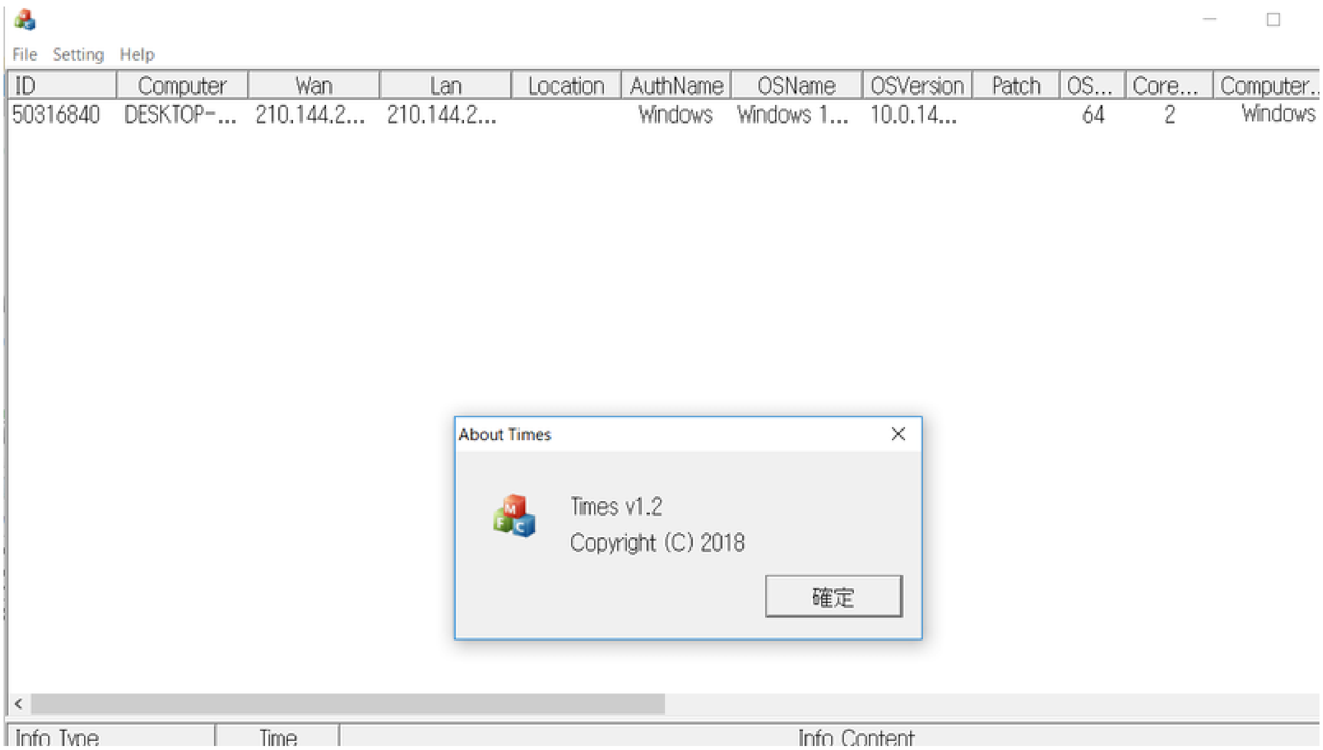
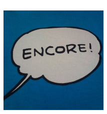
JPCERT Coordination Center official Blog

blogs.jpcert.or.jp/en/2021/10/gh0sttimes.html





朝長 秀誠 (Shusei Tomonaga)

October 4, 2021

Malware Gh0stTimes Used by BlackTech

BlackTech

-
- Email

An attack group BlackTech has been actively conducting attacks against Japanese organisations since 2018. Although it is not as prominent as before, JPCERT/CC is still seeing some cases as of now. This article introduces the details of the malware Gh0stTimes, which is used by this group.

Gh0stTimes overview

Gh0stTimes is customised based on Gh0st RAT and has been used in some attack cases since 2020. Figure 1 shows the comparison of Gh0stTimes and Gh0st RAT code.

