



OWASP 2024
GLOBAL
AppSec

CONGRESS CENTRE
LISBON
JUNE 24-28



OWASP 2024
GLOBAL
AppSec

CONGRESS CENTRE
LISBON
JUNE 24-28

Assessing 3rd Party Libraries more easily with Security Scorecards

Niels Tanis
Sr. Principal Security Researcher

VERACODE



Who am I?

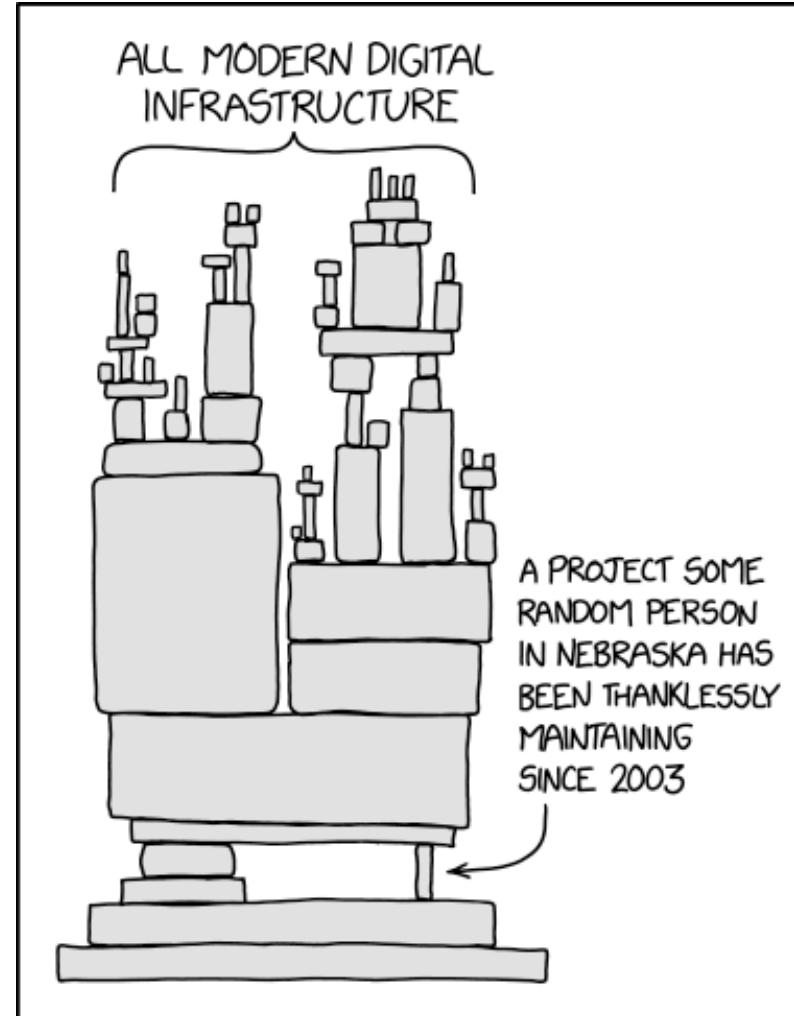
- Niels Tanis
- Sr. Principal Security Researcher
 - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
 - Research on static analysis for .NET apps
 - Enjoying Rust!
- Microsoft MVP – Developer Technologies

