



THE IT CONFERENCE FOR
MICROSOFT TECHNOLOGIES

Sponsors



Partners





Reviewing NuGet Packages security easily using OpenSSF Scorecard

Niels Tanis
Sr. Principal Security Researcher



THE IT CONFERENCE FOR
MICROSOFT TECHNOLOGIES

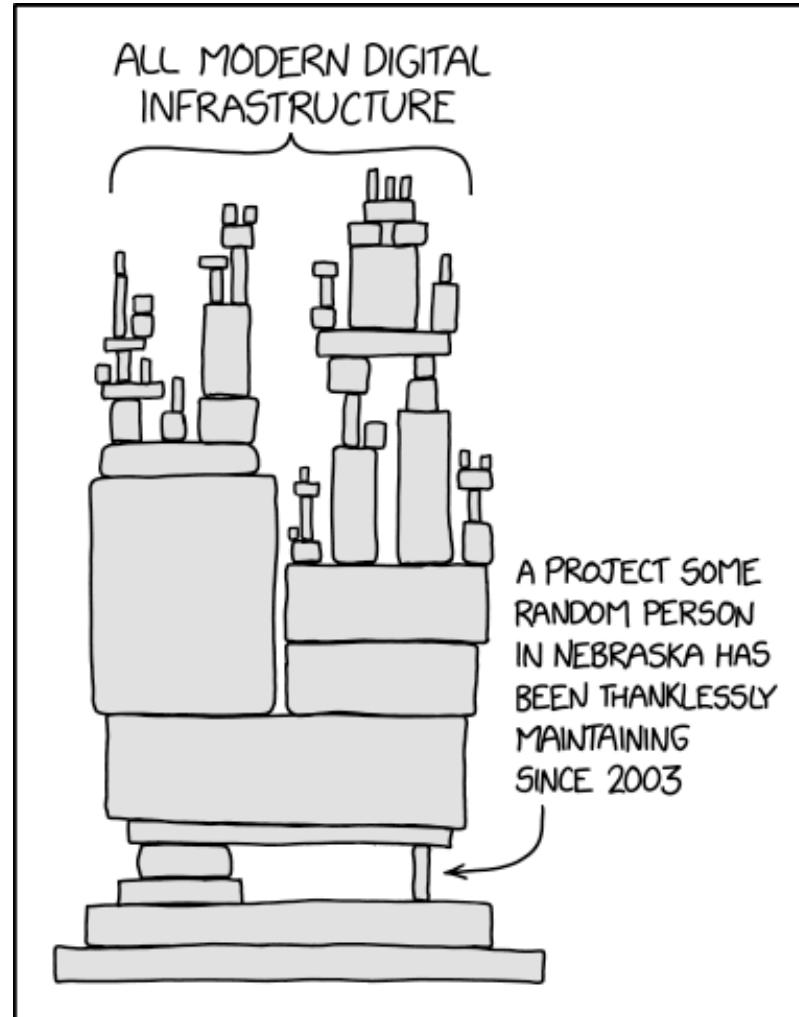
Who am I?

- 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

The Veracode logo, consisting of the word "VERACODE" in a bold, sans-serif font where the "O" is blue.

Modern Application Architecture

XKCD 2347

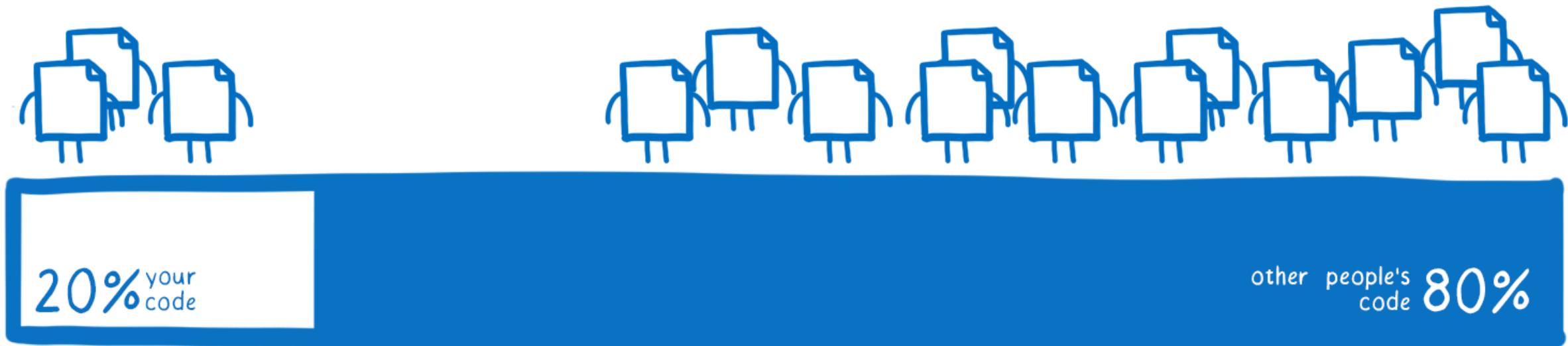


Agenda

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

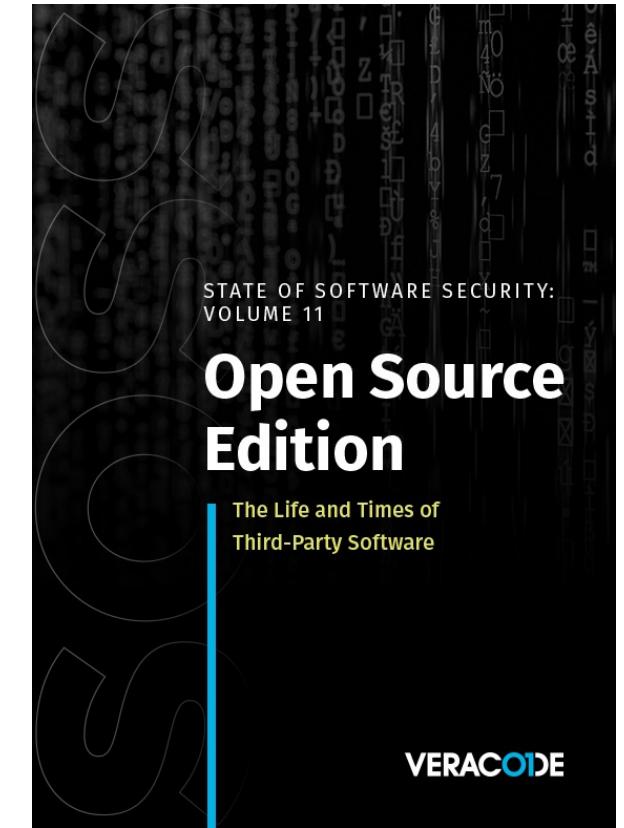


Average codebase composition



State of Software Security v11

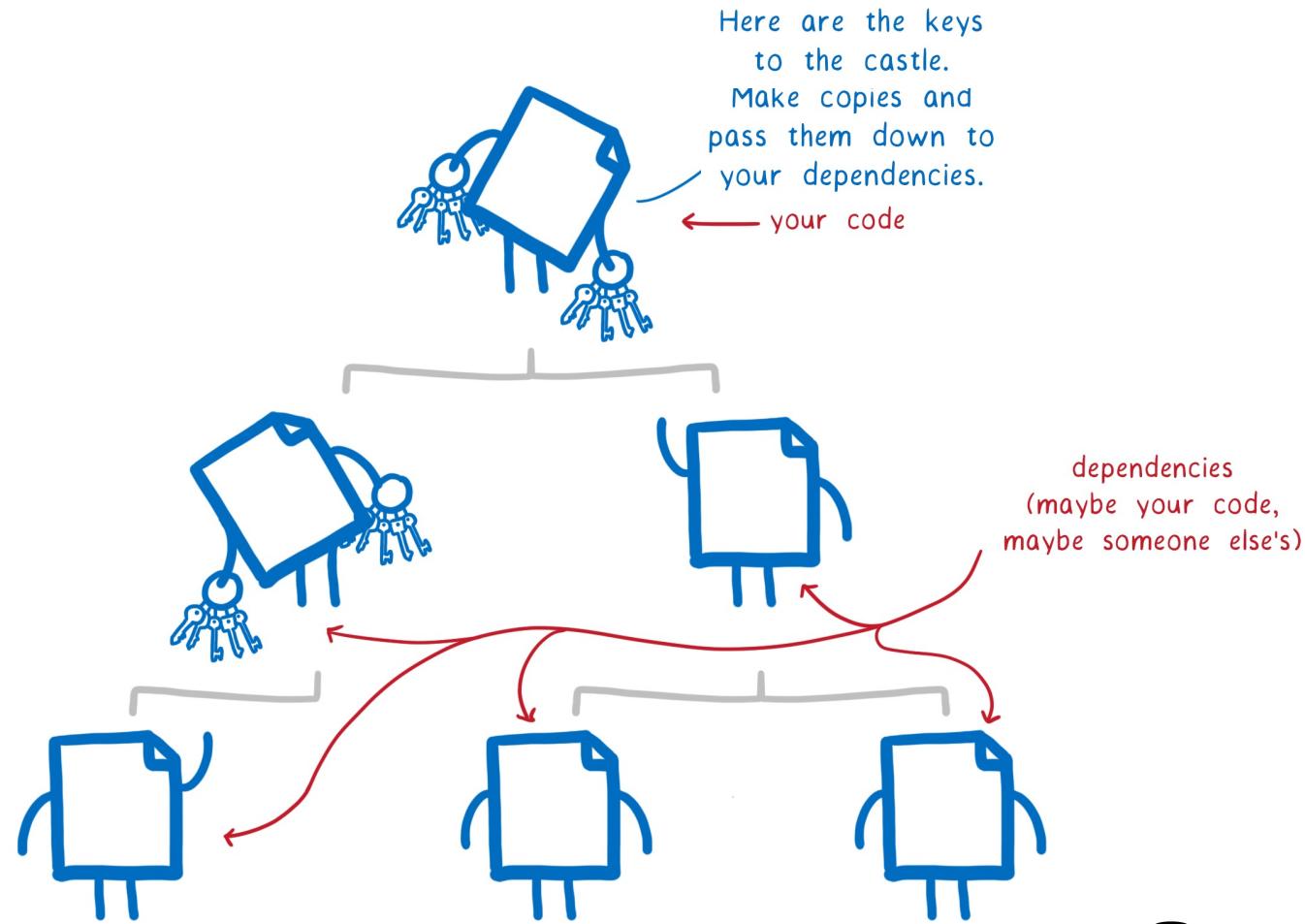
*“Despite this dynamic landscape,
79 percent of the time, developers
never update third-party libraries after
including them in a codebase.”*



