



Reviewing NuGet Packages 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



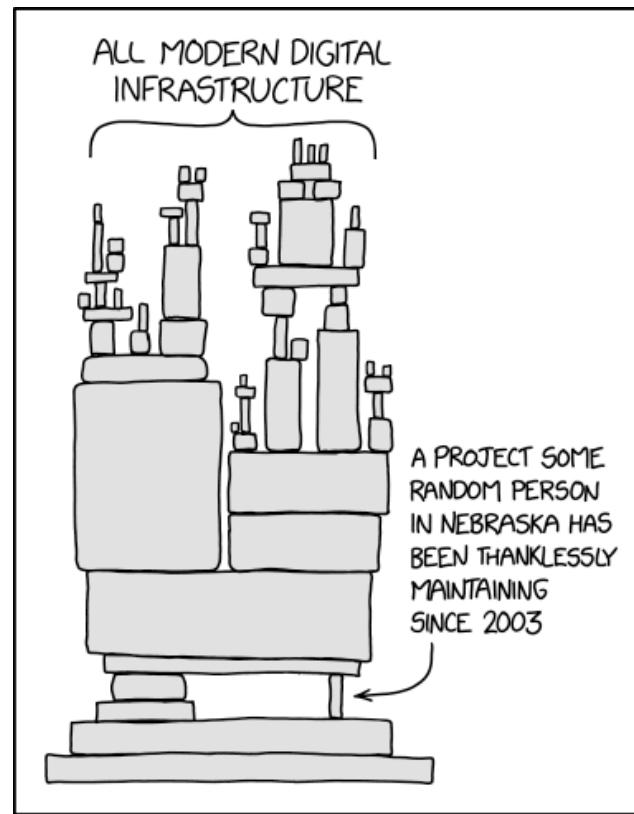
@niels.fennec.dev



@nielstanis@infosec.exchange

Modern Application Architecture

XKCD 2347



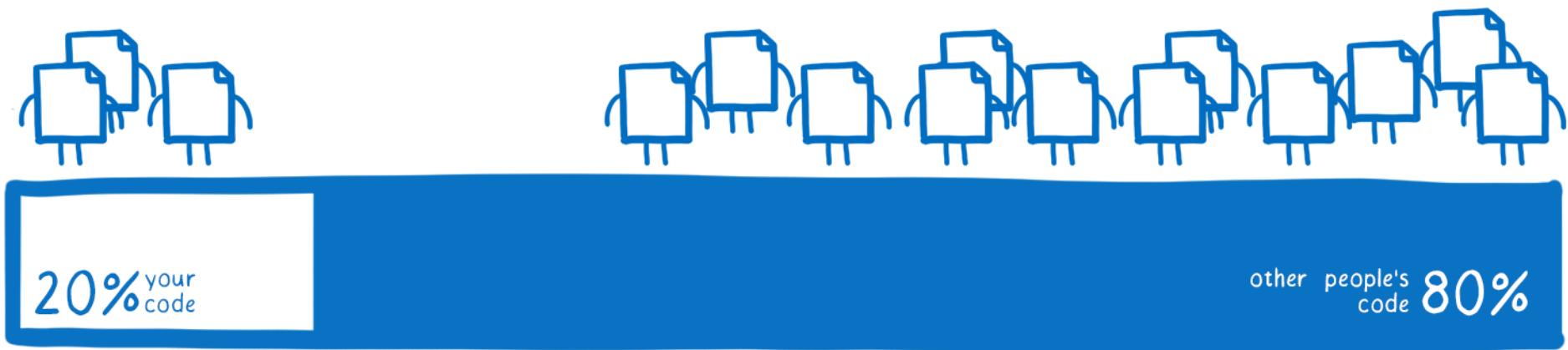
nec.dev  @nielstanis@infosec.exchange

Agenda

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

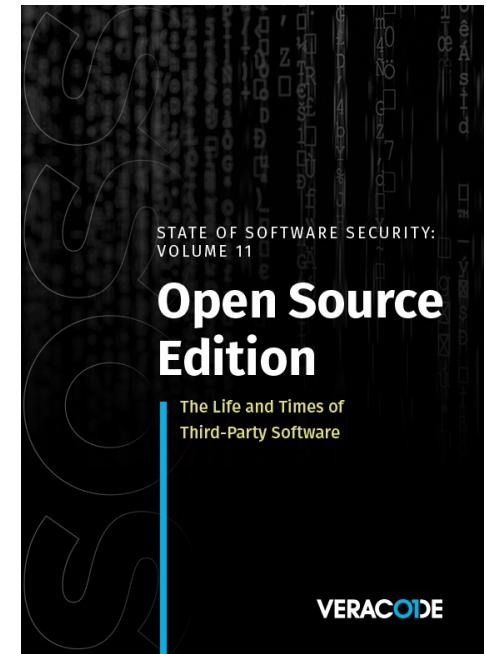


Average codebase composition



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."*

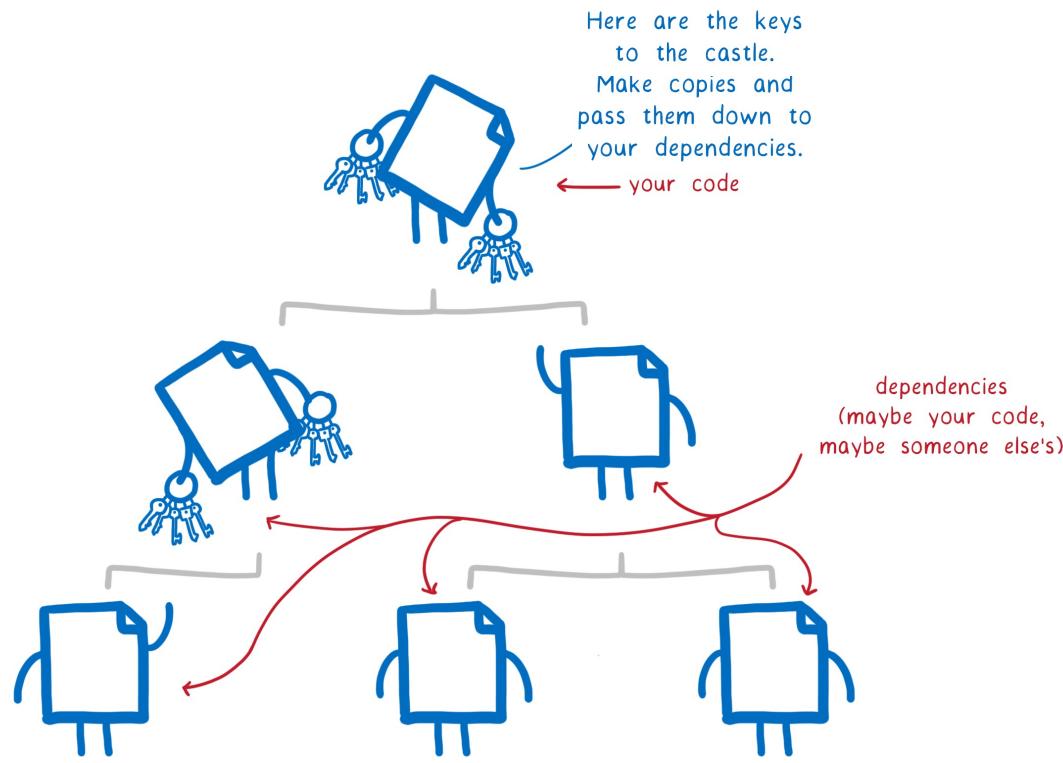


 @niels.fennec.dev  @nielstanis@infosec.exchange

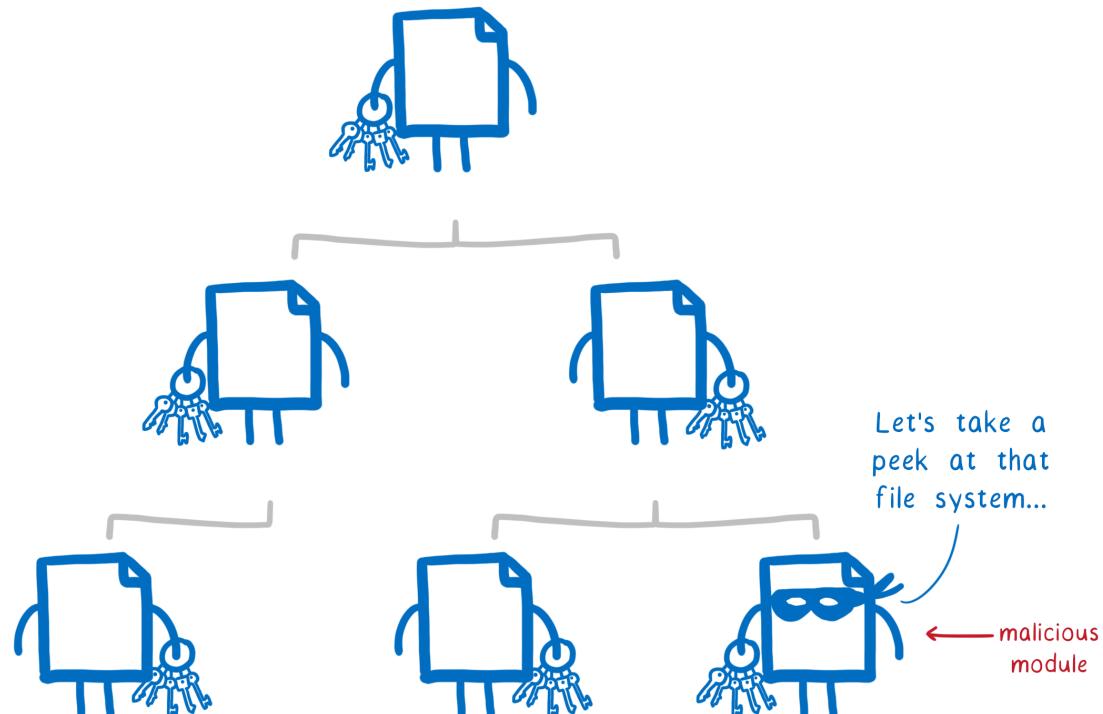
State of Log4j - 2 years later

- 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.

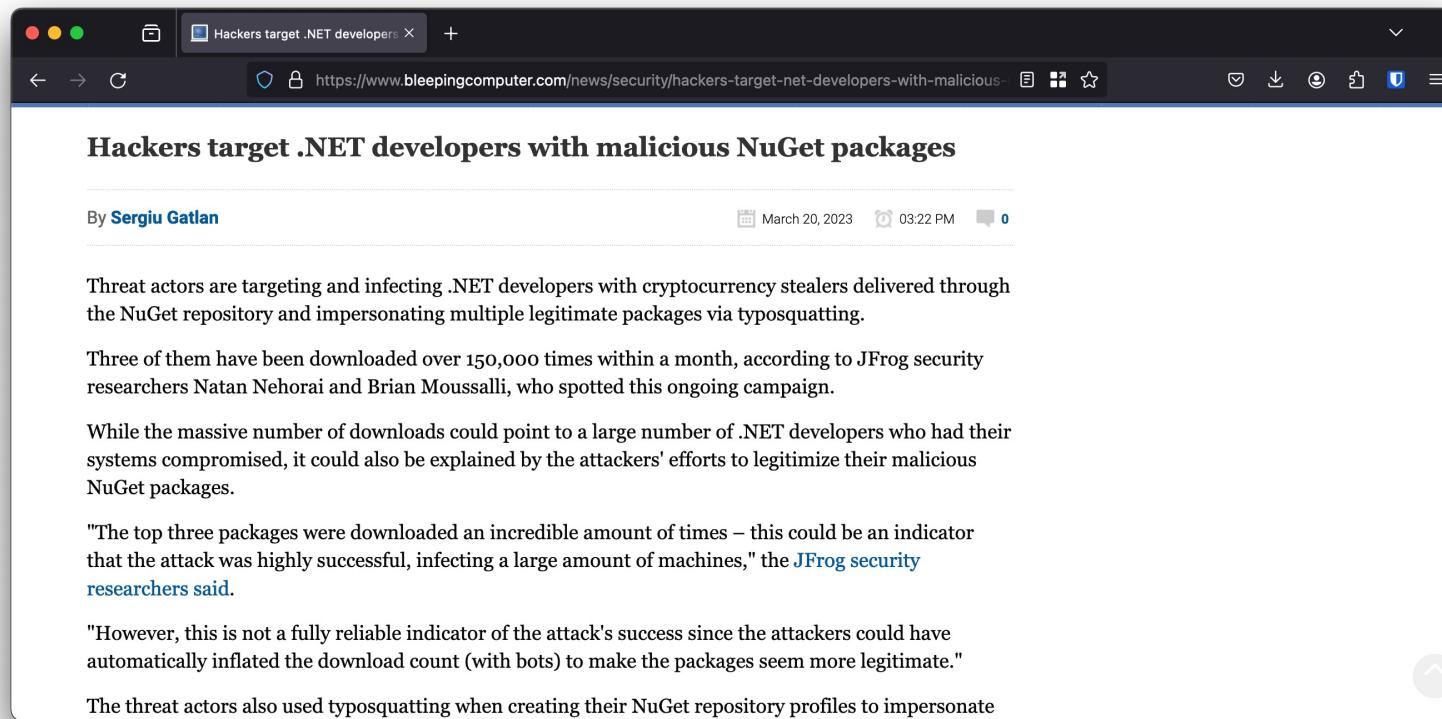
Average codebase composition



Malicious Assembly



Malicious Package



The screenshot shows a web browser window with a dark theme. The title bar reads "Hackers target .NET developers". The main content area displays an article from bleepingcomputer.com. The headline is "Hackers target .NET developers with malicious NuGet packages". Below the headline, it says "By Sergiu Gatlan" and "March 20, 2023 03:22 PM 0". The article text discusses threat actors targeting .NET developers with cryptocurrency stealers delivered through the NuGet repository and impersonating multiple legitimate packages via typosquatting. It mentions three packages downloaded over 150,000 times. Researchers Natan Nehorai and Brian Moussalli spotted the campaign. The text also notes that while the massive number of downloads could point to a large number of .NET developers, it could also be explained by attackers' efforts to legitimize their malicious NuGet packages. Quoted researchers say the attack was highly successful. The threat actors used typosquatting for their NuGet repository profiles.



