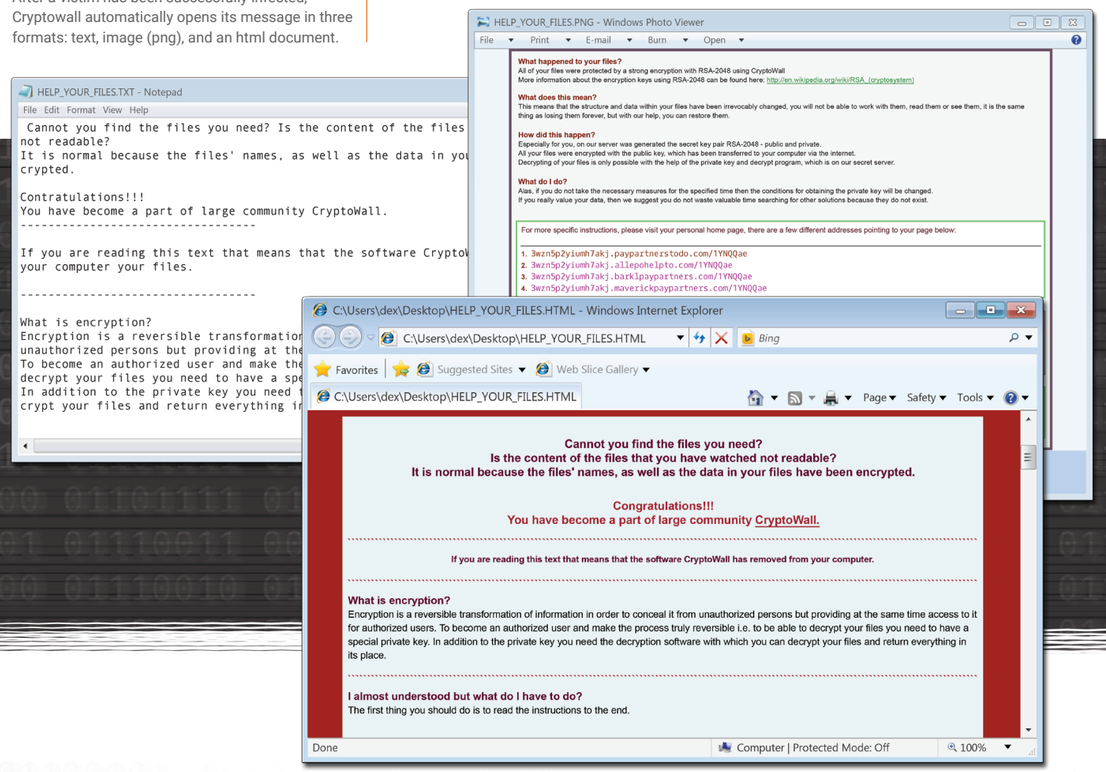
BACKGROUND

For readers that may not be familiar, ransomware is malicious software that is designed to hold users' files (such as photos, documents, and music) for ransom by encrypting their contents and demanding the user pay a fee to decrypt their files. Typically, users are exposed to ransomware via email phishing campaigns and exploit kits. The core functionality of CryptoWall 4 remains the same as it continues to encrypt users' files and then presents a message demanding the user pay a ransom. However, Talos observed several new developments in CryptoWall 4 from previous versions. For example, several encryption algorithms used for holding users' file for ransom have changed. Also, CryptoWall 4 includes a new technique to disable and delete all automatic Windows backup mechanisms, making it almost impossible to recover encrypted files without having an external backup. Finally, CryptoWall 4 has been observed using undocumented API calls not previously used to find the local language settings of the compromised host. These are just a few of the new findings Talos observed in the new iteration of CryptoWall that are detailed further in this paper.

For our technically savvy users, we encourage you to continue reading. As always, we strongly encourage users and organizations to follow recommended security practices and to employ multiple layers of detection in order to reduce the risk of compromise. Our in-depth analysis of the latest CryptoWall version gives us a better opportunity to protect our users by allowing us to identify better detection methods. Though it is ultimately up to you to decide whether to pay the ransom to recover your data, Talos strongly encourages users to **not** pay the ransom as doing so directly funds this malicious activity. Additionally, you should always contact law enforcement.

FIGURE A1.

After a victim has been successfully infected, Cryptowall automatically opens its message in three formats: text, image (png), and an html document.

