



## Reviewing NuGet Packages security easily using OpenSSF Scorecard

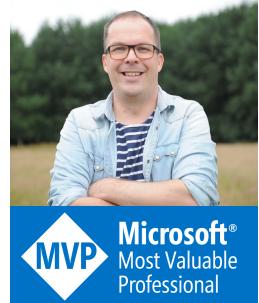
Niels Tanis  
Sr. Principal Security Researcher

**COPENHAGEN  
DEVELOPERS FESTIVAL**  
BY NDC CONFERENCES

## 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**



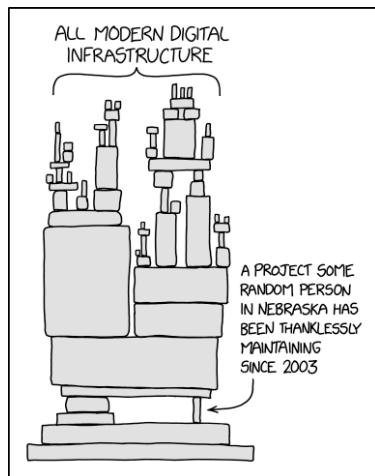
**MVP** Microsoft®  
Most Valuable  
Professional

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 [@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

# Modern Application Architecture XKCD 2347

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUD CONFERENCES



@nielstanis@infosec.exchange

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

# Agenda

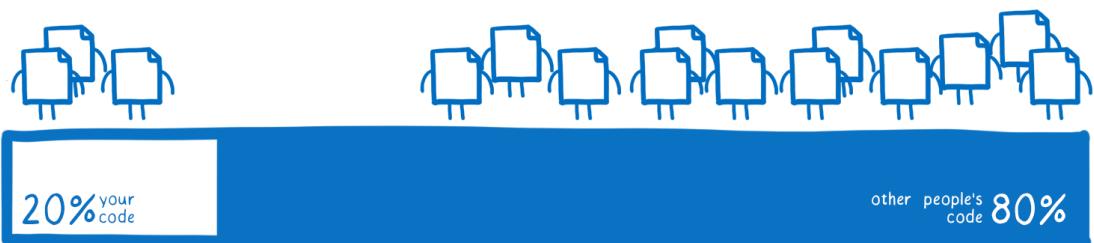
- Risks in 3<sup>rd</sup> party NuGet Packages
- OpenSFF Scorecard
- Measure, New & Improved
- Conclusion - Q&A



 @nielstanis@infosec.exchange

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

## Average codebase composition



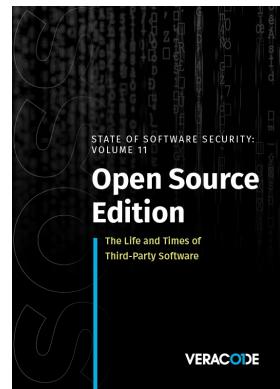
COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@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."*



COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@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.

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

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

# Average codebase composition

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

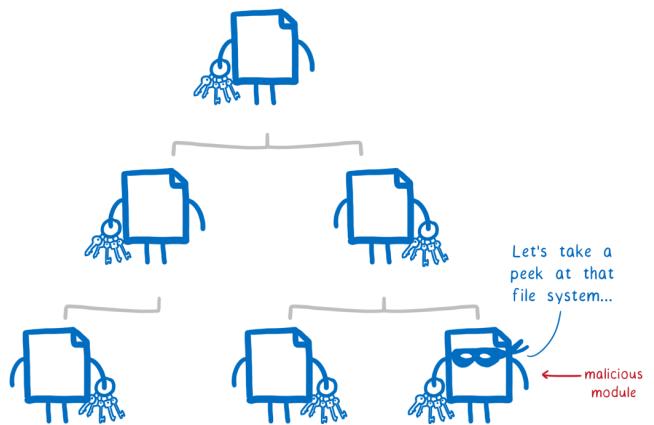
Here are the keys  
to the castle.  
Make copies and  
pass them down to  
your dependencies.  
← your code

dependencies  
(maybe your code,  
maybe someone else's)

