Banking Trojan “Citadel” Returns

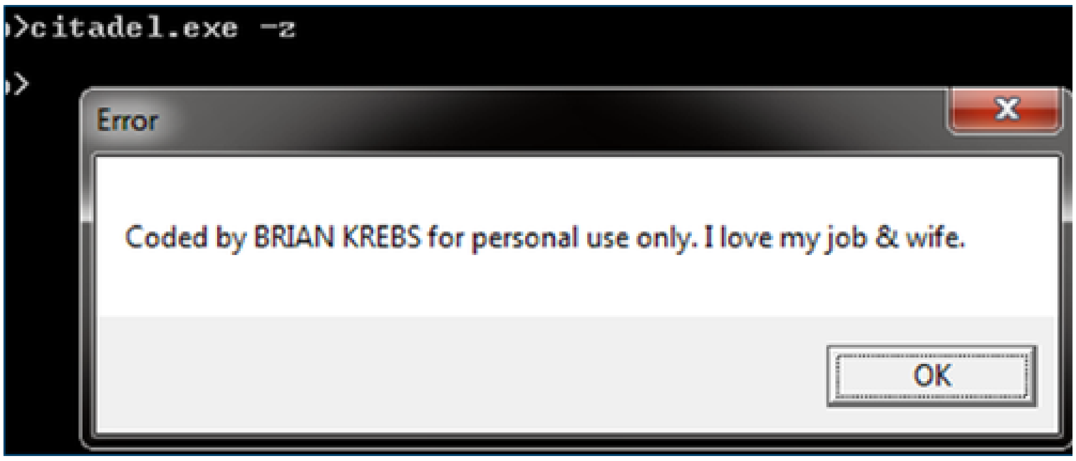
Hello again, this is You ‘Tsuru’ Nakatsuru from Analysis Center. It has been just about two years since I delivered a talk “Fight Against Citadel in Japan” at CODE BLUE 2013 (an international security conference in Tokyo) about the situation on banking trojans observed in Japan at that time and detailed analysis results on Citadel (See my blog entry here). For the presentation material and audio archive, please see Reference [1].

Since then, various kinds of efforts have been implemented against this malware. Citadel was less and less seen among incident reports to JPCERT/CC, and there had been no reports about it since the first half of 2014. However, in late November 2015, we observed an attempt to infect a host with an upgraded Citadel via Drive-by-Download attack.

In this blog article, I will discuss how this upgraded Citadel (here after tentatively named as “Citadel 101” since the version was set as 0.0.1.1) had changed from Citadel version 1.3.5.1 (hereafter “Citadel 1.3.5.1”) which I presented at CODE BLUE. I will also introduce a decryption tool for Citadel 101 created by JPCERT/CC.

Message Impersonating Brian Krebs Removed

It is widely known that Citadel 1.3.5.1 has a function which is not related to its intended behaviour: When executing with argument “-z”, the following message is displayed.



```
>citadel.exe -z

>
```

Error

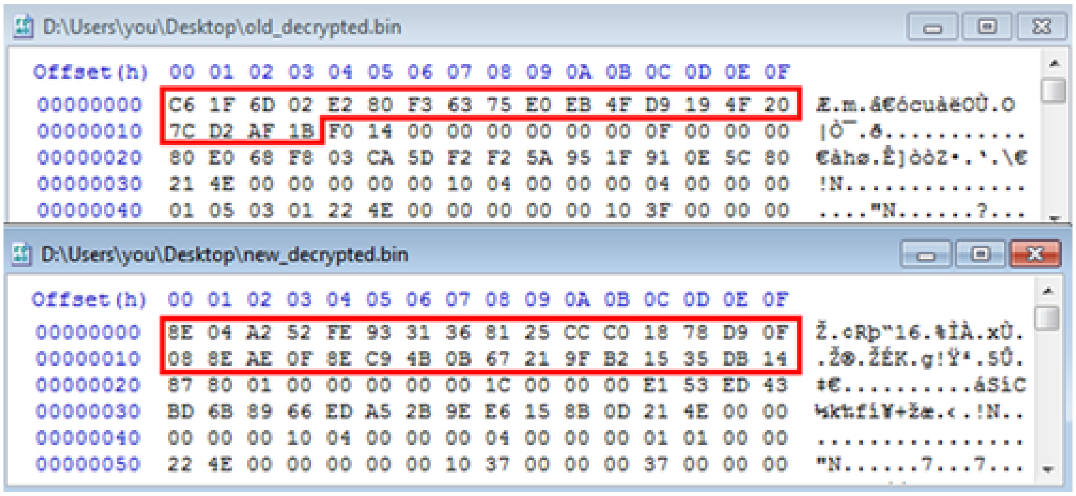
Coded by BRIAN KREBS for personal use only. I love my job & wife.

