

VERACODE

Reviewing 3rd Party Library Security Easily Using OpenSSF Scorecard

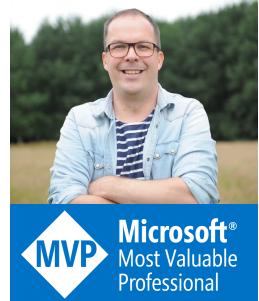
Niels Tanis
Sr. Principal Security Researcher



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



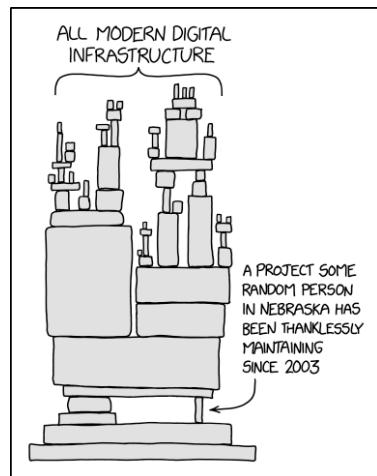
MVP Microsoft®
Most Valuable
Professional

W>

 @nielstanis@infosec.exchange

Modern Application Architecture XKCD 2347

W>



 @nielstanis@infosec.exchange

<https://xkcd.com/2347/>

Agenda

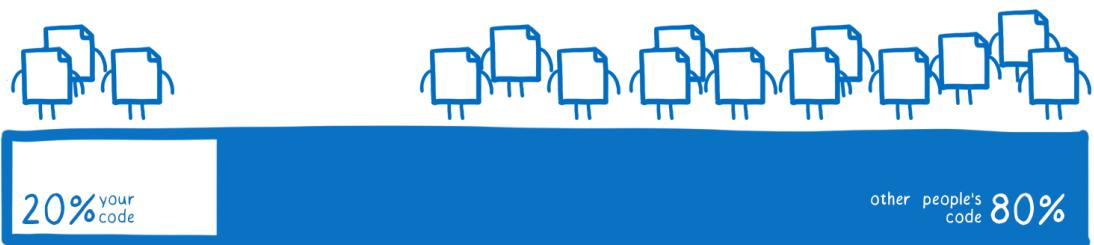
- Risks in 3rd Party Packages
- OpenSFF Scorecard
- Measure, New & Improved
- Conclusion - Q&A

W>



 @nielstanis@infosec.exchange

Average codebase composition



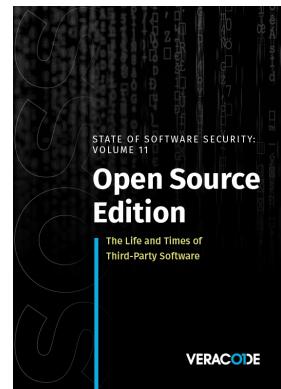
W>

 @nielstanis@infosec.exchange

<https://hacks.mozilla.org/2019/11/announcing-the-bytecode-alliance/>

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."



W>

@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.

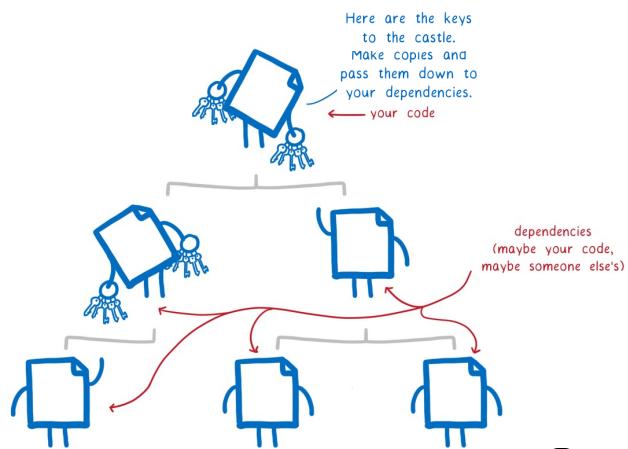


 @nielstanis@infosec.exchange

<https://www.veracode.com/blog/research/state-log4j-vulnerabilities-how-much-did-log4shell-change>