VERACODE



Modern Application Architecture

XKCD 2347





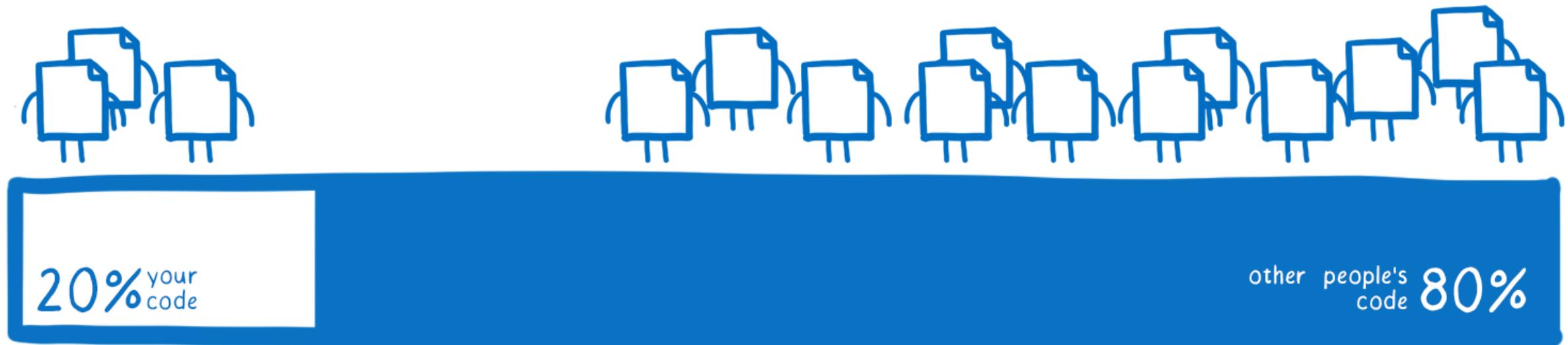
Agenda

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





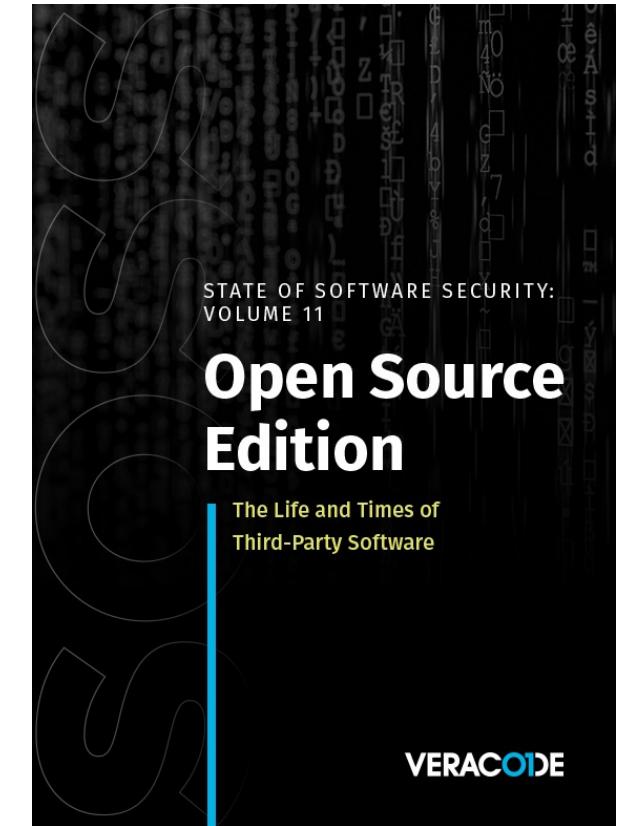
Average codebase composition





State of Software Security v11

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



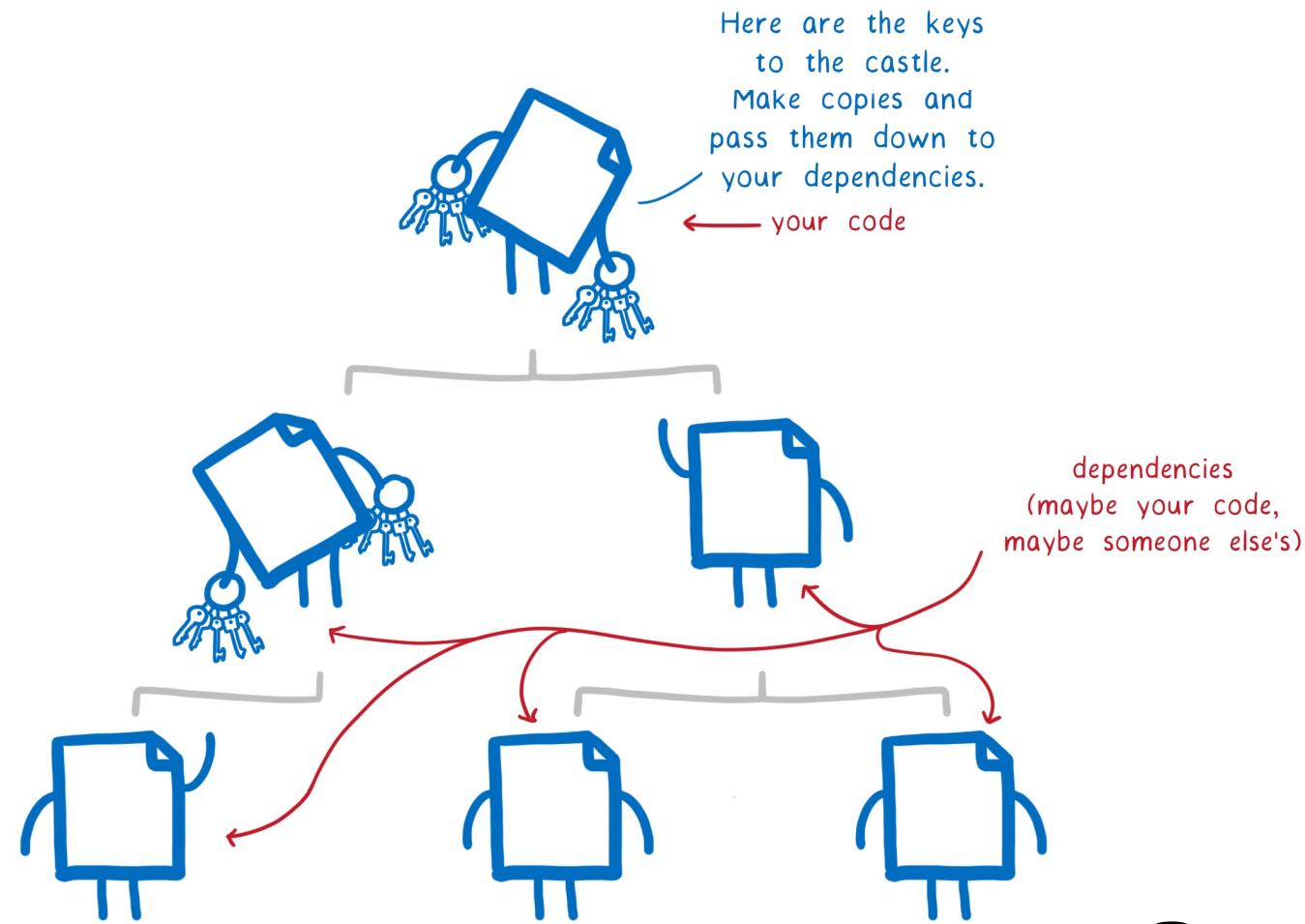


State of Log4j - 2 years later

- 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.

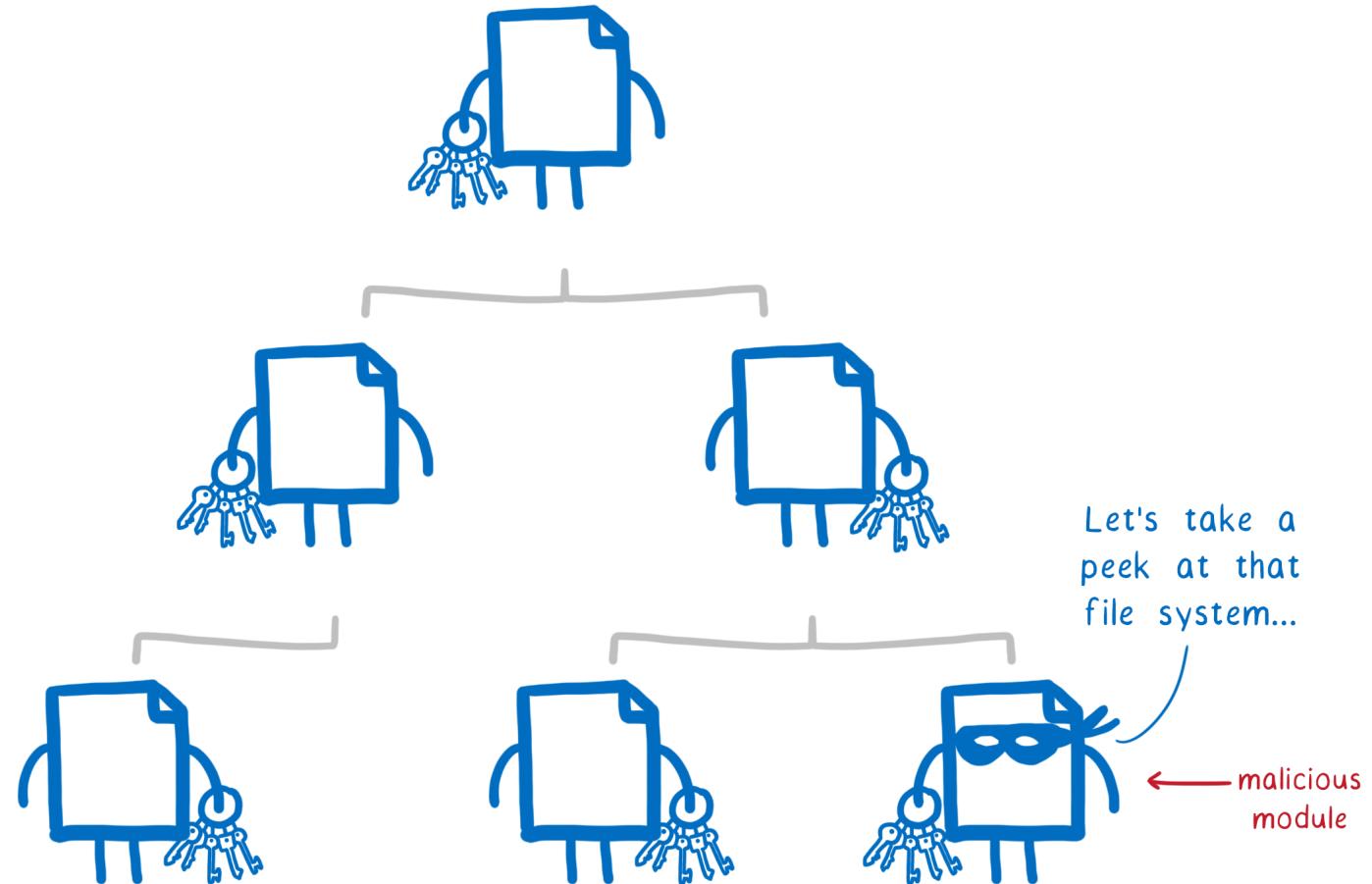


Average codebase composition





Malicious Package





Malicious Package

THN 48 Malicious npm Packages Fou X +

https://thehackernews.com/2023/11/48-malicious-npm-packages-found.html

48 Malicious npm Packages Found Deploying Reverse Shells on Developer Systems

Nov 03, 2023 Newsroom Software Security / Malware



A new set of 48 malicious npm packages have been discovered in the npm repository with capabilities to deploy a reverse shell on compromised systems.