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

XZ Backdoor

A screenshot of a web browser window showing an Ars Technica article. The title of the article is "Backdoor found in widely used Linux utility targets encrypted SSH connections". The article is categorized under "SUPPLY CHAIN ATTACK". The author is Dan Goodin, and the publication date is 3/29/2024, 7:50 PM. The Ars Technica logo is visible in the top left, and there are links for "SUBSCRIBE", "SIGN IN", and search functions.

ars TECHNICA

SUPPLY CHAIN ATTACK —

Backdoor found in widely used Linux utility targets encrypted SSH connections

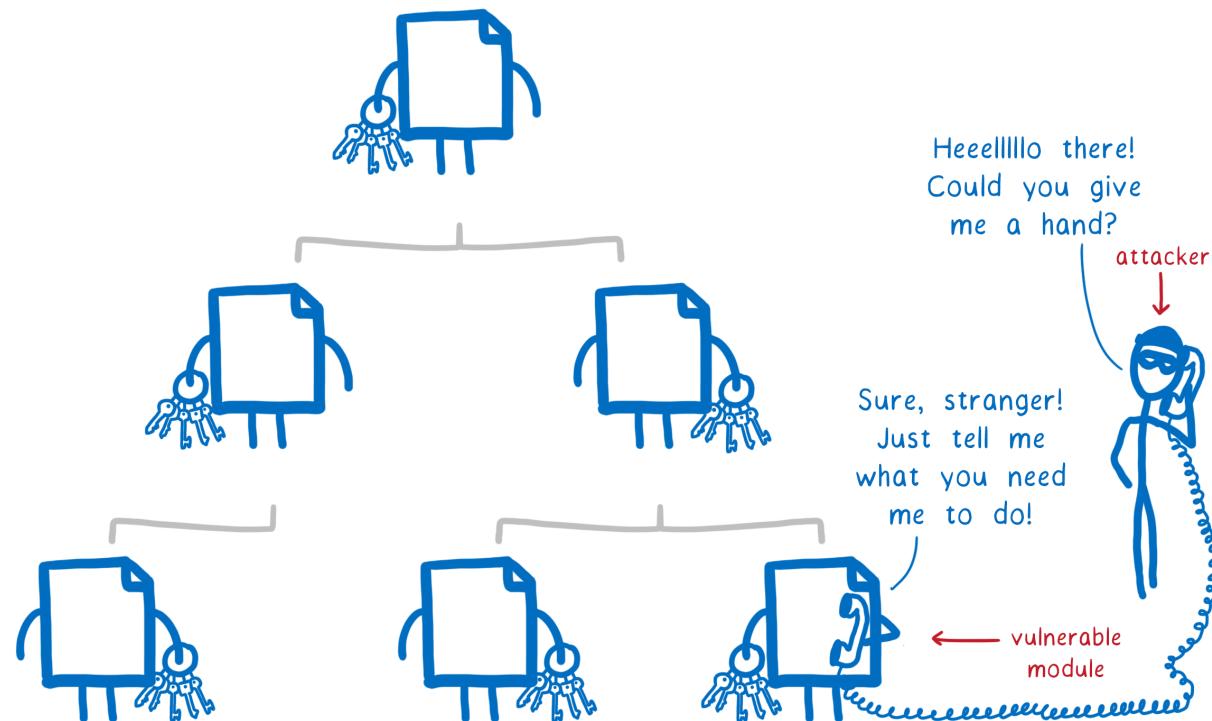
Malicious code planted in xz Utils has been circulating for more than a month.

DAN GOODIN - 3/29/2024, 7:50 PM

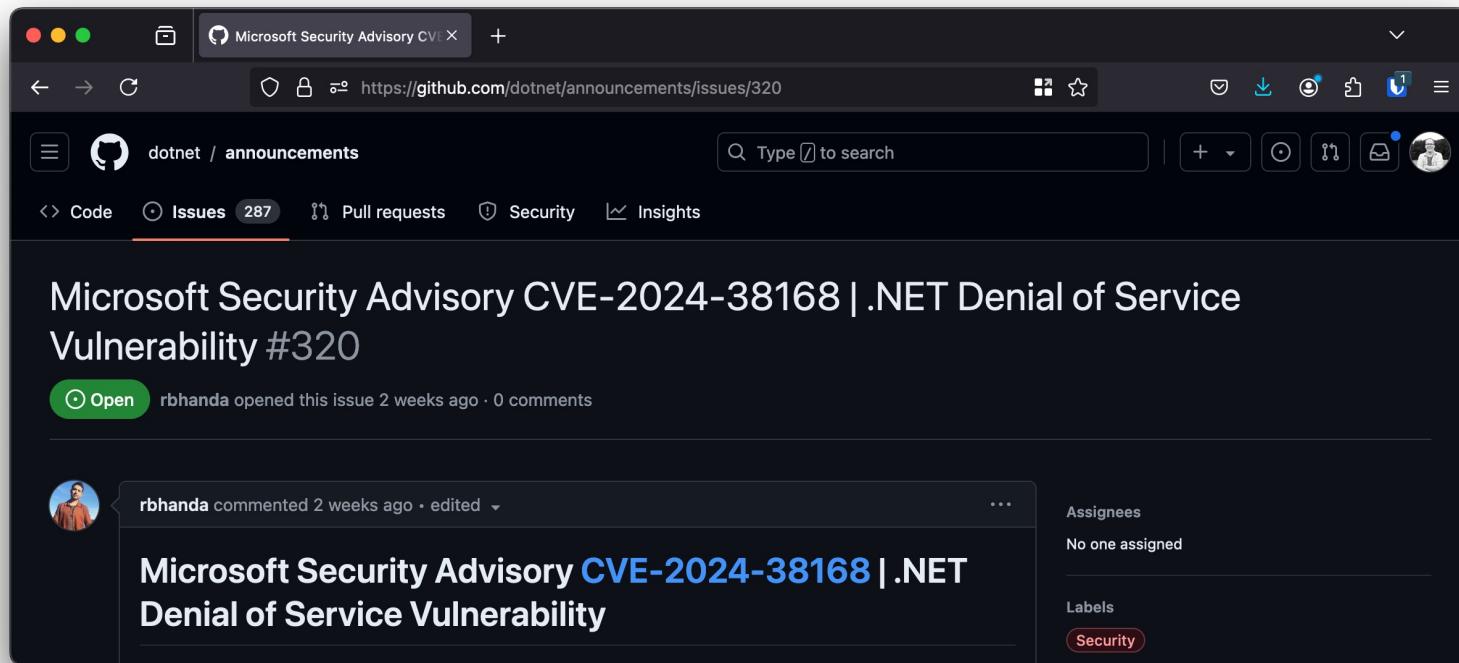


 @niels.fennec.dev  @nielstanis@infosec.exchange

Vulnerable Assembly



Vulnerabilities in Libraries



DotNet CLI

```
nelson@ghost-m2:~/research/consoleapp $ dotnet list package
Project 'consoleapp' has the following package references
[net8.0]:
Top-level Package      Requested   Resolved
> docgenerator         1.0.0       1.0.0

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

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

The given project `consoleapp` has no vulnerable packages given the current sources.
nelson@ghost-m2:~/research/consoleapp $
```

