WELCOME TO Fabulous TECHORAMA UTRECHT

Protect yourself against supply chain attacks

Rob Bos

DevOps Consultant – Xpirit

The Netherlands

https://devopsjournal.io

@robbos81

https://myoctocat.com

Topics

- Why:
  - Supply chain
  - Attack examples
- Protection / Maturity: frameworks
  - OWASP Software Component Verification Standard (SCVS)
  - Supply chain Levels for Software Artifacts (SLSA)

Supply chain – Dependencies

What comes to mind first?

Packages

Container images

CI/CD pipelines

Supply chain – Dependencies

- Libraries used by your application:
  - Authentication
  - Encryption
  - Database connections

$\Rightarrow$ Package managers

- Tooling used for building your application:
  - npm ci
  - dotnet build
  - pipelines
  - SDK’s

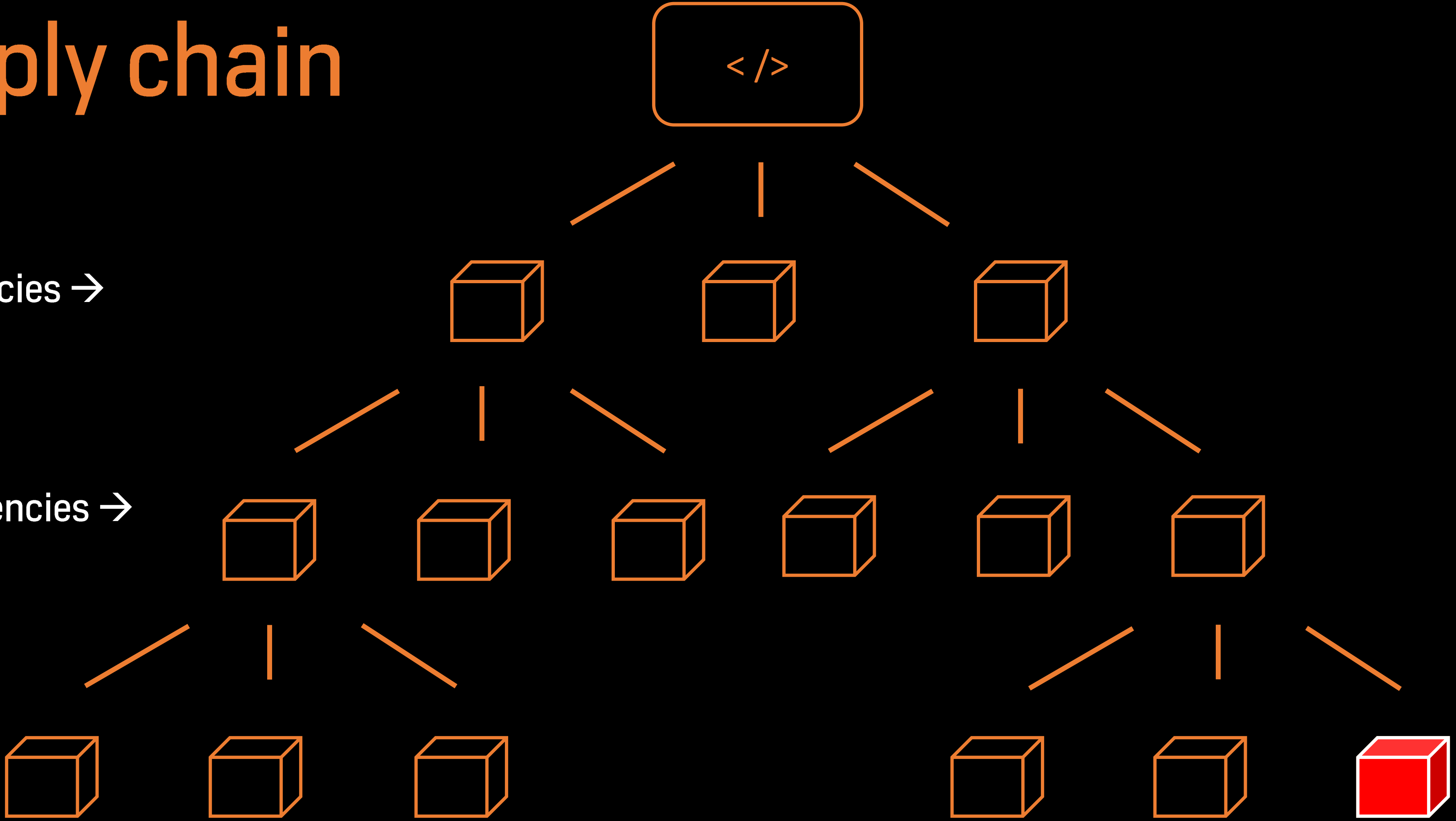
$\Rightarrow$ Build tools

Supply chain

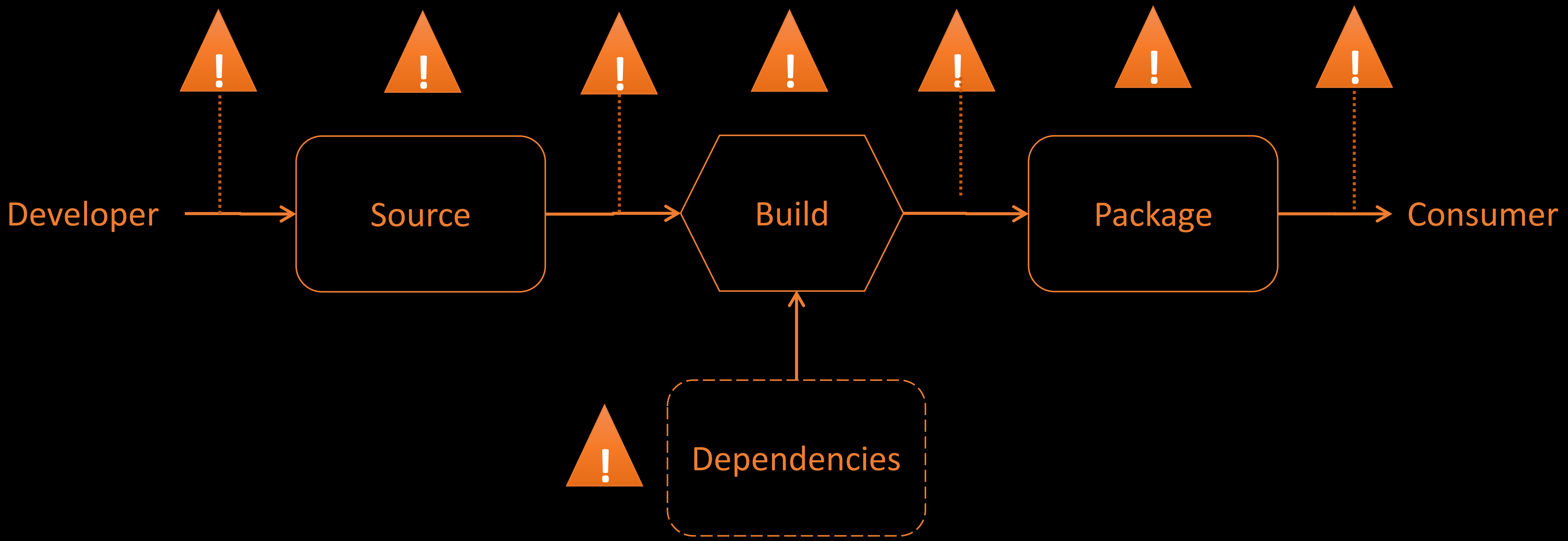
Direct dependencies →

Indirect dependencies →

↘



Attack entries



4 years

On average, vulnerabilities go undetected for four years before being identified.
Sometimes, even longer than that – Log4j was vulnerable for ~7 years

Average remediation time (industry norm)

180 days!

Supply chain: updates

When was the last time you ran an update?

Do you run updates automatically?

Tools:

- Custom scripts against package manager
- GitHub Dependabot
- Mend (prev. WhiteSource) Renovate