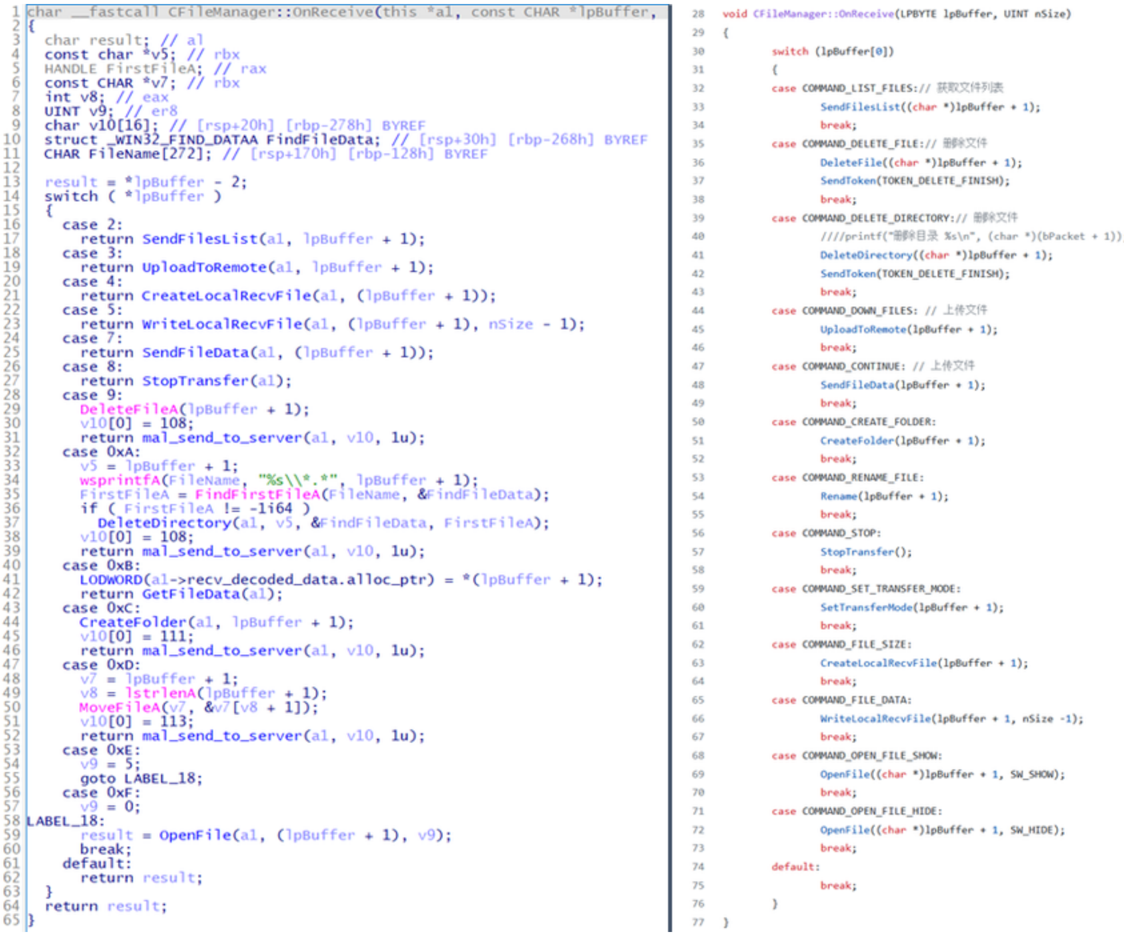


Figure 1: Comparison of Gh0stTimes and Gh0st RAT (CFileManager) code
(Left: Gh0stTimes / Right: Gh0st RAT)

Both sets of code are functions for file operation, and they are almost identical. Many of the Gh0st RAT functions are upgraded in Gh0stTimes, but some parts of the code are just kept as is. The next sections explain the features of Gh0stTimes.

- Communication protocol
- Commands
- Dummy code
- C2 server control panel

Communication protocol

Just like Gh0st RAT, Gh0stTimes communicates with C2 servers with its custom protocol, but the packet format is different. Figure 2 shows the flow of communication.

