





Reviewing NuGet Packages security easily using OpenSSF Scorecard

Niels Tanis
Sr. Principal Security Researcher

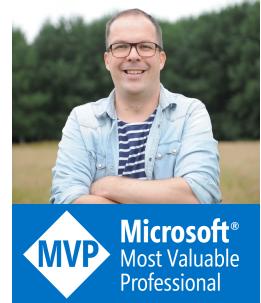


THE IT CONFERENCE FOR
MICROSOFT TECHNOLOGIES

Who am I?



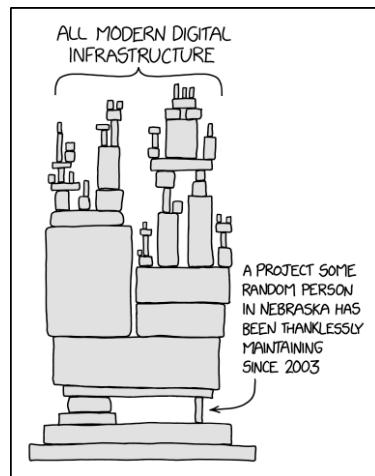
VERACODE



- 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

@nielstanis@infosec.exchange

Modern Application Architecture XKCD 2347



@nielstanis@infosec.exchange

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

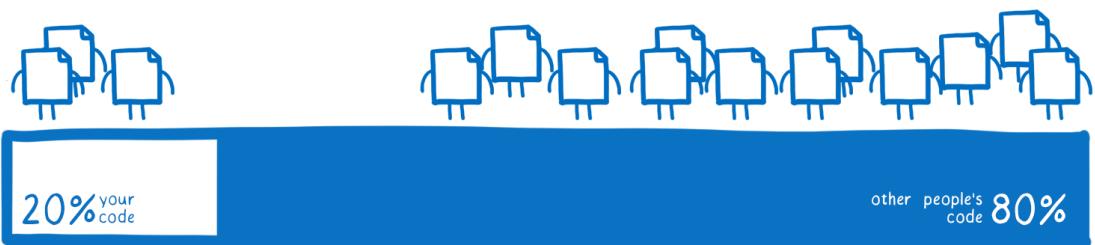
Agenda

- Risks in 3rd NuGet Package
- OpenSFF Scorecard
- New & Improved
- Conclusion - Q&A



 @nielstanis@infosec.exchange

Average codebase composition



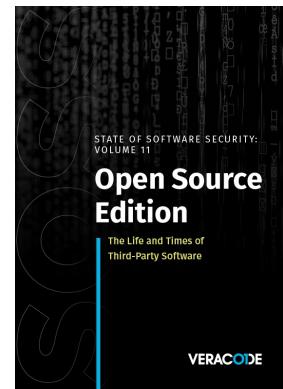
@nielstanis@infosec.exchange

<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."



