VERACODE

Reviewing NuGet Packages security easily using OpenSSF Scorecard

Niels Tanis
Sr. Principal Security Researcher

NDC { Security }

Who am I?

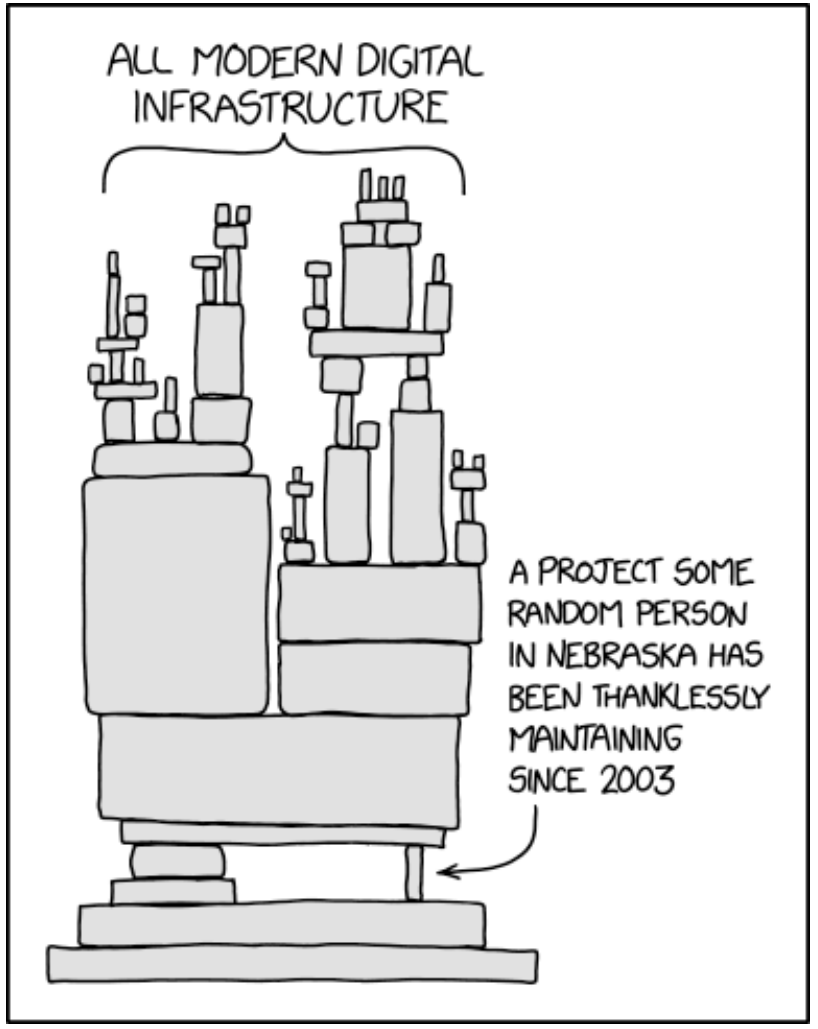
- Niels Tanis
- Sr. Principal Security Researcher
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - Research on static analysis for .NET apps
  - Enjoying Rust!
- Microsoft MVP – Developer Technologies

VERACODE

MVP Microsoft® Most Valuable Professional

@nielstanis@infosec.exchange

Modern Application Architecture XKCD 2347



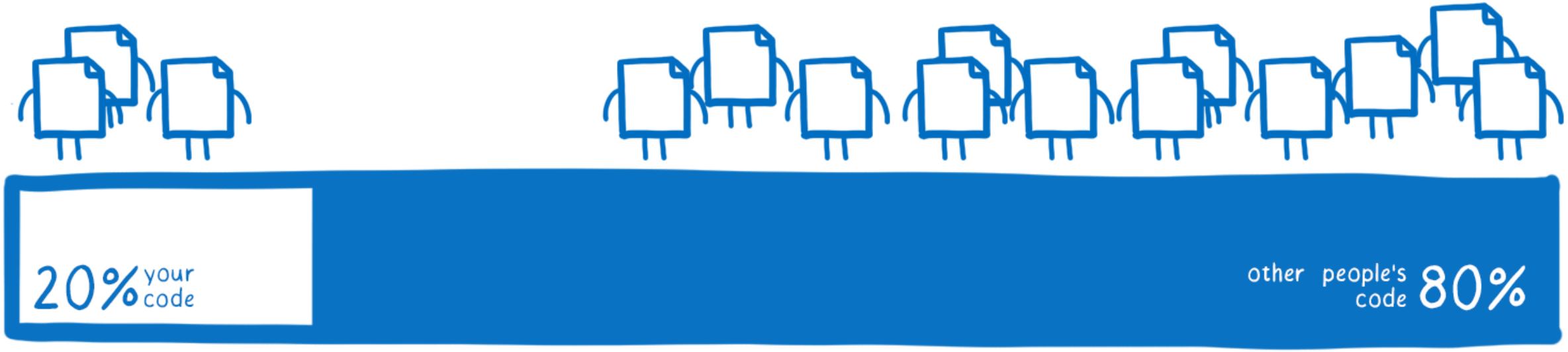
@nielstanis@infosec.exchange

Agenda

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

@nielstanis@infosec.exchange

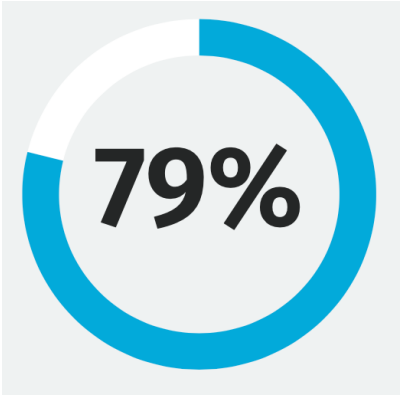
Average codebase composition

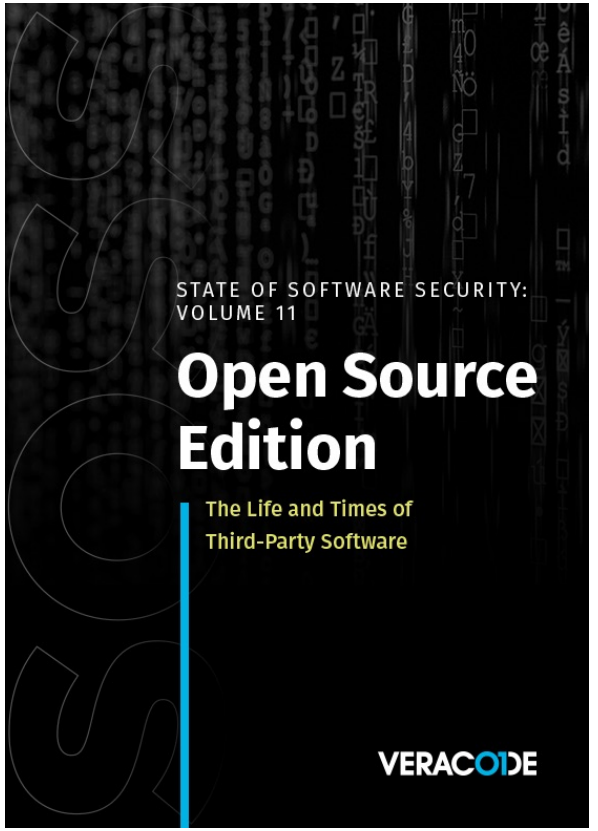


@nielstanis@infosec.exchange

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.”





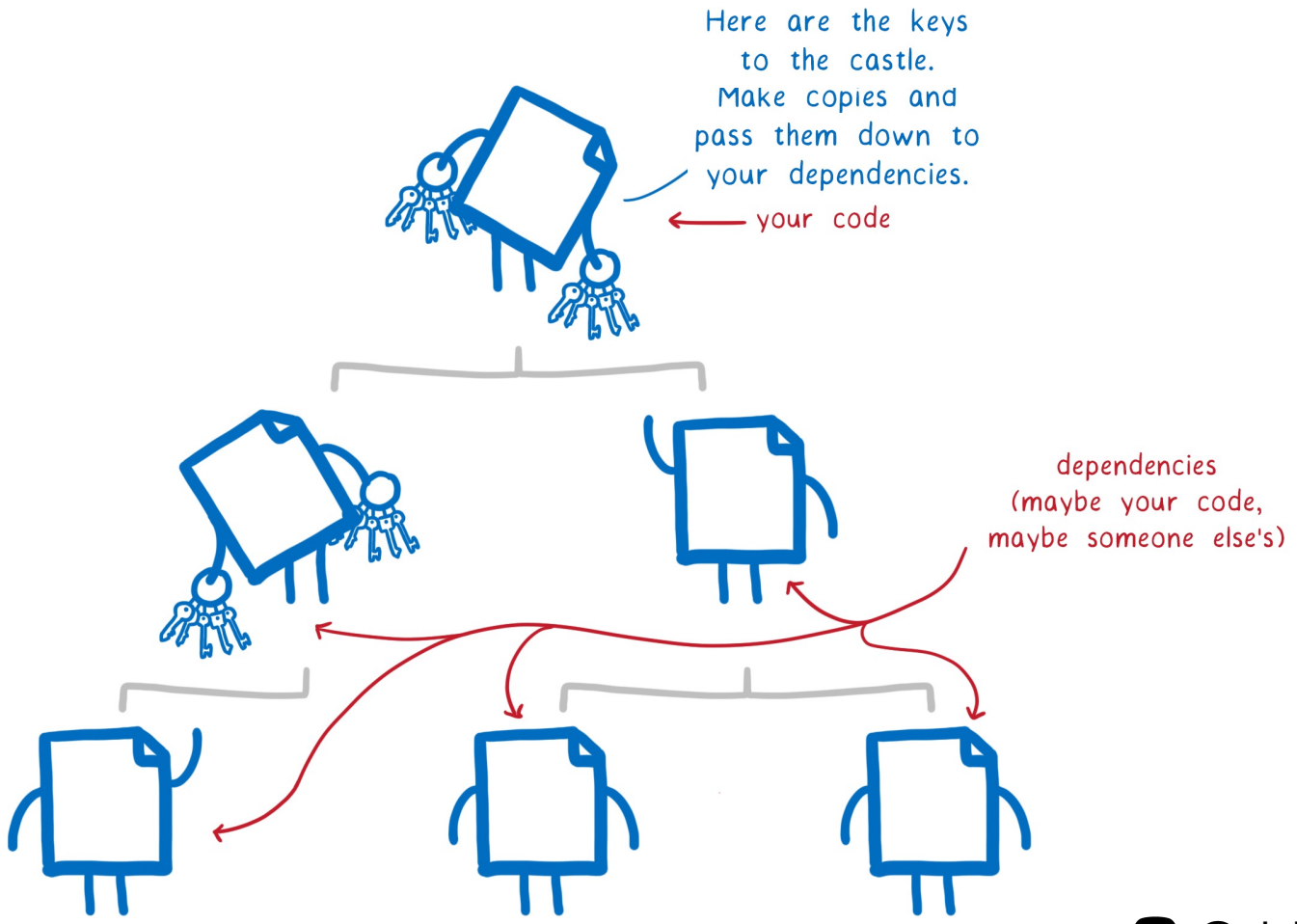
@nielstanis@infosec.exchange

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.