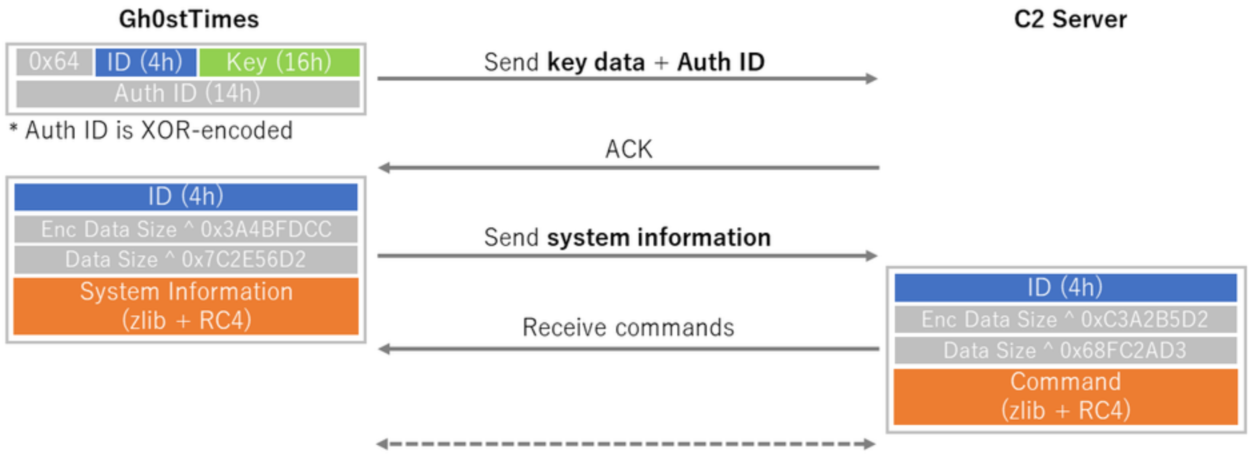


Figure 2: Gh0stTimes communication flow

At the beginning of its communication with a C2 server, Gh0stTimes sends an authentication ID and data (The "Key" in Figure 2) to generate an encryption key for the following communication. The C2 server checks the authentication ID and only accepts the communication with certain IDs. Figure 3 shows an example of the specific authentication IDs.

```
v5 = time64(0i64);
srand(v5);
for ( i = 0i64; i < 4; *(&a1->event + i + 7) = (rand() % 256) ^ 0x99 )
  ++i;
do
  *(&a1->first_senddata.id + ++v1 + 3) = (rand() % 256) ^ 0xCC;
while ( v1 < 16 );
not_use = 0x64793A7B622250DBi64;
auth2 = 0x309FEA572227F433i64;
p_Auth1 = &a1->first_senddata.Auth1;
len = 4i64;
a1->first_senddata.Auth1 = 0x64793A7B622250DBi64;
a1->first_senddata.Auth2 = auth2;
do
{
  v9 = *(p_Auth1 - 16);
  v10 = *(p_Auth1 - 14);
  p_Auth1 = (p_Auth1 + 4);
  v11 = *(p_Auth1 - 2) ^ v10 ^ 0xDD;
  *(p_Auth1 - 4) ^= v9 ^ 0xDD;
  v12 = *(p_Auth1 - 19);
  *(p_Auth1 - 2) = v11;
  v13 = *(p_Auth1 - 1) ^ *(p_Auth1 - 17) ^ 0xDD;
  --len;
  *(p_Auth1 - 3) ^= v12 ^ 0xDD;
  *(p_Auth1 - 1) = v13;
}
while ( len );
```

