

Reviewing 3rd Party Libraries security using scorecards

Niels Tanis

Sr. Principal Security Researcher

VERACODE



CYBER SECURITY
COALITION.be



SecAppDev

Who am I?



- Niels Tanis
- Sr. Principal Security Researcher
 - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
 - Research on static analysis for .NET apps
 - Enjoying Rust!
- Microsoft MVP – Developer Technologies

VERACODE

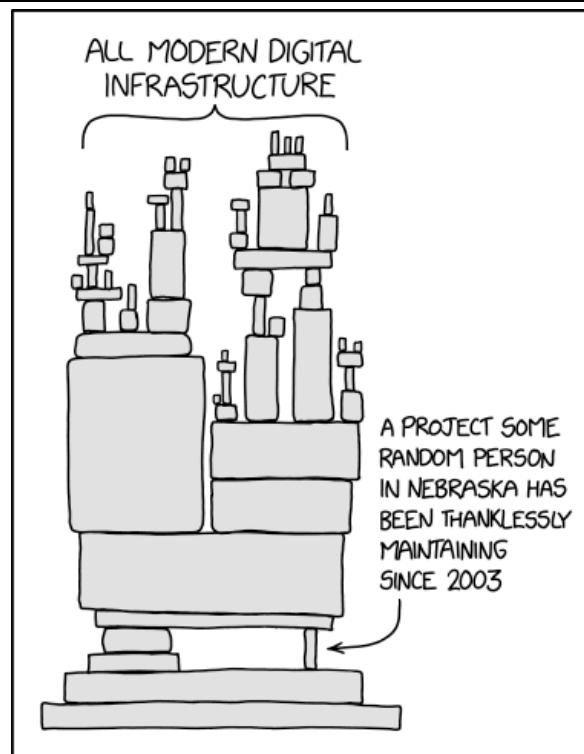


@niels.fennec.dev



@nielstanis@infosec.exchange

Modern Application Architecture XKCD 2347



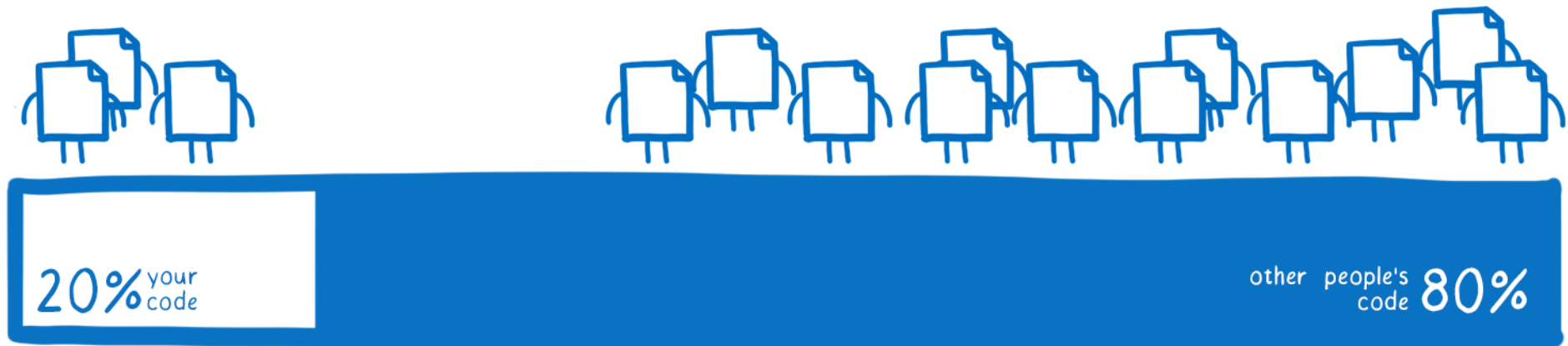
Agenda



- Risks in 3rd party NuGet Packages
- OpenSFF Scorecard
- Measure, New & Improved
- Conclusion - Q&A



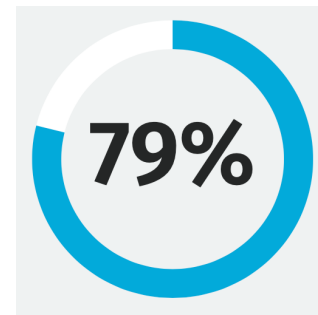
Average codebase composition



State of Software Security v11



"Despite this dynamic landscape, 79 percent of the time, developers never update third-party libraries after including them in a codebase."



 @niels.fennec.dev  @nielstanis@infosec.exchange

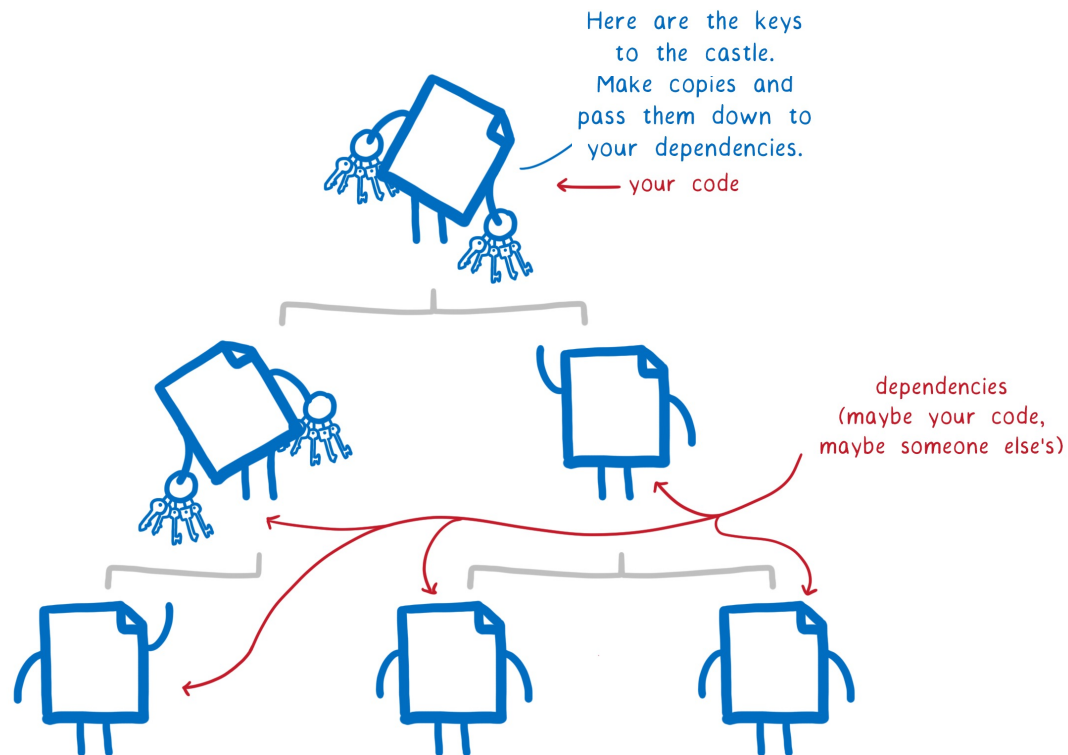
State of Log4j - 2 years later



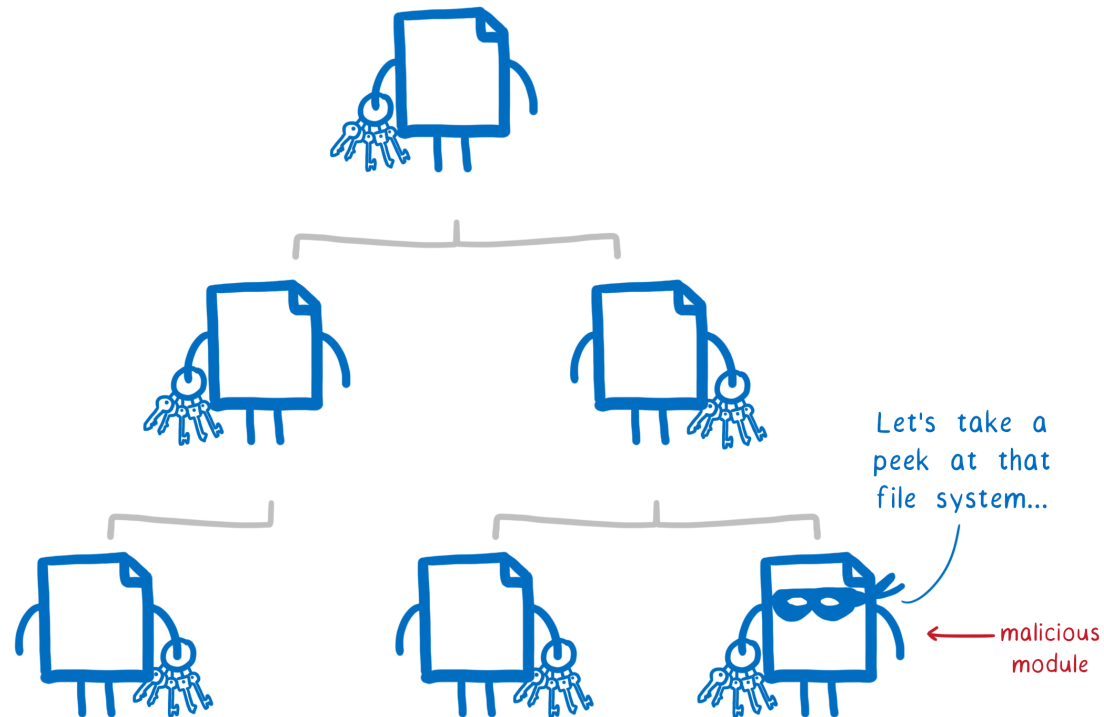
- Analysed our data August-November 2023
 - Total set of almost 39K unique applications scanned
- 2.8% run version vulnerable to Log4Shell
- 3.8% run version patched but vulnerable to other CVE
- 32% rely on a version that's end-of-life and have no support for any patches.