DotNet CLI

```
nelson@ghost-m2:~/research/consoleapp $ dotnet list package --include-transitive
Project 'consoleapp' has the following package references
[net8.0]:
Top-level Package      Requested   Resolved
> docgenerator          1.0.0       1.0.0

Transitive Package                                     Resolved
> iText7                                         7.2.2
> iText7.common                                     7.2.2
> Microsoft.CSharp                                4.0.1
> Microsoft.DotNet.PlatformAbstractions           1.1.0
> Microsoft.Extensions.DependencyInjection            5.0.0
> Microsoft.Extensions.DependencyInjection.Abstractions 5.0.0
> Microsoft.Extensions.DependencyModel             1.1.0
> Microsoft.Extensions.Logging                     5.0.0
> Microsoft.Extensions.Logging.Abstractions        5.0.0
> Microsoft.Extensions.Options                   5.0.0
> Microsoft.Extensions.Primitives                5.0.0
```

DotNet CLI

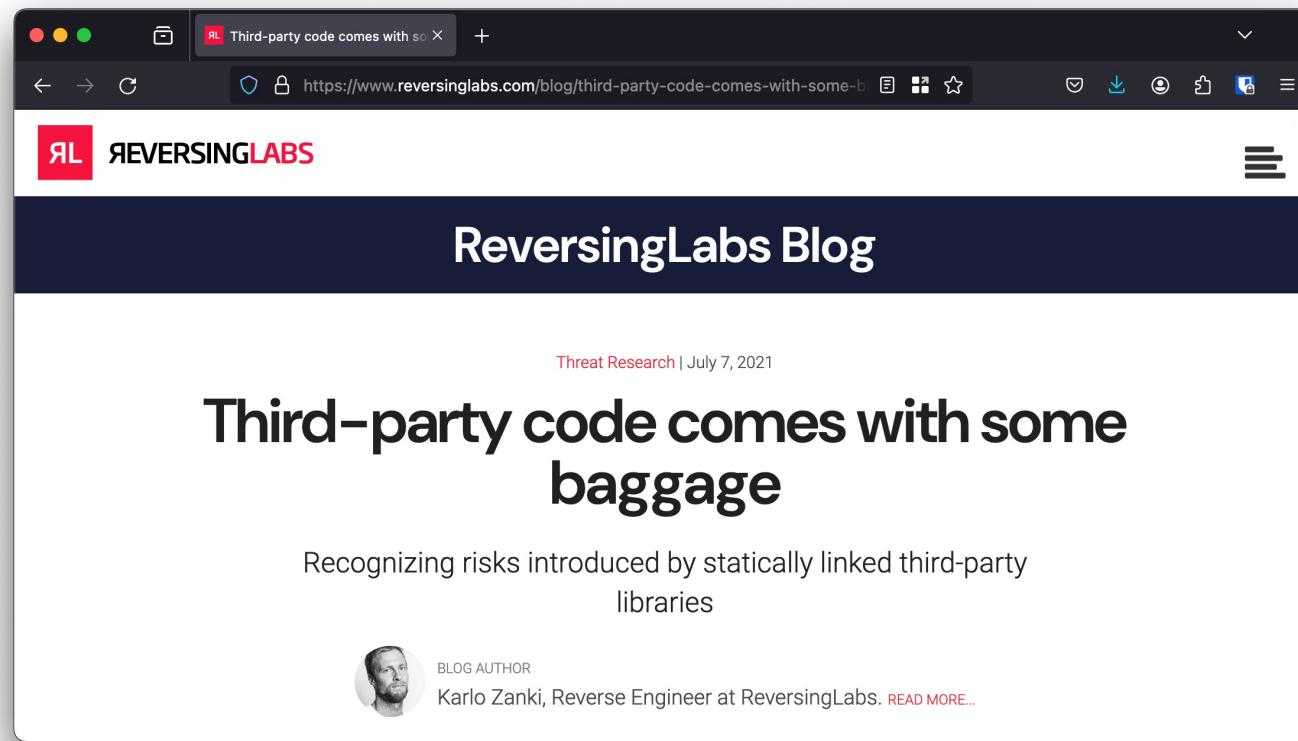
```
nelson@ghost-m2 ~/research/consoleapp $ dotnet list package --vulnerable --include-transitive
nelson@ghost-m2 ~/research/consoleapp $ dotnet list package --vulnerable --include-transitive

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 $
```

Do you know what's inside?



A screenshot of a web browser window showing a blog post from ReversingLabs. The browser has a dark theme with a blue header bar. The tab title is "RL Third-party code comes with some baggage". The URL in the address bar is <https://www.reversinglabs.com/blog/third-party-code-comes-with-some-b>. The page features the ReversingLabs logo (a red square with "RL") and the text "ReversingLabs Blog". Below the header, a timestamp reads "Threat Research | July 7, 2021". The main title of the post is "Third-party code comes with some baggage", described as "Recognizing risks introduced by statically linked third-party libraries". A circular profile picture of a man is shown next to the author's name, Karlo Zanki, Reverse Engineer at ReversingLabs, with a "READ MORE..." link.