Average codebase composition

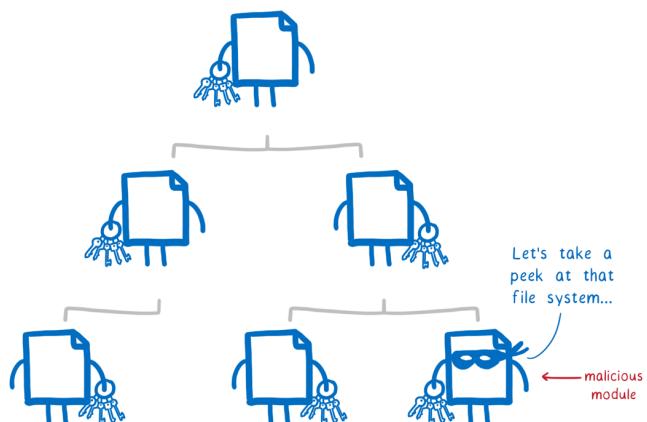
W>



@nielstanis@infosec.exchange

<https://hacks.mozilla.org/2019/11/announcing-the-bytecode-alliance/>

Malicious Library



 @nielstanis@infosec.exchange

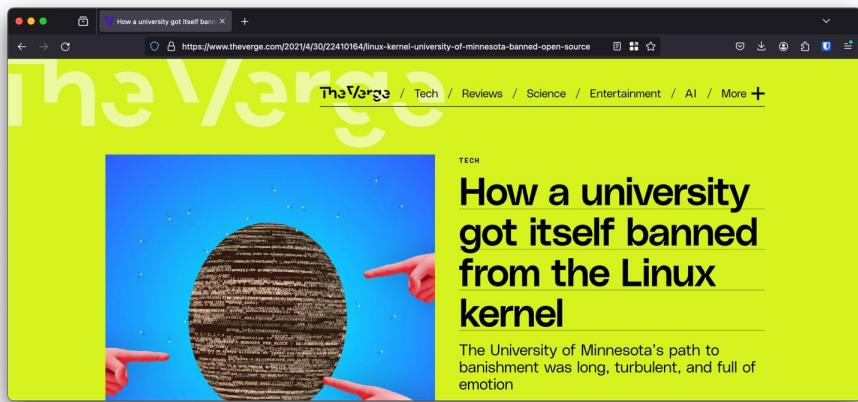
<https://hacks.mozilla.org/2019/11/announcing-the-bytecode-alliance/>

Malicious Package

The screenshot shows a web browser window displaying a news article from The Hacker News. The title of the article is "48 Malicious npm Packages Found Deploying Reverse Shells on Developer Systems". The date is Nov 03, 2023, and the category is Software Security / Malware. Below the title is a graphic of a glowing orange horse head inside a hexagon, set against a background of binary code. A subtitle below the graphic reads: "A new set of 48 malicious npm packages have been discovered in the npm repository with capabilities to deploy a reverse shell on compromised systems." At the bottom right of the article area, there is a social media link for @nielstanis@infosec.exchange.

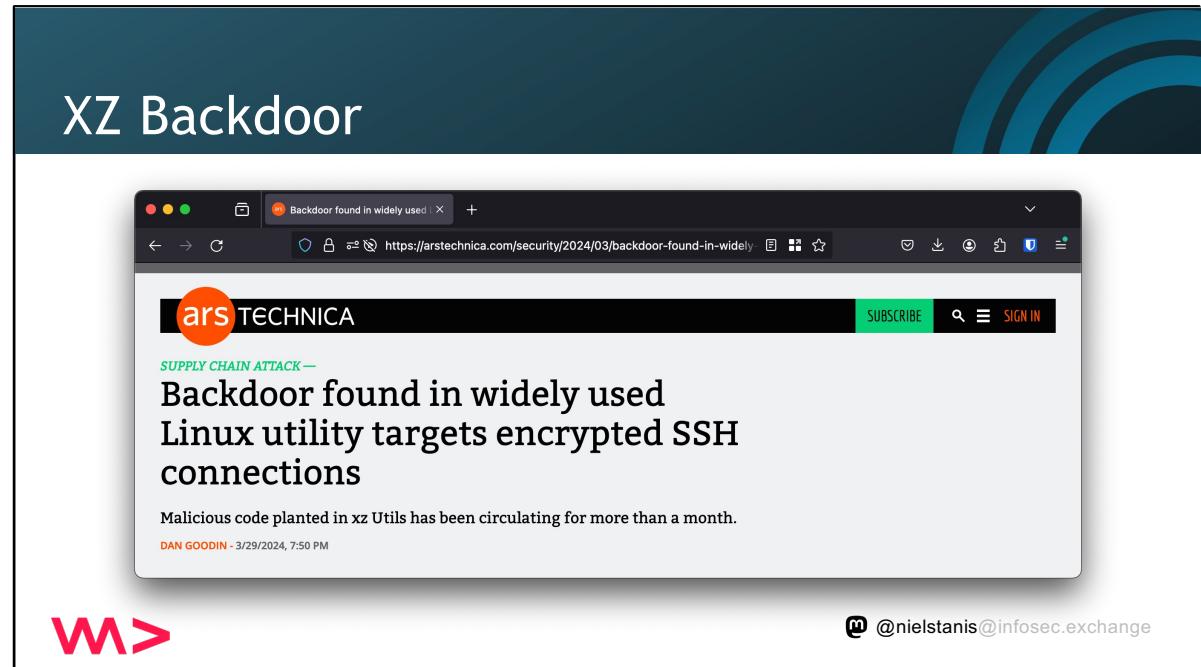
<https://thehackernews.com/2023/11/48-malicious-npm-packages-found.html>