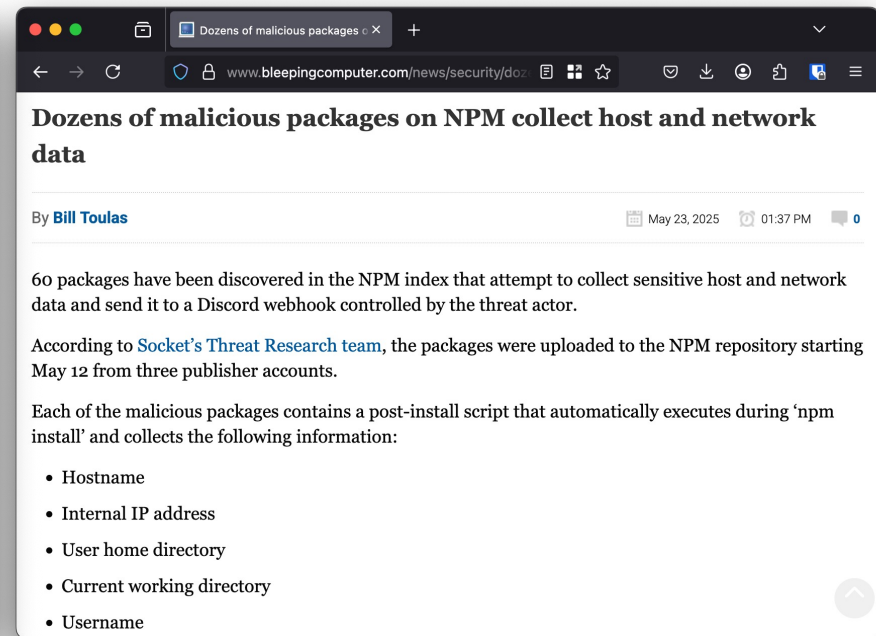
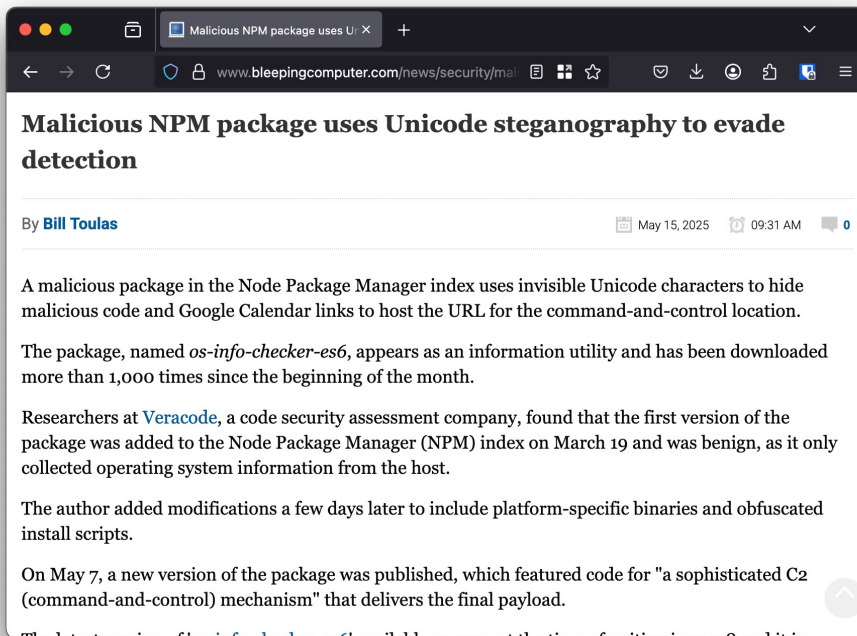
Average codebase composition



Malicious Assembly

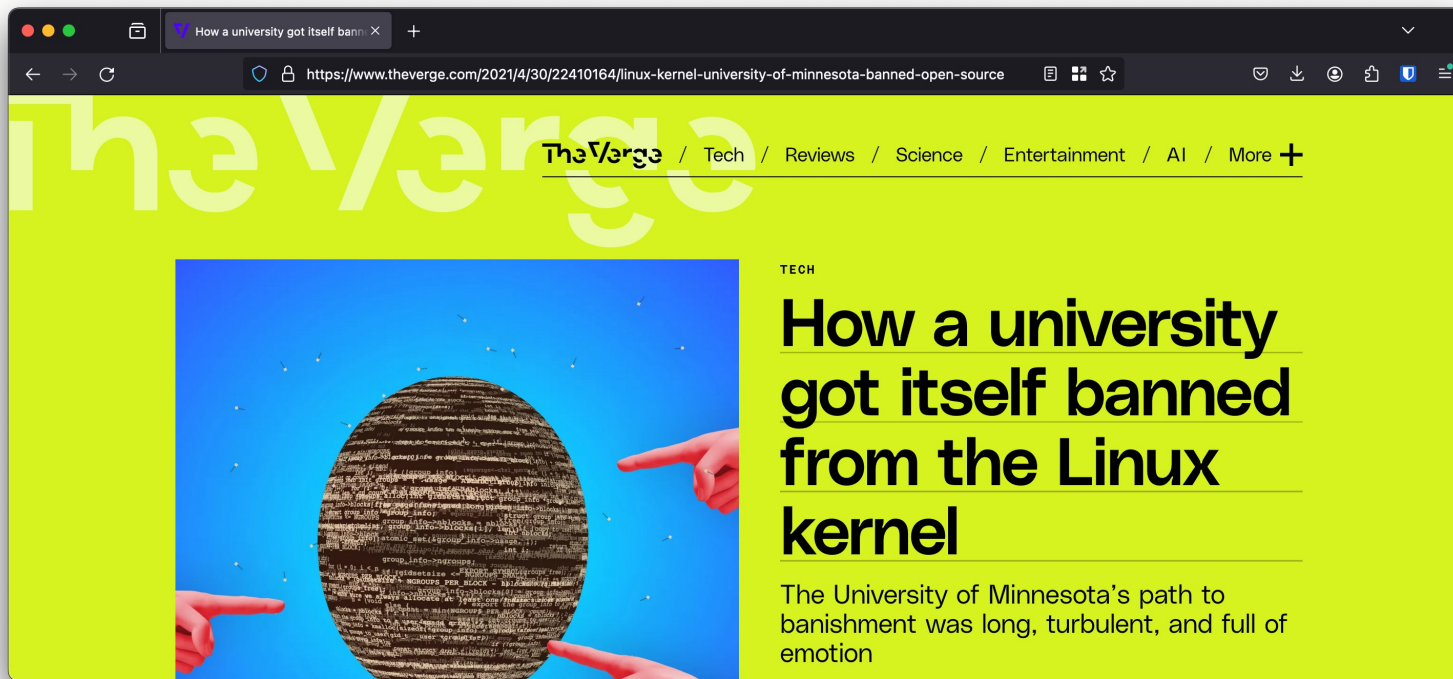


Malicious Package



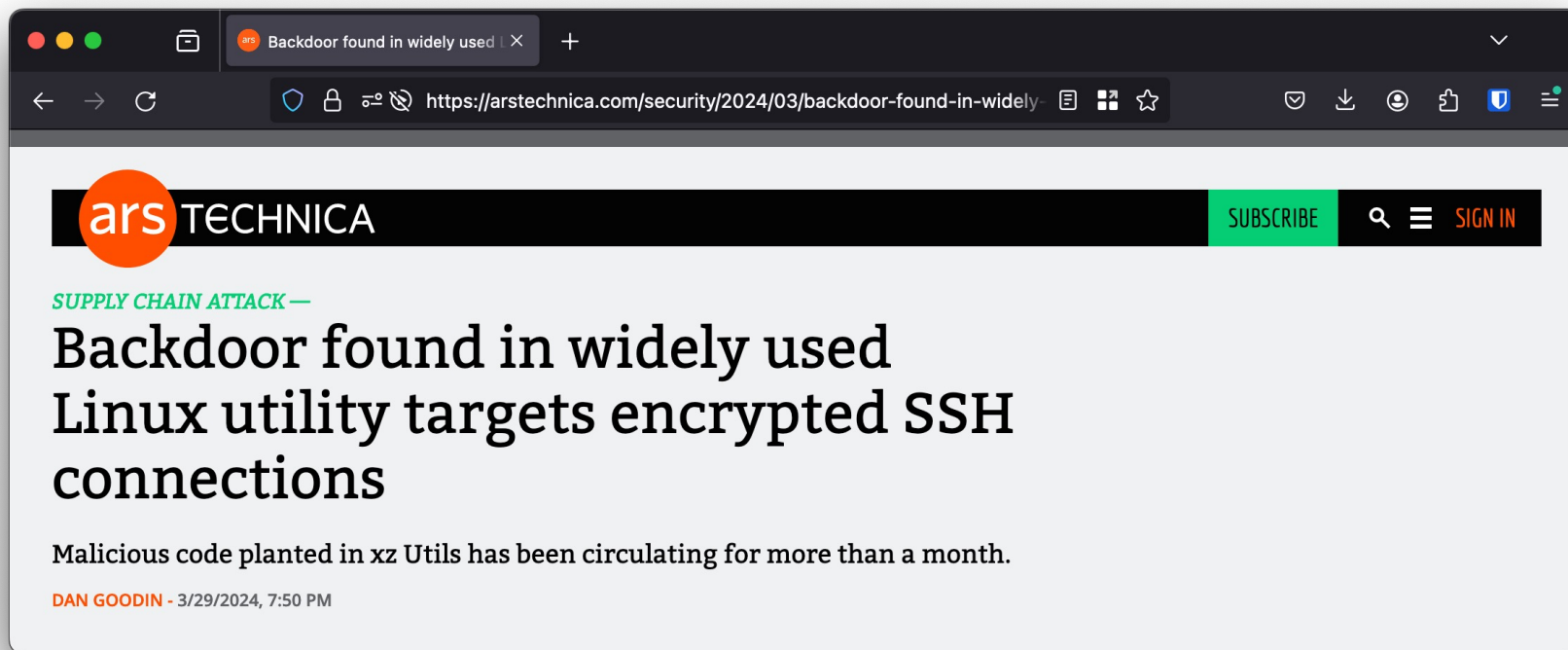
 [@niels.fennec.dev](#)  [@nielstanis@infosec.exchange](#)