@nielstanis@infosec.exchange



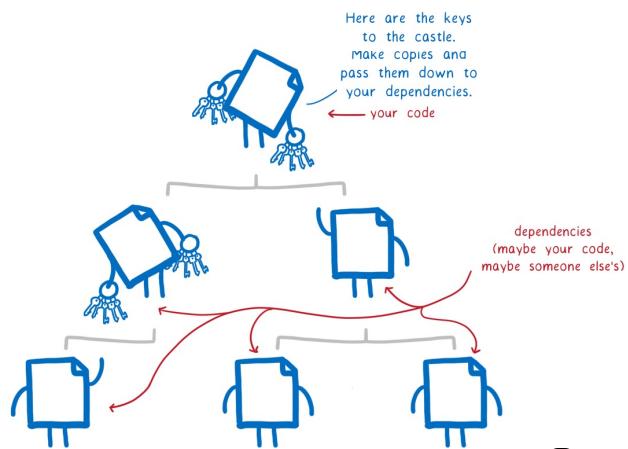
State of Log4j - 2 years later

- 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>

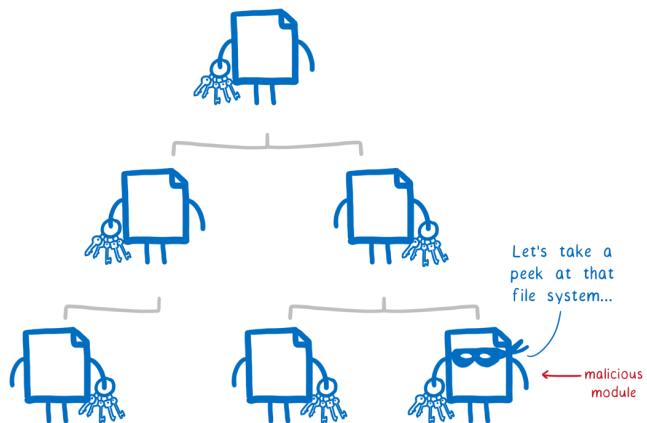
Average codebase composition



@nielstanis@infosec.exchange

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

Malicious Assembly



 @nielstanis@infosec.exchange

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

The screenshot shows a news article titled "Hackers target .NET developers with malicious NuGet packages" from BleepingComputer.com. The article discusses threat actors targeting .NET developers by injecting cryptocurrency stealers into the NuGet repository and impersonating legitimate packages through typosquatting. It quotes researchers from JFrog security and provides details on the attack's success and methods. The article is dated March 20, 2023, at 03:22 PM.

Malicious Package

Hackers target .NET developers with malicious NuGet packages

By Sergiu Gatlan | March 20, 2023 | 03:22 PM | 0 comments

Threat actors are targeting and infecting .NET developers with cryptocurrency stealers delivered through the NuGet repository and impersonating multiple legitimate packages via typosquatting.

Three of them have been downloaded over 150,000 times within a month, according to JFrog security researchers Natan Nehorai and Brian Moussalli, who spotted this ongoing campaign.

While the massive number of downloads could point to a large number of .NET developers who had their systems compromised, it could also be explained by the attackers' efforts to legitimize their malicious NuGet packages.

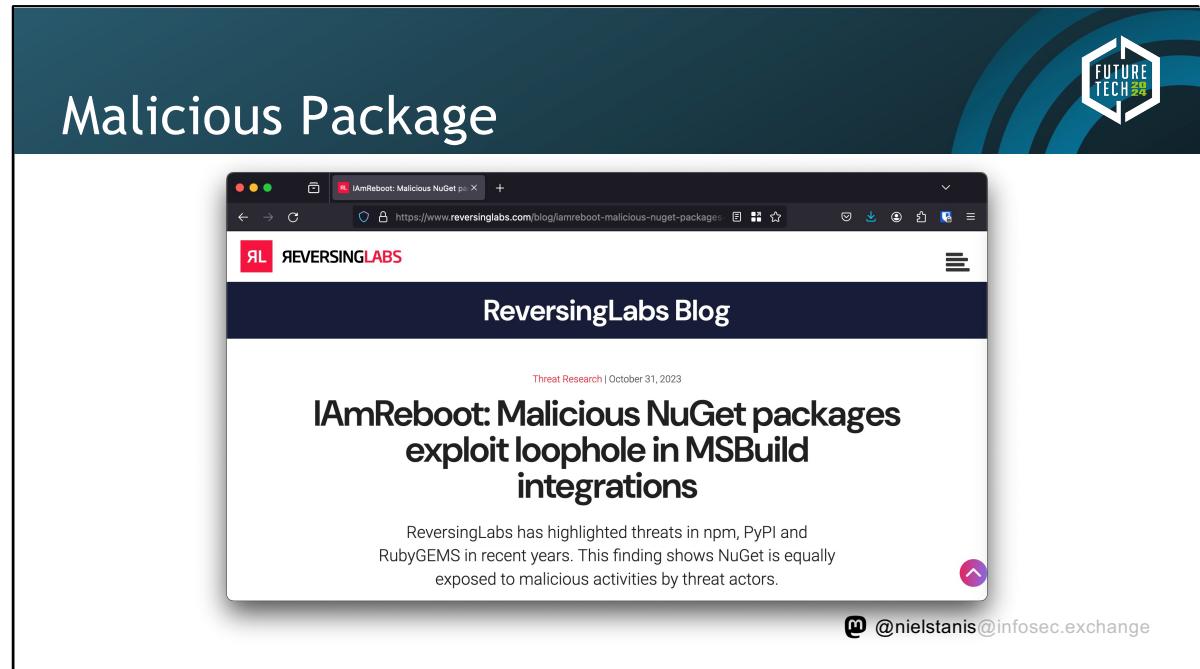
"The top three packages were downloaded an incredible amount of times – this could be an indicator that the attack was highly successful, infecting a large amount of machines," the [JFrog security researchers said](#).