Malicious Package

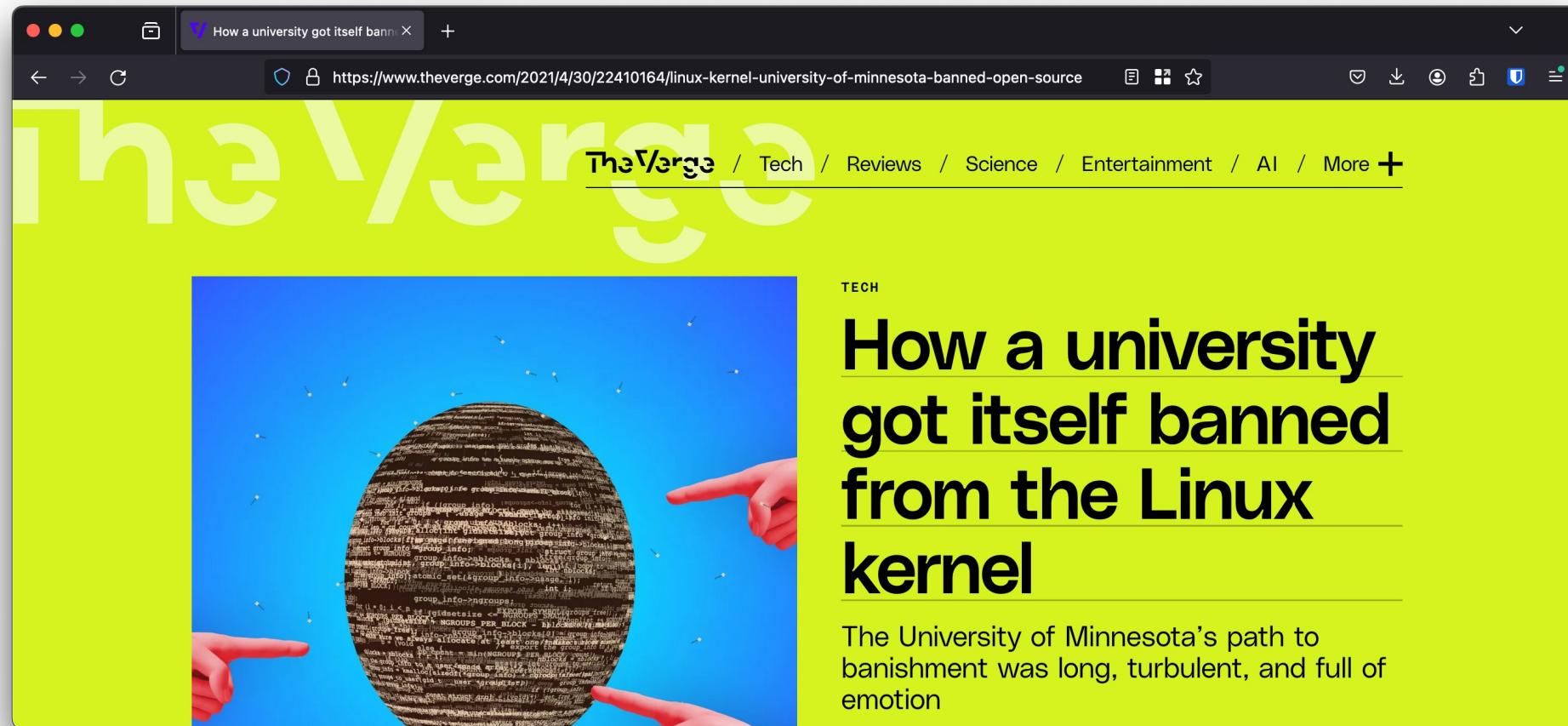
A screenshot of a web browser window showing a blog post from ReversingLabs. The browser has a dark theme with a red tab bar. The main content area shows the ReversingLabs logo (red square with 'RL') and the title 'ReversingLabs Blog'. Below the title is a timestamp 'Threat Research | October 31, 2023'. The main heading is 'IAmReboot: Malicious NuGet packages exploit loophole in MSBuild integrations'. A paragraph of text follows, stating: '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.' In the bottom right corner of the browser window, there is a small purple circular icon with a white upward-pointing arrow.

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.



Hypocrite Commits



TheVerge / Tech / Reviews / Science / Entertainment / AI / More +

TECH

How a university got itself banned from the Linux kernel

The University of Minnesota's path to banishment was long, turbulent, and full of emotion

