Bug in Malware “TSCookie” - Fails to Read Configuration - - JPCERT/CC Eyes

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2018-11-11 · Archived: 2026-04-05 18:17:19 UTC

- BlackTech

In a previous article we have introduced malware ‘TSCookie’, which is assumedly used by an attacker group BlackTech. We have been observing continuous attack activities using the malware until now. In the investigation of an attack observed around August 2018, we have confirmed that there was an update in the malware. There are two points meriting attention in this update:

- Communication with C&C server
- Decoding configuration information

This article will introduce the details of the update.

Communication with C&C server

In the previous version, TSCookie included encrypted contents in the Cookie header to communicate to a C&C server.

```
GET /Default.aspx HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Date: Thu, 18 Jan 2018 10:20:55 GMT
Pragma: no-cache
Accept: */*
Cookie: 1405D7CD01C6978E54E86DA9525E1395C4DD2F276DD28EABCC3F6201ADAA66F55C15352D29D0FFE51BC9D431EB23
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Win32)
Host:[host name]:443
```

In the new version, Cookie header is no longer used. Instead, encrypted contents are placed within the URL parameter as below:

```
GET /t3328483620.aspx?m=4132641264&i=44D6CF457ADC27B2AFAAEAA&p=EF4D5069C30D6CAC9 HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Win32)
Host: [host name]:443
```

If received an ack from the server to this HTTP GET request, an HTTP POST request will be sent as a next step. The communication feature is the same as the previous TSCookie.
For encryption, RC4 is still used, but the key is generated differently. Here is an example code for decoding HTTP GET request parameter.

```
data = "&" + sys.argv[1] # sys.argv[1] = URL path
conf_key = sys.argv[2].decode("hex") # sys.argv[2] = Configuration key
field = data.split("&")

url_key = field[1]
i=2
encdata = ""
while i<len(field):
    value = field[i].split("=")
    encdata += value[1]
    i+=1

key1 = 0
for i in range(len(url_key)):
    key1 = ord(url_key[i]) + ROR(key1, 13)
    key1 = key1 & 0xFFFFFFFF

key2 = 0
for i in range(len(conf_key)):
    key2 = ord(conf_key[i]) + ROR(key2, 13)
    key2 = key2 & 0xFFFFFFFF

key = pack("I", key1) + pack("I", key2)

decode_data = rc4(encdata.decode('hex'), key)
```

Decoding configuration information

TSCookie possesses its own configuration information and operates accordingly. The details of the configuration remain the same in the new version. The difference is the decoding method of the configuration. Previously, TSCookie had its 4-byte RC4 key in the beginning of the configuration, which was used for decoding. In the new version, the size is expanded to 0x80 bytes (Figure 1).

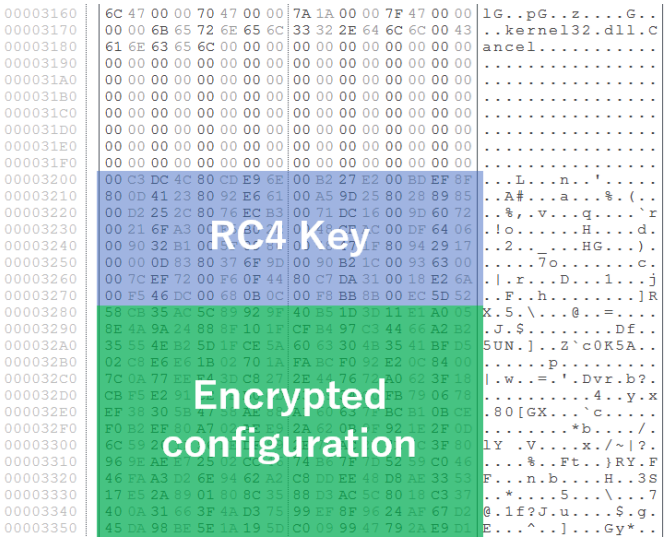
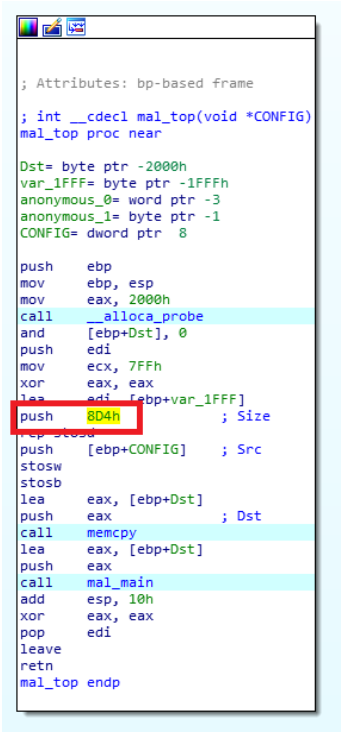