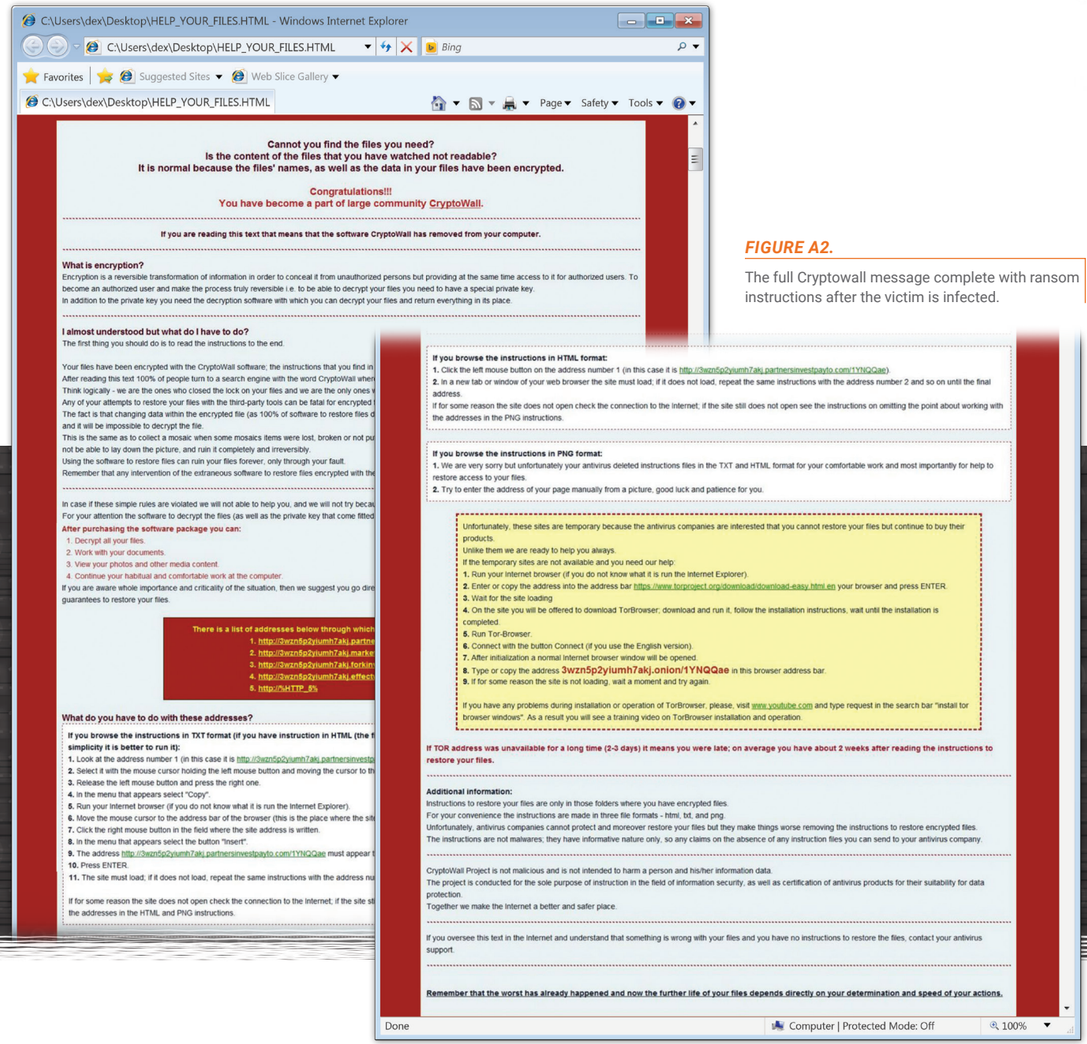


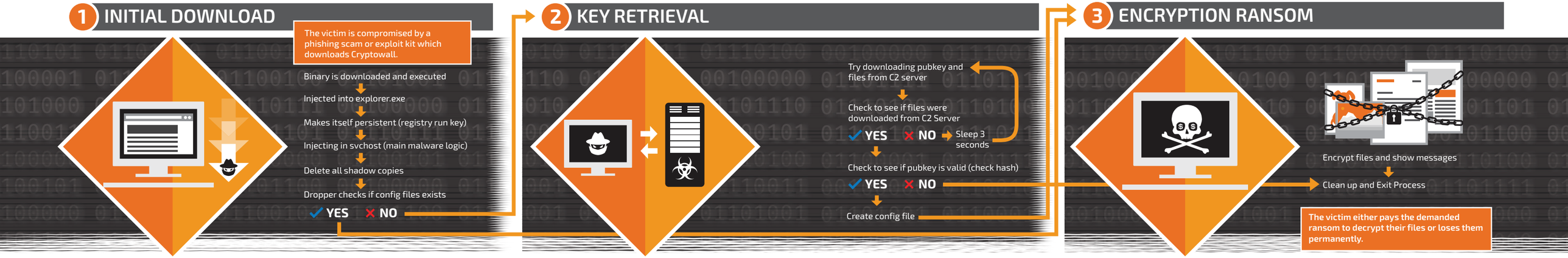
INFECTION PROCESS

The adversaries behind CryptoWall 4 are using phishing and drive-by-download campaigns to distribute their malware to their victims. Once CryptoWall 4 has been successfully executed, the dropper downloads an RSA public encryption key from the command and control (C2) server. All files are encrypted with a temporary AES encryption key, which is later encrypted with the downloaded RSA public key and embedded in the encrypted files. Then it shows its message in three different formats to ensure the victim recieves at least one version of the instructions even if the other formats are blocked by antivirus software. The first is a text file, the second is an image (.png format) and thirdly an HTML document. All are automatically opened on the victim's desktop as shown in Figure A.1 on the previous page. Figure A.2 below shows the full text of the HTML document.

FIGURE A2.

The full Cryptowall message complete with ransom instructions after the victim is infected.