Hypocrite Commits



W>

<https://www.theverge.com/2021/4/30/22410164/linux-kernel-university-of-minnesota-banned-open-source>

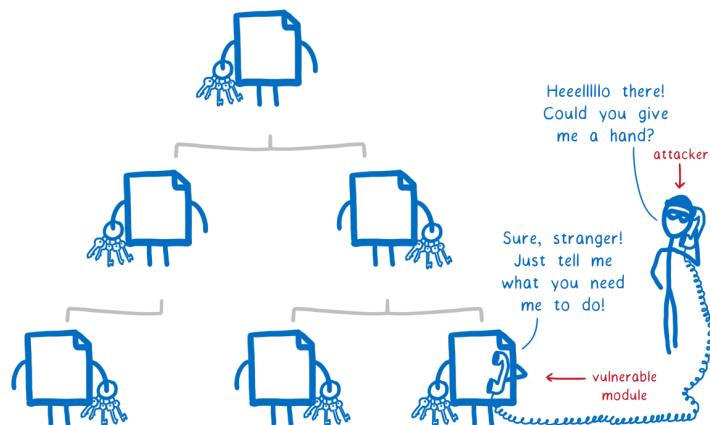


W>

@nielstanis@infosec.exchange

<https://arstechnica.com/security/2024/03/backdoor-found-in-widely-used-linux-utility-breaks-encrypted-ssh-connections/>

Vulnerable Package



@nielstanis@infosec.exchange

<https://hacks.mozilla.org/2019/11/announcing-the-bytecode-alliance/>

W>

Vulnerabilities in Libraries

The screenshot shows a GitHub issue page for Microsoft Security Advisory CVE-2023-36558: .NET Security Feature Bypass Vulnerability. The page title is "Microsoft Security Advisory CVE-2023-36558: .NET Security Feature Bypass Vulnerability #288". The main content area contains the advisory text, which states: "Microsoft is releasing this security advisory to provide information about a vulnerability in ASP.NET Core 6.0, ASP.NET Core 7.0 and, ASP.NET Core 8.0 RC2. This advisory also provides guidance on what developers can do to update their applications to address this vulnerability. A security feature bypass vulnerability exists in ASP.NET where an unauthenticated user is able to bypass validation on Blazor server forms which could trigger unintended actions." Below the text is a "Discussion" section with a link to "dotnet/runtime#94726". On the right side, there is a sidebar with project details: Assignees (None assigned), Labels (Security), Projects (None yet), Milestone (No milestone), Development (No branches or pull requests), and Notifications (Customize).

W>

@nielstanis@infosec.exchange

<https://github.com/dotnet/announcements/issues/288>

DotNet CLI

```
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 $
```



 @nielstanis@infosec.exchange

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



 @nielstanis@infosec.exchange

Do you know what's inside?

The screenshot shows a web browser window with a dark blue header. The main content area has a white background. At the top left of the content area is the ReversingLabs logo. Below the logo is a dark blue navigation bar with the text "ReversingLabs Blog". Underneath the navigation bar, the title of the blog post is displayed in large, bold, black font: "Third-party code comes with some baggage". Below the title, a subtitle reads "Recognizing risks introduced by statically linked third-party libraries". A small profile picture of a man is shown next to the author's name, Karlo Zanki, Reverse Engineer at ReversingLabs. There is also a "READ MORE..." link. At the bottom right of the content area, there is a red "W>" logo. In the top right corner of the browser window, there is a watermark-like graphic of blue concentric arcs.