Figure 3: Gh0stTimes authentication ID sample

After the successful authentication, the communication that follows is encrypted with the key provided at the beginning of the communication. The next round of communication includes the information of infected hosts, such as hostname, username and processor name (Figure 4).

```
00000000  66 57 49 4E 44 4F 57 53 31 30 2D 68 6F 73 74 00  fWINDOWS10-host.
00000010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000020  D2 90 20 64 75 73 65 72 6E 61 6D 65 00 00 00 00  .. dusername....
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000040  00 00 57 69 6E 64 6F 77 73 20 31 30 20 45 6E 74  ..Windows 10 Ent
00000050  65 72 70 72 69 73 65 00 45 00 6E 00 74 00 65 00  erprise.E.n.t.e.
00000060  72 00 70 00 72 00 69 00 73 00 65 00 00 00 00 00  r.p.r.i.s.e.....
00000070  00 00 00 00 0A 00 00 00 00 00 00 00 63 45 00 00  ............cE..
00000080  01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00 40 01 75 73 65 72 6E 61  ........@.userna
000000A0  6D 65 00 00 00 00 00 00 00 00 00 00 00 00 00 00  me..............
000000B0  00 00 00 00 00 00 00 00 D0 10 00 00 42 55 49 4C  ............BUIL
000000C0  54 49 4E 5C 41 64 6D 69 6E 69 73 74 72 61 74 6F  TIN\Administrato
000000D0  72 73 20 65 6E 61 62 6C 65 64 00 00 00 00 00 00  rs enabled......
000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000F0  02 00 00 00 01 00 00 00 00 01 00 00 49 6E 74 65  ............Inte
00000100  6C 28 52 29 20 58 65 6F 6E 28 52 29 20 43 50 55  l(R) Xeon(R) CPU
00000110  20 45 35 2D 32 36 35 30 20 30 20 40 20 32 2E 30   E5-2650 0 @ 2.0
00000120  30 47 48 7A 00 00 00 00 00 00 00 00 00 00 00 00  0GHz............
00000130  00 D0 F7 7F 00 00 00 00 1E 08 00 00 00 00 00 00  ................
00000140  73 76 63 68 6F 73 74 36 34 2D 33 2E 65 78 65 00  svchost64-3.exe.
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000160  00 00 00 00 C0 EF BB EF 0F 3A D6 01 08 C2 B8 83  .........:......
00000178  0F 3A D6 01 00 00 00 00                          .:......
```

Figure 4: Information of infected hosts sent by Gh0stTimes

After sending the information of infected hosts, commands are exchanged. See Appendix A for the format of data exchanged. When exchanging commands, the data is RC4-encrypted and then zlib-compressed. Gh0stTimes uses its custom RC4 algorithm, which has XOR 0xAC process over the encrypted data.

```
if ( len > 0 )
{
  LODWORD(x) = 0;
  i = dst;
  data_len = len;
  y = 0;
  data = src - dst;
  do
  {
    ++i;
    x = (((x + 1) >> 31) + x + 1) - ((x + 1) >> 31);// x = x + 1;
    v9 = y + a1->box[x];
    v10 = a1->box[x];
    y = (BYTE4(v9) + y + v10) - BYTE4(v9);     // y = y + box[x];
    a1->box[x] = a1->box[y];                   // box[x] = box[y];
    a1->box[y] = v10;                          // box[y] = box[x];
    v11 = v10 + a1->box[x];
    result = *(data + i - 1) ^ a1->box[(BYTE4(v11) + v11) - BYTE4(v11)] ^ 0xAC;// result = data[i - 1] ^ box[(box[x] + box[y])] ^ 0xAC
    --data_len;
    *(i - 1) = result;
  }
  while ( data_len );
}
```

Figure 5: Part of Gh0stTimes code to encrypt data with RC4

The following is Python code to decode data exchanged.

```
import zlib

# Load keydata for first packet
with open(args[1], "rb") as fb:
    keydata = fb.read()

# Load encoded packet data
with open(args[2], "rb") as fb:
    data = fb.read()

comp_data = custom_rc4(data[12:], keydata[5:21])
dec_data = zlib.decompress(comp_data)

def custom_rc4(data, keydata):
    key = []
    key_1 = [0x98, 0x19, 0x3C, 0x56, 0xD9, 0xBB, 0xC7, 0x86, 0xFF, 0x3E]
    key_2 = [0] * 16
    key_3 = [0xAC, 0xBB, 0x30, 0x5E, 0xCC, 0xDD, 0x19, 0x23, 0xFC, 0xBD]
    keybox = [7, 0, 2, 3, 9, 10, 4, 13, 14, 8, 1, 11, 5, 6, 12, 15]

    i = 0
    for i in range(16):
        key_2[i] = keydata[keybox[i]]

    key = key_1 + key_2 + key_3
    x = 0
    box = list(range(256))
    for i in range(256):
        x = (x + box[i] + key[i % len(key)]) % 256
        box[i], box[x] = box[x], box[i]

    x = 0
    y = 0
    out = []
    for char in data:
        x = (x + 1) % 256
        y = (y + box[x]) % 256
        box[x], box[y] = box[y], box[x]
        out.append((char ^ box[(box[x] + box[y]) % 256] ^ 0xAC).to_bytes(1,
byteorder='little'))

    return b''.join(out)
```

Commands

Gh0stTimes is equipped with the following 5 types of commands:

- FileManager (command number 0x1): File operation
- ShellManager (command number 0x28): Remote shell execution
- PortmapManager (command number 0x32): C2 server redirect function
- UltraPortmapManager (command number 0x3F): Proxy function
- No name (command number 0): End communication

```
__int64 __fastcall CKernelManager::OnReceive(CKernelManager *a1, unsigned __int8 *a2)
{
  __int64 result; // rax

  result = *a2;
  switch ( *a2 )
  {
    case 0u:
      _InterlockedExchange(&a1->IsActived, 1);
      return result;
    case 1u:
      result = MyCreateThread(0i64, 0i64, Loop_FileManager, a1->lp_this->c2_socket, 0, 0i64, 0);
      goto LABEL_4;
    case 0x28u:
      result = MyCreateThread(0i64, 0i64, Loop_ShellManager, a1->lp_this->c2_socket, 0, 0i64, 1);
      goto LABEL_4;
    case 0x2Au:
      return CreateEventA(0i64, 1, 0, &a1->EventName);
    case 0x32u:
      result = MyCreateThread(0i64, 0i64, Loop_PortmapManager, a1->lp_this->c2_socket, 0, 0i64, 1);
      goto LABEL_4;
    case 0x3Fu:
      result = MyCreateThread(0i64, 0i64, Loop_UltraPortmapManager, a1->lp_this->c2_socket, 0, 0i64, 1);
LABEL_4:
      a1->thread_list[a1->num_threads++] = result;
      break;
    default:
      return result;
  }
  return result;
}
```

Figure 6: List of commands

ShellManager and FileManager are the same as Gh0st RAT's original functions. FileManager has multiple functions to operate files on infected hosts. (See Appendix B for details.) PortmapManager and UltraPortmapManager are unique to Gh0stTimes, which indicates that its relay function has been enhanced compared to Gh0st RAT.

Dummy code

Some types of malware that BlackTech use contains dummy code, which may make analysis difficult. Gh0stTimes has such code (Figure 7), but it does not have much impact to the analysis.

```
v60 = 0i64;
v61 = 0i64;
v62 = 0;
GetLocalTime(&v36);
LODWORD(v33) = v36.wSecond;
LODWORD(v30) = v36.wMinute;
LODWORD(v27) = v36.wHour;
LODWORD(v23) = v36.wDay;
sprintf(&v55, "%d-%d-%d %d:%d:%d", v36.wYear, v36.wMonth, v23, v27, v30, v33);
do
{
  v20 = OpenEventA(0x1F0003u, 0, &Name);
  v21 = WaitForSingleObject(this.event, 0x64u);
  Sleep(0x1F4u);
}
while ( !v20 && v21 );
GetLocalTime(&v36);
LODWORD(v34) = v36.wSecond;
LODWORD(v31) = v36.wMinute;
LODWORD(v28) = v36.wHour;
LODWORD(v25) = v36.wDay;
sprintf(&v55, "%d-%d-%d %d:%d:%d", v36.wYear, v36.wMonth, v25, v28, v31, v34);
if ( !v20 )
{
  GetLocalTime(&v36);
  LODWORD(v35) = v36.wSecond;
  LODWORD(v32) = v36.wMinute;
  LODWORD(v29) = v36.wHour;
  LODWORD(v26) = v36.wDay;
  sprintf(&v55, "%d-%d-%d %d:%d:%d", v36.wYear, v36.wMonth, v26, v29, v32, v35);
  Sleep(0x927C0u);
  CKernelManager_terminatethread(&CKernelManager_);
  continue;
}
break;
}
struc_this_closesocket(&this);
CloseHandle(v20);
```

Figure 7: Gh0stTimes dummy code sample

C2 server control panel

In the course of analysis, we found Gh0stTimes control panel. Figure 8 shows its GUI when the control panel is running. This one was named as "Times v1.2".