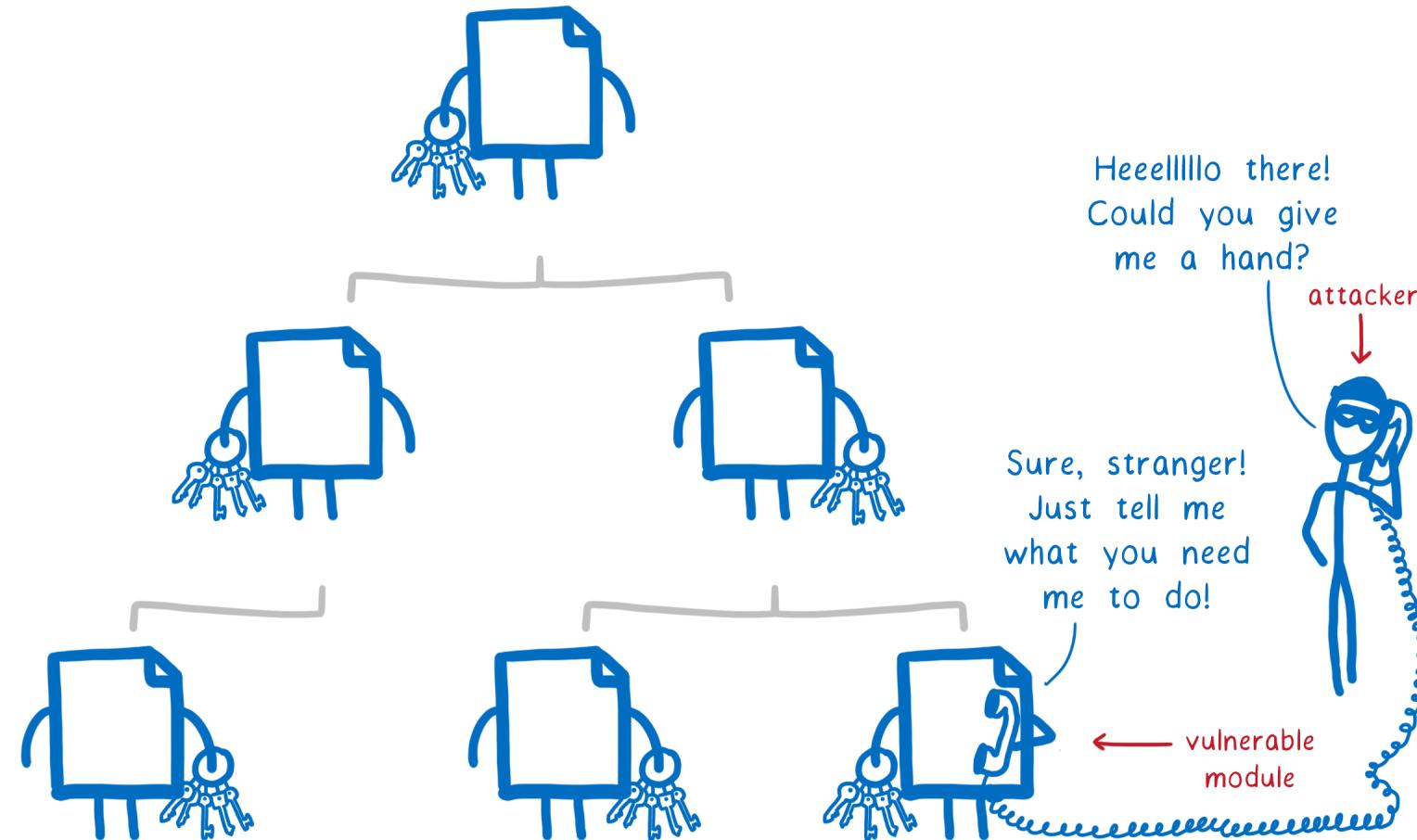


XZ Backdoor

A screenshot of a web browser window displaying an Ars Technica article. The browser has a dark mode interface with a top bar showing the title "Backdoor found in widely used" and a URL bar with the address "https://arstechnica.com/security/2024/03/backdoor-found-in-widely-". The main content area features the Ars Technica logo, a "SUBSCRIBE" button, and a "SIGN IN" button. The article title is "SUPPLY CHAIN ATTACK — Backdoor found in widely used Linux utility targets encrypted SSH connections". A subtitle below the title states "Malicious code planted in xz Utils has been circulating for more than a month." and includes the author's name "DAN GOODIN - 3/29/2024, 7:50 PM".



Vulnerable Package





Vulnerabilities in Libraries

A screenshot of a GitHub issue page. The title of the issue is "Microsoft Security Advisory CVE-2023-36558: .NET Security Feature Bypass Vulnerability #288". The issue is marked as "Open" and was created by user "rbhanda" on Nov 14. The issue body contains a comment from "rbhanda" with the same title. The "Executive summary" section 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." It also mentions: "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 "Discussion" section links to "dotnet/runtime#94726". The right sidebar shows the issue's metadata: Assignees (No one assigned), Labels (Security), Projects (None yet), Milestone (No milestone), Development (No branches or pull requests), and Notifications (Customize).



NPM Audit

```
==== npm audit security report ===
```

```
# Run npm install chokidar@2.0.3 to resolve 1 vulnerability
SEMVER WARNING: Recommended action is a potentially breaking change
```

Low	Prototype Pollution
Package	deep-extend
Dependency of	chokidar
Path	chokidar > fsevents > node-pre-gyp > rc > deep-extend
More info	https://nodesecurity.io/advisories/612



DotNet CLI

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

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

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

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



DotNet CLI

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

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



DotNet CLI

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

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

Project `consoleapp` has the following vulnerable packages
[net8.0]:
Transitive Package      Resolved    Severity   Advisory URL
> Newtonsoft.Json        9.0.1       High       https://github.com/advisories/GHSA-5crp-9r3c-p9vr

