VERACODE

Reviewing 3rd Party Library Security Easily Using OpenSSF Scorecard

Niels Tanis
Sr. Principal Security Researcher

WeAreDevelopers World Congress

Who am I?

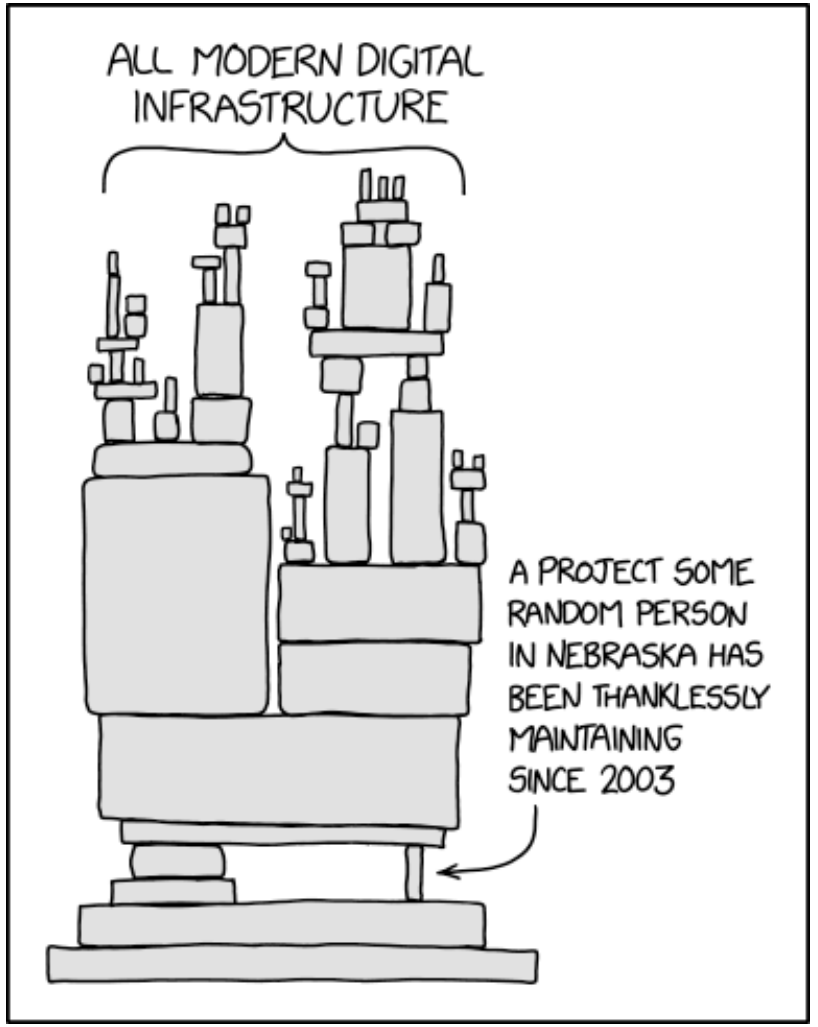
- Niels Tanis
- Sr. Principal Security Researcher
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - Research on static analysis for .NET apps
  - Enjoying Rust!
- Microsoft MVP – Developer Technologies

VERACODE

MVP Microsoft® Most Valuable Professional

@nielstanis@infosec.exchange

Modern Application Architecture XKCD 2347



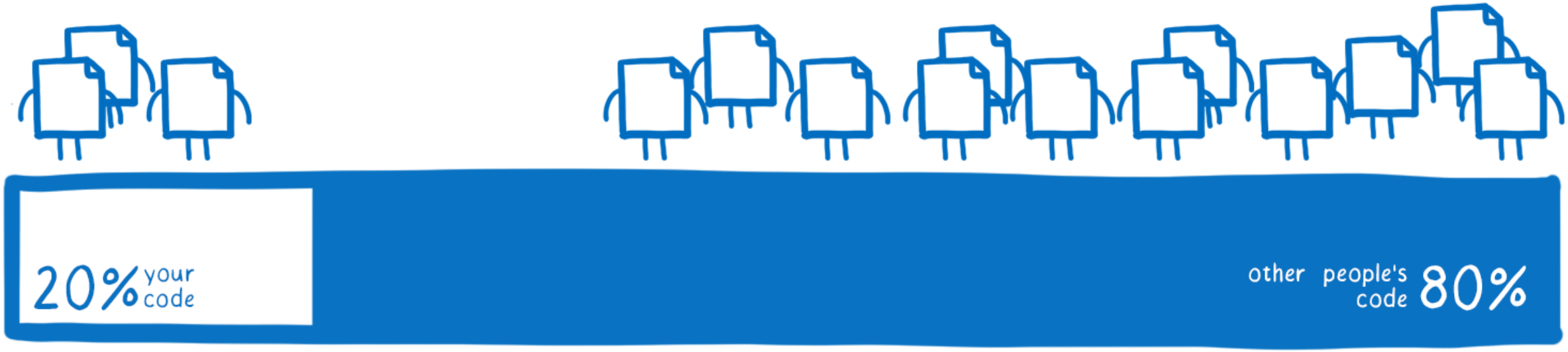
@nielstanis@infosec.exchange

Agenda

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

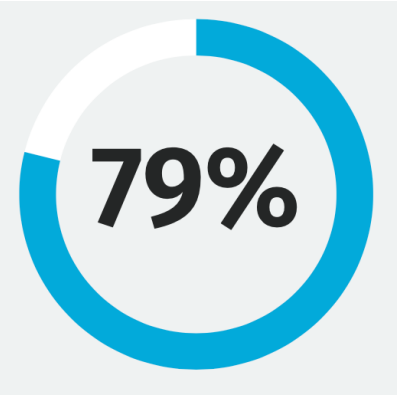
@nielstanis@infosec.exchange

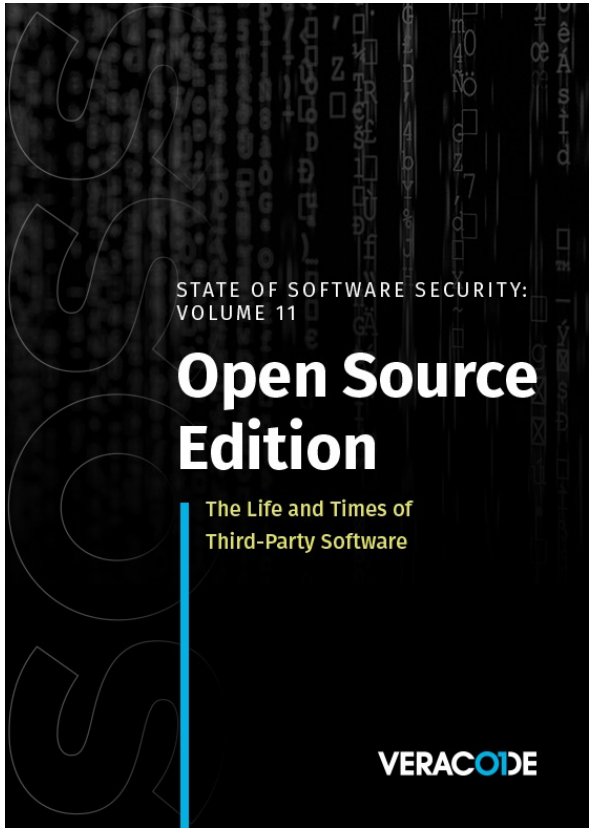
Average codebase composition



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.”