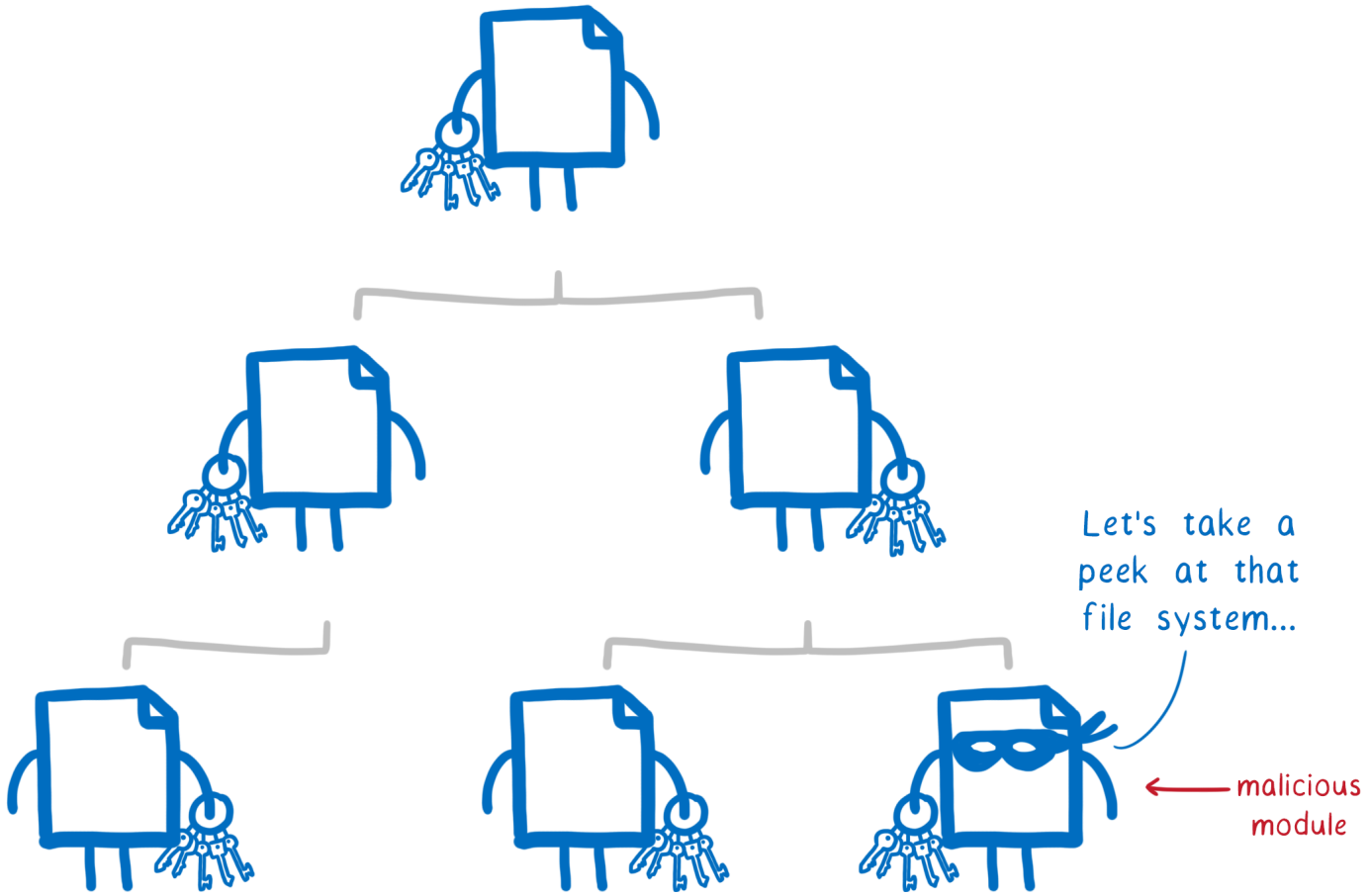
@nielstanis@infosec.exchange

Average codebase composition

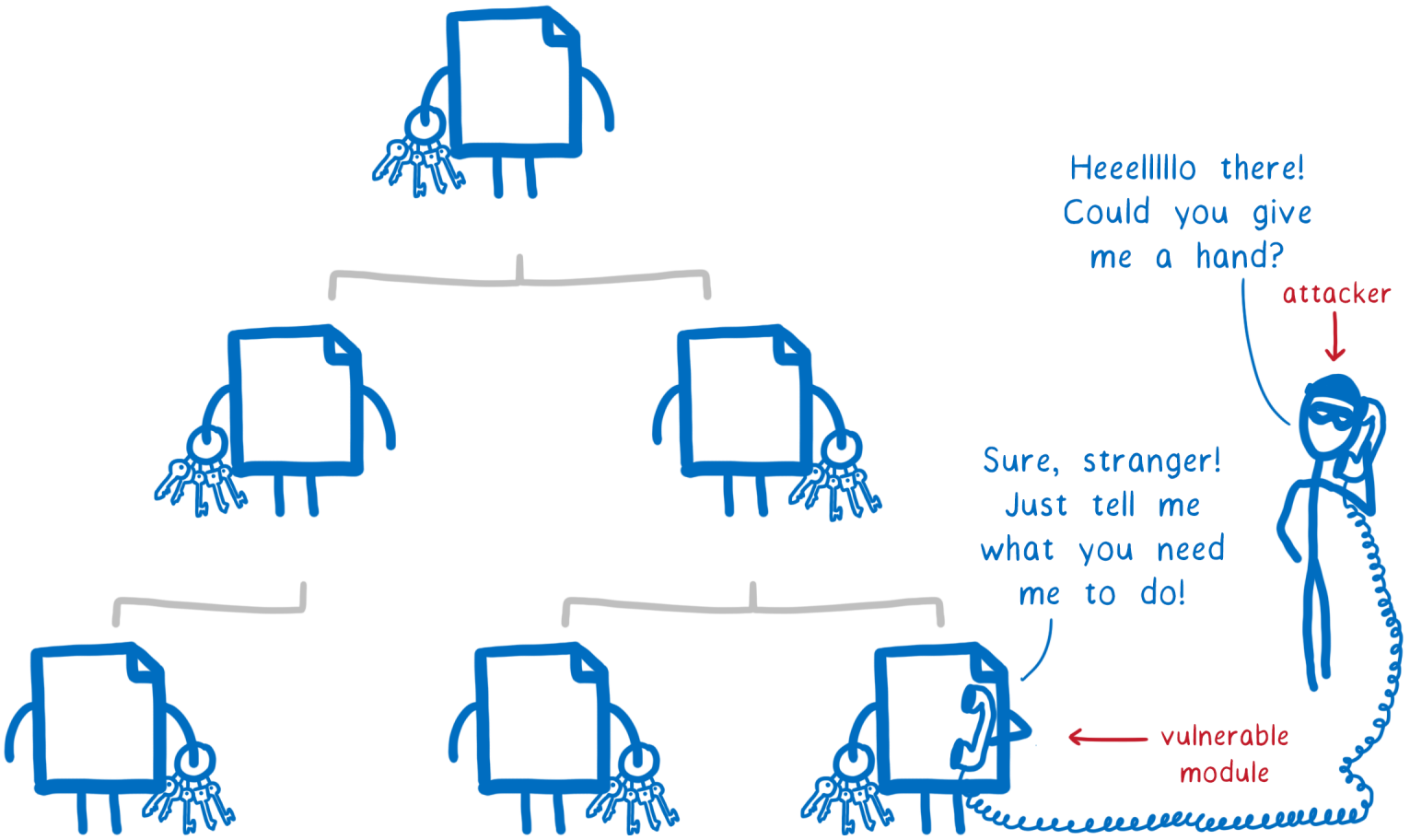


@nielstanis@infosec.exchange

Malicious Assemby

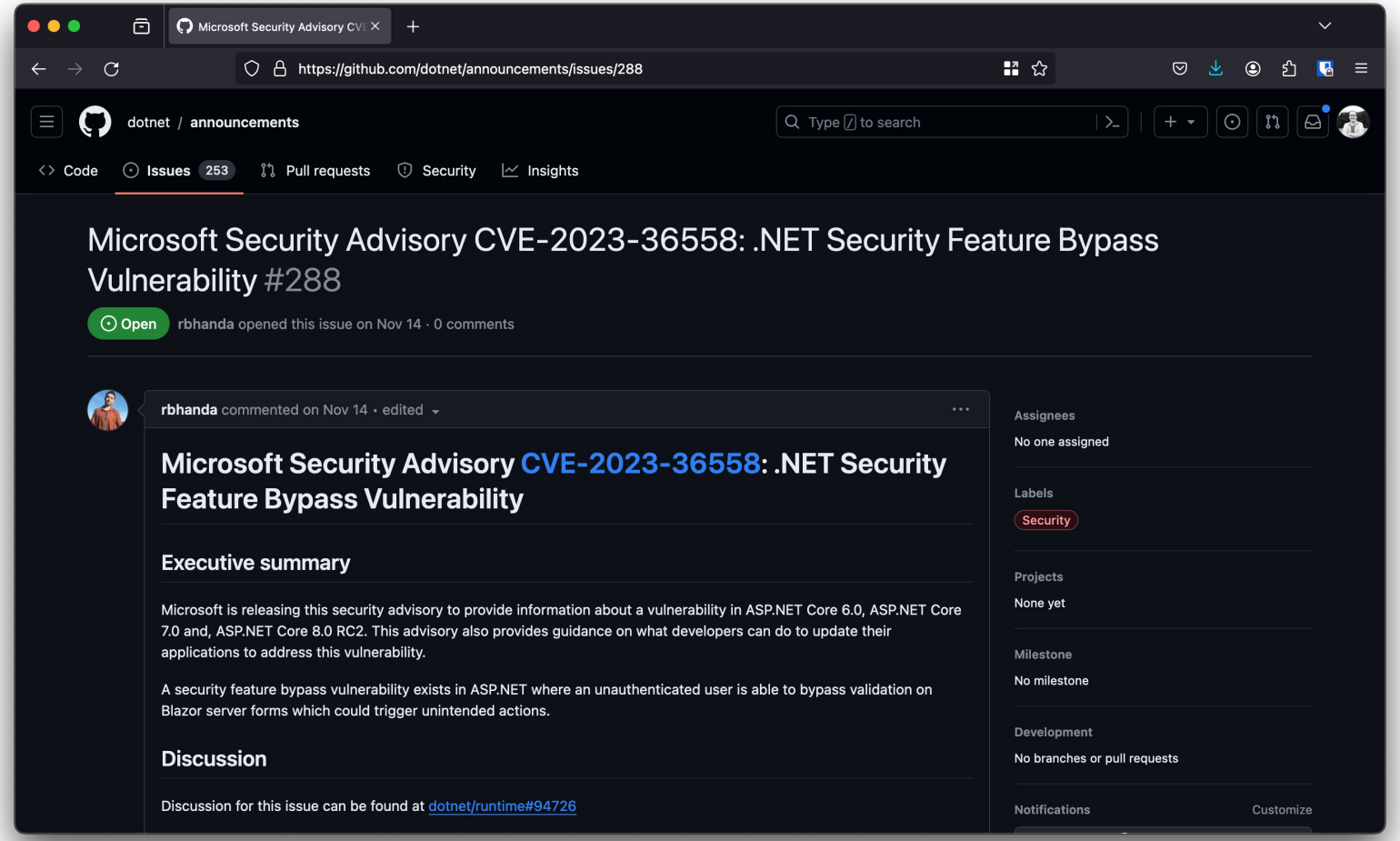


Vulnerable Assemby



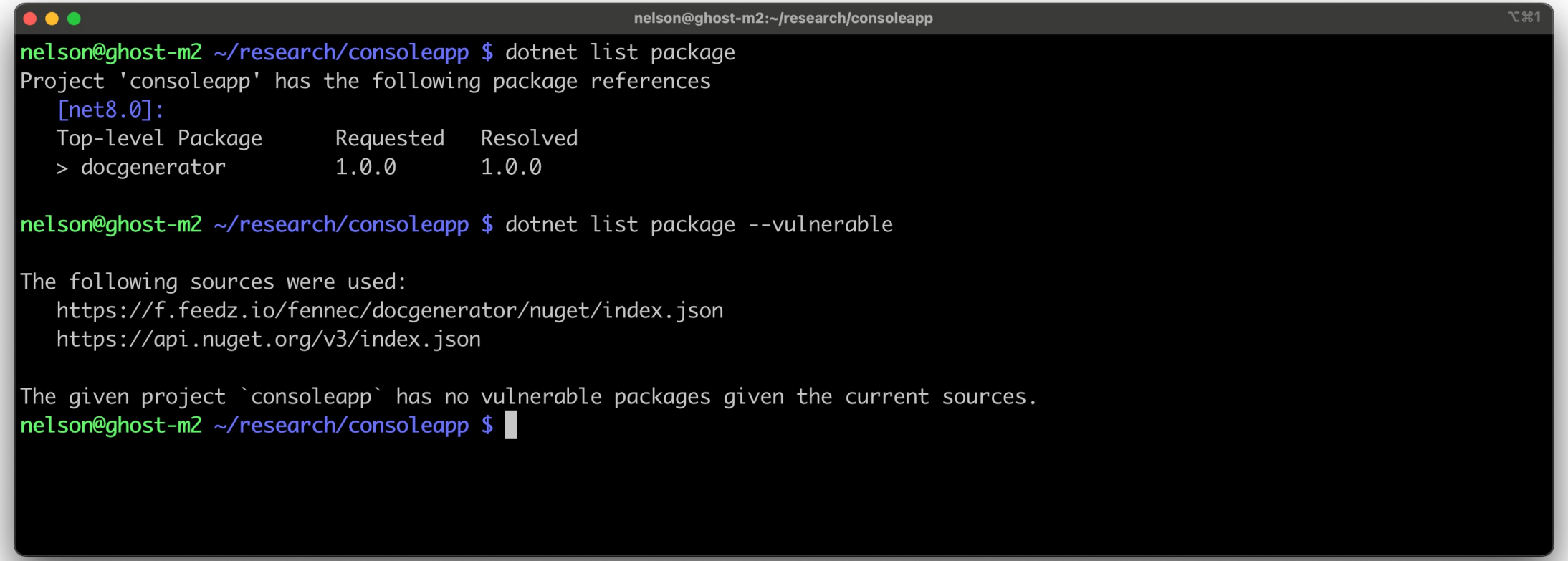
@nielstanis@infosec.exchange

Vulnerabilities in Libraries



@nielstanis@infosec.exchange

DotNet CLI



@nielstanis@infosec.exchange

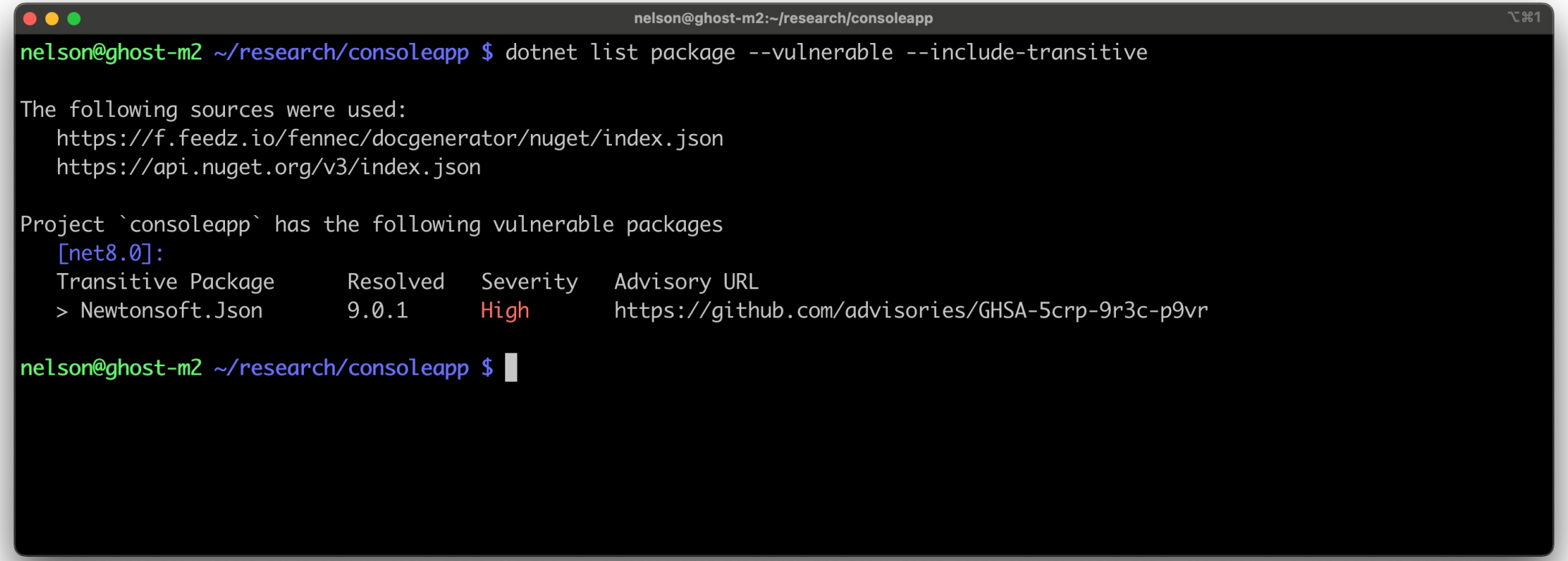
DotNet CLI

```
nelson@ghost-m2:~/research/consoleapp                                    ⌥⌘1
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.commons                                                  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
```

