

# Malware Used by BlackTech after Network Intrusion - JPCERT/CC Eyes

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2019-09-17 · Archived: 2026-04-05 17:14:01 UTC

- [Tool](#)
- [BlackTech](#)

Previously, we explained about malware "[TSCookie](#)" and "[PLEAD](#)" which are used by an attack group BlackTech. Their activities have been continuously observed in Japan as of now. We have been seeing that a new malware variant is being used after they successfully intruded into a target network. This article explains the details of the variant.

## TSCookie used after intrusion

The malware consists of 2 files (TSCookie Loader and TSCookie) as in Figure 1.

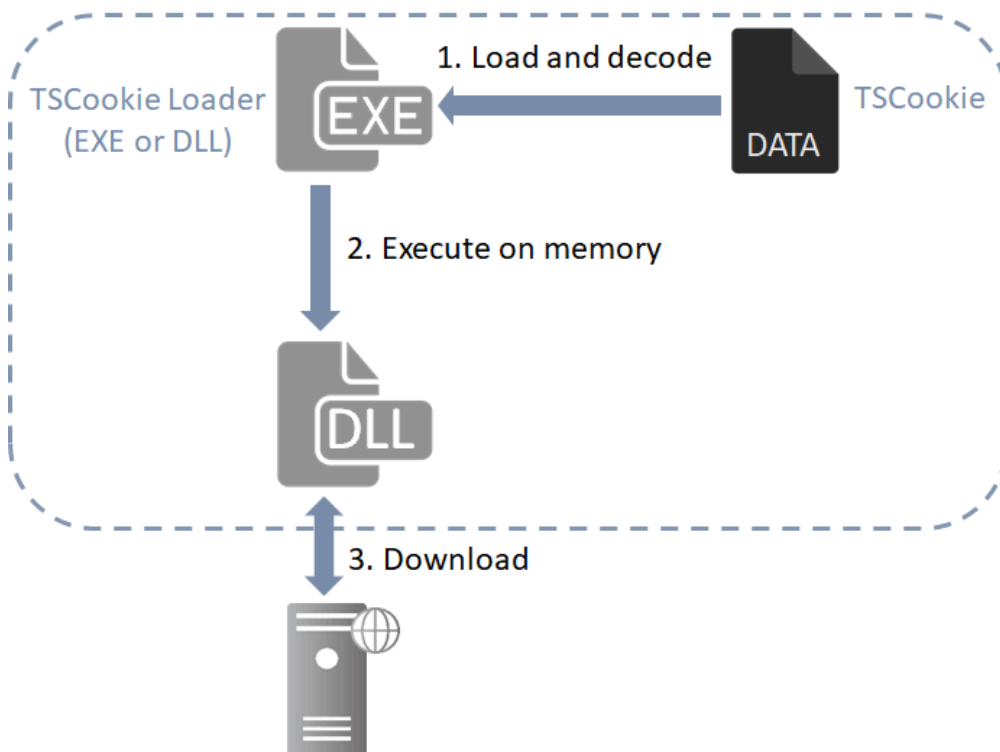


Figure 1: Overview of TSCookie Loader and TSCookie

TSCookie Loader is either in EXE or DLL format, and it reads and executes specific files stored in the same folder or the following locations. (The folders may vary depending on the sample.)

- C:\Windows
- C:\ProgramData\Microsoft

- C:\Users\Public\Documents
- C:\Program Files\Internet Explorer

It reads files that match the following file names:

- desktop.db
- Files with name that match 7???. (wildcard)
- Files with name that match 8???. (wildcard)

For example, files names such as KB78E7269.log and PM89E7267.xml have been confirmed.

TSCookie is RC4-encrypted and can be decoded by TSCookie Loader before being executed on the memory. TSCookie itself is a downloader and operates according to modules downloaded from an external server. Some characteristics such as configuration and communication protocols differ between TSCookie and the variant. Details of TSCookie behaviour is described in the following section.

### TSCookie behaviour

TSCookie supports multiple communication protocols (HTTP, HTTPS and custom protocol). The protocol that each sample uses is described in its configuration. (Please see Appendix A for the details of the configuration.) If it is configured to use HTTP protocol, the following HTTP POST request is sent:

```
POST /index?o=E7E168C4EC82E HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Win32)
Proxy-Connection: Keep-Alive
Content-Length: [size]
Host: [host name]

[Data]
```

There are several patterns of URL path, which are dynamically created with the following random strings and values. (Some of them are described with format specifiers. Other patterns also exist.)