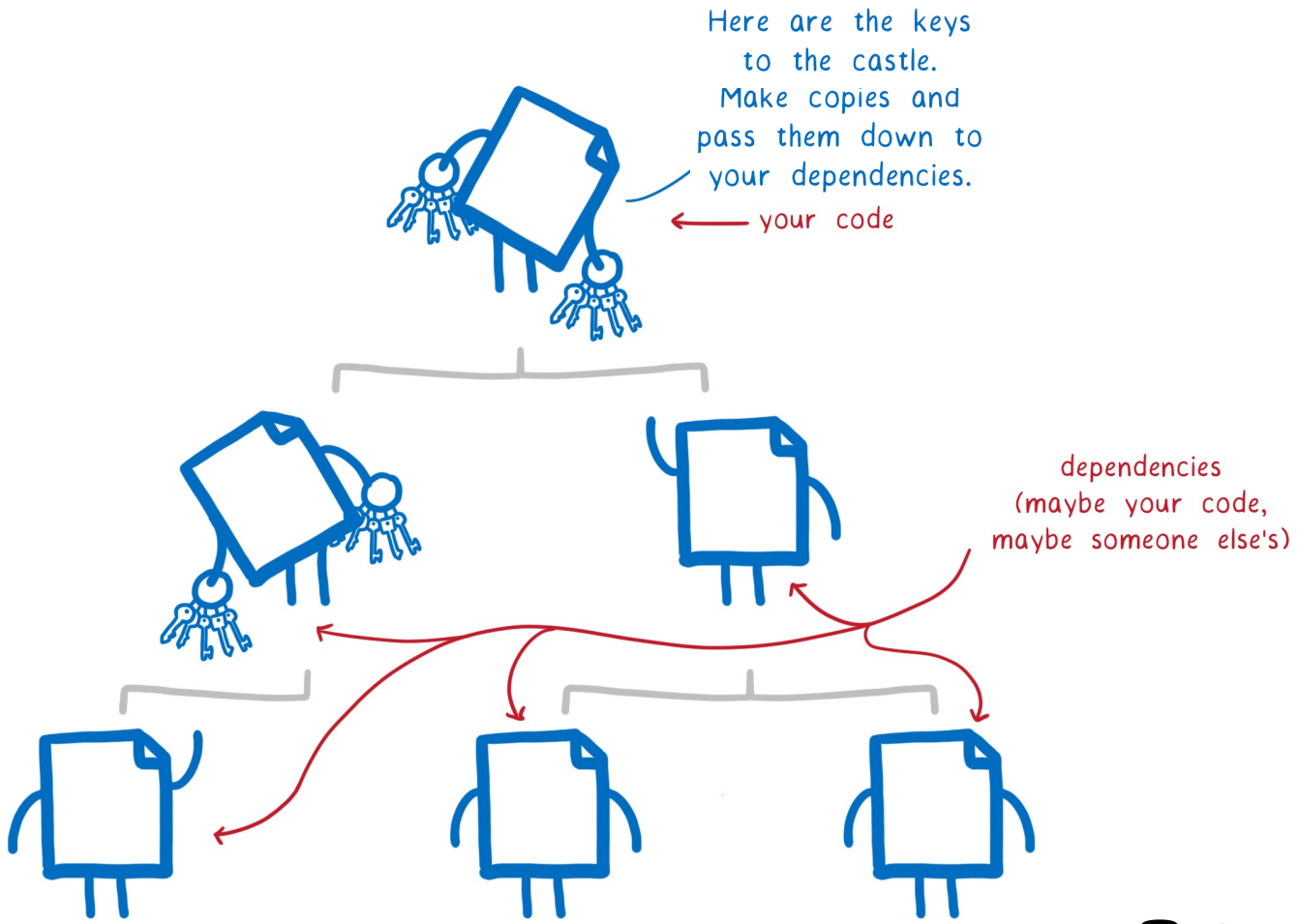




State of Log4j – 2 years later

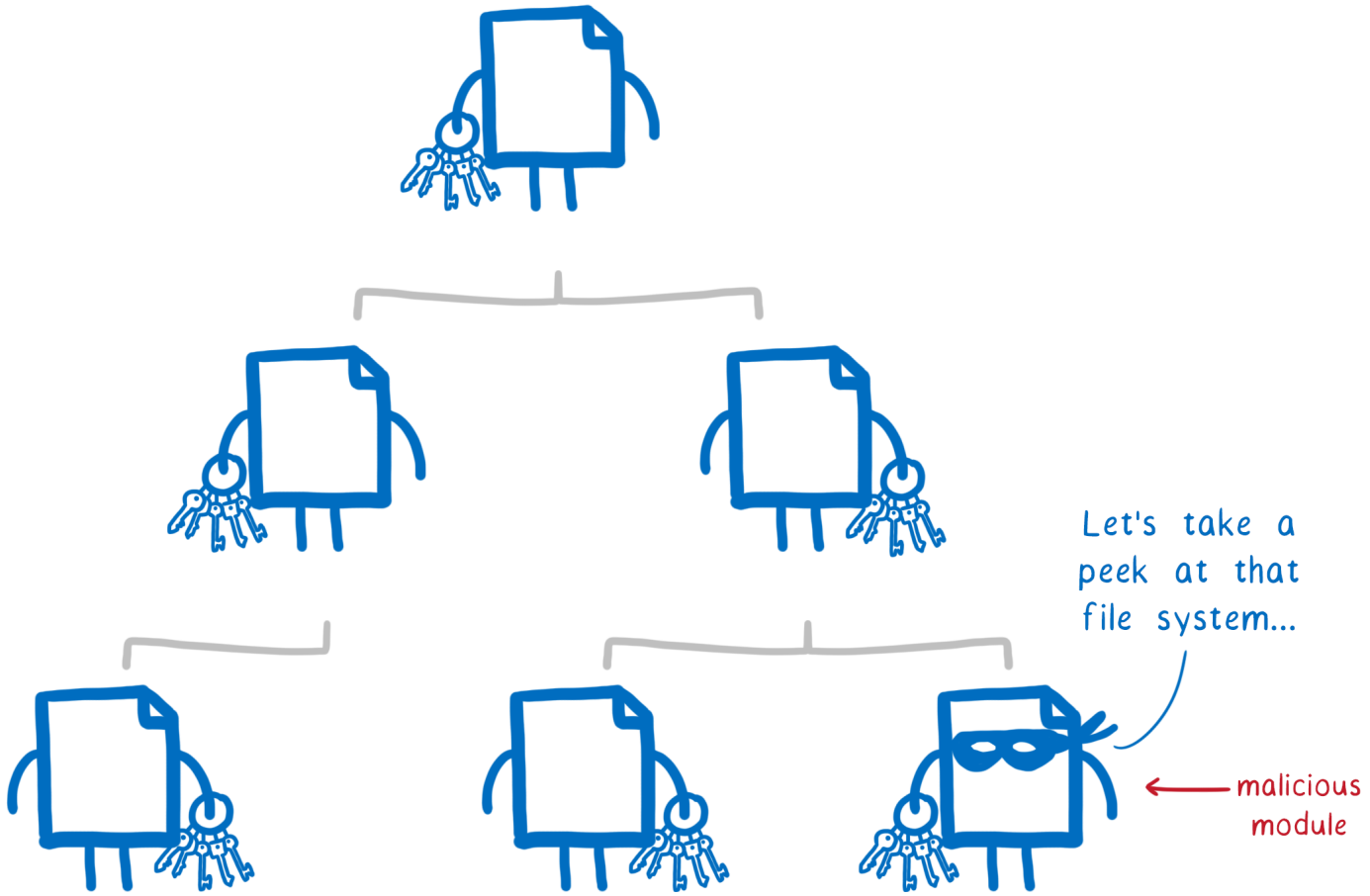
- Analysed our data August-November 2023
  - Total set of almost 39K unique applications scanned
- 2.8% run version vulnerable to Log4Shell
- 3.8% run version patched but vulnerable to other CVE
- 32% rely on a version that's end-of-life and have no support for any patches.