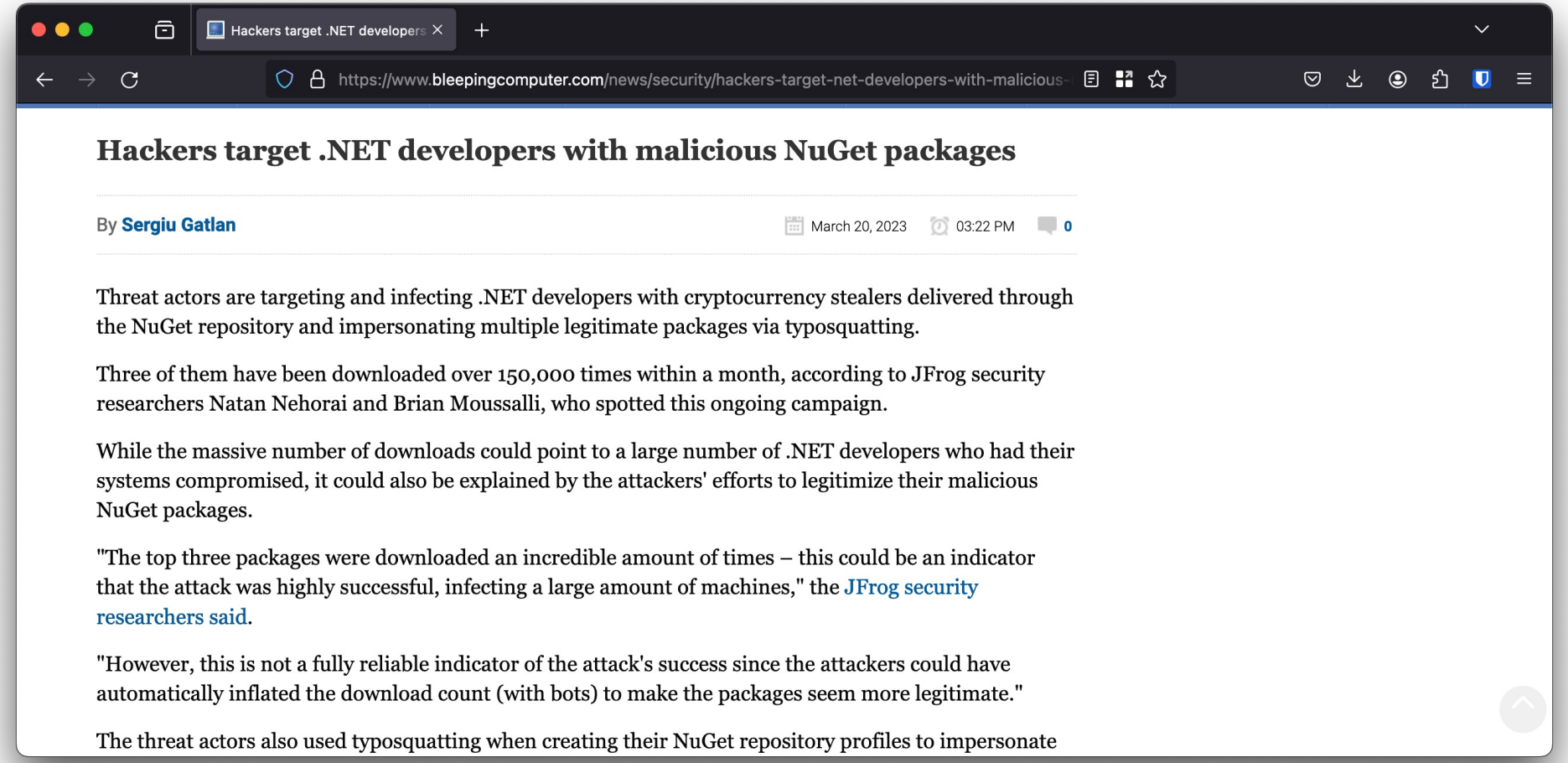

@nielstanis@infosec.exchange

DotNet CLI



@nielstanis@infosec.exchange

Malicious Package



Hackers target .NET developers with malicious NuGet packages

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

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.

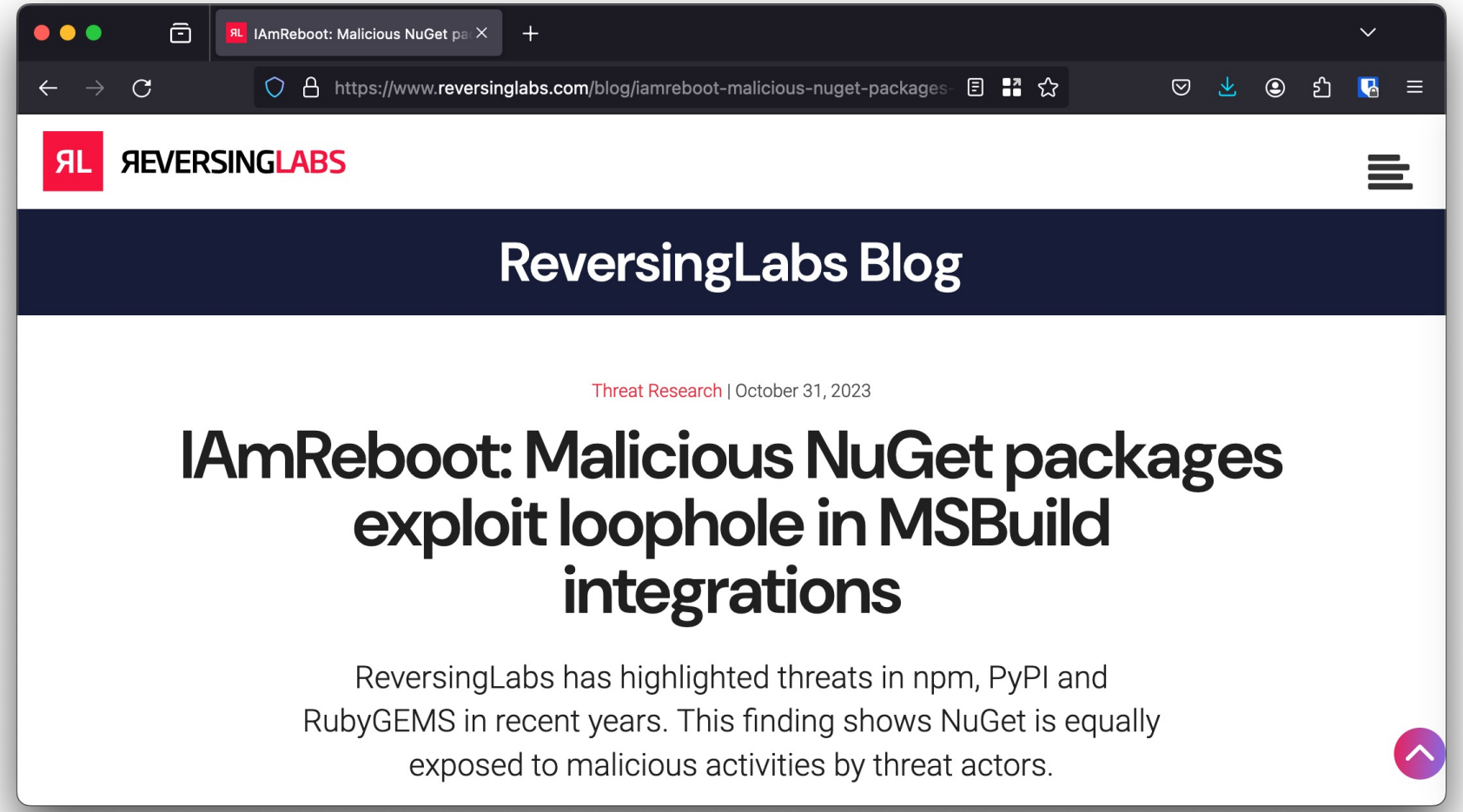
"The top three packages were downloaded an incredible amount of times – this could be an indicator that the attack was highly successful, infecting a large amount of machines," the JFrog security researchers said.

"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

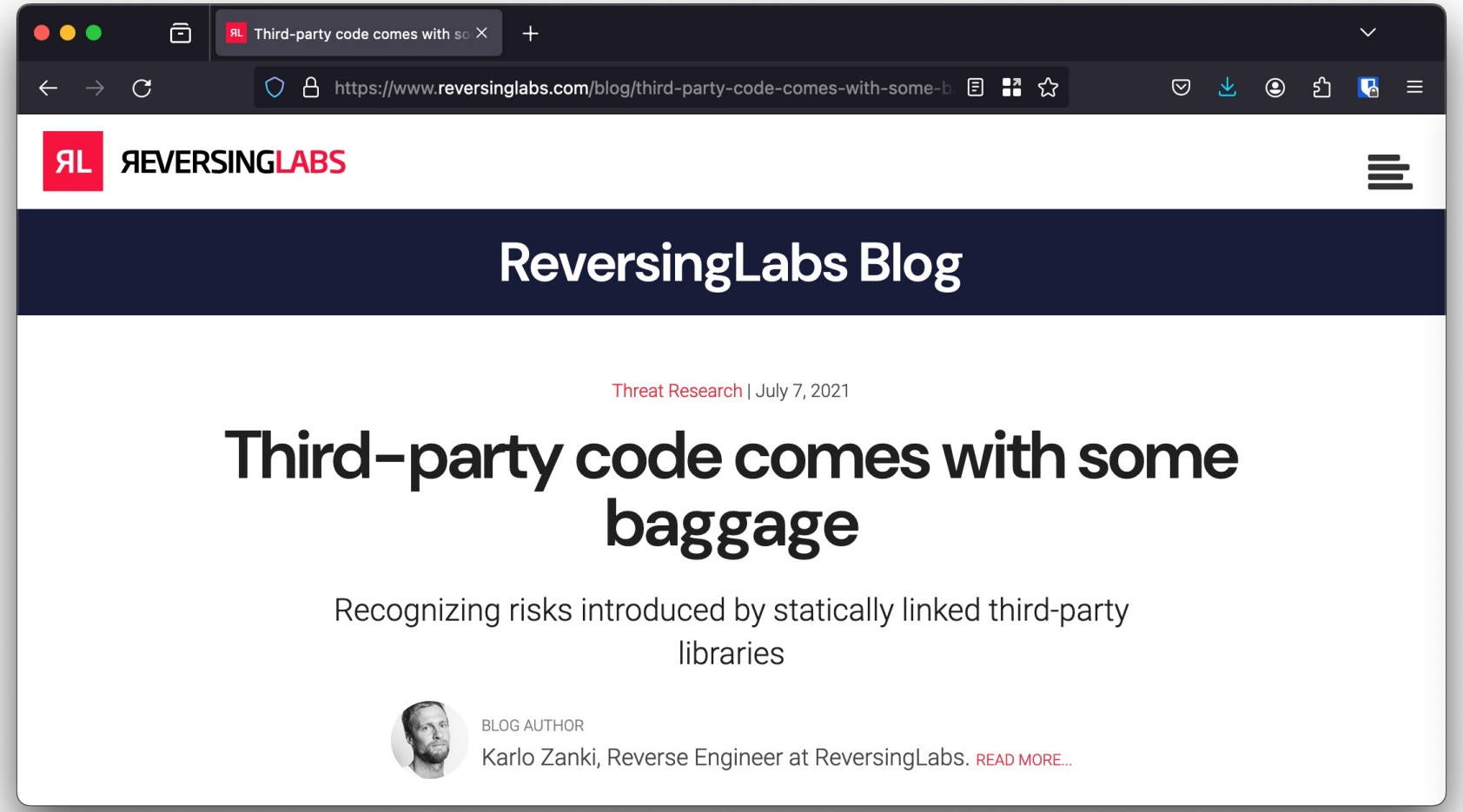
@nielstanis@infosec.exchange

Malicious Package



@nielstanis@infosec.exchange

Do you know what's inside?