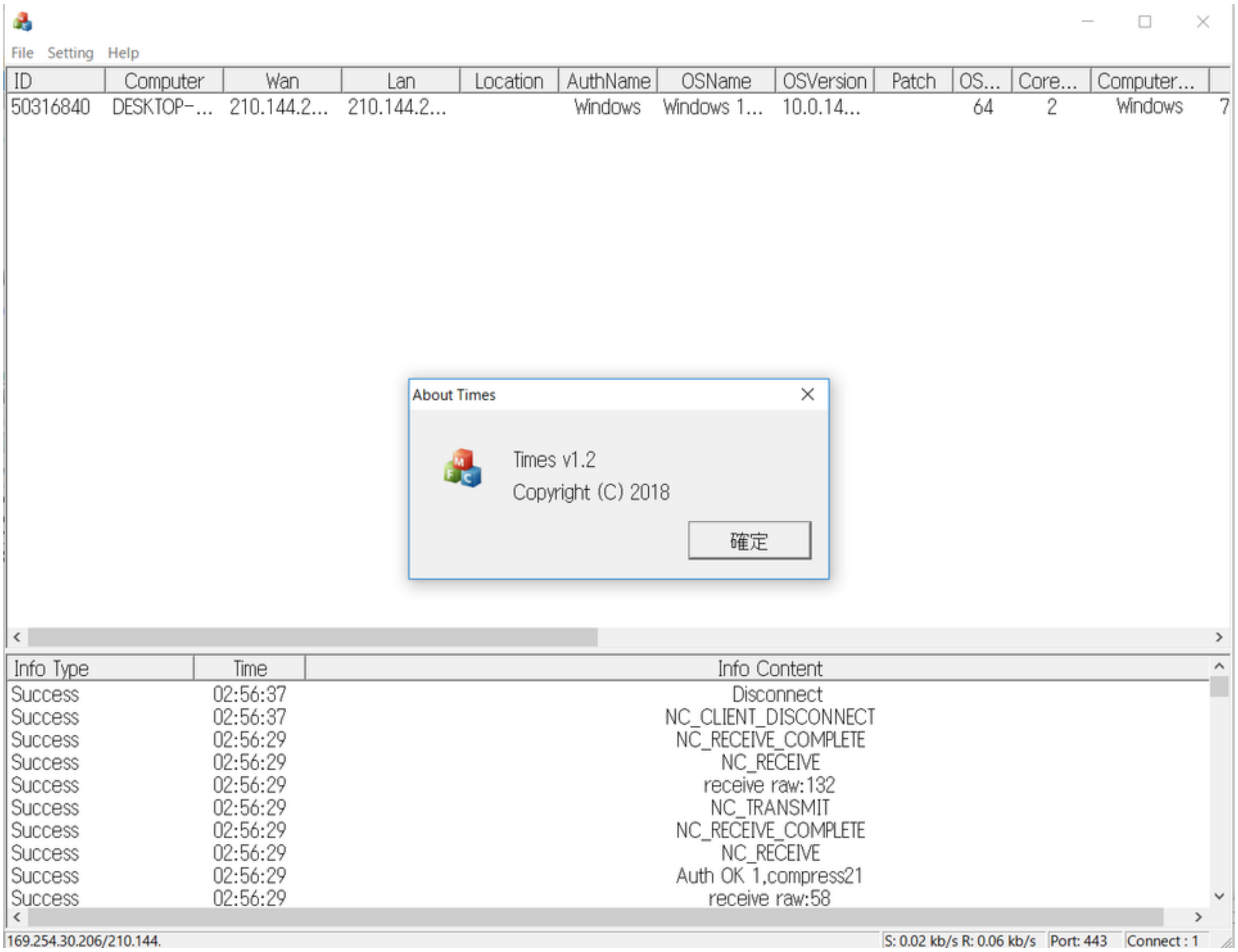


Figure 8: Gh0stTimes control panel

Figure 9 shows the commands that can be executed on the control panel.