nelson@ghost-m2:~/research/consoleapp$
```



Do you know what's inside?

A screenshot of a web browser window showing a blog post from ReversingLabs. The browser has a dark theme with a red tab bar. The main content area shows the ReversingLabs logo (red square with 'RL') and the title 'ReversingLabs Blog'. Below the title is a sub-header 'Threat Research | July 7, 2021'. 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 author bio for 'Karlo Zanki, Reverse Engineer at ReversingLabs' with a 'READ MORE...' link.

RL Third-party code comes with some baggage

https://www.reversinglabs.com/blog/third-party-code-comes-with-some-ba

REVERSINGLABS

ReversingLabs Blog

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?



@nielstanis@infosec.exchange



OpenSSF Scorecards

A screenshot of a web browser displaying the OpenSSF Scorecard homepage. The browser window has a dark theme with a light-colored header bar. The title bar says "OpenSSF Scorecard". The address bar shows the URL "https://securityscorecards.dev". The page content features a large, bold, black text block: "Build better security habits, one test at a time". Above this text, there is a search bar containing the text "openssf scorecard". To the right of the main text, there is a "Star" button with the number "3867". Below the main text, a smaller line of text reads "Quickly assess open source projects for risky practices". At the bottom of the page, there are two orange buttons: "Run the checks" and "Learn more".

openssf
scorecard

Build better security habits, one test at a time

Quickly assess open source projects for risky practices

Run the checks Learn more



OpenSSF Security Scorecards

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

- 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
- What is OpenSSF Scorecard?**

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".

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

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.

Risk Level	Score
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 features a sidebar with links for running checks and learning more, and a central diagram illustrating the "Holistic Security Practices".

The diagram consists of six circles arranged in a hexagonal pattern, all enclosed within a large orange outer circle. The circles are labeled as follows:

- RISK ASSESSMENT (top-left)
- CODE VULNERABILITIES (top-center)
- MAINTENANCE (top-right)
- SOURCE RISK ASSESSMENT (bottom-left)
- CONTINUOUS TESTING (bottom-right)
- HOLISTIC SECURITY PRACTICES (center)

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



Code Vulnerabilities (High)

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

ID	Packages	Summary	Affected versions	Published	Fix
GHSA-32f8-hmr3-7vxg	NuGet/Microsoft.Azure.Storage.DataMovement	Azure Storage Movement Client Library Denial of Service Vulnerability	0.1.0 0.10.1 0.12.0 0.3.0 ...	0.10.0 0.11.0 0.2.0 ... yesterday	Fix available
GHSA-m5vv-6r4h-3vj9	PyPI/azure-identity npm/@azure/identity Maven/com.azure:azure-identity npm/@azure/msal-node NuGet/Microsoft.Identity.Client Go/github.com/Azure/azure-sdk-for-go/sdk/azidentity Maven/com.microsoft.azure:msal4j NuGet/Azure.Identity	Azure Identity Libraries and Microsoft Authentication Library Elevation of Privilege Vulnerability	1.0.0 1.0.0b2 1.0.0b4 1.1.0 ...	1.0.0b1 1.0.0b3 1.0.1 ... yesterday	Fix available