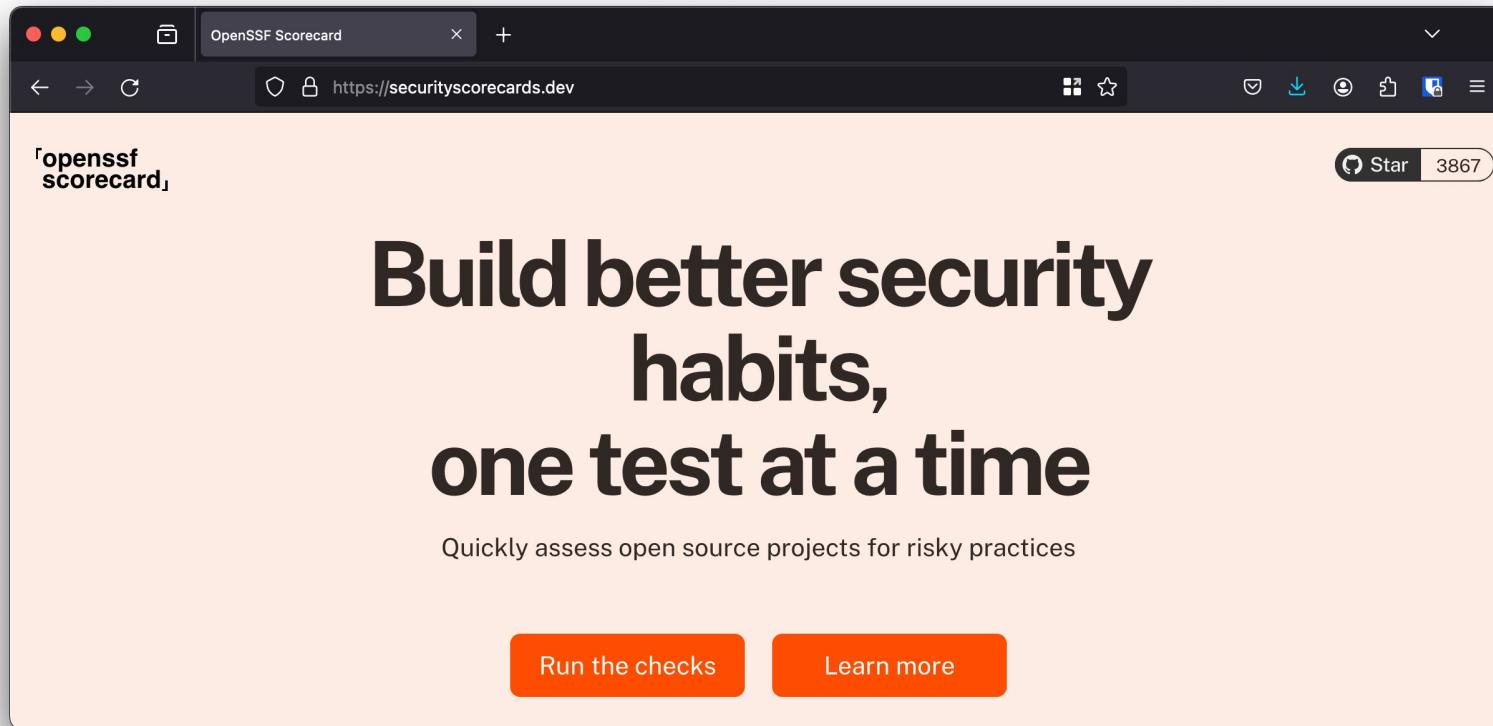


 @niels.fennec.dev  @nielstanis@infosec.exchange

Nutrition Label for Software?

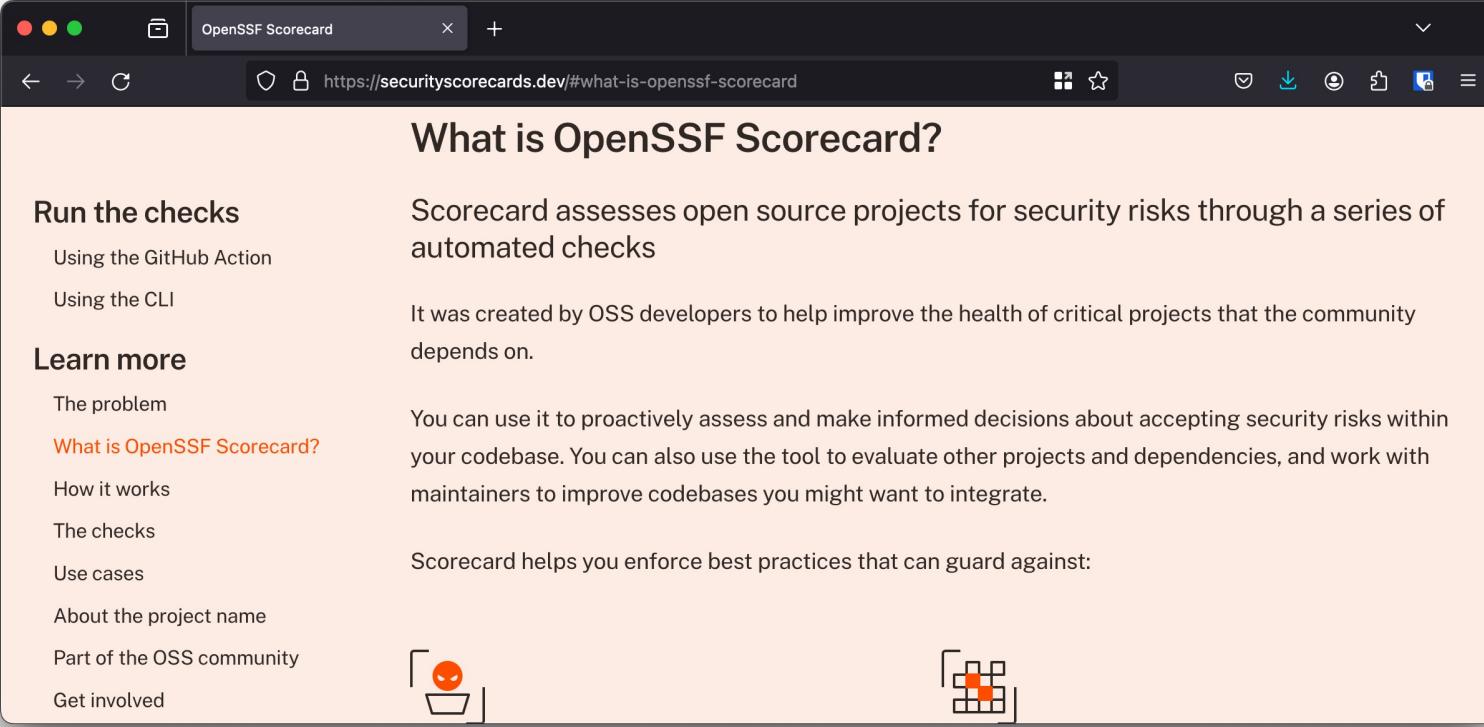


OpenSSF Scorecards



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

OpenSSF Security Scorecards



The screenshot shows a web browser window with the title bar "OpenSSF Scorecard". The URL in the address bar is <https://securityscorecards.dev/#what-is-openssf-scorecard>. The main content area has a light orange background and displays the following information:

What is OpenSSF Scorecard?

Run the checks

- Using the GitHub Action
- Using the CLI

Learn more

- The problem
- What is OpenSSF Scorecard?** (highlighted in orange)
- How it works
- The checks
- Use cases
- About the project name
- Part of the OSS community
- Get involved

Scorecard assesses open source projects for security risks through a series of automated checks