Hypocrite Commits



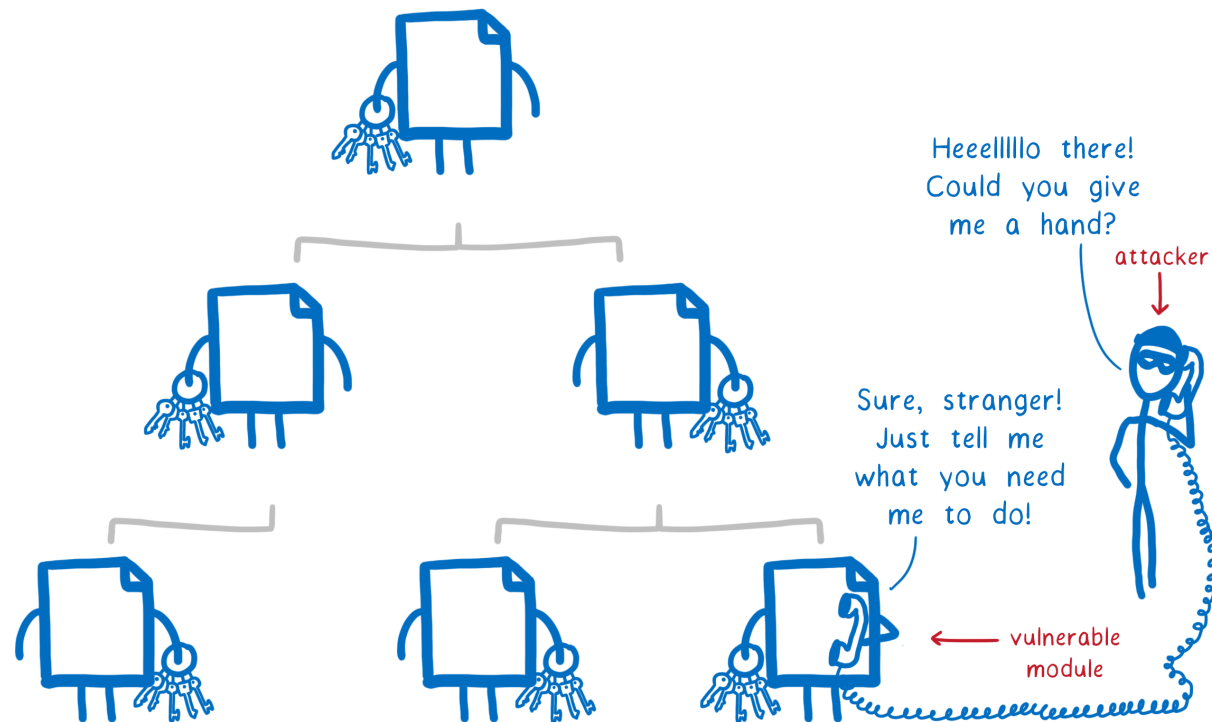
 @niels.fennec.dev  @nielstanis@infosec.exchange

XZ Backdoor

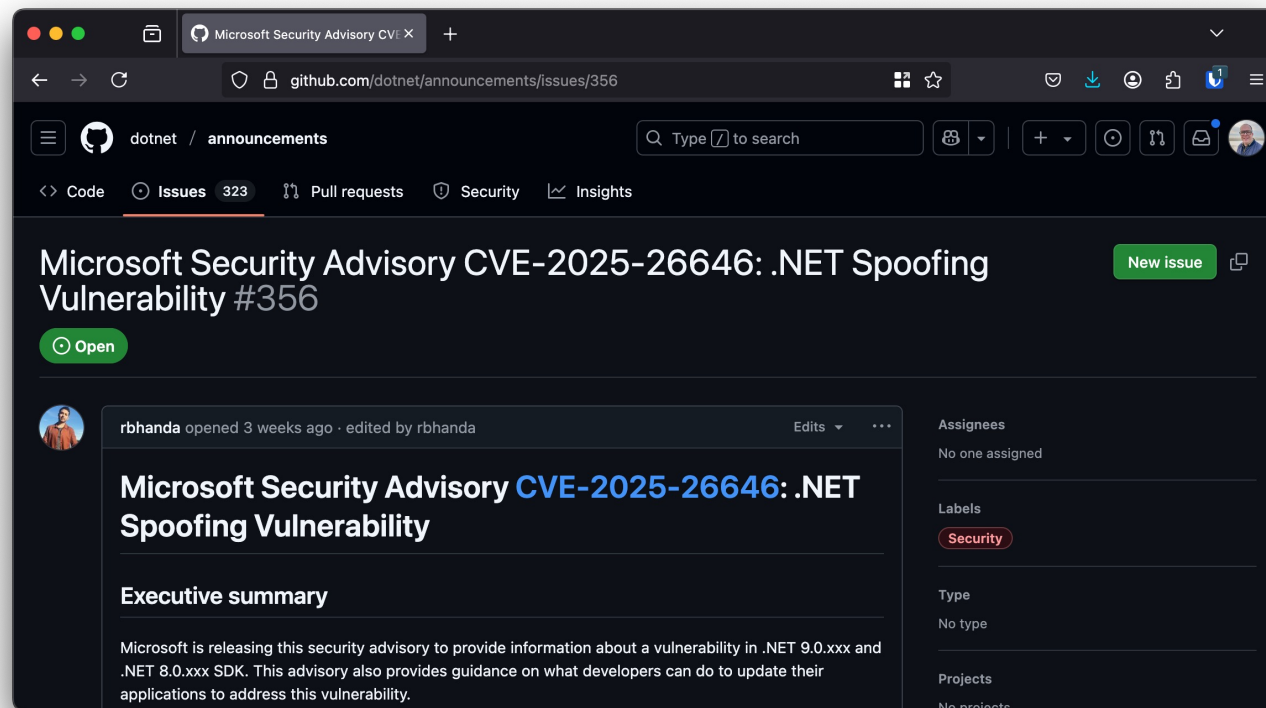


 @niels.fennec.dev  @nielstanis@infosec.exchange

Vulnerable Package



Vulnerabilities in Libraries



 @niels.fennec.dev  @nielstanis@infosec.exchange

DotNet CLI



```
nelson@ghost-m2:~/research/consoleapp
nelson@ghost-m2 ~/research/consoleapp $ dotnet list package --vulnerable --include-transitive

The following sources were used:
https://f.feedz.io/fennec/docgenerator/nuget/index.json
https://api.nuget.org/v3/index.json

Project `consoleapp` has the following vulnerable packages
[net8.0]:
Transitive Package      Resolved  Severity  Advisory URL
> Newtonsoft.Json       9.0.1     High      https://github.com/advisories/GHSA-5crp-9r3c-p9vr

nelson@ghost-m2 ~/research/consoleapp $
```

NPM Audit

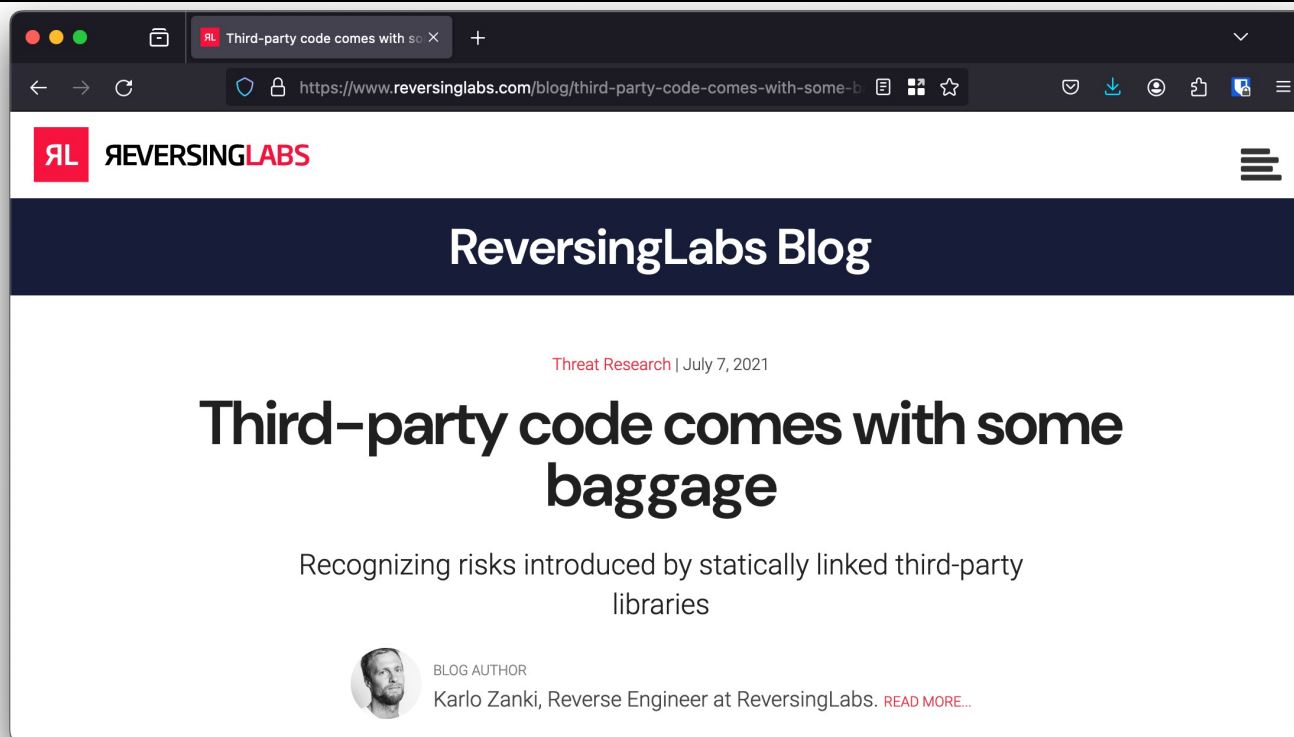


```
=== npm audit security report ===
```

```
# Run npm install chokidar@2.0.3 to resolve 1 vulnerability  
SEMVER WARNING: Recommended action is a potentially breaking change
```