 @nielstanis@infosec.exchange

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

## Malicious Assembly



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

# Malicious Package

The screenshot shows a news article titled "Hackers target .NET developers with malicious NuGet packages" by Sergiu Gatlan. The article discusses threat actors targeting .NET developers with cryptocurrency stealers delivered through the NuGet repository. It mentions three packages downloaded over 150,000 times. Researchers Natan Nehorai and Brian Moussalli spotted the campaign. The article quotes JFrog security researchers and notes the use of typosquatting. At the bottom, there's a mention of the Copenhagen Developers Festival and a Twitter handle @nielstanis@infosec.exchange.

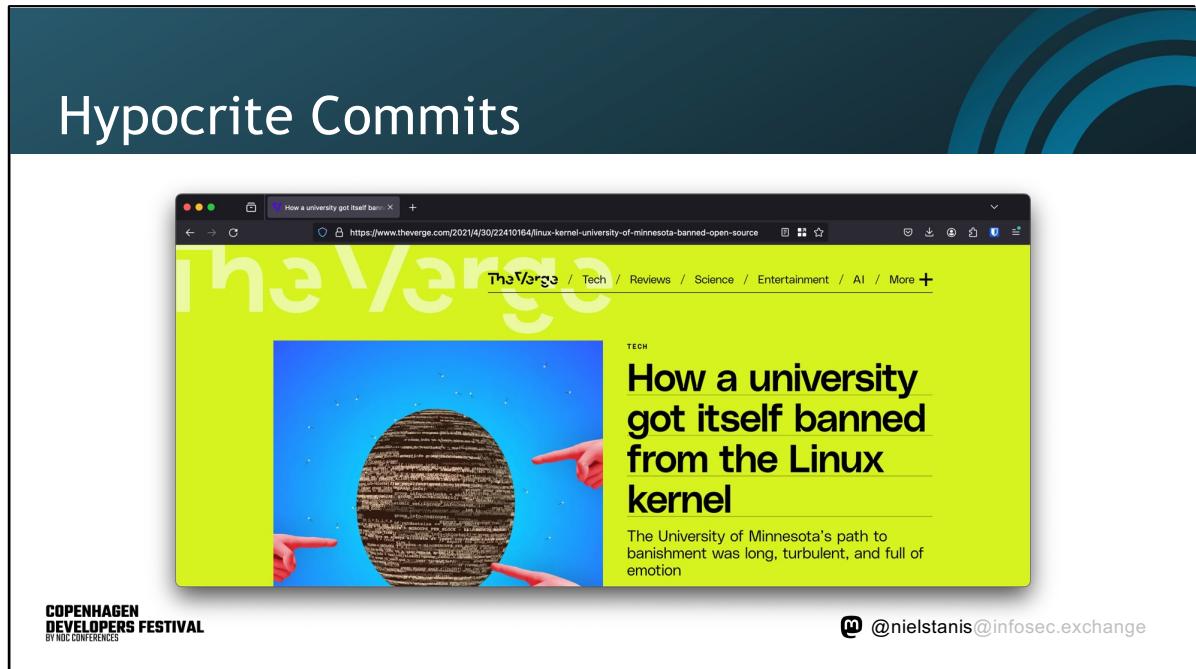
[https://www.bleepingcomputer.com/news/security/hackers-target-net-developers-with-malicious-nuget-packages//](https://www.bleepingcomputer.com/news/security/hackers-target-net-developers-with-malicious-nuget-packages/)