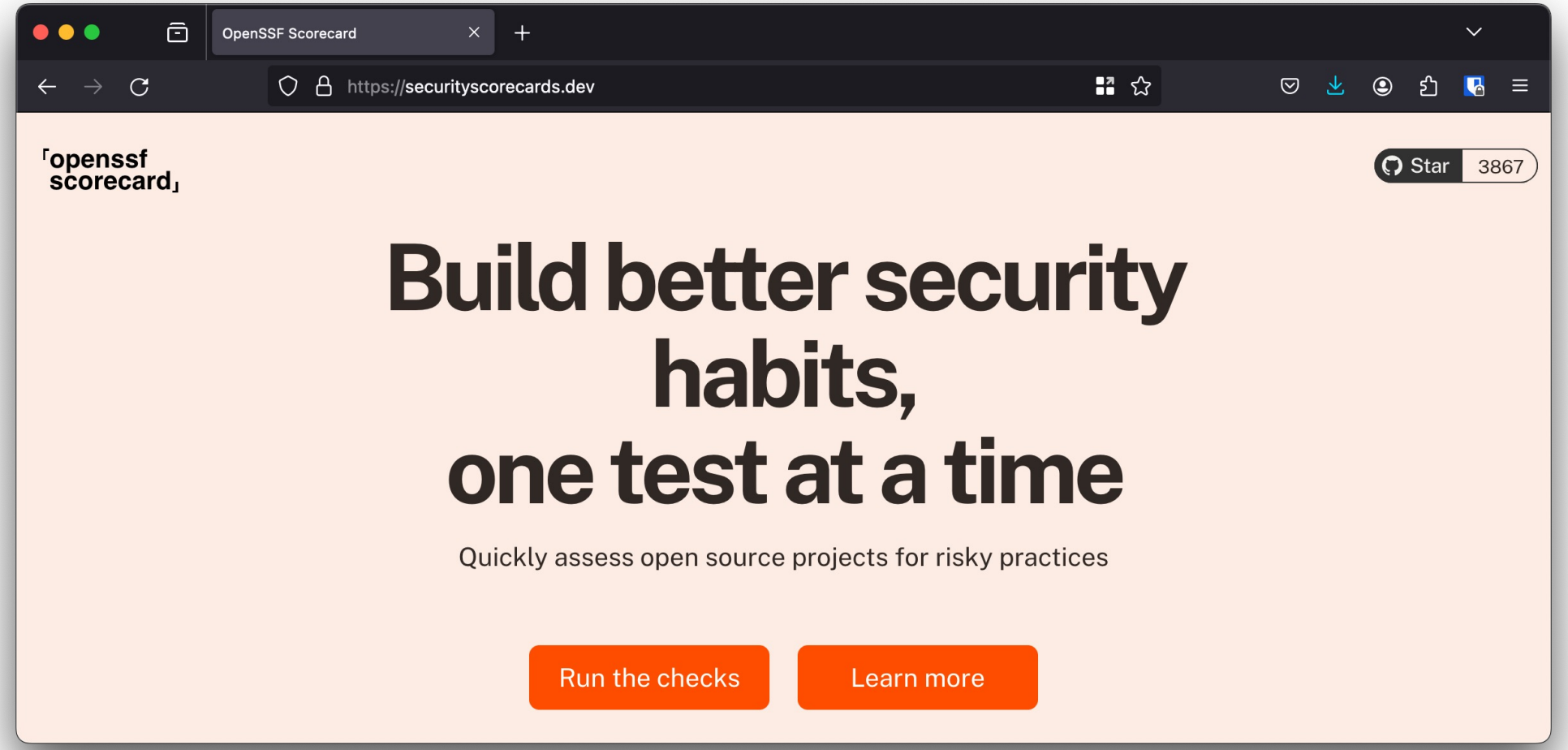


@nielstanis@infosec.exchange

Nutrion Label for Software?

