

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

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2019-05-29 · Archived: 2026-04-06 00:31:49 UTC

May 30, 2019

- [BlackTech](#)

[Our past article](#) has presented a bug in malware “TSCookie”, which is reportedly used by BlackTech attack group. This article is to update the features of the malware.

Even after we published the blog article in October 2018, the adversary had continued using the malware as it was. Just in May 2019, we confirmed that the malware had its bug fixed and was used in some attack cases.

Details of the fix

The malware copies its configuration to the memory. In the previous version, the data size to be copied was incorrectly set, which resulted in the configuration not displayed properly (see the [article](#) for more details). In the updated version, the data size is set to 0x1000 instead of 0x8D4.

<pre>1 signed int Cancel() 2 { 3 mal_top(&CONFIG); 4 return 1; 5 }</pre>	<pre>1 signed int Cancel() 2 { 3 printf(&Format); 4 printf(&Format); 5 printf(&Format); 6 printf(&Format); 7 printf(&Format); 8 GetLastError(); 9 mal_top(&CONFIG, 0x1000u); 10 return 1; 11 }</pre>
<pre>1 int __cdecl mal_top(void *ENC_CONFIG) 2 { 3 char Dst; // [esp+4h] [ebp-2000h]@1 4 char v3; // [esp+5h] [ebp-1FFFh]@1 5 __int16 v4; // [esp+2001h] [ebp-3h]@1 6 char v5; // [esp+2003h] [ebp-1h]@1 7 8 Dst = 0; 9 memset(&v3, 0, 0x1FFCu); 10 v4 = 0; 11 v5 = 0; 12 memcpy(&Dst, ENC_CONFIG, 0x8D4u); 13 mal_main(&Dst); 14 return 0; 15 }</pre>	<pre>1 int __cdecl mal_top(void *ENC_CONFIG, size_t Size) 2 { 3 char Dst; // [esp+4h] [ebp-1400h]@1 4 char v4; // [esp+5h] [ebp-13FFh]@1 5 __int16 v5; // [esp+14001h] [ebp-3h]@1 6 char v6; // [esp+14003h] [ebp-1h]@1 7 8 Dst = 0; 9 memset(&v4, 0, 0x13FFCu); 10 v5 = 0; 11 v6 = 0; 12 memcpy(&Dst, ENC_CONFIG, Size); 13 mal_main((int)&Dst); 14 return 0; 15 }</pre>

Fig 1: Updates in TSCookie (Left: Code with the bug / Right: Updated code)

This update enables TSCookie to decode the configuration correctly. Fig 2 is the comparison of decoded configuration. This update has also fixed the issue where the malware fails to reconnect to a C&C server for a few days.

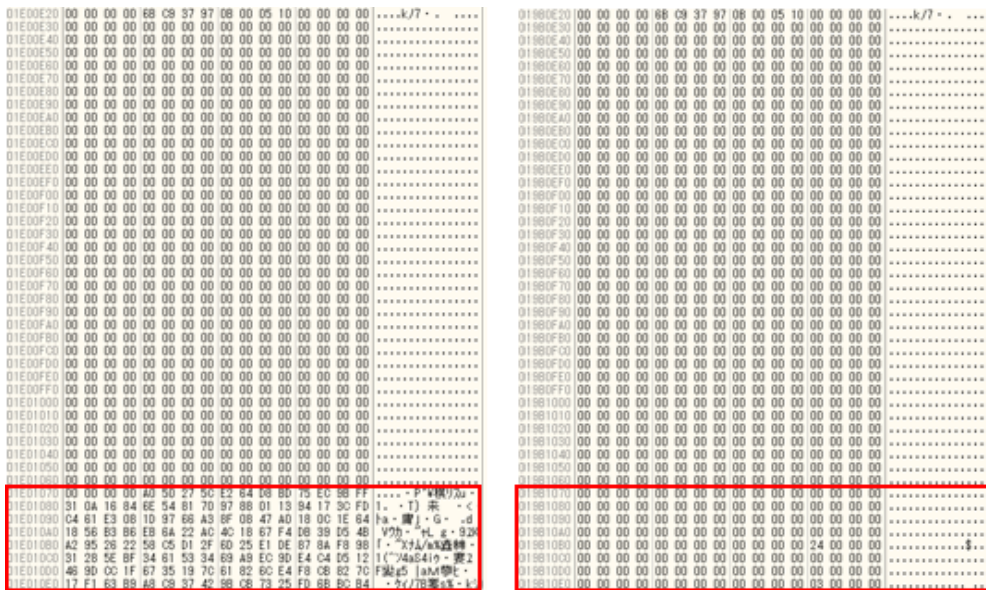


Fig 2: Decoded configurations of TSCookie (Left: Sample with the bug / Right: Updated sample)

In closing

As we pointed out before, it is likely that adversaries also follow publications and blogs from security vendors, etc. We assume that the adversary recognised the bug on our blog and fixed the issue accordingly. If we see any updates on the malware, we will introduce them here.