Supply Chain Confusion

- Typo squatting
- Namespace shadowing
- Configuration files

Typo squatting 101

```
npm install crossenv
```

Steals all your environment variables

```
npm install cross-env
```

Normal package

Example from 2017, went undetected for 2 weeks

Add Namespaces

```
npm install @babel/helper-regex
```

More specific, less change of squatting

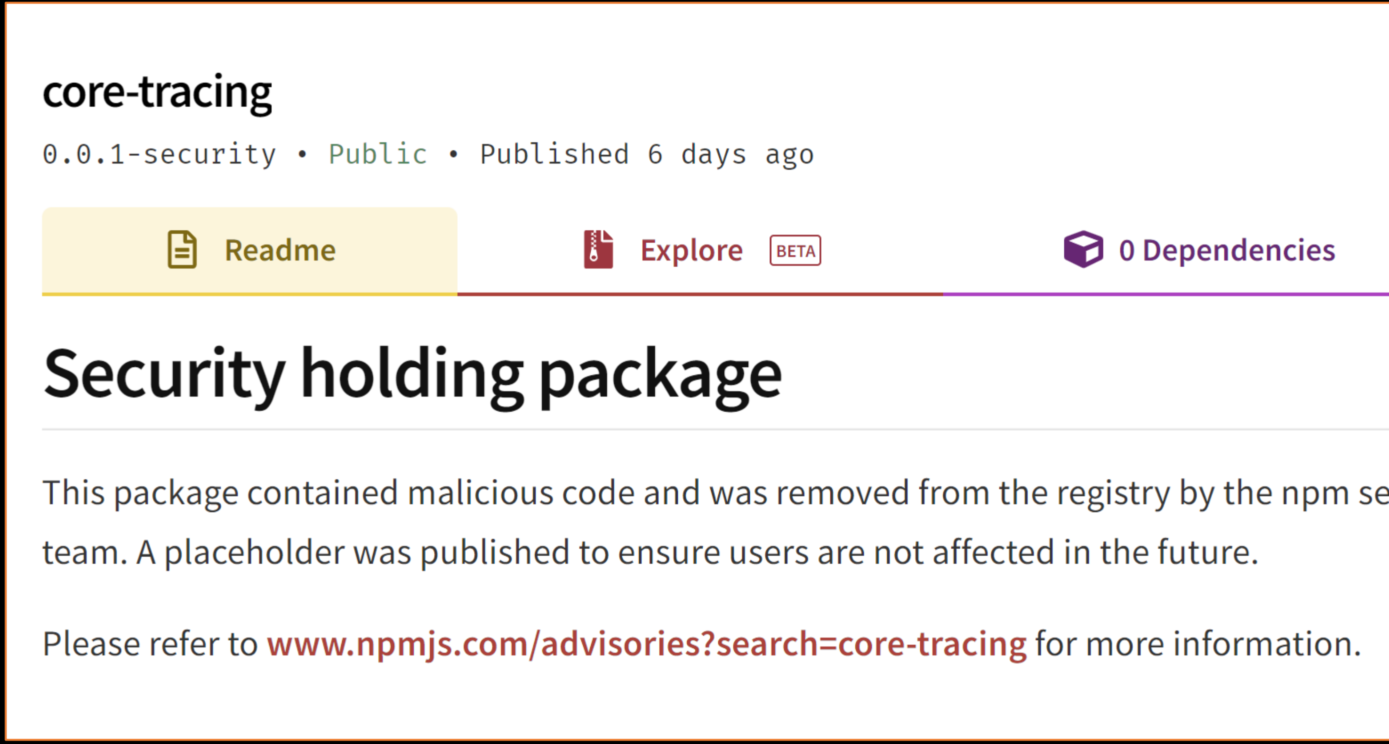
Not all publishers use a namespace

March 2022 – Namespace confusion

```
npm install @azure/core-tracing

vs

npm install core-tracing


→ 218 packages squatted
```

Timelines

- Attacks spread fast, lots of targets
- @azure namespace attack: 50 downloads per package (x 218 packages!)
- Found within 1-2 days
- Blocked by npm after noticing
- Damage is already done by then



core-tracing

0.0.1-security • Public • Published 6 days ago

Security holding package

This package contained malicious code and was removed from the registry by the npm se
team. A placeholder was published to ensure users are not affected in the future.

Please refer to www.npmjs.com/advisories?search=core-tracing for more information.

Protect yourself

- Software Composition Analysis is the starting point

- Package manager scanners:
  - Mend (prev. WhiteSource)
  - BlackDuck
  - JFrog Artifactory + Xray
  - Snyk.io
  - GitHub Dependabot + Vulnerability alerts

Checks package + version against CVE databases

AFTER THE FACT

Protect yourself

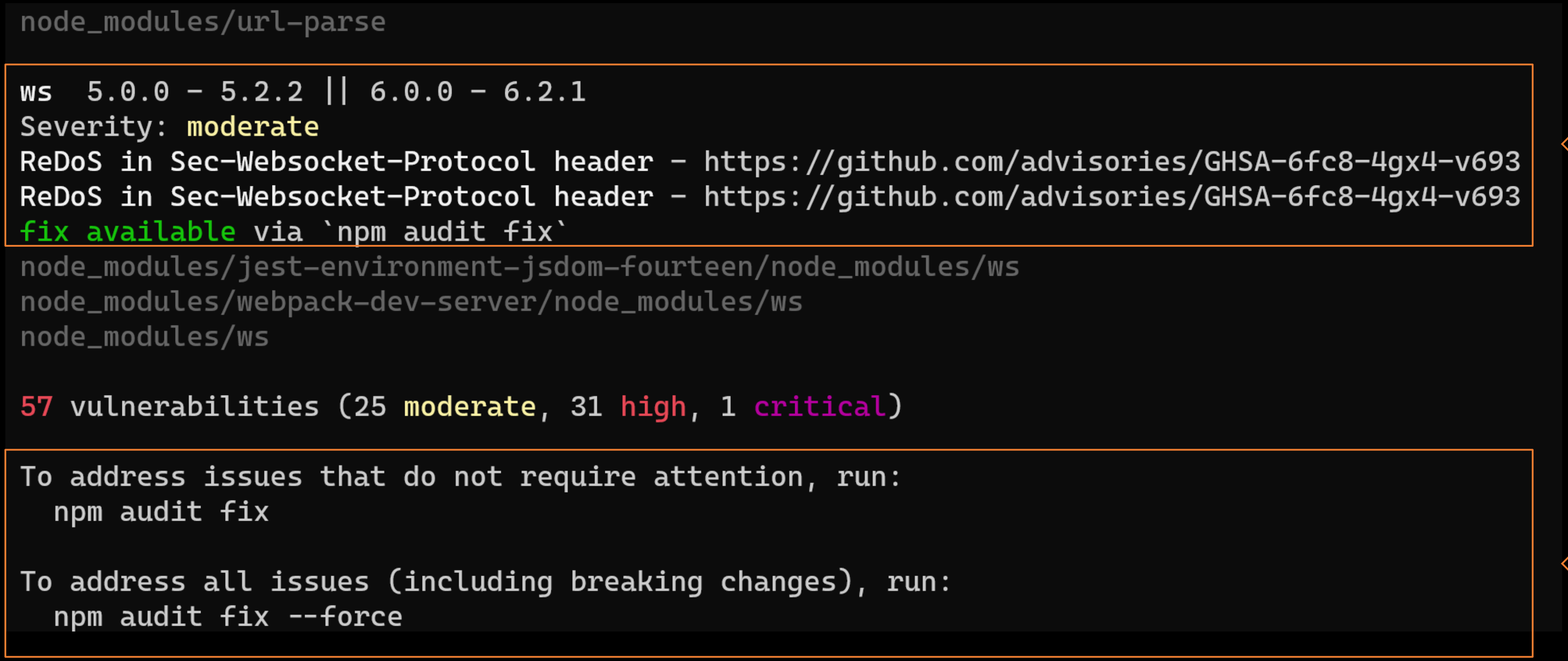
```
npm audit
```

- Gets a list of the dependencies and posts that to: https://registry.npmjs.org/-/npm/v1/security/advisories/bulk

- Might send out private data!
- Any packages without a version field will be ignored!

AFTER THE FACT

npm audit – results

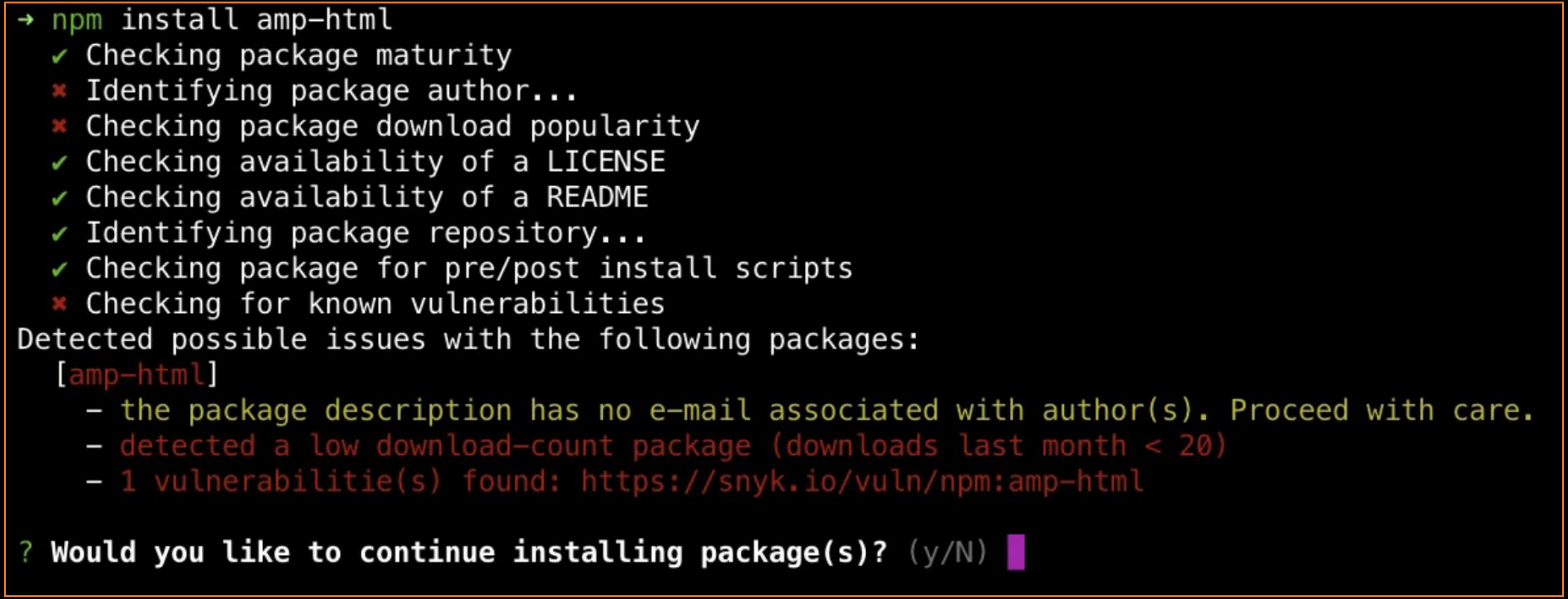


```
node_modules/url-parse

ws  5.0.0 - 5.2.2 || 6.0.0 - 6.2.1
Severity: moderate
ReDoS in Sec-Websocket-Protocol header - https://github.com/advisories/GHSA-6fc8-4gx4-v693
ReDoS in Sec-Websocket-Protocol header - https://github.com/advisories/GHSA-6fc8-4gx4-v693
fix available via `npm audit fix`
node_modules/jest-environment-jsdom-fourteen/node_modules/ws
node_modules/webpack-dev-server/node_modules/ws
node_modules/ws

57 vulnerabilities (25 moderate, 31 high, 1 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force
```