# Malicious Package

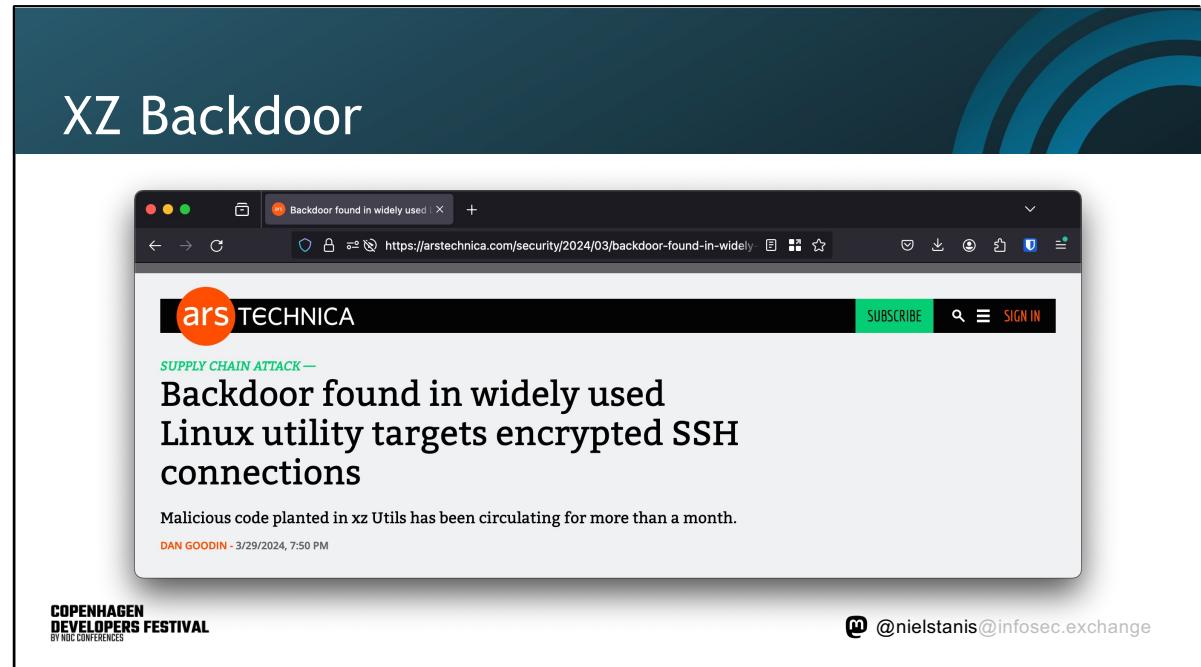
The screenshot shows a web browser displaying a blog post from ReversingLabs. The title of the post is "Malicious NuGet campaign uses homoglyphs and IL weaving to fool devs". The post discusses how malware authors used homoglyphs and IL weaving to inject malicious code. The sidebar on the right lists various topics such as Threat Research, All Blog Posts, AppSec & Supply Chain Security, Dev & DevSecOps, Threat Research, Security Operations, Products & Technology, and Company & Events. The URL of the post is <https://www.reversinglabs.com/blog/malicious-nuget-campaign-uses-homoglyphs-and-il-weaving-to-fool-devs>.

<https://www.reversinglabs.com/blog/malicious-nuget-campaign-uses-homoglyphs-and-il-weaving-to-fool-devs>

## Hypocrite Commits

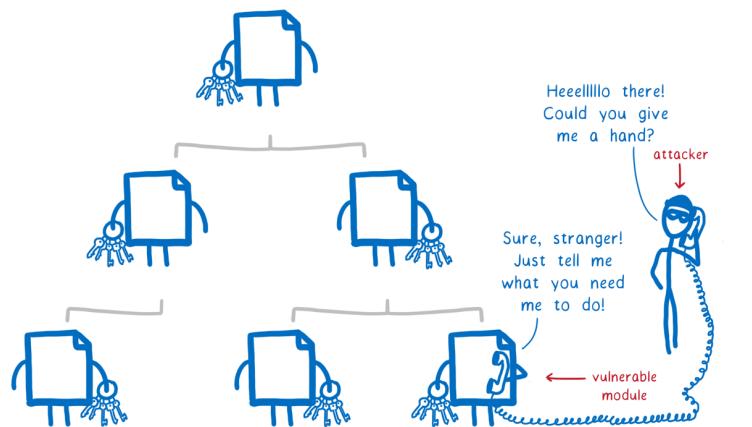


<https://www.theverge.com/2021/4/30/22410164/linux-kernel-university-of-minnesota-banned-open-source>