Average codebase composition



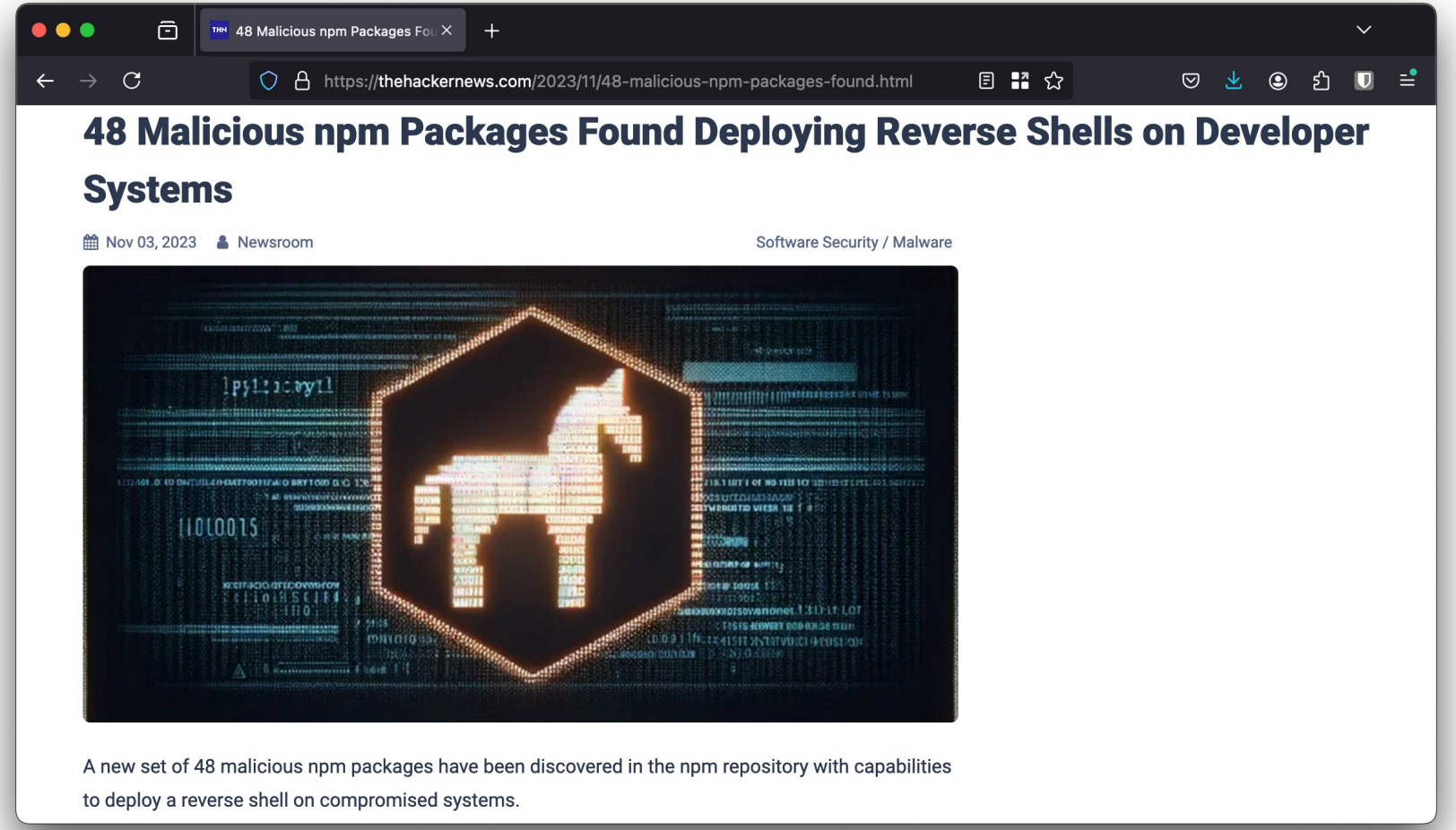
@nielstanis@infosec.exchange

Malicious Library



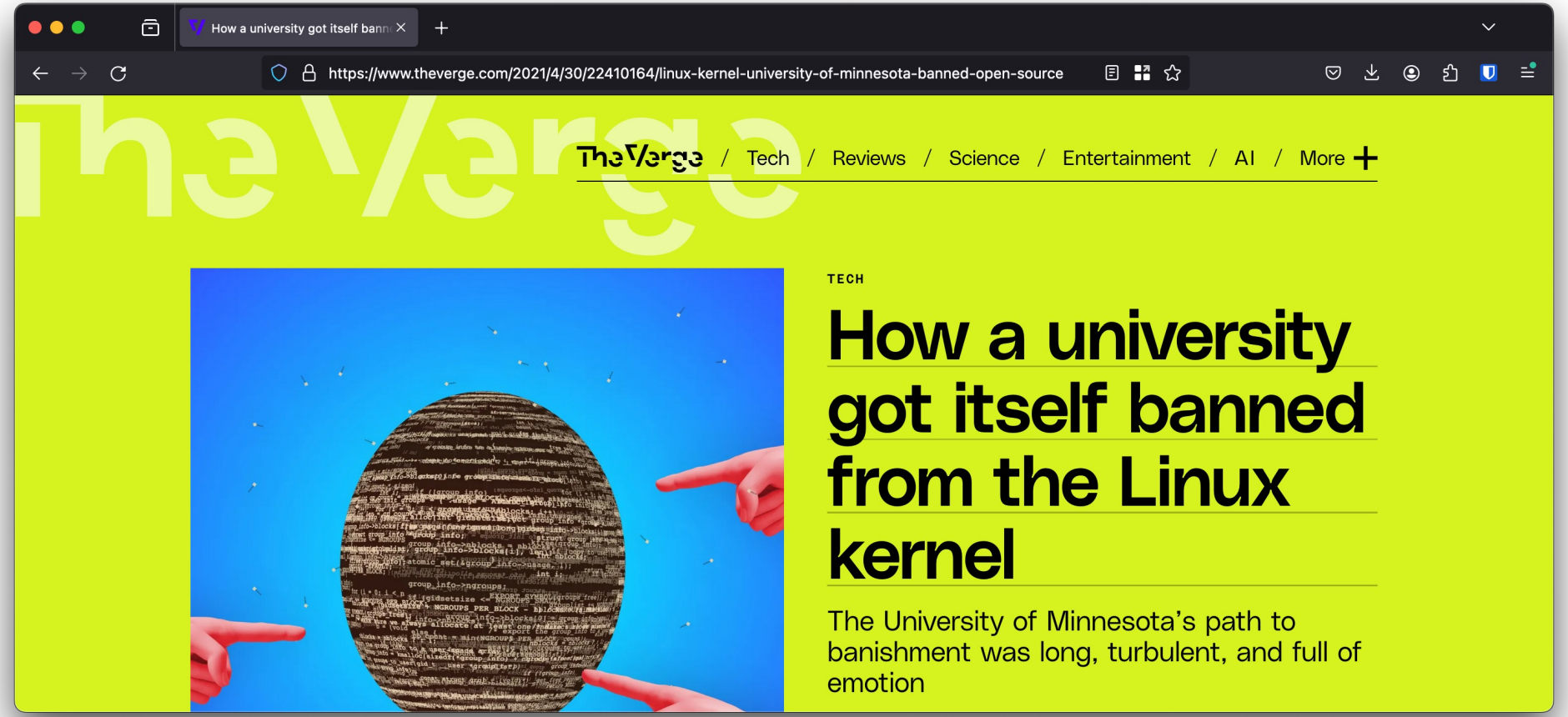
@nielstanis@infosec.exchange

Malicious Package

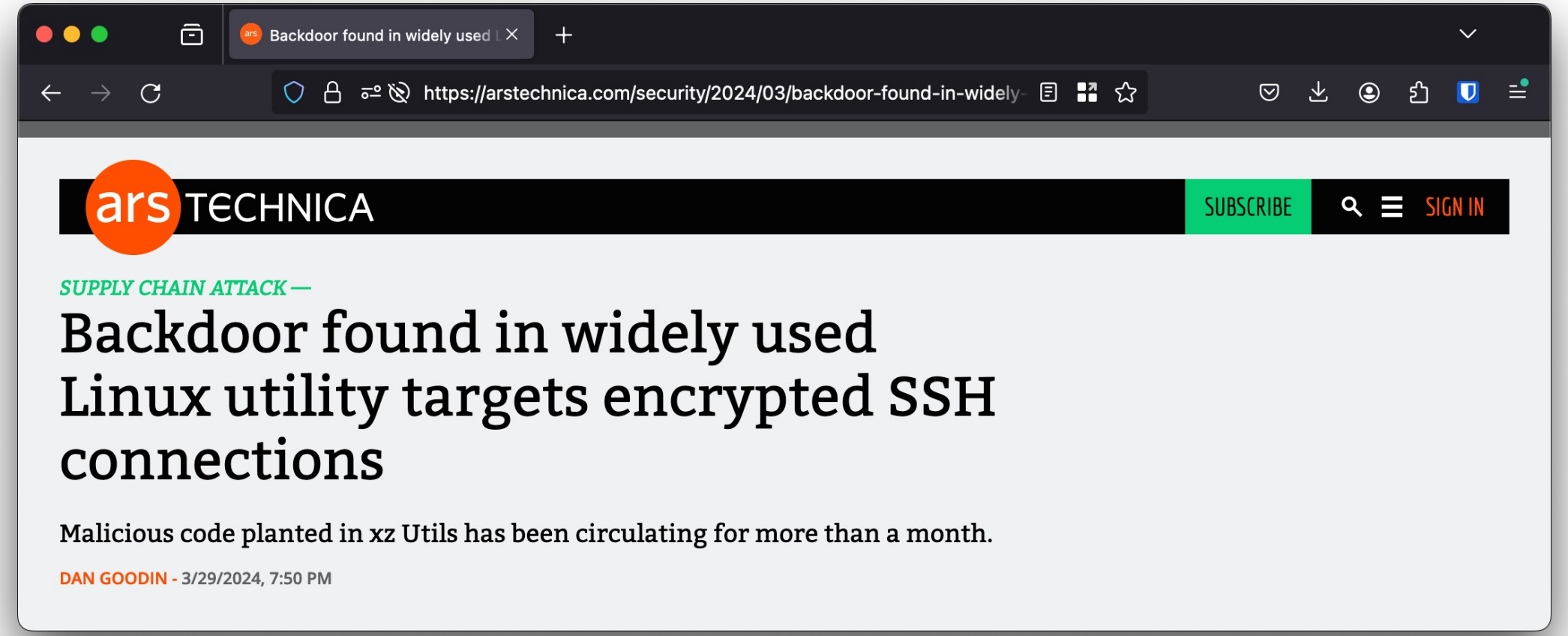


@nielstanis@infosec.exchange

Hypocrite Commits

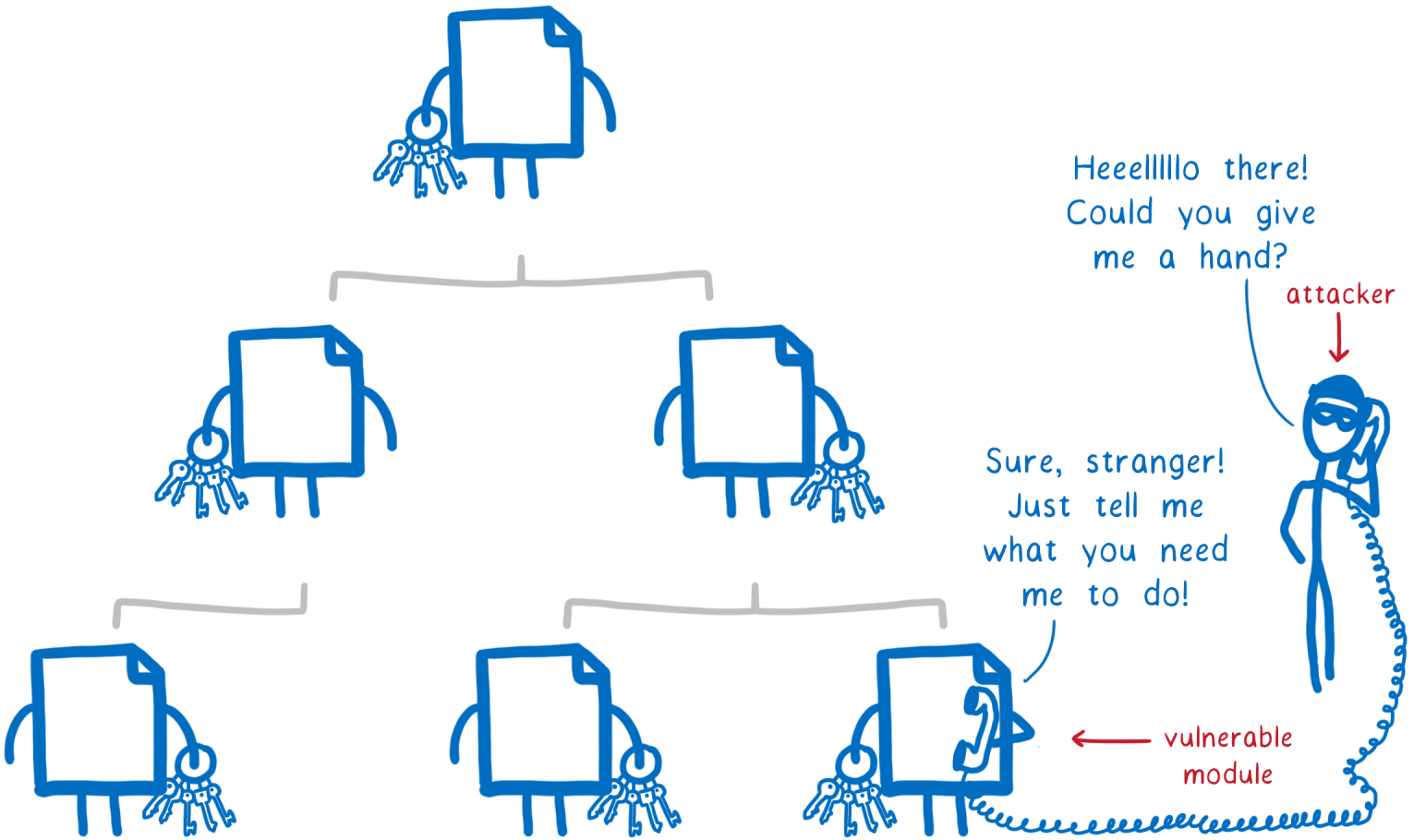