Hash values of the samples described in the article are listed in Appendix A, along with C&C servers in Appendix B. Please make sure that none of your devices is accessing these hosts.

Thank you for reading.

Shusei Tomonaga
(Translated by Yukako Uchida)

Appendix A: Hash value of the samples

Malware with the bug

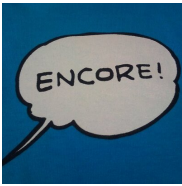
- 96723797870a5531abec4e99fa84548837e9022e9f22074cf99973ab7df2a2e7

Updated malware

- 1ec19677d1e48e4f6ff5f9fe7808b13964059e2ffd48ece19f7305d78e04ec4a
- c2c062ff84a18ad02e92dea0d6e12cafa66ff167ea8d02663fc9aae44de7f4e0

Appendix B: List of C&C servers

- www.google.com.dns-report.com
- microsoft.com.appstore.dynamicdns.co.uk
- cartview.viamisoftware.com



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.

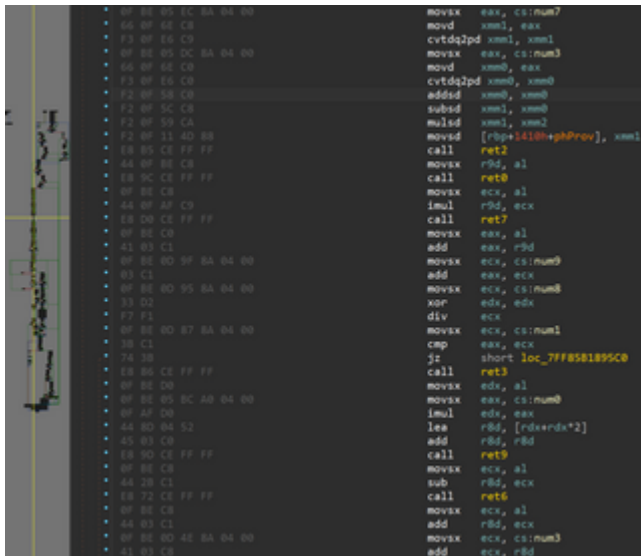
Related articles

```
*key = 0x427c7482;
*key[4] = 0x015833c2;
*key[8] = 0x6d72834;
*key[12] = 0x80077909;
Dv[4] = 0x147a411;
Zv[1] = 0x48803488;
Zv[2] = 0x30788529;
Zv[3] = 0x00000007;
v4 = m_ret_argOffset0x350(a1 + 1);
if ( !((C2->CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x1, 0xf000000) )
return 0;
v5 = m_ret_argOffset0x350(a1 + 3);
handlehash0 = a1 + 1;
if ( !((C2->CryptCreateHash)(*a1, 0x0004, 0, 0, a1 + 1) )
{
LABEL_0:
if ( *a1 )
return 0;
v6 = m_ret_argOffset0x350(a1 + 3);
(C2->CryptReleaseContext)(*a1, 0);
return 0;
}
if ( !CryptHashData(*handlehash0, key, 16u, 0)
|| (v6 = m_ret_argOffset0x350(a1 + 3);
v6 = a1 + 1;
!(v6->CryptDeriveKey)(*a1, 0x0004, *handlehash0, 0x000000, a1 + 2)) // CALG_AES_128
{
if ( *handlehash0 )
{
v5 = m_ret_argOffset0x350(a1 + 3);
(C2->CryptDestroyHash)(*handlehash0);
}
goto LABEL_0;
}
v8 = m_ret_argOffset0x350(a1 + 3);
(v8->CryptSetKeyParam)(*v8, 3, 0x0001, 0);
v9 = m_ret_argOffset0x350(a1 + 3);
(v9->CryptSetKeyParam)(*v9, 1, 0x, 0); // DV = parameter
v10 = m_ret_argOffset0x350(a1 + 3);
(v10->CryptSetKeyParam)(*v10, 0, 0x0002, 0); // 0x_0002 = CBC
return *v8;
}
```

