Reviewing 3rd Party Libraries security using 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

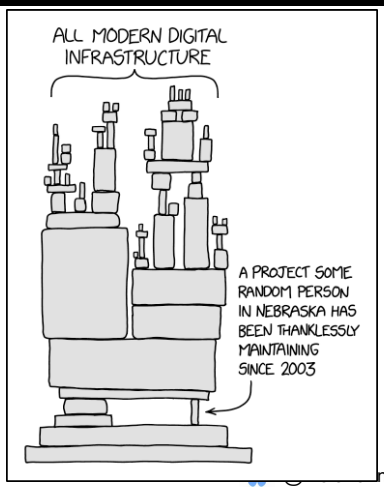
VERACODE

MVP Microsoft® Most Valuable Professional

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Modern Application Architecture
XKCD 2347



SecAppDev

nec.dev @nielstanis@infosec.exchange

https://xkcd.com/2347/

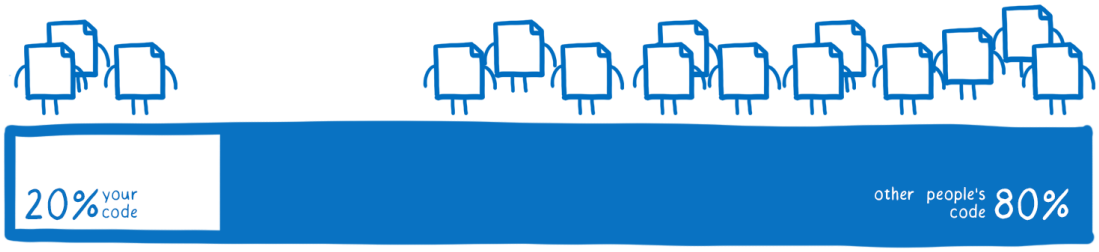
Agenda

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

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Average codebase composition



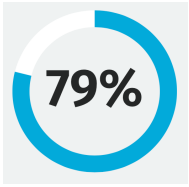
SecAppDev

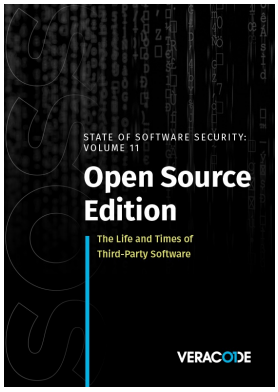
@niels.fennec.dev @nielstanis@infosec.exchange

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.”





SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

State of Log4j - 2 years later

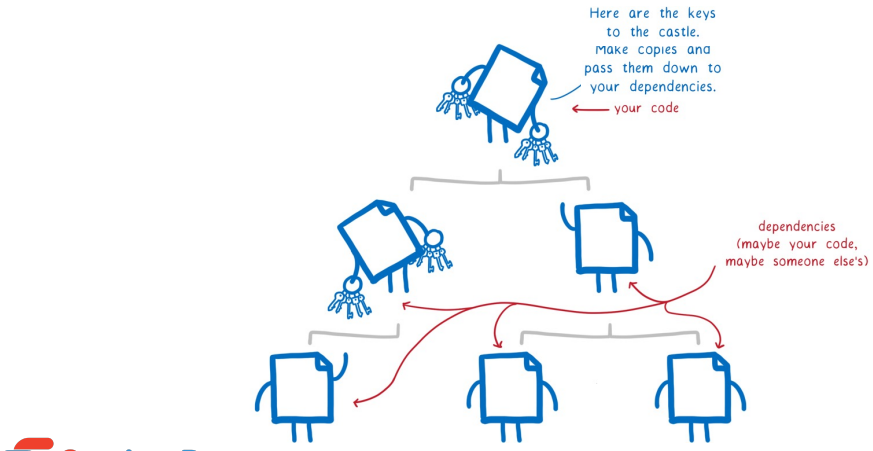
- Analysed our data August-November 2023
  - Total set of almost 39K unique applications scanned
- 2.8% run version vulnerable to Log4Shell
- 3.8% run version patched but vulnerable to other CVE
- 32% rely on a version that's end-of-life and have no support for any patches.

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Average codebase composition

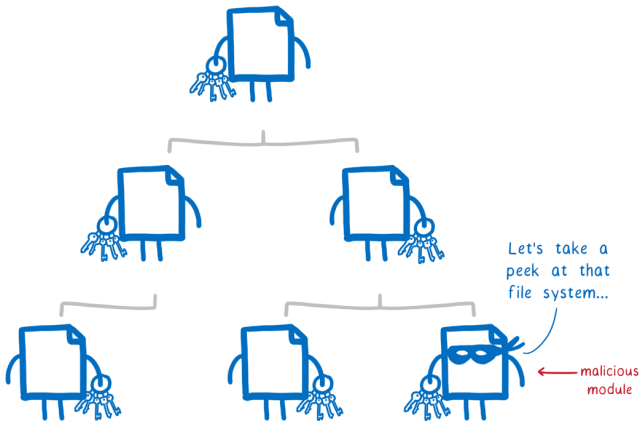


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Malicious Assembly

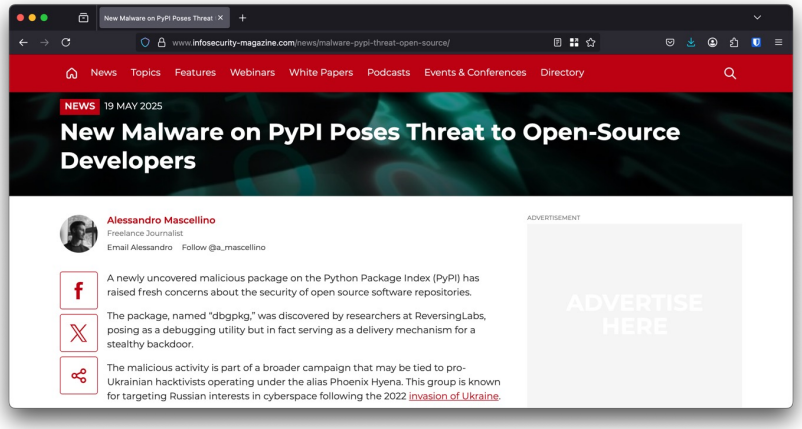


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Malicious Package

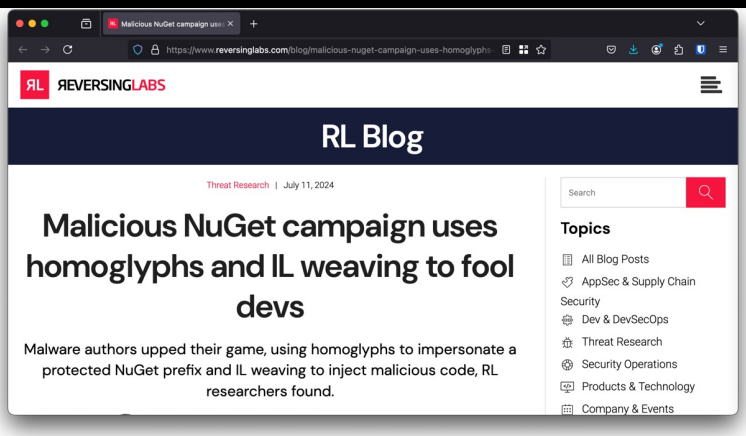


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://www.infosecurity-magazine.com/news/malware-pypi-threat-open-source/

