HackInBo®

Winter 2025 Edition

25ª EDIZIONE

America's Cyber Defense Agency

NATIONAL COORDINATOR FOR CRITICAL INFRASTRUCTURE SECURITY AND RESILIENCE

Search

HACK IN BO Winter 2025 Edition 25ª EDIZIONE

Topics Spotlight Resources & Tools News & Events Careers About

Home / News & Events / Cybersecurity Advisories / Alert / Widespread Supply Chain Compromise Impacting npm Ecosystem

SHARE:

ALERT

Widespread Supply Chain Compromise Impacting npm Ecos

Release Date: Septembe

THAT'S YOUR OPEN SOURCE, NOT SOMEONE ELSE'S

CISA is releasing this Alert to provide guidance in response to a widespread software supply chain compromise involving the world's largest JavaScript registry, npmjs.com. A self-replicating worm—publicly known as "Shai-Hulud"—has compromised over 500 packages.[i]

After gaining initial access, the malicious cyber actor deployed malware that scanned the environment for sensitive credentials. The cyber actor then targeted GitHub Personal Access Tokens (PATs) and application programming

Hack and Defend (your) Open Source

Roman Zhukov

Principal Architect - Security Communities Lead
ex. - Head of Product Security at Intel's Data Center SW BU
Security Lead and Contributor to Open Source

DISCLAIMER

The opinions expressed are solely my own and do not necessarily reflect the official views or opinions of my current or previous employer(s).

Red Hat

Do you use open source in your work?

1,658 projects scanned by Black Duck audits

97% of the codebases contained open source

70% of scanned code had its origin in open source

An average **911** OSS components were found per application

64% of OSS components were transitive dependencies

86% of the codebases contained at least one vulnerability

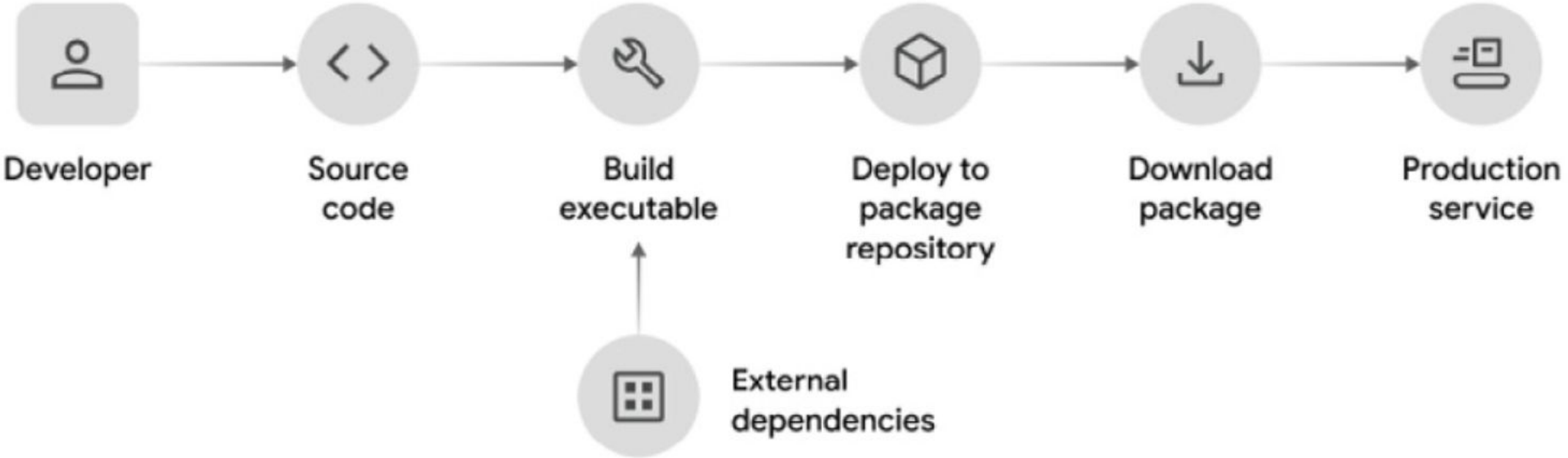
81% of the codebases contained high- or critical-risk vulnerabilities

Maintenance and Operational Risk

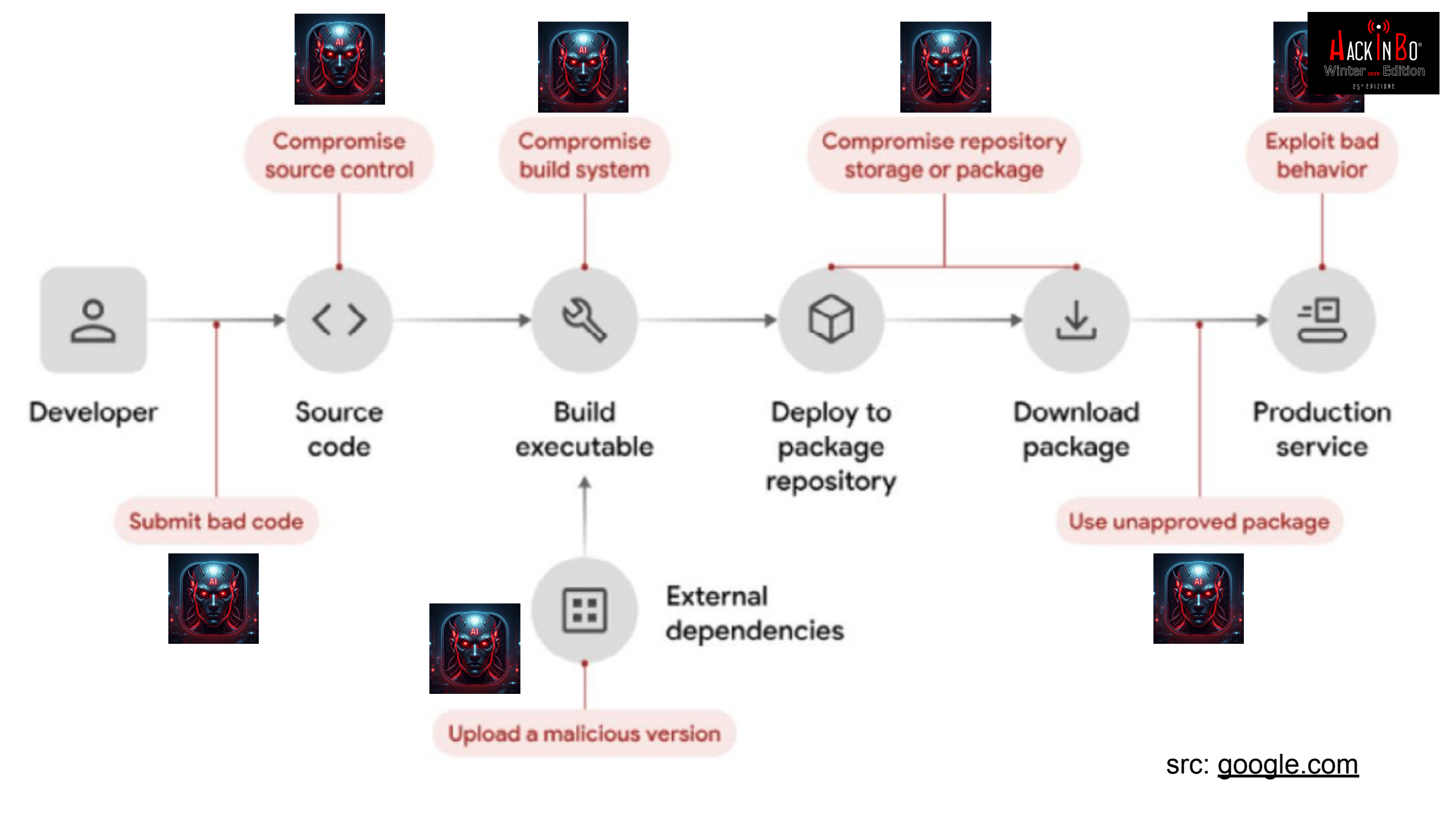
91% of all codebases contained outdated OSS components

90% of all codebases contained components more than 10 versions behind the most current version

2025 Open Source Security and Risk Analysis Report, blackduck.com



src: google.com



src: google.com

The Top 10 OSS Risks

OWASP

owasp.org/www-project-open-source-software-top-10/

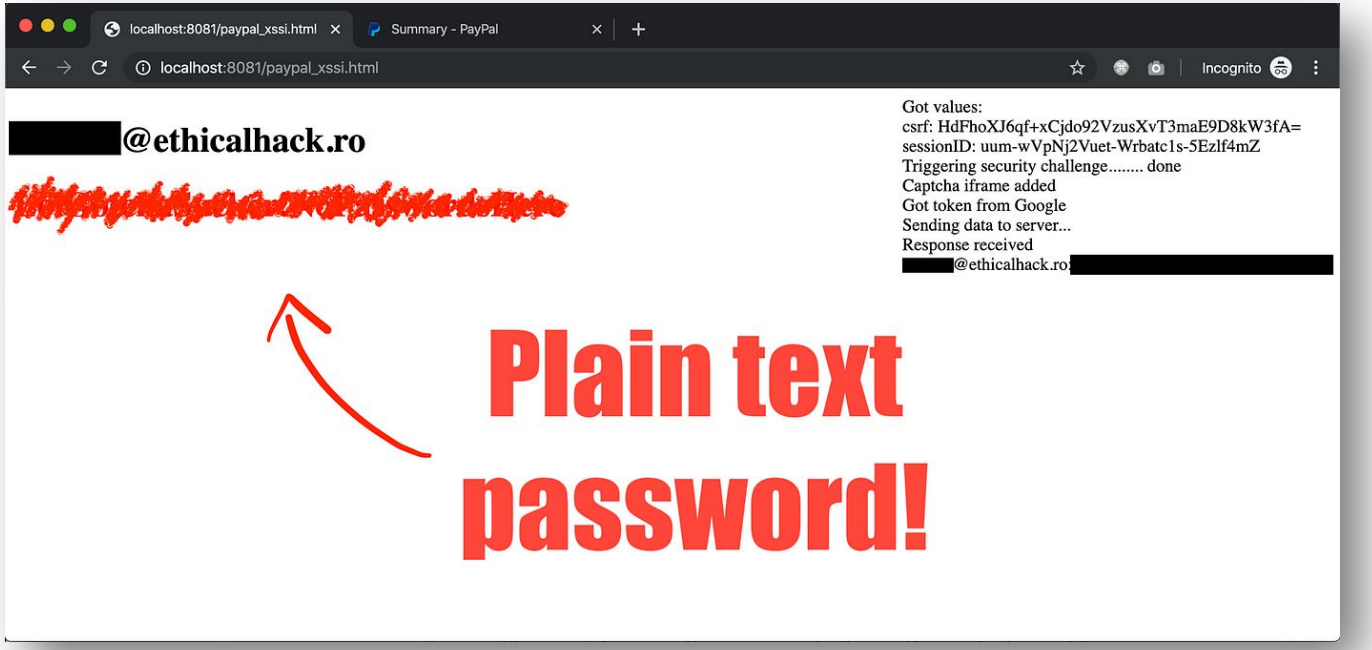
OSS-RISK-1	Known Vulnerabilities
OSS-RISK-2	Compromise of Legitimate Package
OSS-RISK-3	Name Confusion Attacks
OSS-RISK-4	Unmaintained Software
OSS-RISK-5	Outdated Software
OSS-RISK-6	Untracked Dependencies
OSS-RISK-7	License Risk
OSS-RISK-8	Immature Software
OSS-RISK-9	Unapproved Change (Mutable)
OSS-RISK-10	Under/Over-Sized Dependency

INSPIRED BY A TRUE STORY
Homemadesoul

HACK IN BO Winter 2025 Edition 25ª EDIZIONE

1 Dependency confusion

> npm install **analytics-paypal**
// Used to launch malicious script to make DNS requests to dns.alexbirsan-hacks-paypal.com to run data exfiltration. Now deprecated.