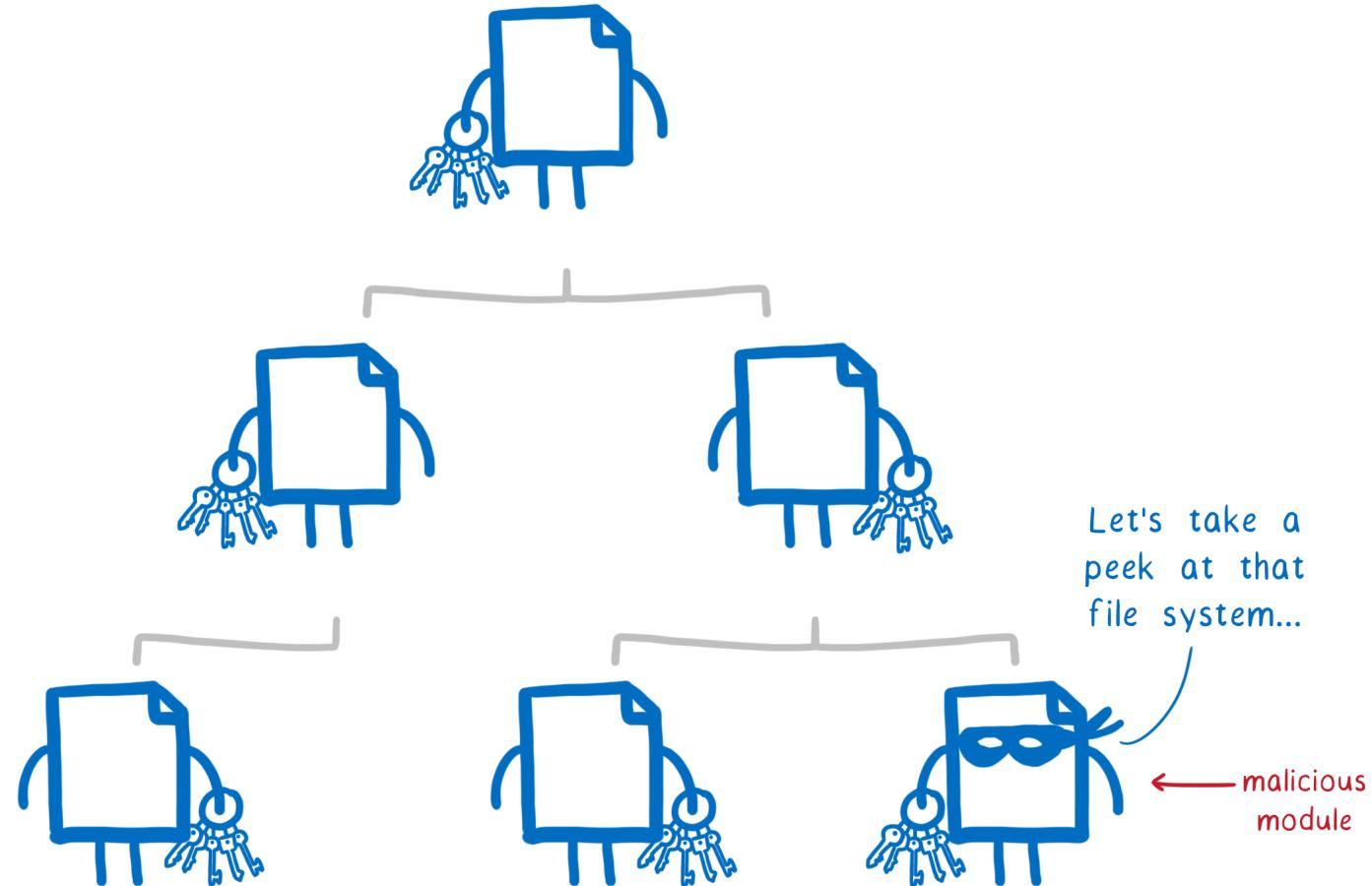
State of Log4j - 2 years later

- 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.