One interesting observation with CryptoWall 4 is if it cannot retrieve the public RSA encryption key from the C2 server it will enter a loop constantly attempting to download the public key. As long as the key cannot be obtained, CryptoWall will not "harm" the victim's computer. Talos also observed that this sample exhibited some additional safety checks, such as terminating the infection process if the local language is not supported. The following is a non-exhaustive list of some of the unsupported languages: **Russian, Kazakh, Ukrainian, Uzbek, Belarusian, Azeri, Armenian, Kyrgyz, Georgian.**

This clearly shows the adversaries want to exclude certain regions from infection.

The technical infection process is depicted in Figure B at the top of the page.

All functions starting from "Delete all shadow copies" in the above diagram are executed from the CryptoWall 4 code injected in the svchost host process. Injection into this process is increasing the privileged level of access to the compromised machine; this allows the deletion of all available shadow copies without the end user being prompted with the UAC (User Account Control) dialog to 'Approve' the deletion if the user has administrator level access rights.

The network communication is described in Figure C. The network communication is using HTTP, but with an encrypted payload, see Figure D and Figure E.

FIGURES D & E

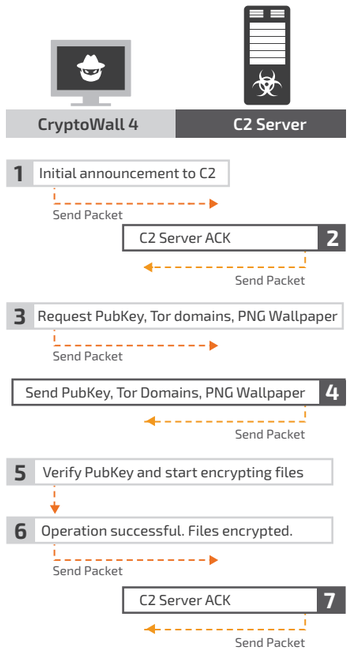


FIGURE C.

CryptoWall 4 network communication process with C2 Server.

FIGURE B (Above):

Technical infection process for CryptoWall.

CryptoWall 4 is using a **new file name generation algorithm** which is used for the encrypted files, which works as follows:

1. Scan hard drive for directories, skip excluded directories.
2. Get original file from directory, skip excluded filename and extensions.
3. Generate random value for the filename string size between 5 and 10.
4. Build a filename string with the length of this random size out of the character set 'a-z'. (Get random value between 0-1000, do a MOD 26 and convert it to an ASCII character).
5. Null terminate the filename string.
6. Get a random number between the half of the string and the string size
7. Get a random number between 1 and this random number from Step 6 (This number is used as value for how many random numbers are inserted into the string in the next step)
8. Generate a random ASCII number (char) between 0-9 and insert it at a random position in the string
9. Do step 8 as many times as the random value in step 7 says.
10. Use the same algorithm to generate the extension, but with a min. size of 2 characters and max. size of 5.
11. Append the extension to the file name string.

CryptoWall 4 is using CRC32 checksums to exclude some directories, filenames and extensions. Table 1 provides a list of what is excluded from encryption.

EXTENSIONS		DIRECTORIES	FILES
exe	com	windows	help_your_files.txt
dll	hta	temp	help_your_files.html
pif	cpl	cache	help_your_files.png
scr	msc	sample pictures	thumbs.db
sys	bat	default pictures	
msi	cmd	Sample Music	
msp	scf	program files	
		program files (x86)	
		games	
		sample videos	
		user account pictures	
		packages	

TABLE 1:

Excluded filenames. A more in depth list can be found in Appendix A.

We believe these directories, files and file extensions are avoided to ensure stability of the operating system, this means the compromised user can still use their machine to pay the ransom. Any infected user should remember that if persistence is successful the encryption function will run again on the next reboot to encrypt any files the user created after the initial infection.

After generating the names for the new files, it starts the encryption algorithm shown in Figure F on the right of the page. This also clearly shows there is no way back after CryptoWall 4 has encrypted the files, without having the private key to decrypt the temporary AES encryption key. Unfortunately, for the victim, this only exists on the attacker side and is never transmitted to the victim's computer. In other words, there will be no viable way to recover the files without paying the ransom or getting the private key from the adversaries in some other way, such as law enforcement seizing their infrastructure. Users are encouraged to have backups of important files to ensure they can sufficiently recover from these types of attacks without having to pay any ransom.

TECHNICAL DETAILS

THE DROPPER

The dropper is compressed and encrypted with different customized packers, full of useless code, API calls, and AV anti-emulation tricks such as calling random APIs with strange parameters to confuse the emulation engines. See Figure G for a snapshot of this.

The second stage uses a form of Spaghetti code implemented in the following way:

```
INSTRUCTION 1
INSTRUCTION 2
JNO nextStep
```

THE DECOMPRESSED CODE

The main code is very similar to the old iteration of CryptoWall: the malware builds its Import Address Table (IAT), obtains all the needed system information and creates its main event object for process synchronization (the name is generated from the workstation information MD5). This event has 2 goals: prevent any other CryptoWall 4 infection starting during execution and synchronization between the different involved processes.