GHSA-rpj9-xjwm-wr6w	NuGet/Umbraco.Commerce	Umbraco Commerce vulnerable to Stored Cross-site Scripting on Print Functionality	12.0.0 12.1.0-rc1	12.1.0 12.1.1 2 weeks ago	Fix available

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



- Does the project use a dependency update tool?
For example Dependabot or Renovate bot?
- Out-of-date dependencies make a project vulnerable to known flaws and prone to attacks.

Maintenance Security Policy (Medium)



- Does project have published security policy?
- E.g. a file named **SECURITY.md** (case-insensitive) in a few well-known directories.
- A security policy can give users information about what constitutes a vulnerability and how to report one securely so that information about a bug is not publicly visible.

Maintenance License (**Low**)



- Does project have license published?
- A license can give users information about how the source code may or may not be used.
- The lack of a license will impede any kind of security review or audit and creates a legal risk for potential users.

Maintenance CII Best Practices (**Low**)



- OpenSSF Best Practices Badge Program
- Way for Open Source Software projects to show that they follow best practices.
- Projects can voluntarily self-certify, at no cost, by using this web application to explain how they follow each best practice.



openssf best practices **passing**

Continuous testing

CI Tests (**Low**)



- Does the project run tests before pull requests are merged?
- The check works by looking for a set of CI-system names in GitHub CheckRuns and Statuses among the recent commits (~30).

Continuous testing

Fuzzing (Medium)



- This check tries to determine if the project uses fuzzing by checking:
 - Added to [OSS-Fuzz](#) project.
 - If [ClusterFuzzLite](#) is deployed in the repository;
- Does it make sense to do fuzzing on managed languages based projects?

Continuous testing

Static Code Analysis (Medium)



- This check tries to determine if the project uses Static Application Security Testing (SAST), also known as static code analysis. It is currently limited to repositories hosted on GitHub.
 - CodeQL
 - SonarCloud
- Definitely room for improvement!

Source Risk Assessment

Binary Artifacts (**High**)



- This check determines whether the project has generated executable (binary) artifacts in the source repository.
- Binary artifacts cannot be reviewed, allowing possible obsolete or maliciously subverted executables.
- There is need for **reproducible** builds!

Source Risk Assessment Branch Protection (**High**)



- This check determines whether a project's default and release branches are protected with GitHub's branch protection or repository rules settings.
 - Requiring code review
 - Prevent force push, in case of public branch all is lost!

Source Risk Assessment

Dangerous Workflow (**Critical**)



- This check determines whether the project's GitHub Action workflows has dangerous code patterns.
 - Untrusted Code Checkout with certain triggers
 - Script Injection with Untrusted Context Variables
- <https://securitylab.github.com/research/github-actions-preventing-pwn-requests/>

Source Risk Assessment Code Review (**Low**)



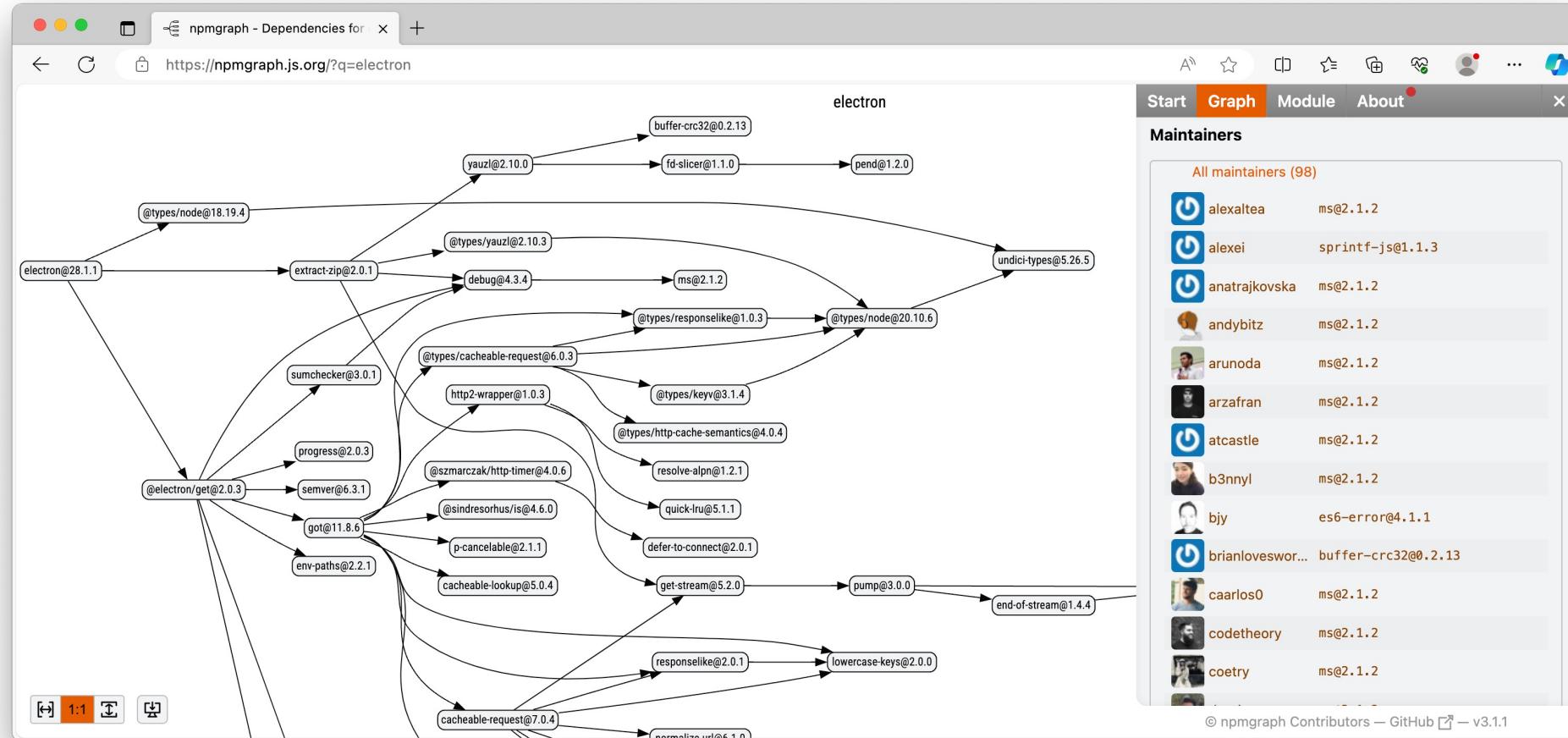
- This check determines whether the project requires human code review before pull requests are merged.
- The check determines whether the most recent changes (over the last ~30 commits) have an approval on GitHub and merger!=committer (implicit review)