XZ Backdoor



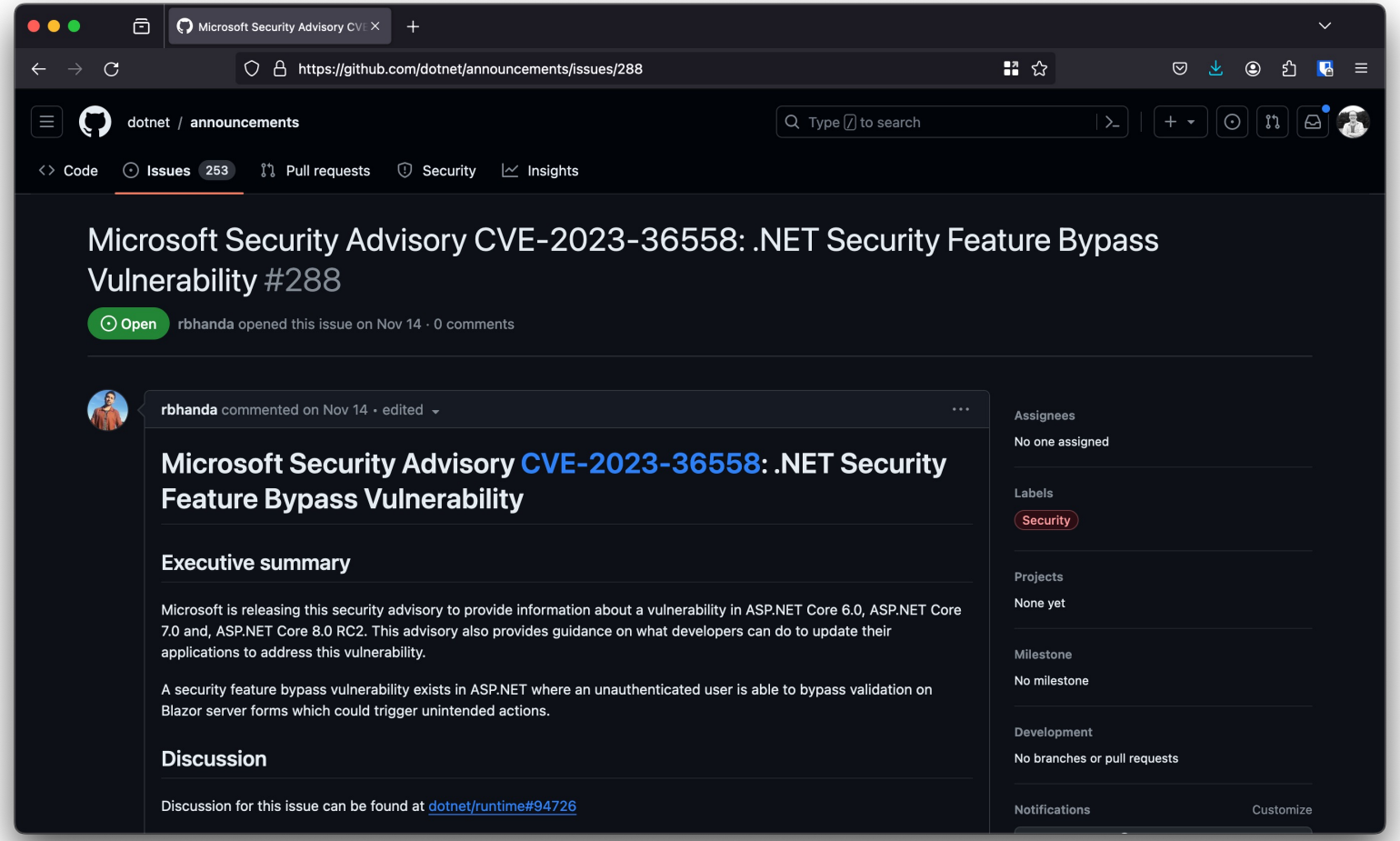
@nielstanis@infosec.exchange

Vulnerable Package



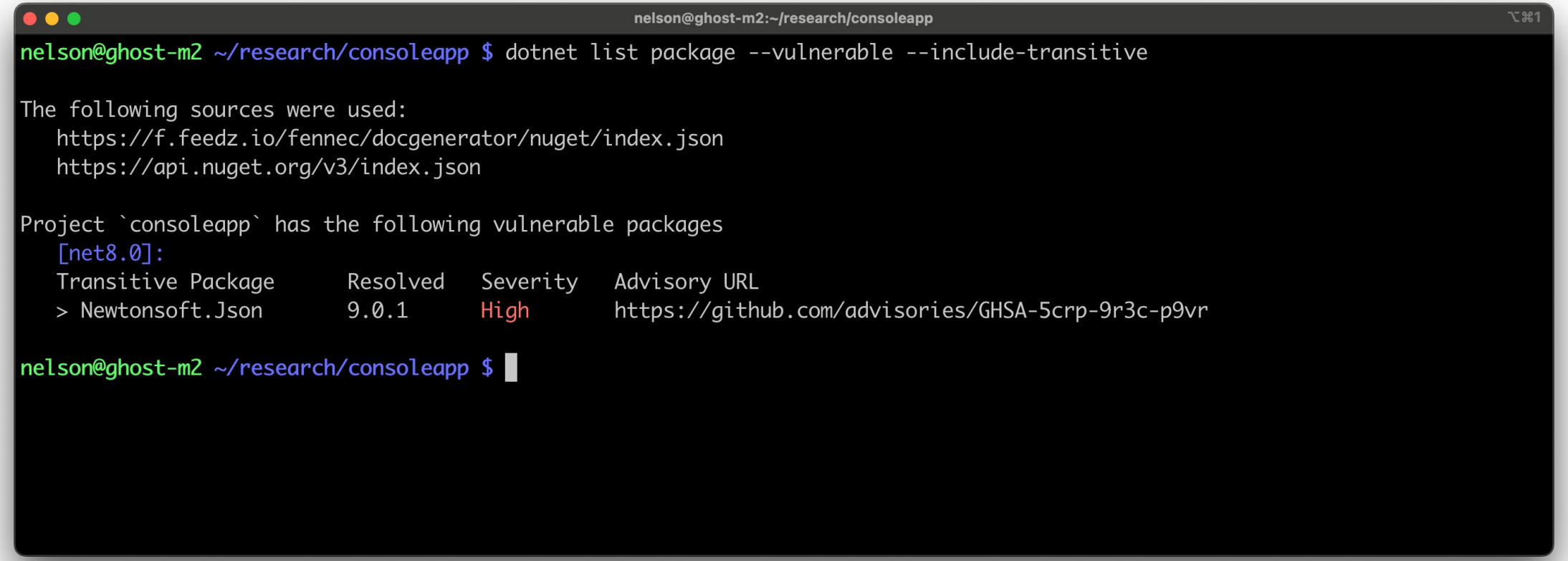
@nielstanis@infosec.exchange

Vulnerabilities in Libraries



@nielstanis@infosec.exchange

DotNet CLI


nelson@ghost-m2:~/research/consoleapp
⌥⌘1
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 $