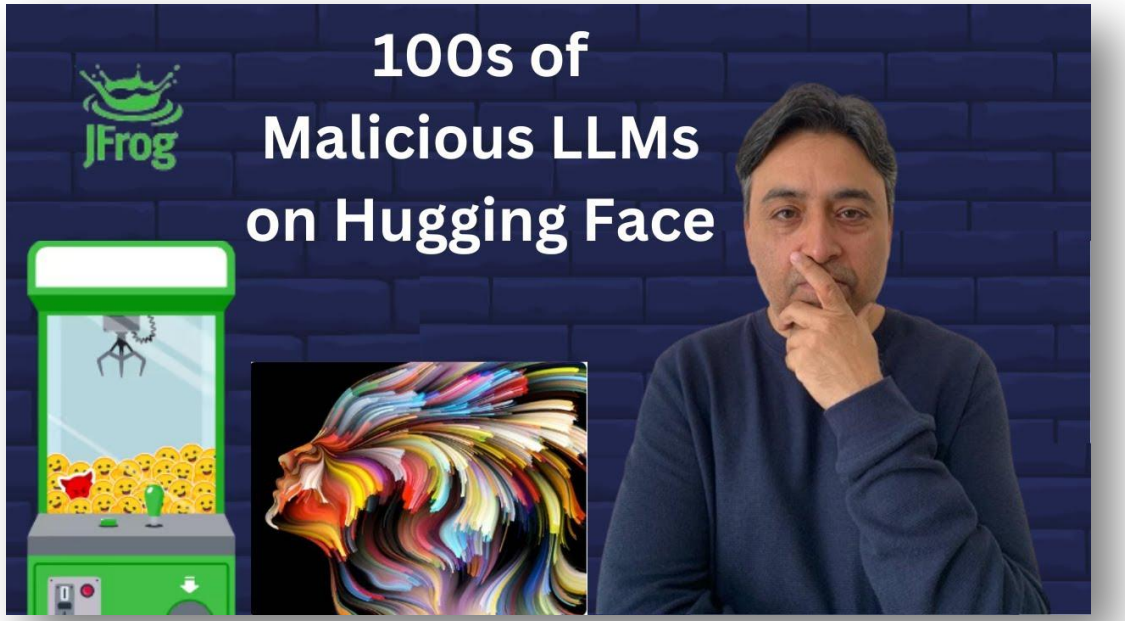


INSPIRED BY A TRUE STORY
Homemadesoul

HACK IN BO Winter 2025 Edition 25ª EDIZIONE

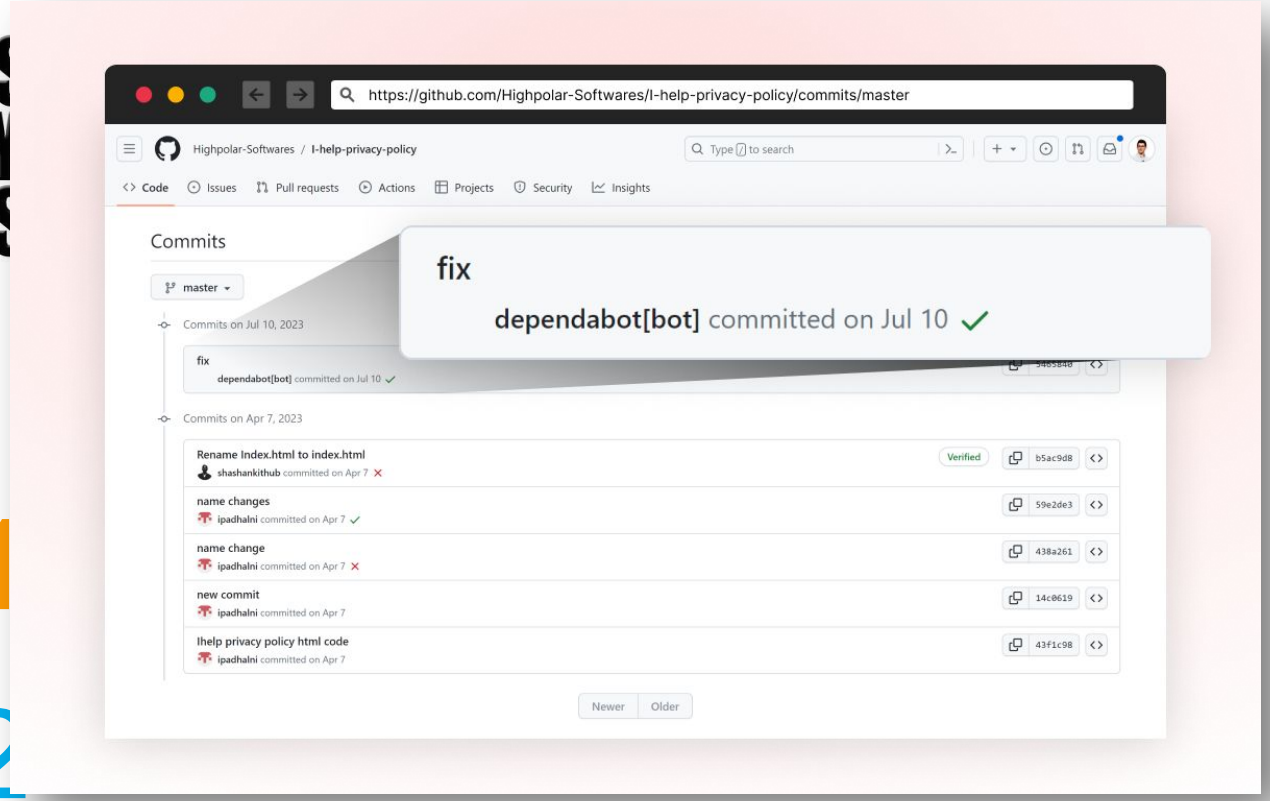
1 Dependency confusion

2 Typosquatting



EleuterAI instead of Eleu**th**erAI
//fake LLM model was added to Hugging Face

INS
BY
S

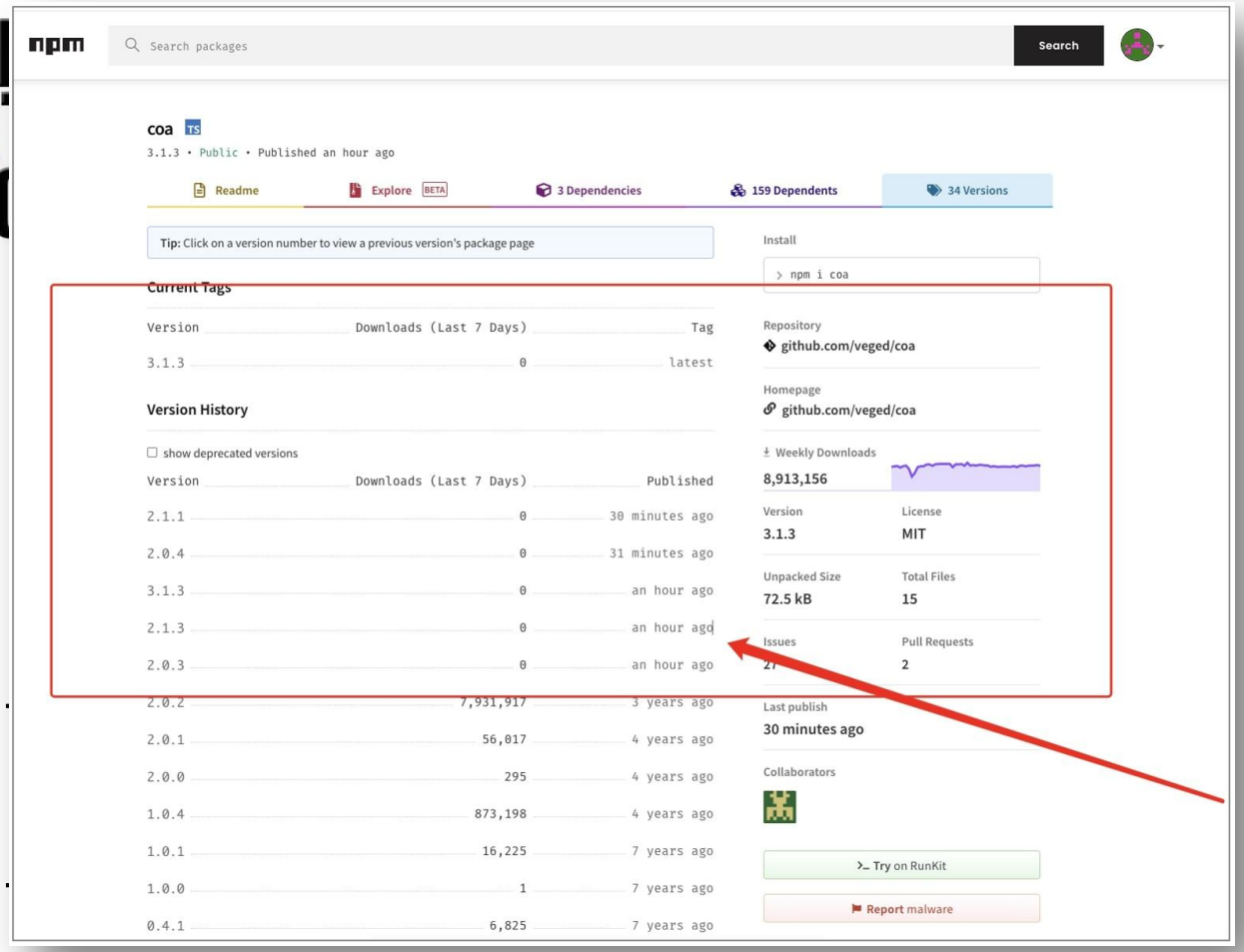


1

2

3 Trojan injection

> **dependabot**[bot] committed *//Malicious pull request that's accepted contains info stealer*



1

2

3

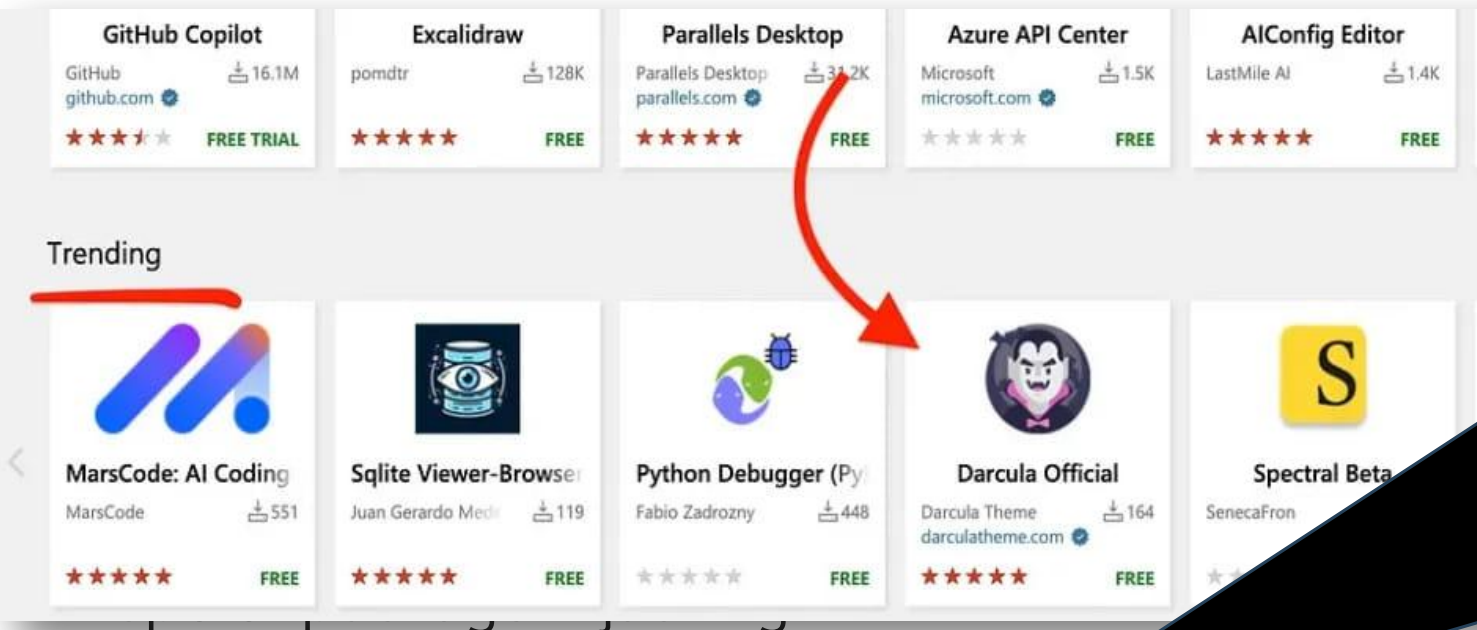
4 Repo or package Hijacking

> **coa**
was hijacked over npm account takeover incorporating password stealer
//command line option parser with over 9 Million weekly downloads