Update on Attacks by Threat Group APT-C-60

```
python parse_cross2beacon_config.py beacon.bin
[+] Decoded Config Data
Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Encode to ASCII
000000 29 01 00 00 7f 00 00 01 b3 15 00 00 09 00 00 00 ).....
000010 31 32 37 2e 30 2e 30 2e 31 00 00 00 00 0c 01 00 127.0.0.1.....
000020 00 2d 2d 2d 2d 2d 42 45 47 49 4e 20 50 55 42 4c ,----BEGIN,PUBL
000030 49 43 20 4b 45 59 2d 2d 2d 2d 2d 2d 0a 4d 49 47 66 IC.KEY----.MIGF
000040 4d 41 30 47 43 53 71 47 53 49 62 33 44 51 45 42 MA8GCSqGS1b3QEB
000050 41 51 55 41 41 34 47 4e 41 44 43 42 69 51 4b 42 AQUAAAGNADCB1QKB
000060 67 51 43 4e 53 33 38 6c 48 50 32 56 33 4a 44 34 gQCNS3B1HP2V3J04
000070 47 54 39 55 63 61 4c 68 41 6b 70 4d 64 51 41 47 GT9UcaLhAkpM4QAG
000080 52 6e 36 4e 77 36 52 48 6e 56 35 54 2f 69 48 4a Rn6Nw6RHnVST/1HJ
000090 2b 7a 48 4c 48 38 32 71 37 58 4b 6d 6f 2b 72 55 +zHLH82q7XKmo+rU
0000A0 2b 49 7a 59 70 58 6e 57 55 37 70 4d 73 69 53 64 +IzYpXnwU7pMs1Sd
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4e 6f 71 32 55 q+cRxoTLmhNoq2U
0000C0 54 57 4b 39 6f 39 52 6f 64 63 5a 7a 5a 58 73 6b TwK9o9RodcZtZxsk
0000D0 62 4d 37 54 7a 4b 37 55 5a 6a 79 61 70 54 49 4a bM7Tzk7UZjyapTIJ
0000E0 66 63 71 36 42 57 4d 64 73 4d 78 36 67 48 34 4f fcq6BwMdsMx6gh40
0000F0 73 6c 42 2f 35 77 6e 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wXub0a
000100 71 45 6f 6b 4b 6f 72 5a 77 6d 68 55 33 77 49 44 eOkKorZwmHU3wID
000110 41 51 41 42 0a 2d 2d 2d 2d 2d 45 4e 44 20 50 55 AQAB.----END.PU
000120 42 4c 49 43 20 4b 45 59 2d 2d 2d 2d 2d 41 41 41 BLIC.KEY----AAA
000130 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....
[+] Config Data
C2: 127.0.0.1:5555
PUBLICKEY: ----BEGIN PUBLIC KEY----
MIGFMA8GCSqGS1b3QEBAQUAAAGNADCB1QKBgQCNS3B1HP2V3J04GT9UcaLhAkpM4QAGRn6Nw6
RHnVST/1HJ+zHLH82q7XKmo+rU+IzYpXnwU7pMs1Sdq+cRxoTLmhNoq2UTwK9o9RodcZtZxsk
bM7Tzk7UZjyapTIJfcq6BwMdsMx6gh40s1B/Swnc3wXub0aqEokKorZwmHU3wIDAQAB
----END PUBLIC KEY----
```

CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks

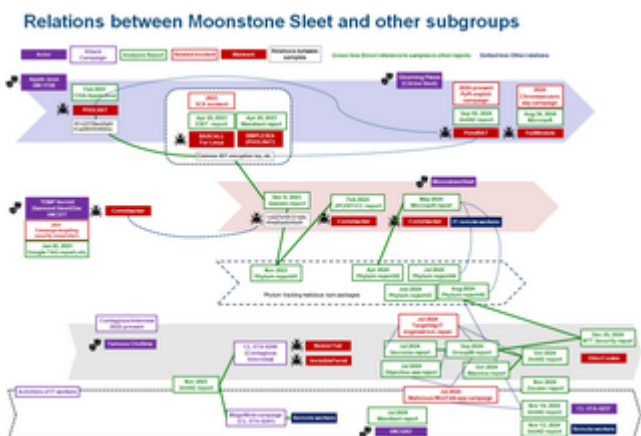


[Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities](#)

```
__int64 __fastcall mal_decode(__int64 encbuf, int bufsize)
{
    __int64 j_1; // rax
    int i; // [rsp+18h] [rbp-Ch]

    if ( encbuf )
    {
        for ( i = 0; ; ++i )
        {
            j_1 = (unsigned int)i;
            if ( i >= bufsize )
                break;
            *(_BYTE *)(encbuf + i) ^= Key1to7[i % 7];
        }
    }
    return j_1;
}
```

[DslodgRAT 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.jpccert.or.jp/en/2019/05/tscookie3.html>