Low	Prototype Pollution
Package	deep-extend
Dependency of	chokidar
Path	chokidar > fsevents > node-pre-gyp > rc > deep-extend
More info	https://nodesecurity.io/advisories/612

Do you know what's inside?



 @niels.fennec.dev  @nielstanis@infosec.exchange

Nutrition Label for Software?



OpenSSF (OSSF) Scorecard



OpenSSF Scorecard

securityscorecards.dev

Build better security habits, one test at a time

Quickly assess open source projects for risky practices

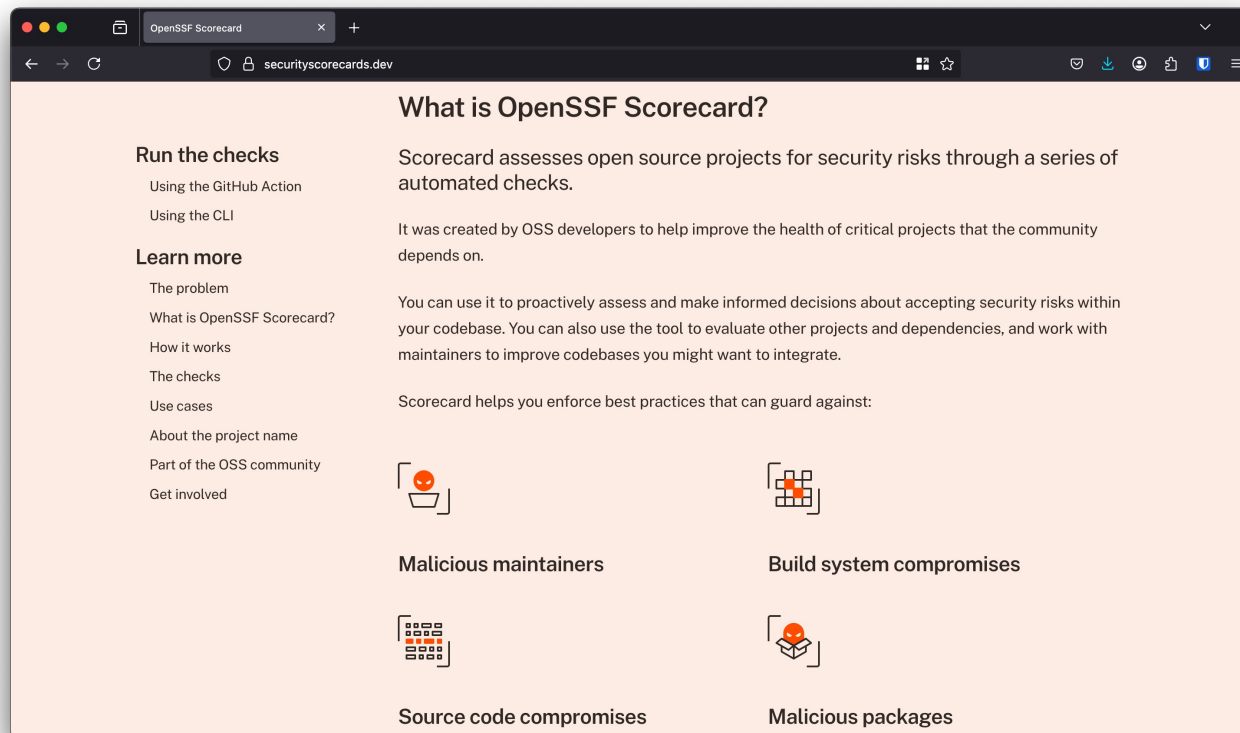
[Run the checks](#) [Learn more](#)

Critical	[09 / 10]	Branch-Protection	Branch protection is not maximal
Low	[00 / 10]	CI-Tests	No pull request found
Low	[00 / 10]	CII-Best-Practices	No badge found
High	[10 / 10]	Code-Review	Branch protection is enabled
Low	[00 / 10]	Contributors	No contributors found
High	[10 / 10]	Dangerous-Workflow	Workflow is safe
High	[00 / 10]	Dependency-Update-Tool	No update tool found



 @niels.fennec.dev  @nielstanis@infosec.exchange

OSSF Scorecard



 @niels.fennec.dev  @nielstanis@infosec.exchange

OSSF Scorecard Scoring



- **Total** = $\Sigma(\text{CheckScore} \times \text{RiskWeight}) / \Sigma(\text{RiskWeight})$
- **Severity Level** → **RiskWeight**

CRITICAL RISK	10
HIGH RISK	7.5
MEDIUM RISK	5
LOW RISK	2.5

Code Vulnerabilities (High)



- Does the project have unfixed vulnerabilities?
Uses the OSV service.

ID	Packages	Summary	Published ↓	Attributes
GHSA-hrww-x3fq-xcvh	NuGet/Umbraco.Cms	Umbraco CMS Improper Access Control vulnerability	20 Aug	<div>Fix available</div> <div>Severity - 6.3 (Medium)</div>
GHSA-77gj-crhp-3gvx	NuGet/Umbraco.Cms.Api.Management	Umbraco CMS vulnerable to Generation of Error Message Containing Sensitive Information	20 Aug	<div>Fix available</div> <div>Severity - 5.3 (Medium)</div>
GHSA-7grv-8f9x-3h32	NuGet/Microsoft.AspNetCore.App.Runtime.win-arm NuGet/Microsoft.AspNetCore.App.Runtime.win-arm64 NuGet/Microsoft.AspNetCore.App.Runtime.win-x64 NuGet/Microsoft.AspNetCore.App.Runtime.win-x86	Microsoft Security Advisory CVE-2024-38168 .NET Denial of Service Vulnerability	13 Aug	<div>Fix available</div> <div>Severity - 8.7 (High)</div>

Maintenance Dependency-Update-Tool (High)



- Does the project use a dependency update tool?
For example Dependabot or Renovate bot?
- Out-of-date dependencies make a project vulnerable to known flaws and prone to attacks.

Maintenance Security Policy (**Medium**)



- Does project have published security policy?
- E.g. a file named **SECURITY.md** (case-insensitive) in a few well-known directories.
- A security policy can give users information about what constitutes a vulnerability and how to report one securely so that information about a bug is not publicly visible.

Maintenance License (**Low**)



- Does project have license published?
- A license can give users information about how the source code may or may not be used.
- The lack of a license will impede any kind of security review or audit and creates a legal risk for potential users.

Maintenance CII Best Practices (**Low**)



- OpenSSF Best Practices Badge Program
- Way for Open Source Software projects to show that they follow best practices.
- Projects can voluntarily self-certify, at no cost, by using this web application to explain how they follow each best practice.



openssf best practices **passing**



 @niels.fennec.dev  @nielstanis@infosec.exchange

Continuous testing CI Tests (**Low**)



- Does the project run tests before pull requests are merged?
- The check works by looking for a set of CI-system names in GitHub CheckRuns and Statuses among the recent commits (~30).

Continuous testing Fuzzing (**Medium**)



- This check tries to determine if the project uses fuzzing by checking:
 - Added to [OSS-Fuzz](#) project.
 - If [ClusterFuzzLite](#) is deployed in the repository
 - Language based property testers

Continuous testing Static Code Analysis (Medium)



- This check tries to determine if the project uses Static Application Security Testing (SAST), also known as static code analysis. It is currently limited to repositories hosted on GitHub.
 - CodeQL
 - SonarCloud
 - Qodana

Source Risk Assessment Binary Artifacts (**High**)



- This check determines whether the project has generated executable (binary) artifacts in the source repository.
- Binary artifacts cannot be reviewed, allowing possible obsolete or maliciously subverted executables.
- There is need for **reproducible** builds!

Source Risk Assessment Branch Protection (High)



- This check determines whether a project's default and release branches are protected with GitHub's branch protection or repository rules settings.
 - Requiring code review
 - Prevent force push, in case of public branch all is lost!