Malicious Package

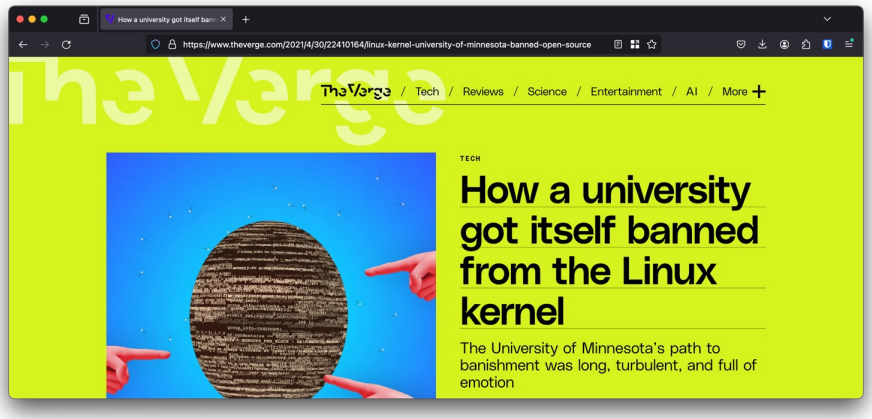


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Hypocrite Commits

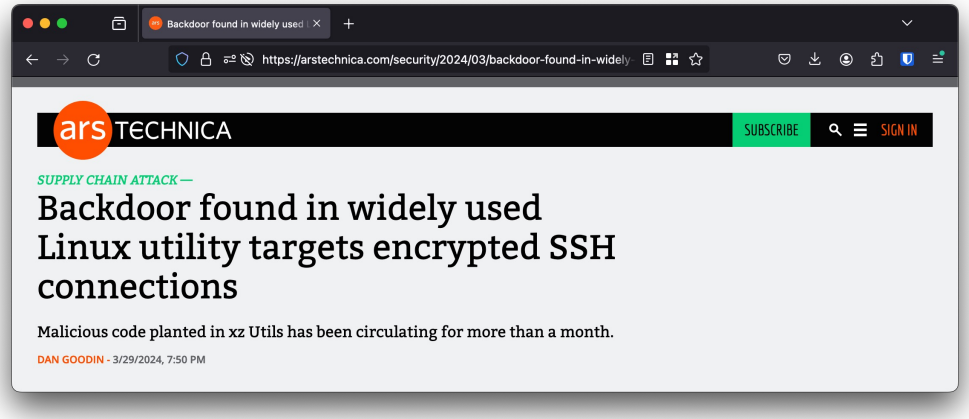


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

XZ Backdoor

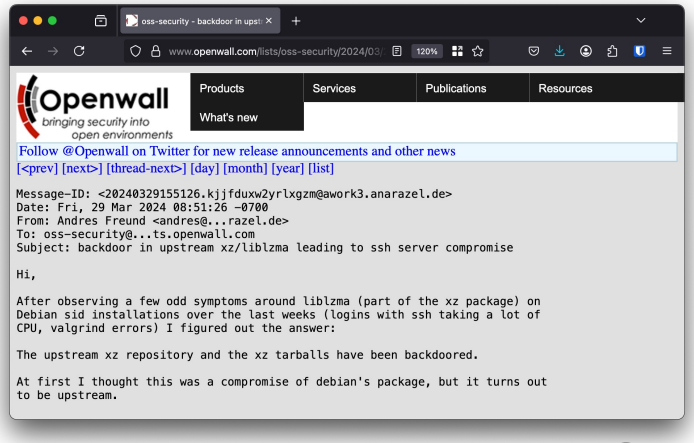


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

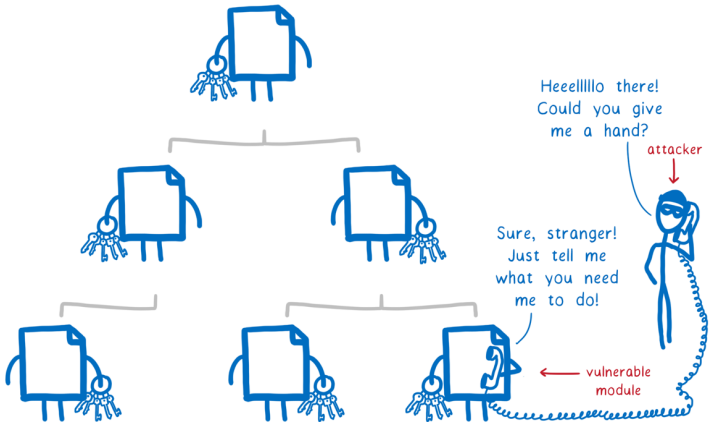
https://arstechnica.com/security/2024/03/backdoor-found-in-widely-used-linux-utility-breaks-encrypted-ssh-connections/

XZ Backdoor



https://www.openwall.com/lists/oss-security/2024/03/29/4

Vulnerable Assembly

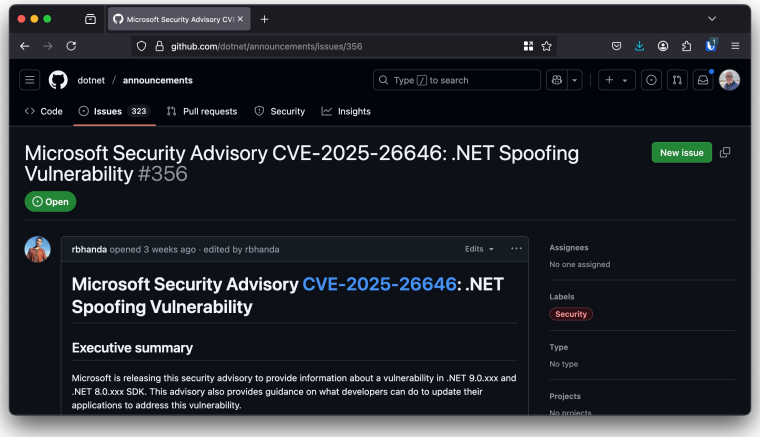


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Vulnerabilities in Libraries