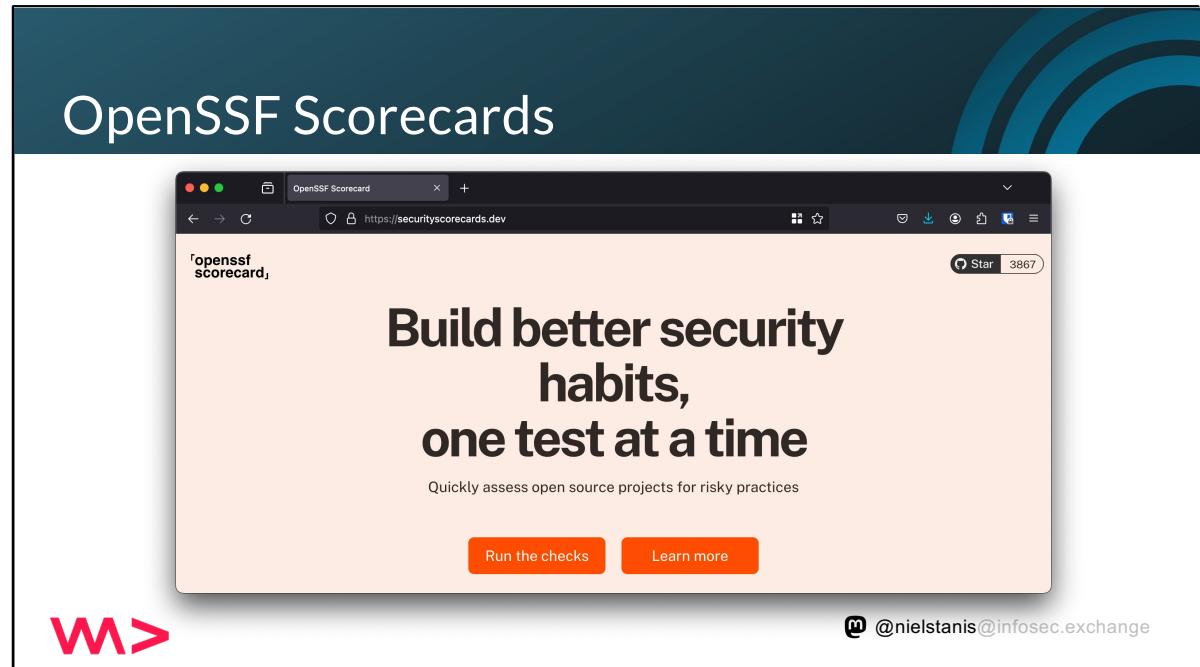
<https://www.reversinglabs.com/blog/third-party-code-comes-with-some-baggage>

Nutrition Label for Software?



<https://securityscorecards.dev/>

@nielstanis@infosec.exchange



<https://securityscorecards.dev/>

The screenshot shows a web browser window with the title "OpenSSF Security Scorecards". The main content area displays the "What is OpenSSF Scorecard?" page. On the left, there is a sidebar with sections for "Run the checks" (Using the GitHub Action, Using the CLI) and "Learn more" (The problem, What is OpenSSF Scorecard?, How it works, The checks, Use cases, About the project name, Part of the OSS community, Get involved). The main content area has a heading "What is OpenSSF Scorecard?" followed by a paragraph explaining what the scorecard does and its purpose. Below the text are two small icons: a red one with a white circle and a blue one with a grid pattern.