Figure 1: RC4 key and encrypted configuration

We have confirmed that this update made TSCookie fail to read part of the configuration. Figure 2 shows the code copying encrypted configuration (0x8D0 bytes) and RC4 key (0x80 bytes).



```
; Attributes: bp-based frame

; int __cdecl mal_top(void *CONFIG)
mal_top proc near

Dst= byte ptr -2000h
var_1FFF= byte ptr -1FFFh
anonymous_0= word ptr -3
anonymous_1= byte ptr -1
CONFIG= dword ptr  8

push    ebp
mov     ebp, esp
mov     eax, 2000h
call    __alloca_probe
and     [ebp+Dst], 0
push    edi
mov     ecx, 7FFh
xor     eax, eax
lea     edi, [ebp+var_1FFF]
push    8D4h            ; Size
rep stosd
push    [ebp+CONFIG]    ; Src
stosw
stosb
lea     eax, [ebp+Dst]
push    eax             ; Dst
call    memcpy
lea     eax, [ebp+Dst]
push    eax
call    mal_main
add     esp, 10h
xor     eax, eax
pop     edi
leave
retn
mal_top endp
```

Figure 2: Code copying RC4 Key and encrypted configuration

The code copies data sized 0x8D4 (0x8D0 + 4 bytes), which ignores the updated RC4 key size. To copy the updated RC4 key and configuration correctly, it needs to be set to 0x950 (0x8D0 + 0x80 bytes). With this fault, configuration cannot be decoded properly. Figure 3 describes how TSCookie configuration is decoded.

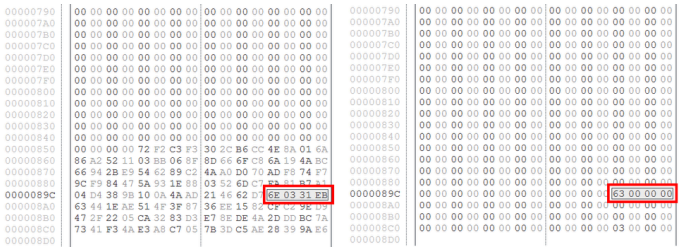


Figure 3: Decoded TSCookie configuration
(Left: Copy size 0x8D4, Right: Copy size 0x950)

Decoded results differ in the left figure (with the wrong, smaller copy size) and right figure (with correct, expanded copy size). Data at 0x89C byte (4 bytes) specifies the waiting time (seconds) before reconnecting to a C&C server. The attackers initially set this to 99 (0x63) seconds (as in the right figure), however, it will not be reconnected for few days since it is not read properly (left figure).

In closing

It is often the case that attackers give an update to their malware based on analysis reports provided from security vendors. We assume that this bug will be fixed sooner or later. We will update when we confirm new malware features.

The malware sample's hash value is available in Appendix A, and we also list some C&C servers in Appendix B. We hope this is helpful in identifying signs of infection.

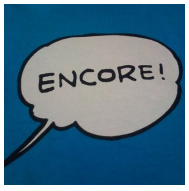
(Translated by Yukako Uchida)

Appendix A SHA-256 Hash Value of a sample

- a5c75f4d882336c670f48f15bf3b3cc3dfe73dba7df36510db0a7c1826d29161

Appendix B C&C server

- mediaplayer.dnset.com
- mediaplayers.ssl443.org
- fashion.androiddatacenter.com
- sakurings.flnet.org

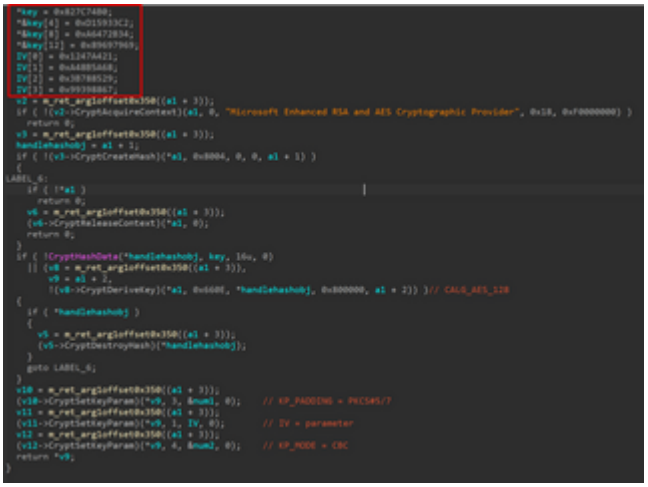


朝長 秀誠 (Shusei Tomonaga)