AFTER THE FACT

npm install -g npq

- Wrapper from snyk.io
- Alias to overwrite npm commands



```
→ npm install amp-html
  ✔ Checking package maturity
  ✖ Identifying package author...
  ✖ Checking package download popularity
  ✔ Checking availability of a LICENSE
  ✔ Checking availability of a README
  ✔ Identifying package repository...
  ✔ Checking package for pre/post install scripts
  ✖ Checking for known vulnerabilities
Detected possible issues with the following packages:
  [amp-html]
    - the package description has no e-mail associated with author(s). Proceed with care.
    - detected a low download-count package (downloads last month < 20)
    - 1 vulnerabilitie(s) found: https://snyk.io/vuln/npm:amp-html

? Would you like to continue installing package(s)? (y/N)
```

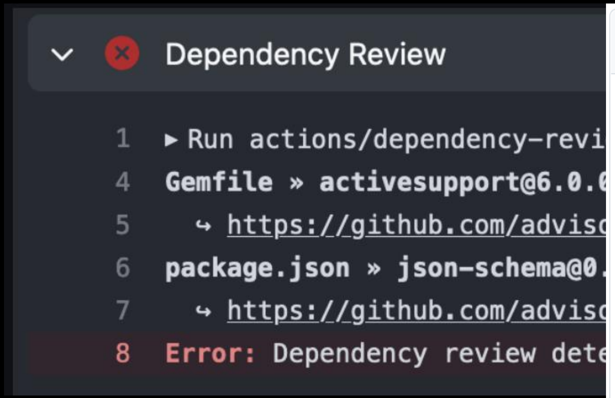
.npmrc – misconfiguration

```
registry=https://registry.npmjs.org/

@myscope:registry=https://mycustomregistry.example.org
```

All your private packages will now get pulled from npmjs.org!

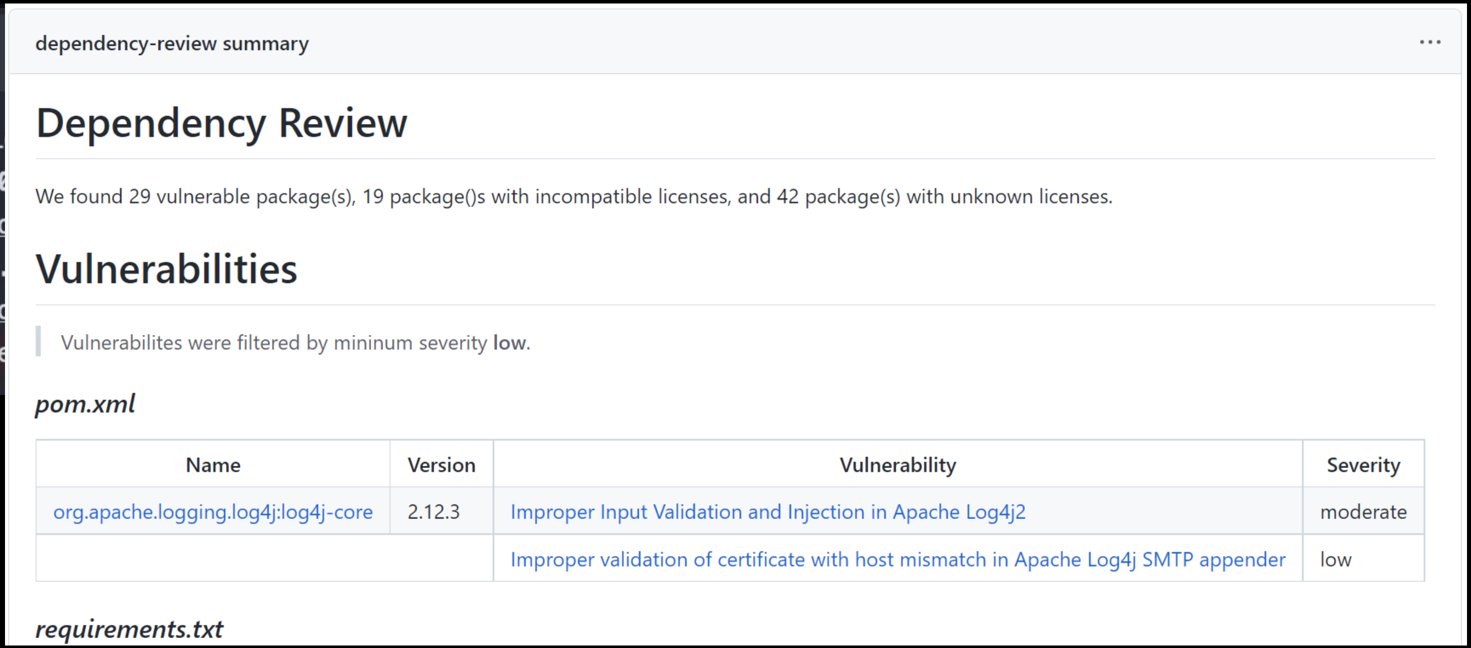
Remediation

- Know what dependencies you have (e.g. with Dependabot)
- Check incoming changes / new dependencies in your Pull Requests
  - use [dependency-review-action]()



Dependency Review

```
▸ Run actions/dependency-revi
Gemfile » activesupport@6.0.0
  ↪ https://github.com/adviso
package.json » json-schema@0.
  ↪ https://github.com/adviso
Error: Dependency review dete
```



dependency-review summary

Dependency Review

We found 29 vulnerable package(s), 19 package()s with incompatible licenses, and 42 package(s) with unknown licenses.

Vulnerabilities

> Vulnerabilites were filtered by mininum severity **low**.

pom.xml

Name	Version	Vulnerability	Severity
org.apache.logging.log4j:log4j-core	2.12.3	Improper Input Validation and Injection in Apache Log4j2	moderate
		Improper validation of certificate with host mismatch in Apache Log4j SMTP appender	low

requirements.txt

Topics

- Why:
  - Supply chain
  - Attack examples
- Protection / Maturity: frameworks
  - OWASP Software Component Verification Standard (SCVS)
  - Supply chain Levels for Software Artifacts (SLSA)

Frameworks

SCVS

Software Component Verification Standard

Version 1.0

OWASP

Software Component Verification Standard

v1 since 2020: https://xpir.it/SCVS

SLSA

Supply chain Levels for Software Artifacts

Currently in Alpha: https://slsa.dev/

Used by Google internally since 2013

OWASP SCVS: Software Component Verification Standard

- Assessment of your software components and how they came to be

Packages/Containers/Source

Pipelines

OWASP SCVS

	L1	L2	L3
V1 – Inventory			
V2 – Software Bill of Materials (SBOM)			
V3 – Build Environment			
V4 - Package Management			
V5 – Component Analysis			
V6 – Pedigree and Provenance			

https://owasp-scvs.gitbook.io

OWASP SCVS – V1 Inventory

All direct and transitive components and their versions are known at completion of a build