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

# Vulnerable Assembly

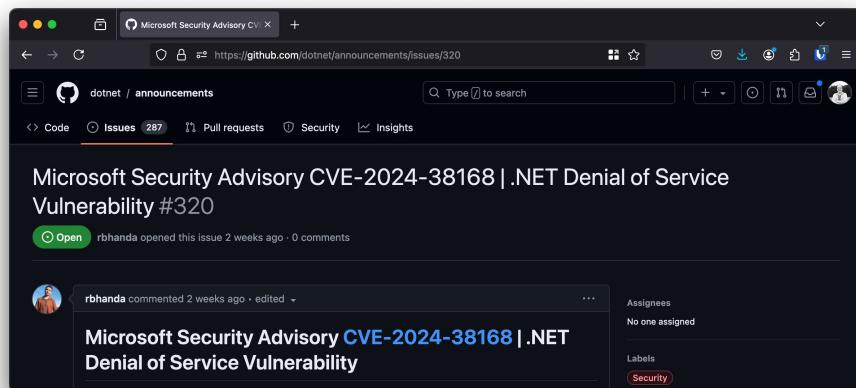


COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

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

# Vulnerabilities in Libraries



COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

<https://github.com/dotnet/announcements/issues/320>

# 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 $
```

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 @nielstanis@infosec.exchange

# 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
```

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 @nielstanis@infosec.exchange