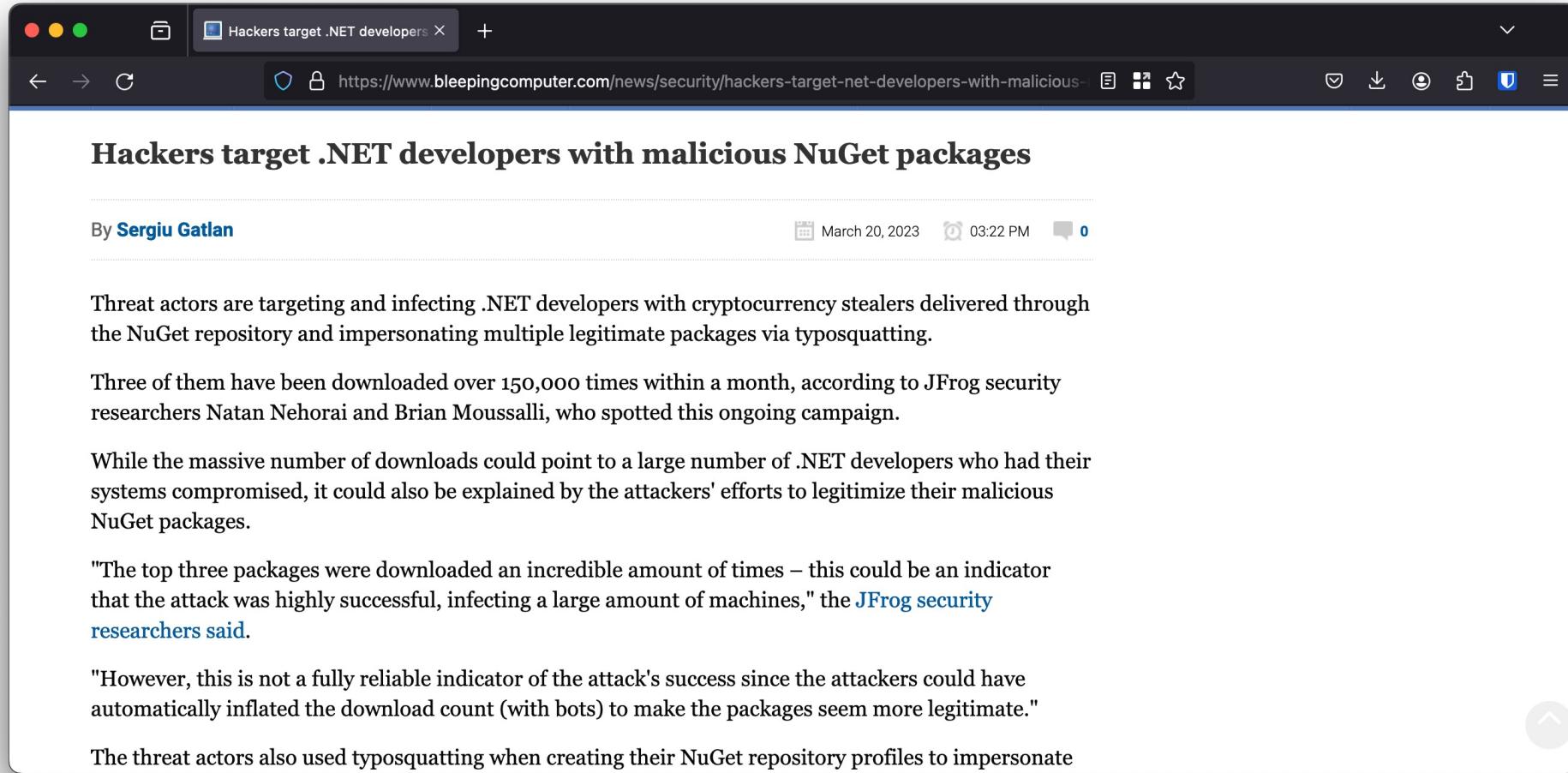
Average codebase composition



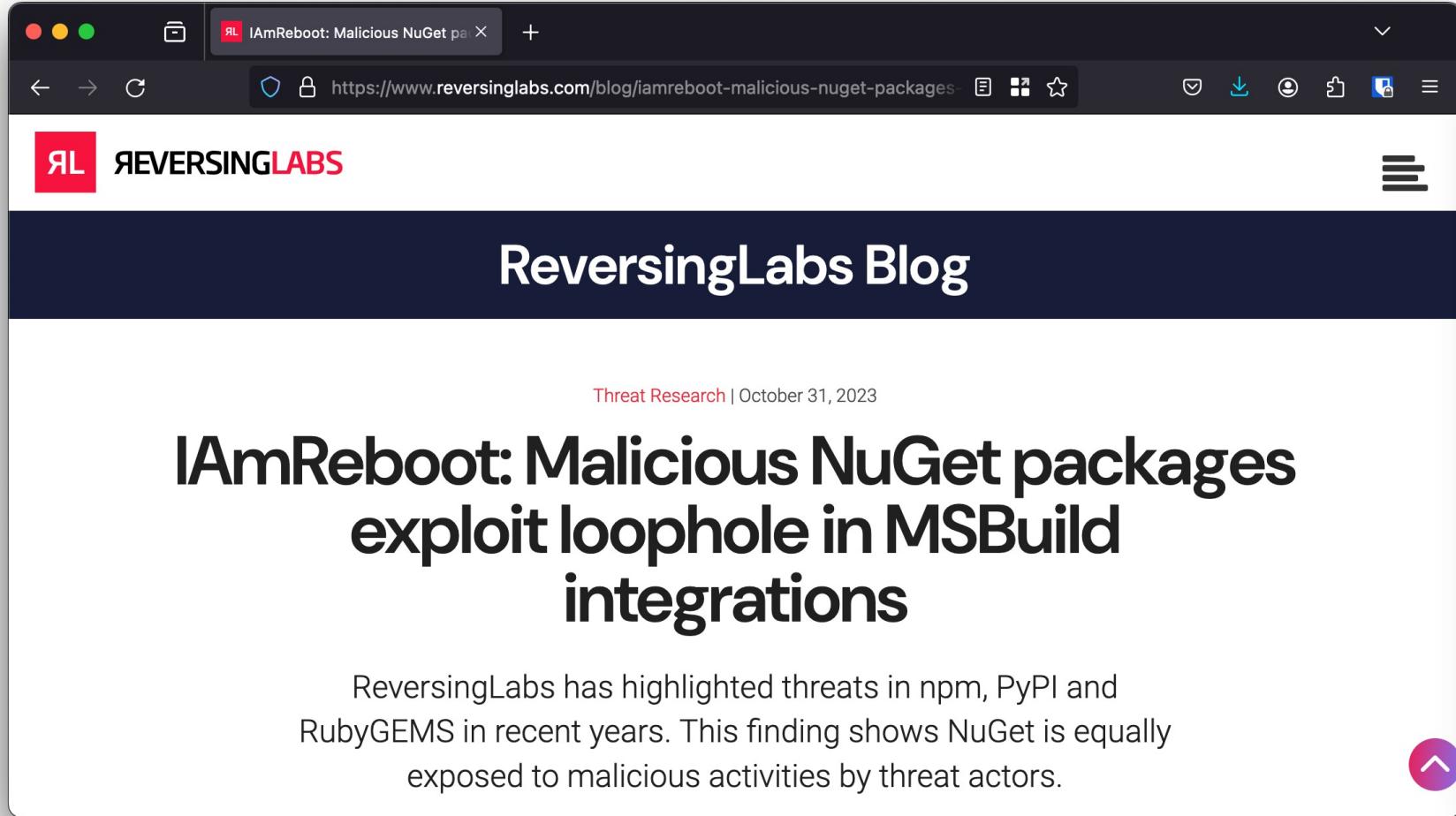
Malicious Assembly



Malicious Package

A screenshot of a web browser window showing a news article from BleepingComputer.com. The title of the article is "Hackers target .NET developers with malicious NuGet packages". The article 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 and quotes JFrog security researchers Natan Nehorai and Brian Moussalli. The browser interface includes a navigation bar, a search bar, and various icons for file operations.

Malicious Package



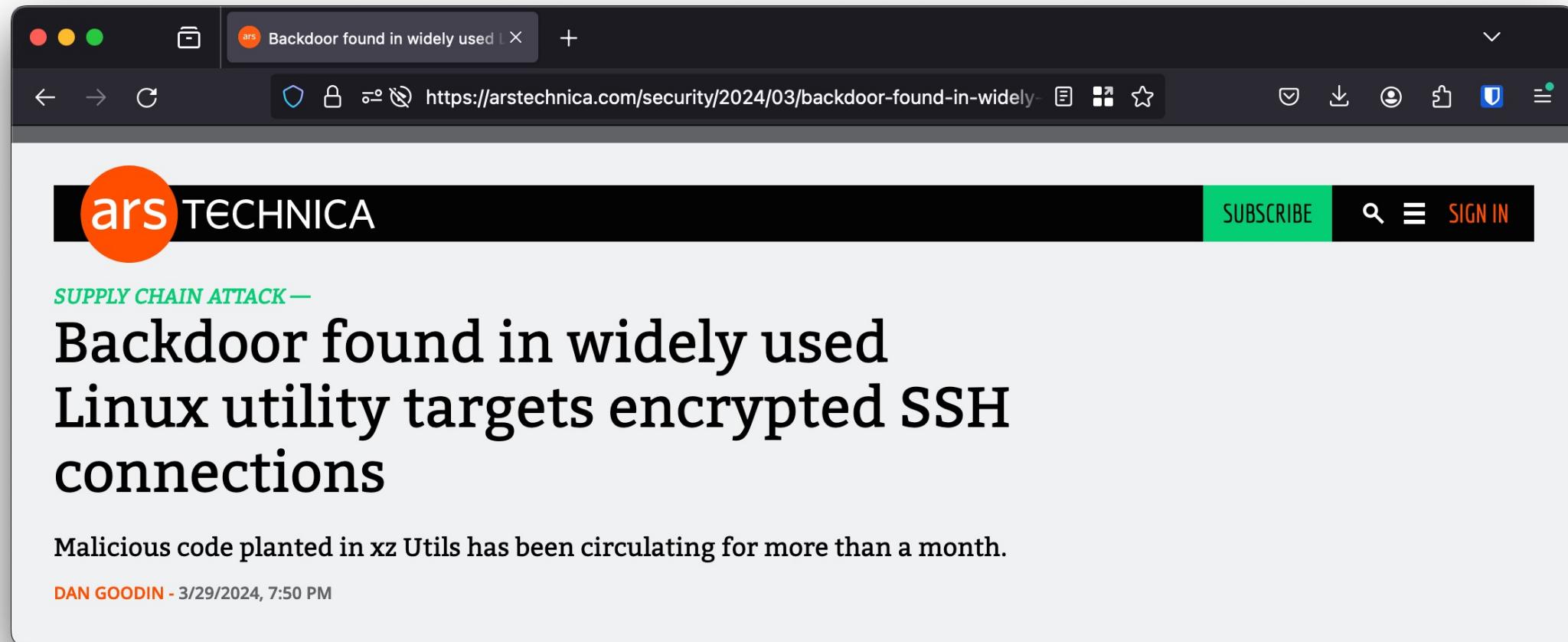
A screenshot of a web browser window showing a blog post from ReversingLabs. The title of the post is "IAmReboot: Malicious NuGet packages exploit loophole in MSBuild integrations". The post discusses threats in npm, PyPI, and RubyGEMS, stating that NuGet is also exposed to malicious activities. The browser's address bar shows the URL <https://www.reversinglabs.com/blog/iamreboot-malicious-nuget-packages>.

Threat Research | October 31, 2023

IAmReboot: Malicious NuGet packages exploit loophole in MSBuild integrations

ReversingLabs has highlighted threats in npm, PyPI and RubyGEMS in recent years. This finding shows NuGet is equally exposed to malicious activities by threat actors.

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 discusses a supply chain attack where malicious code was planted in xz Utils. The browser's address bar shows the URL: <https://arstechnica.com/security/2024/03/backdoor-found-in-widely-used-linux-utility-targets-encrypted-ssh-connections/>.