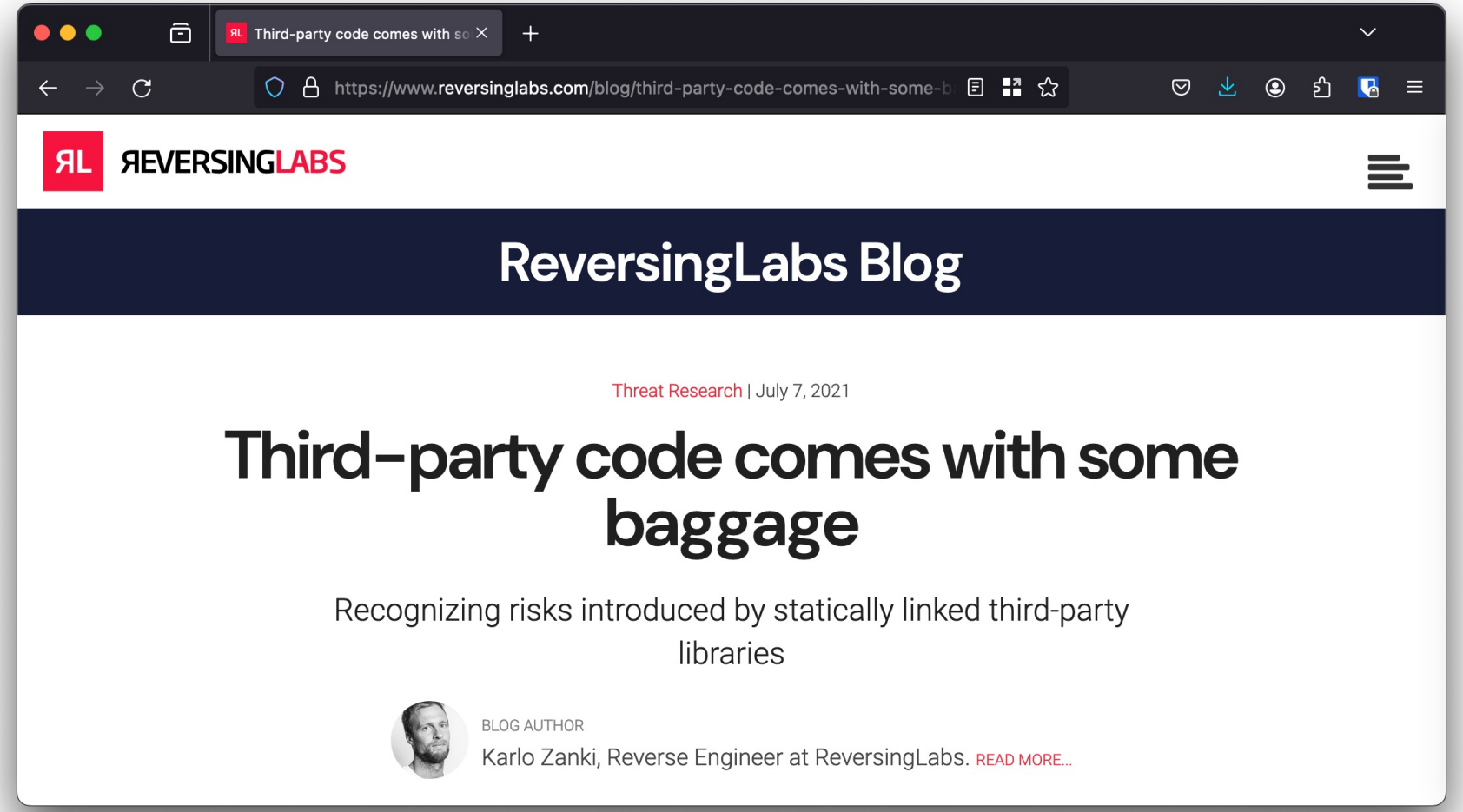
NPM Audit

```
                    === 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?

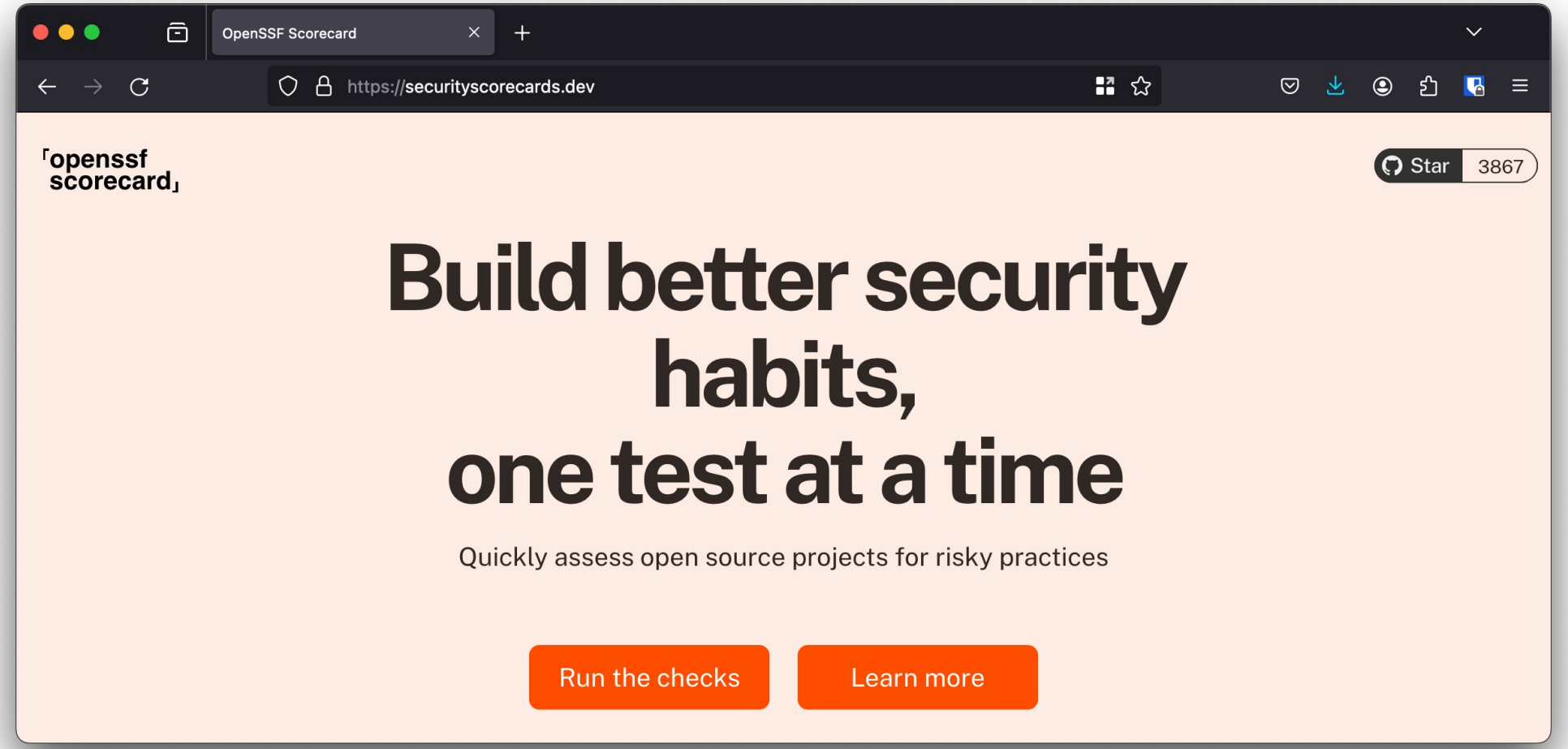


@nielstanis@infosec.exchange

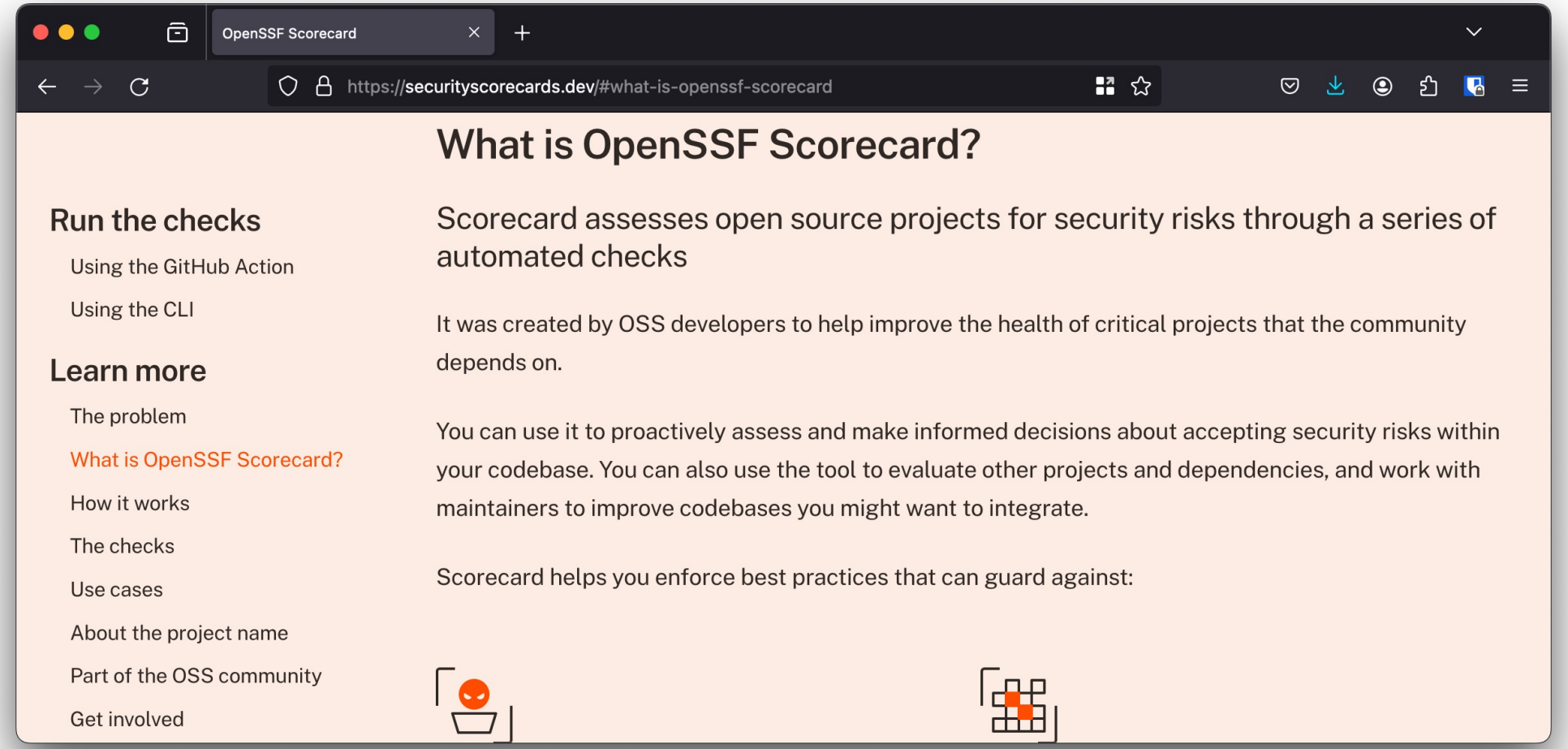
Nutrion Label for Software?

@nielstanis@infosec.exchange

OpenSSF Scorecards

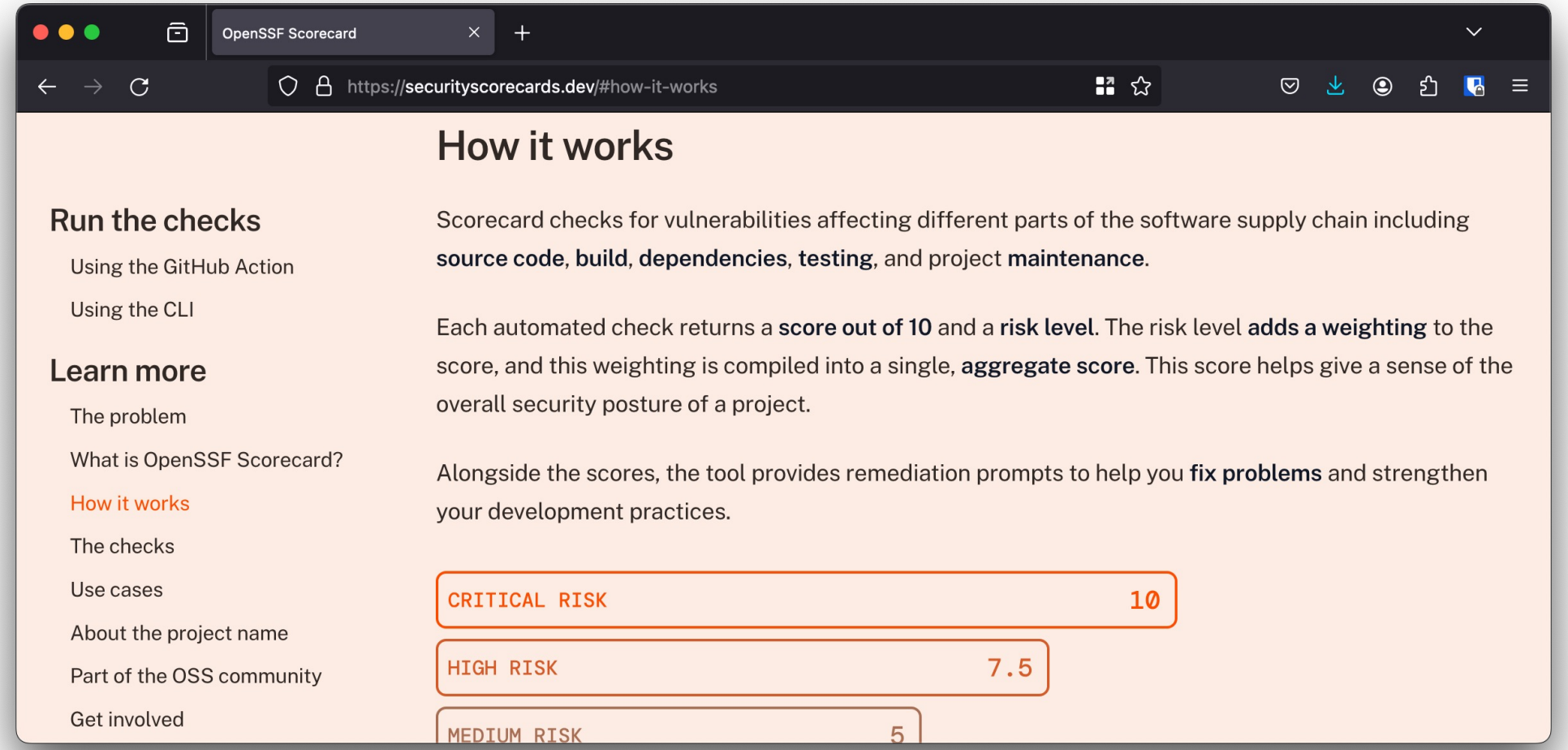


OpenSSF Security Scorecards



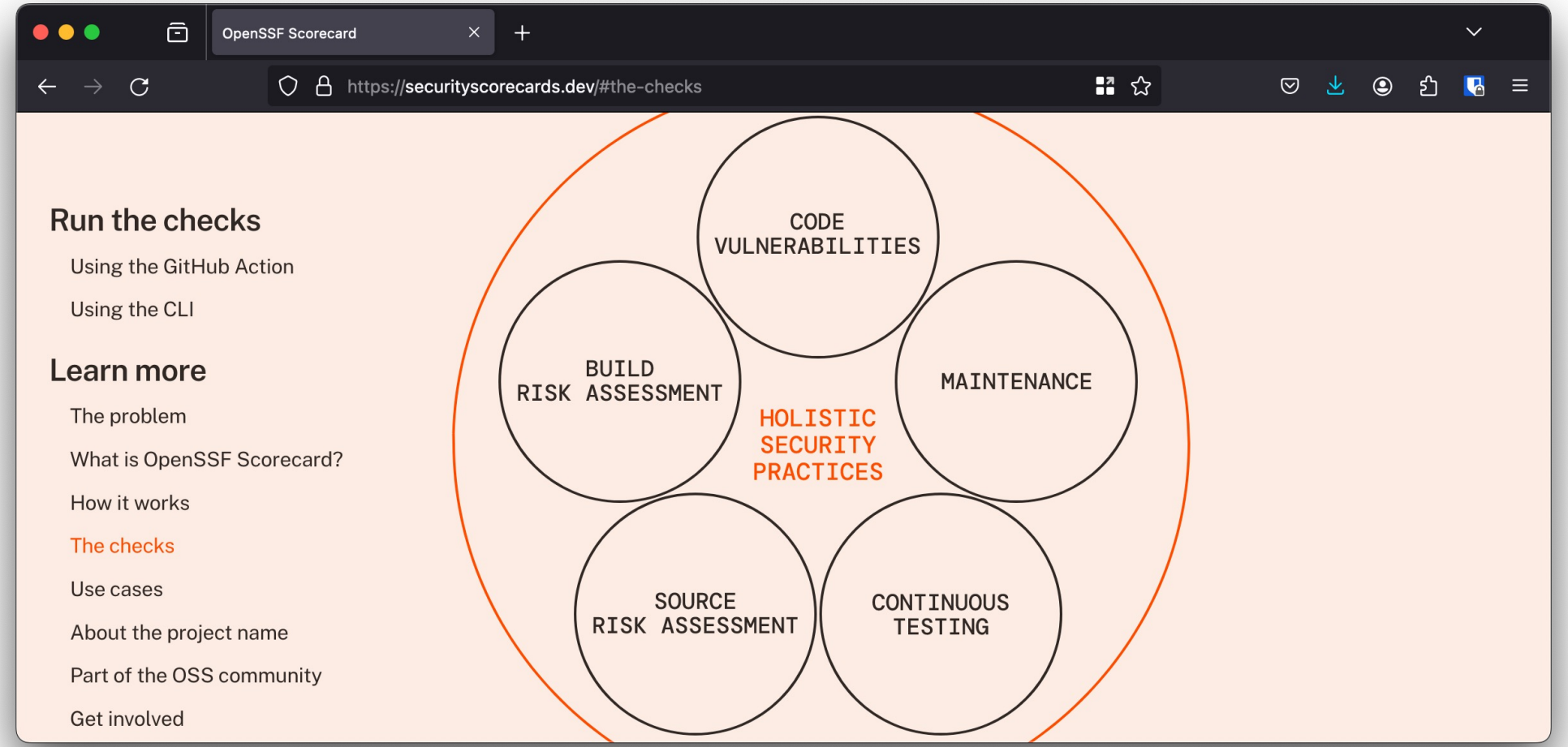
@nielstanis@infosec.exchange

OpenSSF Security Scorecards



@nielstanis@infosec.exchange

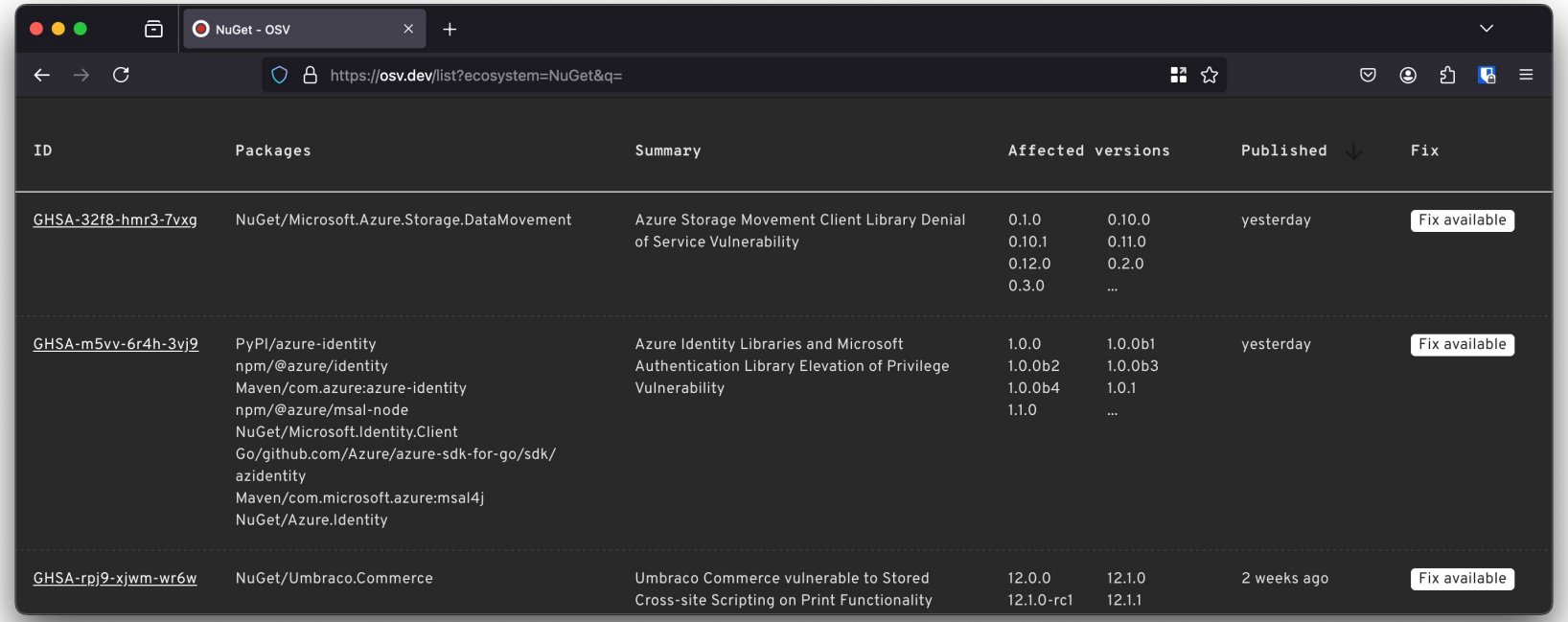
OpenSSF Security Scorecards



@nielstanis@infosec.exchange

Code Vulnerabilities (High)

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



ID	Packages	Summary	Affected versions		Published	Fix
GHSA-32f8-hmr3-7vxg	NuGet/Microsoft.Azure.Storage.DataMovement	Azure Storage Movement Client Library Denial of Service Vulnerability	0.1.0 0.10.1 0.12.0 0.3.0	0.10.0 0.11.0 0.2.0 ...	yesterday	Fix available
GHSA-m5vv-6r4h-3vj9	PyPI/azure-identity npm/@azure/identity Maven/com.azure:azure-identity npm/@azure/msal-node NuGet/Microsoft.Identity.Client Go/github.com/Azure/azure-sdk-for-go/sdk/azidentity Maven/com.microsoft.azure:msal4j NuGet/Azure.Identity	Azure Identity Libraries and Microsoft Authentication Library Elevation of Privilege Vulnerability	1.0.0 1.0.0b2 1.0.0b4 1.1.0	1.0.0b1 1.0.0b3 1.0.1 ...	yesterday	Fix available