"However, this is not a fully reliable indicator of the attack's success since the attackers could have automatically inflated the download count (with bots) to make the packages seem more legitimate."

The threat actors also used typosquatting when creating their NuGet repository profiles to impersonate

@nielstanis@infosec.exchange

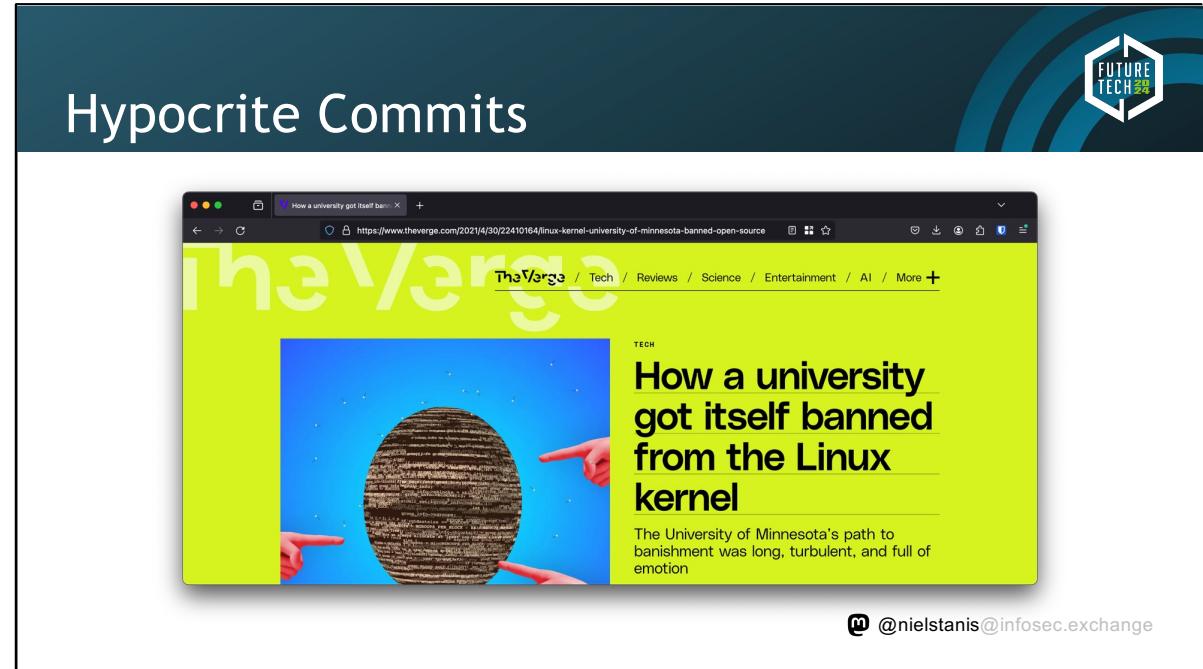
[https://www.bleepingcomputer.com/news/security/hackers-target-net-developers-with-malicious-nuget-packages//](https://www.bleepingcomputer.com/news/security/hackers-target-net-developers-with-malicious-nuget-packages/)



<https://www.reversinglabs.com/blog/iamreboot-malicious-nuget-packages-exploit-msbuild-loophole>

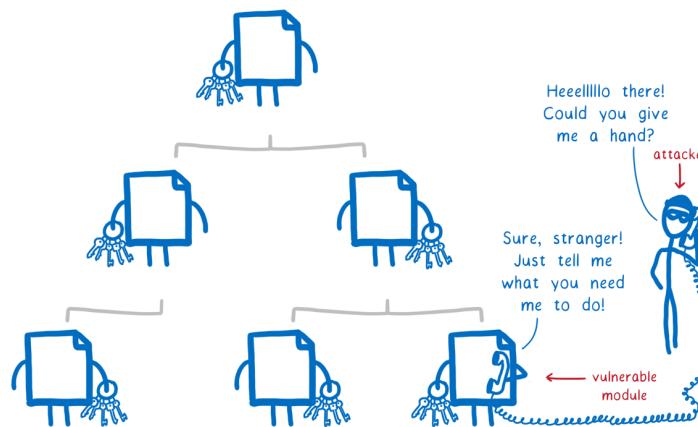
The screenshot shows a web browser displaying an Ars Technica article. The title of the article is "XZ Backdoor". The main headline reads: "Backdoor found in widely used Linux utility targets encrypted SSH connections". Below the headline, a sub-headline states: "Malicious code planted in xz Utils has been circulating for more than a month." The author's name is DAN GOODIN, and the date is 3/29/2024, 7:50 PM. The Ars Technica logo is visible at the top left, and there are navigation links like "SUBSCRIBE", "SIGN IN", and search functions at the top right. The overall layout is clean and professional.