Package managers are used to manage all third-party binary components

Software bill of materials continuously maintained and current for all systems

Software bill of materials are required for new procurements

The component type is known throughout inventory

The component function is known throughout inventory

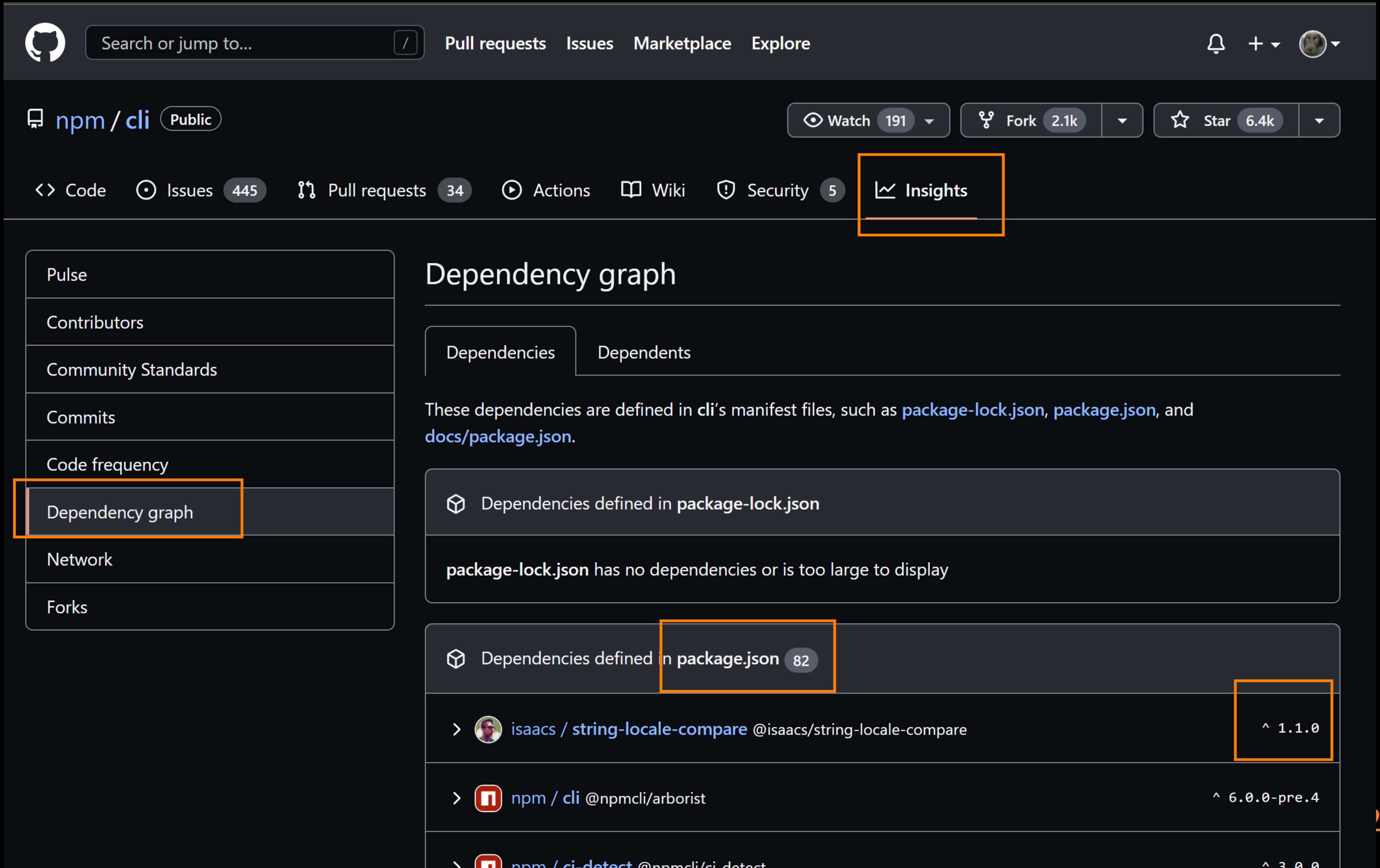
V1 – Inventory

Software Composition Analysis

- GitHub Dependabot
- Black Duck
- Mend (WhiteSource)
- Snyk
- Jfrog Xray

Dependabot Example: https://github.com/npm/cli/network/dependencies

Demo: npm/cli – Dependencies



Search or jump to... / Pull requests Issues Marketplace Explore

npm / cli Public — Watch 191 · Fork 2.1k · Star 6.4k

Code · Issues 445 · Pull requests 34 · Actions · Wiki · Security 5 · Insights

- Pulse
- Contributors
- Community Standards
- Commits
- Code frequency
- Dependency graph
- Network
- Forks

Dependency graph

Dependencies | Dependents

These dependencies are defined in **cli**'s manifest files, such as package-lock.json, package.json, and docs/package.json.

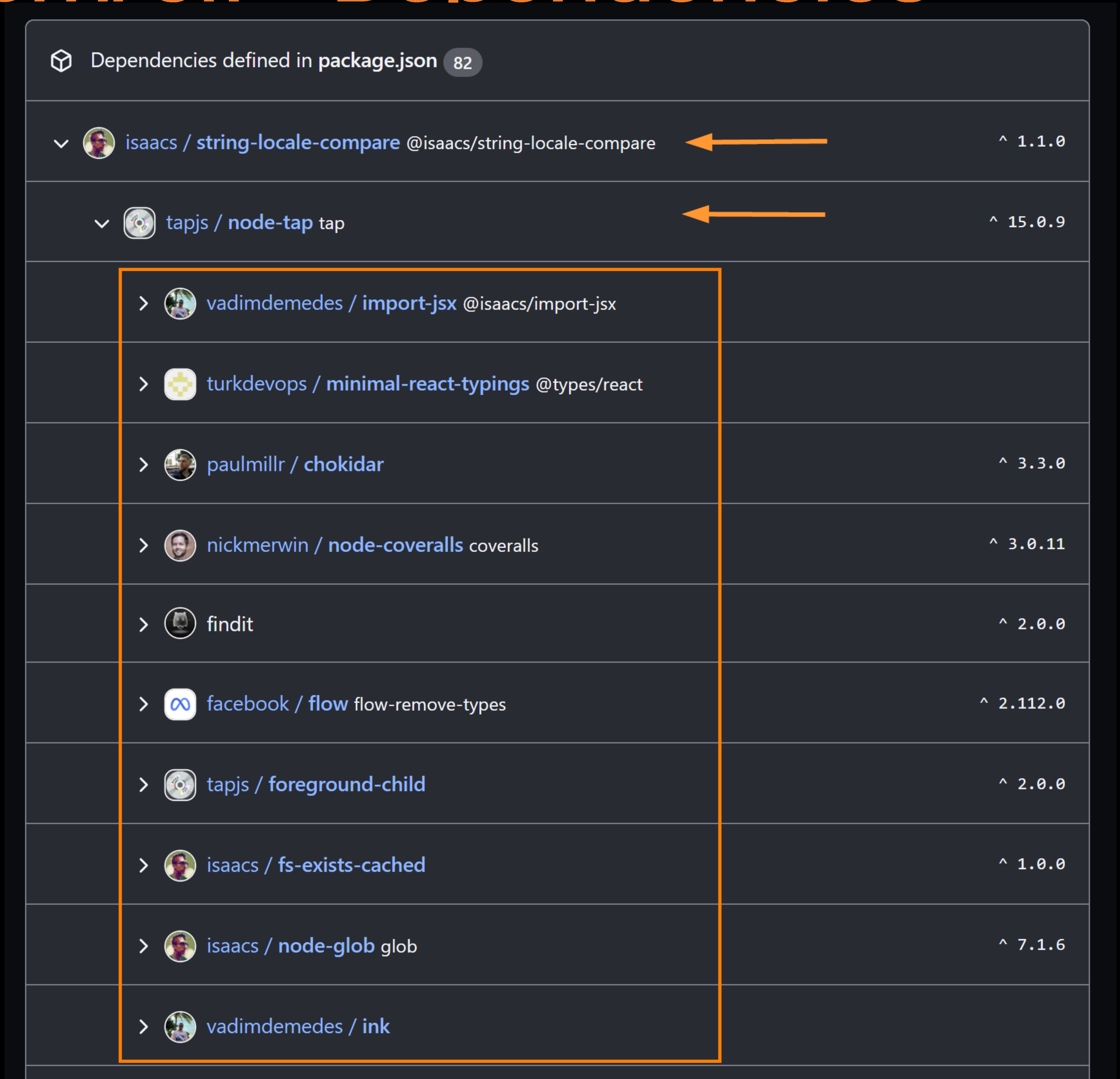
Dependencies defined in **package-lock.json**

package-lock.json has no dependencies or is too large to display

Dependencies defined in **package.json** 82

isaacs / **string-locale-compare** @isaacs/string-locale-compare	^ 1.1.0
npm / cli @npmcli/arborist	^ 6.0.0-pre.4
npm / ci-detect @npmcli/ci-detect	^ 3.0.0