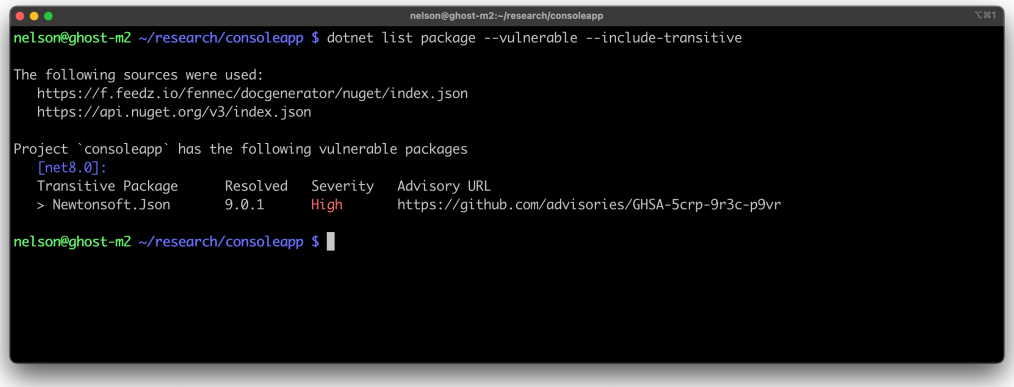


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://github.com/dotnet/announcements/issues/356