ars TECHNICA

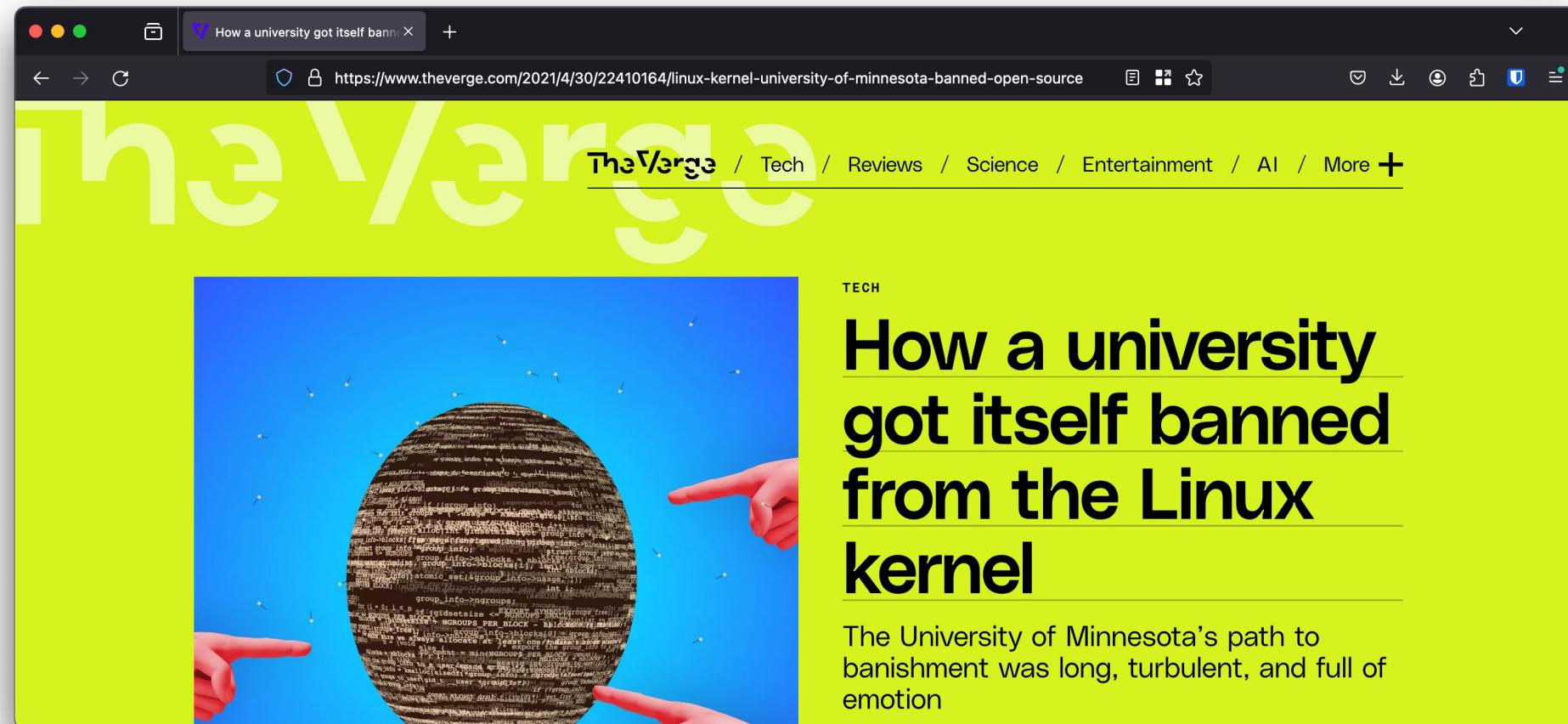
SUPPLY CHAIN ATTACK —

Backdoor found in widely used Linux utility targets encrypted SSH connections

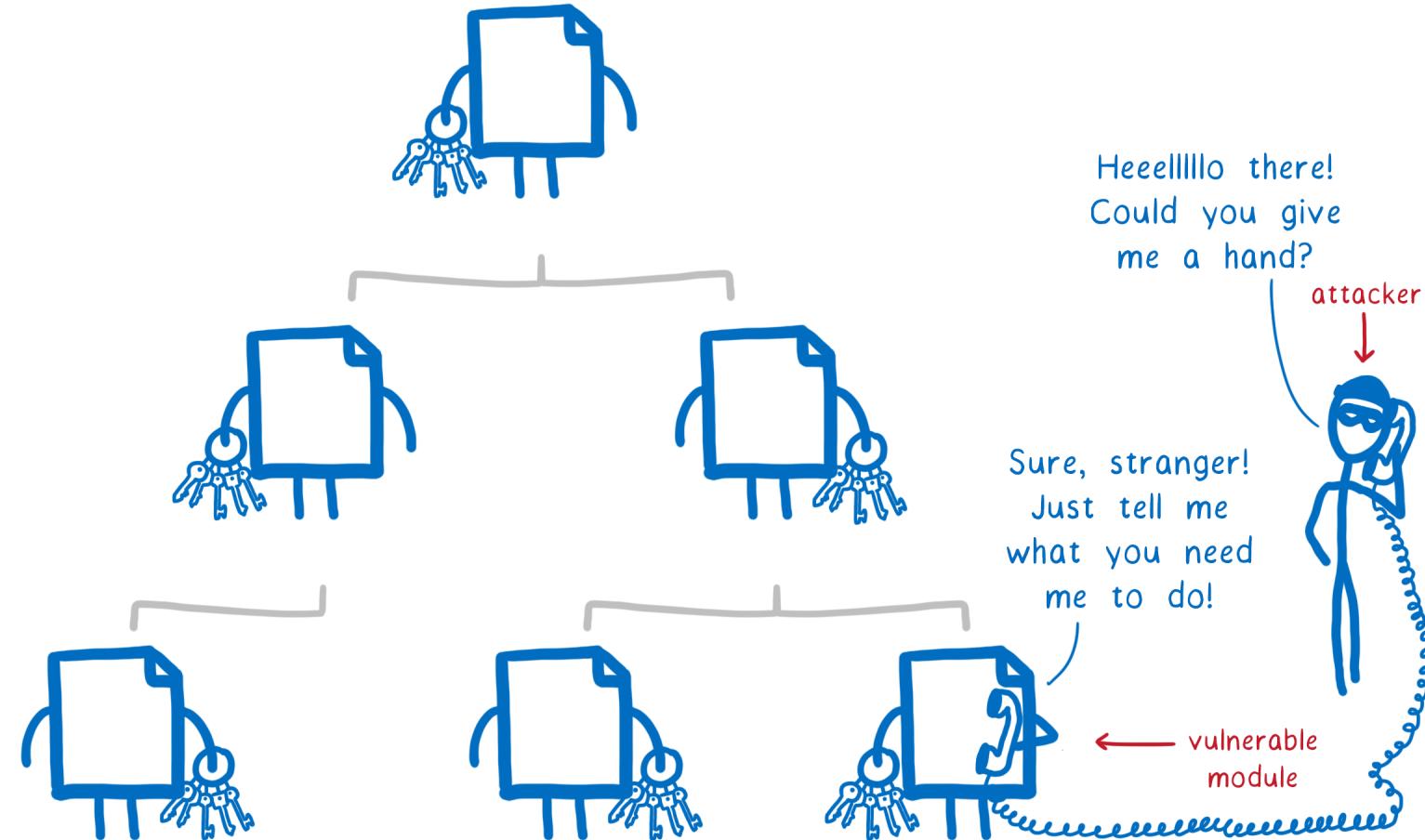
Malicious code planted in xz Utils has been circulating for more than a month.

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

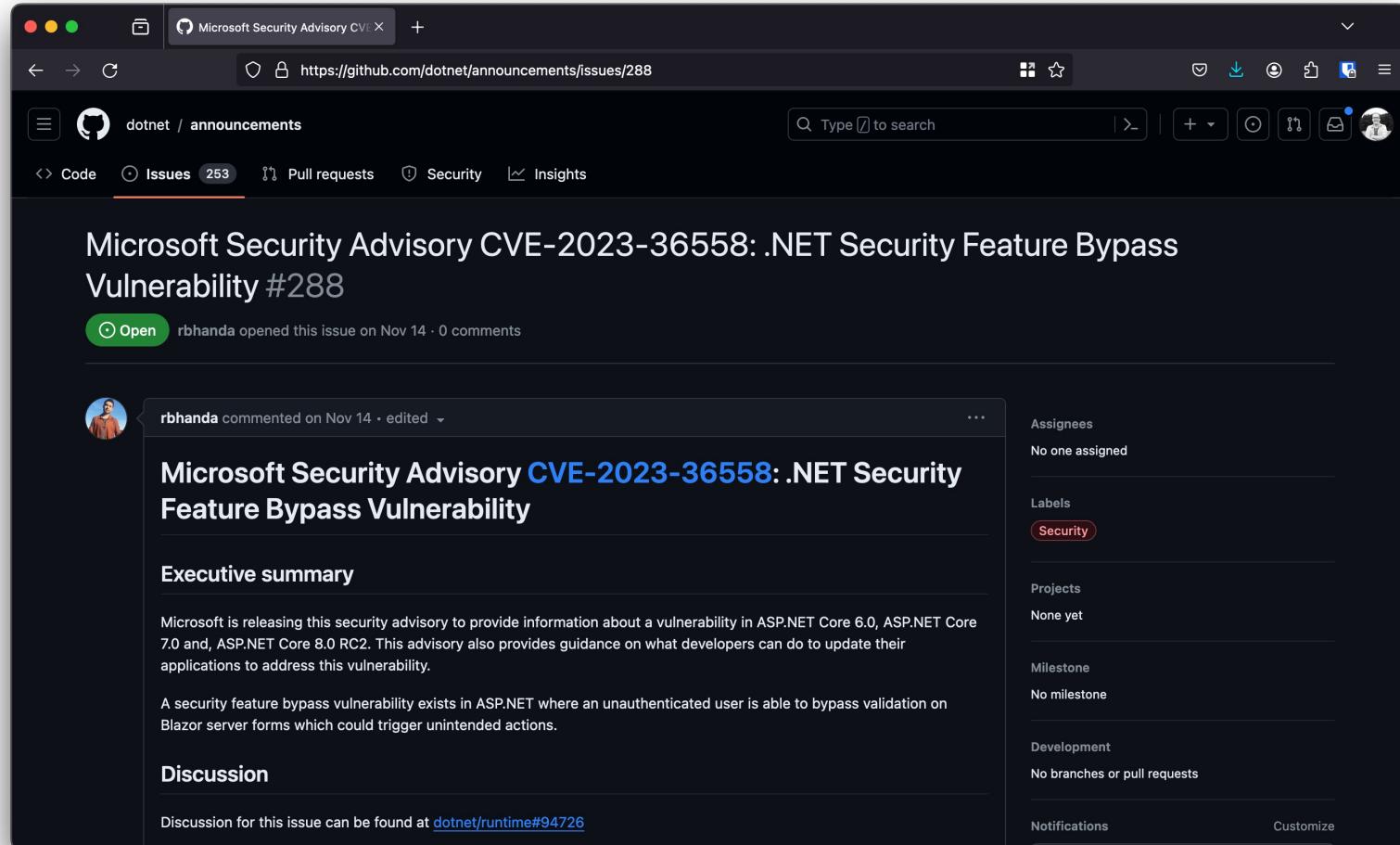
Hypocrite Commits



Vulnerable Assembly



Vulnerabilities in Libraries



The screenshot shows a GitHub issue page for Microsoft Security Advisory CVE-2023-36558: .NET Security Feature Bypass Vulnerability. The page is titled "Microsoft Security Advisory CVE-2023-36558: .NET Security Feature Bypass Vulnerability #288". The issue was opened by rbhanda on Nov 14, with 0 comments. A comment from rbhanda on Nov 14, edited, discusses the vulnerability. The comment text is identical to the advisory title. The advisory text states: "Microsoft is releasing this security advisory to provide information about a vulnerability in ASP.NET Core 6.0, ASP.NET Core 7.0 and, ASP.NET Core 8.0 RC2. This advisory also provides guidance on what developers can do to update their applications to address this vulnerability. A security feature bypass vulnerability exists in ASP.NET where an unauthenticated user is able to bypass validation on Blazor server forms which could trigger unintended actions." The issue page includes sections for Executive summary, Discussion, and various project management details like Assignees, Labels, Projects, Milestone, Development, and Notifications.