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



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

Vulnerable Assembly



@nielstanis@infosec.exchange

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

Vulnerabilities in Libraries

A screenshot of a GitHub issue page. The URL in the address bar is https://github.com/dotnet/announcements/issues/288. The page title is "Microsoft Security Advisory CVE-2023-36558: .NET Security Feature Bypass Vulnerability". The issue was opened by rbhanda on Nov 14 and has 0 comments. The main content area contains a summary of the vulnerability, stating that an unauthenticated user can bypass validation on Blazor server forms. A link to the discussion in the runtime repository is provided. On the right side, there is a sidebar with project management details: Assignees (None assigned), Labels (Security selected), Projects (None yet), Milestone (None), Development (No branches or pull requests), and Notifications (Customize).

Microsoft Security Advisory CVE-2023-36558: .NET Security Feature Bypass Vulnerability

rbhanda commented on Nov 14 · edited

Microsoft Security Advisory **CVE-2023-36558: .NET Security Feature Bypass Vulnerability**

Executive summary

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.

Discussion

Discussion for this issue can be found at [dotnet/runtime#94726](#)

Assignees
No one assigned

Labels
Security

Projects
None yet

Milestone
None

Development
No branches or pull requests

Notifications
Customize

@nielstanis@infosec.exchange

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

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 $
```

@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
```

@nielstanis@infosec.exchange

DotNet CLI

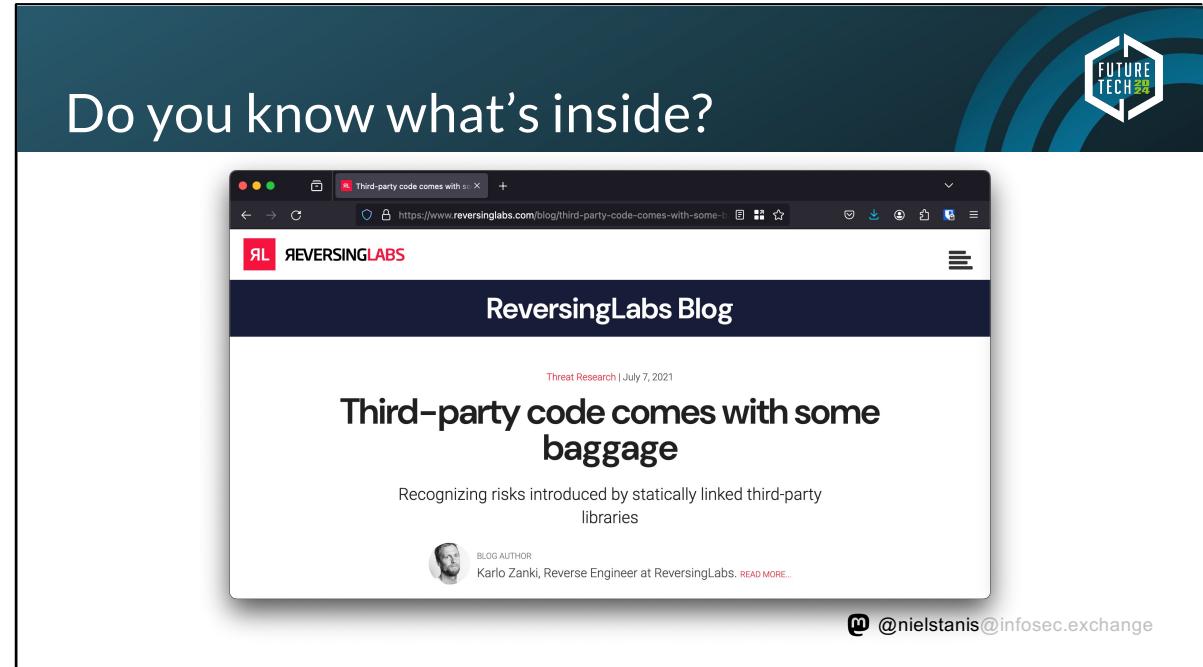


```
nelson@ghost-m2 ~/research/consoleapp $ dotnet list package --vulnerable --include-transitive
The following sources were used:
https://f.feedz.io/fennec/docgenerator/nuget/index.json
https://api.nuget.org/v3/index.json

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 $
```