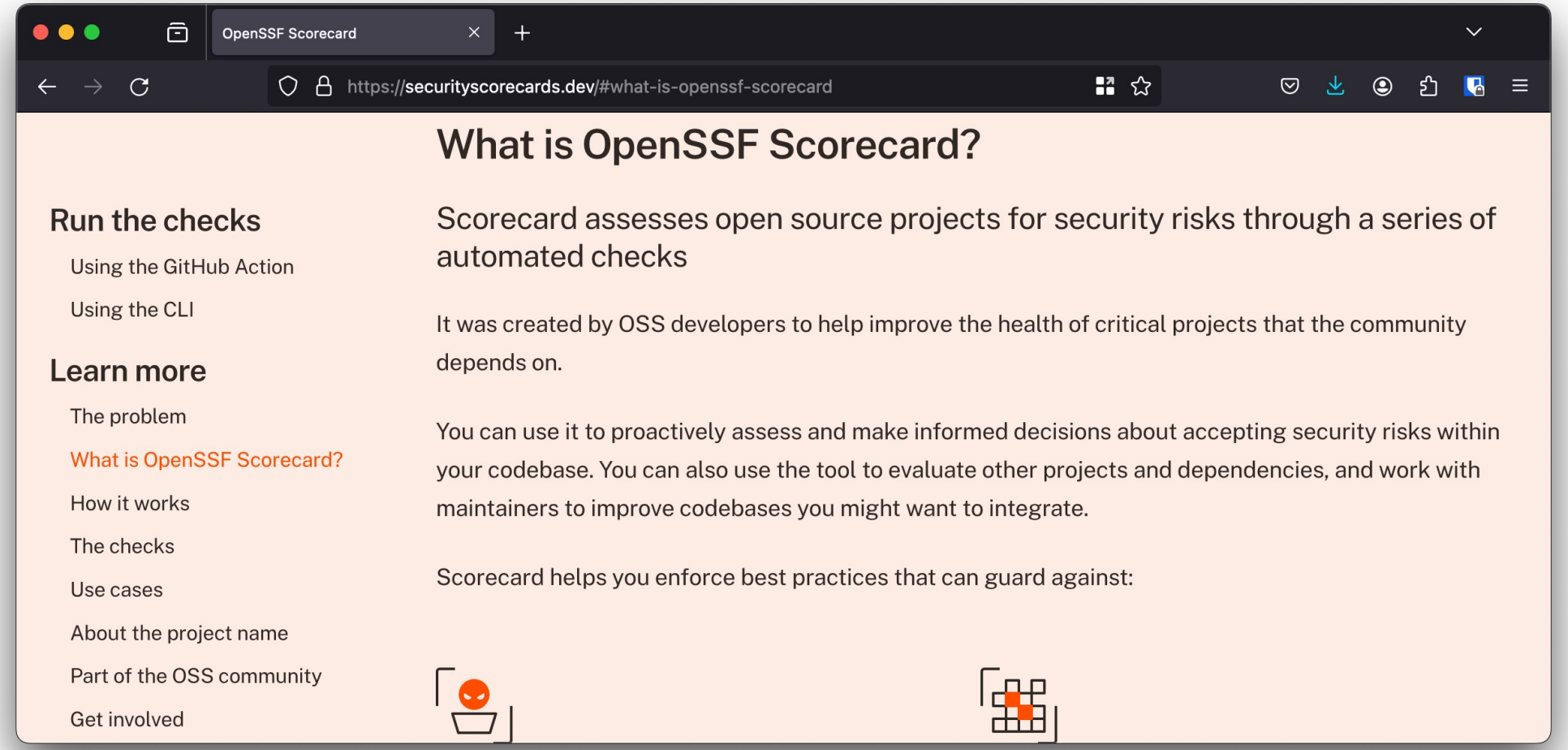
@nielstanis@infosec.exchange

OpenSSF Scorecards



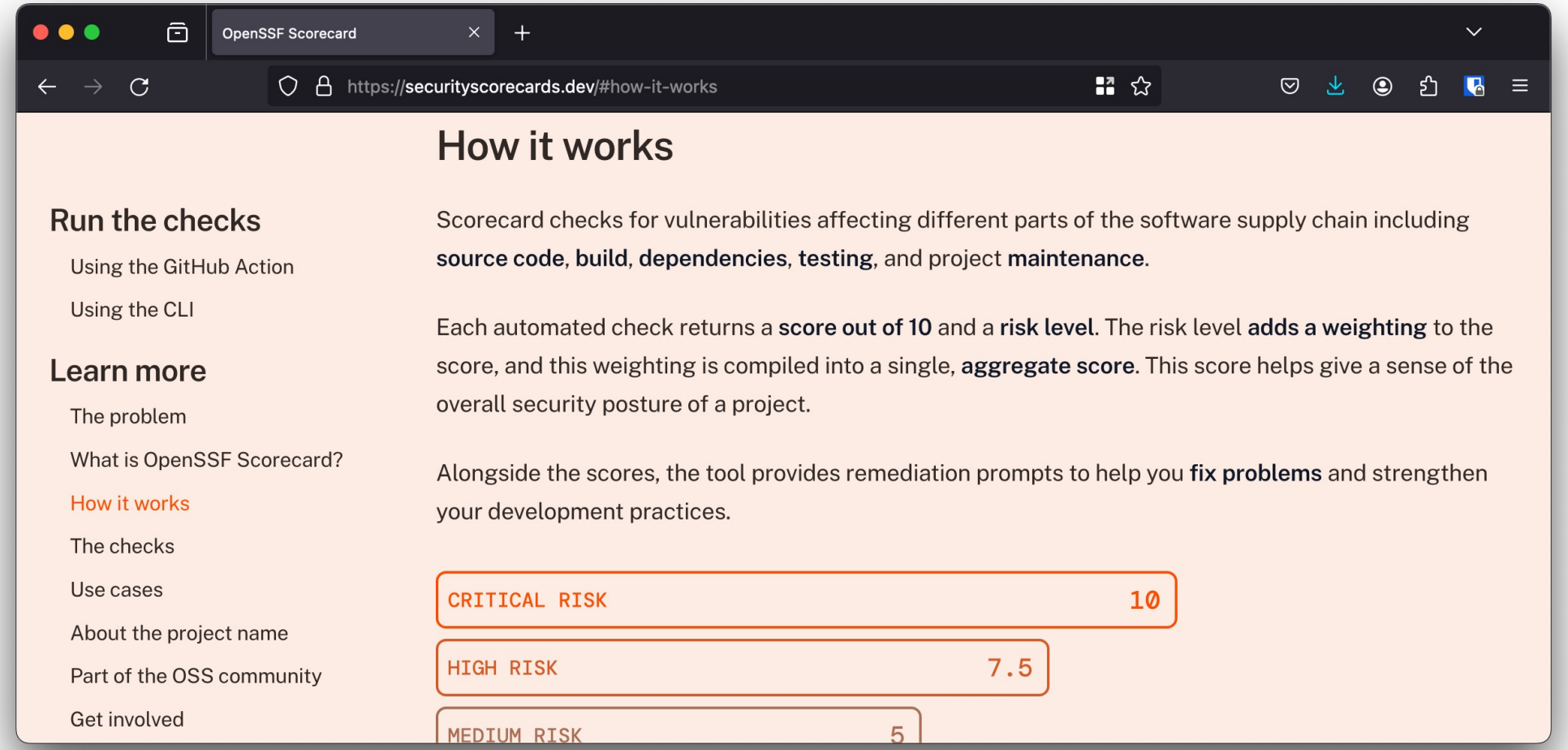
@nielstanis@infosec.exchange

OpenSSF Security Scorecards



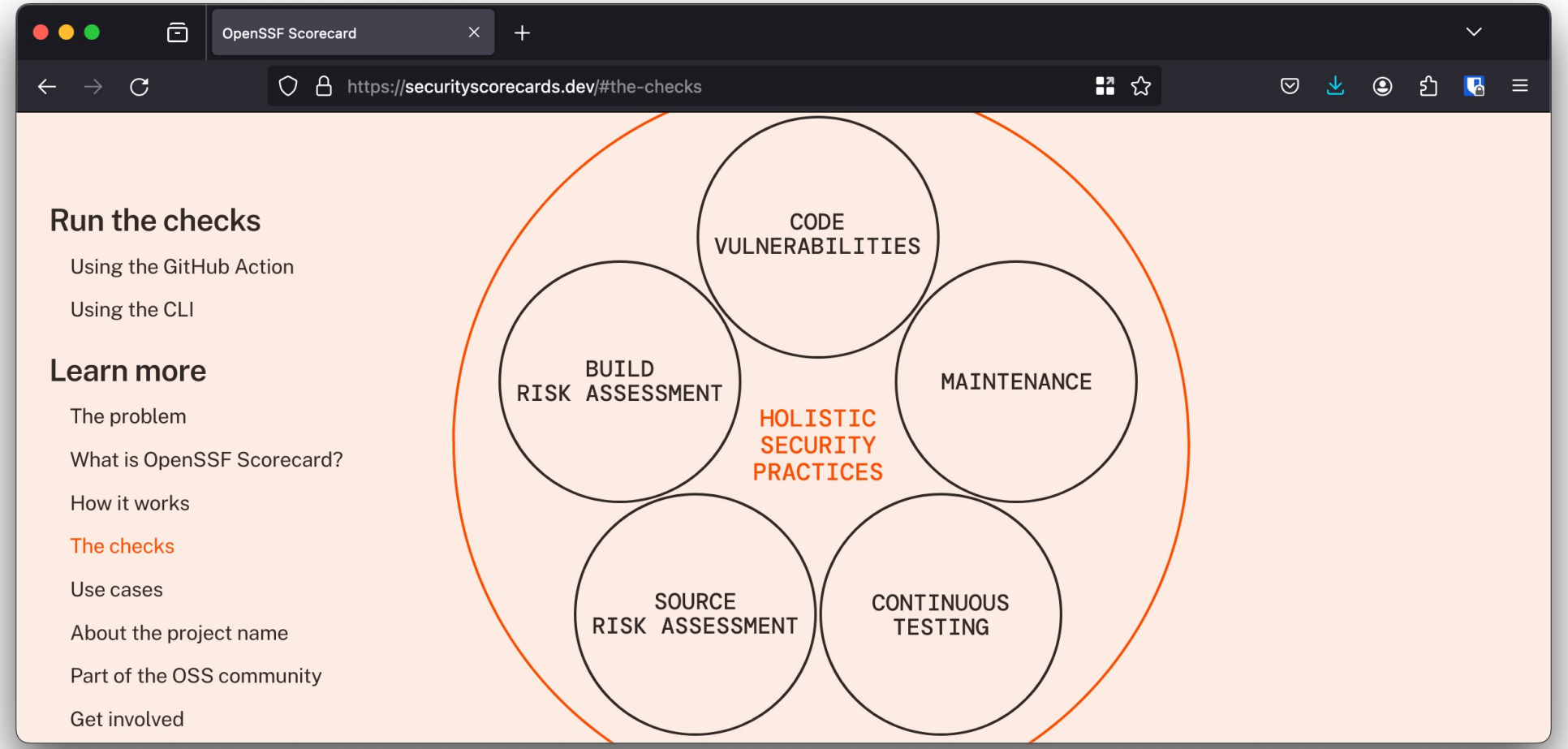
@nielstanis@infosec.exchange

OpenSSF Security Scorecards



@nielstanis@infosec.exchange

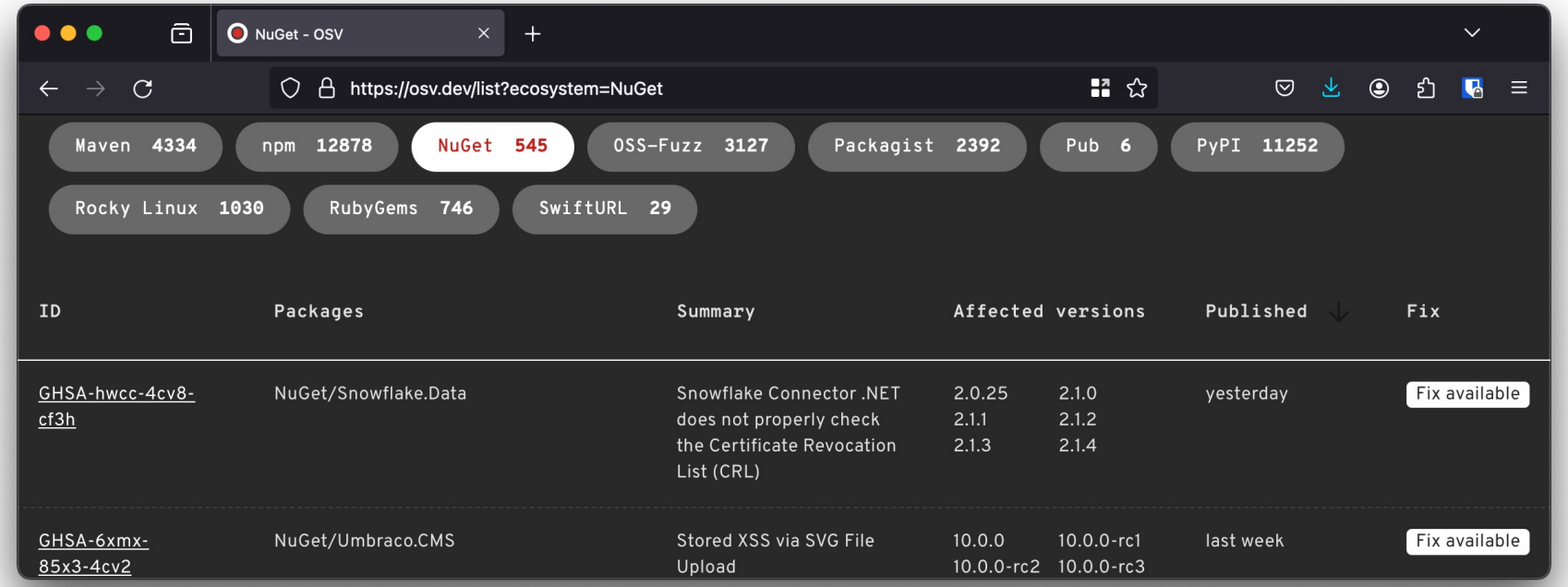
OpenSSF Security Scorecards



@nielstanis@infosec.exchange

Code Vulnerabilities (High)

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



@nielstanis@infosec.exchange

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.

@nielstanis@infosec.exchange

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 Assesement 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 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!