INSPIRED BY A TRUE STORY
Homemadesoul

HACK IN BO Winter 2025 Edition 25ª EDIZIONE

1 2 3 4



> “Dracula Official” is a popular colour **VSCode** Marketplace **plugin** with **over 7 million installs**.
*//Malicious “**Dar**cula Official” is an infostealer sends user’s data to a remote server via an HTTPS POST*

5 Dev environment compromise

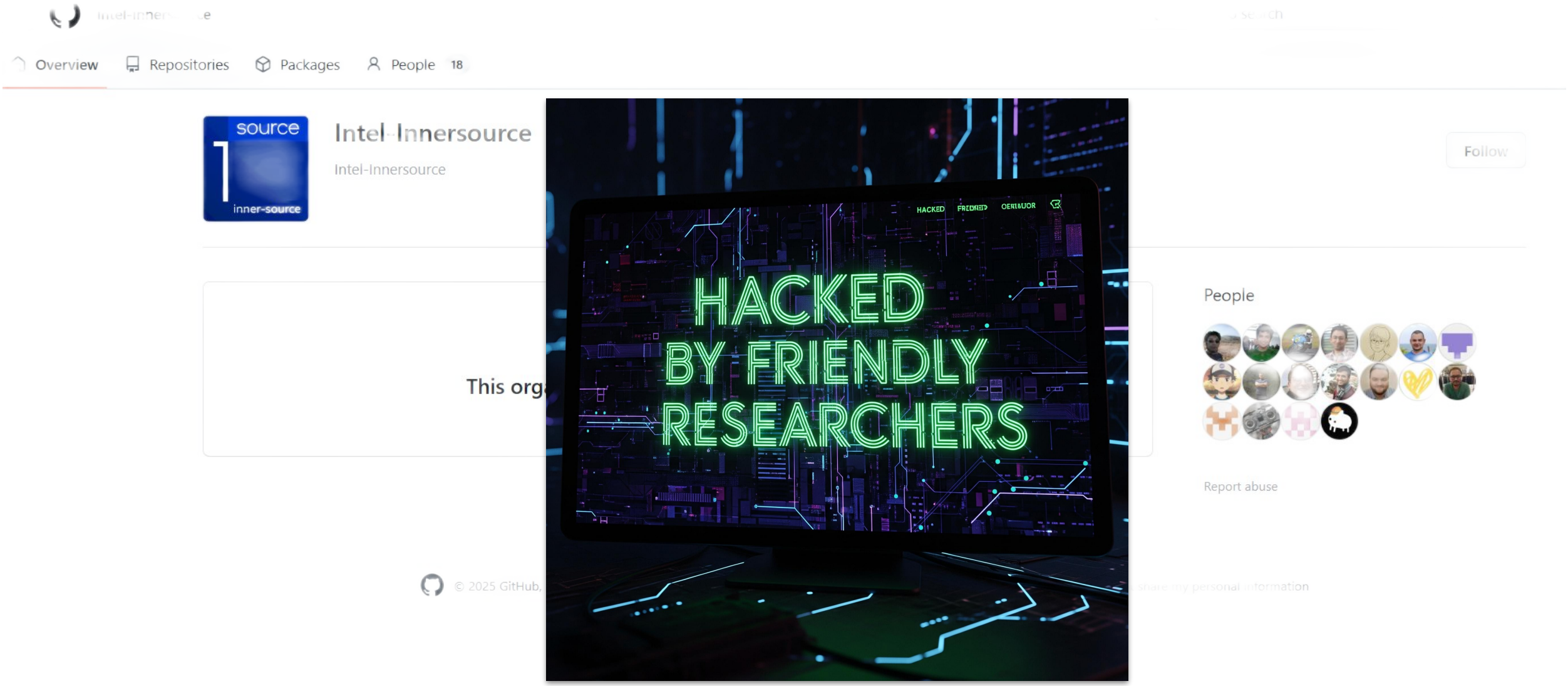
Top 10 CI/CD Security Risks

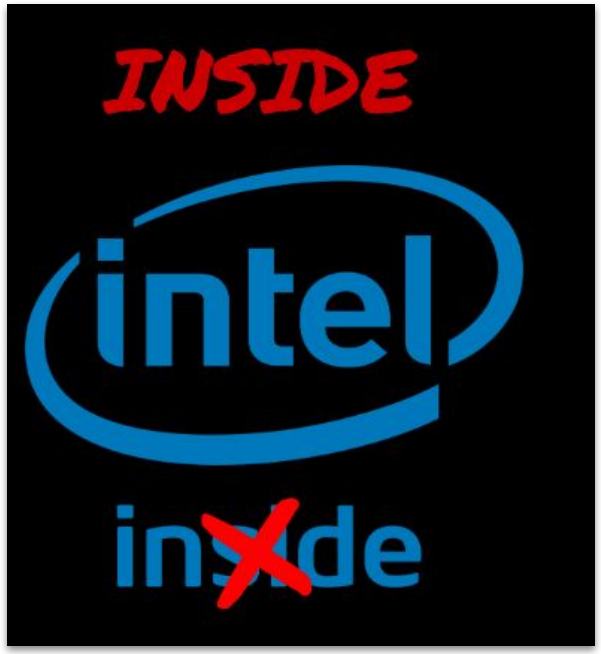
owasp.org/www-project-top-10-ci-cd-security-risks/

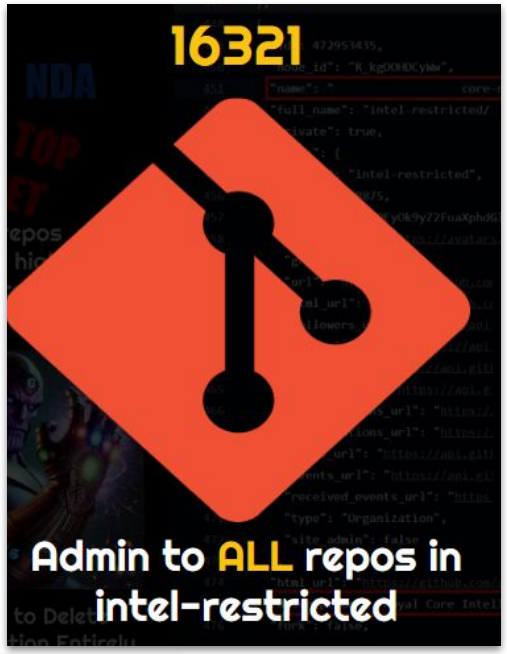
OWASP

CICD-SEC-1	Insufficient Flow Control Mechanisms
CICD-SEC-2	Inadequate Identity and Access Management
CICD-SEC-3	Dependency Chain Abuse
CICD-SEC-4	Poisoned Pipeline Execution (PPE)
CICD-SEC-5	Insufficient PBAC (Pipeline-Based Access Controls)
CICD-SEC-6	Insufficient Credential Hygiene
CICD-SEC-7	Insecure System Configuration
CICD-SEC-8	Ungoverned Usage of 3rd Party Services
CICD-SEC-9	Improper Artifact Integrity Validation
CICD-SEC-10	Insufficient Logging and Visibility