Figure 1: Dialog box displayed when executing with argument "-z"

This is a message impersonating Brian Krebs, a well-known security researcher of Krebs on Security. He himself actually has written an article about this (See Reference [2]). In Citadel 101, this function has been removed.

Change in Structure

There have been some slight changes in structures used within Citadel. For instance, in BinStrage (used as data format for configuration files), the size of random data at the beginning of the header had been changed from 20 bytes to 32 bytes as in Figure 2.



D:\Users\you\Desktop\old_decrypted.bin

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  C6 1F 6D 02 E2 80 F3 63 75 E0 EB 4F D9 19 4F 20  Æ.m.â€ócuàëOÙ.O 
00000010  7C D2 AF 1B F0 14 00 00 00 00 00 00 0F 00 00 00  |Ò¯.ð...........
00000020  80 E0 68 F8 03 CA 5D F2 F2 5A 95 1F 91 0E 5C 80  €àhø.Ê]òòZ•.‘.\€
00000030  21 4E 00 00 00 00 00 10 04 00 00 00 04 00 00 00  !N..............
00000040  01 05 03 01 22 4E 00 00 00 00 00 10 3F 00 00 00  ...."N......?...
```

D:\Users\you\Desktop\new_decrypted.bin

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  8E 04 A2 52 FE 93 31 36 81 25 CC C0 18 78 D9 0F  Ž.¢Rþ“16.%ÌÀ.xÙ.
00000010  08 8E AE 0F 8E C9 4B 0B 67 21 9F B2 15 35 DB 14  .Ž®.ŽÉK.g!Ÿ².5Û.
00000020  87 80 01 00 00 00 00 00 1C 00 00 00 E1 53 ED 43  ‡€..........áSíC
00000030  BD 6B 89 66 ED A5 2B 9E E6 15 8B 0D 21 4E 00 00  ½k‰fí¥+žæ.‹.!N..
00000040  00 00 00 10 04 00 00 00 04 00 00 00 01 01 00 00  ................
00000050  22 4E 00 00 00 00 00 10 37 00 00 00 37 00 00 00  "N......7...7...
```

Figure 2: Random data (in red boxes) at the beginning of BinStrage – in Citadel 1.3.5.1 (upper) and Citadel 101 (lower)

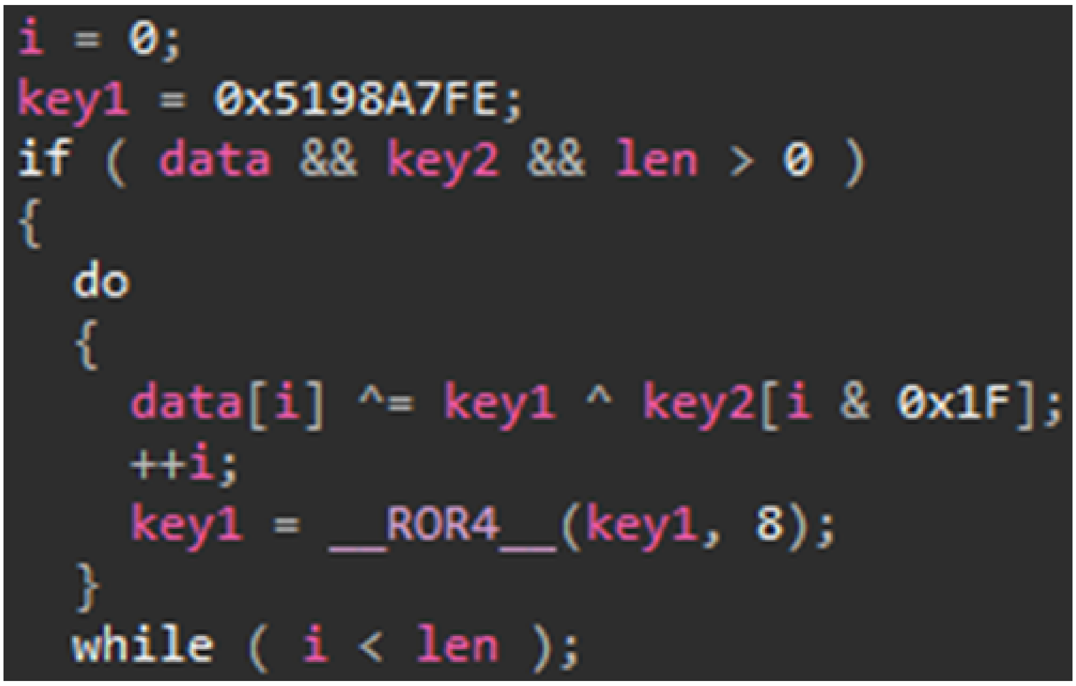
XOR Operation Added to Encryption Process

As I introduced at CODE BLUE, Citadel 1.3.5.1 encrypts files, registries and communication data, as indicated in Table 1.