- /news?%c=%X%X
- /index?%c=%X%X
- /?id=%X%X
- /Default.aspx?%c=%X%X
- /m%u.jsp?m=%d
- /N%u.jsp?m=%d

Sent data is RC4-encrypted. Please see Table B-1 and B-2 in Appendix B for the format of the data.

Data downloaded by the HTTP POST request is RC4-encrypted by the 8-byte value consisting of the RC4 key

(Table A-1 in Appendix A) and another value in the received data (RC4 key as in Table B-3 in Appendix B). The downloaded data contains modules, and they are executed on the memory.

### TSCookie decoding tool

We have developed a tool to decode TSCookie files and extract configuration. This is available on GitHub for your use.

JPCERTCC/aa-tools - GitHub

[https://github.com/JPCERTCC/aa-tools/blob/master/tscookie\\_data\\_decode.py](https://github.com/JPCERTCC/aa-tools/blob/master/tscookie_data_decode.py)

### In closing

We have received many reports about TSCookie infection. Please make sure that there is no infection in your organisation, referring to the file names and communication protocols described in this article. The hash value of the samples described in this article are listed in Appendix C.

Shusei Tomonaga

(Translated by Yukako Uchida)

### Appendix A: TSCookie Configuration

Table A: Configuration

| Offset | Contents                                 | Remarks   |
|--------|--|---|
| 0x000  | Destination server and port number       | Multiple hosts can be specified by listing with a semicolon ";"   |
| 0x400  | RC4 key                                  | Used for encryption   |
| 0x404  | Sleep times                              |   |
| 0x42C  | Mutex                                    |   |
| 0x44C  | Communication mode                       | - 1,2,3: HTTP protocol supporting authentication proxy<br>- 6,7,8: HTTPS protocol<br>- 0: Custom protocol<br>- 5: HTTP protocol |
| 0x454  | HTTP connection keep                     |   |
| 0x458  | ICMP recipient setting                   | Receive information of the destination server by ICMP   |
| 0x4D4  | IP address to receive ICMP communication |   |
| 0x624  | Process injection mode                   | - 0: Launch<br>- 1: Already running   |

|       |                              |   |
|-------|------------------------------|---|
|       |                              | - 2: Launch offset 0x62C  |
| 0x628 | Process to be injected       | - 0: svchost.exe<br>- 1: iexplorer.exe<br>- 2: explorer.exe<br>- 3: Default Browser<br>- 4: Process in offset 0x62C |
| 0x62C | Process name                 |   |
| 0x72C | Proxy server                 |   |
| 0x76C | Proxy port number            |   |
| 0x770 | Proxy username               |   |
| 0x790 | Proxy password               |   |
| 0x7B0 | Proxy mode                   | - 1: Use configuration data<br>- 0: Detect Proxy automatically  |
| 0x7B4 | Proxy authentication process | AuthScheme  |

- Some samples may not inject processes.

### Appendix B: Data exchanged by TSCookie

Table B-1: Format of sent data

| Offset | Length | Contents   |
|--------|--------|--|
| 0x00   | 4      | Number of received data (begins with 0xFFFFFFFF)   |
| 0x04   | 4      | Length of data sent  |
| 0x08   | 4      | Times of communication   |
| 0x0C   | 4      | Fixed value (Set to 0x5322 at the beginning, then to 0x5324 or 0x5325 while receiving modules) |
| 0x1C   | 4      | Random data (RC4 key)  |
| 0x20   | -      | Random data after first communication (See Table B-2 for first communication)                  |

- Up to offset 0x1C, the contents are RC4-encrypted with the key in the configuration and random data.

Table B-2: Data format of first communication after offset 0x20

| Offset | Length | Contents |
|--------|--------|----------|
|--------|--------|----------|

|      |   |                            |
|------|---|----------------------------|
| 0x00 | 4 | 0x9A65001E                 |
| 0x04 | 4 | Process ID                 |
| 0x08 | 4 | 0x5322                     |
| 0x0C | 4 | Random data                |
| 0x10 | 4 | Data size from offset 0x14 |
| 0x14 | - | Random data (RC4 key)      |

- Up to offset 0x14, the contents are RC4-encrypted with the key in the configuration and random data.