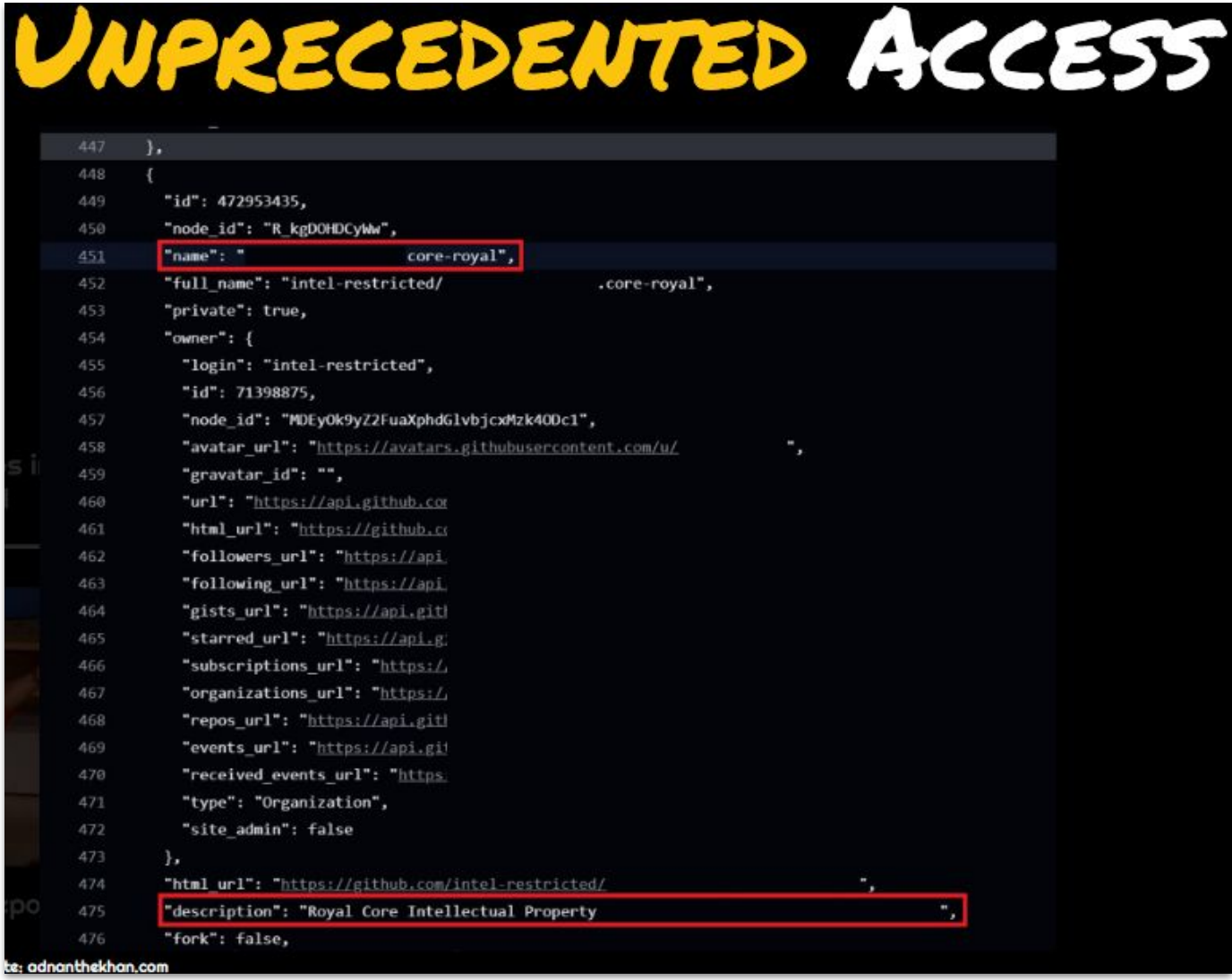
Usual Tuesday Morning

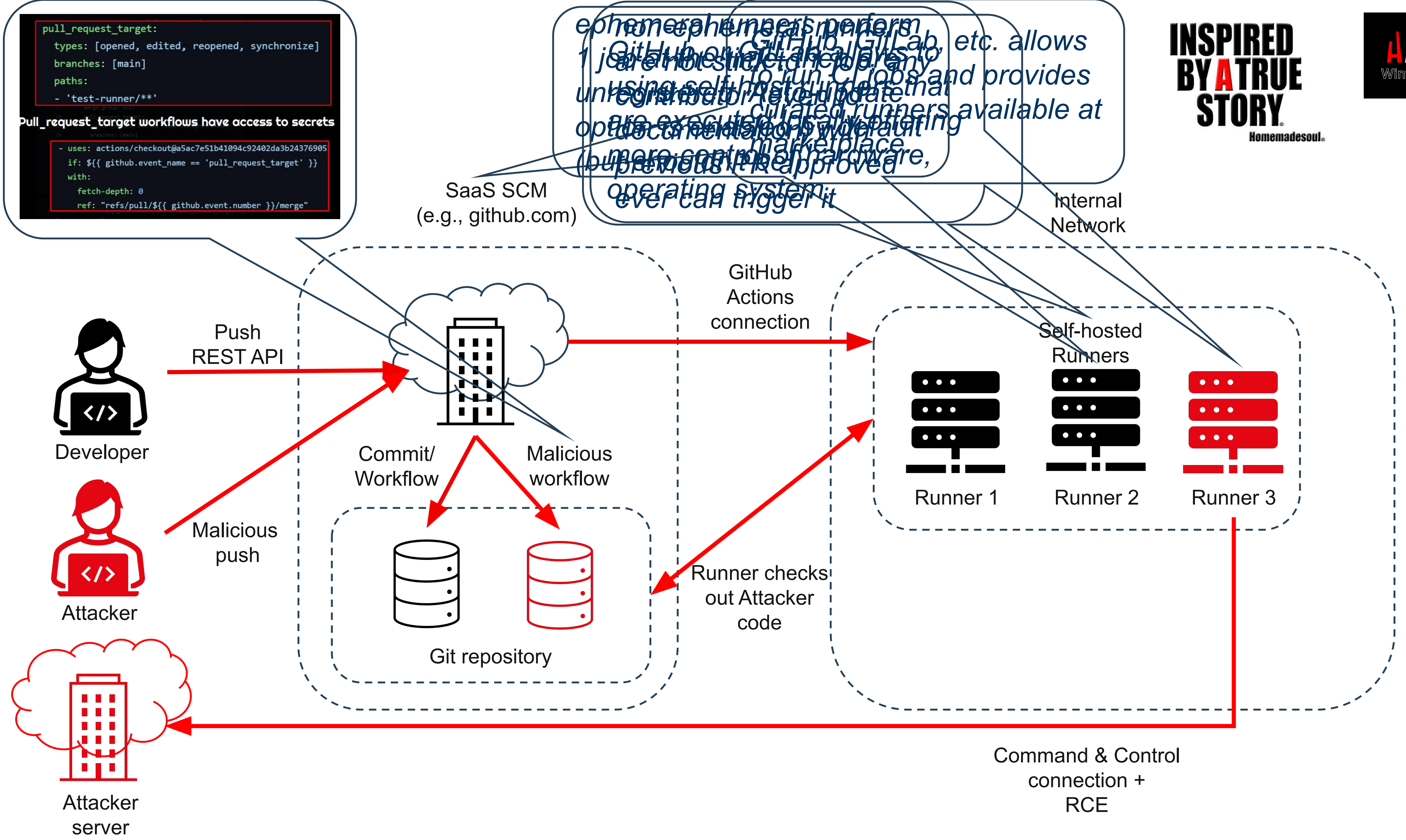



INSIDE
intel
inside


16321
Admin to ALL repos in intel-restricted


OPEN SOURCE
ALL THE REPOS
Ability to make all repos public

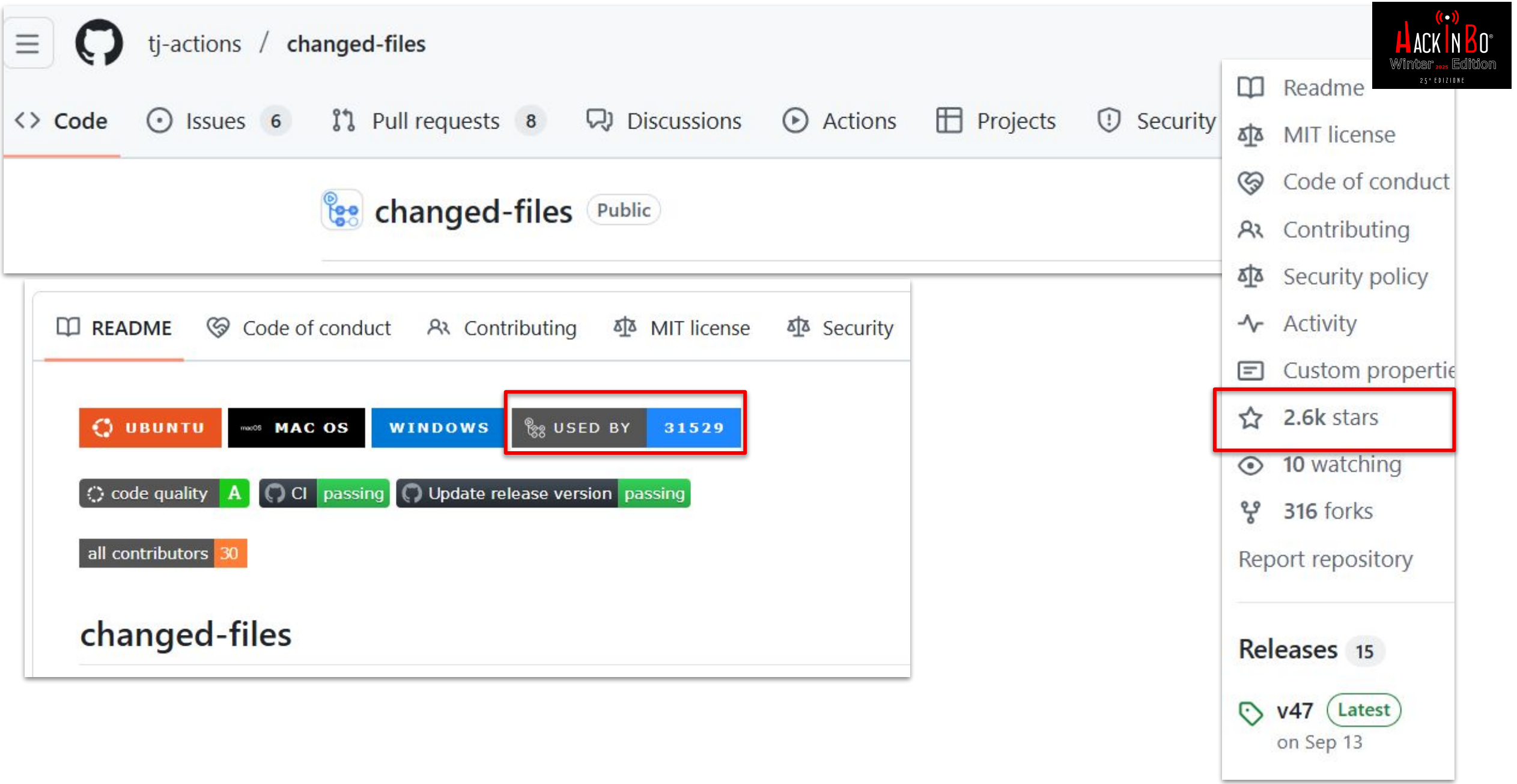

UNPRECEDENTED ACCESS
"name": " core-royal",
"full_name": "intel-restricted/ .core-royal",
"private": true,
"owner": {
"login": "intel-restricted",
"id": 71398875,
"node_id": "MDEyOk9yZ2FuaXphdGlvbjcxMzk4ODc1",
"avatar_url": "https://avatars.githubusercontent.com/u/ ",
"gravatar_id": "",
"html_url": "https://github.com/intel-restricted/ ",
"description": "Royal Core Intellectual Property ",
"fork": false,



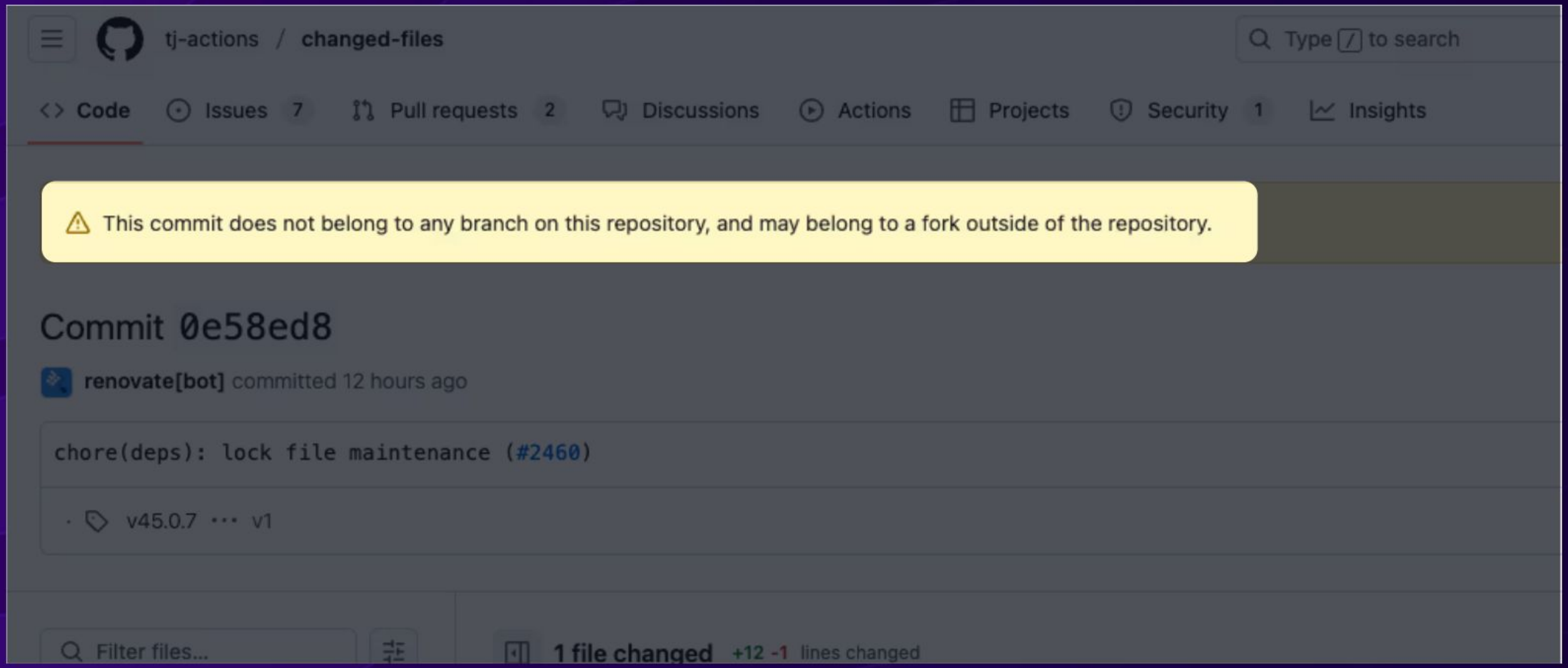
INSPIRED BY A TRUE STORY
Homemadesoul

HackInBo Winter 2025 Edition
25° EDIZIONE

How do you choose an open source?