@nielstanis@infosec.exchange



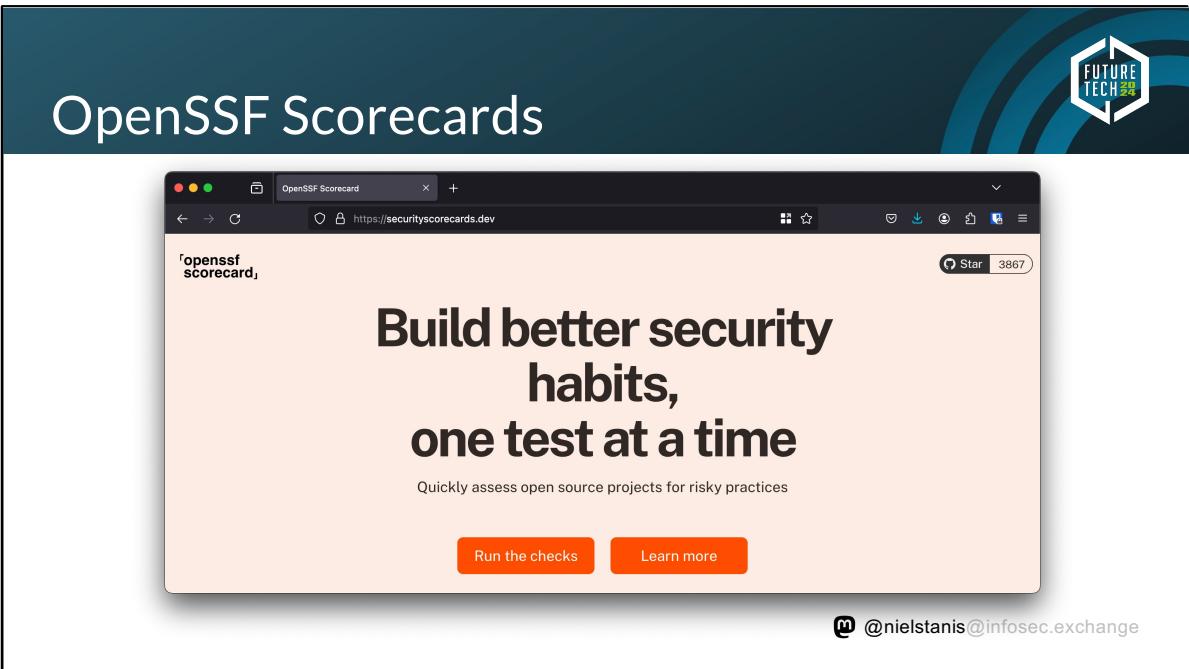
<https://www.reversinglabs.com/blog/third-party-code-comes-with-some-baggage>

Nutrition Label for Software?



@nielstanis@infosec.exchange

<https://securityscorecards.dev/>



<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" (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 three columns: 1) A section about what the scorecard does, mentioning automated checks and OSS developer creation. 2) A section about its purpose, mentioning proactive assessment and improving codebases. 3) A section about best practices enforcement. At the bottom right of the content area are two small icons: a red circle with a white dot and a grid of squares.

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:

@nielstanis@infosec.exchange

<https://securityscorecards.dev/>

The screenshot shows a web browser window with the title "OpenSSF Security Scorecards" at the top. In the top right corner of the slide, there is a logo for "FUTURE TECH 2024". The browser's address bar shows the URL <https://securityscorecards.dev/#how-it-works>. The main content area is titled "How it works". 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". To the right of these sections, there is explanatory text about how the scorecard checks for vulnerabilities across the software supply chain and how automated checks return a score out of 10 and a risk level. It also mentions that the tool provides remediation prompts. Below this text are three horizontal bars representing risk levels: "CRITICAL RISK" (value 10), "HIGH RISK" (value 7.5), and "MEDIUM RISK" (value 5). At the bottom right of the slide, there is a small icon and the text "@nielstanis@infosec.exchange".