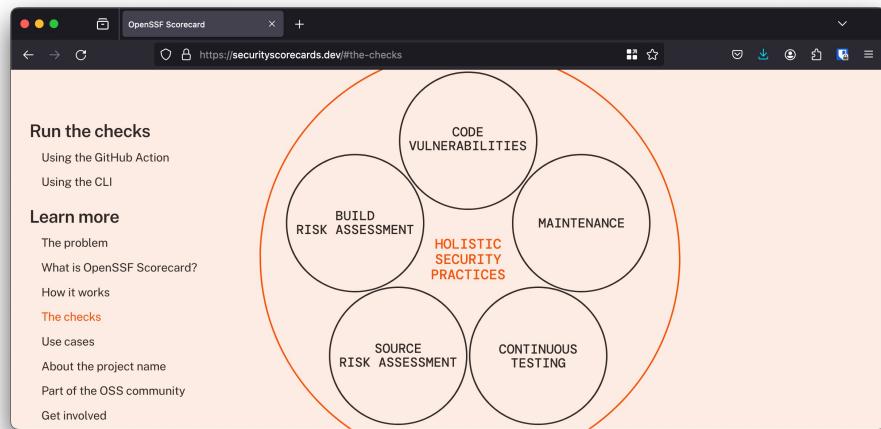
<https://securityscorecards.dev/>

OpenSSF Security Scorecards

The screenshot shows a web browser window titled "OpenSSF Scorecard" with the URL <https://securityscorecards.dev/#how-it-works>. The page has a dark blue header and a light orange main content area. The title "How it works" is at the top. On the left, there are two sections: "Run the checks" and "Learn more". "Run the checks" includes links for "Using the GitHub Action" and "Using the CLI". "Learn more" includes links for "The problem", "What is OpenSSF Scorecard?", "How it works" (which is highlighted in red), "The checks", "Use cases", "About the project name", "Part of the OSS community", and "Get involved". The main content area describes how the scorecard checks for vulnerabilities across the software supply chain and provides a weighted aggregate score. It also mentions remediation prompts. Below this, there are three horizontal bars representing risk levels: "CRITICAL RISK" (10), "HIGH RISK" (7.5), and "MEDIUM RISK" (5). In the bottom right corner, there is a small profile icon and the email address @nielstanis@infosec.exchange.

<https://securityscorecards.dev/>

OpenSSF Security Scorecards

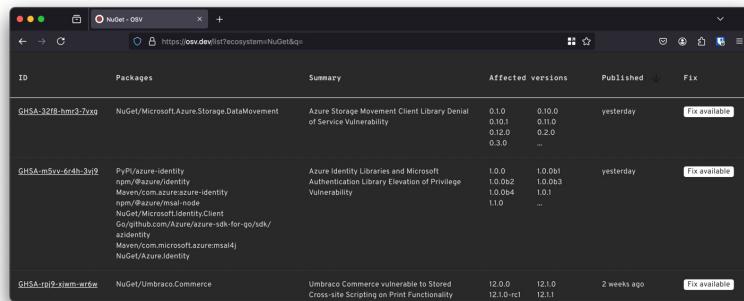


@nielstanis@infosec.exchange

<https://securityscorecards.dev/>

Code Vulnerabilities (High)

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



The screenshot shows a table of code vulnerabilities for the NuGet ecosystem. The columns are ID, Packages, Summary, Affected versions, Published, and Fix. There are three rows of data:

ID	Packages	Summary	Affected versions	Published	Fix
GHSA-32fb-hm:3-7vxp	NuGet/Microsoft.Azure.Storage.DataMovement	Azure Storage Movement Client Library Denial of Service Vulnerability	0.1.0 0.10.1 0.11.0 0.12.0 0.2.0 0.3.0 —	yesterday	Fix available
GHSA-m5cv-6rdh-3yf9	PIP: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/msal4 NuGet/Azure.Identity	Azure Identity Libraries and Microsoft Authentication Library Elevation of Privilege Vulnerability	1.0.0 1.0.0b2 1.0.0b3 1.0.0b4 1.0.1 1.1.0 —	yesterday	Fix available
GHSA-rq9-xiwm-wrfw	NuGet/Umbraco.Commerce	Umbraco Commerce vulnerable to Stored Cross-site Scripting on Print Functionality	12.0.0 12.1.0 12.1.0-re1 12.1.1	2 weeks ago	Fix available



 @nielstanis@infosec.exchange

<https://osv.dev/list?ecosystem=NuGet>

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.

W>

 @nielstanis@infosec.exchange

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.

W>

 @nielstanis@infosec.exchange

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.

W>

 @nielstanis@infosec.exchange

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

<https://www.bestpractices.dev/en/criteria/0>

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).

W>

 @nielstanis@infosec.exchange

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?

W>

 @nielstanis@infosec.exchange

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!

W>

 @nielstanis@infosec.exchange

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!**

W>

 @nielstanis@infosec.exchange

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!