Table B-3: Format of received data

| Offset | Length | Contents   |
|--------|--------|--|
| 0x00   | 4      | Number of received data                            |
| 0x04   | 4      | Length of received data                            |
| 0x0C   | 4      | -  |
| 0x10   | 4      | Whether the contents from offset 0x20 is encrypted |
| 0x1C   | 4      | RC4 key  |
| 0x20   | -      | Module data  |

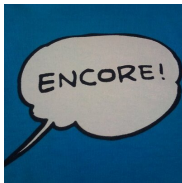
- Up to 0x1C, the contents are RC4-encrypted with the key contained in the configuration and another key in the received data.

## Appendix C: SHA-256 value of the samples

### TSCookie Loader

- 072f24d2691fb3930628be91bc46cefb8bc3364d1d09d72ab0cb3863681cb107
- f49956f498042feb237c3e898f74a8e14500c27cda2746efca2d973a5390baa8
- 3e12938df72380e4ae7a2dcb3322e563de3da102f5f32b26a29662ba594e73d1
- 23ca1a3ca26ada00502bbd1abf4d42302343dafba32cbc0711847d52884ff8e1
- 6ec56de53ef1ea66c81b3e48f9a9b3cf3dc8e3ebda1ec08bf95cc21228a4c7b3
- bd89b972de19c8ab2be0fb3e2aa44638a95e465e4b52920c94e6f59c25ce4693
- c5d7e5a12c8eab9c14f008c93d92e0070f84f358d39f28ac089ee917c652f5a8
- 85536a139b9d44157aea2908a6a6e53e4ac19077355680b69edd8e84c70254bc
- 0d00d12d71dd080d2861e9da89906a67bb822c64366b4c6b72a55bb8c26a4ea3
- 81dfce847a9fd6a3a0080a927bbb740709bdcc099bfe1b0cfc99958f6ddeb52f
- 48fdc29e7f47e5d38c88a89667ed85740628bf4f4ce95045019f7ebfeb4bbb5c

- d5909d06ddb394dea114052e9e174fa1e88324d805d153edb6076c53842fd2f2
- 9e10a1abbff4d421eae20040fb2a9270c4efb6d75ee6cd728b09bac1042bfa6
- ae5528cc802c81946f2787c7e884656416acebc89466989eeca9379fa066ad96
- 69b07aae04af6ca57d6066fdbcfbeb4c4849bfd2cd65b01c1e576f45b1c24d79
- 784b331d30d46ee9e7a264ecb45e3a39d7cef135d189bf0e712e89935728c13f
- 0eb9947a1ef4b810517f6cba175a321c4d69c3058d688bdd73492d54e7932c86



### [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 = 0x27C7A8B;
*key[4] = 0x215933C2;
*key[8] = 0x0472854;
*key[12] = 0x0407969;
*key[16] = 0x1247A22;
*key[20] = 0x4000000;
*key[24] = 0x30780529;
*key[28] = 0x0030800;

v0 = m_ret_argIffft0x350(a1 + 3);
if ( !v0->CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x1, 0x00000000 )
return 0;
v1 = m_ret_argIffft0x350(a1 + 3);
*handIshashobj = a1 + 3;
if ( !v1->CryptCreateHash)(a1, 0x0000, 0, 0, a1 + 3 )
{
LABEL_0:
if ( !*a1 )
return 0;
v0 = m_ret_argIffft0x350(a1 + 3);
(v0->CryptInitializeContext)(a1, 0);
return 0;
}
if ( !CryptHashData(*handIshashobj, key, 16u, 0) )
{
v0 = m_ret_argIffft0x350(a1 + 3);
v1 = a1 + 3;
v2 = CryptDeriveKey(a1, 0x0000, *handIshashobj, 0x000000, a1 + 3) // CALS_AES_128
{
if ( !*handIshashobj )
{
v0 = m_ret_argIffft0x350(a1 + 3);
(v0->CryptDestroyHash)(*handIshashobj);
}
goto LABEL_0;
}
v0 = m_ret_argIffft0x350(a1 + 3);
(v0->CryptSetKeyParam)(v0, 1, 0x0000, 0); // SP_P00000 = 0x00000000
v1 = m_ret_argIffft0x350(a1 + 3);
(v1->CryptSetKeyParam)(v1, 1, 0x0000, 0); // SP_P00000 = 0x00000000
v2 = m_ret_argIffft0x350(a1 + 3);
(v2->CryptSetKeyParam)(v2, 4, 0x0000, 0); // SP_P0000 = CBC
return v0;
}
```

### [Update on Attacks by Threat Group APT-C-60](#)