<https://securityscorecards.dev/>

OpenSSF Security Scorecards

A screenshot of a web browser displaying the OpenSSF Security Scorecard website at <https://securityscorecards.dev/#the-checks>. The page features a large diagram in the center consisting of five overlapping circles. The circles are labeled: "CODE VULNERABILITIES" (top), "BUILD RISK ASSESSMENT" (left), "MAINTENANCE" (right), "SOURCE RISK ASSESSMENT" (bottom-left), and "CONTINUOUS TESTING" (bottom-right). In the center of these overlapping circles is the text "HOLISTIC SECURITY PRACTICES" in red capital letters. To the left of the diagram, there is a sidebar with 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
- The checks**
- Use cases
- About the project name
- Part of the OSS community
- Get involved

@nielstanis@infosec.exchange

<https://securityscorecards.dev/>

Code Vulnerabilities (High)



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

ID	Packages	Summary	Affected versions	Published	Fix
GHSA-x674-1x45-1eww	NuGet/Microsoft.Identity.Client	MSAL.NET applications targeting Xamarin Android and .NET Android (MAUI) susceptible to local denial of service	4.46.0 4.49.0 4.50.0 4.52.0	4.48.1 4.49.1 4.51.0 —	9 hours ago Fix available
GHSA-5x7m-6727-26ct	NuGet/SixLabors.ImageSharp	SixLabors.ImageSharp vulnerable to data leakage	1.0.0 1.0.0-beta0002 1.0.0-beta0004 1.0.0-beta0006	1.0.0-beta0001 1.0.0-beta0003 1.0.0-beta0005 —	yesterday Fix available
GHSA-q85r-6x2g-45w7	NuGet/SixLabors.ImageSharp	SixLabors.ImageSharp vulnerable to Memory Allocation with Excessive Size Value	1.0.0 1.0.0-beta0002 1.0.0-beta0004 1.0.0-beta0006	1.0.0-beta0001 1.0.0-beta0003 1.0.0-beta0005 —	yesterday Fix available

@nielstanis@infosec.exchange

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

Maintenance Dependency-Update-Tool (**High**)



- This check tries to determine if the project uses a dependency update tool, use of: Dependabot, Renovate bot
- Out-of-date dependencies make a project vulnerable to known flaws and prone to attacks.

 @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.

 @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.

 @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)



- This check tries to determine if the project runs 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).

 @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;
 - If there are user-defined language-specified fuzzing functions in the repository.
- Does it make sense to do fuzzing on .NET projects?

 @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!

 @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!

 @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!

 @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/>

 @nielstanis@infosec.exchange

Source Risk Assessment Code Review (Low)



- This check determines whether the project requires human code review before pull requests (merge requests) are merged.
- The check determines whether the most recent changes (over the last ~30 commits) have an approval on GitHub or if the merger is different from the committer (implicit review)

 @nielstanis@infosec.exchange

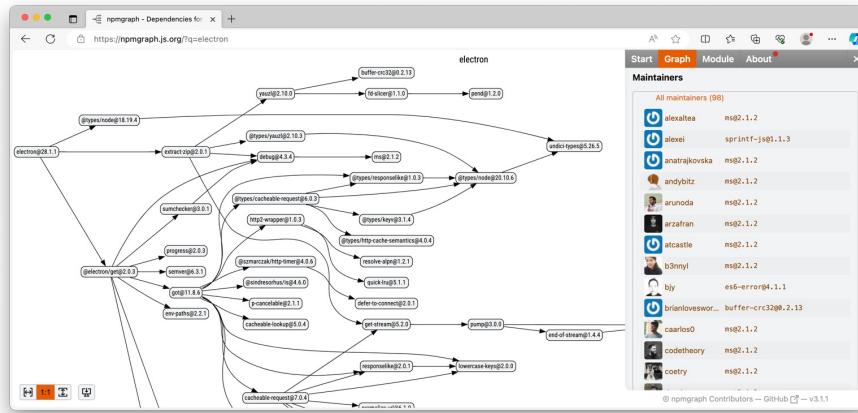
Source Risk Assessment Contributors (Low)



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

 @nielstanis@infosec.exchange

Source Risk Assessment Contributors (Low)



 @nielstanis@infosec.exchange

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



- This check tries to determine if the project pins 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?