# 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 $
```

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 @nielstanis@infosec.exchange

# Do you know what's inside?

The screenshot shows a web browser window with the URL <https://www.reversinglabs.com/blog/third-party-code-comes-with-some-baggage>. The page title is "ReversingLabs Blog". The main content is a blog post titled "Third-party code comes with some baggage" by Karlo Zanki, Reverse Engineer at ReversingLabs. The post discusses recognizing risks introduced by statically linked third-party libraries. The author's profile picture and name are shown, along with a "READ MORE..." link. The browser has a dark theme, and the top bar shows the title of the post. A watermark for "COPENHAGEN DEVELOPERS FESTIVAL" is visible on the left side of the browser window.

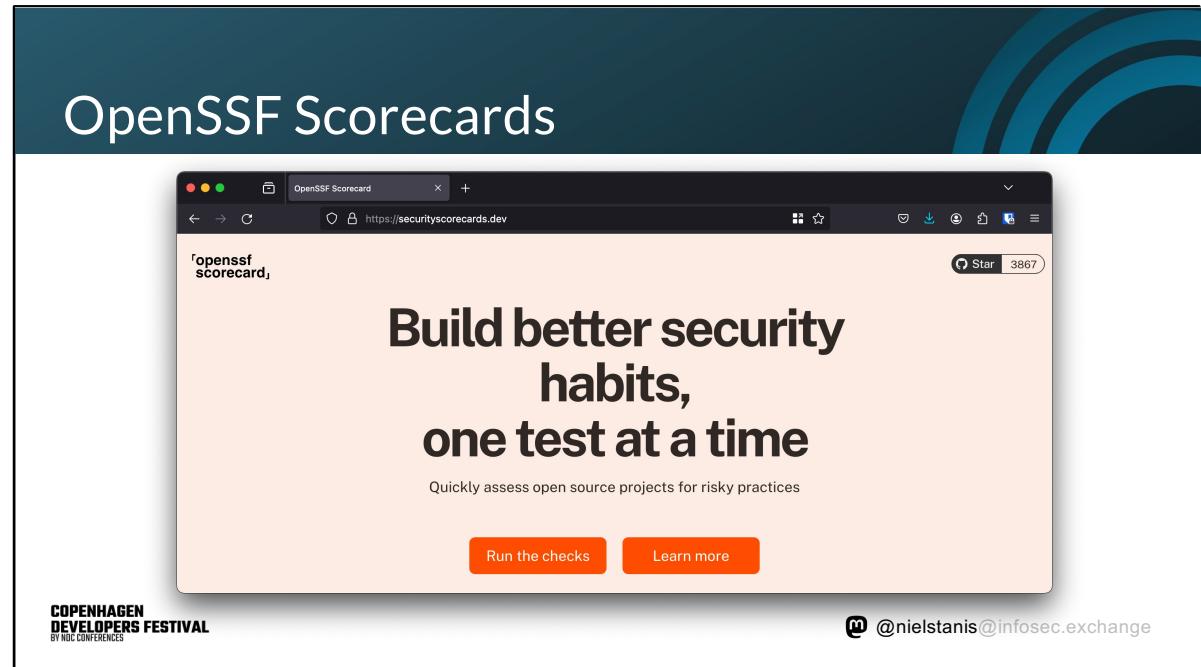
<https://www.reversinglabs.com/blog/third-party-code-comes-with-some-baggage>

## 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's a sidebar with sections for "Run the checks" (GitHub Action, CLI) and "Learn more" (problem, how it works, checks, use cases, project name, OSS community, get involved). The main content area explains what the scorecard is, its purpose, and how it can be used. It also features two small icons: a red circle with a white dot and a grid of squares.

**OpenSSF Security Scorecards**

**What is OpenSSF Scorecard?**

**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

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:

<https://securityscorecards.dev/>

The screenshot shows a web browser window with the title 'OpenSSF Security Scorecards' at the top. The main content area is titled 'How it works'. On the left, there's 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 explains the tool's functionality, mentioning automated checks for vulnerabilities across the software supply chain, a score out of 10, risk levels (CRITICAL RISK, HIGH RISK, MEDIUM RISK), and remediation prompts. At the bottom right, there's a social media handle '@nielstanis@infosec.exchange'.

<https://securityscorecards.dev/>

# OpenSSF Security Scorecards

The screenshot shows a web browser window for the OpenSSF Scorecard. The URL in the address bar is <https://securityscorecards.dev/#the-checks>. The page features a large graphic in the center with a red border containing five overlapping circles. The circles are labeled: CODE VULNERABILITIES (top), BUILD ASSESSMENT (left), MAINTENANCE (right), SOURCE RISK ASSESSMENT (bottom-left), and CONTINUOUS TESTING (bottom-right). In the center of these circles is the text "HOLISTIC SECURITY PRACTICES". To the left of the graphic, there is a sidebar with two sections: "Run the checks" and "Learn more". The "Run the checks" section includes links for "Using the GitHub Action" and "Using the CLI". The "Learn more" section includes links for "The problem", "What is OpenSSF Scorecard?", "How it works", "The checks" (which is highlighted in orange), "Use cases", "About the project name", "Part of the OSS community", and "Get involved". At the bottom left of the page is a small logo for "COPENHAGEN DEVELOPERS FESTIVAL BY NDC CONFERENCES". At the bottom right is a social media link for "@nielstanis@infosec.exchange".