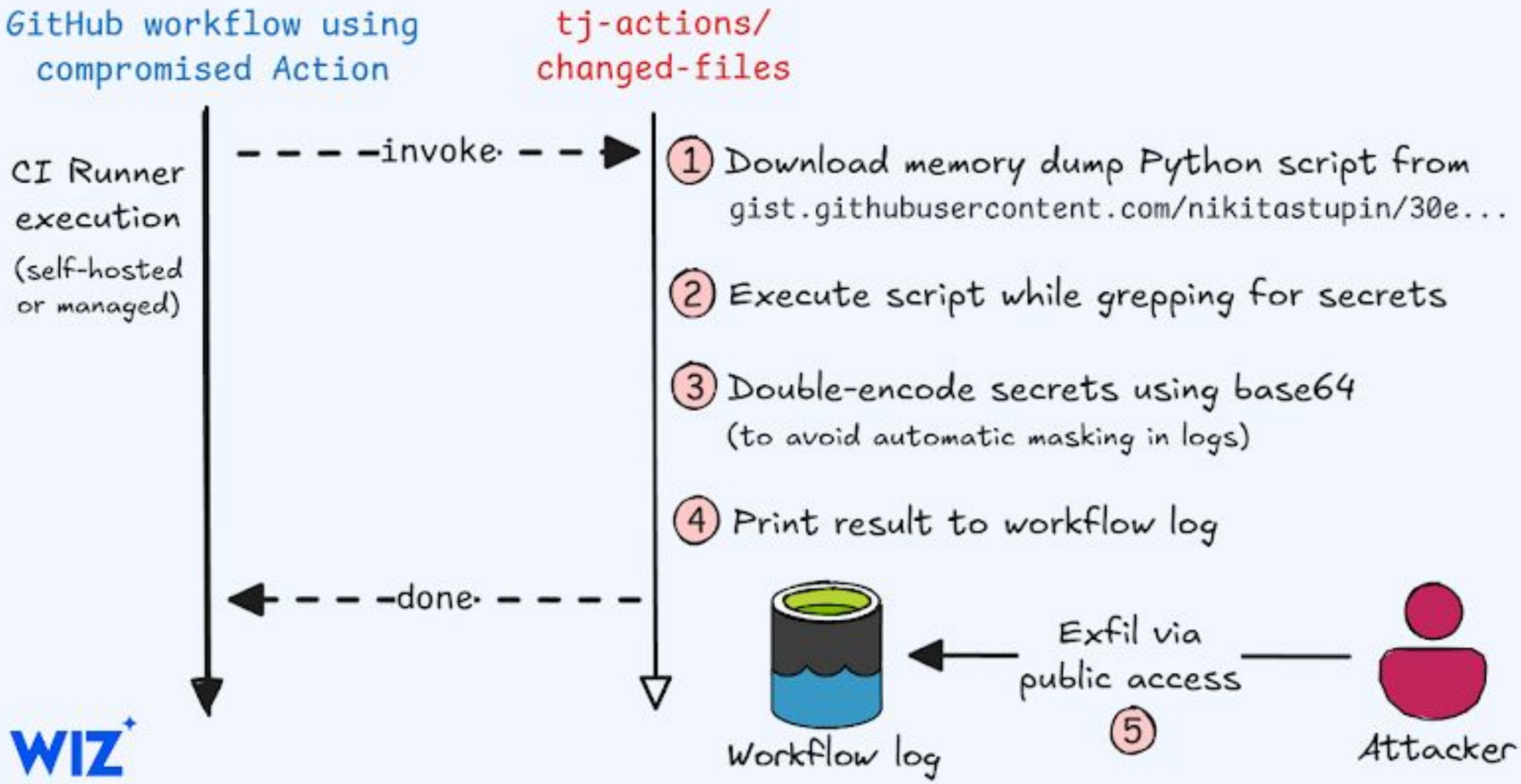

tj-actions / changed-files
Code
Issues 6
Pull requests 8
Discussions
Actions
Projects
Security
changed-files
Public
README
Code of conduct
Contributing
MIT license
Security
UBUNTU
MAC OS
WINDOWS
USED BY 31529
code quality A
CI passing
Update release version passing
all contributors 30
changed-files
Readme
MIT license
Code of conduct
Contributing
Security policy
Activity
Custom properties
2.6k stars
10 watching
316 forks
Report repository
Releases 15
v47 Latest
on Sep 13

The Malicious Imposter Commit



tj-actions/changed-files Supply Chain Attack

Post-compromise secret exfiltration flow from dependent repositories



How to “hack” open source?

README CC-BY-SA-4.0 license

Offensive AI Compilation

A curated list of useful resources that cover Offensive AI.

Contents

🚫 Abuse 🚫

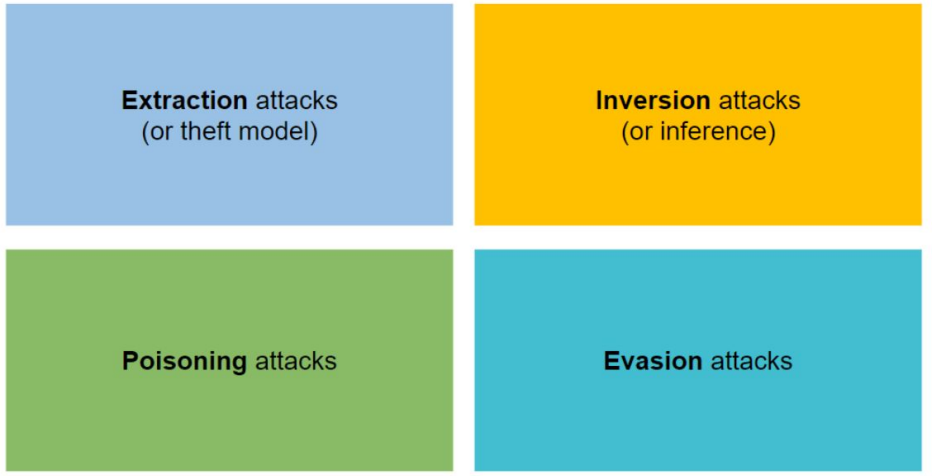
Exploiting the vulnerabilities of AI models.

🧠 Adversarial Machine Learning 🧠

Adversarial Machine Learning is responsible for assessing their weaknesses and providing countermeasures.

⚡ Attacks ⚡

It is organized into four types of attacks: extraction, inversion, poisoning and evasion.



🌐 OSINT 🌐

- SNAP_R: Generate automatically spear-phishing posts on social media. stars 150
- SpyScrap: SpyScrap combines facial recognition methods to filter the results and uses natural language processing to obtain important entities from the website the user appears. stars 201

🎯 Phishing 🎯

- DeepDGA: Implementation of DeepDGA: Adversarially-Tuned Domain Generation and Detection. stars 26
- ScamAgents: How AI Agents Can Simulate Human-Level Scam Calls

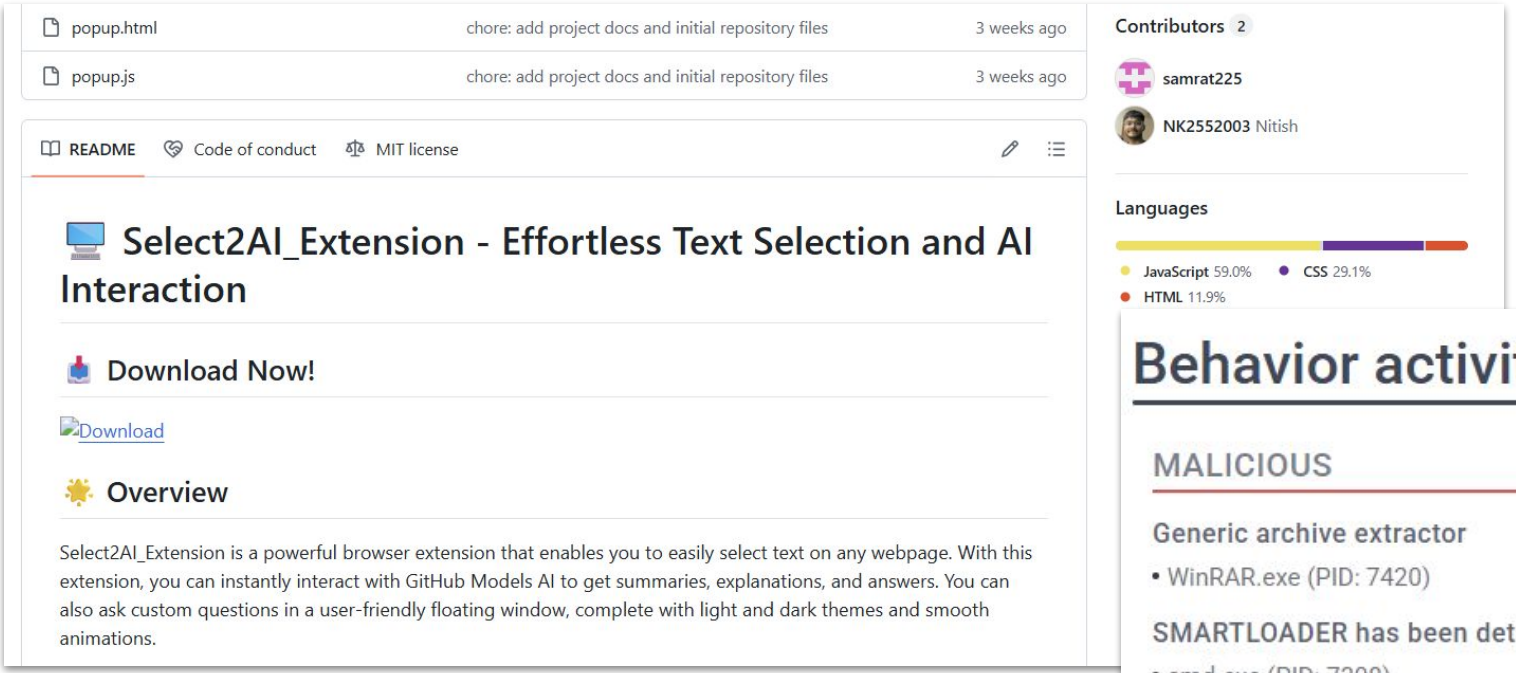
🕵️ Threat Intelligence 🕵️

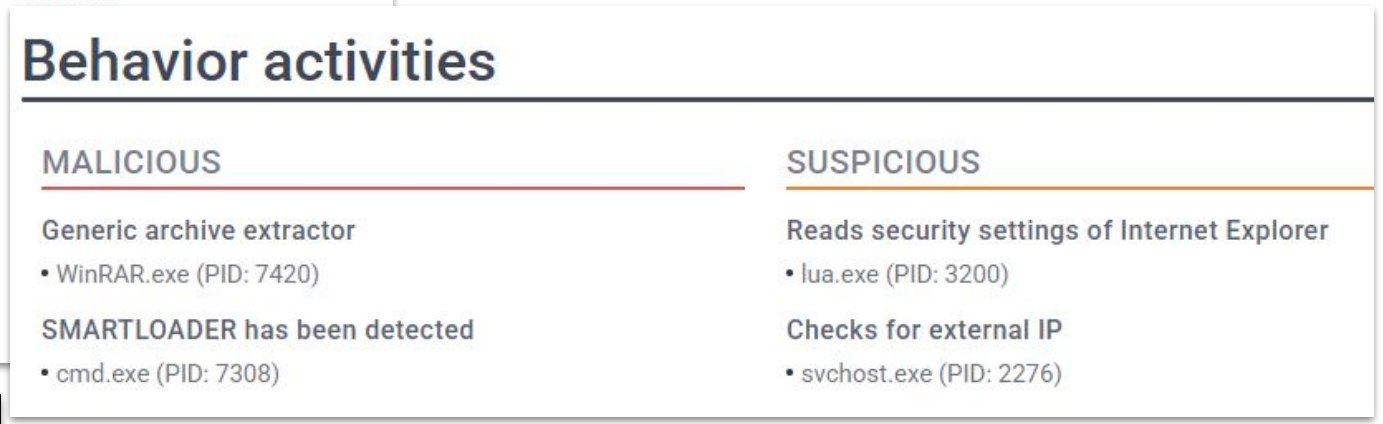
- From Sands to Mansions: Enabling Automatic Full-Life-Cycle Cyberattack Construction with LLM