W>

 @nielstanis@infosec.exchange

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/>

W>

 @nielstanis@infosec.exchange

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)

W>

 @nielstanis@infosec.exchange

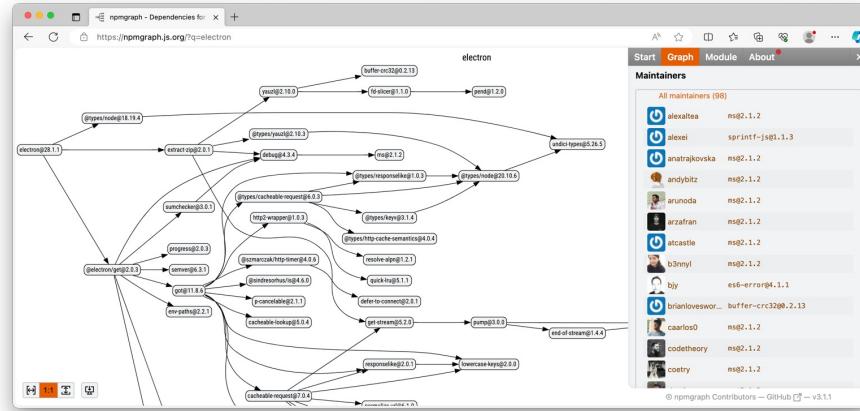
Source Risk Assessment Contributors (Low)

- 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?

W>

 @nielstanis@infosec.exchange

Source Risk Assessment Contributors (Low)



W>

@nielstanis@infosec.exchange

Build Risk Assessment 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?

W>

 @nielstanis@infosec.exchange

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.

W>

 @nielstanis@infosec.exchange

<https://securitylab.github.com/research/github-actions-preventing-pwn-requests/>

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.

W>

 @nielstanis@infosec.exchange

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

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

W>

 @nielstanis@infosec.exchange

Demo OpenSSF Scorecard

Running checks



W>

 @nielstanis@infosec.exchange

Measure?



W>

 @nielstanis@infosec.exchange

<https://www.bestpractices.dev/en/criteria/0>

OpenSSF Annual Report 2023

OpenSSF Scorecard project
has **3,776 stars** on GitHub,
and runs a **weekly automated**
assessment scan against
software security criteria
of over **1M OSS projects**



 @nielstanis@infosec.exchange

W>

<https://openssf.org/download-the-2023-openssf-annual-report/>

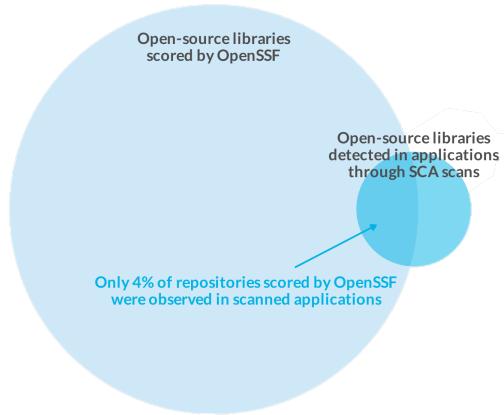
SOSS & OpenSSF Scorecard



@nielstanis@infosec.exchange

W>

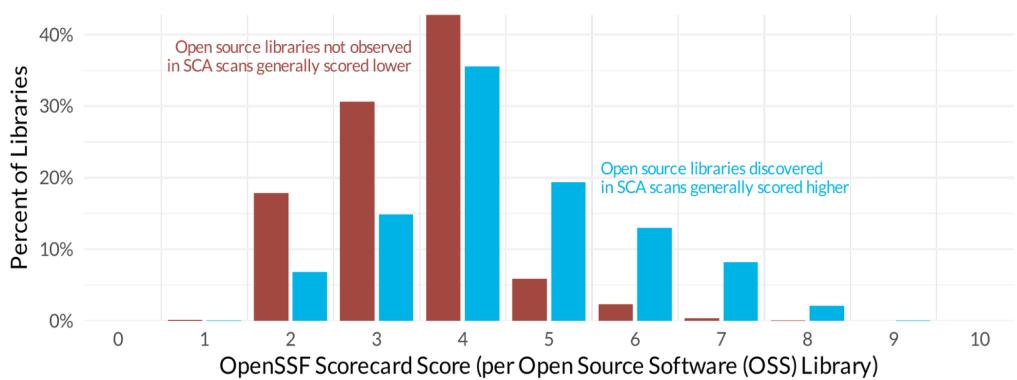
SOSS & OpenSSF Scorecard



 @nielstanis@infosec.exchange

<https://www.rsaconference.com/Library/presentation/usa/2024/quantifying%20the%20probability%20of%20flaws%20in%20open%20source>