Source Risk Assessment Contributors (**Low**)



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

Source Risk Assessment Contributors (Low)



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?

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



Measure?





OpenSSF Annual Report 2023

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





SOSS & OpenSSF Scorecard



The image shows the cover of the Veracode State of Software Security 2024 report. The cover is light blue with a dark blue footer. In the top left corner is the Veracode logo. In the top right corner is the year "2024". The main title "State of Software Security" is in large, bold, black font, centered vertically. Below it is a subtitle "Addressing the Threat of Security Debt" in smaller black font. In the bottom left corner, there is a silhouette of a person sitting on a dark mountain peak, looking at a laptop. The mountain peak is composed of binary code (0s and 1s). The overall background is a gradient from light blue to dark blue.

VERACODE

2024

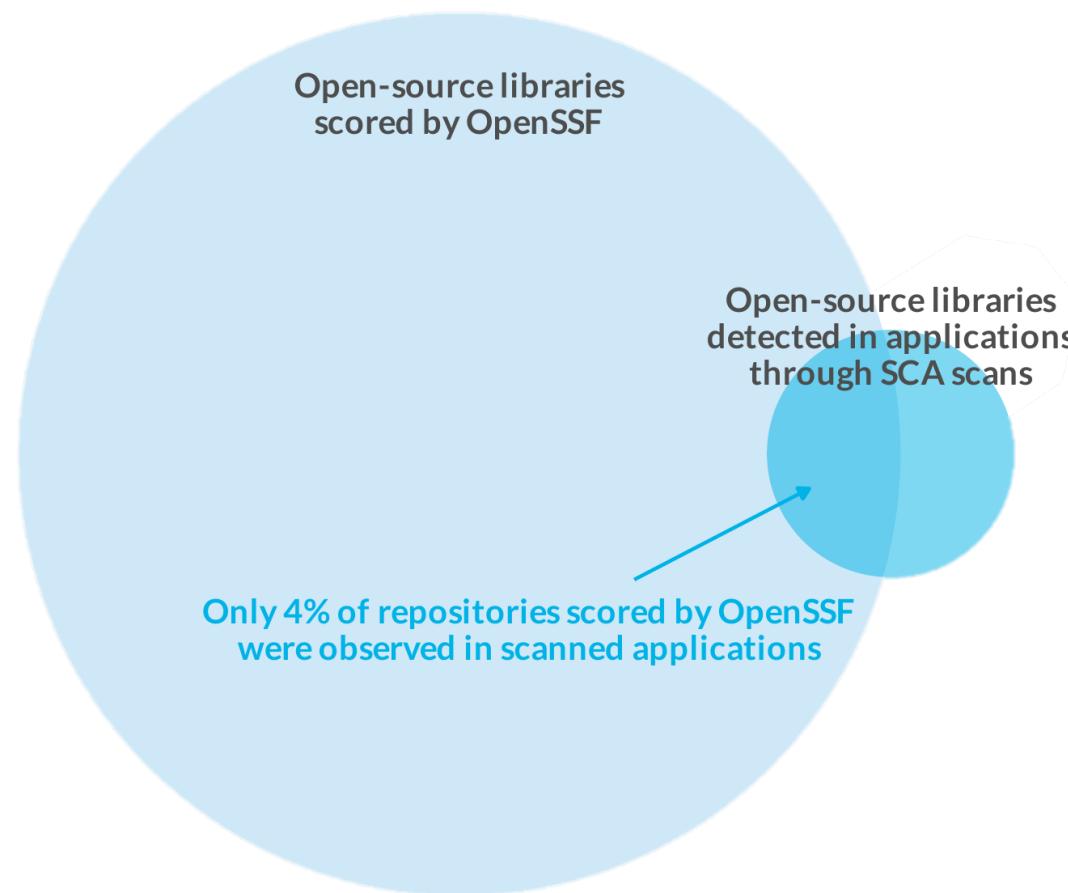
State of
Software
Security

Addressing the
Threat of Security Debt



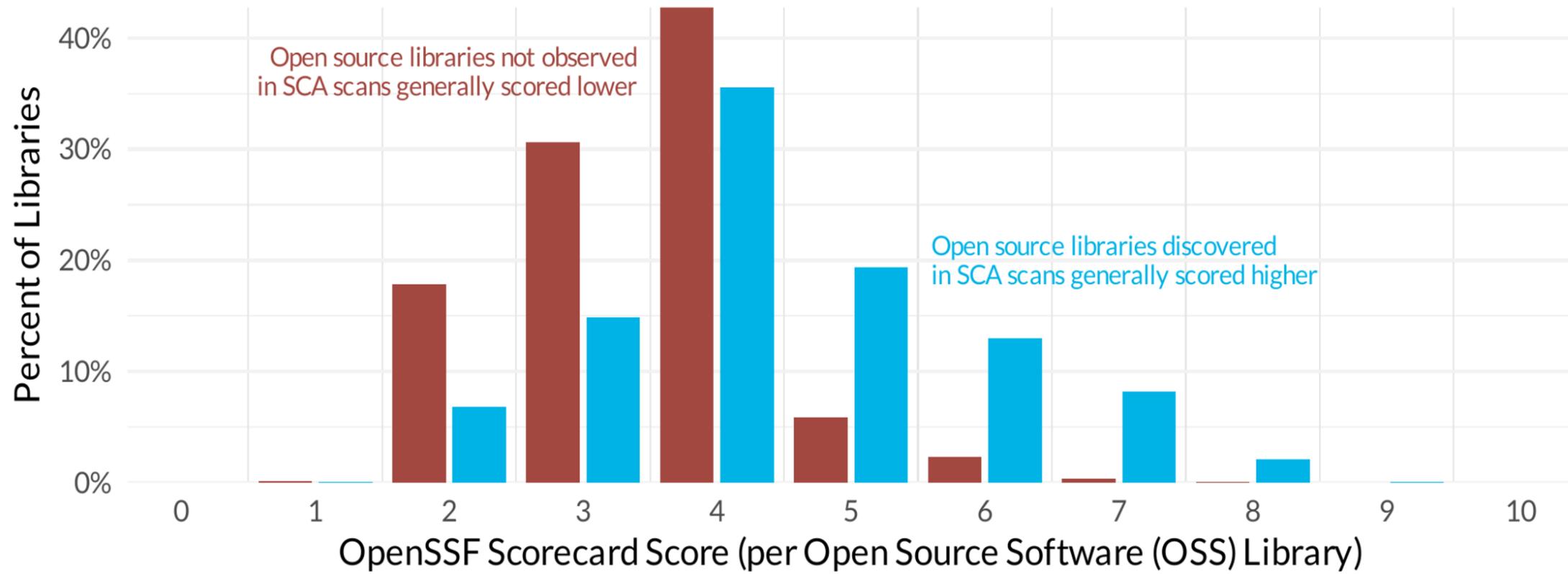


SOSS & OpenSSF Scorecard



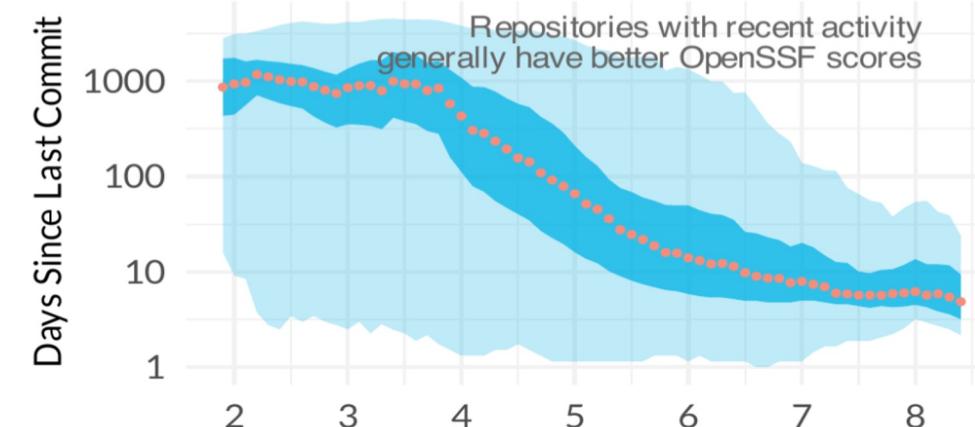
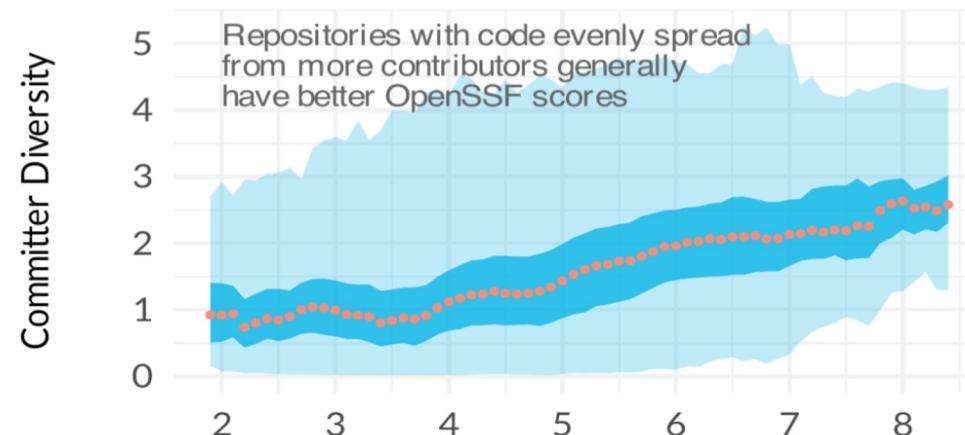
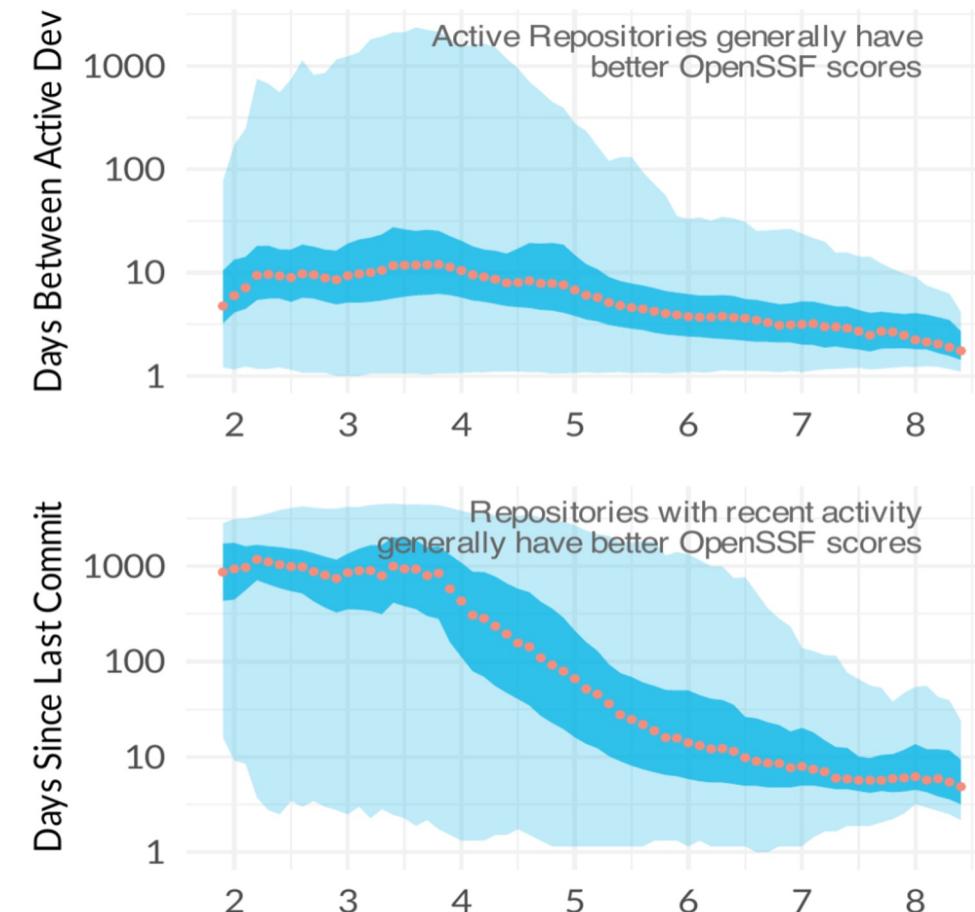
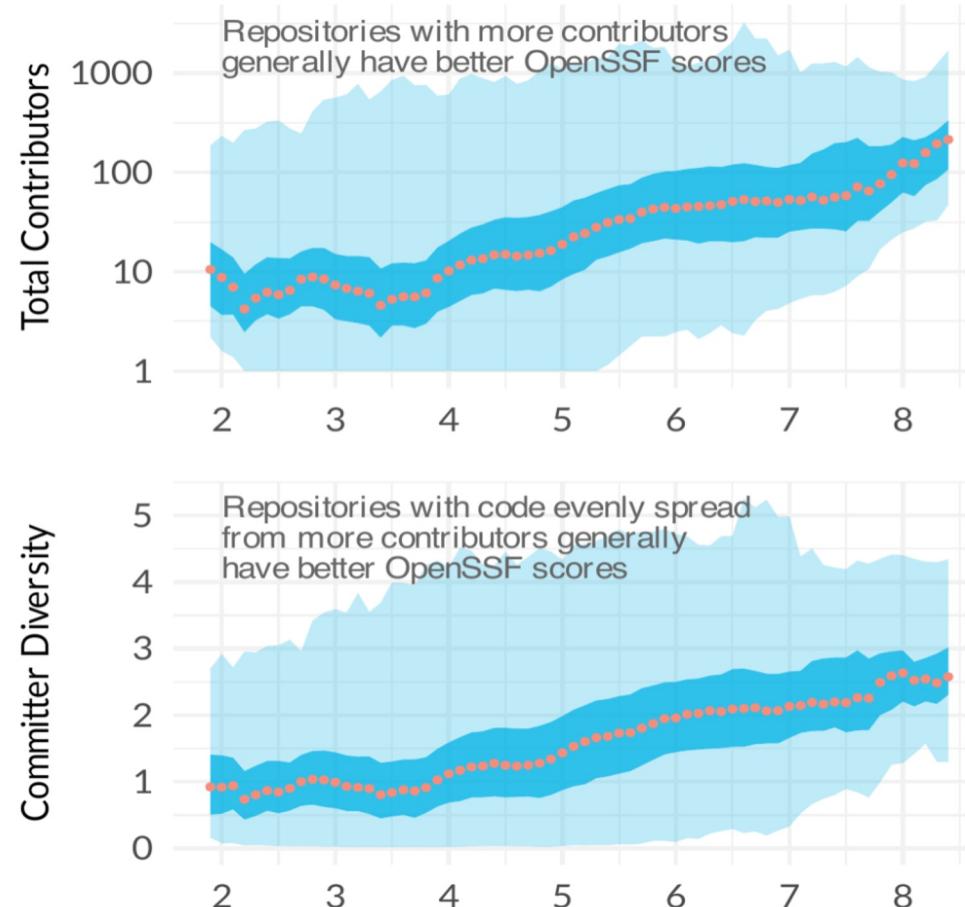


Correlation between SOSS

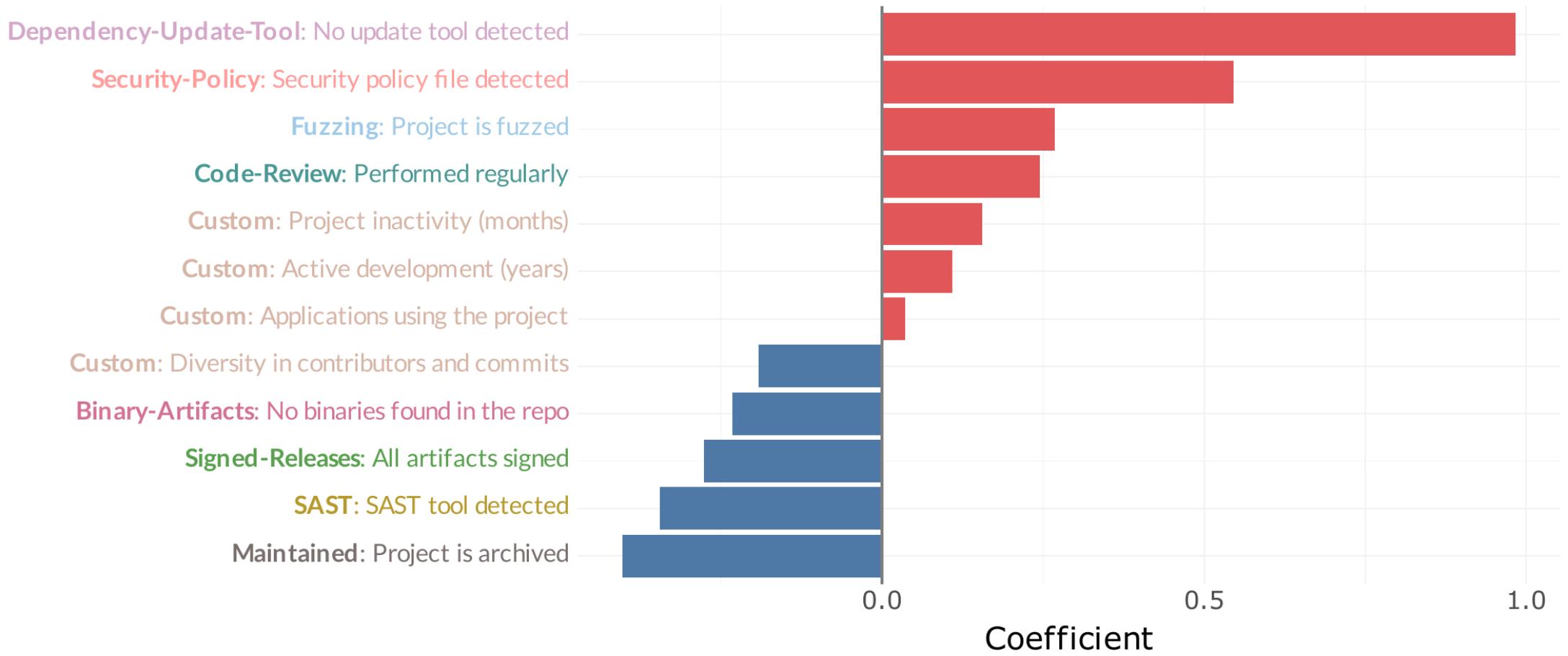




Github commits vs OpenSSF



What really contributes to OSS Security?





What to improve?



Fuzzing managed languages?

New &
Improved!

- 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 and/or Java?

Fuzzing .NET & SharpFuzz



The screenshot shows a web browser window with the following details:

- Title Bar:** Five years of fuzzing .NET with Shar X
- Address Bar:** https://mijailovic.net/2023/07/23/sharpfuzz-anniversary/
- Page Content:**
 - Header:** Nemanja Mijailovic's Blog
 - Section Title:** Five years of fuzzing .NET with SharpFuzz
 - Date:** Jul 23, 2023
 - Text:** 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](#)
 - Text:** 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.
 - Text:** 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

New &
Improved!

The screenshot shows a web browser window with the title "Five years of fuzzing .NET with SharpFuzz". The URL in the address bar is <https://mijailovic.net/2023/07/23/sharpfuzz-anniversary/>. The main content is titled "Trophies". It discusses the growth of bugs found by SharpFuzz, mentioning over 80 entries, and highlights several interesting bugs found using the tool. It also notes two CVEs found using SharpFuzz. The browser interface includes standard navigation buttons, a search bar, and various extension 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

New & Improved!

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

Fuzzomatic: Using AI to Fuzz Rust

New & Improved!

The screenshot shows a web browser window with the title "Introducing Fuzzomatic: Using / X". The URL in the address bar is <https://research.kudelskisecurity.com/2023/12/07/introducing-fuzzomatic-using-ai-to-automatically-fuzz-rust-projects-from-scratch/>. The page content is as follows:

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 Reblog Subscribe ...

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");
        }
    }
}
```