DotNet CLI

```
nelson@ghost-m2 ~$ 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 ~$ 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 ~$
```

DotNet CLI

```
nelson@ghost-m2 ~$ 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

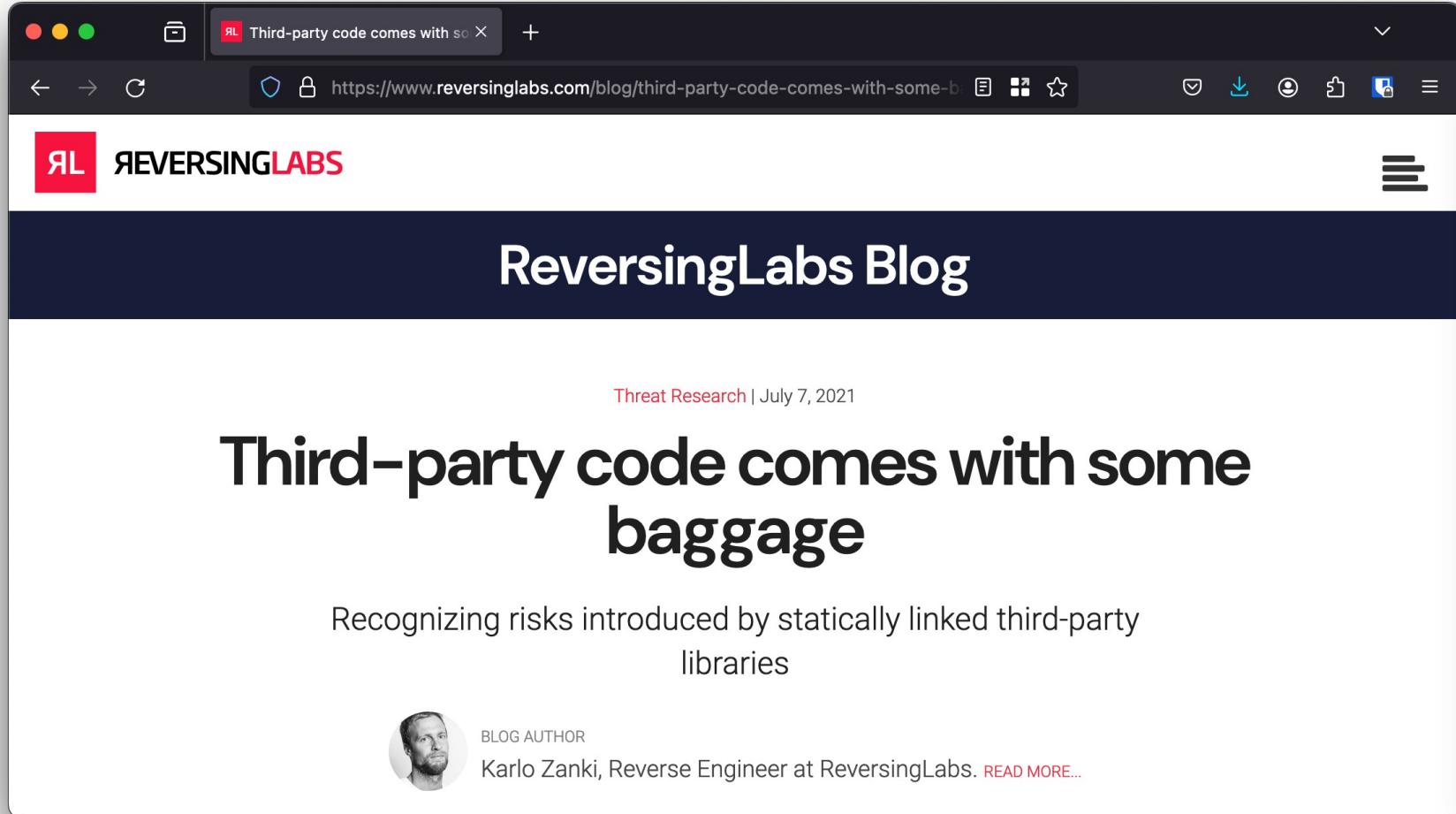
```
nelson@ghost-m2:~/research/consoleapp$ dotnet list package --vulnerable --include-transitive

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 mode interface. The tab bar shows a single tab titled "Third-party code comes with some baggage". The address bar displays the URL "https://www.reversinglabs.com/blog/third-party-code-comes-with-some-ba...". The main content area features the ReversingLabs logo (a red square with "RL") and the text "REVERSINGLABS". Below this is a dark header bar with the text "ReversingLabs Blog". The main article title is "Third-party code comes with some baggage", with a subtitle "Recognizing risks introduced by statically linked third-party libraries". At the bottom, there is a profile picture of a man, a "BLOG AUTHOR" label, and the name "Karlo Zanki, Reverse Engineer at ReversingLabs. [READ MORE...](#)".

Threat Research | July 7, 2021

Third-party code comes with some baggage

Recognizing risks introduced by statically linked third-party libraries

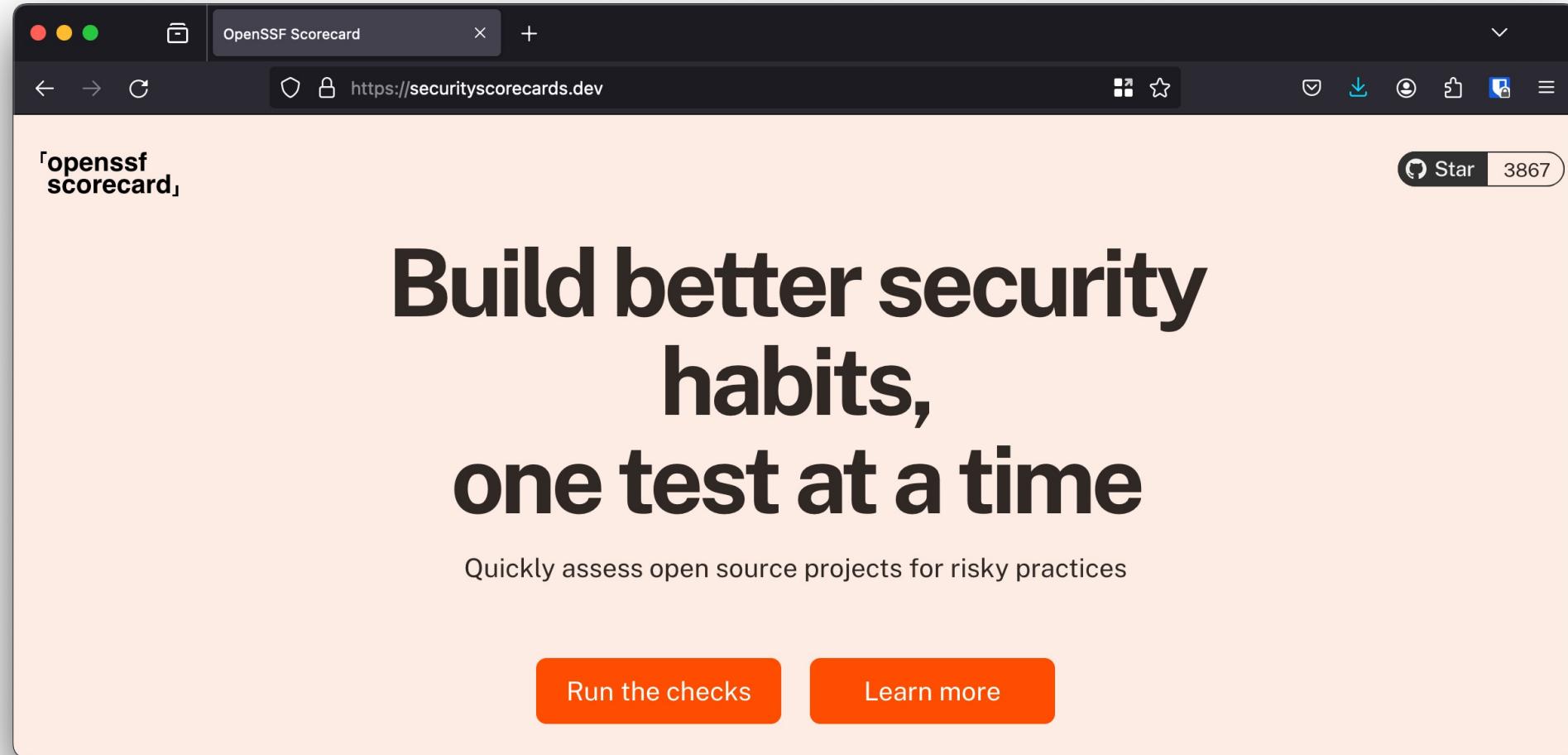


BLOG AUTHOR
Karlo Zanki, Reverse Engineer at ReversingLabs. [READ MORE...](#)

Nutrition Label for Software?