```

λ python parse_crossc2beacon_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 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 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 MA0GCSqGS1b3DQEB
000050 41 51 55 41 41 34 47 4e 41 44 43 42 69 51 4b 42 AQUAA4GNADCB1QKB
000060 67 51 43 4e 53 33 38 6c 48 50 32 56 33 4a 44 34 gQcNS381HP2V3JD4
000070 47 54 39 55 63 61 4c 68 41 6b 70 4d 64 51 41 47 GT9UcalhAkPMDQAG
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 +IzYpXmU7pMs1Sd
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4e 6f 71 32 55 q+cRxMoTLmhNoq2U
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 fcq6BwMdsMx6gH4O
0000F0 73 6c 42 2f 35 77 6e 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wXubOa
000100 71 45 6f 6b 4b 6f 72 5a 77 6d 68 55 33 77 49 44 qEokKorZumHU3wID
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-----
MIGFMA0GCSqGS1b3DQEBQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcalhAkPMDQAGRN6Nw6
RHnVST/1HJ+zHLH82q7XKmo+rU+IzYpXmU7pMs1Sdq+cRxMoTLmhNoq2UTWK9o9RodcZtZXsk
bM7TzK7UZjyapTIJfcq6BwMdsMx6gH4Os1B/Swnc3wXubOaqEokKorZumHU3wIDAQAAB
-----END PUBLIC KEY-----

```

[CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks](#)

```

* 73 0F 16 C9
* 86 0F 16 C8
* 73 0F 16 C8
* 72 0F 58 C8
* 72 0F 5C C8
* 72 0F 59 CA
* 72 0F 11 40 00
* 18 05 C1 FF FF
* 18 0C C1 FF FF
* 0F 0E C8
* 44 0F AF C9
* 18 00 C1 FF FF
* 0F 0E C8
* 41 03 C1
* 0F 05 00 0F 0A 04 00
* 03 C1
* 0F 0E 00 05 0A 04 00
* 33 02
* 77 F1
* 0F 0E 00 87 0A 04 00
* 10 C1
* 74 18
* 18 05 C1 FF FF
* 0F 0E D0
* 0F 0E 05 0C 0A 04 00
* 0F AF D0
* 44 00 04 52
* 45 03 C9
* 18 00 C1 FF FF
* 0F 0E C8
* 44 28 C1
* 18 72 C1 FF FF
* 0F 0E C8
* 44 03 C1
* 0F 0E 00 42 0A 04 00
* 41 03 C8
movsx eax, cs:num7
movd xmm1, eax
cvtdq2pd xmm1, xmm1
movsx eax, cs:num3
movd xmm0, eax
cvtdq2pd xmm0, xmm0
addsd xmm0, xmm0
subsd xmm1, xmm0
mulsd xmm1, xmm2
movsd [rbp+1410h+ph0prev], xmm1
call ret2
movsx r9d, al
call ret0
movsx ecx, al
imul r9d, ecx
call ret7
movsx eax, al
add eax, r9d
movsx ecx, cs:num9
add ecx, ecx
movsx ecx, cs:num8
xor edx, edx
div ecx
movsx ecx, cs:num1
cmp eax, ecx
jz short loc_7FF85B1895C8
call ret1
movsx edx, al
movsx eax, cs:num0
imul edx, eax
lee r8d, [rdx+rdx*2]
add r8d, r8d
call ret9
movsx ecx, al
sub r8d, ecx
call ret6
movsx ecx, al
add r8d, ecx
movsx ecx, cs:num3
add ecx, r8d

```

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

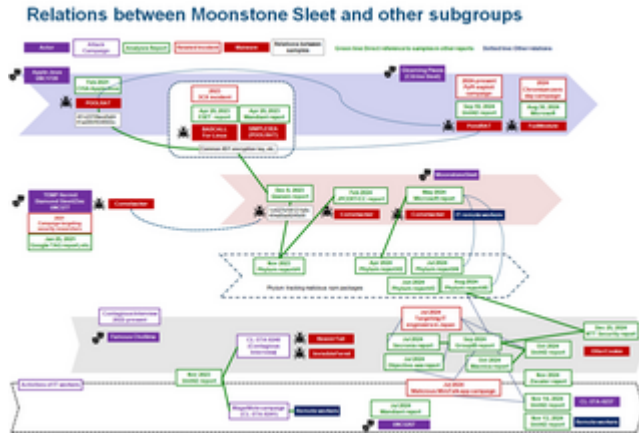
```

__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;
}

```

[DslugdRAT 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/2019/09/tscookie-loader.html>