 @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.

 @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.

 @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

 @nielstanis@infosec.exchange

Demo OpenSSF Scorecard Fennec CLI



Running checks



 @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

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



What can we improve?



 @nielstanis@infosec.exchange

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

Fuzzing .NET



- Fuzzing, or fuzz testing, is defined as an automated software testing method that uses a wide range of **invalid** and unexpected data as input to find flaws in the software undergoing the test.
- Used a lot for finding C/C++ memory issues
- Can it be of any value with managed languages like .NET?

 @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.

@nielstanis@infosec.exchange

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

The screenshot shows a blog post titled "Fuzzing .NET & SharpFuzz". The header features a "New & Improved!" badge and a "FUTURE TECH 2024" logo. The main content is a screenshot of a web browser displaying a page about SharpFuzz trophies. The browser window has a title bar showing "Five years of fuzzing .NET with SharpFuzz" and the URL "https://mijailovic.net/2023/07/23/sharpfuzz-anniversary/". The page content includes a section titled "Trophies" with a list of bugs found by SharpFuzz, two CVE entries, and a note about finding security vulnerabilities. At the bottom, there's a link to a GitHub repository. A footer at the bottom of the screenshot shows the URL again and a social media handle "@nielstanis@infosec.exchange".

Fuzzing .NET & SharpFuzz

New & Improved!

FUTURE TECH 2024

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.

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

@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) { }
    });
}
```

@nielstanis@infosec.exchange

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

Fuzzomatic: Using AI to Fuzz Rust

New & Improved!

FUTURE TECH 2024

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

@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);
}
```

@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>

.NET Reproducibility

New & Improved!



- Reproducible builds are a set of software development practices that create an independently-verifiable path from source to binary code.
- .NET Roslyn Deterministic Inputs
- How reproducible is a simple console app?

 @nielstanis@infosec.exchange

The screenshot shows the Microsoft Application Inspector application window. The title bar says "Application Inspector". In the top right corner, there is a red starburst badge with the text "New & Improved!". The main content area is titled "Application Features". It contains a sidebar with "Feature Groups" such as "Select Features", "General Features", "Development", "Active Content", "Data Storage", "Sensitive Data", "Cloud Services", "OS Integration", "OS System Changes", and "Other". To the right of the sidebar, there is a table titled "Associated Rules" with columns for "Name" and "Authentication". The table lists three rows: "Authentication: Microsoft (Identity)", "Authentication: General", and "Authentication: (OAuth)". At the bottom right of the window, there is a footer with a mail icon and the email address "@nielstanis@infosec.exchange".

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

The screenshot shows the Microsoft Application Inspector interface. At the top, there's a dark header with the title "Application Inspector". To the right of the title is a red circular badge with the text "New & Improved!". Below the header is a navigation bar with the text "Select Features". The main content area is a table with the following columns: "Feature", "Confidence", and "Details". The table lists six features:

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

At the bottom right of the interface, there's a footer with the text "@nielstanis@infosec.exchange" next to a small profile icon.

<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 "New & Improved!" badge. The main content area is titled "Cargo Vet" and contains an introduction to the tool. It explains that `cargo vet` is a tool to help projects ensure that third-party Rust dependencies have been audited by a trusted entity. The tool is described as being lightweight and easy to integrate. The page also lists several key features:

- **Sharing:** Public crates are often used by many projects. These projects can share their findings with each other to avoid duplicating work.
- **Relative Audits:** Different versions of the same crate are often quite similar to each other. Developers can inspect the difference between two versions, and record that if the first version was vetted, the second can be considered vetted as well.
- **Deferred Audits:** It is not always practical to achieve full coverage. Dependencies can be added to a list of exceptions which can be ratcheted down over time. This makes it trivial to introduce `cargo vet` to a new project and guard against future vulnerabilities while vetting the

At the bottom right of the page, there is a footer with the text "nielstanis@infosec.exchange".

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

Conclusion



- Scorecard helps out to security review 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!
- NuGet Package Scoring (NET Score)
- Room for .NET specific improvements with Fennec CLI & contributions to OpenSSF Scorecard project



 @nielstanis@infosec.exchange

Questions?



 @nielstanis@infosec.exchange



Links

- <https://github.com/nielstanis/futuretech2024/>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://www.fennec.dev> & <https://blog.fennec.dev>

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