Table 1: Objects that Citadel 1.3.5.1 encrypts and its methods

Encryption object		Data format	Encryption method
Communication data	Report	Encrypted BinStrage	RC4+
	Dynamic Config	Encrypted BinStrage	AES+
	Additional modules	Executable files	RC4+ * 2
Files	Report files	StrageArray	AES+ using Installed Data
	Module backup	StrageArray	AES+ using Installed Data
Registry	Dynamic Config backup	Encrypted BinStrage	AES+ using Installed Data

Citadel 101 encrypts the same objects and uses the same encryption methods as Citadel 1.3.5.1, however, XOR operation has been added at the beginning of the encryption process and the end of the decryption process. Data retrieved through the same decryption method as Citadel 1.3.5.1 includes XOR key (key2) and encoded data (data), which are used in the XOR decoding process (shown in Figure 3).



```
i = 0;
key1 = 0x5198A7FE;
if ( data && key2 && len > 0 )
{
  do
  {
    data[i] ^= key1 ^ key2[i & 0x1F];
    ++i;
    key1 = __ROR4__(key1, 8);
  }
  while ( i < len );
```

Figure 3: XOR decoding process added to Citadel 101

The default value of key1, which is one of the XOR keys used here, is hardcoded in Citadel itself. In the incident handling process, this value has to be newly retrieved in order to decrypt data encrypted by Citadel.

Updated Citadel Decryptor Published

Citadel Decryptor is a tool to decrypt data encrypted by Citadel, which I created and presented at CODE BLUE. This time, I expanded its feature so that it can decrypt data encrypted by Citadel 101 as well, and published it on GitHub.

JPCERTCC/aa-tools/citadel_decryptor
https://github.com/JPCERTCC/aa-tools/citadel_decryptor

Major changes are as follows:

- Added the process to obtain encryption keys, etc., hardcoded in Citadel 101 itself
- Added XOR decoding process at the end of decryption process

  ⇒In case it fails to obtain the XOR key, it judges that the version is Citadel 1.3.5.1 and does not perform XOR decoding

- Made changes to correspond to different structures between Citadel 101 and Citadel 1.3.5.1.

There is no change in its usage. Here below is an example of using the tool to decrypt the configuration file which Citadel 101 receives from a C&C server.

```
> citadel_decryptor.py -v -d root.xml citadel_main.bin
[*] start to decrypt root.xml
[*] get base config & several params
[*] found base config at RVA:0x000047f0, RA:0x000047f0
[*] found login key: D8F3A28A92E53179A3EC2100B314A5CB
[*] use RC4 key at (base config + 0x000001fd)
[*] found following xor key for AES plus:
```

```
[40, 40, 84, 92, 146, 121, 93, 197, 4, 73, 90, 178, 167, 220, 62, 44]
[*] found RC4 salt: 0x5198A7FE
[*] found xor key using after Visual Decrypt: 0x5198A7FE
[*] try to unpack
[*] decrypt data using following key:
[58, 225, (snip.) 50, 247, 122, 107, 114, 177, 190, 29, 60, 230, 186, 94]
[*] try to AES+ decryption
[*] use following AES key:
[181, 55, (snip.) , 252, 170, 168, 99, 231, 208, 131, 229, 244, 121]
[*] parse decrypted data... OK
[*] decompress decrypted data
[*] wrote decrypted data to root_decrypted.bin
```

Way Forward

Since JPCERT/CC has only seen a small number of Citadel 101 cases so far, there has not yet been enough testing on the Citadel Decryptor. There may be some other changes that are not fully covered in this update. I would like to make improvements based on your feedbacks, including Citadel samples which cannot be decrypted with the Citadel Decryptor. Your pull request is also highly appreciated!

Thank you for reading!

- You Nakatsuru

Reference

[1] ARCHIVE || Lecture of past || CODE BLUE 2013
http://codeblue.jp/2015/en/archive/2013/#speaker-you

[2] Krebs, KrebsOnSecurity, As Malware Memes — Krebs on Security
http://krebsonsecurity.com/2013/05/krebs-krebsonsecurity-as-malware-memes/

Appendix: SHA-256 hash value

dd16014eb3fa62d483758b63b2f412017381e6d9cc03347152dff3eb9f8e6e3b