DotNet CLI



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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
```

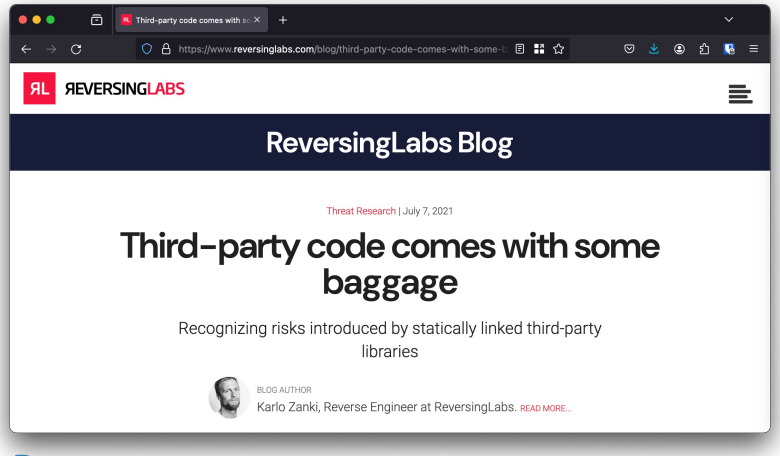
Low	**Prototype Pollution**
Package	deep-extend
Dependency of	chokidar
Path	chokidar > fsevents > node-pre-gyp > rc > deep-extend
More info	https://nodesecurity.io/advisories/612

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://docs.npmjs.com/auditing-package-dependencies-for-security-vulnerabilities

Do you know what's inside?



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

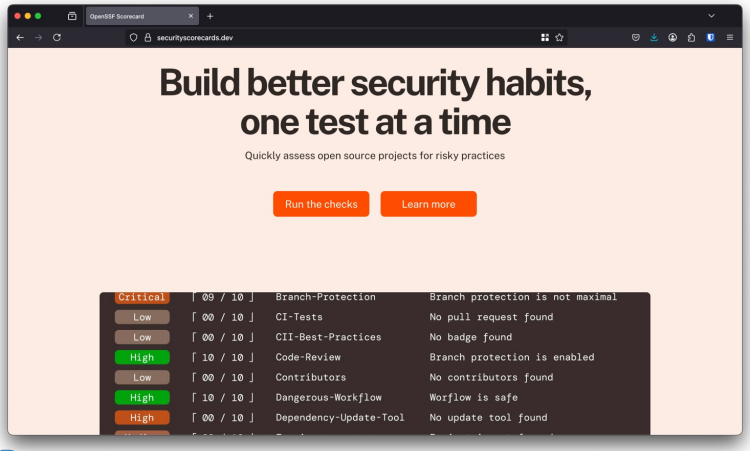
Nutrion Label for Software?

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://securityscorecards.dev/

OpenSSF (OSSF) Scorecard

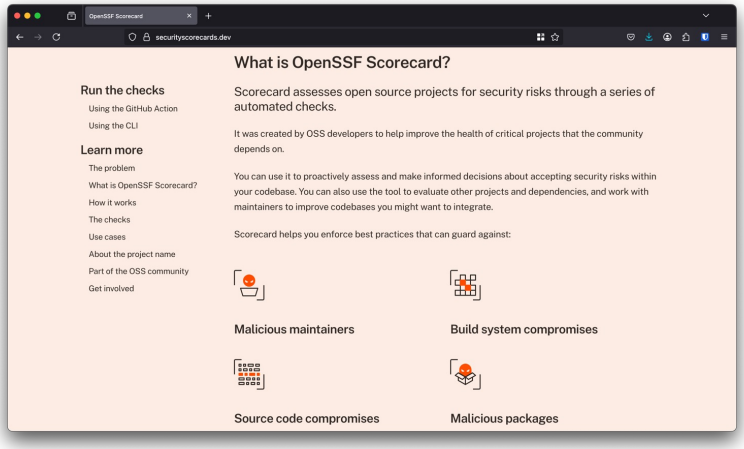


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://securityscorecards.dev/

OSSF Scorecard



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://securityscorecards.dev/

OSSF Scorecard Scoring

- **Total** = Σ(**CheckScore** × **RiskWeight**) / Σ(**RiskWeight**)
- Severity Level → **RiskWeight**

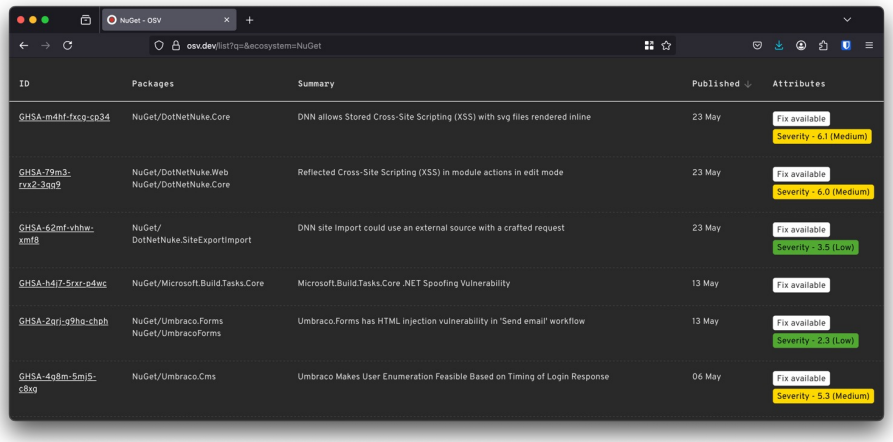
CRITICAL RISK	10
HIGH RISK	7.5
MEDIUM RISK	5
LOW RISK	2.5

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://securityscorecards.dev/

Code Vulnerabilities (High)



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Code Vulnerabilities (High)

- Score = max(0, 10 - number of vulnerabilities)
- Maximum Score (10): No vulnerabilities found
- Penalty: **-1 point** per vulnerability detected
- Minimum Score (0): **10** or more vulnerabilities found

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Maintenance
Dependency-Update-Tool (High)

- It recognises the following tools base configuration:
  - Dependabot (+ recognise commiter)
  - RennovateBot
  - PyUp
- Score is all-or-nothing 0 or 10
- Out-of-date dependencies make a project vulnerable to known flaws and prone to attacks.

Maintenance
How well maintained? (High)

- Immediate Failures (Score = 0)
  - Archived Repository: Project is marked as archived
  - Recently Created: Project created within last 90 days
- Score = min(10, (Total_Activities × 10) / Expected_Activities)
  - Total_Activities = Commits + Issue_Activities
  - Expected_Activities = (90 days ÷ 7 days/week) × 1 activity/week = ~13

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.

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Policy Present: Required (0 points, but necessary for other scoring)
Contains Links/Emails: 6 points (email addresses or URLs for reporting)
Contains Text: 3 points (substantial content beyond links)
Contains Vulnerability Language: 1 point (disclosure terminology)

Maintenance License (Low)

- Does project have license published?
- Possible scores 0 or 9-10
- 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.

SecAppDev

@niels.fennec.dev @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.

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Maintenance
CII Best Practices (Low)

- Basics
  - Project Documentation: Clear description of software purpose, contribution guidelines, and basic documentation
  - FLOSS License: Must be released under a Free/Libre Open Source license, preferably OSI-approved, with license posted in standard location
  - HTTPS Support: All project sites must support HTTPS/TLS
  - Community Engagement: Searchable discussion mechanisms and maintenance evidence

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

Maintenance
CII Best Practices (Low)

- Change Control
  - Version Control: Public, trackable source repository with interim versions
  - Release Management: Unique version identifiers, semantic versioning, and comprehensive release notes
  - Vulnerability Disclosure: Release notes must identify fixed CVEs

@niels.fennec.dev @nielstanis@infosec.exchange

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

Maintenance
CII Best Practices (Low)

- Reporting
  - Bug Tracking: Process for submitting and responding to bug reports with public archives
  - Vulnerability Reporting: Published vulnerability reporting process with timely responses (≤14 days)

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Maintenance
CII Best Practices (Low)

- Quality
  - Build System: Automated, reproducible builds using common/FLOSS tools
  - Testing: Automated test suites with clear execution instructions and continuous integration
  - Code Quality: Compiler warnings enabled and addressed, linter tools usage

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Maintenance
CII Best Practices (Low)

- Security
  - Secure Development: Indicating there is SDLC
  - Cryptography: Use of published algorithms, appropriate key lengths, secure random generation
  - Secure Delivery: MITM-resistant delivery mechanisms, no leaked credentials