Source Risk Assessment Dangerous Workflow (**Critical**)



- This check determines whether the project's GitHub Action workflows has dangerous code patterns.
 - Untrusted Code Checkout with certain triggers
 - Script Injection with Untrusted Context Variables
- <https://securitylab.github.com/research/github-actions-preventing-pwn-requests/>

Source Risk Assessment Code Review (**Low**)



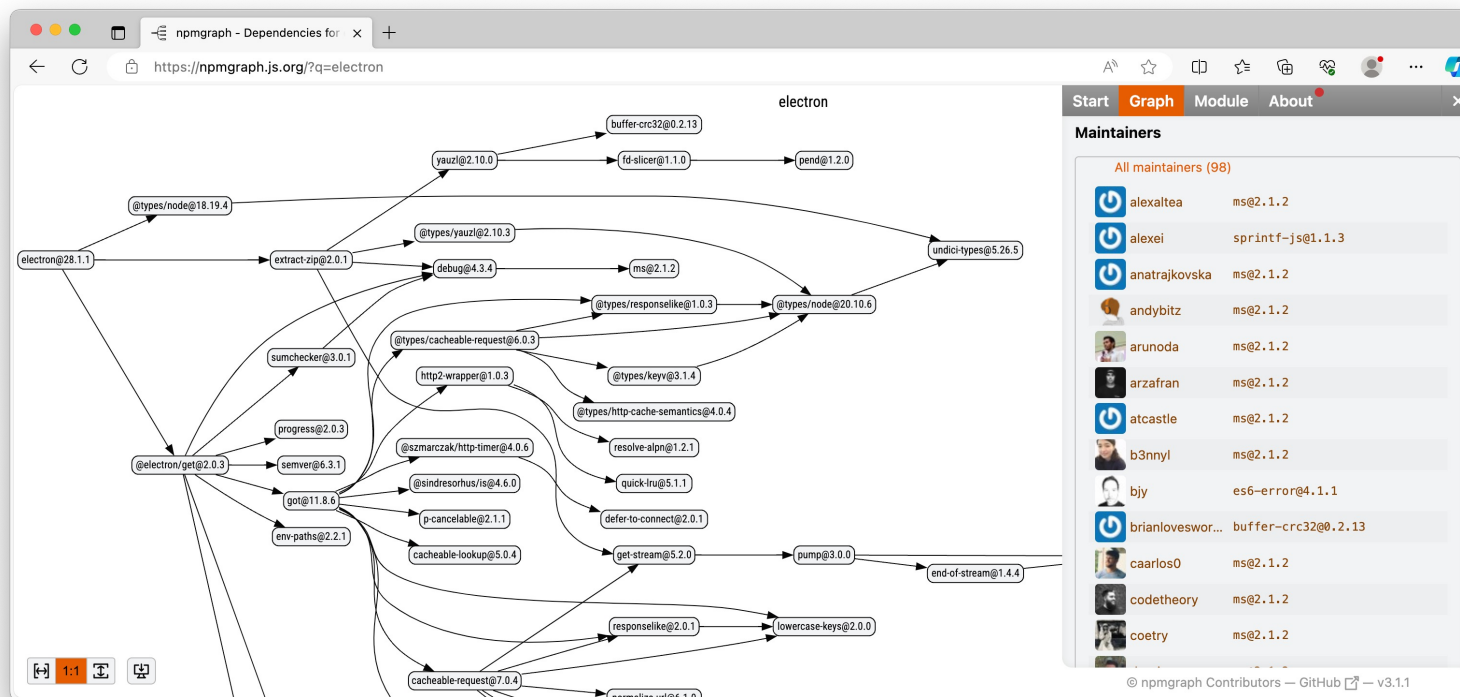
- This check determines whether the project requires human code review before pull requests are merged.
- The check determines whether the most recent changes (over the last ~30 commits) have an approval on GitHub and `merger!=committer` (implicit review)

Source Risk Assessment Contributors (**Low**)



- This check tries to determine if the project has recent contributors from multiple organizations (e.g., companies).
- Relying on single contributor is a risk for sure!
- But is a large list of contributors good?

Source Risk Assessment Contributors (**Low**)



Build Risk Assessment Pinned Dependencies (**High**)



- Does the project pin dependencies used during its build and release process.
- If Workflow is present what about the Actions used?

Build Risk Assessment Token Permission (High)



- This check determines whether the project's automated workflows tokens follow the principle of least privilege.
- This is important because attackers may use a compromised token with write access to, for example, push malicious code into the project.

Build Risk Assessment Packaging (Medium)



- This check tries to determine if the project is published as a package.
- Packages give users of a project an easy way to download, install, update, and uninstall the software by a package manager.

Build Risk Assessment Signed Releases (**High**)



- This check tries to determine if the project cryptographically signs release artifacts.
 - Signed release packages
 - Signed build provenance

Demo OpenSSF Scorecard Fennec CLI

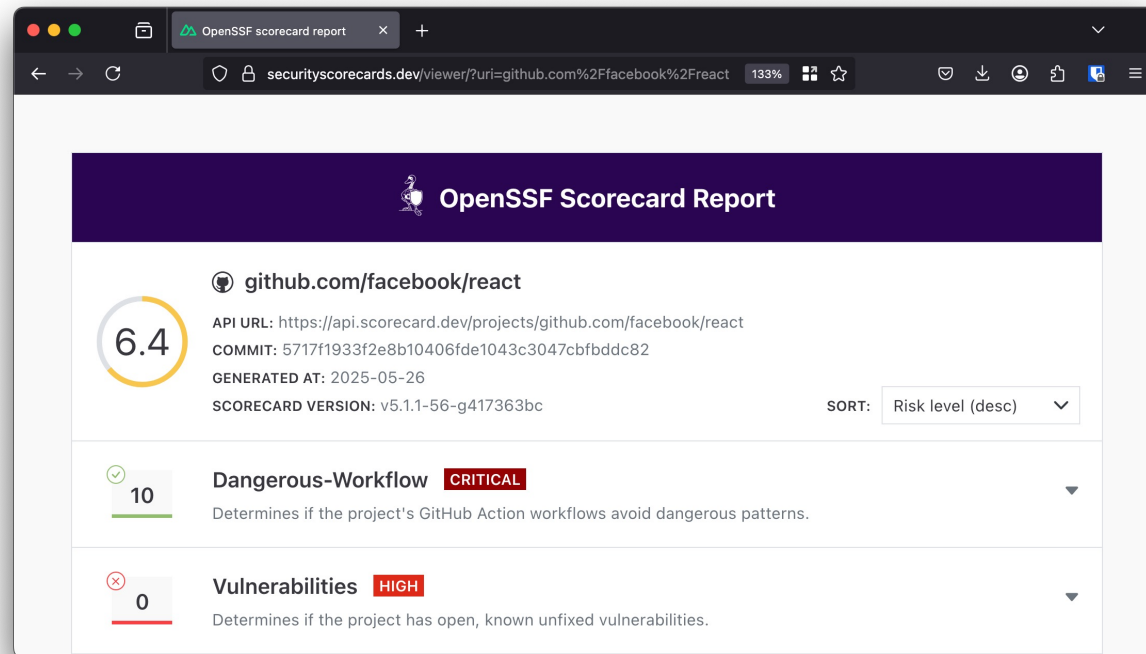


Running checks



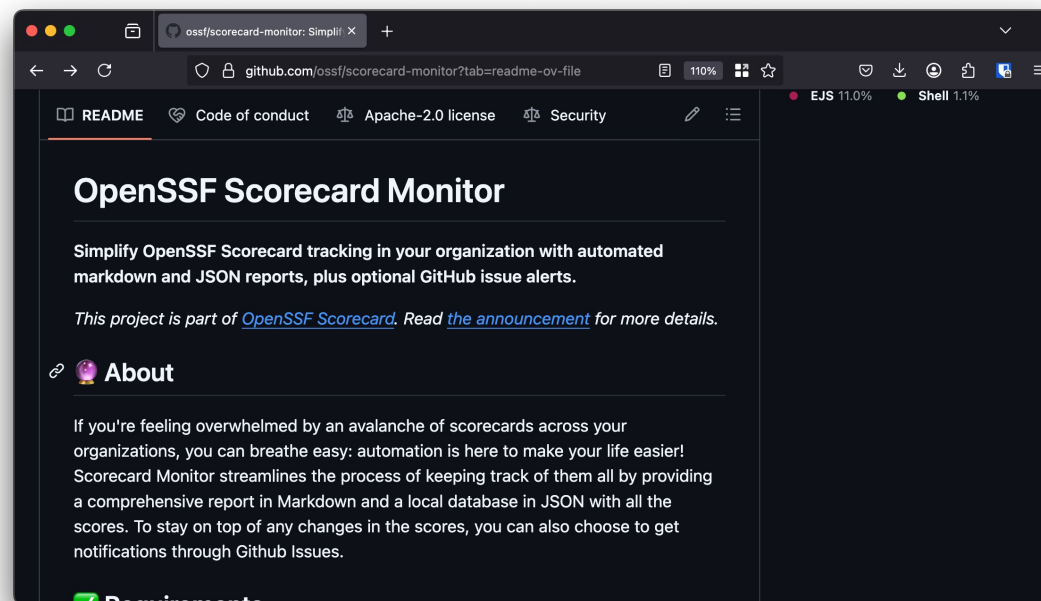
 @niels.fennec.dev  @nielstanis@infosec.exchange

Scorecard Viewer



 @niels.fennec.dev  @nielstanis@infosec.exchange

Scorecard Monitor



 @niels.fennec.dev  @nielstanis@infosec.exchange

Measure?



 @niels.fennec.dev  @nielstanis@infosec.exchange

OpenSSF Annual Report 2024



OpenSSF Scorecard: Released v5.0.0, featuring Structured Results and Maintainer Annotations. This expands the existing 18 checks into 47 probes providing enhanced granularity and customization.

Boasts 7.6k installs of scorecard-action and 3.4k repositories displaying OpenSSF Scorecard badges in their READMEs.