OpenSSF Scorecards

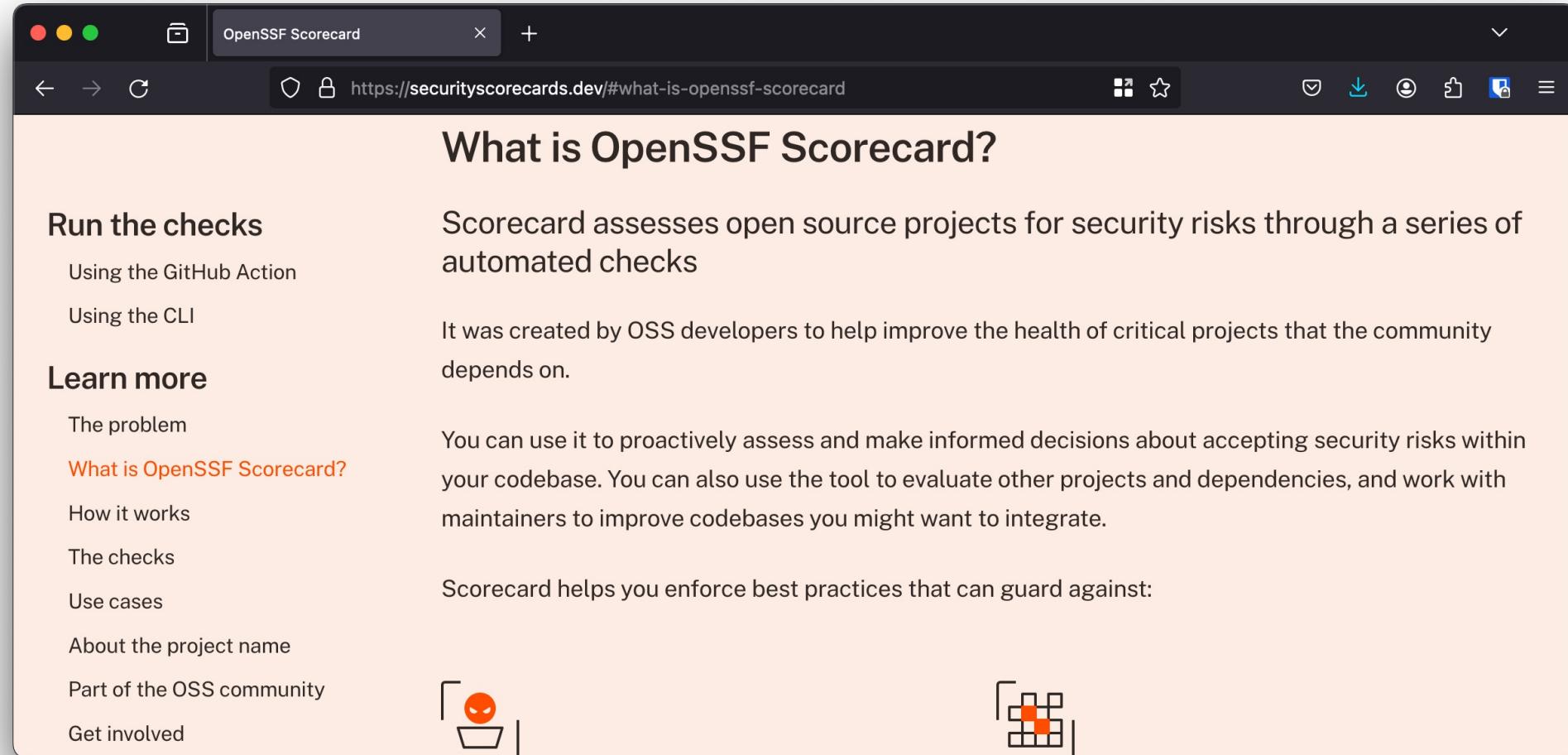


A screenshot of a web browser displaying the OpenSSF Scorecard homepage. The browser window has a dark header with tabs and a search bar showing the URL <https://securityscorecards.dev>. The main content area features a large, bold, black text block:

**Build better security
habits,
one test at a time**

Below this text, a smaller subtext reads: "Quickly assess open source projects for risky practices". At the bottom of the page are two orange rectangular buttons with white text: "Run the checks" and "Learn more".

OpenSSF Security Scorecards



The screenshot shows a web browser window titled "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 features the following text:

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

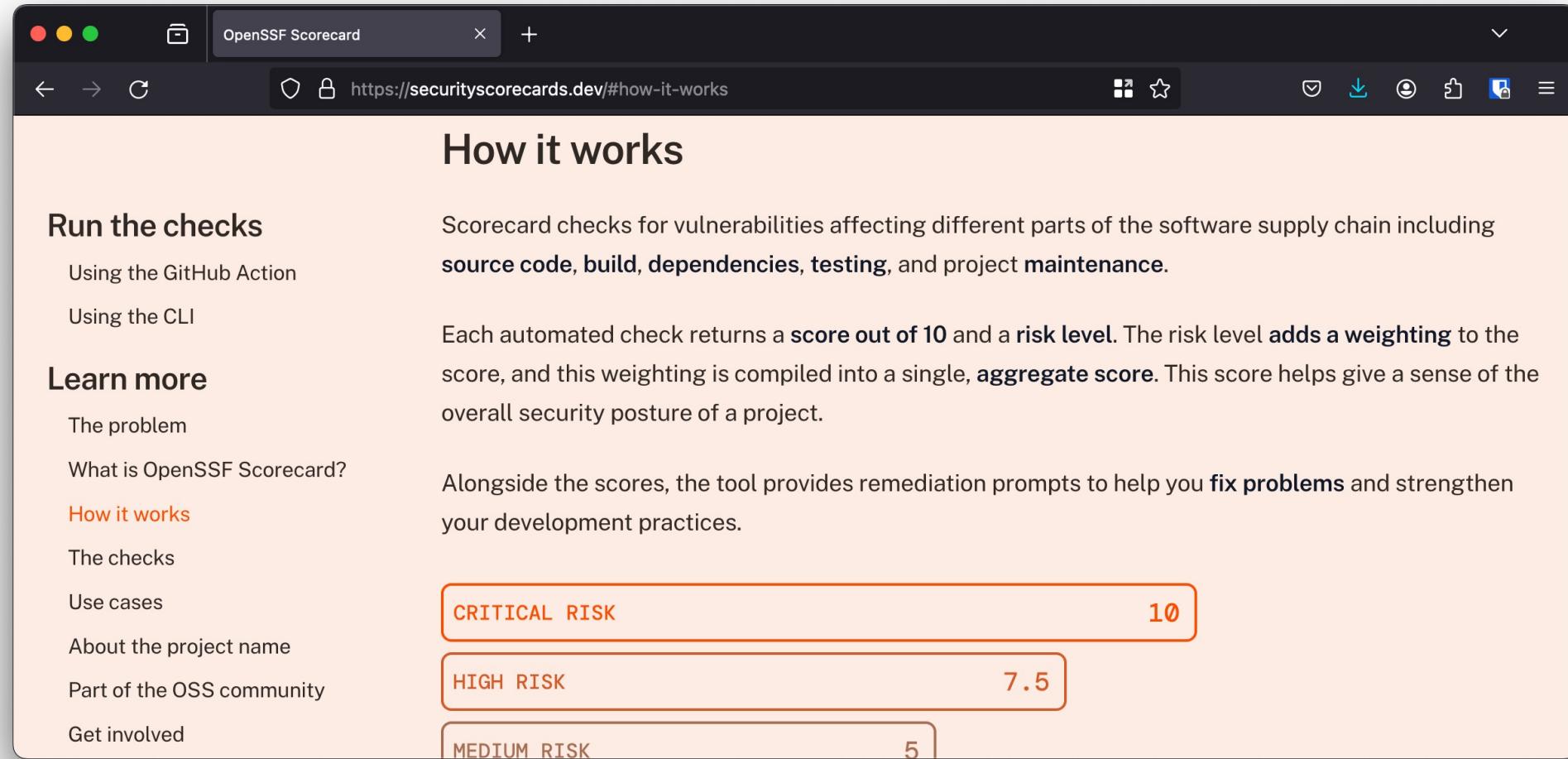
It was created by OSS developers to help improve the health of critical projects that the community depends on.

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:



OpenSSF Security Scorecards



The screenshot shows a web browser window for the OpenSSF Scorecard. The URL is <https://securityscorecards.dev/#how-it-works>. The page title is "How it works". On the left, there's a sidebar with sections like "Run the checks" (with links to GitHub Action and CLI), "Learn more" (with links to The problem, What is OpenSSF Scorecard?, How it works, The checks, Use cases, About the project name, Part of the OSS community, and Get involved), and a "CRITICAL RISK" score card at the bottom. The main content area describes the scoring process and provides remediation prompts.

How it works

Run the checks

Scorecard checks for vulnerabilities affecting different parts of the software supply chain including **source code, build, dependencies, testing, and project maintenance**.

Using the GitHub Action
Using the CLI

Learn more

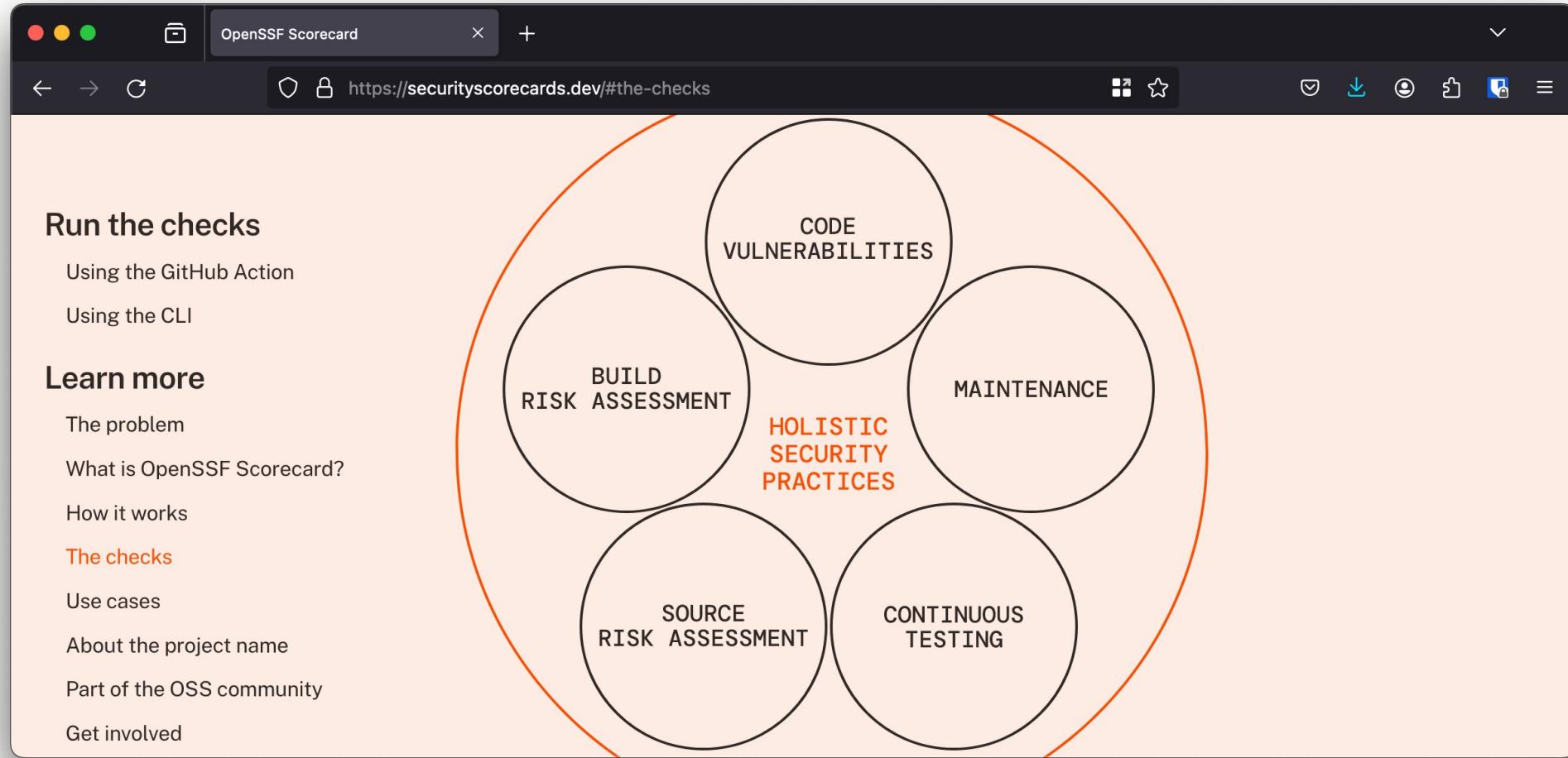
The problem
What is OpenSSF Scorecard?
How it works
The checks
Use cases
About the project name
Part of the OSS community
Get involved

Each automated check returns a **score out of 10** and a **risk level**. The risk level **adds a weighting** to the score, and this weighting is compiled into a single, **aggregate score**. This score helps give a sense of the overall security posture of a project.