  - Vulnerability Management: Timely patching of known vulnerabilities (≤60 days for medium+ severity)

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Maintenance
CII Best Practices (Low)

- Analysis
  - Static Analysis: Required static code analysis tools for major releases
  - Dynamic Analysis: Recommended dynamic analysis including memory safety tools for unsafe languages
  - Timely Fixes: All discovered medium+ severity vulnerabilities must be fixed promptly

@niels.fennec.dev @nielstanis@infosec.exchange

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

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).
  - 2 out of 5 PR's → Score 4
  - 5 out of 5 PR's → Score 10

Continuous testing
Fuzzing (Medium)

- OSS-Fuzz
- ClusterFuzzLite

- Go - Native Go fuzz function
- Haskell - QuickCheck, Hedgehog, SmallCheck, and validity libraries
- Javascript & Typescript - fast-check property-based testing library
- Erlang - proper and eqc (QuickCheck) libraries
- Python: Atheris fuzzing (import atheris)

Continuous testing
Fuzzing (Medium)

- C/C++: LibFuzzer (LLVMFuzzerTestOneInput)
- Rust: Cargo-fuzz (libfuzzer_sys)
- Swift: LibFuzzer (LLVMFuzzerTestOneInput)
- Java: Jazzer fuzzer (com.code_intelligence.jazzer.api.FuzzedDataProvider)
- Does it make sense to do fuzzing managed languages like Java and/or .NET?
- If any present score will be 10, hard check to distinct properly!

Continuous testing
Static Code Analysis (Medium)

- CodeQL: Searches for github/codeql-action/analyze in GitHub workflows
- SonarCloud/SonarQube: Looks for sonar.host.url configuration in pom.xml files
- Snyk: Detects snyk/actions/* in workflows
- Pysa: Searches for facebook/pysa-action
- Qodana: Looks for JetBrains/qodana-action

- If identified → Score 10 except for CodeQL that will also look at PR's

Source Risk Assesement Binary Artifacts (High)

- **Dual Detection**: Uses both file extensions and magic number analysis
- **Content Analysis**: Distinguishes between text and binary content for ambiguous extensions
- **Exception Handling**: Special treatment for validated Gradle wrappers
- **Simple Penalty**: Each binary file reduces the score by 1 point
- **Zero Tolerance**: Aims for completely binary-free repositories

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Source Risk Assesement
Branch Protection (High)

- Tier 1 (3 Points)
  - Prevent force pushes
  - Prevent branch deletion
- Tier 2 (6 Points)
  - RequiredApprovingReviewCount ≥ 1
  - Require PRs prior to code changes (Required = true)
  - Require branch to be up to date before merging
  - Require approval of most recent reviewable push

Source Risk Assesement
Branch Protection (High)

- Tier 3 (8 points)
  - Require branch to pass at least 1 status check before merging
- Tier 4 (9 points)
  - Require at least 2 reviewers
  - Require review from code owners
- Tier 5 (10 points)
  - Dismiss stale reviews when new commits are pushed
  - Include administrators in review requirements

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/

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

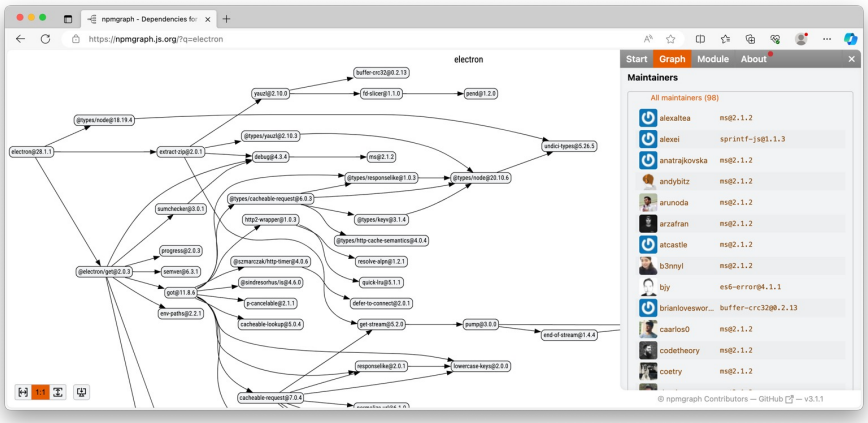
Source Risk Assesement Code Review (Low)

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

Source Risk Assesement Contributors (Low)

- Minimum threshold: 3 companies/organizations (numberCompaniesForTopScore = 3)
- Proportional scoring: Score = (number of entities / 3) × 10
- Maximum score: 10 points when ≥ 3 different organizations/companies are found
- Relying on single contributor is a risk for sure!
- What about a large list of contributors?

Source Risk Assesement Contributors (Low)



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Build Risk Assesement
Pinned Dependencies (High)

- Does the project pin dependencies used during its build and release process.
- For .NET **RestorePackagesWithLockFile** in MSBuild results in **packages.lock.json** file containing versioned dependency tree with hashes
- If Workflow is present what about the Actions used?
- Docker Image uses SHA256 digest

SecAppDev

@niels.fennec.dev @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.

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://securitylab.github.com/research/github-actions-preventing-pwn-requests/

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.
- Any packager workflow detected will give score 10.

Build Risk Assessment
Signed Releases (High)

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

What about GitLab?

- Checks not Supported:
  - Branch-Protection - **High**
  - Contributors - **Low**
  - Dangerous-Workflow - **Critical**
  - Dependency-Update-Tool - **High**
  - SAST – **Medium**
  - Token-Permissions - **High**

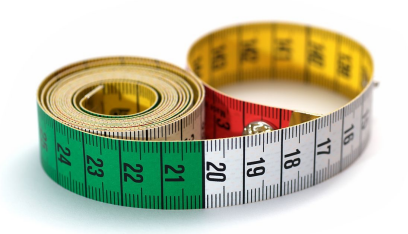
Demo OpenSSF Scorecard
Fennec CLI