@niels.fennec.dev



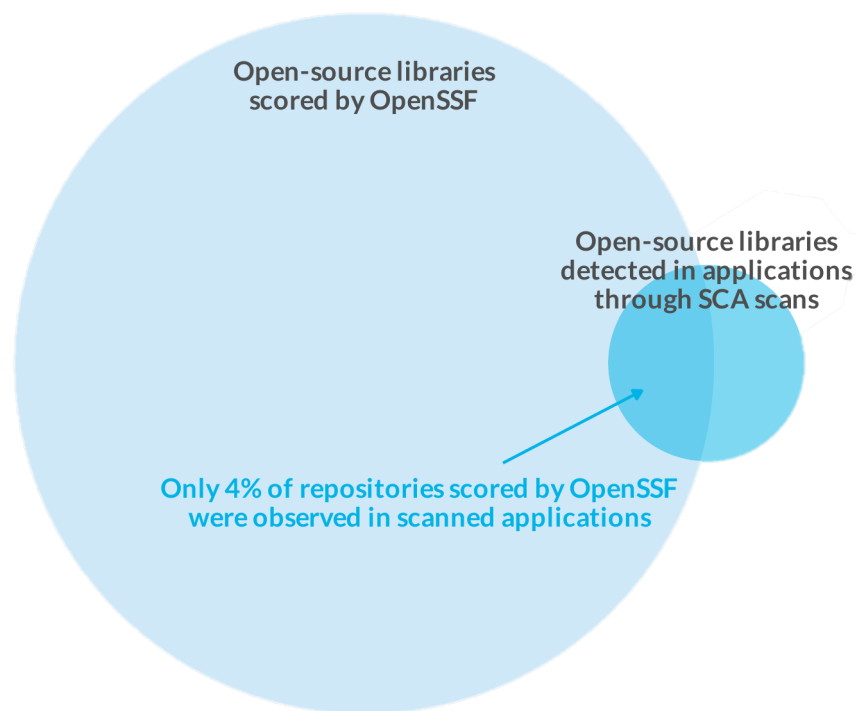
@nielstanis@infosec.exchange

SOSS & OpenSSF Scorecard



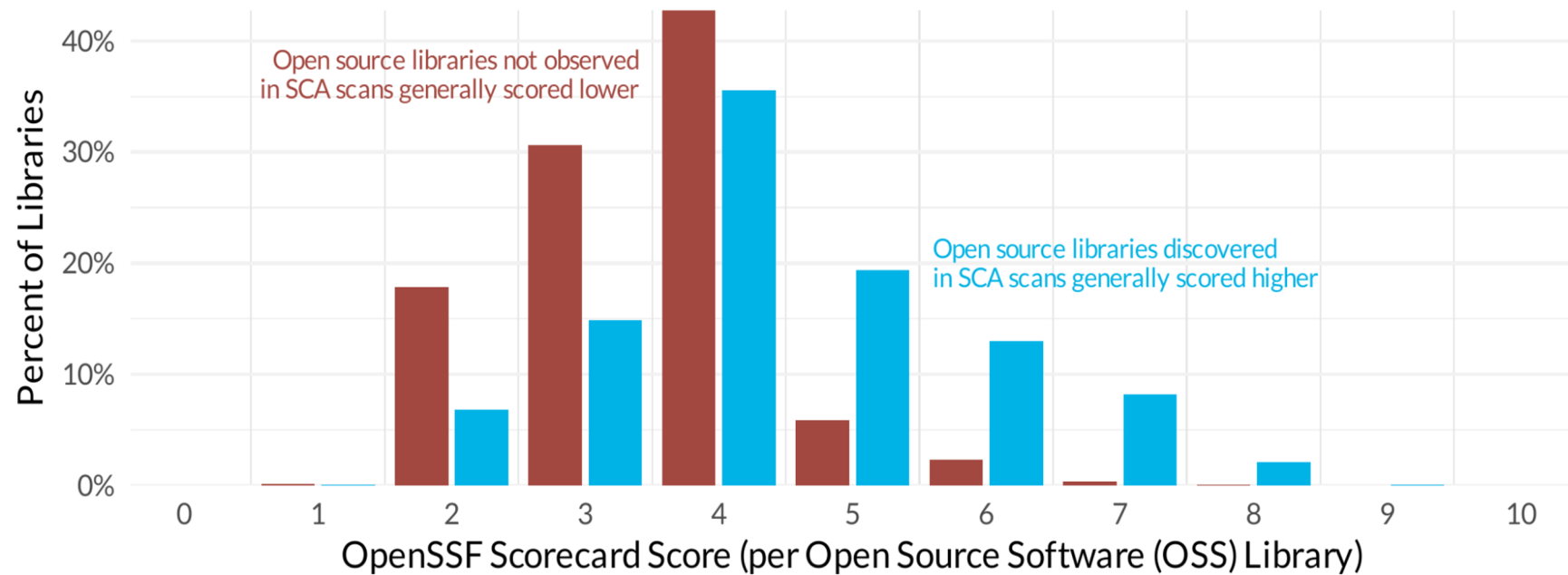
 [@niels.fennec.dev](https://twitter.com/niels.fennec.dev)  [@nielstanis@infosec.exchange](https://mastodon.social/@nielstanis)

SOSS & OpenSSF Scorecard

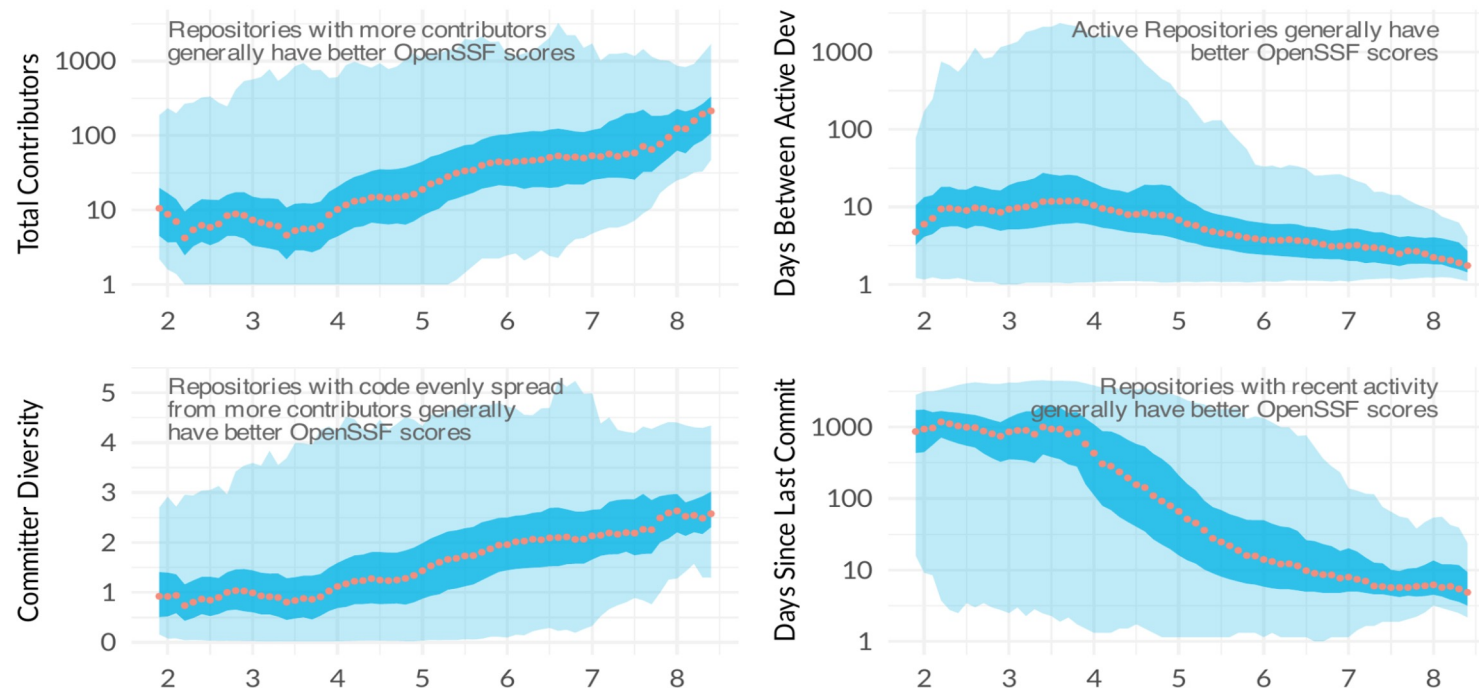


 @niels.fennec.dev  @nielstanis@infosec.exchange

Correlation between SOSS



Github commits vs OpenSSF



What really contributes to OSS Security?