Alongside the scores, the tool provides remediation prompts to help you **fix problems** and strengthen your development practices.

CRITICAL RISK	10
HIGH RISK	7.5
MEDIUM RISK	5

OpenSSF Security Scorecards

A screenshot of a web browser displaying the OpenSSF Scorecard website at <https://securityscorecards.dev/#the-checks>. The page has a dark header with the title "OpenSSF Scorecard". On the left, there's a sidebar with sections for "Run the checks" (GitHub Action, CLI) and "Learn more" (problem, what it is, how it works, checks, use cases, project name, community, get involved). The main content area features a diagram titled "HOLISTIC SECURITY PRACTICES" enclosed in a large orange circle. Inside the circle are five smaller black-outlined circles: "CODE VULNERABILITIES" (top), "BUILD RISK ASSESSMENT" (left), "MAINTENANCE" (right), "SOURCE RISK ASSESSMENT" (bottom-left), and "CONTINUOUS TESTING" (bottom-right).

Code Vulnerabilities (High)

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

ID	Packages	Summary	Affected versions	Published	Fix
GHSA-x674-v45j-fwxw	NuGet/Microsoft.Identity.Client	MSAL.NET applications targeting Xamarin Android and .NET Android (MAUI) susceptible to local denial of service	4.48.0 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-6737-26cr	NuGet/SixLabors.ImageSharp	SixLabors.ImageSharp vulnerable to data leakage	1.0.0 1.0.0-beta0002 1.0.0-beta0003 1.0.0-beta0004 1.0.0-beta0005 1.0.0-beta0006 ...	1.0.0-beta0001 1.0.0-beta0002 1.0.0-beta0003 1.0.0-beta0004 1.0.0-beta0005 1.0.0-beta0006 ...	yesterday Fix available
GHSA-g85r-6x2q-45w7	NuGet/SixLabors.ImageSharp	SixLabors.ImageSharp vulnerable to Memory Allocation with Excessive Size Value	1.0.0 1.0.0-beta0002 1.0.0-beta0003 1.0.0-beta0004 1.0.0-beta0005 1.0.0-beta0006 ...	1.0.0-beta0001 1.0.0-beta0002 1.0.0-beta0003 1.0.0-beta0004 1.0.0-beta0005 1.0.0-beta0006 ...	yesterday Fix available

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.

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)



- 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).

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?

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**)



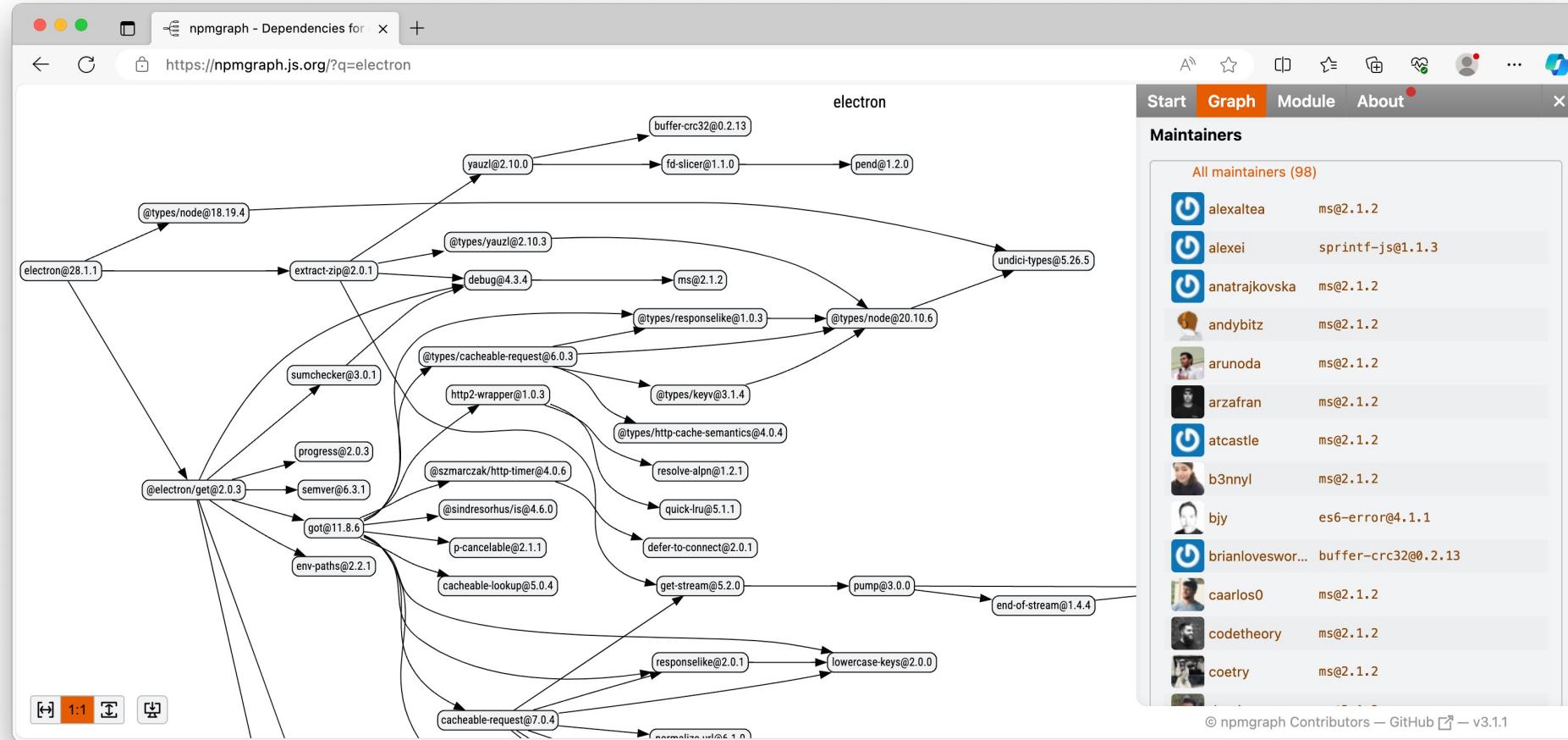
- This check determines whether the project requires human code review before pull requests (merge requests) are merged.
- The check determines whether the most recent changes (over the last ~30 commits) have an approval on GitHub or if the merger is different from the committer (implicit review)

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?

Source Risk Assessment Contributors (Low)



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?

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.

Build Risk Assesement Packaging (Medium)



- This check tries to determine if the project is published as a package.
- Packages give users of a project an easy way to download, install, update, and uninstall the software by a package manager.

Build Risk Assessment Signed Releases (High)



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

Demo OpenSSF Scorecard Fennec CLI



Running checks



 @nielstanis@infosec.exchange

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**



What can we improve?



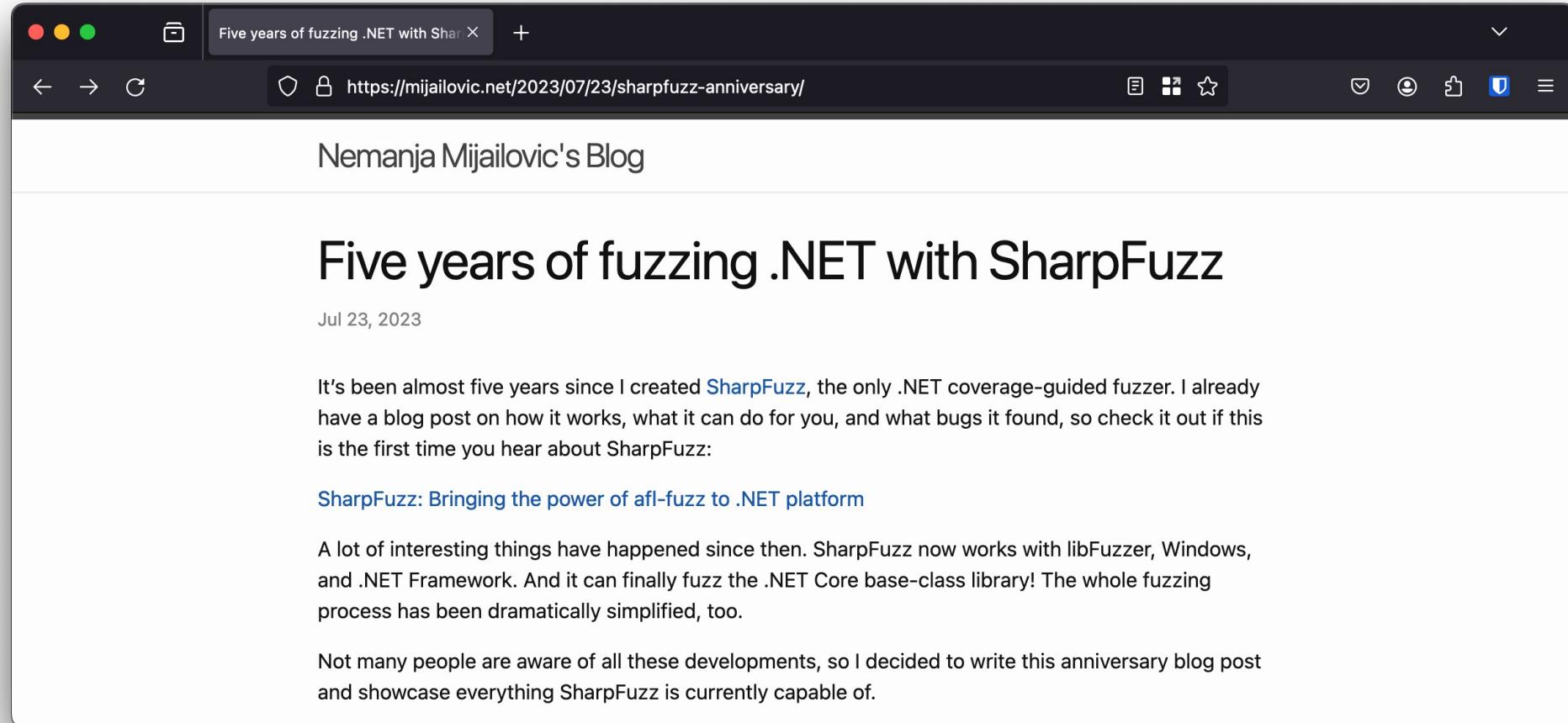
Fuzzing .NET



- 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?