<https://securityscorecards.dev/>

## Code Vulnerabilities (High)

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

ID	Packages	Summary	Published	Attributes
GHSA-hrwv-x3fq-vcv7	NuGet/Umbraco.Cms	Umbraco CMS Improper Access Control vulnerability	20 Aug	<span>Fix available</span> Severity - 6.3 (Medium)
GHSA-77gj-chp-39yx	NuGet/Umbraco.Cms.Api.Management	Umbraco CMS vulnerable to Generation of Error Message Containing Sensitive Information	20 Aug	<span>Fix available</span> 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	<span>Fix available</span> Severity - 8.7 (High)

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUG CONFERENCE

@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.

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 [@nielstanis@infosec.exchange](mailto:@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.

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUD CONFERENCES

 @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.

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUD CONFERENCES

 @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>

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

## 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).

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 [@nielstanis@infosec.exchange](mailto:@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 .NET projects?

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 @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!

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUD CONFERENCES

 [@nielstanis@infosec.exchange](mailto:@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!**

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 @nielstanis@infosec.exchange

## 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!

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUD CONFERENCES

 [@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

## 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/>

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUD CONFERENCES

 @nielstanis@infosec.exchange

## 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)

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 [@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

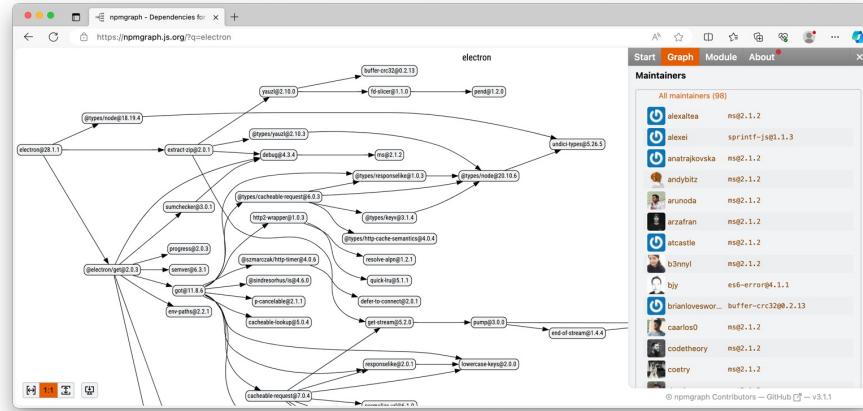
## Source Risk Assessment Contributors (Low)

- 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?

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 @nielstanis@infosec.exchange

# Source Risk Assesement Contributors (Low)



COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

## Build Risk Assessment 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?

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUD CONFERENCES

 @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.

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUD CONFERENCES

 @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.

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 [@nielstanis@infosec.exchange](mailto:@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

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 [@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

# Demo OpenSSF Scorecard Fennec CLI

Running checks



COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