Dependency-Update-Tool: No update tool detected

Security-Policy: Security policy file detected

Fuzzing: Project is fuzzed

Code-Review: Performed regularly

Custom: Project inactivity (months)

Custom: Active development (years)

Custom: Applications using the project

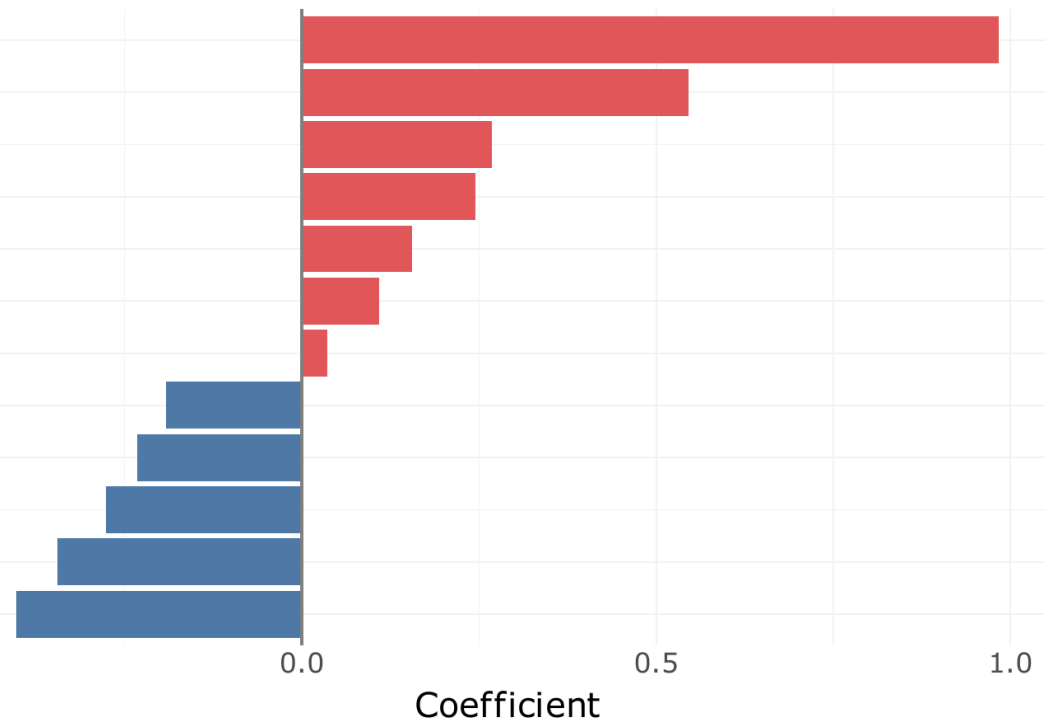
Custom: Diversity in contributors and commits

Binary-Artifacts: No binaries found in the repo

Signed-Releases: All artifacts signed

SAST: SAST tool detected

Maintained: Project is archived



@niels.fennec.dev



@nielstanis@infosec.exchange

What's next?



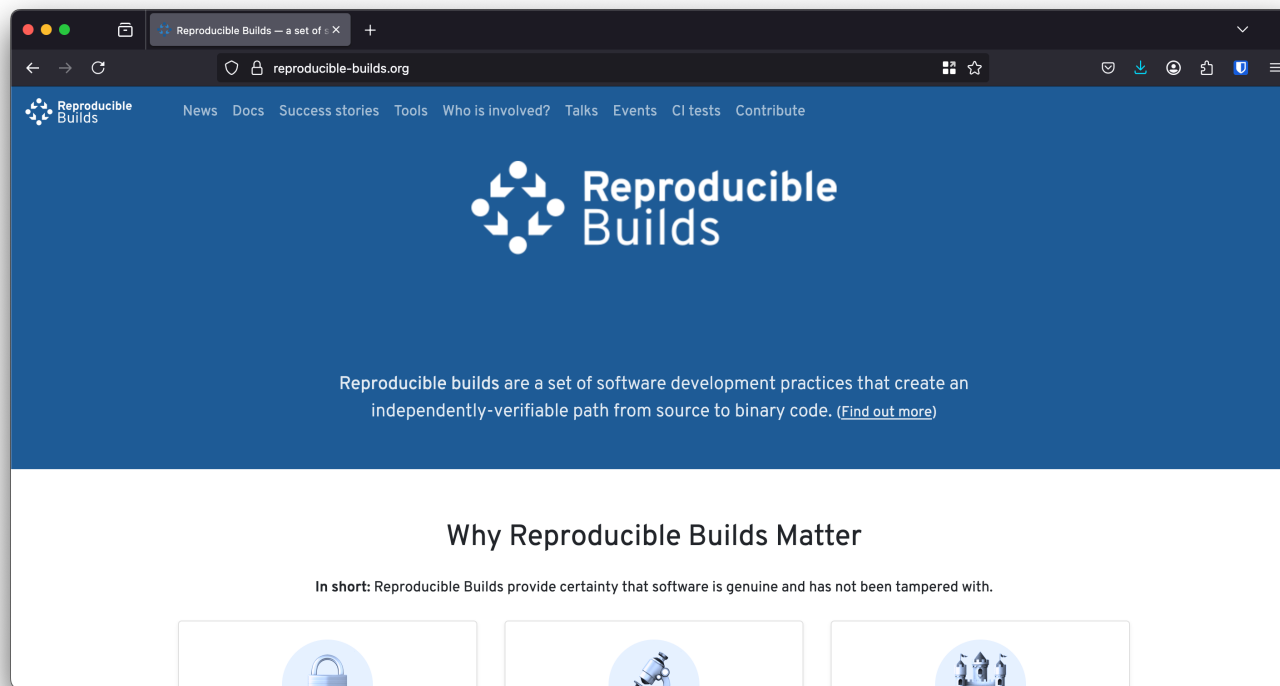
 [@niels.fennec.dev](https://twitter.com/niels.fennec.dev)  [@nielstanis@infosec.exchange](https://mastodon.social/@nielstanis)

What can be improved?



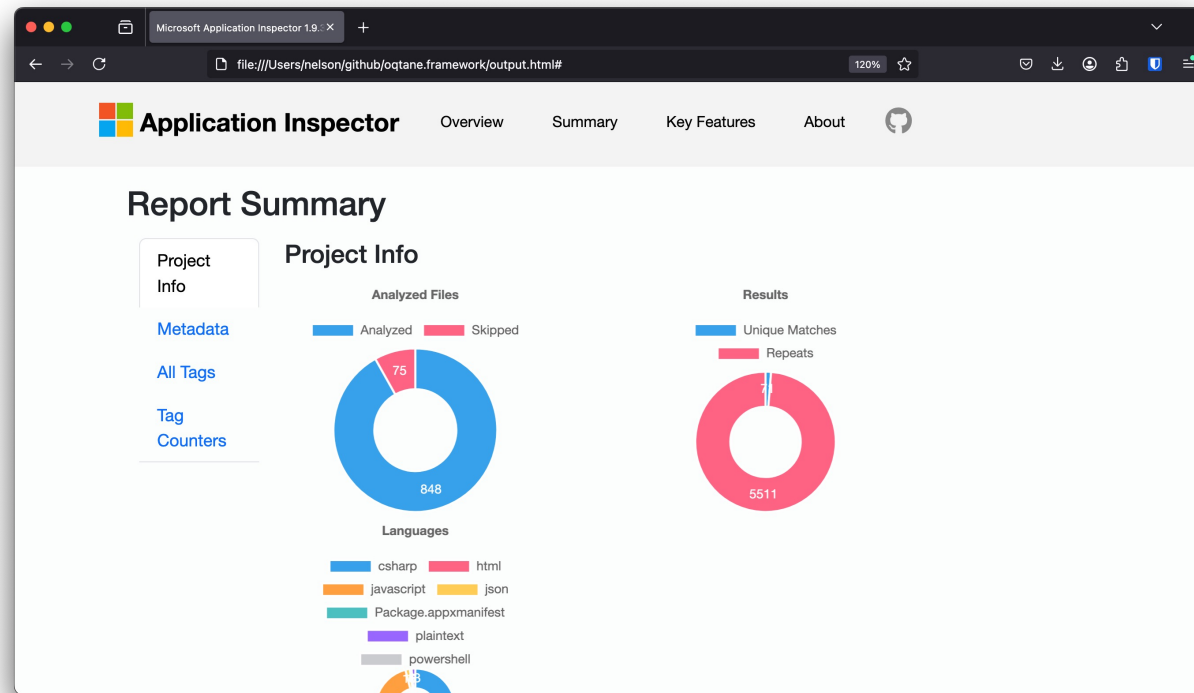
- Better support for recognizing fuzzing including the managed languages like Java/.NET
- Better support for SAST tools & working with the results.
- Reproducible builds
- What's the library using?

Reproducible Builds



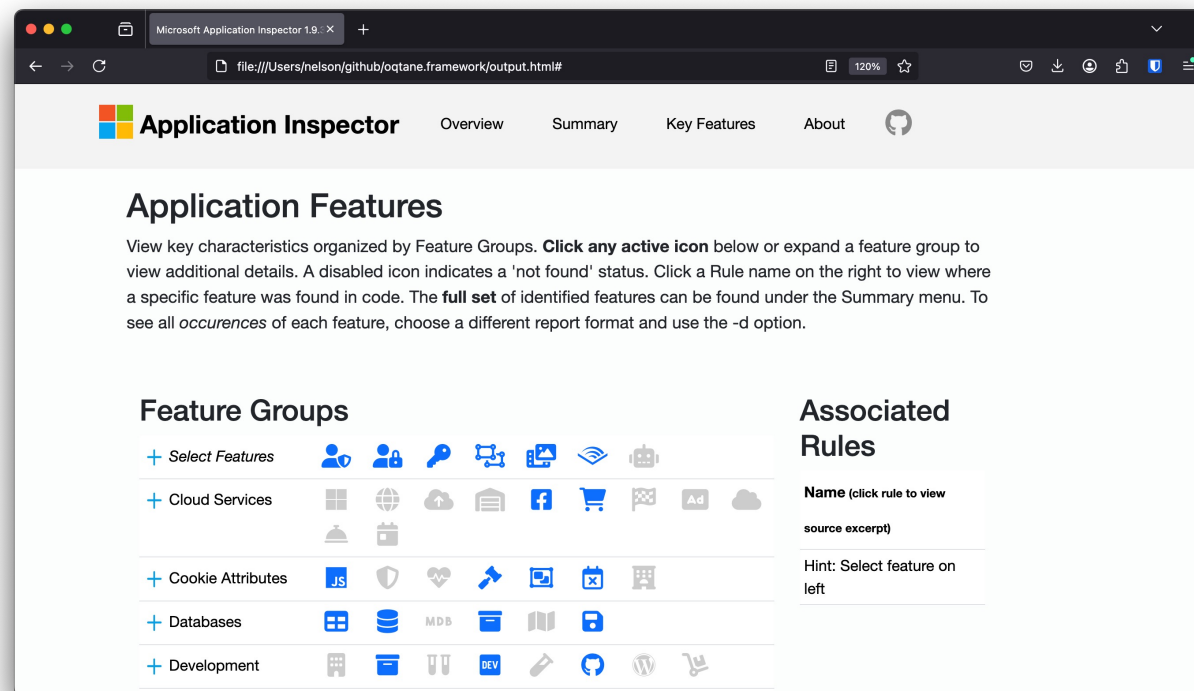
 @niels.fennec.dev  @nielstanis@infosec.exchange

Application Inspector



 @niels.fennec.dev  @nielstanis@infosec.exchange

Application Inspector



@niels.fennec.dev



@nielstanis@infosec.exchange

Application Inspector



Microsoft Application Inspector 1.9

file:///Users/nelson/github/oqtane.framework/output.html#

Application Features

View key characteristics organized by Feature Groups. **Click any active icon** below or expand a feature group to view additional details. A disabled icon indicates a 'not found' status. Click a Rule name on the right to view where a specific feature was found in code. The **full set** of identified features can be found under the Summary menu. To see all *occurrences* of each feature, choose a different report format and use the -d option.