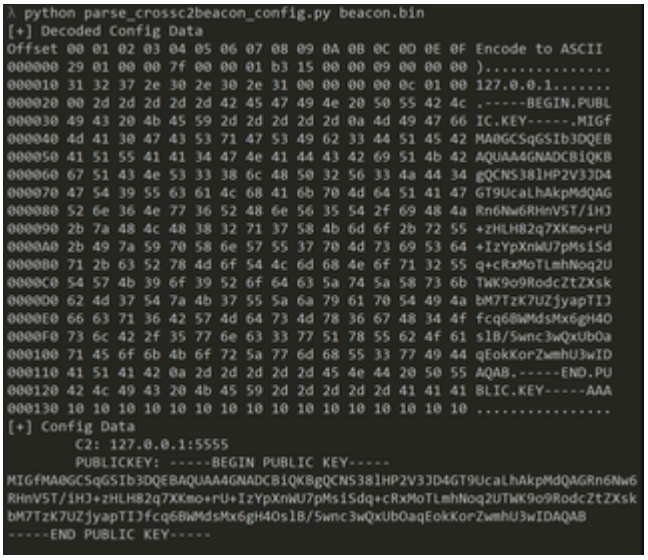
Since December 2012, he has been engaged in malware analysis and forensics investigation, and is especially involved in analyzing incidents of targeted attacks. Prior to joining JPCERT/CC, he was engaged in security

monitoring and analysis operations at a foreign-affiliated IT vendor. He presented at CODE BLUE, BsidesLV, BlackHat USA Arsenal, Botconf, PacSec and FIRST Conference. JSAC organizer.

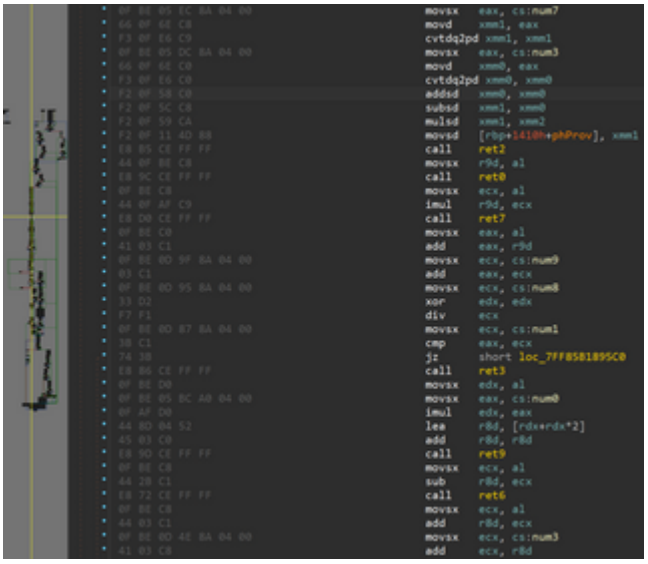
Related articles

- 

  Update on Attacks by Threat Group APT-C-60

- 

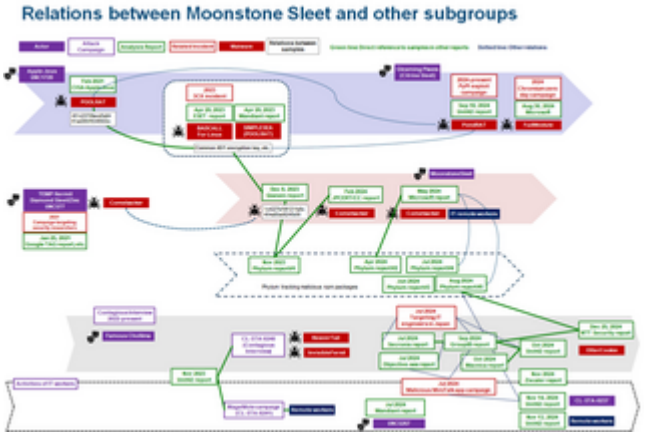
  CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks

- 

  Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities

- 

  DslogdRAT Malware Installed in Ivanti Connect Secure

- 

  Tempted to Classifying APT Actors: Practical Challenges of Attribution in the Case of Lazarus's Subgroup

---

Source: https://blogs.jpcert.or.jp/en/2018/11/tscookie2.html