# Measure?



COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 @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**



COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

[@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

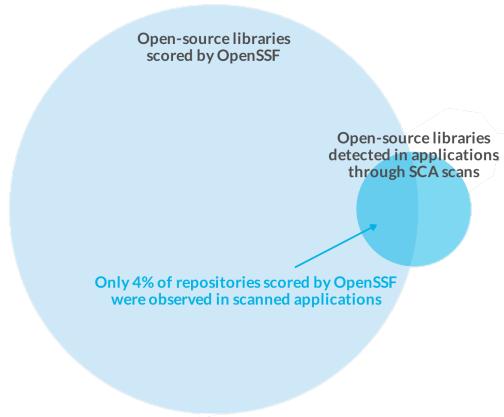
<https://openssf.org/download-the-2023-openssf-annual-report/>

## SOSS & OpenSSF Scorecard



 @nielstanis@infosec.exchange

# SOSS & OpenSSF Scorecard

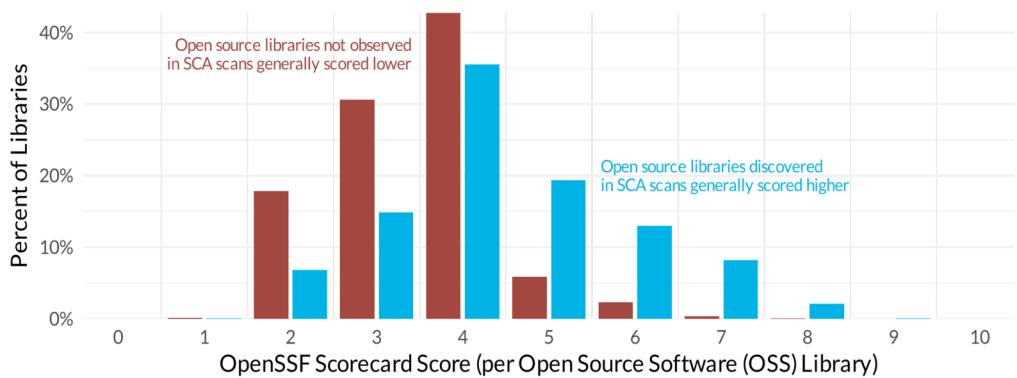


COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUD CONFERENCES

@nielstanis@infosec.exchange

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

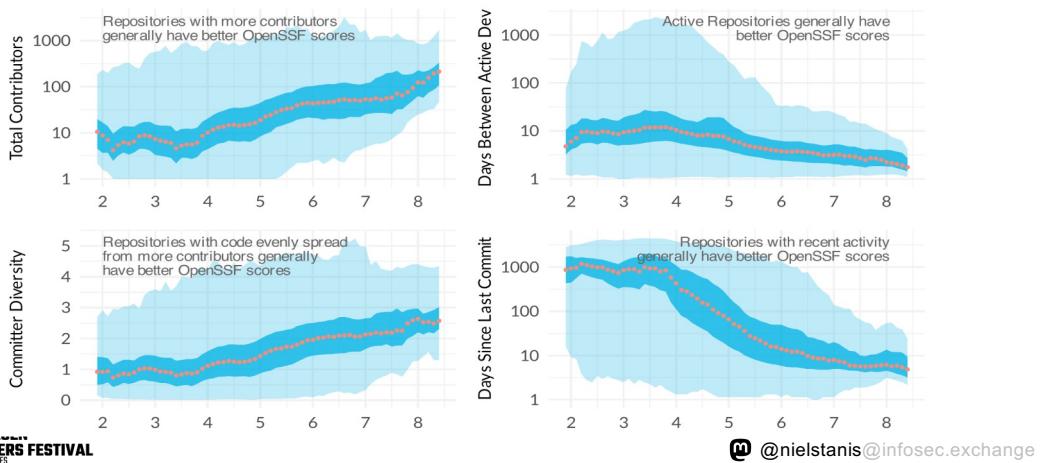
## Correlation between SOSS



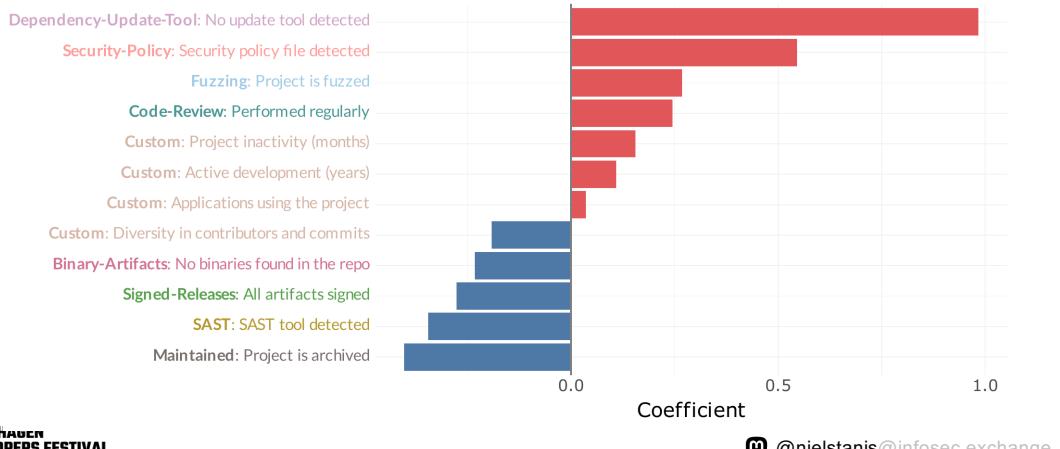
COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

# Github commits vs OpenSSF



# What really contributes to OSS Security?



COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

## What can we improve?



COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 @nielstanis@infosec.exchange

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

## 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?

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 @nielstanis@infosec.exchange

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

**Fuzzing .NET & SharpFuzz**

New & Improved!

Nemanja Mijailovic's Blog

## 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.

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

<https://mijailovic.net/2023/07/23/sharpfuzz-anniversary/>

The screenshot shows a blog post titled "Fuzzing .NET & SharpFuzz". A red starburst badge in the top right corner says "New & Improved!". The post content discusses the evolution of SharpFuzz, mentioning over 80 bugs found, including three CVEs: CVE-2019-0980, CVE-2019-0981, and CVE-2019-0982. It also highlights the discovery of correctness bugs. The footer includes links to the Copenhagen Developers Festival and the author's email.

**Fuzzing .NET & SharpFuzz**

**New & Improved!**

**Trophies**

The list of bugs found by SharpFuzz has been growing steadily and it now contains more than 80 entries. I'm pretty confident that some of the bugs in the .NET Core standard library would have been impossible to discover using any other testing method:

- BigInteger.TryParse out-of-bounds access
- Double.Parse throws AccessViolationException on .NET Core 3.0
- G17 format specifier doesn't always round-trip double values

As you can see, SharpFuzz is capable of finding not only crashes, but also correctness bugs—the more creative you are in writing your fuzzing functions, the higher your chances are for finding an interesting bug.

SharpFuzz can also find serious security vulnerabilities. I now have two CVEs in my trophy collection:

- CVE-2019-0980: .NET Framework and .NET Core Denial of Service Vulnerability
- CVE-2019-0981: .NET Framework and .NET Core Denial of Service Vulnerability

If you were ever wondering if fuzzing managed languages makes sense, I think you've got your answer right here.

**COPENHAGEN  
DEVELOPERS FESTIVAL**  
BY NDC CONFERENCES

**@nielstanis@infosec.exchange**

<https://mijailovic.net/2023/07/23/sharpfuzz-anniversary/>

## 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) { }
    });
}
```