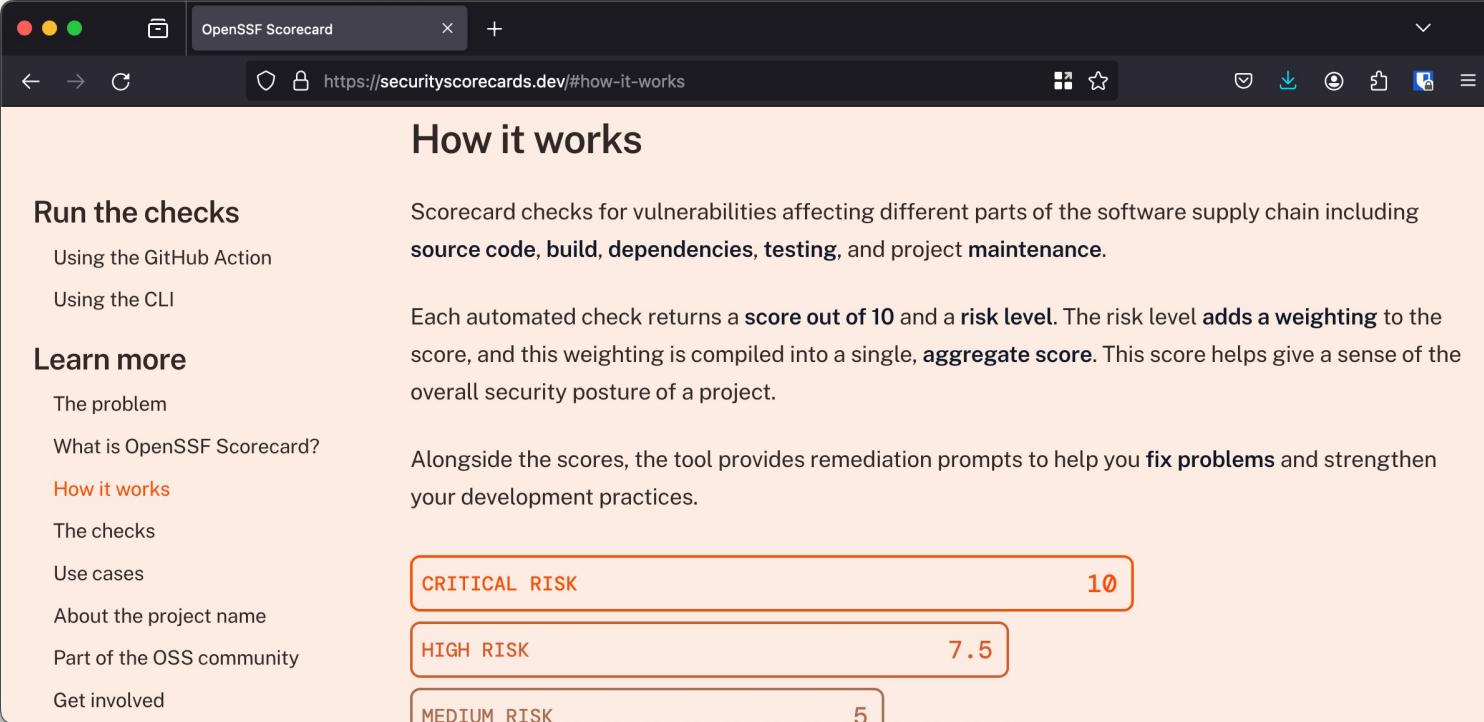
It was created by OSS developers to help improve the health of critical projects that the community depends on.

You can use it to proactively assess and make informed decisions about accepting security risks within your codebase. You can also use the tool to evaluate other projects and dependencies, and work with maintainers to improve codebases you might want to integrate.

Scorecard helps you enforce best practices that can guard against:

Two small icons are shown at the bottom: a red face in a white circle with a black outline, and a 4x4 grid of squares with one red square in the middle.

OpenSSF Security Scorecards



The screenshot shows a web browser window with the title bar "OpenSSF Scorecard". The URL in the address bar is <https://securityscorecards.dev/#how-it-works>. The main content area has a light orange background and features the heading "How it works". Below this, there are two sections: "Run the checks" and "Learn more".

Run the checks

- Using the GitHub Action
- Using the CLI

Learn more

- The problem
- What is OpenSSF Scorecard?
- How it works** (This item is highlighted in red)
- The checks
- Use cases
- About the project name
- Part of the OSS community
- Get involved

On the right side of the page, there is a summary of the score results:

| | |
|---------------|-----|
| CRITICAL RISK | 10 |
| HIGH RISK | 7.5 |
| MEDIUM RISK | 5 |

OpenSSF Security Scorecards

The screenshot shows a web browser window for the OpenSSF Scorecard. The URL is <https://securityscorecards.dev/#the-checks>. The page content includes:

- Run the checks**
 - Using the GitHub Action
 - Using the CLI
- 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

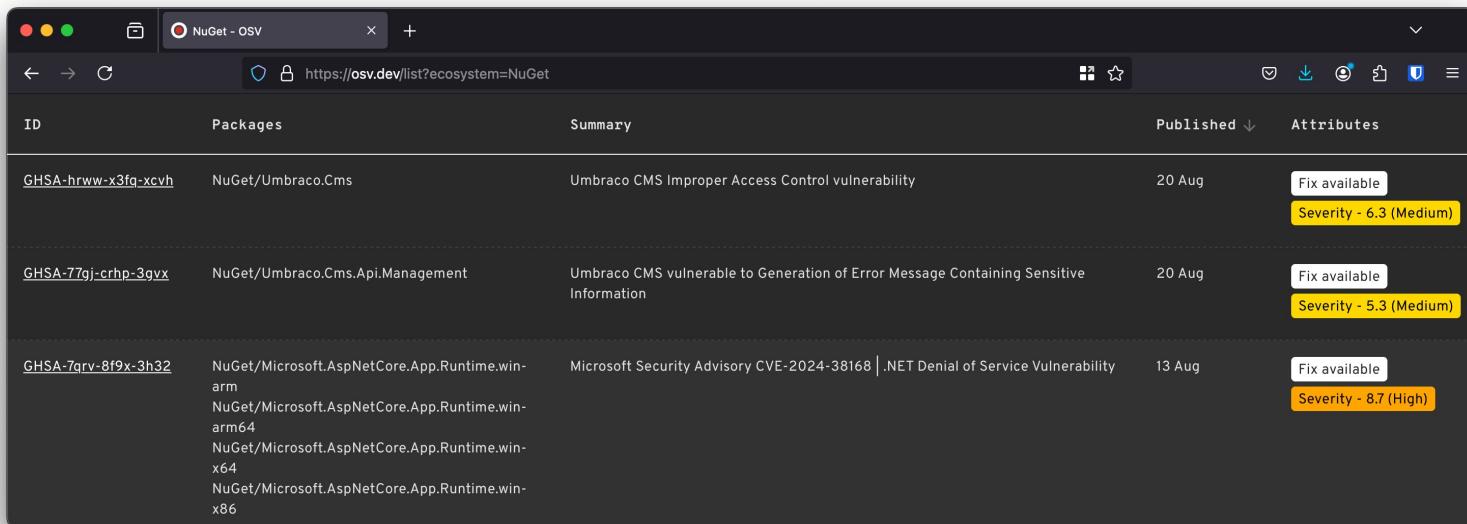
On the right side of the page is a graphic consisting of five circles arranged in a circle, all contained within a larger outer circle. The circles are labeled:

- RISK ASSESSMENT
- SOURCE RISK ASSESSMENT
- CONTINUOUS TESTING
- MAINTENANCE
- CODE VULNERABILITIES

In the center of the inner circle, the text "HOLISTIC SECURITY PRACTICES" is written in orange capital letters.

Code Vulnerabilities (High)

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



The screenshot shows a dark-themed web browser window with the title "NuGet - OSV". The URL in the address bar is <https://osv.dev/list?ecosystem=NuGet>. The page displays a table of vulnerabilities:

| ID | Packages | Summary | Published | Attributes |
|-------------------------------------|--|--|-----------|---|
| GHSA-hrwu-x3fq-xcvh | NuGet/Umbraco.Cms | Umbraco CMS Improper Access Control vulnerability | 20 Aug | Fix available Severity - 6.3 (Medium) |
| GHSA-77gj-crhp-3gvx | NuGet/Umbraco.Cms.Api.Management | Umbraco CMS vulnerable to Generation of Error Message Containing Sensitive Information | 20 Aug | Fix available Severity - 5.3 (Medium) |
| GHSA-7qrv-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 | Fix available Severity - 8.7 (High) |

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

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 .NET projects?

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 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?

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

- Does the project pin dependencies used during its build and release process.
- **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 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

Measure?