Feature Groups

[Select Features](#)

Feature	Confidence	Details
Authentication		View
Authorization		View
Cryptography		View
Object deserialization		View
AV media parsing		View
Dynamic command execution		View
AI		N/A

Associated Rules

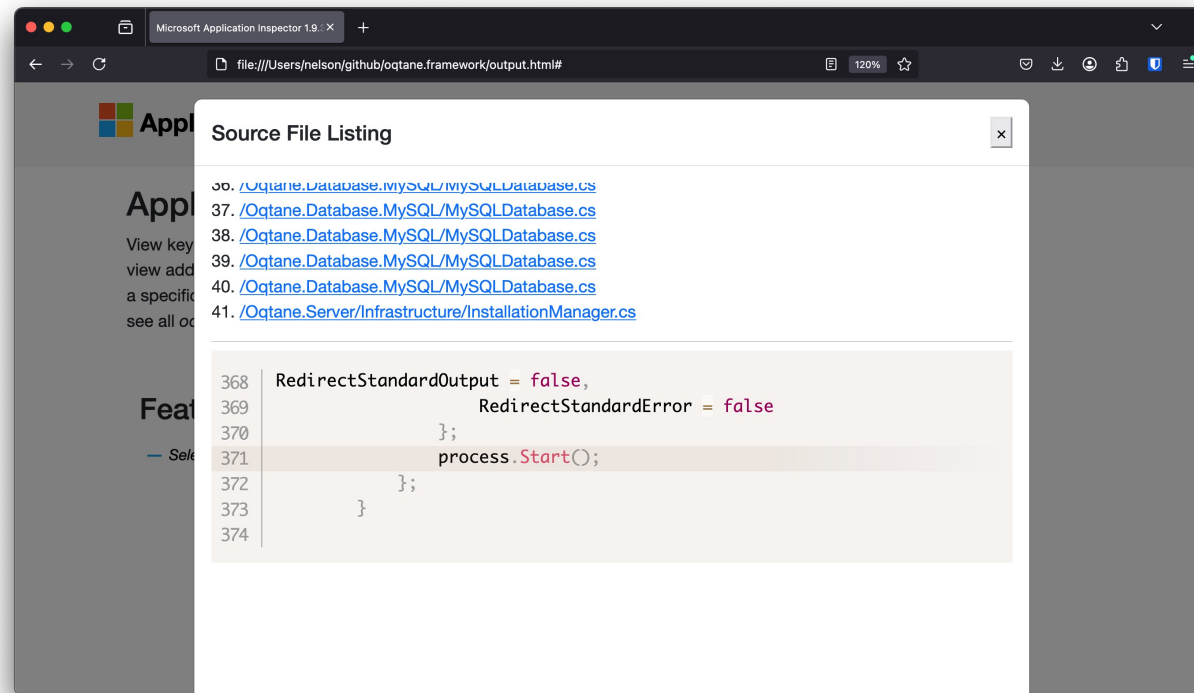
Name (click rule to view source excerpt)

[OS: Dynamic Execution](#)



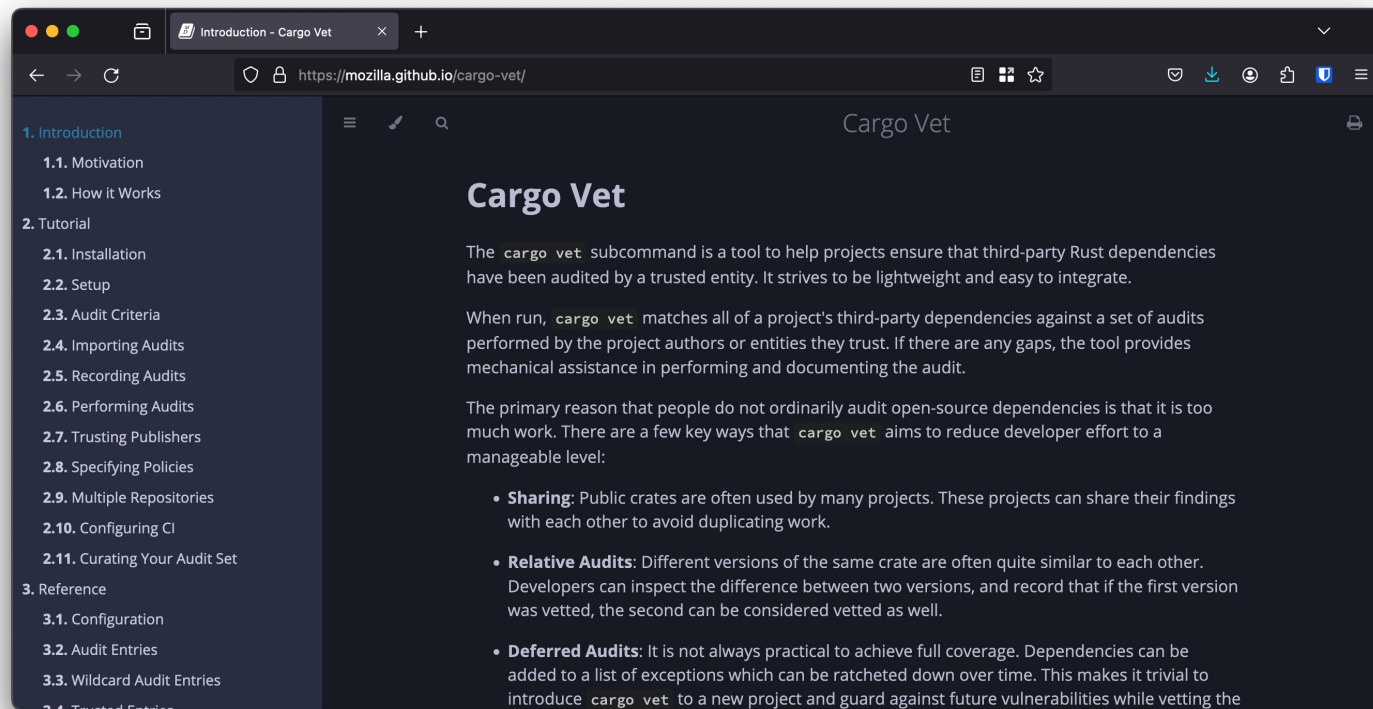
@niels.fennec.dev @nielstanis@infosec.exchange

Application Inspector



 @niels.fennec.dev  @nielstanis@infosec.exchange

Community Review



@niels.fennec.dev

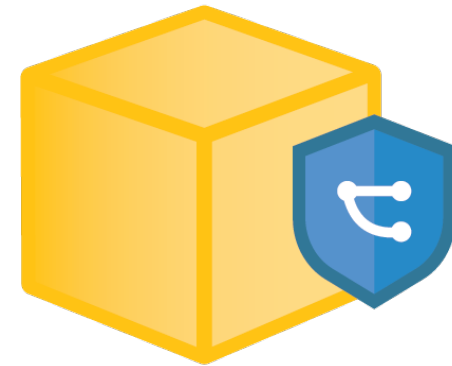


@nielstanis@infosec.exchange

OSSF Secure Supply Chain Consumption Framework Project



- The Secure Supply Chain Consumption Framework (S2C2F) is a security assurance and risk reduction process that is focused on securing how developers consume open source software.



OSSF Secure Supply Chain Consumption Framework Project



3 core concepts



3 goals of the framework



 @niels.fennec.dev  @nielstanis@infosec.exchange

OSSF Secure Supply Chain Consumption Framework Project



- **Level 1** - Implements foundational OSS security practices including package caching, inventory management, vulnerability scanning, and regular updates.
- **Level 2** - Focuses on automated, rapid response capabilities to patch OSS vulnerabilities faster than attackers can exploit them, with improved configuration security and incident response.
- **Level 3** - Proactively analyses the most-used OSS components for undiscovered vulnerabilities and implements malware scanning to prevent consumption of malicious packages.
- **Level 4** - Rebuilds OSS components on trusted internal infrastructure to defend against sophisticated build-time supply chain attacks, though this approach is difficult to implement at scale.

Conclusion



- Scorecard helps security reviewing a 3rd Party Package
- Better understand what's inside, how it's build/maintained and what are the risks
- Scorecard should not be a goal on its own!
- In what way can you use scorecard data?
- Look into frameworks like S2C2F to help out !

Merci! Bedankt! Thanks!



- <https://github.com/nielstanis/cyberseccoalition25/>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://www.fennec.dev>
<https://blog.fennec.dev>



 @niels.fennec.dev  @nielstanis@infosec.exchange