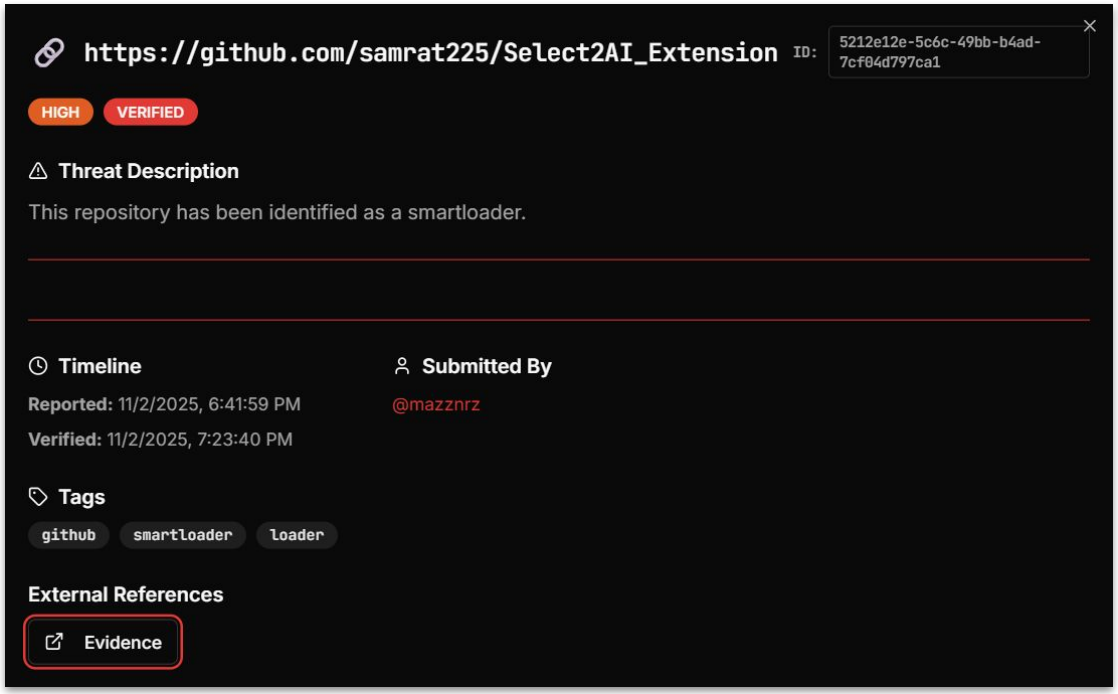
🌀 Side channels 🌀

- SCAAML: Side Channel Attacks Assisted with Machine Learning. stars 171

github.com/jiep/offensive-ai-compilation







opensourcemalware.com

Defense Frameworks and Tools

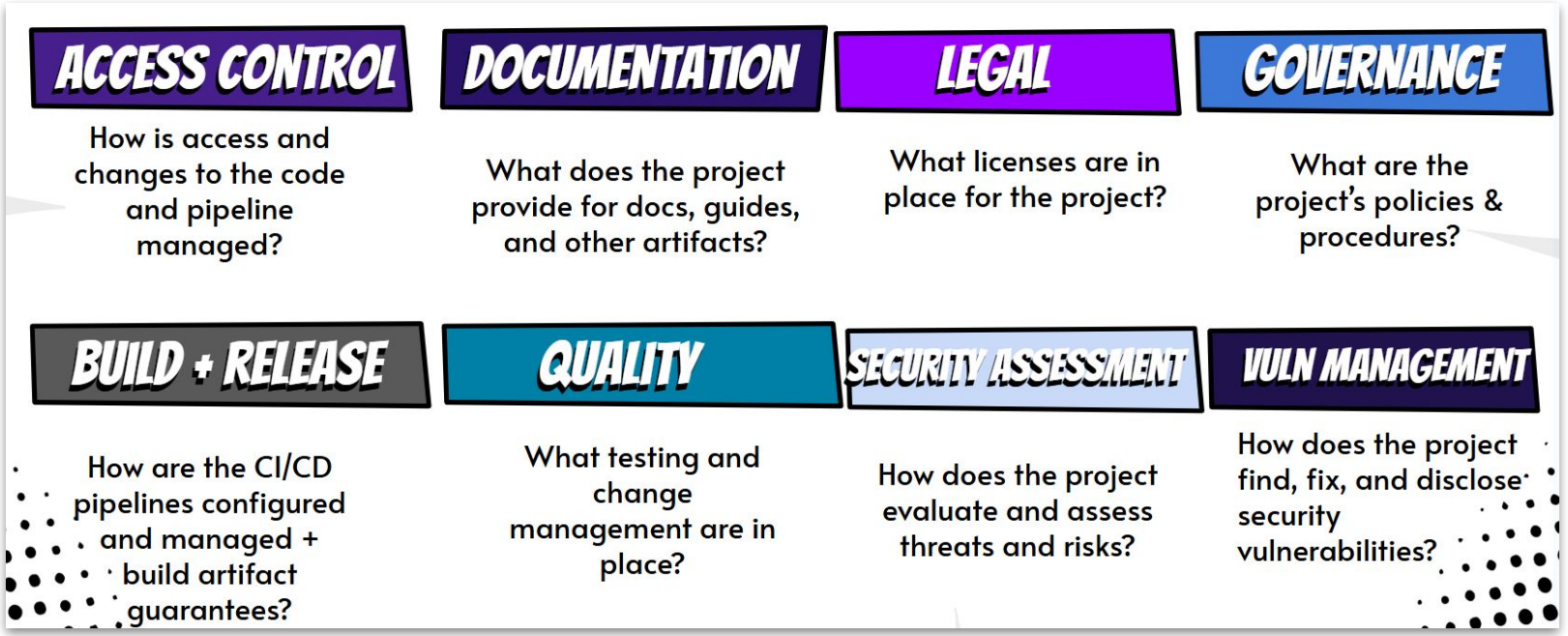
Secure Supply Chain Consumption Framework (S2C2F)

github.com/ossf/s2c2f

Level 1	Level 2	Level 3	Level 4
Minimum OSS Governance Program • Use package managers [ING-1] • Local copy of artifact [ING-2] • Scan with known vulns [SCA-1] • Scan for software licenses [SCA-2] • Inventory OSS [INV-1] • Manual OSS updates [UPD-1]	**Secure Consumption and Improved MTTR** • Scan for end of life [SCA-3] • Have an incident response plan [INV-2] • Auto OSS updates [UPD-2] • Alerts on vulns at PR time [UPD-3] • Audit that consumption is through approved ingestion method [AUD-2] • Validate integrity of OSS [AUD-3] • Secure package source file configuration [ENF-1]	**Malware Defense and Zero-Day Detection** • Deny list capability [ING-3] • Clone OSS source [ING-4] • Scan for malware [SCA-4] • Proactive security reviews [SCA-5] • Enforce OSS provenance [AUD-1] • Enforce consumption from curated feed [ENF-2]	**Advanced Threat Defense** • Validate the SBOMs of OSS consumed [AUD-4] • Rebuild OSS on trusted infrastructure [REB-1] • Digitally sign rebuilt OSS [REB-2] • Generate SBOM for rebuilt OSS [REB-3] • Digitally sign protected SBOMs [REB-4] • Implement fixes [FIX-1]

Security Baseline - OSPS

62 requirements across 3 levels of maturity covering 8 areas of security practices



github.com/ossf/security-baseline

OSPS-AC-02.01: When a new collaborator is added, the version control system MUST require manual permission assignment, or restrict the collaborator permissions to the lowest available privileges by default.

Focused - no SHOULD, only MUST
Realistic - practical to implement

OWASP Secure Pipeline

1 Plan → 2 Develop → 3 Integrate → 4 Release → 5 Operate

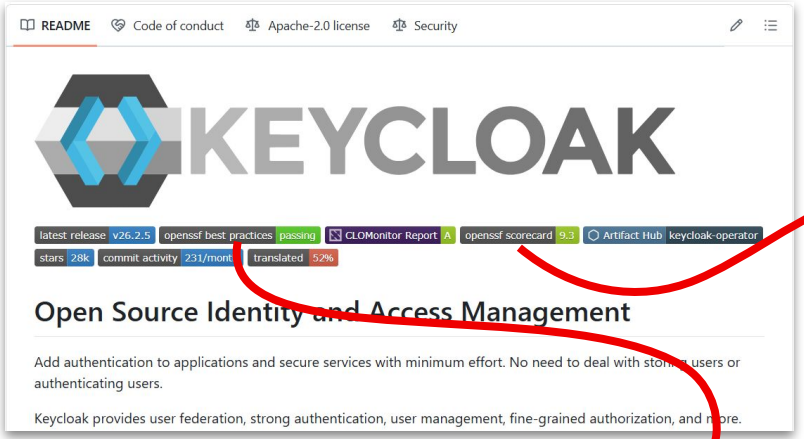
Just released

Level	Purpose	Typical Users	Example Practices
Level 1 – Foundational	Establishes baseline pipeline hygiene. Focuses on simple, enforceable controls.	Small teams or organizations building their first secure pipeline.	MFA, endpoint protection, source control hardening, approved tooling.
Level 2 – Standard	Expands to automation, monitoring, and evidence of enforcement.	Mature DevSecOps teams managing regulated workloads.	Automated secret scanning, policy-as-code enforcement, release reviews.
Level 3 – Advanced	Demonstrates continuous verification, independence, and assurance at scale.	Enterprises or high-risk environments.	Automated gating, audit logging, verified builds, third-party attestations.