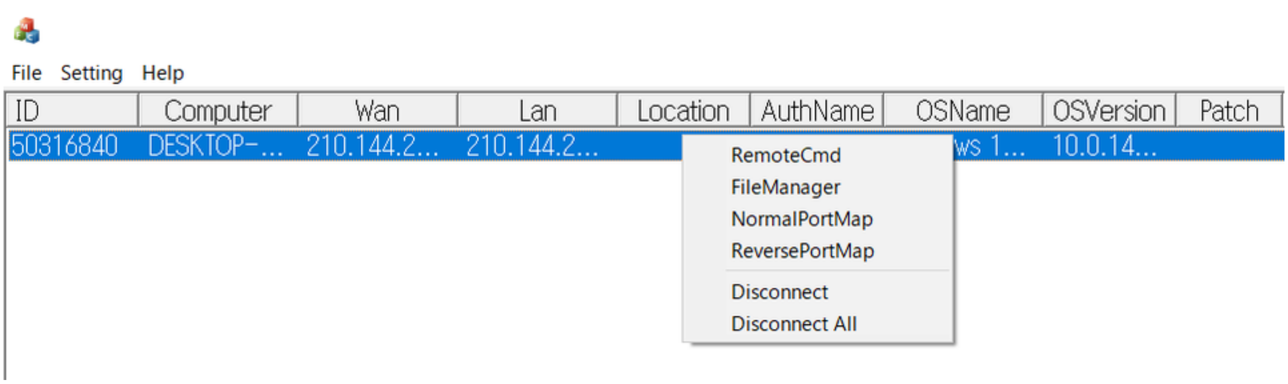


Figure 9: List of commands on Gh0stTimes control panel

In closing

As BlackTech has been actively carrying out attacks, we will continue our analysis and monitoring. A list of IoC is available in Appendix C. Please make sure that none of your devices is communicating with them.

We have identified that servers infected with Gh0stTimes are also affected by other types of malware (downloader, backdoor, ELF Bifrose) and attack tools listed below. Please be aware that these tools are possibly used by BlackTech.

- https://github.com/Yang0615777/PocList
- https://github.com/liuxu54898/CVE-2021-3019
- https://github.com/knownsec/pocsuite3
- Citrix exploit tool
- MikroTik exploit tool
- Exploit for CVE-2021-28482
- Exploit for CVE-2021-1472/CVE-2021-1473
- Exploit for CVE-2021-28149/CVE-2021-28152
- Exploit for CVE-2021-21975/CVE-2021-21983
- Exploit for CVE-2018-2628
- Exploit for CVE-2021-2135

Acknowledgement

We would like to acknowledge the support and information shared by @r3dbU7z regarding this attack group.

Shusei Tomonaga
(Translated by Yukako Uchida)

Appendix A: Data exchanged

Table A-1: Format of data sent

Offset	Length	Contents
0x00	4	ID
0x04	4	Data length xor 0x3A4BFDCC
0x08	4	Data length after 0x0C before compression xor 0x7C2E56D2
0x0C	-	Encrypted data (zlib + RC4)

Table A-2: Format of data received

Offset	Length	Contents
0x00	4	ID
0x04	4	Data length xor 0xC3A2B5D2
0x08	4	Data length after 0x0C before compression xor 0x68FC2AD3

0x0C	-	Encrypted data (zlib + RC4)

Appendix B: Commands

Table B: FileManager commands

Value	Contents
2	SendFilesList
3	UploadToRemote
4	CreateLocalRecvFile
5	WriteLocalRecvFile
7	SendFileData
8	StopTransfer
9	DeleteFile
10	DeleteDirectory
11	GetFileData
12	CreateFolder
13	MoveFile
14	OpenFile（SW_SHOW）
15	OpenFile（SW_HIDE）

Appendix C: C2 servers

- tftpupdate.ftpserver.biz
- 108.61.163.36
- update.centosupdates.com
- 107.191.61.40
- osscach2023.hicloud.tw
- 103.85.24.122
- 106.186.121.154

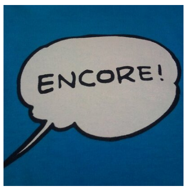
Appendix D: Malware hash value

- 01581f0b1818db4f2cdd9542fd8d663896dc043efb6a80a92aadfac59ddb7684
- 18a696b09d0b7e41ad8ab6a05b84a3022f427382290ce58f079dec7b07e86165
- 15b8dddbfa37317ccdfbc340764cd0f43b1fb8915b1817b5666c4816ccb98e7c

- 849ec6055f0c18eff76170912d8500d3da7be1435a9117d67f2134138c7e70c3
- f19ab3fcbc555a059d953196b6d1b04818a59e2dc5075cf1357cee84c9d6260b
- 836b873ab9807fbdd8855d960250084c89af0c4a6ecb75991542a7deb60bd119
- a69a2b2a6f5a68c466880f4c634bad137cb9ae39c2c3e30c0bc44c2f07a01e8a
- bd02ca03355e0ee423ba0e31384d21b4afbd8973dc888480bd4376310fe6af71

-
- Email

Author



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

Back
Top
Next