The code is injected in a newly spawned "explorer.exe" process. The actual code is written in the target process using one of two different techniques:

1. ZwCreateSection and ZwMapViewOfSection native APIs
2. ZwAllocateVirtualMemory, ZwWriteVirtualMemory and ZwProtectVirtualMemory native APIs

At the end the code is properly relocated and another 2 different techniques are employed for the actual code injection:

1. ZwQueueApcThread internal API used to queue an APC in the target process.
2. The classical CreateRemoteThread method

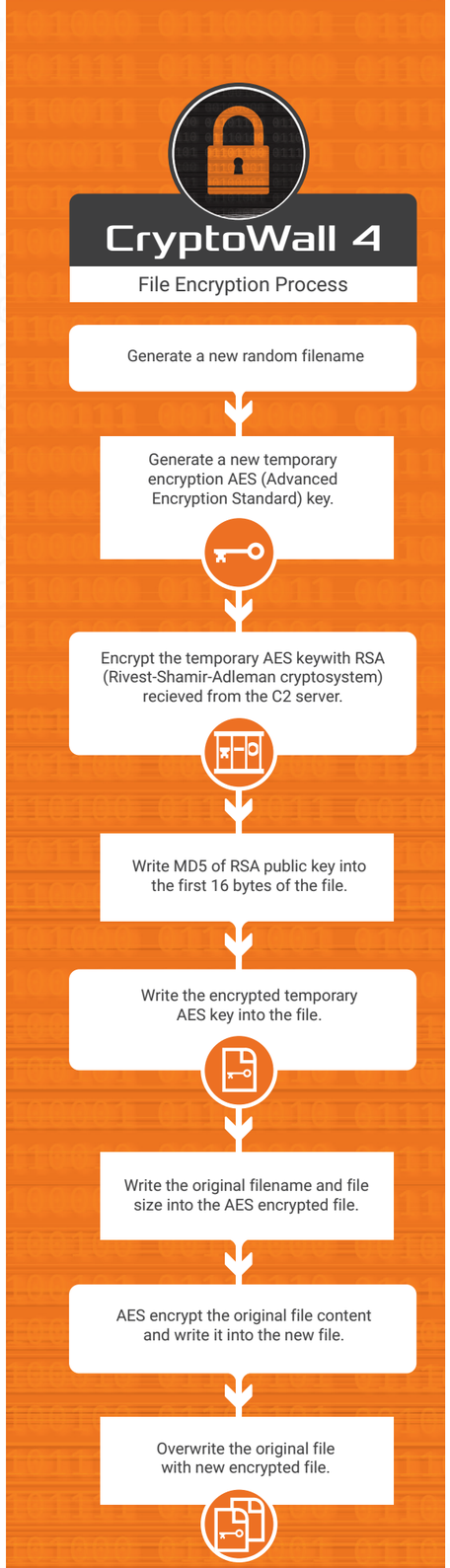
FIGURE F. (RIGHT)

Cryptowall's process for generating names for the new encrypted files.

```
push    0               ; dwInitParam
push    0               ; lpDialogFunc
push    0               ; hWndParent
push    0               ; lpTemplateName
push    0               ; hInstance
call    ds:DialogBoxParamW
```

FIGURE G.

Cryptowall's dropper utilizes a variety of obfuscation methods including useless code (as shown above) and anti-emulation tricks.



The code that resides in the newly spawned "explorer.exe" process (see Figure B) has the ultimate goal to execute and infect the system with CryptoWall 4 and obtain persistence on the machine. The dropper is copied in the %APPDATA% folder and a registry value is created inside the standard "Run" key of the local user registry root path.

The dropper tries to disable all the System Restore Points and Windows Shadow Copies using a novel method previously unseen: it calls the SRRemoveRestorePoint API, with an index from 0 to 1000, until it returns ERROR_INVALID_DATA. It writes the value named "DisableSR" (set to 1) to the "HKLM\Software\Microsoft\Windows Nt\SystemRestore" registry key aiming to completely disable the Windows System Restore. Finally it launches the standard command to delete the Volume shadow Copies:

```
vssadmin.exe Delete Shadows /All /Quiet
```

Execution continues in the "svchost.exe" process, The code re-builds the IAT, creates another event object (used only for the "svchost.exe" instance), and then tries to generate and open its configuration file. More on the configuration file later. At this point the routine fails because the configuration file doesn't exist yet. The dropper opens and decompresses the C&C URL list located inside itself (compressed with an LZ compression algorithm) and finally it tries to connect to one of its C&C servers using an announcement packet.

You can find a list of the C&C servers found in the analyzed dropper in the IOC section.

The CryptoWall 4 network packet is specifically crafted as below:

```
|<request Id>|crypt7|<workstation MD5>[|subRequest Id
1|subRequest 1 Data| … ]
```

At the time of writing, we have successfully isolated five different network packet types (request ID):

- **1, 3** - Announcement packet - used to communicate to the C&C server that a new victim has been infected
- **7** - Multi purpose packets. The first sub-request ID differentiates packets:
  - **1** - Public key request - used to ask to the C&C server for a new public key (and the language-dependant PNG wallpaper) to properly encrypt all the files
  - **2** - End announcement packet - used to communicate that an infection is ended. Another sub-request ID defines more precisely how the infection is ended:
    - **1** - Success
    - **2, 3** - Unsupported OS language packet - Exit