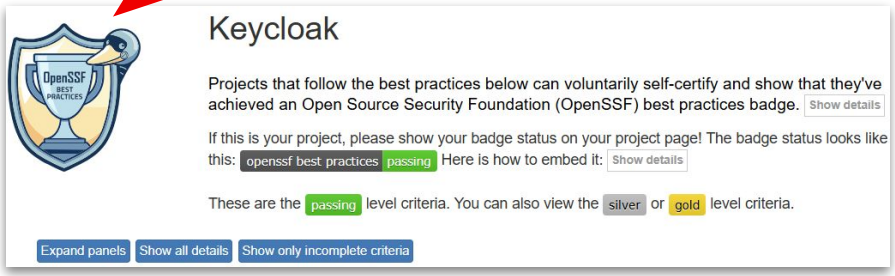
github.com/OWASP/www-project-spvs

Automate It

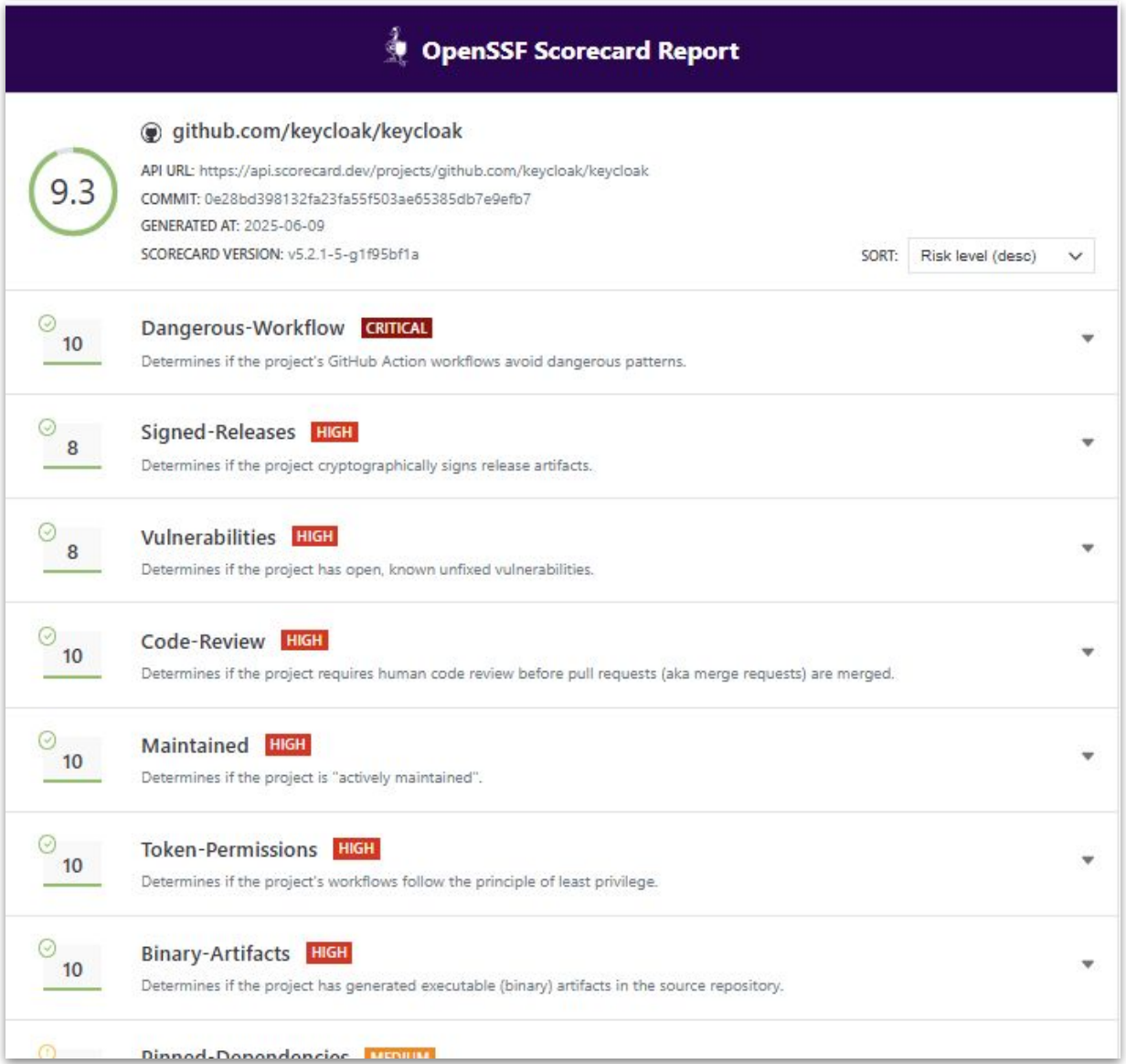
github.com/ossf/scorecard



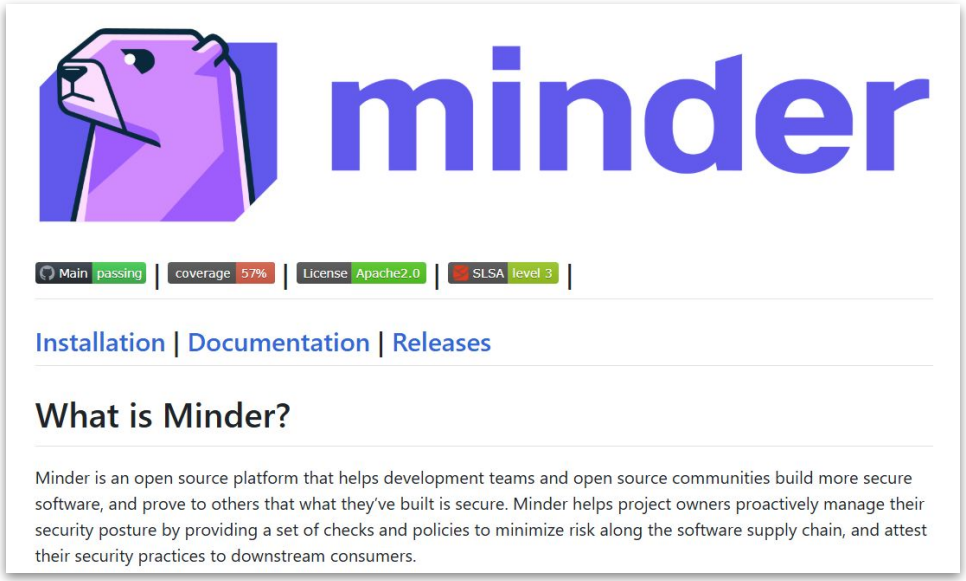
github.com/keycloak/



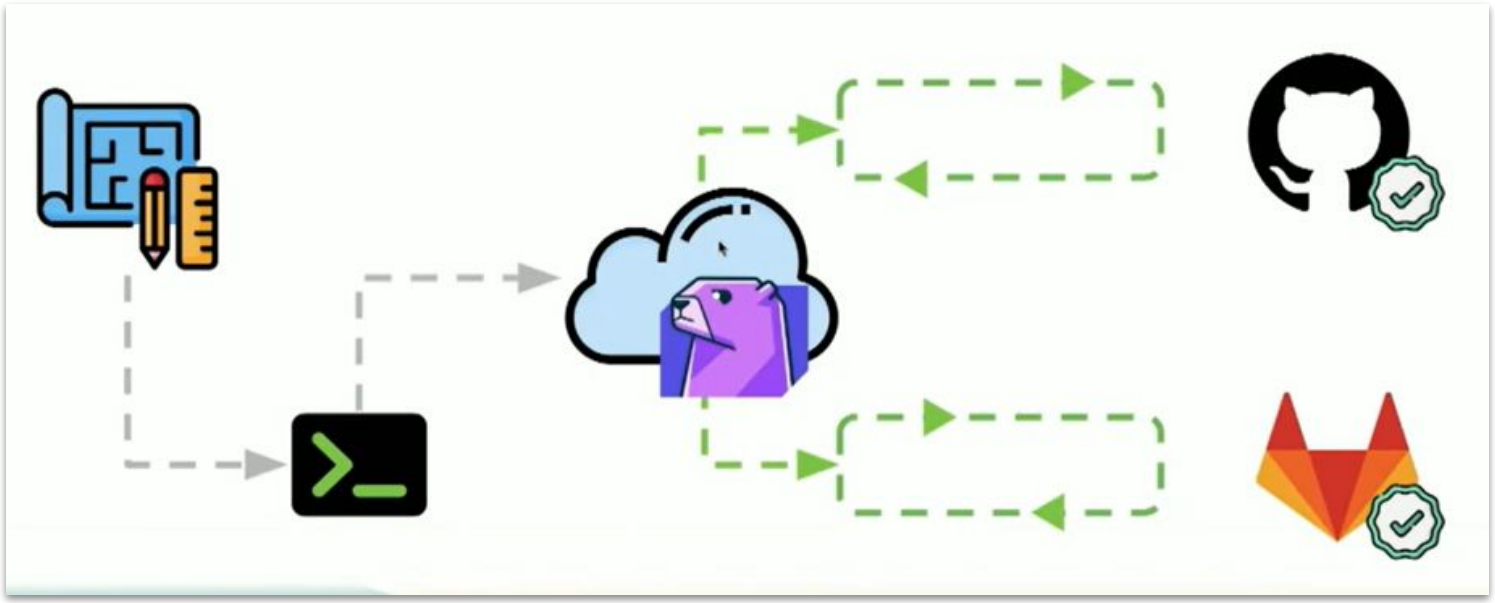
bestpractices.dev



Automate It



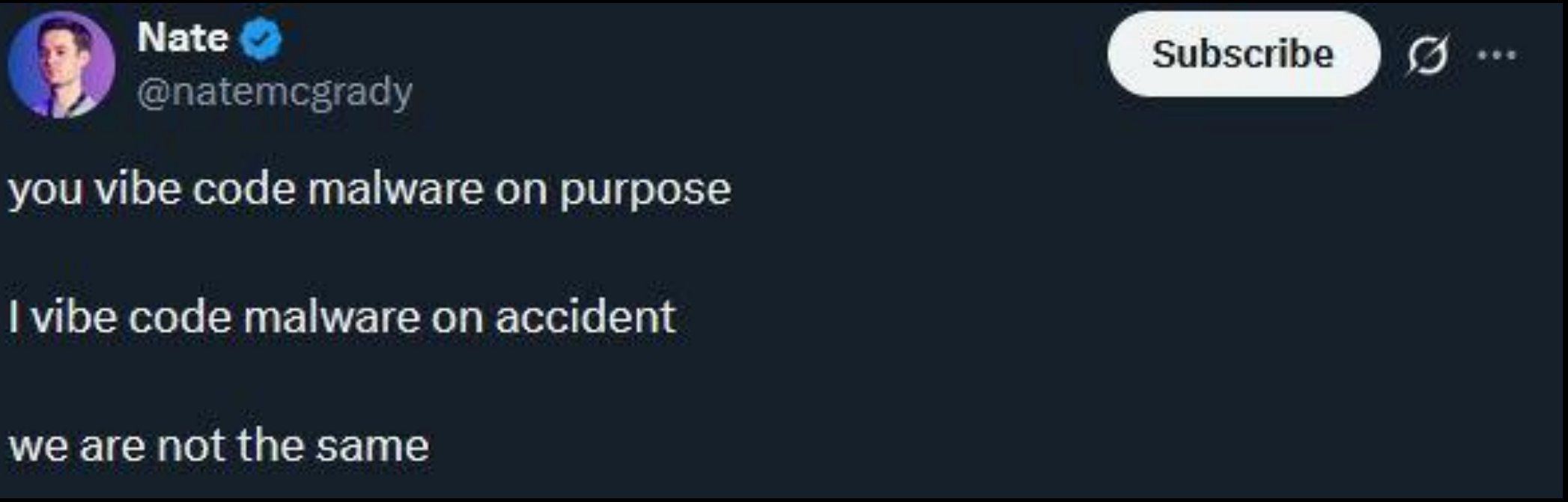
github.com/mindersec/minder

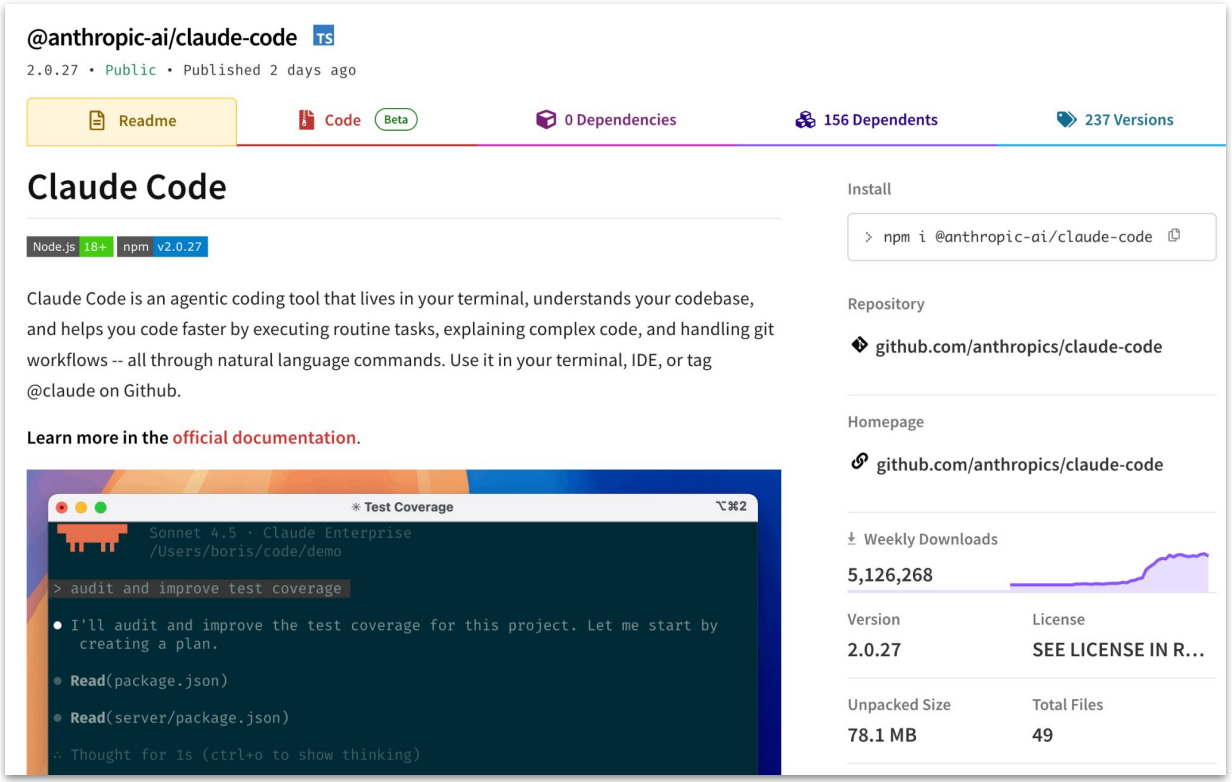


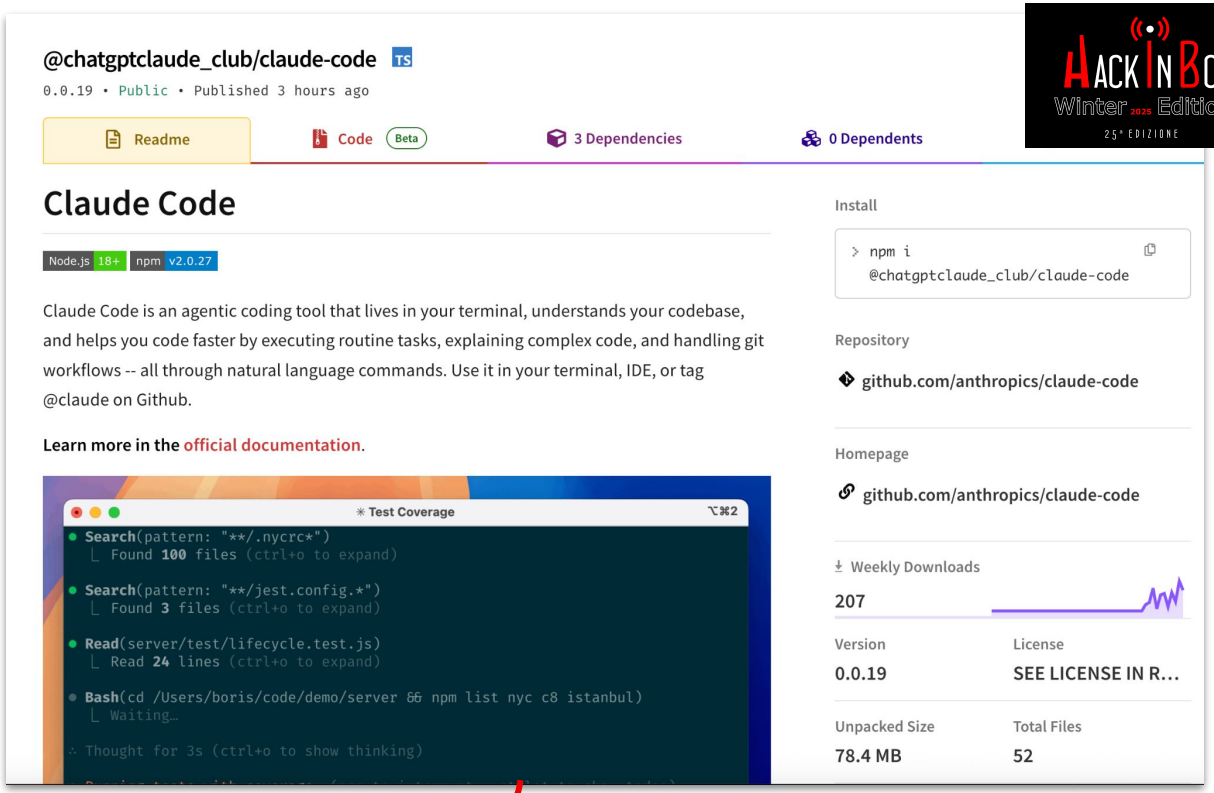
minder-rules-and-profiles / rule-types / **github** /

- trufflehog_github_action.testdata
- README.md
- actions_check_default_permissions.yaml
- actions_check_pinned_tags.yaml
- allowed_selected_actions.yaml
- artifact_attestation_slsa.yaml
- artifact_signature.yaml

Wait. How about AI?

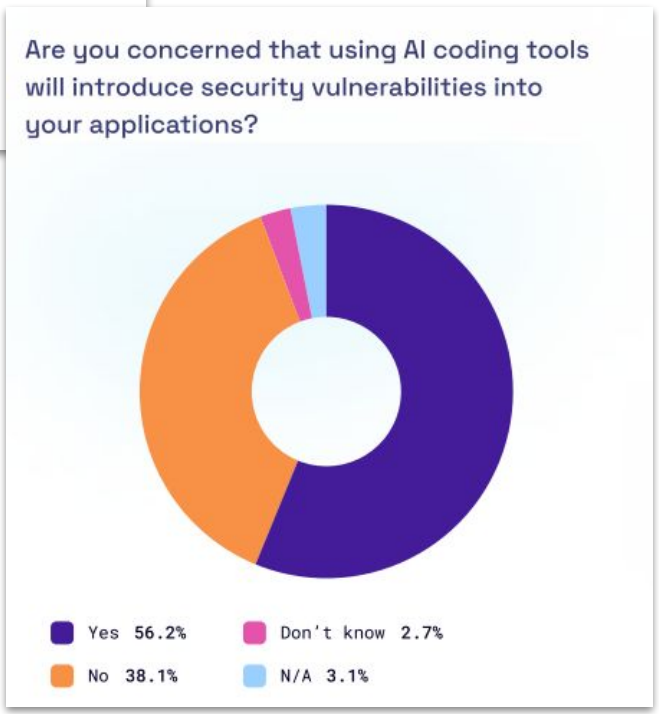

Nate
@natemcgrady
Subscribe
you vibe code malware on purpose
I vibe code malware on accident
we are not the same





getsafety.com

- This npm package deploys a payload that seeks Claude Code installations
- Steal Anthropic credentials
- Setup a proxy between legitimate *api.anthropic.com*
- Includes a bidirectional C2 server *claude-code.chatgptclaude.club*



2024 State of open source security report, snyk.io

1. TL;DR

Short on time? Here's what really matters:

- **You Are the Developer – AI is the Assistant:** The developer (you) remains in full control of the code, and you a
that may be caused by the code. Critically evaluate and edit AI-generated code just as you would code writter
never blindly accept suggestions in situations where that could eventually cause harm. [ifip2021] [anssibsi20
- **Apply Engineering Best Practices Always:** AI-generated code isn't a shortcut around engineering processes s
static analysis, documentation, and version control discipline. [markvero2025a]
- **Be Security-Conscious:** Assume AI-written code can have bugs or vulnerabilities, because it often does. AI co
security issues like using outdated cryptography or outdated dependencies, ignoring error handling, or leakin
or sensitive data in the suggested code. Make sure dependency suggestions are safe and not pulling in known
[shihchiehdai2025a], [anssibsi2024b]
- **Guide the AI:** AI is a powerful assistant, but it works best with your guidance. Write clear precise prompts that specify security requirements. Don't hesitate to modify or reject AI outputs. Direct your AI tool to build its own instructions file based on this guide. [swaroopdora2025a] [haoyan2025a]