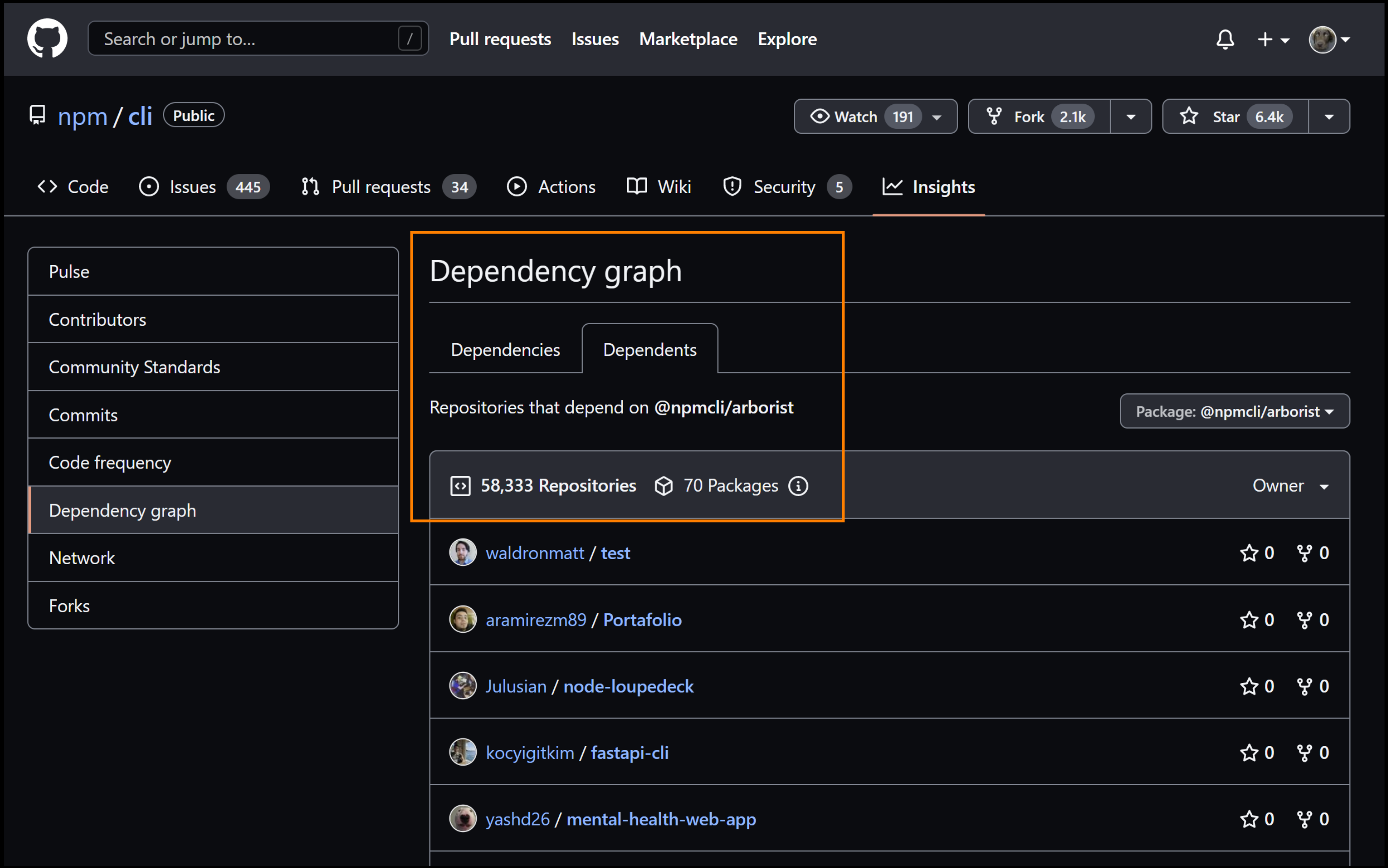
Demo: npm/cli – Dependencies



Dependencies defined in **package.json** 82

isaacs / **string-locale-compare** @isaacs/string-locale-compare	^ 1.1.0
tapjs / **node-tap** tap	^ 15.0.9
vadimdemedes / **import-jsx** @isaacs/import-jsx	
turkdevops / **minimal-react-typings** @types/react	
paulmillr / **chokidar**	^ 3.3.0
nickmerwin / **node-coveralls** coveralls	^ 3.0.11
findit	^ 2.0.0
facebook / **flow** flow-remove-types	^ 2.112.0
tapjs / **foreground-child**	^ 2.0.0
isaacs / **fs-exists-cached**	^ 1.0.0
isaacs / **node-glob** glob	^ 7.1.6
vadimdemedes / **ink**	

Demo: npm/cli – Dependents



npm / cli Public

Watch 191 · Fork 2.1k · Star 6.4k

Code · Issues 445 · Pull requests 34 · Actions · Wiki · Security 5 · Insights

Pulse
Contributors
Community Standards
Commits
Code frequency
Dependency graph
Network
Forks

Dependency graph

Dependencies | Dependents

Repositories that depend on **@npmcli/arborist**

Package: @npmcli/arborist

58,333 Repositories · 70 Packages · Owner

waldronmatt / **test**	☆ 0	0
aramirezm89 / **Portafolio**	☆ 0	0
Julusian / **node-loupedeck**	☆ 0	0
kocyigitkim / **fastapi-cli**	☆ 0	0
yashd26 / **mental-health-web-app**	☆ 0	0

@robbos81

OWASP SCVS – V2 Software Bill of Materials

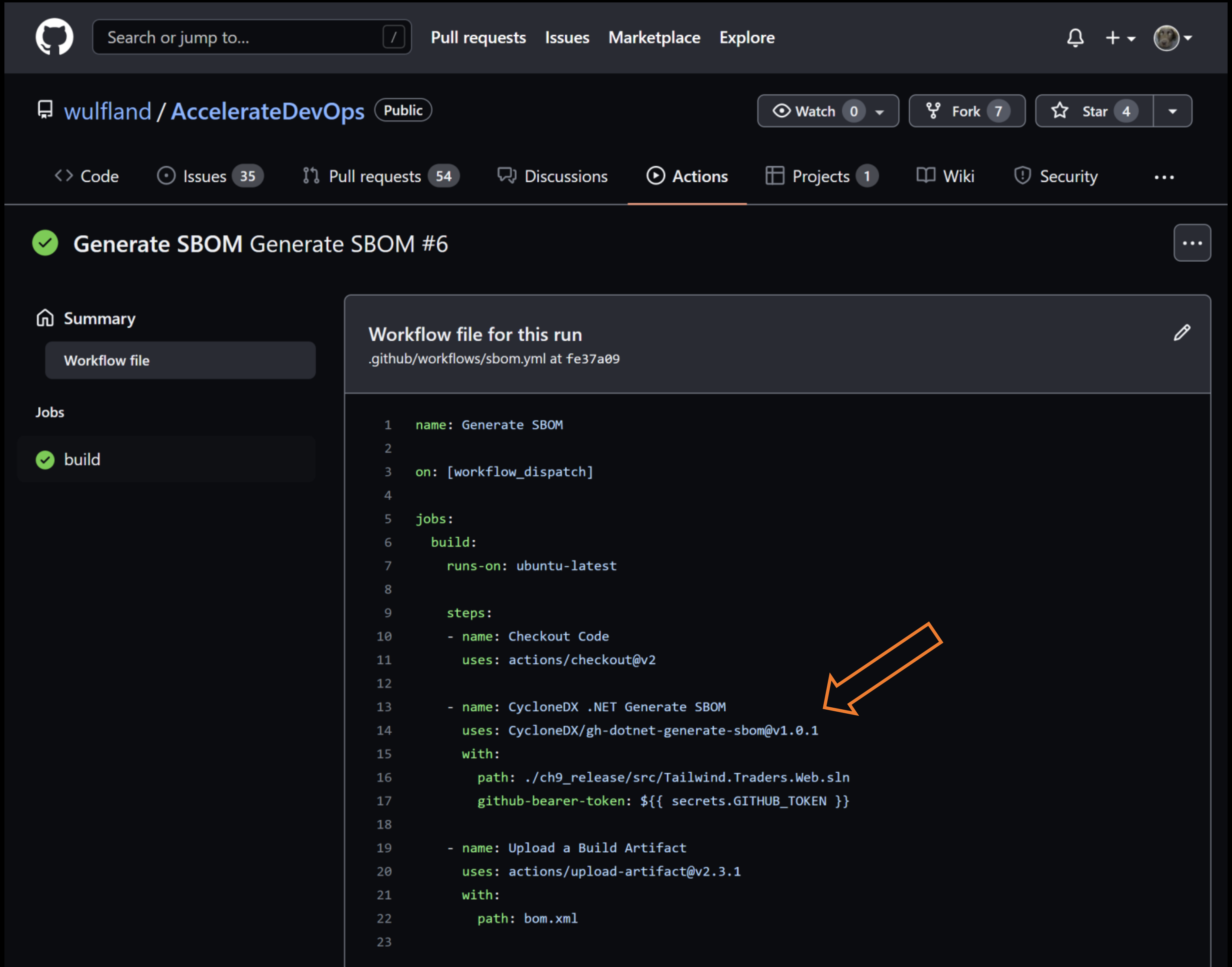
- SBOM creation is automated and reproducible
- SBOM has been signed by publisher, supplier, or certifying authority
- SBOM signature verification exists and is performed
- SBOM is analyzed for risk