Fuzzing .NET & SharpFuzz



A screenshot of a web browser window showing a blog post. The title of the post is "Five years of fuzzing .NET with SharpFuzz". The post is dated Jul 23, 2023. The content discusses the history and evolution of SharpFuzz, mentioning its initial creation, the .NET coverage-guided fuzzer, and its current capabilities across various platforms like libFuzzer, Windows, and .NET Framework. It also highlights the ability to fuzz the .NET Core base-class library and the simplification of the fuzzing process.

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.

Fuzzing .NET & SharpFuzz



The screenshot shows a web browser window with the title "Five years of fuzzing .NET with SharpFuzz". The page content is titled "Trophies" and discusses the growth of bugs found by SharpFuzz, mentioning several specific bugs like BigInteger.TryParse and Double.Parse. It also highlights two security vulnerabilities found using SharpFuzz. The browser interface includes standard navigation buttons, a search bar, and a toolbar with various icons.

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.



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) { }
    });
}
```



New &
Improved!

Fuzzomatic: Using AI to Fuzz Rust

The screenshot shows a web browser window with the URL <https://research.kudelskisecurity.com/2023/12/07/introducing-fuzzomatic-using-ai-to-automatically-fuzz-rust-projects-from-scratch/>. The page title is "Introducing Fuzzomatic: Using / X". The main content is titled "How does it work?". It explains that Fuzzomatic uses libFuzzer and cargo-fuzz as backends, combining AI and deterministic techniques. It mentions using OpenAI API models like gpt-3.5-turbo and gpt-3.5-turbo-16k. A section titled "Fuzz targets and coverage-guided fuzzing" shows a code snippet for a Rust fuzz target:

```
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});
```

The text below the code explains that this needs to be compiled into an executable and describes coverage-guided fuzzing.

At the bottom right of the browser window, there are buttons for Comment, Reblog, Subscribe, and more.



Static Code Analysis (SAST)

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



Static Code Analysis (SAST)

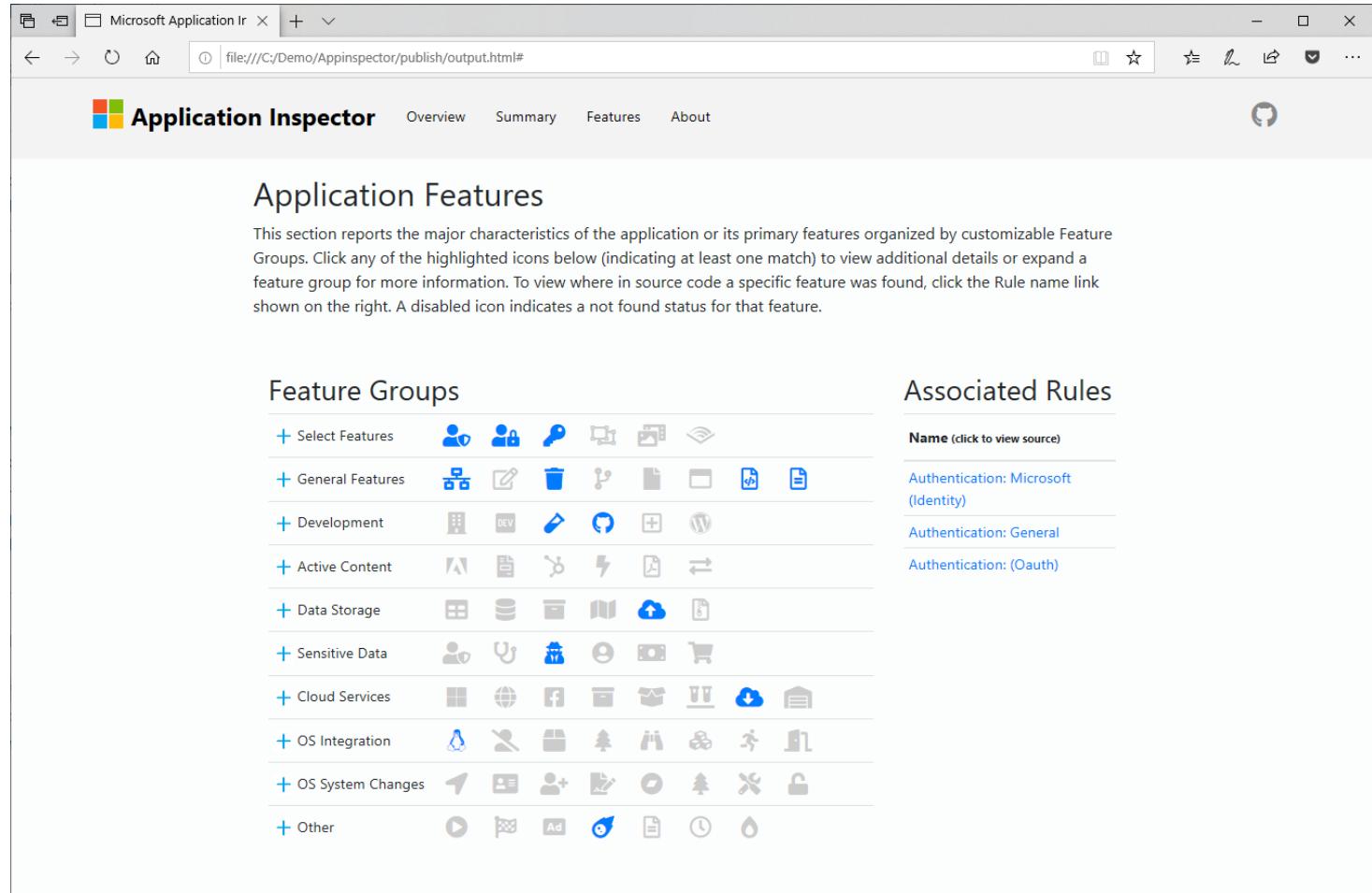
```
public class CustomerController : Controller
{
    public IActionResult GenerateCustomerReport(string customerID)
    {
        var data = Reporting.GenerateCustomerReportOverview(customerID)
        return View(data);
    }
    public static class Reporting
    {
        public static byte[] GenerateCustomerReportOverview(string ID)
        {
            return System.IO.File.ReadAllBytes("./data/{ID}.pdf");
        }
    }
}
```

.NET Reproducibility



- Reproducible builds 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?

Application Inspector



The screenshot shows a web browser window titled "Microsoft Application Inspector" displaying the "Application Features" page. The page includes a navigation bar with links to Overview, Summary, Features, and About, along with a GitHub icon. Below the navigation is a section titled "Application Features" with a descriptive paragraph. To the right, there's a sidebar titled "Associated Rules" listing several rule names. The main content area is divided into sections for "Feature Groups" and "Associated Rules".

Associated Rules

- Name (click to view source)
 - Authentication: Microsoft (Identity)
 - Authentication: General
 - Authentication: (Oauth)

Feature Groups

- + Select Features
- + General Features
- + Development
- + Active Content
- + Data Storage
- + Sensitive Data
- + Cloud Services
- + OS Integration
- + OS System Changes
- + Other

Each group has a corresponding set of icons below it.

New &
Improved!

Application Inspector

Select Features

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



New &
Improved!

Community Review

The screenshot shows a web browser window displaying the [Cargo Vet](https://mozilla.github.io/cargo-vet/) documentation. The page has a dark theme with white text. The left sidebar contains a table of contents for the "Introduction" section, which includes:

- 1. Introduction
 - 1.1. Motivation
 - 1.2. How it Works
- 2. Tutorial
 - 2.1. Installation
 - 2.2. Setup
 - 2.3. Audit Criteria
 - 2.4. Importing Audits
 - 2.5. Recording Audits
 - 2.6. Performing Audits
 - 2.7. Trusting Publishers
 - 2.8. Specifying Policies
 - 2.9. Multiple Repositories
 - 2.10. Configuring CI
 - 2.11. Curating Your Audit Set
- 3. Reference
 - 3.1. Configuration
 - 3.2. Audit Entries
 - 3.3. Wildcard Audit Entries
 - 3.4. Trusted Entries

The main content area is titled "Cargo Vet" and describes the tool's purpose: "The `cargo vet` subcommand is a tool to help projects ensure that third-party Rust dependencies have been audited by a trusted entity. It strives to be lightweight and easy to integrate." It explains that the tool matches project dependencies against audits performed by authors or entities they trust, providing assistance for gaps. The primary reason for not auditing open-source dependencies is the effort required. The page then lists three key features:

- Sharing:** Public crates share findings.
- Relative Audits:** Developers inspect differences between versions and consider the first vetted version for the second.
- Deferred Audits:** Dependencies can be added to exceptions, allowing for gradual coverage over time.

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



Questions?



Links

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



THE IT CONFERENCE FOR
MICROSOFT TECHNOLOGIES

Sponsors



Partners