@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 Assesement
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)

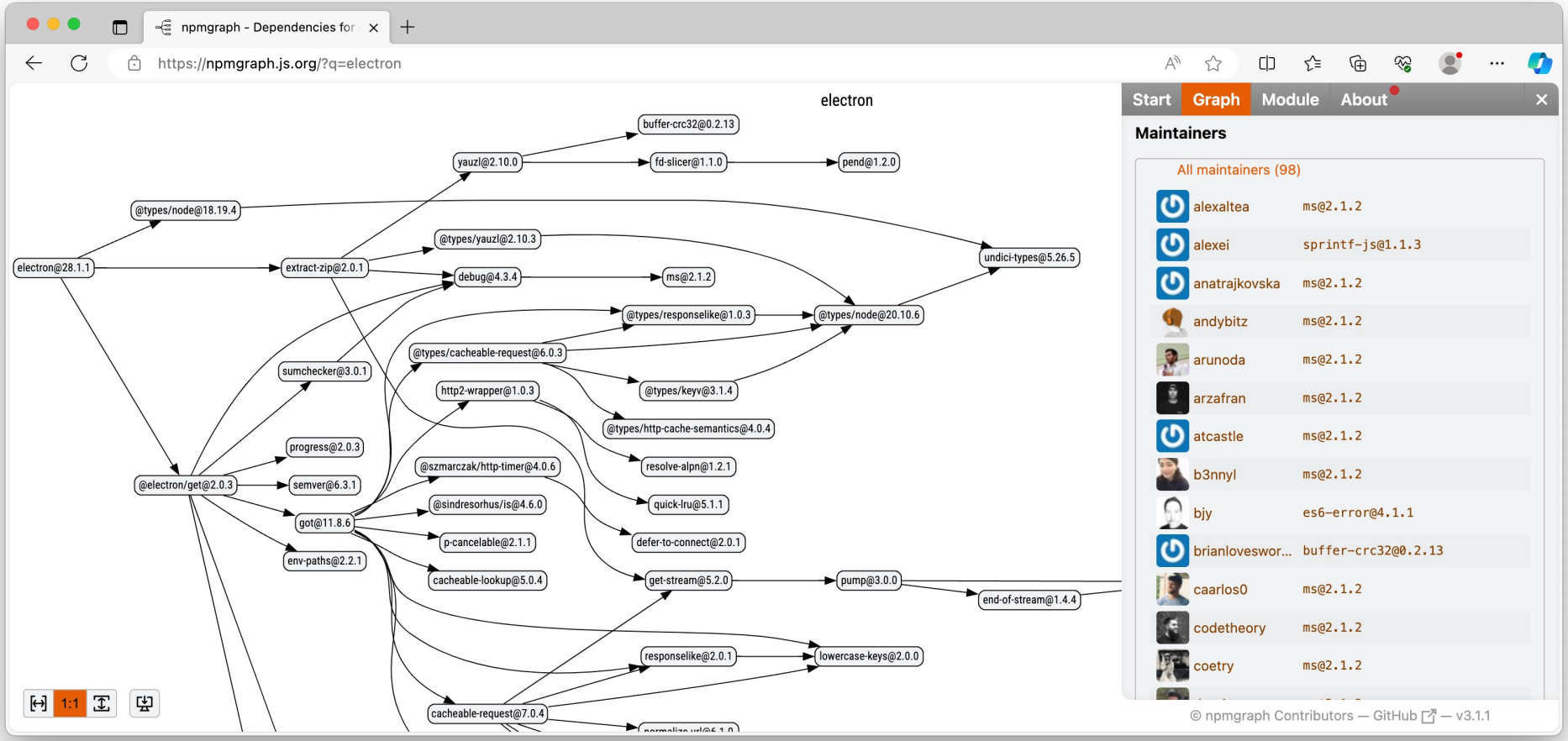
@nielstanis@infosec.exchange

Source Risk Assesement 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 Assesement Contributors (Low)



@nielstanis@infosec.exchange

Build Risk Assesement
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

@nielstanis@infosec.exchange

Build Risk Assesement
Token Permission (High)

- This check determines whether the project's automated workflows tokens follow the principle of least privilege.
- This is important because attackers may use a compromised token with write access to, for example, push malicious code into the project.