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@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 | Retag | Subscribe | ...

**COPENHAGEN  
DEVELOPERS FESTIVAL**  
BY NDC CONFERENCES

**@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)



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

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@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");
        }
    }
}
```

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

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

## .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)

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 @nielstanis@infosec.exchange

The screenshot shows the Microsoft Application Inspector interface. At the top right, there is a red starburst badge with the text "New & Improved!". The main window displays "Application Inspector" and "Application Features". On the left, there's a sidebar titled "Feature Groups" with categories like "General Features", "Development", "Active Content", etc., each with corresponding icons. On the right, there's a section titled "Associated Rules" with a table showing rules for "Authentication: Microsoft (Identity)", "Authentication: General", and "Authentication: (OAuth)". A watermark for "COPENHAGEN DEVELOPERS FESTIVAL BY NDC CONFERENCES" is at the bottom left, and a Twitter handle "@nielstanis@infosec.exchange" is at the bottom right.

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

# Application Inspector



Feature	Confidence	Details
Authentication		<a href="#">View</a>
Authorization		<a href="#">View</a>
Cryptography		<a href="#">View</a>
Object Deserialization		N/A
AV Media Parsing		N/A
Dynamic Command Execution		N/A

**COPENHAGEN  
DEVELOPERS FESTIVAL**  
BY NDC CONFERENCES

[@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

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

The screenshot shows a web browser displaying the [Cargo Vet](https://mozilla.github.io/cargo-vet/) documentation. The page has a dark blue header with the title "Community Review" and a red starburst badge on the right that says "New & Improved!". The main content area is titled "Cargo Vet" and contains several sections of text and bullet points. On the left, there is a sidebar with a navigation menu. At the bottom left, there is a small logo for "COPENHAGEN DEVELOPERS FESTIVAL BY NDC CONFERENCES". At the bottom right, there is a social media handle "@nielstanis@infosec.exchange".

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

## 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!

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NUG CONFERENCE

 [@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

## Conclusion

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



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

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

## Questions?



COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

@nielstanis@infosec.exchange

## Links

- <https://github.com/nielstanis/cphdevfest2024/>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://www.fennec.dev> & <https://blog.fennec.dev>

COPENHAGEN  
DEVELOPERS FESTIVAL  
BY NDC CONFERENCES

 [@nielstanis@infosec.exchange](mailto:nielstanis@infosec.exchange)