V2 Software Bill of Materials

- Multiple standards for SBOM formats:
  - SPDX (Software Package Data Exchange) – Linux Foundation
    - Focusses on license information
    - ISO Standard
  - CycloneDX – OWASP
    - Focusses on vulnerabilities and security
- Example SBOM creation with GitHub Actions & CycloneDX:

  https://github.com/wulfland/AccelerateDevOps/actions/runs/3176320773

V2 Software Bill of Materials – demo

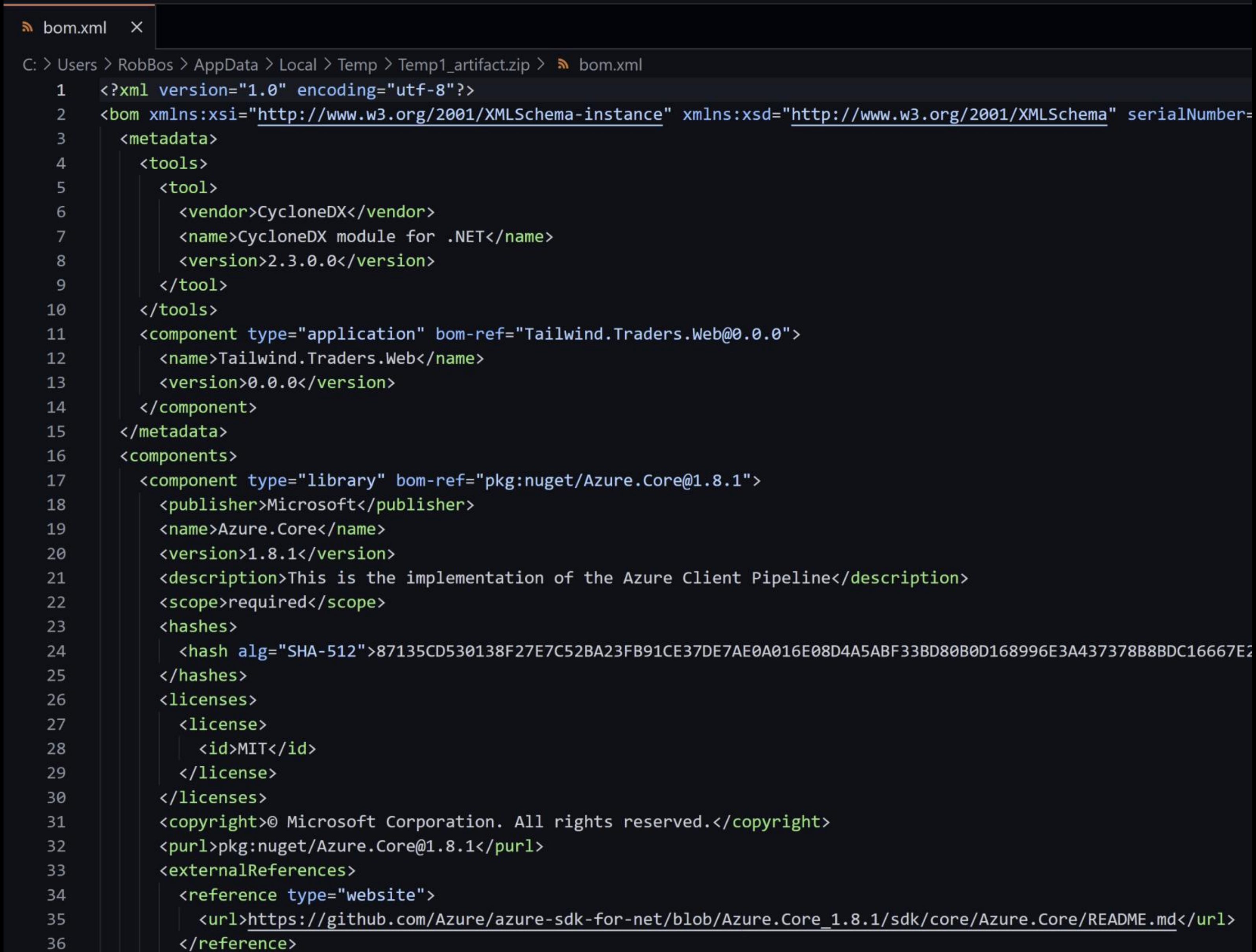


Generate SBOM Generate SBOM #6

Workflow file for this run
.github/workflows/sbom.yml at fe37a09

```yaml
name: Generate SBOM

on: [workflow_dispatch]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout Code
      uses: actions/checkout@v2

    - name: CycloneDX .NET Generate SBOM
      uses: CycloneDX/gh-dotnet-generate-sbom@v1.0.1
      with:
        path: ./ch9_release/src/Tailwind.Traders.Web.sln
        github-bearer-token: ${{ secrets.GITHUB_TOKEN }}

    - name: Upload a Build Artifact
      uses: actions/upload-artifact@v2.3.1
      with:
        path: bom.xml