@nielstanis@infosec.exchange

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.

@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

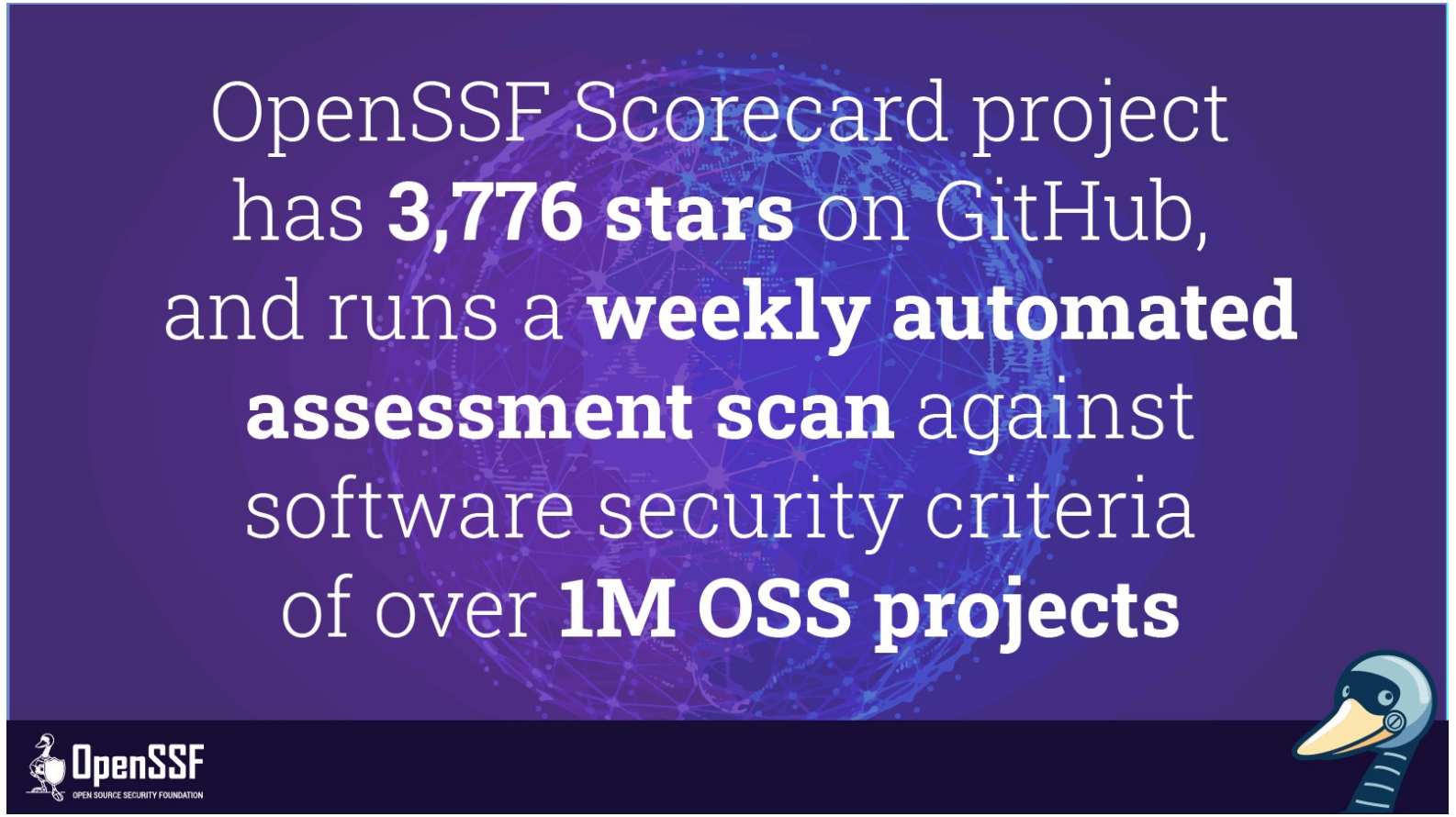
@nielstanis@infosec.exchange

Demo OpenSSF Scorecard


```
Running checks
```


@nielstanis@infosec.exchange

OpenSSF Annual Report 2023



@nielstanis@infosec.exchange

What can we improve?

@nielstanis@infosec.exchange

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

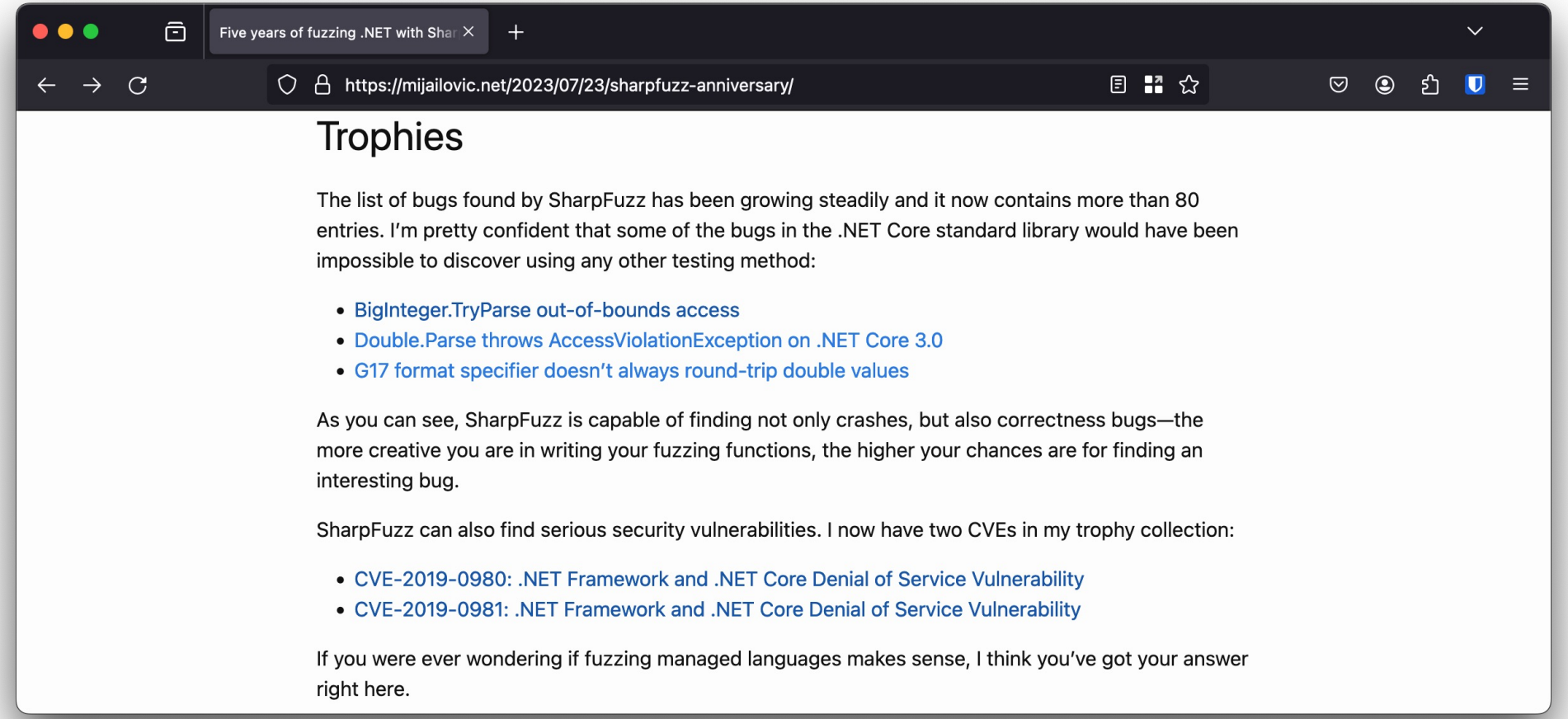
Fuzzing .NET & SharpFuzz

New & Improved!

@nielstanis@infosec.exchange

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.