OpenSSF Annual Report 2023

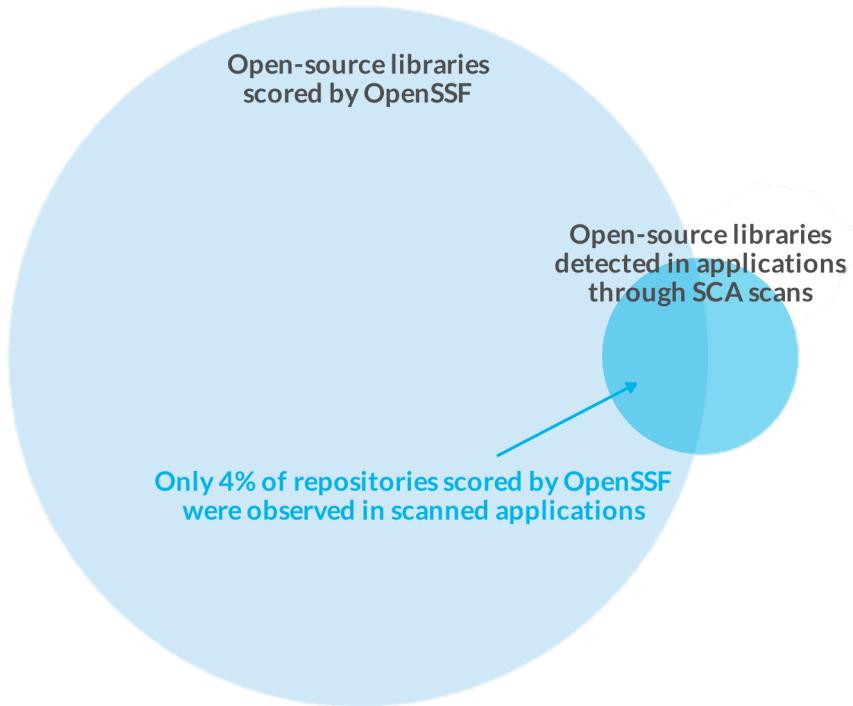
OpenSSF Scorecard project
has **3,776 stars** on GitHub,
and runs a **weekly automated**
assessment scan against
software security criteria
of over **1M OSS projects**