GHSA-rpj9-xjwm-wr6w	NuGet/Umbraco.Commerce	Umbraco Commerce vulnerable to Stored Cross-site Scripting on Print Functionality	12.0.0 12.1.0-rc1	12.1.0 12.1.1	2 weeks ago	Fix available

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

@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;
- Does it make sense to do fuzzing on managed languages like Java and/or .NET?

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
- Definitely room for improvement!

Source Risk Assesement 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 Assesement 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 Assesement 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/

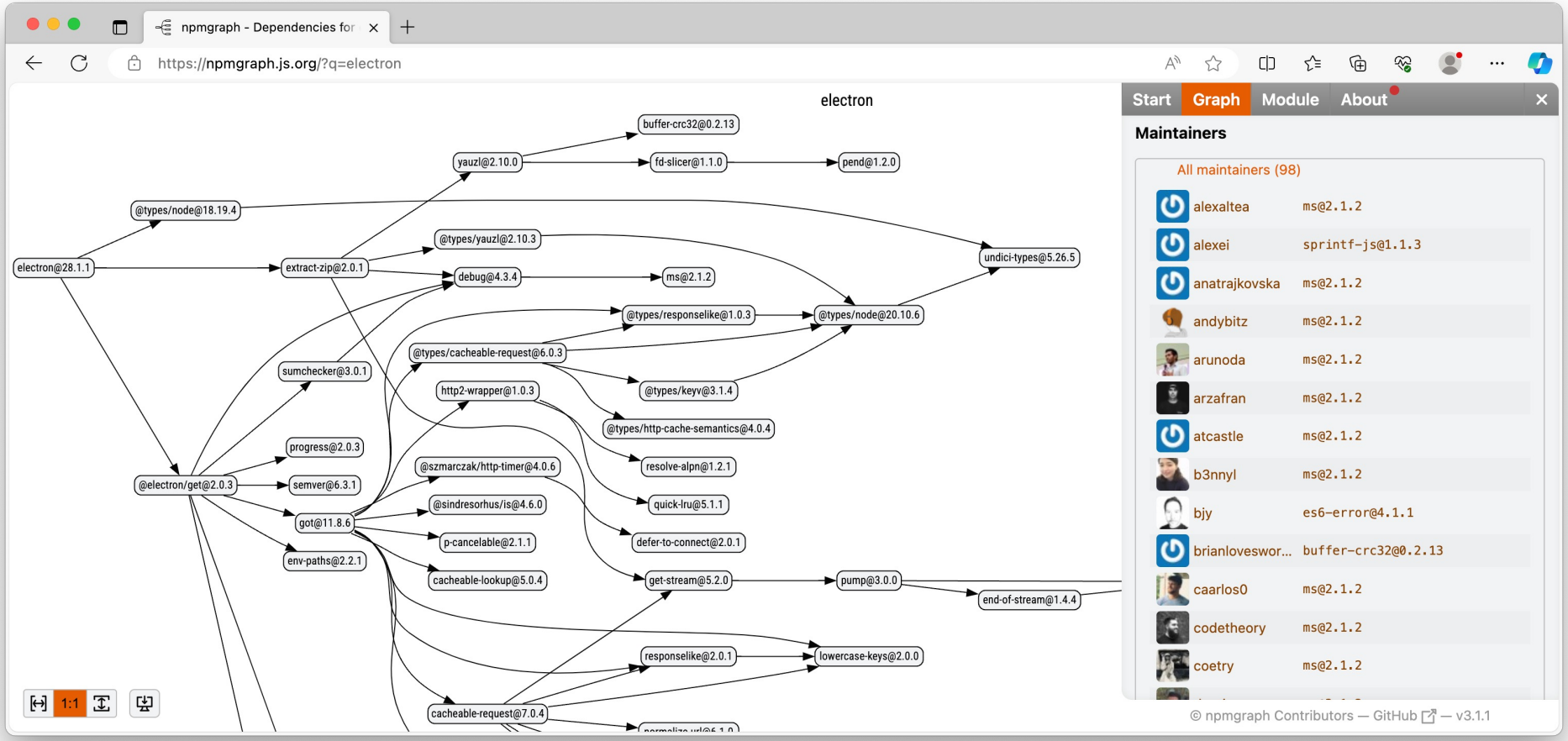
Source Risk Assesement
Code Review (Low)

- 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 Assesement 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 Assesement Contributors (Low)



@nielstanis@infosec.exchange

Build Risk Assesement
Pinned Dependencies (High)

- Does the project pin dependencies used during its build and release process.
- For .NET → **RestorePackagesWithLockFile** in MSBuild results in **packages.lock.json** file containing versioned dependency tree with hashes
- If Workflow is present what about the Actions used?

Build Risk Assesement
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 Assesement 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

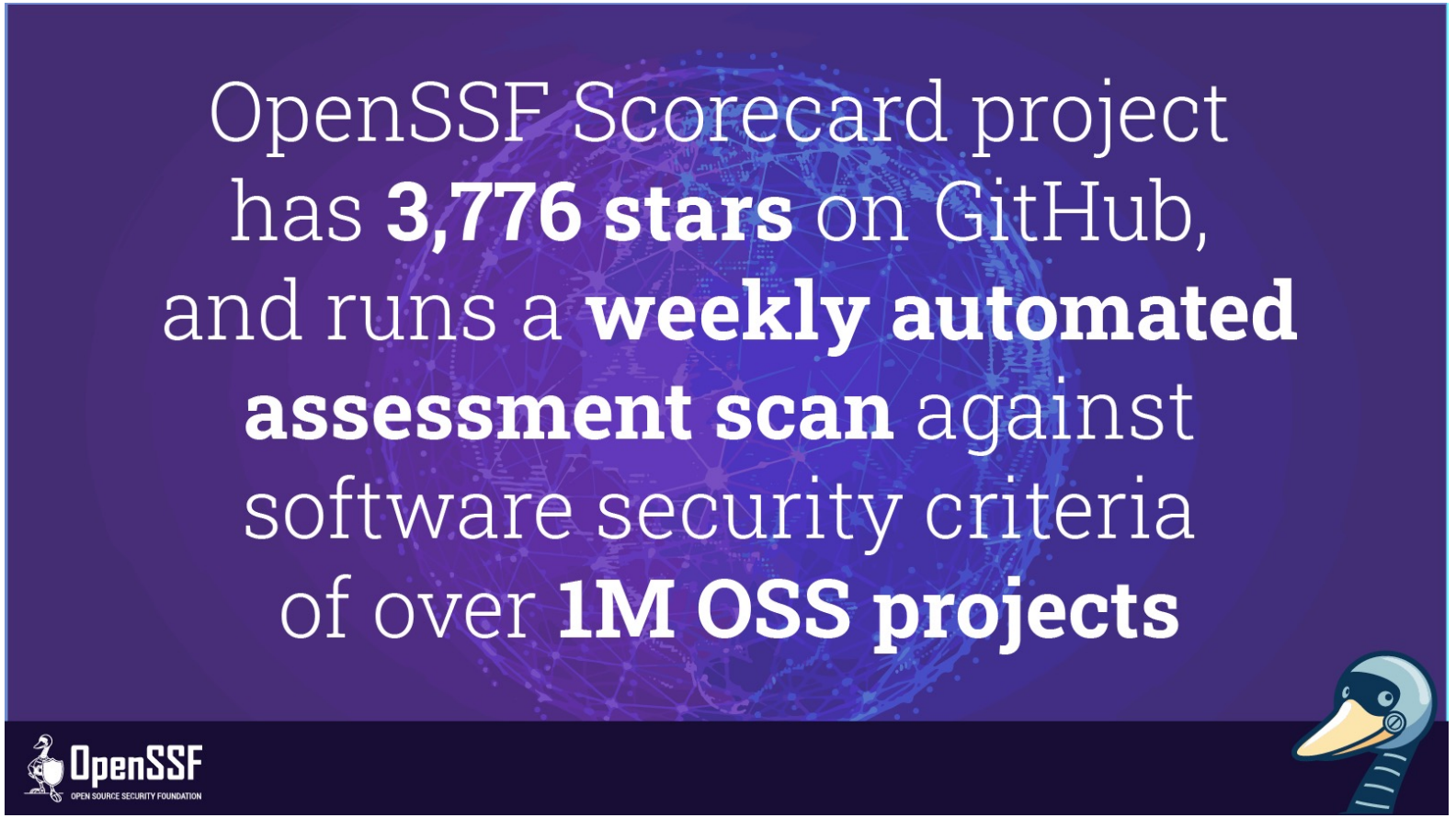
Demo OpenSSF Scorecard

```
Running checks
```

@nielstanis@infosec.exchange

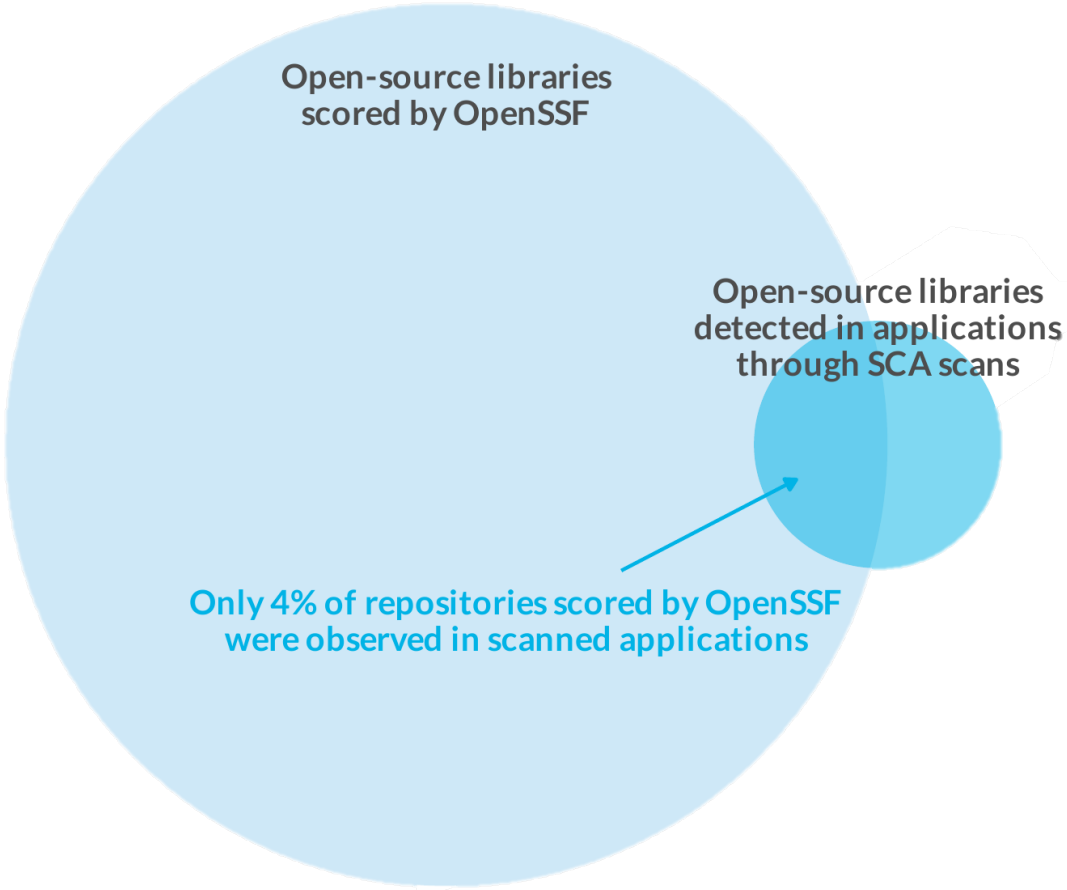
Measure?

OpenSSF Annual Report 2023

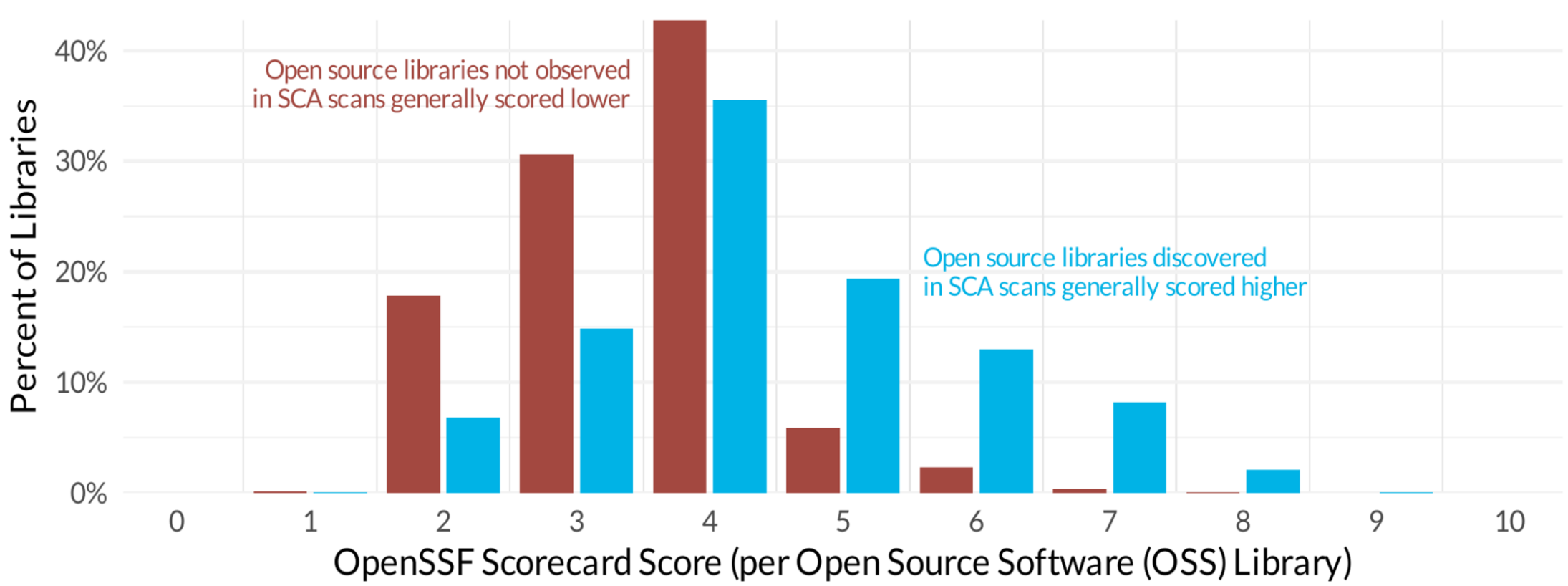


SOSS & OpenSSF Scorecard

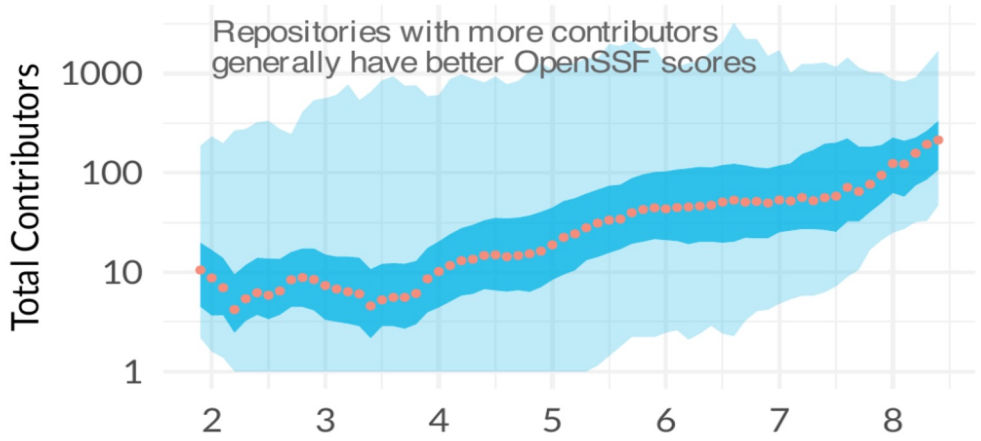
SOSS & OpenSSF Scorecard

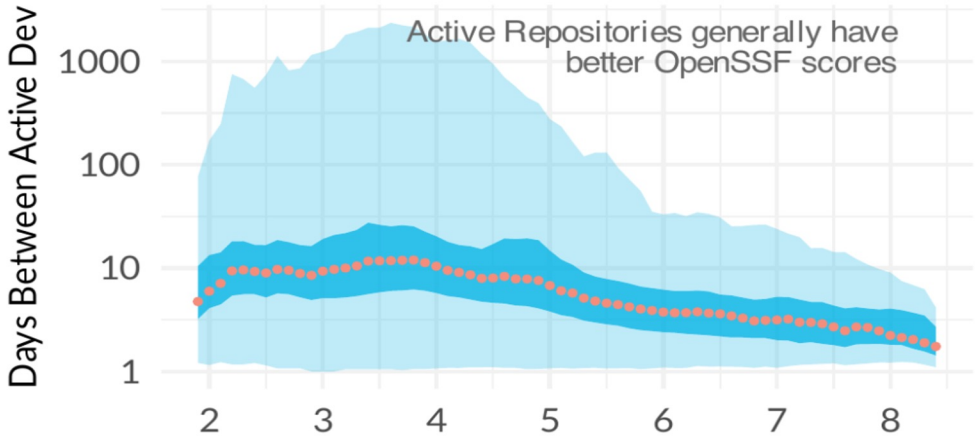


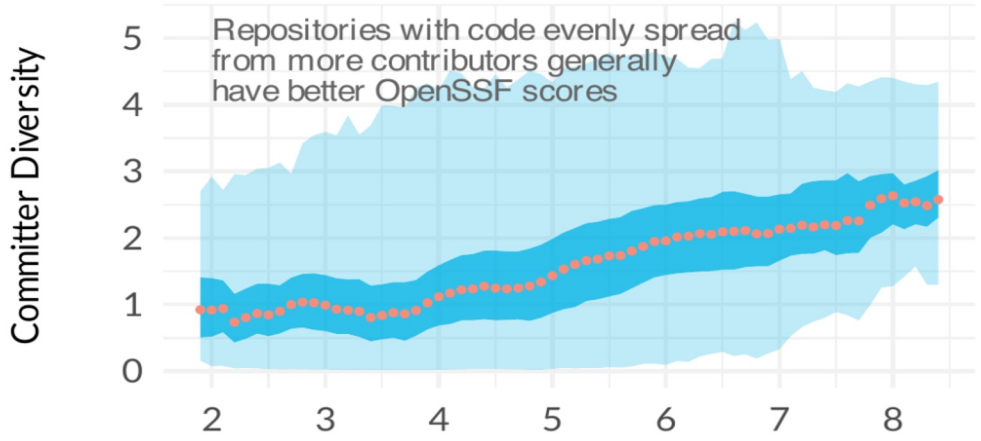
Correlation between SOSS

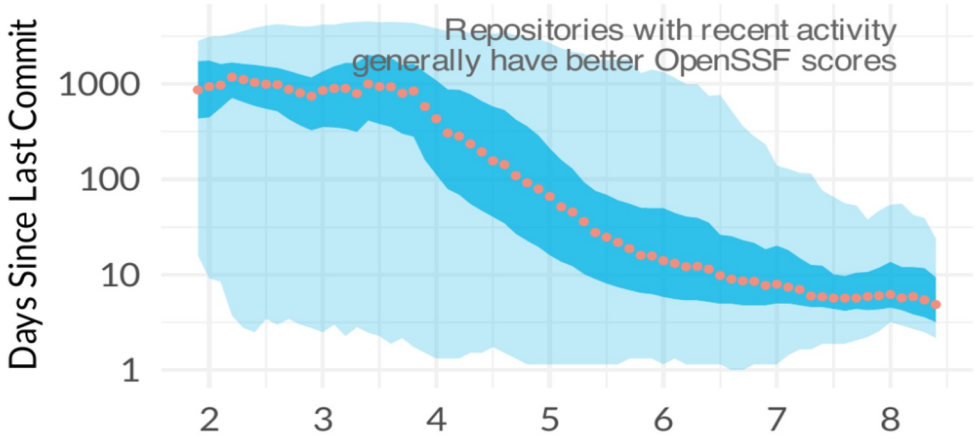


Github commits vs OpenSSF



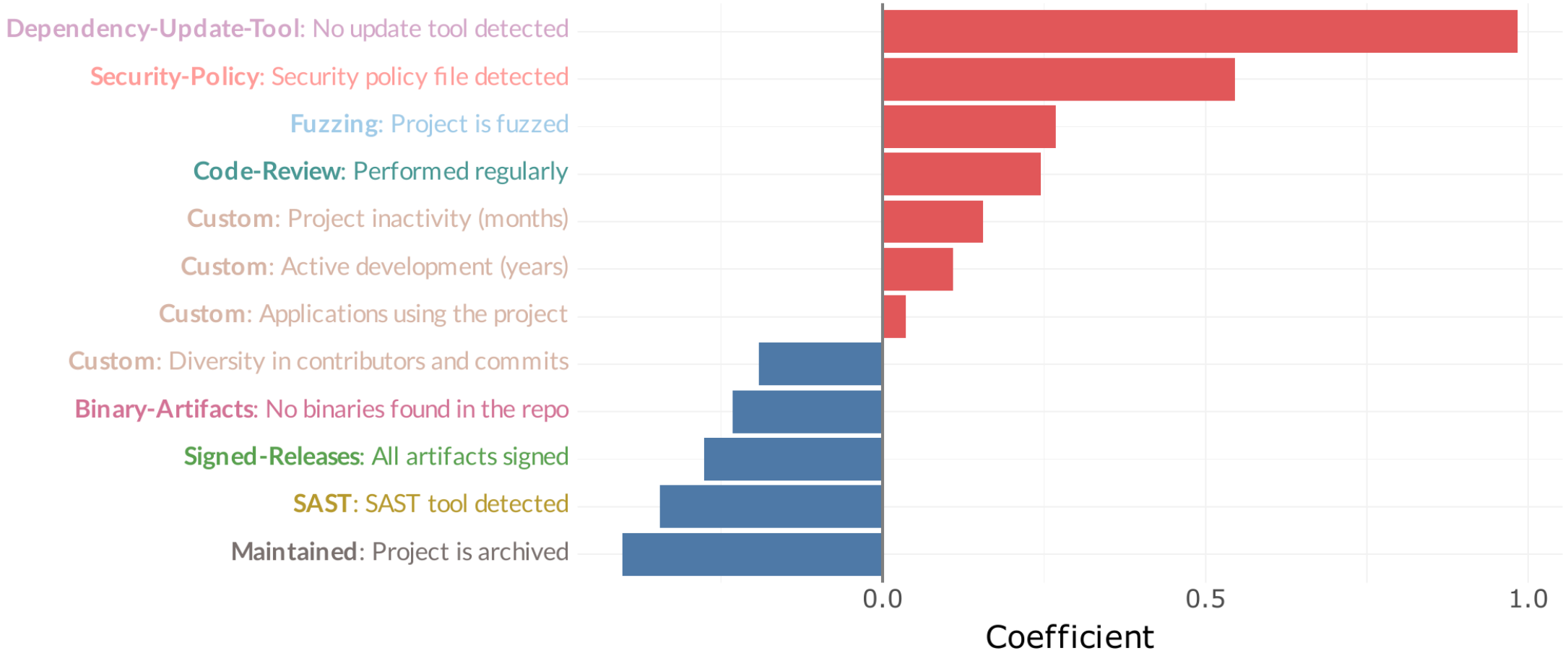






@nielstanis@infosec.exchange

What really contributes to OSS Security?



@nielstanis@infosec.exchange

What can we improve?

Fuzzing

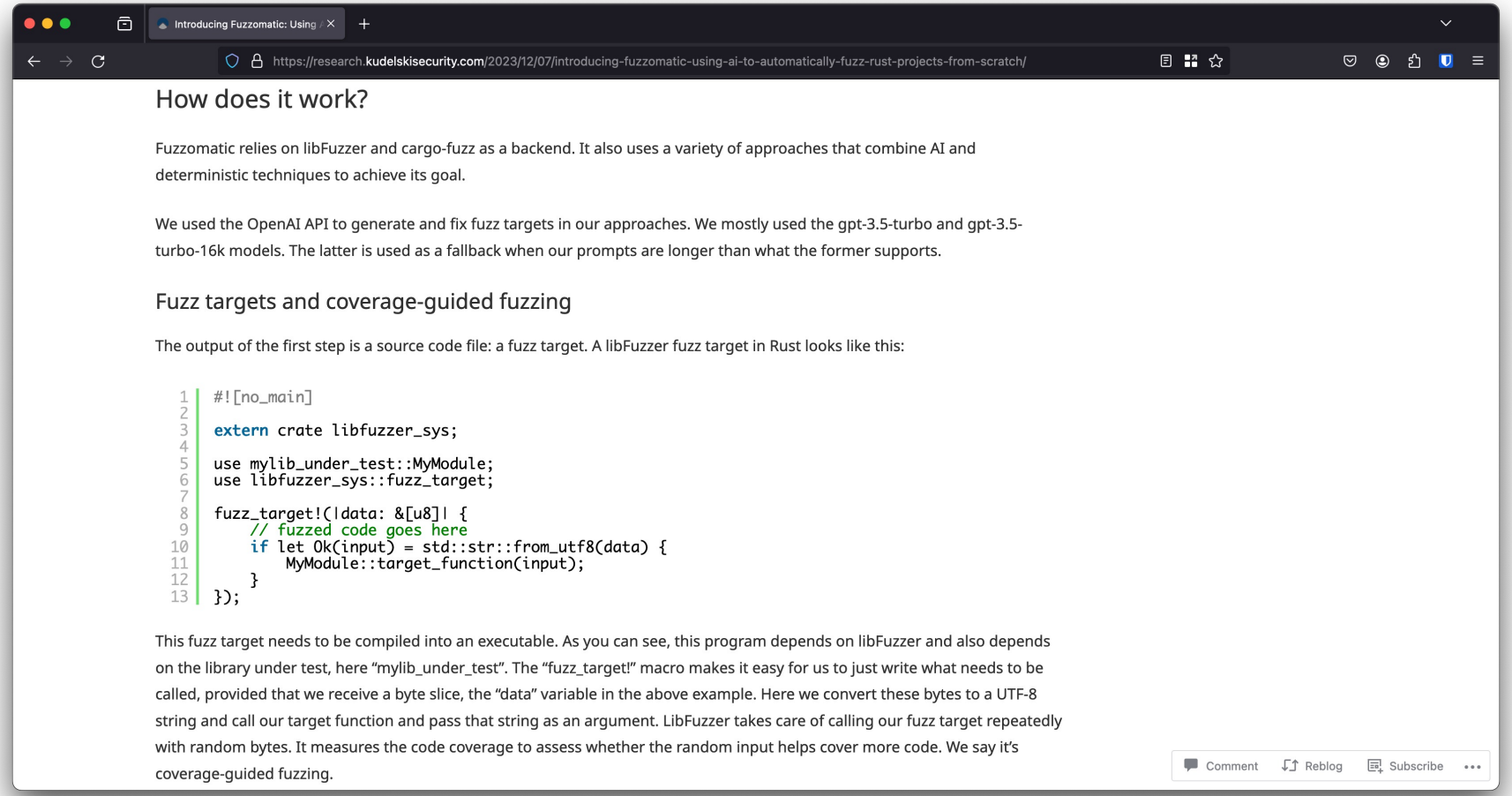
- Fuzzing, or fuzz testing
  - Automated software testing method that uses a wide range of ***invalid*** and unexpected data as input to find flaws
- Definitely good for finding C/C++ memory issues
- Can it be of any value with managed languages like .NET and/or Java?

Fuzzing .NET – Jil JSON Serializer


```
public static void Main(string[] args)
{
        SharpFuzz.Fuzzer.OutOfProcess.Run(stream => {
        try
        {
                using (var reader = new System.IO.StreamReader(stream))
                JSON.DeserializeDynamic(reader);
        }
        catch (DeserializationException) { }
        });
}
```

@nielstanis@infosec.exchange

Fuzzomatic: Using AI to Fuzz Rust



@nielstanis@infosec.exchange

Static Code Analysis (SAST)


```
public byte[] CreateHash(string password)
{
    var b = Encoding.UTF8.GetBytes(password);
    return SHA1.HashData(b);
}
```

Static Code Analysis (SAST)


```
public class CustomerController : Controller
{
    public IActionResult GenerateCustomerReport(string customerID)
    {
        var data = Reporting.GenerateCustomerReportOverview(customerID)
        return View(data);
    }
}

public static class Reporting
{
    public static byte[] GenerateCustomerReportOverview(string ID)
    {
        return System.IO.File.ReadAllBytes($"./data/{ID}.pdf");
    }
}
```

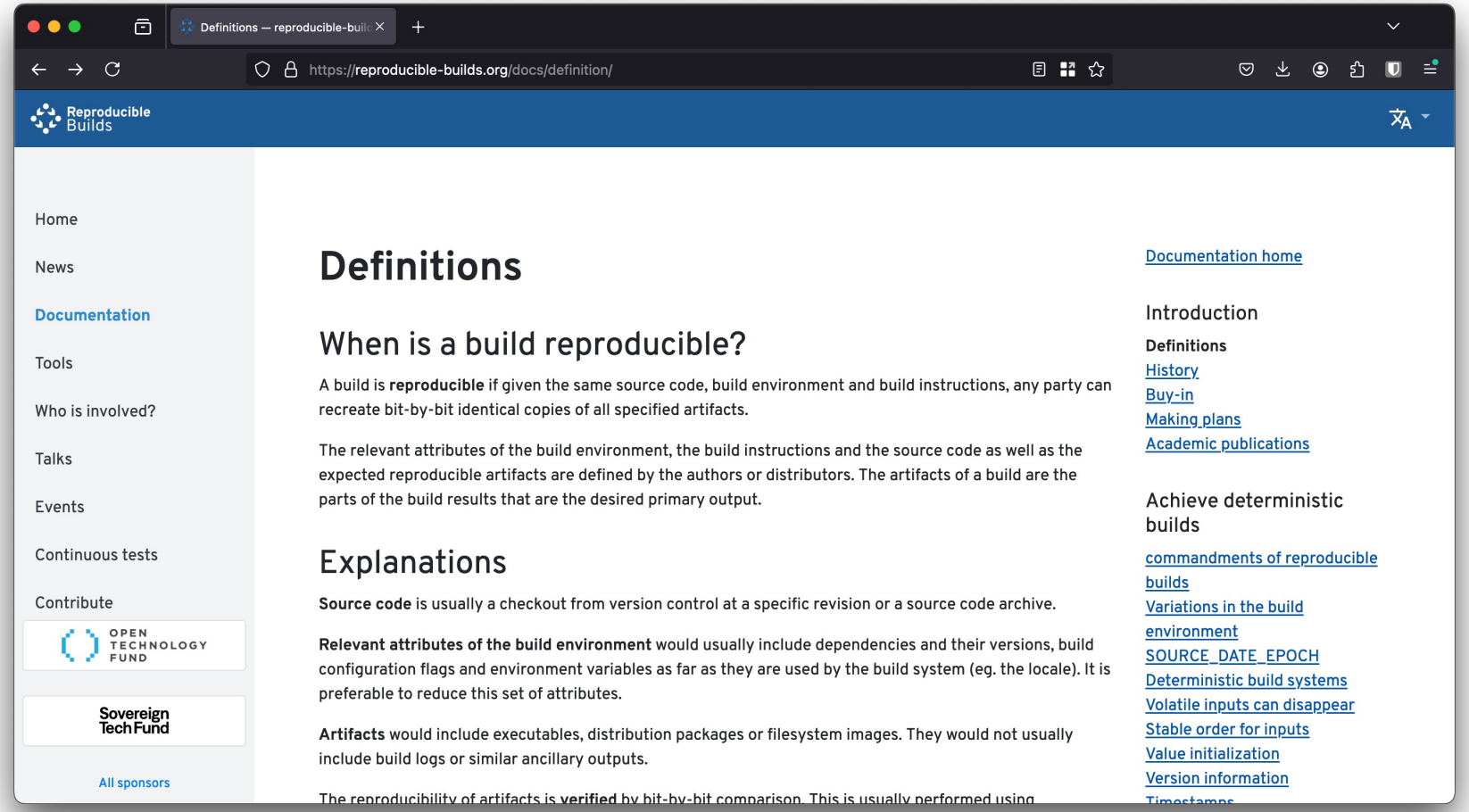
@nielstanis@infosec.exchange

Package Reproducibility

- A build is **reproducible** if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts.

Reproducible-Builds.org

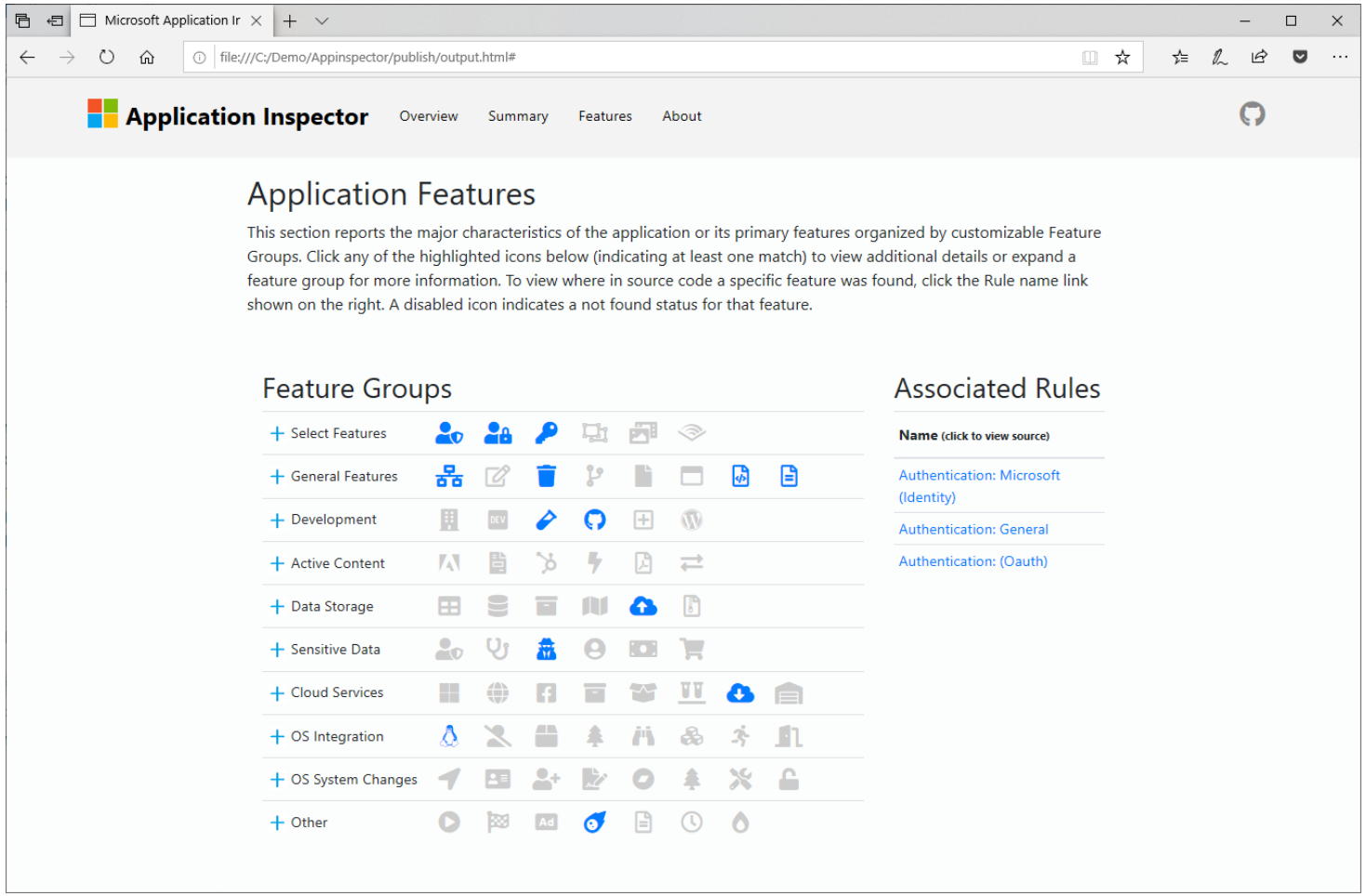
New & Improved!



@nielstanis@infosec.exchange

Application Inspector

New & Improved!



@nielstanis@infosec.exchange

Application Inspector

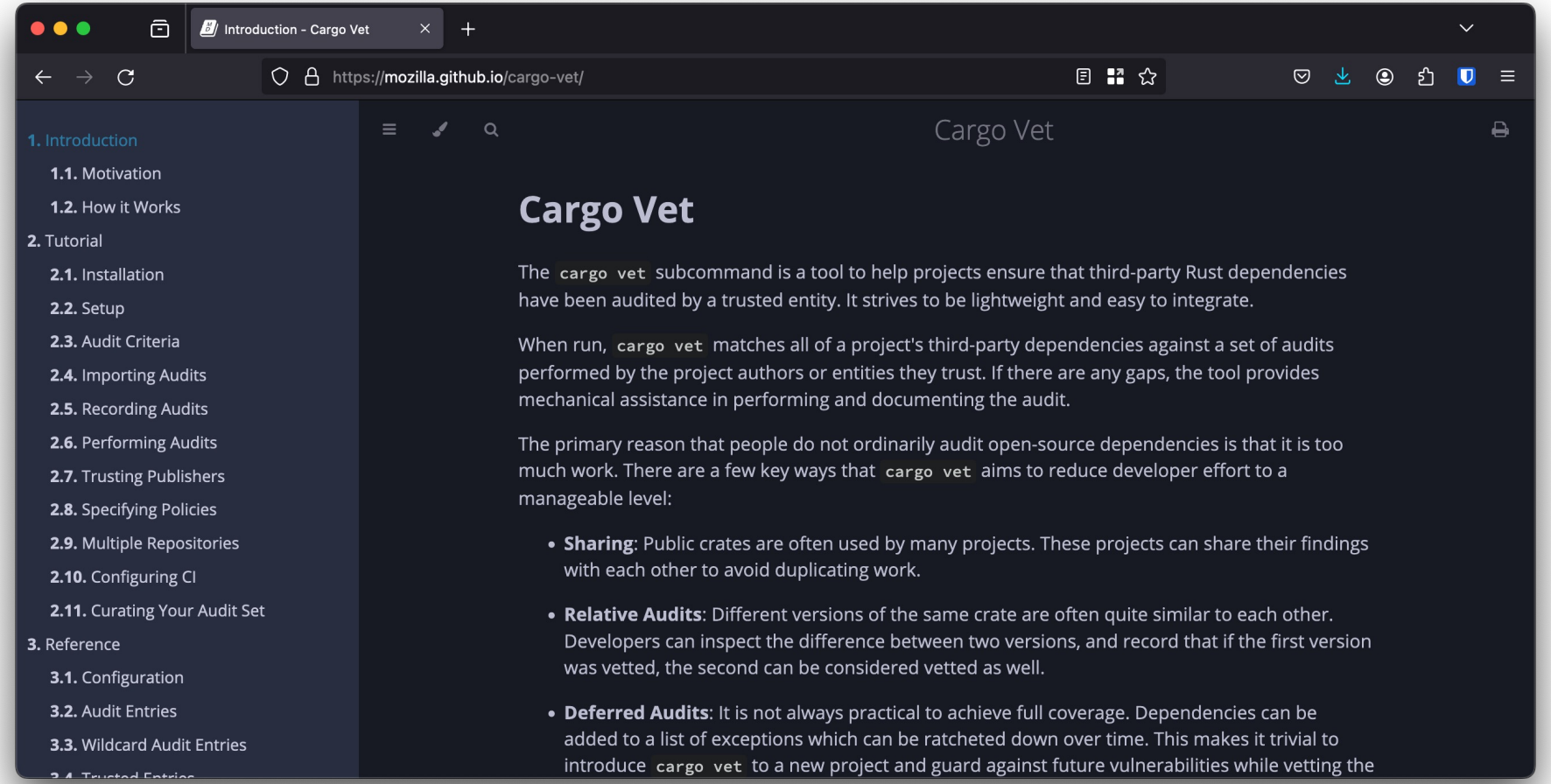
Select Features

Feature	Confidence	Details
Authentication		View
Authorization		View
Cryptography		View
Object Deserialization		N/A
AV Media Parsing		N/A
Dynamic Command Execution		N/A

@nielstanis@infosec.exchange

Community Review

New & Improved!



@nielstanis@infosec.exchange

Conclusion

- Scorecard helps security reviewing a used Package
- Better understand what's inside, how it’s build/maintained and what are the risks!
- Scorecard should not be a goal on it’s own!

Conclusion

- Room for improvements
  - Reproduciblity Tools (diff, insights)
  - Reporting
  - Trust Graph
  - Contribute back to OpenSSF Scorecard

Questions?

@nielstanis@infosec.exchange

Links

- https://github.com/nielstanis/wearedevs2024/
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- https://www.fennec.dev & https://blog.fennec.dev