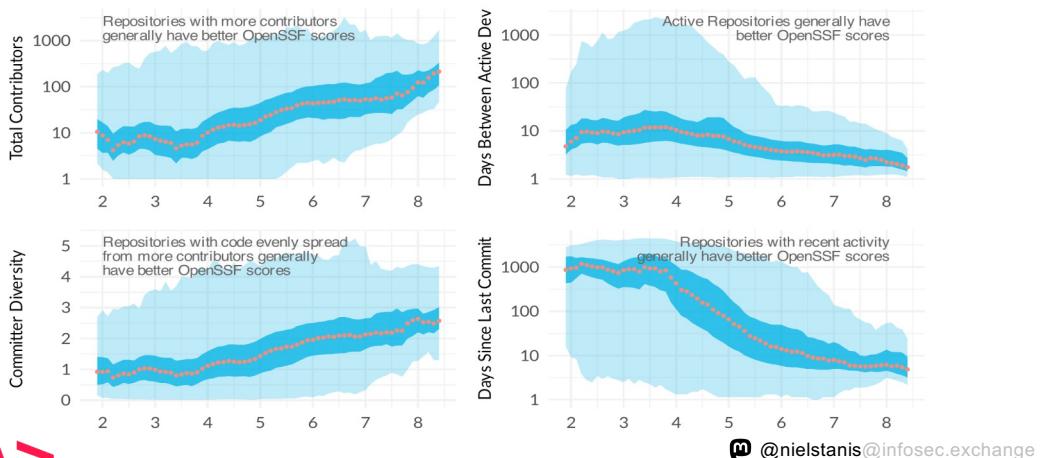
Correlation between SOSS



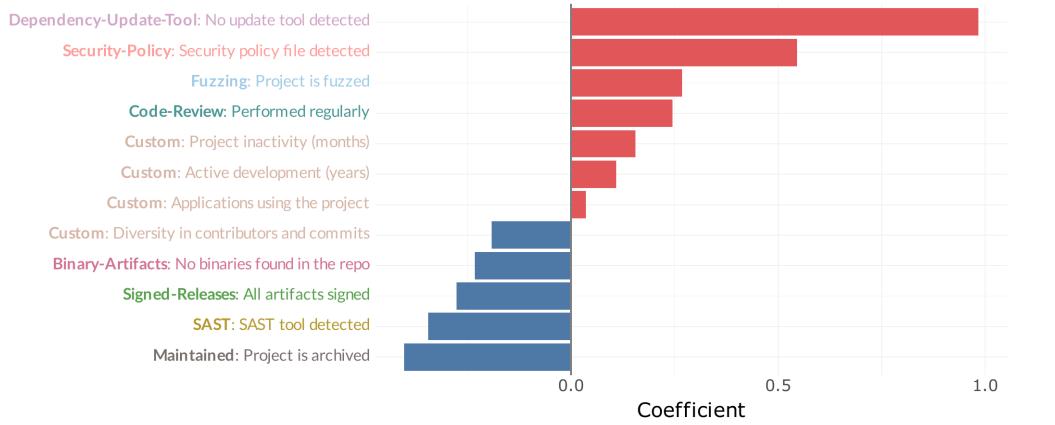
W>

@nielstanis@infosec.exchange

Github commits vs OpenSSF



What really contributes to OSS Security?



W>

@nielstanis@infosec.exchange

What can we improve?



W>

 @nielstanis@infosec.exchange

<https://www.bestpractices.dev/en/criteria/0>

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?