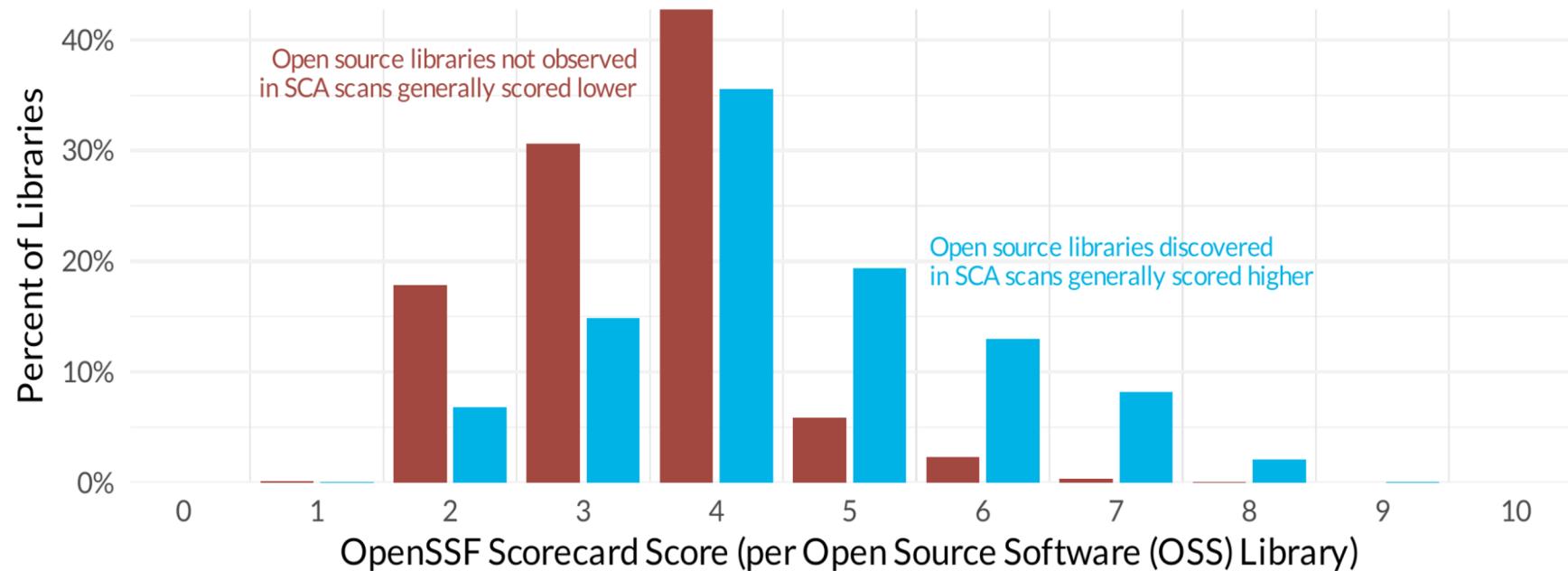
SOSS & OpenSSF Scorecard



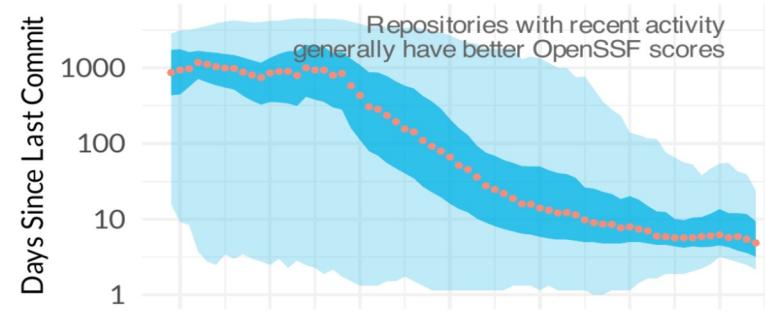
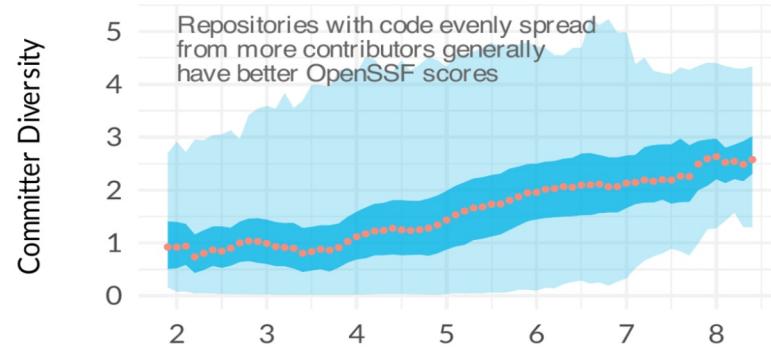
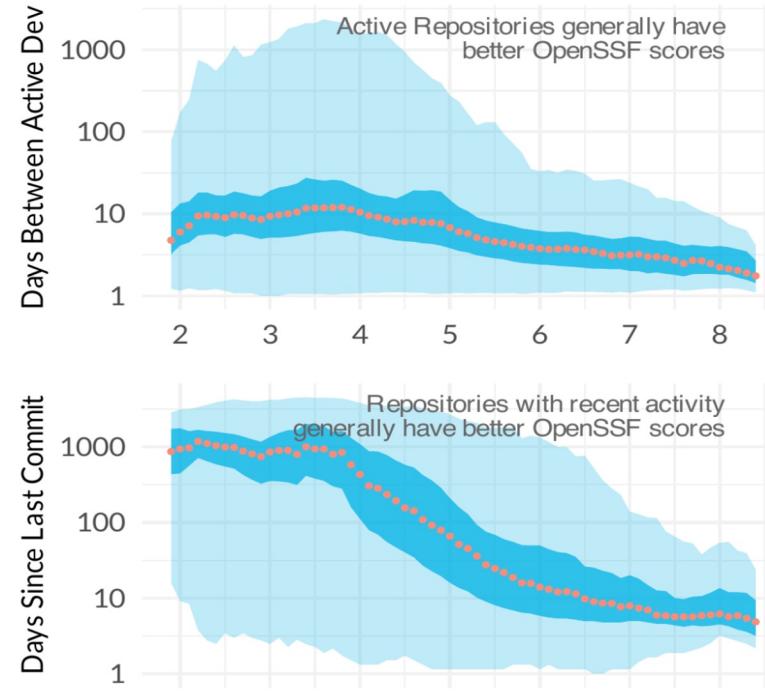
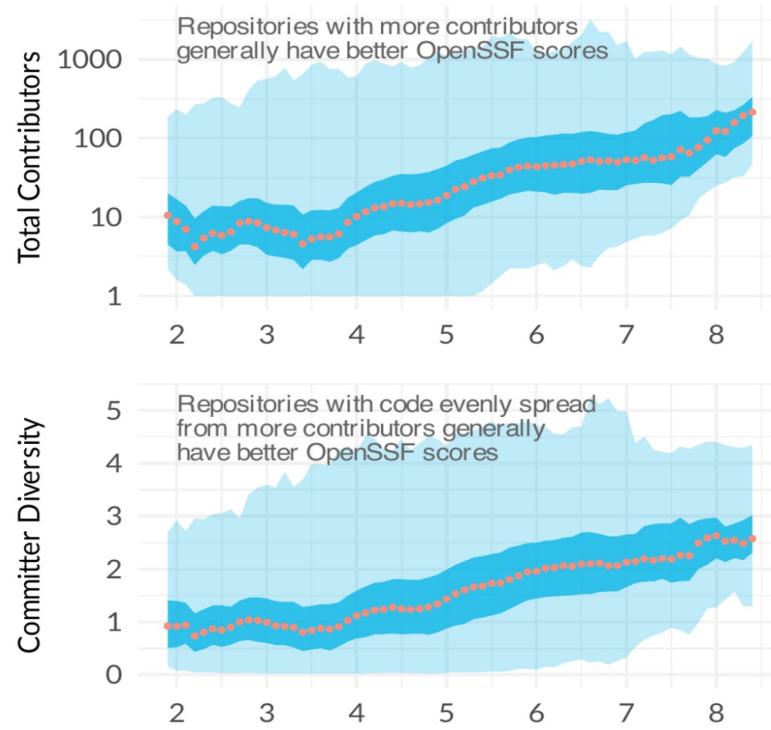
SOSS & OpenSSF Scorecard



Correlation between SOSS



Github commits vs OpenSSF



What really contributes to OSS Security?



What can we improve?



Fuzzing .NET



- 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?

Fuzzing .NET & SharpFuzz

New &
Improved!

The screenshot shows a web browser window with a dark theme. The title bar reads "Five years of fuzzing .NET with Shar X". The address bar shows the URL "https://mijailovic.net/2023/07/23/sharpfuzz-anniversary/". The main content area is titled "Nemanja Mijailovic's Blog" and features a large heading "Five years of fuzzing .NET with SharpFuzz". Below the heading is the date "Jul 23, 2023". The text discusses the history of SharpFuzz, mentioning its creation five years ago and its evolution to support libFuzzer, Windows, and .NET Framework, including the .NET Core base-class library. It also notes that the fuzzing process has been simplified. At the bottom, it states that not many people are aware of these developments and encourages reading the anniversary blog post.

Five years of fuzzing .NET with SharpFuzz

Jul 23, 2023

It's been almost five years since I created **SharpFuzz**, the only .NET coverage-guided fuzzer. I already have a blog post on how it works, what it can do for you, and what bugs it found, so check it out if this is the first time you hear about SharpFuzz:

[SharpFuzz: Bringing the power of afl-fuzz to .NET platform](#)

A lot of interesting things have happened since then. SharpFuzz now works with libFuzzer, Windows, and .NET Framework. And it can finally fuzz the .NET Core base-class library! The whole fuzzing process has been dramatically simplified, too.

Not many people are aware of all these developments, so I decided to write this anniversary blog post and showcase everything SharpFuzz is currently capable of.



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

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) { }
    });
}
```



@niels.fennec.dev



@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");
        }
    }
}
```



.NET Reproducibility



- Reproducible builds → independently-verifiable path from source to binary code.
- .NET Roslyn Deterministic Inputs
- How reproducible is a simple console app?
- Fennec Diff (work-in-progress)

Application Inspector

New &
Improved!

The screenshot shows the Microsoft Application Inspector interface. At the top, there's a navigation bar with tabs for Overview, Summary, Features, and About. Below this is a section titled "Application Features" which contains a detailed description of how it organizes application features into customizable Feature Groups. To the right of this description is a "Feature Groups" section listing various categories with their corresponding icons. Further down is an "Associated Rules" section, which lists specific rules like "Authentication: Microsoft (Identity)" and "Authentication: General".



@niels.fennec.dev



@nielstanis@infosec.exchange

Application Inspector

New &
Improved!

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 |



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

NuGet Blog

A screenshot of a Microsoft Dev Blogs article. The title is "OpenSSF Scorecard for .NET and the NuGet ecosystem". The article was published on November 4th, 2024. It features three authors: Ioana, Avishay, and Mélanie. The content discusses the OpenSSF Scorecard tool, which provides automated security assessments for open-source projects. The sidebar includes a "Table of contents" section with links to "Overview", "What is Scorecard and why should you use it", and "What are the checks run by Scorecard". A "Feedback" button is also present.

November 4th, 2024

OpenSSF Scorecard for .NET and the NuGet ecosystem

Ioana, Avishay, Mélanie

[OpenSSF Scorecard](#) is a tool developed by the Open Source Security Foundation (OpenSSF) that provides automated security assessments for open-source projects. The primary goal of the Scorecard project is to help developers and users determine the security posture of open-source software by generating a score based on a series of security-related checks.

Table of contents

- Overview
- What is Scorecard and why should you use it
- What are the checks run by Scorecard

Feedback



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

Conclusion

- Scorecard helps security reviewing a NuGet 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!

Conclusion

- Room for .NET specific improvements with Fennec CLI
 - Tools (diff, insights)
 - Transitive Trust Graph
 - Contribute back to OpenSSF Scorecard



```
dotnet tool install -g fennec --prerelease
```

Danke, Merci, Bedankt!

- <https://github.com/nielstanis/eliadevcon2024/>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://www.fennec.dev>
<https://blog.fennec.dev>