```
Running checks
```

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Measure?

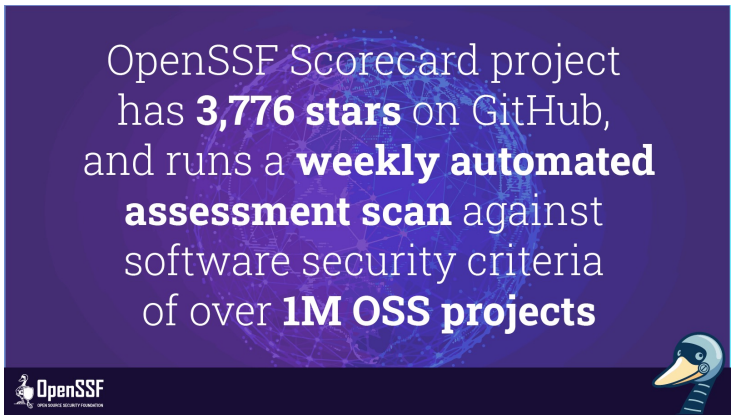


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

OpenSSF Annual Report 2023



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

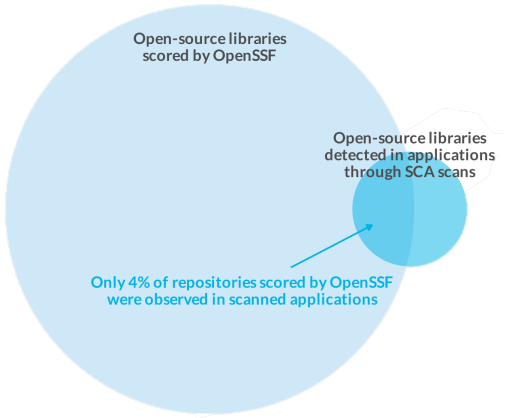
https://openssf.org/download-the-2023-openssf-annual-report/

SOSS & OpenSSF Scorecard

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

SOSS & OpenSSF Scorecard

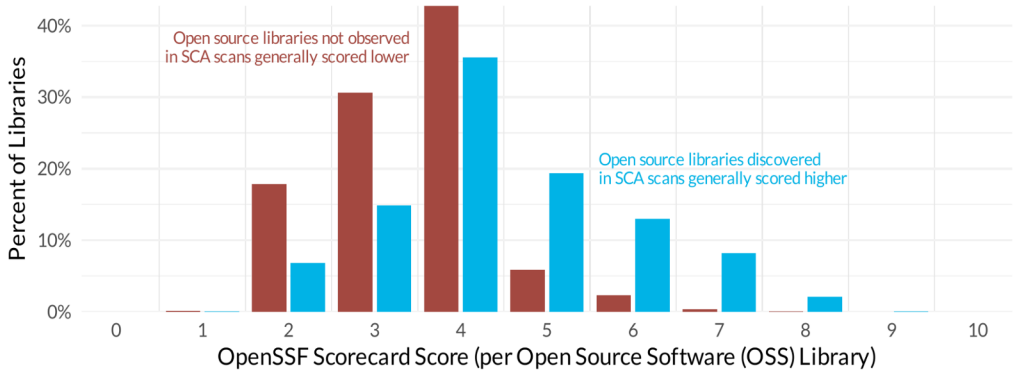


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

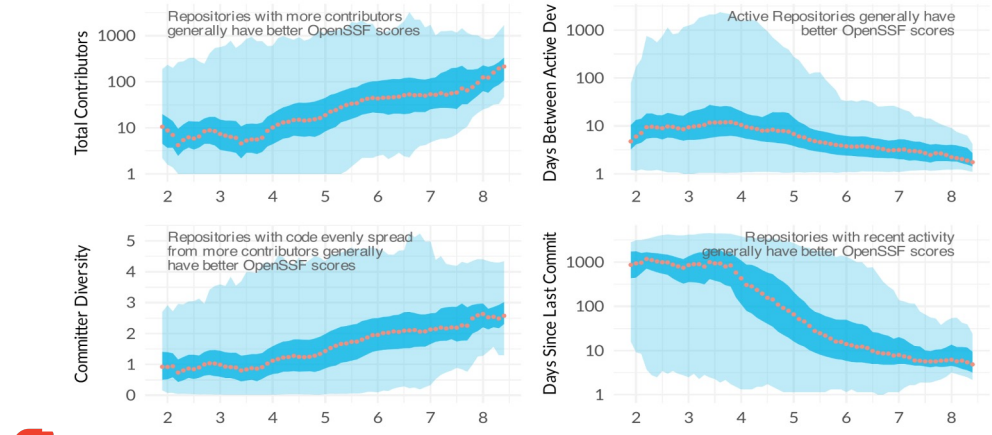
Correlation between SOSS



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

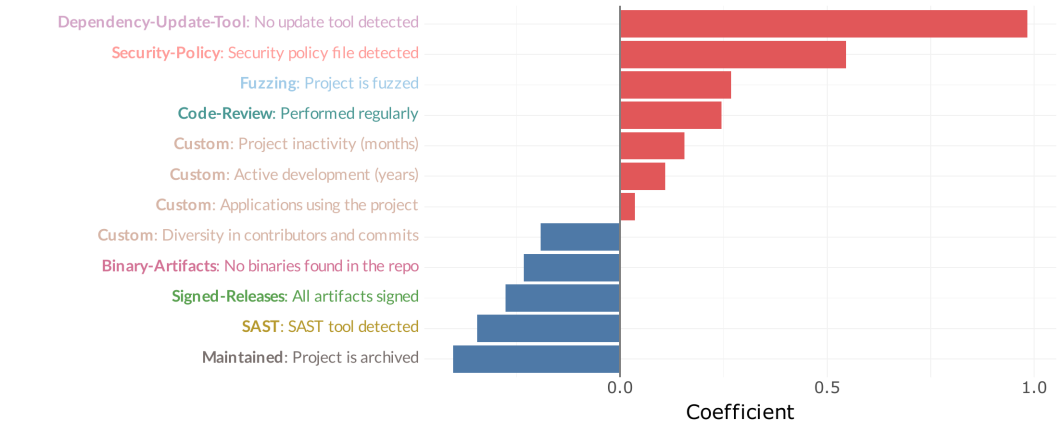
Github commits vs OpenSSF



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

What really contributes to OSS Security?



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

What can we improve?

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Fuzzing Managed Languages?

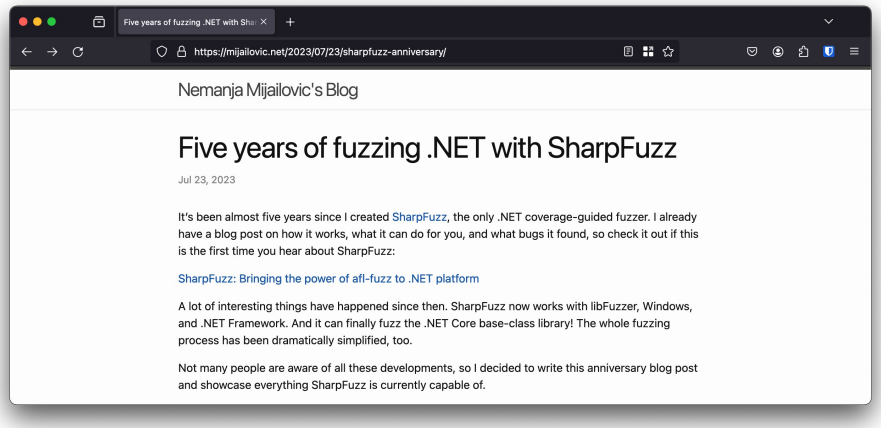
- Fuzzing, or fuzz testing
  - Automated software testing method that uses a wide range of ***invalid*** and unexpected data as input to find flaws
- Definitely good for finding C/C++ memory issues
- Can it be of any value with managed languages like .NET and/or Java?

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Fuzzing .NET & SharpFuzz

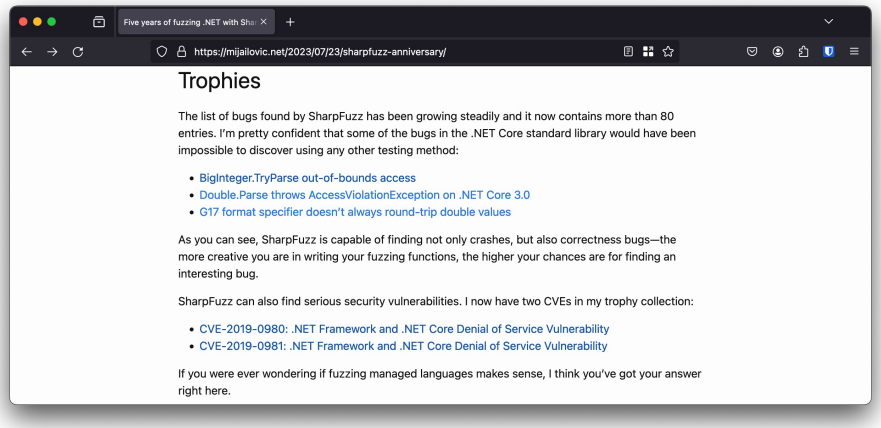


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Fuzzing .NET & SharpFuzz



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Fuzzing .NET – Jil JSON Serializer

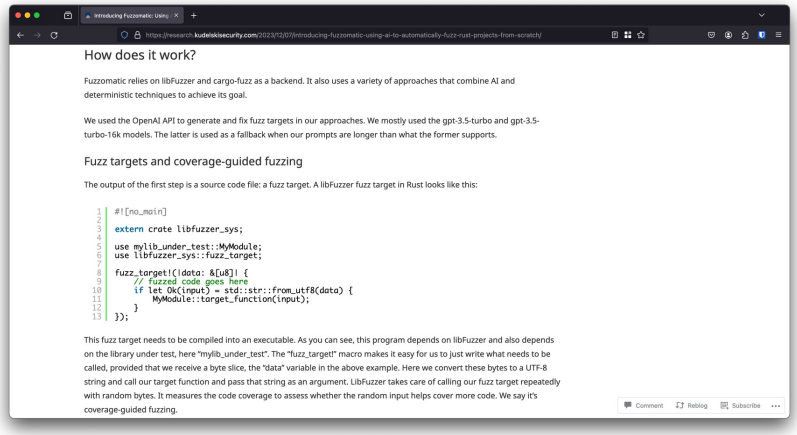
```
public static void Main(string[] args)
{
        SharpFuzz.Fuzzer.OutOfProcess.Run(stream => {
        try
        {
                using (var reader = new System.IO.StreamReader(stream))
                JSON.DeserializeDynamic(reader);
        }
        catch (DeserializationException) { }
        });
}
```

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Fuzzomatic: Using AI to Fuzz Rust



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:

```
#![no_main]

extern crate libfuzzer_sys;

use mylib_under_test::MyModule;
use libfuzzer_sys::fuzz_target;

fuzz_target!(|data: &[u8]| {
    // fuzzed code goes here
    if let Ok(input) = std::str::from_utf8(data) {
        MyModule::target_function(input);
    }
});
```

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.

SecAppDev

@niels.fennec.dev @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);
}
```

SecAppDev

@niels.fennec.dev @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);
    }
}
```

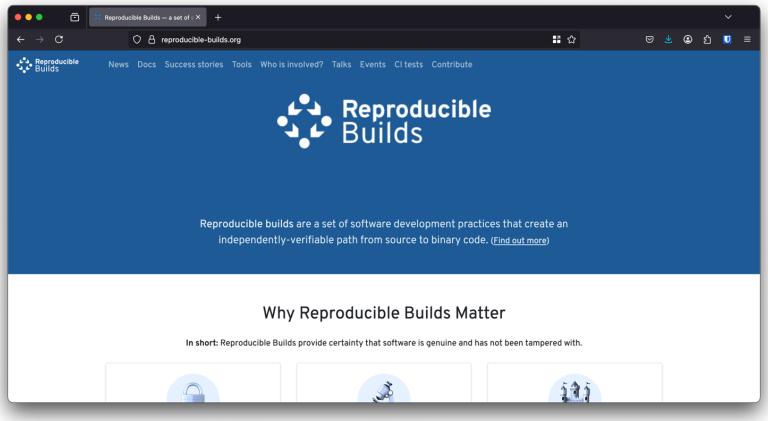
```
public static class Reporting
{
    public static byte[] GenerateCustomerReportOverview(string ID)
    {
        return System.IO.File.ReadAllBytes($"./data/{ID}.pdf");
    }
}
```

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Reproducible Builds

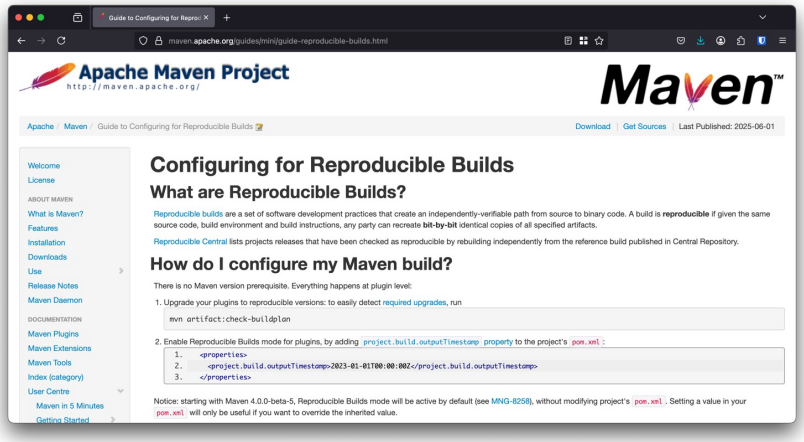


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://reproducible-builds.org/

Maven Reproducible Builds



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://maven.apache.org/guides/mini/guide-reproducible-builds.html

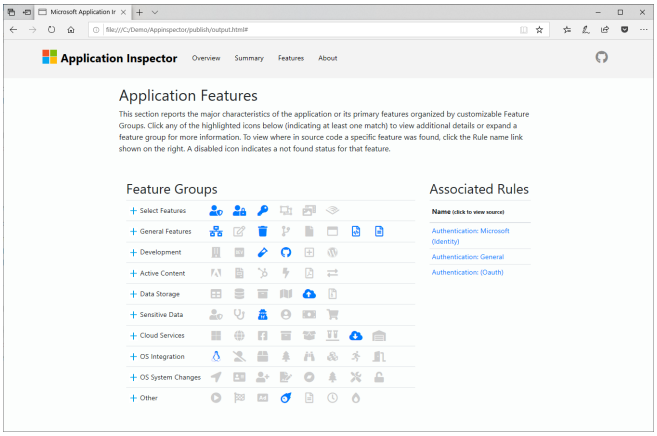
.NET Reproducibility

- Reproducible builds → independently-verifiable path from source to binary code.
- .NET Roslyn Deterministic Inputs
- How reproducible is a simple console app?
- Fennec Diff

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Application Inspector

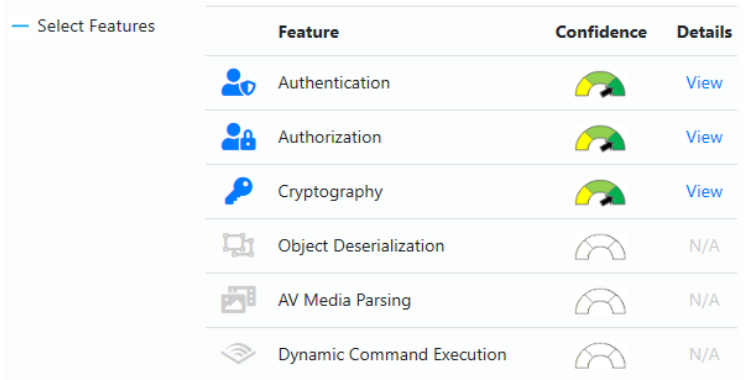


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://github.com/microsoft/ApplicationInspector

Application Inspector

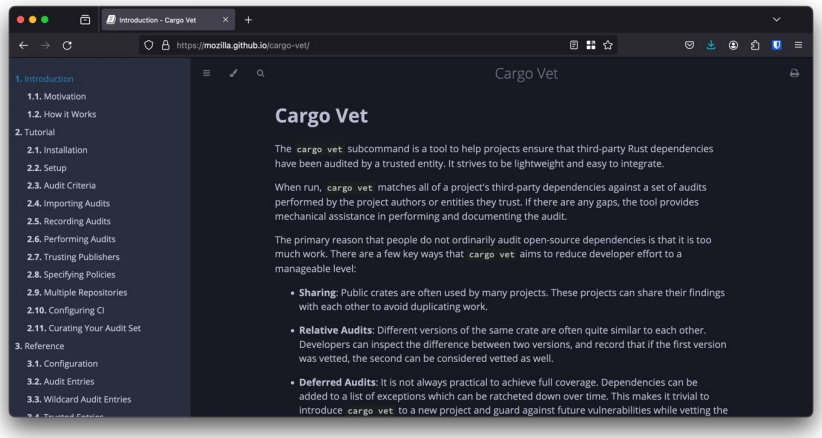


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://github.com/microsoft/ApplicationInspector

Community Review

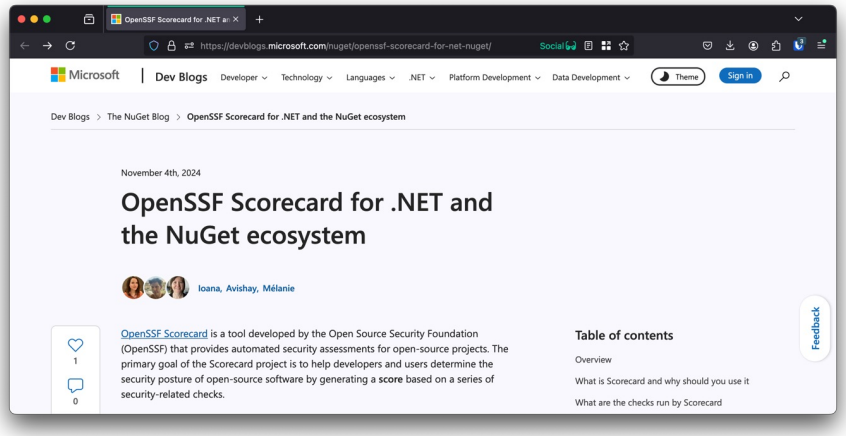


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

NuGet Blog



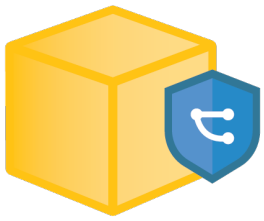
SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://devblogs.microsoft.com/nuget/openssf-scorecard-for-net-nuget/

Secure Supply Chain Consumption Framework (S2C2F) Project

- The Secure Supply Chain Consumption Framework (S2C2F) is a security assurance and risk reduction process that is focused on securing how developers consume open source software.

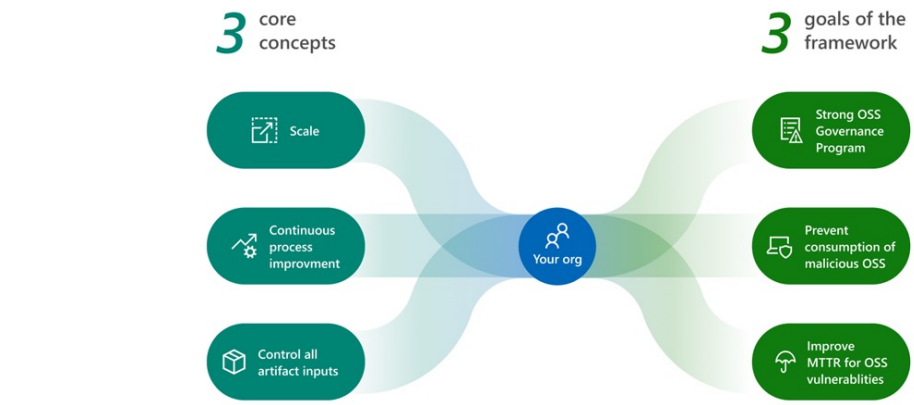