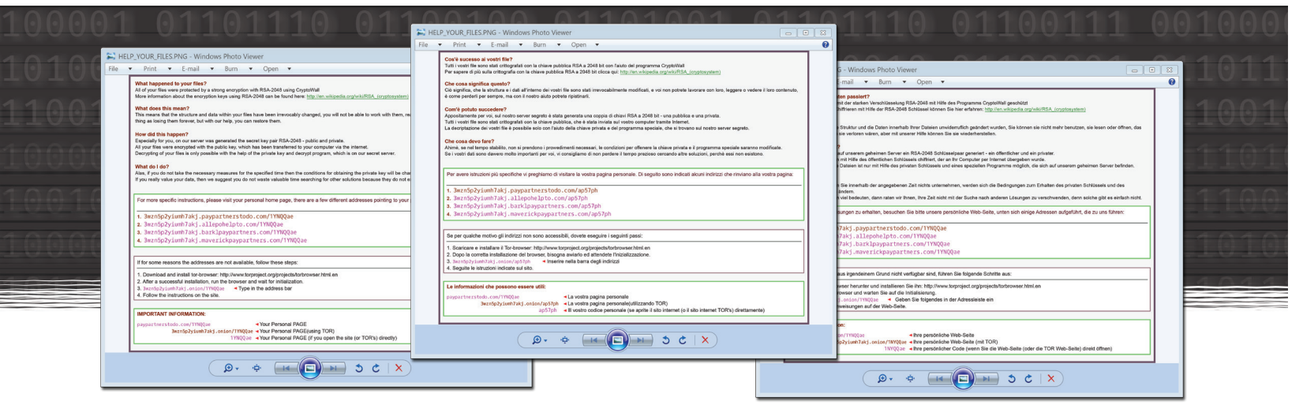
The network packet is sent to the web using a standard HTTP Post request, but is encrypted right before. The encryption algorithm is a custom one that uses a random string as key and produces a stream like the following one:

```
|<Letter>|=|<encryption Key in Hex>|<encrypted stream>|

e.g. s=6975376e7a9b0fd24886fbd0c0de32d3ab4d-
d97174462ca3b06af16a1c840ae893eddacafbd93e56847c23a41352d4f
45fc75468e4408
```

After the announcement process has successfully contacted the C&C it will try to obtain the public key. Here is a weak point of CryptoWall 4: if a strong firewall or IPS is able to intercept and block the CryptoWall 4 packets, the infection will not continue. The RSA-2048 public key is retrieved using a packet with ID set to 7. The C&C server answers with a packet composed as the following:

1. List of the payment URL (Tor URLs and standard URLs).
2. RSA-2048 Public key codified using base-64 encoding.
3. A base-64 encoded language-dependant PNG picture (English or Italian or German or others, depending on the local language settings).
4. (see Figure H., e.g. English or Italian or German or others, depending on the local language settings)



The public key is decoded using the CryptStringToBinary API. The decrypted buffer is stored in a global variable. The HTML and text files (compressed with LZ algorithm) are extracted from the dropper and finally the configuration file is generated, encrypted and stored in the following location:

```
C:\Users\<Username>\AppData\Roaming\<Random 8 digits>
```

The CryptoWall 4 configuration file contains the information needed for a successful execution of the malware. It also makes sure that the malware can continue to encrypt files in case the encryption process was interrupted e.g. because the PC was shut down. The file contains various continuous blocks, all preceded with a 4-bytes value that specify the size of the block.

CryptoWall 4 stores the following information inside the configuration file:

- The received public key binary data
- The HTML page showed to the user in the proper language
- The text file showed to the user in the proper language
- Finally the localized PNG picture in the proper language

The last 3 files will be written in each folder of the victim's workstation after the file-encryption process.

The configuration file is finally compressed with the LZ algorithm (using RtlCompressBuffer API, with index 2 - COMPRESSION_FORMAT_LZNT1) and written to disk.

If all goes well, the main thread will be spawned and the original one terminated (using RtlExitUserThread API).

FIGURE H.

Examples of CryptoWall decryption instructions provided in various targeted languages.

THE MAIN THREAD

The main thread starts by importing the public key. This means translating the encoded public key binary data in a proper data structure that can be used by the Windows Crypto APIs. CryptoWall 4 does this process using the CryptDecodeObjectEx API. In this way, the binary data could be transformed in a CERT_PUBLIC_KEY_INFO structure. Finally, the new data structure is imported in the Crypto APIs using CryptImportPublicKeyInfo function: a crypto-handle is returned. Then the MD5 hash of the public key is calculated. This is a very important step because it will be used to verify if a particular victim's file has been already encrypted.

At this point, the real encryption process take place. For each logical drive found in the victim system, a check is performed:

```
LPWSTR pngFilePath = new TCHAR[MAX_PATH];
// This produces something like "C:\HELP_YOUR_FILES.PNG"
ComposePngPath(driveName, "HELP_YOUR_FILES.PNG", pngFilePath, MAX_PATH);
if (!FileExists(pngFilePath) == TRUE) {
 // Proceed with the encryption
 // … … …
}
```

Practically speaking, if the root path of the drive contains a file named "HELP_YOUR_FILES.PNG", it will be skipped. We don't know if this is a bug or a voluntary behavior. For each drive that passes the test, a new encryption thread is spawned (the thread argument is a small structure that contains a copy of the public key and a pointer to a string with the drive name).

The main thread waits until all the encryption threads finish, then writes the three files that contains the decryption instructions in two locations: the startup folder of the Start Menu, and the desktop of the victim user.

Finally an end announcement network packet (type 7, sub-id 2, 1) is built and sent to the C2 server; the configuration file is deleted and the process is terminated using *ZwTerminateProcess* native API.

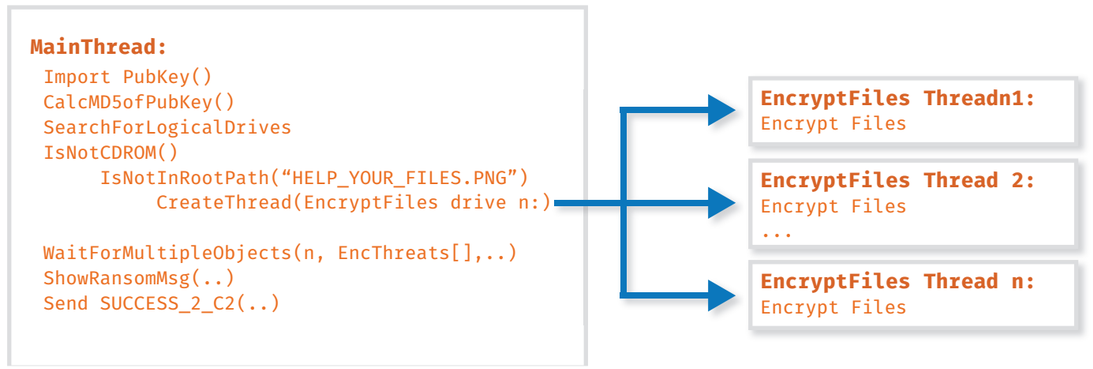


FIGURE I.

The MainThread uses multiple child threads to encrypt files.

THE ENCRYPTION THREAD

The encryption thread performs two main tasks: first it calls "DoFilesEncryption" routine to encrypt all the files that belongs to a volume and finally it writes "HELP_YOUR_FILES.PNG" wallpaper in the root path.

The "DoFilesEncryption" routine cycles between each folder and files of the target drive.

For each encountered sub-folder, it checks its name, filtering it by CRC32 (in this way some standard folders like "windows", "system32", "temp" will be ignored) and checks if the "HELP_YOUR_FILES.PNG" file exists, and if not, it calls itself again, specifying the new folder path as argument.

Each found file name is filtered two times: by extension and by name. If the filtering routines report that the name is good, the "EncryptFile" routine will be called.

"EncryptFile", as the name implies, is the routine that actually encrypts the target file: the original file is opened and its attributes queried. The "IsFileAlreadyEncrypted" function verifies the target file has been already encrypted by reading the first 16 bytes and comparing them with the MD5 hash of the public key.

A random filename and extension are generated by the malware at this point. The algorithm make use of the RtlRandomEx API to generate each readable character. The algorithm is on the following page.

The new file is created, a new AES-CBC 256 key is generated using CryptGenKey and CryptExportKey Windows Crypto APIs. This 32-byte key will be used to encrypt the entire file.

At this point CryptoWall 4 does a smart thing: it encrypts the generated AES key using the RSA-2048 public key (256 bytes in size) received from the C&C (using CryptEncrypt API). This will produce an encrypted 256-bit stream decryptable only by the bad guys.

The MD5 hash of the public RSA-2048 key is written in the first 16 bytes of the encrypted file; then CryptoWall 4 writes the 256-bit encrypted stream. The original file attributes and size are written to the next eight bytes. The original file name is encrypted using the generated AES key and written with its size inside the new file.

At this point the real file content encryption takes place, the original file is read in blocks that are 512 KByte chunks. Each block is encrypted with the generated key, using an AES-CBC 256 algorithm, and written directly to the new file (together with the block size in the first four bytes).

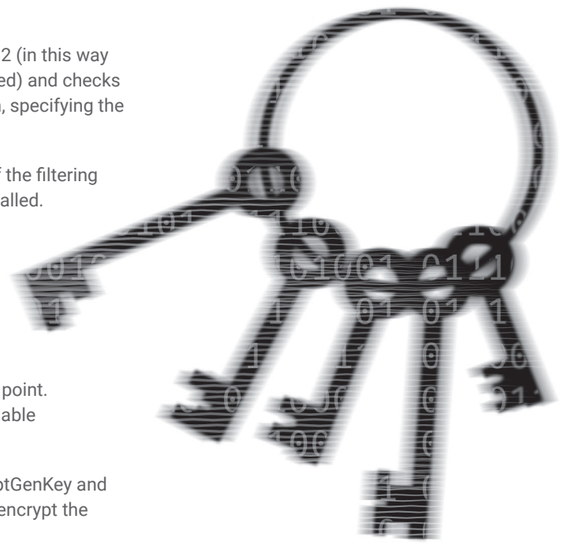
Once completed all the CryptoWall 4 resources are released and the original file is now deleted in an interesting manner:

```
// Move the new encrypted file name in the old original position, replacing the old one

bRetVal = MoveFileEx(newEncFileName, lpOrgFileName,
MOVEFILE_WRITE_THROUGH | MOVEFILE_REPLACE_EXISTING);
if (!bRetVal)
// Delete the old file in the standard manner:
    DeleteFile(lpOrgFileName);
else {
    // Rename the original replaced file in the new random file name
    bRetVal = MoveFileEx(lpOrgFileName, newEncFileName, MOVEFILE_REPLACE_EXISTING);
}
```