```

V2 Software Bill of Materials – demo



bom.xml

C: > Users > RobBos > AppData > Local > Temp > Temp1_artifact.zip > bom.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<bom xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" serialNumber=
  <metadata>
    <tools>
      <tool>
        <vendor>CycloneDX</vendor>
        <name>CycloneDX module for .NET</name>
        <version>2.3.0.0</version>
      </tool>
    </tools>
    <component type="application" bom-ref="Tailwind.Traders.Web@0.0.0">
      <name>Tailwind.Traders.Web</name>
      <version>0.0.0</version>
    </component>
  </metadata>
  <components>
    <component type="library" bom-ref="pkg:nuget/Azure.Core@1.8.1">
      <publisher>Microsoft</publisher>
      <name>Azure.Core</name>
      <version>1.8.1</version>
      <description>This is the implementation of the Azure Client Pipeline</description>
      <scope>required</scope>
      <hashes>
        <hash alg="SHA-512">87135CD530138F27E7C52BA23FB91CE37DE7AE0A016E08D4A5ABF33BD80B0D168996E3A437378B8BDC16667E2
      </hashes>
      <licenses>
        <license>
          <id>MIT</id>
        </license>
      </licenses>
      <copyright>© Microsoft Corporation. All rights reserved.</copyright>
      <purl>pkg:nuget/Azure.Core@1.8.1</purl>
      <externalReferences>
        <reference type="website">
          <url>https://github.com/Azure/azure-sdk-for-net/blob/Azure.Core_1.8.1/sdk/core/Azure.Core/README.md</url>
        </reference>
```

OWASP SCVS V3 – Build Environment

- Application build pipeline may only perform builds of source code maintained in version control systems
- Application build pipeline prohibits alteration of DNS and network settings during build
- Application build pipeline prohibits alteration of certificate trust stores
- Application build pipeline enforces authentication and defaults to deny
- Application build pipeline enforces authorization and defaults to deny
- Application build pipeline requires separation of concerns for the modification of system settings
- Application build pipeline maintains a verifiable audit log of all system changes
- Application build pipeline maintains a verifiable audit log of all build job changes
- Application build pipeline has required maintenance cadence where the entire stack is updated, patched, and re-certified for use
- Compilers, version control clients, development utilities, and software development kits are analyzed and monitored for tampering, trojans, or malicious code
- All build-time manipulations to source or binaries are known and well defined
- Checksums of all first-party and third-party components are documented for every build
- Checksums of all components are accessible and delivered out-of-band whenever those components are packaged or distributed
- Unused direct and transitive components have been identified
- Unused direct and transitive components have been removed from the application

https://owasp-scvs.gitbook.io

OWASP SCVS V3 – Build Environment

- Application build pipeline maintains a verifiable audit log of:
  - build job changes
  - system changes

- The entire build stack is updated, patched, and re-certified for use

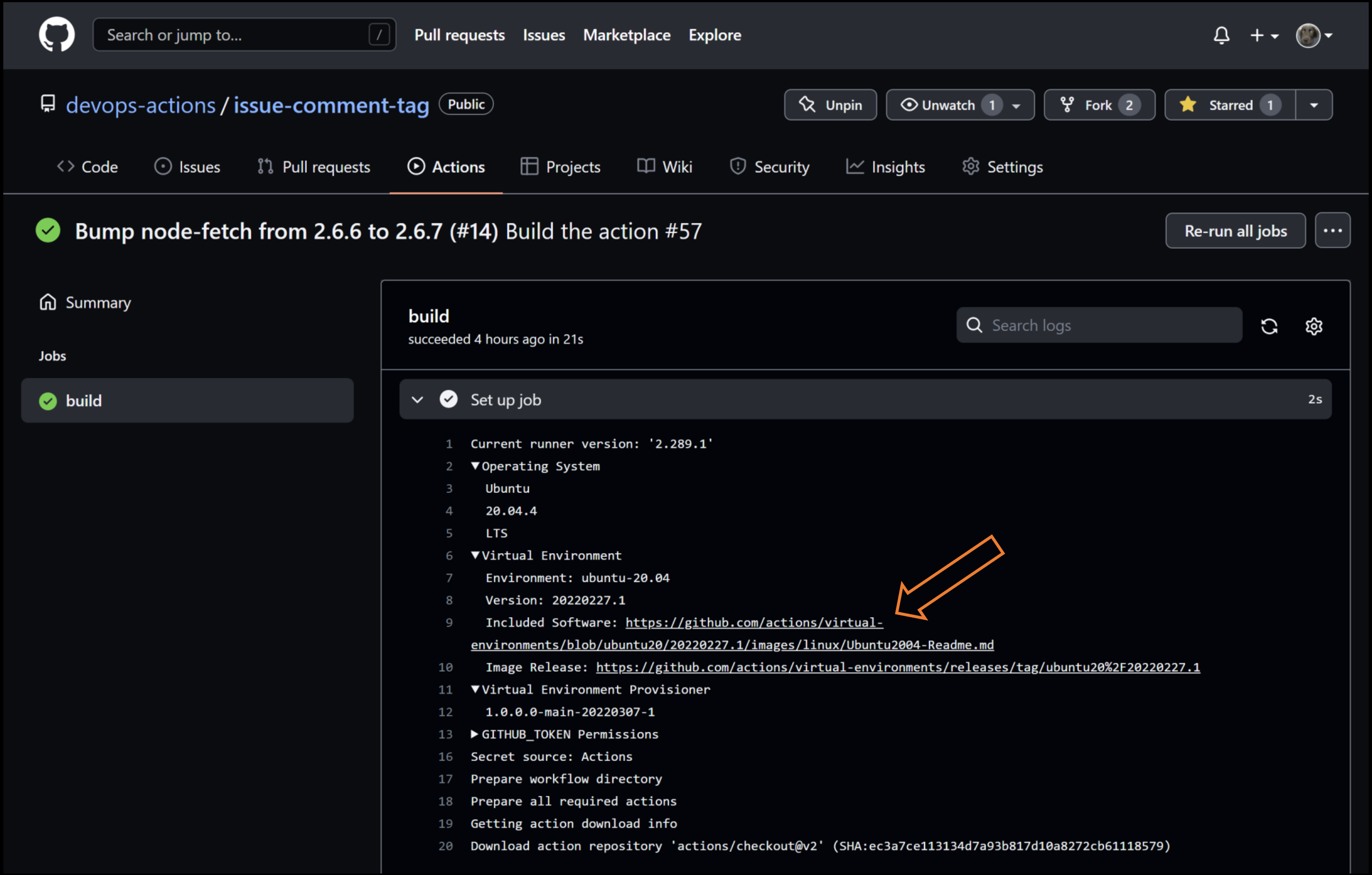
- Everything is analyzed and monitored for tampering, trojans, or malicious code
  - Compilers
  - Version control clients
  - Development utilities
  - Software development kits

V3 – Build Environment

- Application build pipeline maintains a verifiable audit log of
  - build job changes
  - system changes

- Example in GitHub Actions: end-to-end traceability:
  - Workflow changes in commits
  - Execution environment in the logs
    - https://github.com/devops-actions/issue-comment-tag/actions

V3 – Build Environment – demo



Bump node-fetch from 2.6.6 to 2.6.7 (#14) Build the action #57

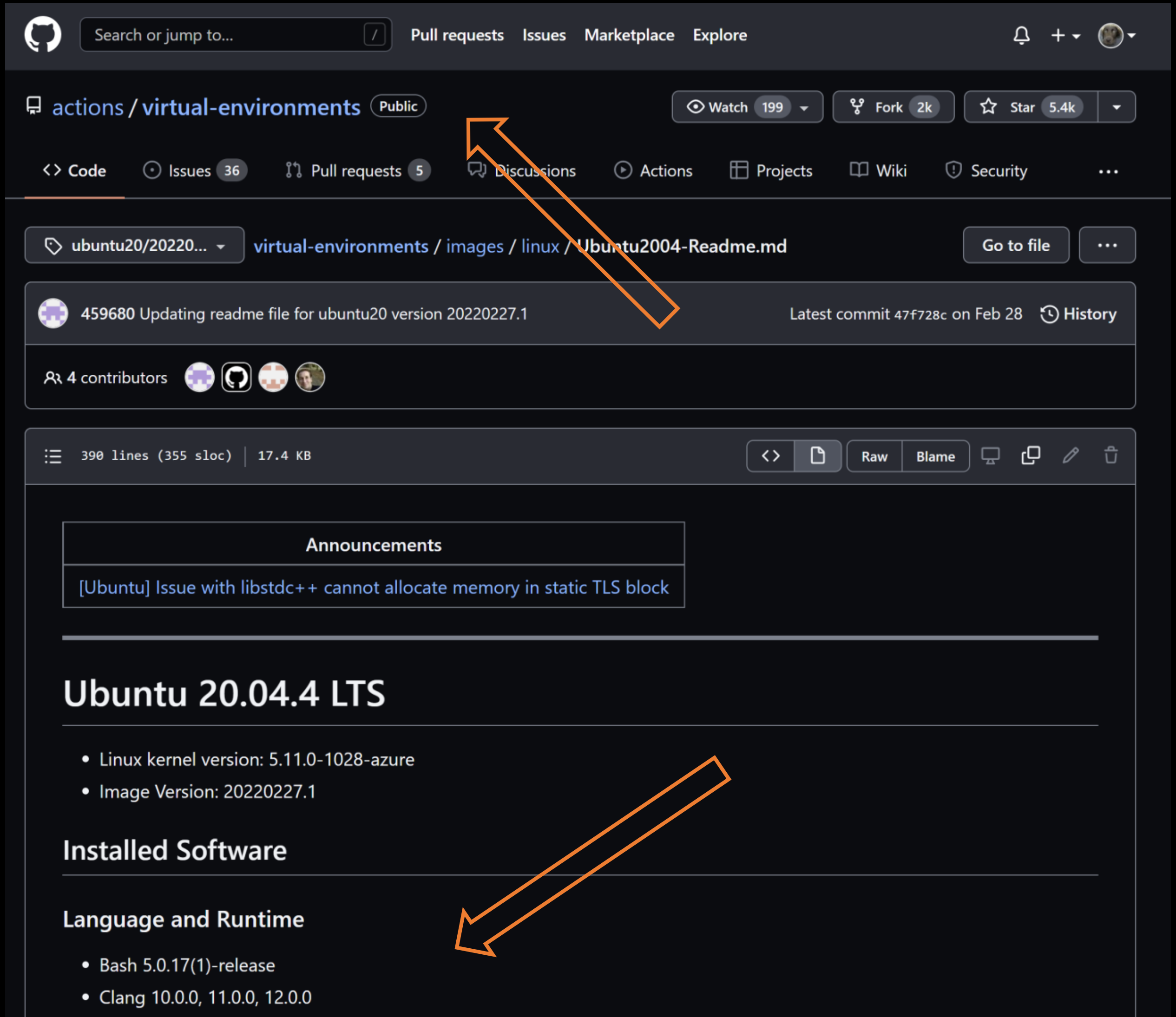
build
succeeded 4 hours ago in 21s

Set up job 2s