Package Reproducibility



- A build is **reproducible** if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts.
 - .NET Roslyn is deterministic →
.NET8 Console App Demo in repo

Reproducible-Builds.org

A screenshot of a web browser displaying the 'Definitions' page from the Reproducible-Builds.org website. The page has a dark blue header with the site's logo and navigation links for Home, News, Documentation, Tools, Who is involved?, Talks, Events, Continuous tests, and Contribute. The Documentation link is currently selected. The main content area features a large heading 'Definitions' and a sub-section 'When is a build reproducible?'. Below this, there are sections on 'Explanations' and 'The reproducibility of artifacts is verified by bit-by-bit comparison. This is usually performed using'. On the right side, there is a sidebar with links to 'Documentation home', 'Introduction', 'Definitions', 'History', 'Buy-in', 'Making plans', 'Academic publications', 'Achieve deterministic builds', and a long list of related topics including 'commandments of reproducible builds', 'Variations in the build environment', 'SOURCE_DATE_EPOCH', 'Deterministic build systems', 'Volatile inputs can disappear', 'Stable order for inputs', 'Value initialization', 'Version information', and 'Timestamps'.



@nielstanis@infosec.exchange

Application Inspector



The screenshot shows the Microsoft Application Inspector interface running in a Microsoft Edge browser window. The title bar reads "Microsoft Application Ir" and the address bar shows "file:///C:/Demo/Appinspector/publish/output.html#". The main content area has a header "Application Features" with a sub-instruction about viewing major characteristics by feature group. Below this is a "Feature Groups" section containing ten categories with corresponding icons: Select Features, General Features, Development, Active Content, Data Storage, Sensitive Data, Cloud Services, OS Integration, OS System Changes, and Other. To the right is an "Associated Rules" section with a heading "Name (click to view source)" and four listed items: Authentication: Microsoft (Identity), Authentication: General, and Authentication: (Oauth). A GitHub icon is also present in the top right corner of the browser window.



@nielstanis@infosec.exchange

Application Inspector

New &
Improved!

Select Features

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

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 light-colored text. On the left is a sidebar with a table of contents:

- 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". It starts with a brief introduction:

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.

When run, `cargo vet` matches all of a project's third-party dependencies against a set of audits performed by the project authors or entities they trust. If there are any gaps, the tool provides mechanical assistance in performing and documenting the audit.

The primary reason that people do not ordinarily audit open-source dependencies is that it is too much work. There are a few key ways that `cargo vet` aims to reduce developer effort to a manageable level:

- **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

Conclusion & QA

- Scorecard helps security reviewing a 3rd Party 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!
- Look into metrics that matter for you!



Conclusion QA

- <https://github.com/nielstanis/appseceu2024>
 - NPM rocket project
 - NuGet Package Scoring (NET Score)
 - Fennec CLI & Generic tool





Questions?



@nielstanis@infosec.exchange



OWASP 2024
GLOBAL
AppSec

CONGRESS CENTRE
LISBON
JUNE 24-28

THANK YOU