FIGURE J. (Right)

The algorithm used by the encryption thread.



```
// Generate a random value
DWORD GenerateRandValue(int min, int max) {
    if (min == max) return max;
    // Get the random value
    DWORD dwRandValue = RtlRandomEx(&g_qwStartTime.LowPart);
    DWORD dwDelta = max - min + 1;
    dwRandValue = (dwRandValue % dwDelta) + min;
    return dwRandValue;
}
// Generate a Random unicode string
LPWSTR GenerateRandomUString(int minSize, int maxSize) {
    DWORD dwStringSize = 0;             // Generated string size
    DWORD dwNumOfDigits = 0;            // Number of number letters inside the string
    LPWSTR lpRandString = NULL;         // Random unicode string
    // Generate the string size, and alloc buffer
    dwStringSize = GenerateRandValue(minSize, maxSize);
    lpRandString = new TCHAR[dwStringSize+1];
    for (int i = 0; i < (int)dwStringSize; i++) {
          DWORD dwLetter = 0;                       // Generated letter
          dwLetter = GenerateRandValue(0, 1000);
          dwLetter = (dwLetter % 26) + (DWORD)'a';
          lpRandString[i] = (TCHAR)dwLetter;
    }
    // NULL-terminate the string
    lpRandString[dwStringSize] = 0;
    // Now insert the digits inside the string
    DWORD dwUpperHalf = GenerateRandValue(dwStringSize / 2, dwStringSize);
    dwNumOfDigits = GenerateRandValue(1, dwUpperHalf);
    for (int i = 0; i < (int)dwNumOfDigits; i++) {
          DWORD dwValue = 0, dwPos = 0;       // Generated value and position
          dwValue = GenerateRandValue(0, 9) + (DWORD)'0';
          dwPos = GenerateRandValue(0, dwStringSize-1);
          lpRandString[dwPos] = (TCHAR)dwValue;
    }
    return lpRandString;
}
// Generate a random file name starting from a file full path
BOOLEAN GenerateRandomFileName(LPWSTR lpFileFullPath, LPWSTR * lppNewFileFullPath,
 LPWSTR * lppOrgFileName) {
    LPWSTR lpRandFileName = NULL;         // New random file name (without extension)
    LPWSTR lpRandExt = NULL;              // New random file extension
    LPWSTR lpNewFileName = NULL;          // The new file full name
    DWORD dwSize = 0;                     // size of the new filename
    // Check the arguments
    if (!lpFileFullPath || !lppNewFileFullPath || !lppOrgFileName)
          return FALSE;
    // Generate the new file name (without extension)
    lpRandFileName = GenerateRandomUString(5, 10);
    // Generate the random file extension
    lpRandExt = GenerateRandomUString(2,5);
    // Combine the new file name and extension and generate the final new file path
    // ....
    dwSize = wcslen(lpRandFileName) + wcslen(lpRandExt) + 1;
    lpNewFileName = new TCHAR[dwSize+1];
    swprintf_s(lpNewFileName, dwSize+1, L"%s.%s", lpRandFileName, lpRandExt);
    // ....
}
```

As you can see from the pseudo code on the previous page, the sectors used for the original clean file are specifically overwritten on the hard disk to ensure that any potential data recovery process is difficult. This is an interesting and novel method used by the malware author to ensure the highest chance of payment; reducing the ability to recover lost files is beneficial to their business model. Figure K below shows how the encrypted file is composed:

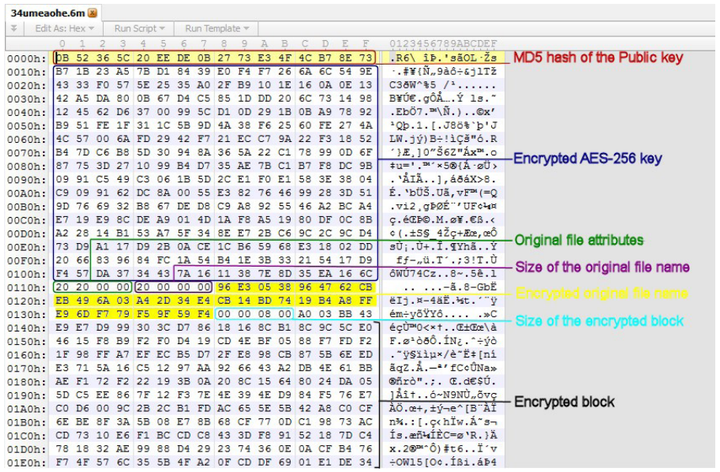


FIGURE K.

Breakdown of the encrypted file composition.