```
Current runner version: '2.289.1'
▼Operating System
  Ubuntu
  20.04.4
  LTS
▼Virtual Environment
  Environment: ubuntu-20.04
  Version: 20220227.1
  Included Software: https://github.com/actions/virtual-environments/blob/ubuntu20/20220227.1/images/linux/Ubuntu2004-Readme.md
  Image Release: https://github.com/actions/virtual-environments/releases/tag/ubuntu20%2F20220227.1
▼Virtual Environment Provisioner
  1.0.0.0-main-20220307-1
▶GITHUB_TOKEN Permissions
Secret source: Actions
Prepare workflow directory
Prepare all required actions
Getting action download info
Download action repository 'actions/checkout@v2' (SHA:ec3a7ce113134d7a93b817d10a8272cb61118579)
```

V3 – Build Environment – demo

https://github.com/actions/virtual-environments/blob/ubuntu20/20220227.1/images/linux/Ubuntu2004-Readme.md



virtual-environments / images / linux / Ubuntu2004-Readme.md

459680 Updating readme file for ubuntu20 version 20220227.1 — Latest commit 47f728c on Feb 28

4 contributors

390 lines (355 sloc) | 17.4 KB

Announcements
[Ubuntu] Issue with libstdc++ cannot allocate memory in static TLS block

Ubuntu 20.04.4 LTS

- Linux kernel version: 5.11.0-1028-azure
- Image Version: 20220227.1

Installed Software

Language and Runtime

- Bash 5.0.17(1)-release
- Clang 10.0.0, 11.0.0, 12.0.0

OWASP SCVS V4 – Package management

- Package repository components have been published with multi-factor authentication
- Package repository notifies publishers & users of security issues
- Package repository requires code signing to publish packages to production repositories
- Package manager verifies the integrity of packages when they are retrieved:
  - From remote repository
  - From file system

OWASP SCVS V4 – Package management

- Package repository components have been published with multi-factor authentication – npm improvements with 2FA

- Package repository notifies publishers & users of security issues
  - Dependabot does this

- Package repository requires code signing to publish packages to production repositories
  - Most package managers do not have support
  - npm focusses on 2FA
  - NuGet, Maven / Gradle / Ant (uploads through Maven central), RubyGems

OWASP SCVS V5 – Component Analysis

- Component is analyzed using linters and/or static analysis tools
- Linting and/or static analysis is performed <u>with every upgrade</u>
- An automated process for:
  - Identifying all publicly disclosed vulnerabilities is used
  - Identifying confirmed dataflow exploitability is used
  - For identifying end-of-life / end-of-support components is used

V5 – Component Analysis

- Component is analyzed using linters and/or static analysis tools
  - Trigger workflow on push
- Linting and/or static analysis is performed with every upgrade
  - Branch protection rules

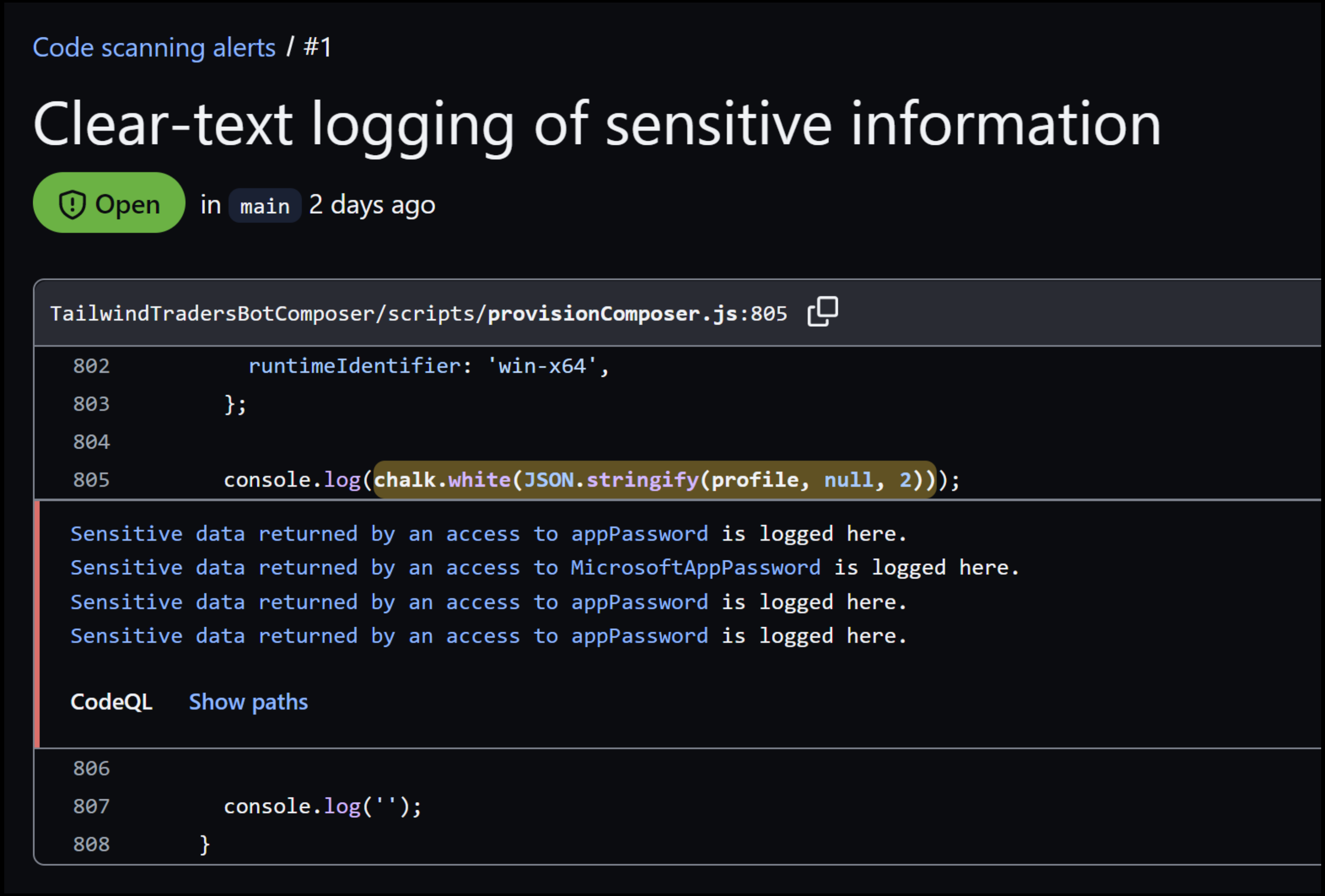
- An automated process for:
  - Identifying all publicly disclosed vulnerabilities is used: Dependabot
  - Identifying confirmed dataflow exploitability is used: SAST (CodeQL) / DAST
  - For identifying end-of-life / end-of-support components is used: ?

CodeQL

Code Query Language: Database + Queries

Demo: Code scanning alerts · github.com

CodeQL Demo



Code scanning alerts / #1

Clear-text logging of sensitive information

Open in main 2 days ago

TailwindTradersBotComposer/scripts/provisionComposer.js:805

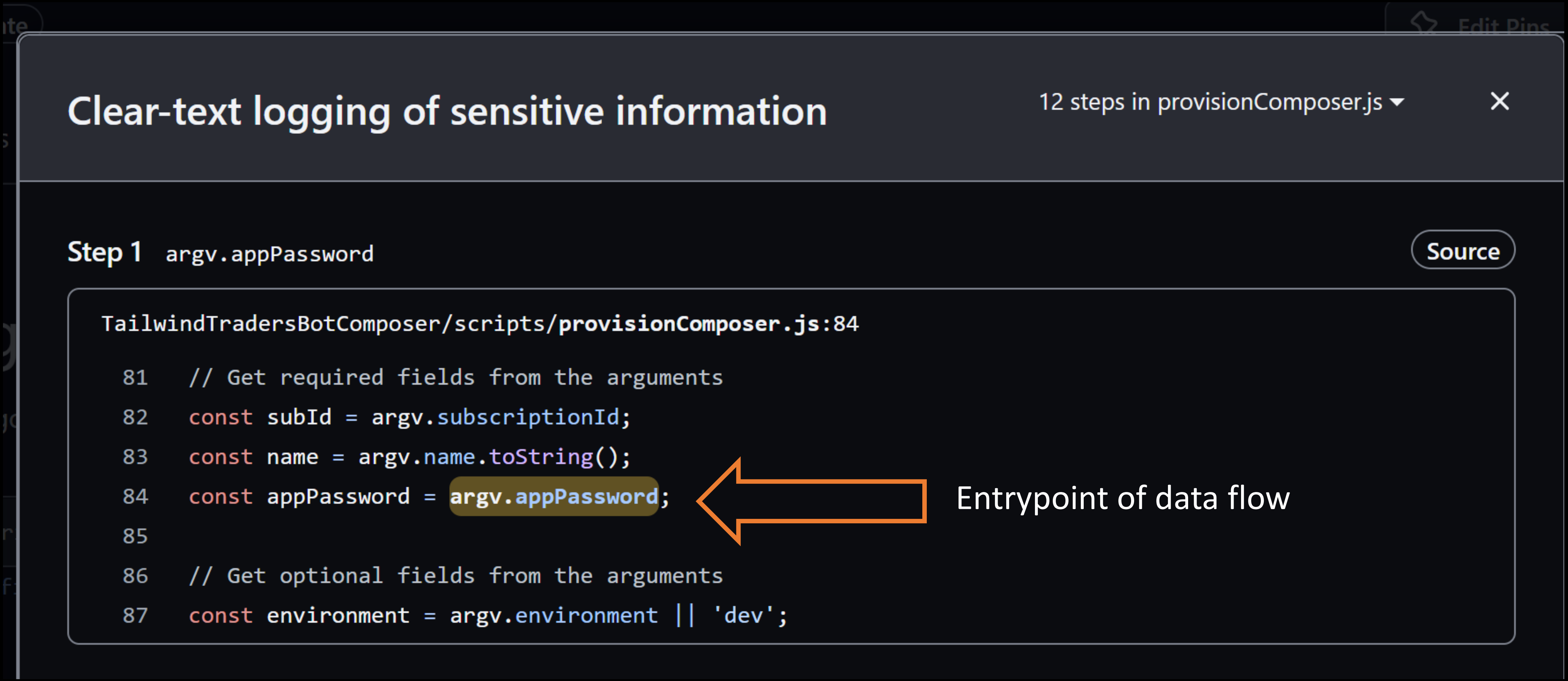
```
            runtimeIdentifier: 'win-x64',
          };

          console.log(chalk.white(JSON.stringify(profile, null, 2)));
```

```
Sensitive data returned by an access to appPassword is logged here.
Sensitive data returned by an access to MicrosoftAppPassword is logged here.
Sensitive data returned by an access to appPassword is logged here.
Sensitive data returned by an access to appPassword is logged here.
```

CodeQL Show paths

```

          console.log('');
        }
```

CodeQL demo


Clear-text logging of sensitive information
12 steps in provisionComposer.js
Step 1 argv.appPassword
Source

```
TailwindTradersBotComposer/scripts/provisionComposer.js:84

// Get required fields from the arguments
const subId = argv.subscriptionId;
const name = argv.name.toString();
const appPassword = argv.appPassword;

// Get optional fields from the arguments
const environment = argv.environment || 'dev';
```

Entrypoint of data flow

OWASP SCVS V6 – Pedigree and Provenance

- Point of origin is verifiable for components
- Chain of custody is auditable for components

V6 – Pedigree and Provenance

- Point of origin is verifiable for components
  - Dependabot dependency graph
- Chain of custody is auditable for components
  - Where did the component came from:
    - Pipeline link
    - Commit history

Example: https://github.com/npm/cli/releases

V6 – Pedigree and Provenance

- Point of origin is verifiable for components
  - Cosign + sigstore = verification options for containers

- Chain of custody is auditable for components
  - SLSA is doing this for other binaries

OWASP SCVS

	L1	L2	L3
V1 – Inventory			
V2 – Software Bill of Materials (SBOM)			
V3 – Build Environment			
V4 - Package Management			
V5 – Component Analysis			
V6 – Pedigree and Provenance			

https://owasp-scvs.gitbook.io

WELCOME TO Fabulous TECHORAMA UTRECHT

Protect yourself against supply chain attacks

Rob Bos
DevOps Consultant – Xpirit
The Netherlands
https://devopsjournal.io

@robbos81

https://myoctocat.com