- **Ask the AI to review and improve its own work**. Once you have some AI-written code, where possible, ask it to review and improve its own work (repeating these steps as necessary). This technique is sometimes called Recursive Criticism and Improvement (RCI) and can be remarkably effective. For instance, "Review your previous answer and find problems with your answer" followed by "Based on the problems you found, improve your answer" for one or more iterations. Encourage the use of tools such as linters, SAST, dependency checkers, etc. through the improvement cycles. [catherinetony2024a]
- **Express your concerns to the AI**. If you have concerns about something AI has generated, express your concerns in detail, and ask it to analyze that code to determine whether or not it's okay. Include relevant information to increase the likelihood of a useful response. Ensure that if something is stated as a fact, it's actually a fact. Review that answer.

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

New OpenSSF Guidance on AI Code Assistant Instructions

best.openssf.org/Security-Focused-Guide-for-AI-Code-Assistant-Instructions

Key Takeaways

- **Understand** where the risks come from for you. Repeat.

https://www.redhat.com/en/resources/product-security-risk-report-2024

- **Never stop** hacking - test often. It's open [source] for everybody.

- **AI is attack amplifier.** Keep in mind basic security principles. Educate others.

- **Automate it** with a bunch of decent open source tools available for you.

Quiz time - top CWE in the Age of AI

CWE	Percentage of codebases with vulns linked to CWE	Description
CWE-400	70% 	**Uncontrolled Resource Consumption:** The software does not properly control the allocation and release of system resources, such as memory, CPU time, or disk space. This can lead to DoS attacks.
CWE-200	60% 	**Exposure of Sensitive Information to an Unauthorized Actor:** The software exposes sensitive information, such as passwords, credit card numbers, or personal data, to unauthorized actors. This can happen through various means, such as insecure storage, unencrypted transmission, or improper access controls.
CWE-79	56% 	**Improper Neutralization of Input During Web Page Generation (Cross-Site Scripting):** The software does not neutralize or incorrectly neutralizes user-supplied input before including that input in an HTML page. This allows attackers to inject malicious scripts that can steal user data or take control of their browser.
CWE-185	48% 	**Incorrect Regular Expression:** The software specifies a regular expression in a way that causes data to be improperly matched or compared. Regular expressions should be subjected to thorough testing techniques such as equivalence partitioning, boundary value analysis, and robustness testing.

2025 Open Source Security and Risk Analysis Report, blackduck.com

SECURE OPEN SOURCE.
IT'S YOURS, NOT
SOMEONE ELSE'S.

HACK IN BO®
Winter 2025 Edition
25ª EDIZIONE

LINKEDIN.COM/IN/ROZHUKOV

Red Hat