IOC DETAILS

Downloadable copy available at http://blogs.cisco.com/wp-content/uploads/cryptowall-4-iocs.txt

SAMPLE ANALYZED

3a73bb154506d8a9a3f4f658bac9a8b38d7590d296496e843503323d5f9b7801

SIMILAR SAMPLES FOUND

2d04d2a43e1d5a6920a806d8086da9c47f90e1cd25aa99b95af182ee9e1960b3
bf352825a70685039401abde5daf1712fd968d6eee233ea72393cbc6faffe5a2
299b298b433d1cc130f699e2b5c2d1cb3c7e5eb6dd8a5c494a8c5022eafa9223

THREATGRID REPORT (LICENSED USER ONLY)

https://panacea.threatgrid.com/samples/d25f94dc4f2ac59e0428f54d685ead38

C2 URL LIST

abelindia.com/1LaXd8.php
purposenowacademy.com/5_YQDI.php
mycampusjuice.com/z9r0qh.php
theGinGod.com/HS0ILJ.php
yahoosupportaustralia.com/8gX7hN.php
successafter60.com/iCqjno.php
alltimefacts.com/EiFSld.php
csscott.com/YuF59b.php
smfinternational.com/eRs70a.php
lexscheep.com/OIsSCj.php
successafter60.com/r_kfhH.php
posrednik-china.com/etdhIk.php
ks0407.com/VoZQ_j.php
stwholesaleinc.com/yL54uH.php
ainahanaudoula.com/GH09Dp.php
httthanglong.com/yzoLR7.php
myshop.lk/6872VF.php
parsimaj.com/60wEBT.php
kingalter.com/uVRfPv.php
shrisaisales.in/ZUQce4.php
cjforudesigns.com/E8B2gt.php
mabawamathare.org/WEAbCT.php
manisidhu.in/zJE0fD.php
adcconsulting.net/XEGeul.php
frc-pr.com/dA91lI.php
localburialinsuranceinfo.com/zDJRc8.phpsmfinternational.com/AYNILr.php

CONCLUSION

In this analysis we have put CryptoWall 4 under the microscope. The dropper doesn't bring anything terribly innovative, but it still has some clever features. The weak point in the infection process is that it deeply relies on the C2 server communication. If a firewall or IPS is able to detect and block its communication packets, the infection will not be able to operate correctly because it doesn't have any public key needed to properly encrypt the victim files. However, once CryptoWall 4 has encrypted the victim's files, there is no known way to recover the private key and decrypt them without paying the ransom as the RSA private key is never managed by the victim's workstation. The private key resides only on the adversaries' infrastructure.

As our analysis shows, the adversaries behind CryptoWall have continued to innovate and evolve to ensure it remains effective against users. Addressing the overall threat that ransomware presents requires organizations to be aware that adversaries will continue to evolve. Utilizing a multi-layered defensive approach will help organizations be able to detect and protect against threats like CryptoWall. Talos will continue to monitor CryptoWall as it evolves to identify better detection methods and build better defenses to protect our customers. We strongly encourage users and organizations to follow recommended security practices, such as installing security patches as they become available, exercising caution when receiving messages from unknown third-parties, and ensuring a robust backup solution is in place. These practices will help reduce the threat of a compromise and should aid in the recovery of any such attack.

APPENDIX A - EXCLUDED FILES *Note: This is not a comprehensive list

EXCLUDED FILES CRC32 CHECKSUMS	EXCLUDED EXTENSIONS CRC32 CHECKSUMS	EXCLUDED DIRECTORIES CRC32 CHECKSUMS
8E87F076h = help_your_files.txt	6B63B6F0h = exe	0E3E7859Bh = windows
0A73B295Ch = help_your_files.html	3DD3B336h = dll	0B5385CAh = temp
11A8ACA3h = help_your_files.png	0BB5EA5C1h = pif	0ED4E242h
88068F93h	592D276Fh = scr	9608161Ch
775DBED4h	9E07ED22h = sys	41476BE7h = cache
60479578h	8F3272A8h = msi	0F5832EB4h
7BD40679h = iconcache.db	0A45BDDC1h = no three letter ext	0D8601609h
48F43013h = thumbs.db	0B65F578Ah = no three letter ext	1DF021B7h
95ED794Ah	0EB59DA68h = msp	0B91A5F78h = sample pictures
884F3F52h	64B6C6E6h = com	0A622138Ah = default pictures
7DAC63A1h	0C863AEB6h = hta	3FF79651h = sample Music
4208466h	0DEEBF8EEh = cpl	62288CBBh = program files
0BA069E4Ch	6FE79BB6h = msc	224CD3A8h = program files (x86)
0EC619E8Dh	9F9C299Fh = bat	72D480B3h
9B0FD8B3h	2F5C1CC0h = cmd	0FF232B31h = games
	43F7F312h = scf	0A33D086Ah = sample videos
		78B7E09h = user account pictures
		9BB5C0A7h = packages
		24FA8EBDh

PROTECTING USERS

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors.

CWS or WSA web scanning prevents access to malicious websites and detects malware used in these attacks.

The Network Security protection of IPS and NGFW have up-to-date signatures to detect malicious network activity by threat actors.

ESA can block malicious emails sent by threat actors as part of their campaign.

PRODUCT	PROTECTION
AMP	✓
CWS	✓
ESA	✓
Network Security	✓
WSA	✓

TALOS

WWW.TALOSINTELLIGENCE.COM