W>

 @nielstanis@infosec.exchange

<https://www.bestpractices.dev/en/criteria/0>

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) { }
    });
}
```

W>

 @nielstanis@infosec.exchange

<https://github.com/google/fuzzing/blob/master/docs/structure-aware-fuzzing.md>

Fuzzomatic: Using AI to Fuzz Rust

New & Improved!

How does it work?

Fuzzomatic relies on libFuzzer and cargo-fuzz as a backend. It also uses a variety of approaches that combine AI and deterministic techniques to achieve its goal.

We used the OpenAI API to generate and fix fuzz targets in our approaches. We mostly used the gpt-3.5-turbo and gpt-3.5-turbo-16k models. The latter is used as a fallback when our prompts are longer than what the former supports.

Fuzz targets and coverage-guided fuzzing

The output of the first step is a source code file: a fuzz target. A libFuzzer fuzz target in Rust looks like this:

```

1  #[no_main]
2  extern crate libfuzzer_sys;
3  use mylib_under_test::MyModule;
4  use libfuzzer_sys::fuzz_target;
5
6  fuzz_target!(data: [u8]) {
7      // Fuzzed code goes here
8      if let Ok(input) = std::str::from_utf8(data) {
9          MyModule::target_function(input);
10     }
11 }
12
13 }
```

This fuzz target needs to be compiled into an executable. As you can see, this program depends on libFuzzer and also depends on the library under test, here "mylib_under_test". The "fuzz_target!" macro makes it easy for us to just write what needs to be called, provided that we receive a byte slice, the "data" variable in the above example. Here we convert these bytes to a UTF-8 string and call our target function and pass that string as an argument. LibFuzzer takes care of calling our fuzz target repeatedly with random bytes. It measures the code coverage to assess whether the random input helps cover more code. We say it's coverage-guided fuzzing.

Comment | Reblog | Subscribe | ...

W>

@nielstanis@infosec.exchange

<https://research.kudelskisecurity.com/2023/12/07/introducing-fuzzomatic-using-ai-to-automatically-fuzz-rust-projects-from-scratch/>

Static Code Analysis (SAST)

New &
Improved!

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

W>

 @nielstanis@infosec.exchange

<https://www.bestpractices.dev/en/criteria/0>

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

<https://www.bestpractices.dev/en/criteria/0>

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.

W>

 @nielstanis@infosec.exchange

The screenshot shows a web browser displaying the [Definitions](https://reproducible-builds.org/docs/definition/) page of the Reproducible-Builds.org documentation. The page title is "Definitions". The left sidebar includes links for Home, News, Documentation (which is currently selected), Tools, Who is involved?, Talks, Events, Continuous tests, Contribute, and Sponsors. The main content area discusses what makes a build reproducible, mentioning source code, build environment, and build instructions. It also covers explanations, commandments of reproducible builds, and achieving deterministic builds. A red starburst badge in the top right corner of the page area says "New & Improved!".

Reproducible-Builds.org

@nielstanis@infosec.exchange

Application Inspector

New & Improved!

Application Features

This section reports the major characteristics of the application or its primary features organized by customizable Feature Groups. Click any of the highlighted icons below (indicating at least one match) to view additional details or expand a feature group for more information. To view where in source code a specific feature was found, click the Rule name link shown on the right. A disabled icon indicates a not found status for that feature.

Feature Groups	Associated Rules
+ Select Features	Name (click to view source)
+ General Features	Authentication: Microsoft (Identity)
+ Development	Authentication: General
+ Active Content	Authentication: (OAuth)
+ Data Storage	
+ Sensitive Data	
+ Cloud Services	
+ OS Integration	
+ OS System Changes	
+ Other	

W>

@nielstanis@infosec.exchange

<https://github.com/microsoft/ApplicationInspector>

Application Inspector

New & Improved!

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

W>

@nielstanis@infosec.exchange

<https://github.com/microsoft/ApplicationInspector>

Community Review

New &
Improved!

The screenshot shows a web browser window displaying the 'Cargo Vet' documentation at <https://mozilla.github.io/cargo-vet/>. The page has a dark theme with a sidebar on the left containing a table of contents for various sections like Introduction, Tutorial, and Reference. The main content area is titled 'Cargo Vet' and discusses the tool's purpose of auditing third-party Rust dependencies. It includes sections on sharing findings, relative audits, and deferred audits. A red starburst badge in the top right corner says 'New & Improved!'. The URL 'https://mozilla.github.io/cargo-vet/' is visible in the browser's address bar.



@nielstanis@infosec.exchange

<https://mozilla.github.io/cargo-vet/>

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 its own!

W>

 @nielstanis@infosec.exchange

Conclusion

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

W>

 @nielstanis@infosec.exchange

Questions?



 @nielstanis@infosec.exchange

W>

Links

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

W>

 nielstanis@infosec.exchange