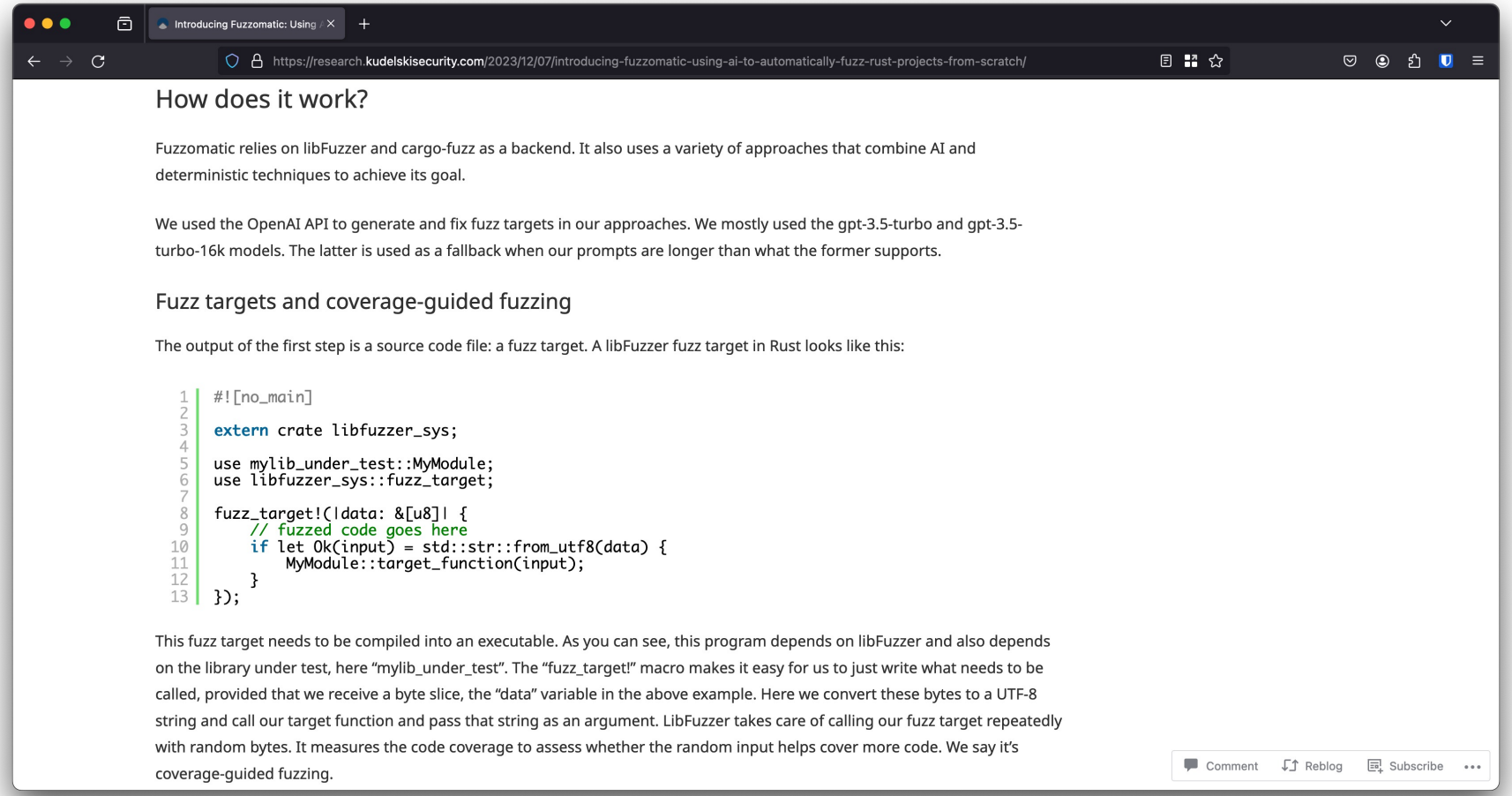
@nielstanis@infosec.exchange

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

Fuzzomatic: Using AI to Fuzz Rust



New & Improved!

@nielstanis@infosec.exchange

Static Code Analysis (SAST)


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


@nielstanis@infosec.exchange

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

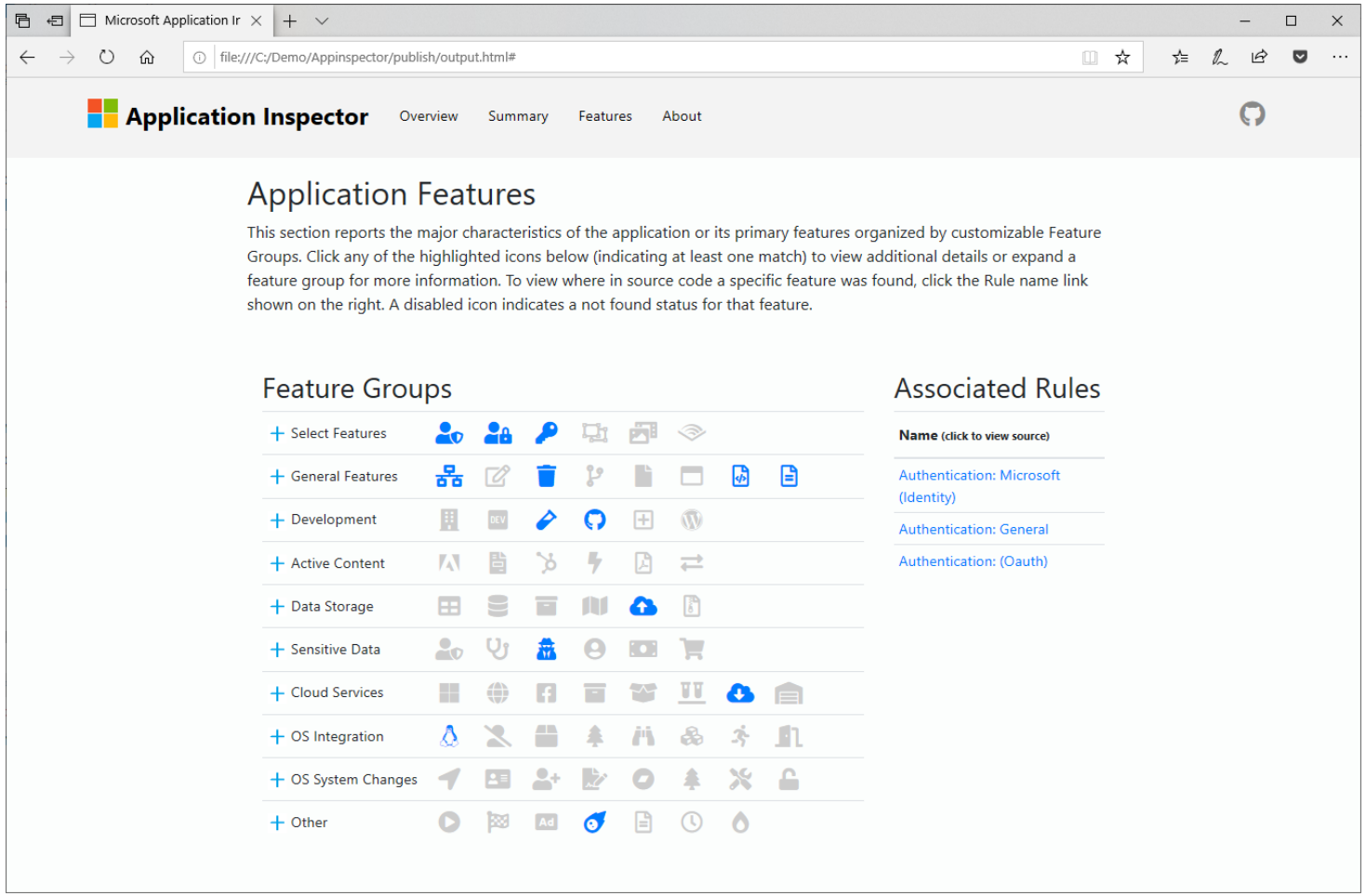
.NET Reproducibility

- 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?

@nielstanis@infosec.exchange

Application Inspector

New & Improved!



@nielstanis@infosec.exchange

Application Inspector

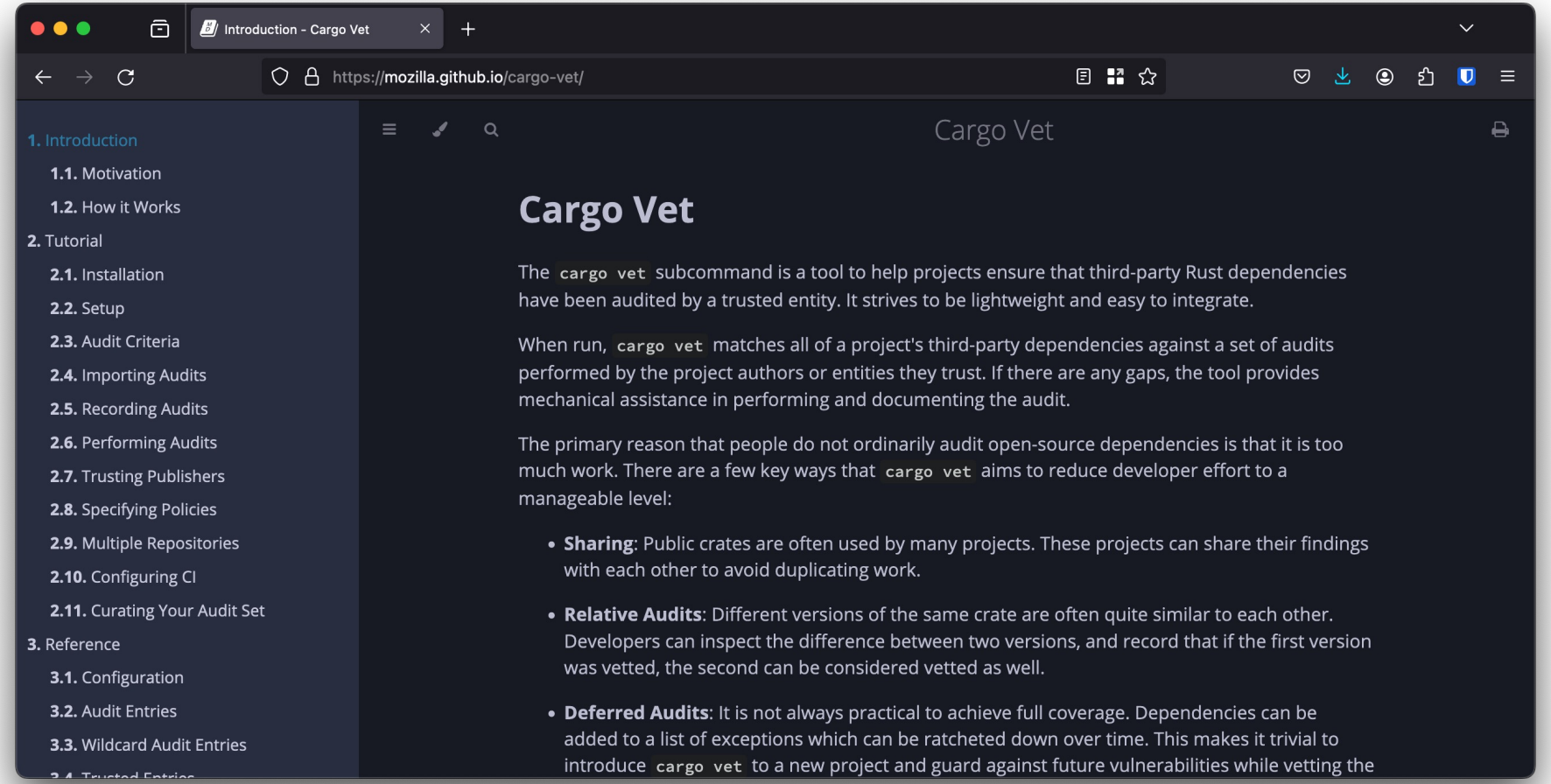
Select Features

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

@nielstanis@infosec.exchange

Community Review

New & Improved!



@nielstanis@infosec.exchange

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 it’s own!
- Room for .NET specific improvements with Fennec CLI & contributions to OpenSSF Scorecard project

Questions?

- https://github.com/nielstanis/ndcsecurity2024/
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- https://www.fennec.dev & https://blog.fennec.dev
- Takk! Thank you!