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://github.com/ossf/s2c2f/blob/main/specification/framework.md#about-the-secure-supply-chain-consumption-framework

Secure Supply Chain Consumption Framework (S2C2F) Project



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://github.com/ossf/s2c2f/blob/main/specification/framework.md#about-the-secure-supply-chain-consumption-framework

Secure Supply Chain Consumption Framework (S2C2F) Project

Level 1	Level 2	Level 3	Level 4
Minimum OSS Governance Program • Use package managers [ING-1] • Local copy of artifact [ING-2] • Scan with known vulns [SCA-1] • Scan for software licenses [SCA-2] • Inventory OSS [INV-1] • Manual OSS updates [UPD-1]	**Secure Consumption and Improved MTTR** • Scan for end of life [SCA-3] • Have an incident response plan [INV-2] • Auto OSS updates [UPD-2] • Alerts on vulns at PR time [UPD-3] • Audit that consumption is through approved ingestion method [AUD-2] • Validate integrity of OSS [AUD-3] • Secure package source file configuration [ENF-1]	**Malware Defense and Zero-Day Detection** • Deny list capability [ING-3] • Clone OSS source [ING-4] • Scan for malware [SCA-4] • Proactive security reviews [SCA-5] • Enforce OSS provenance [AUD-1] • Enforce consumption from curated feed [ENF-2]	**Advanced Threat Defense** • Validate the SBOMs of OSS consumed [AUD-4] • Rebuild OSS on trusted infrastructure [REB-1] • Digitally sign rebuilt OSS [REB-2] • Generate SBOM for rebuilt OSS [REB-3] • Digitally sign protected SBOMs [REB-4] • Implement fixes [FIX-1]

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://github.com/ossf/s2c2f/blob/main/specification/framework.md#about-the-secure-supply-chain-consumption-framework

Conclusion

- 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 its own!
- Look into frameworks like S2C2F to help out

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Conclusion

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


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

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Merci! Bedankt! Thanks!

- https://github.com/nielstanis/secappdev25scorecard/
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- https://www.fennec.dev
  https://blog.fennec.dev

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange