Beyond Trust: Building Community-Driven Security Analysis for Your .NET Software Supply Chain

Niels Tanis

Sr. Principal Security Researcher

VERACODE NDC { Manchester }

Who am I?

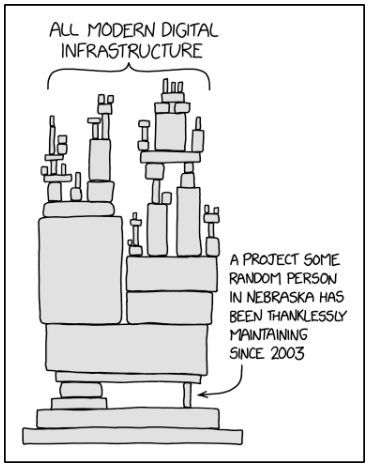
- Niels Tanis
- Sr. Principal Security Researcher
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - Research on static analysis for .NET apps
  - Enjoying Rust!
- Microsoft MVP – Developer Technologies

VERACODE

Microsoft® Most Valuable Professional

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

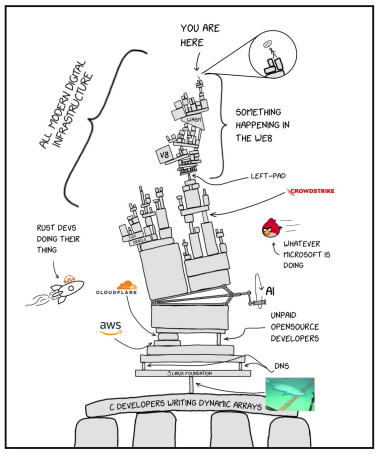
Modern Application Architecture
XKCD 2347



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

https://xkcd.com/2347/

You are here 😂



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

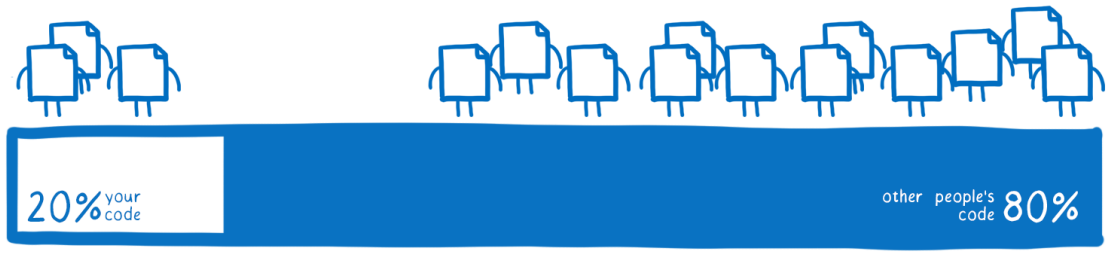
https://xkcd.com/2347/

Agenda

- Security challenges in our .NET software supply chain
- How can we start securing our supply-chain?
- Fennec Labs
  - Understanding what projects do for security
  - Know what's inside package
  - Review package
  - Feeback & sharing
  - Future and idea's
- Conclusion and Q&A

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

Average codebase composition



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

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

State Of Software Security 2025

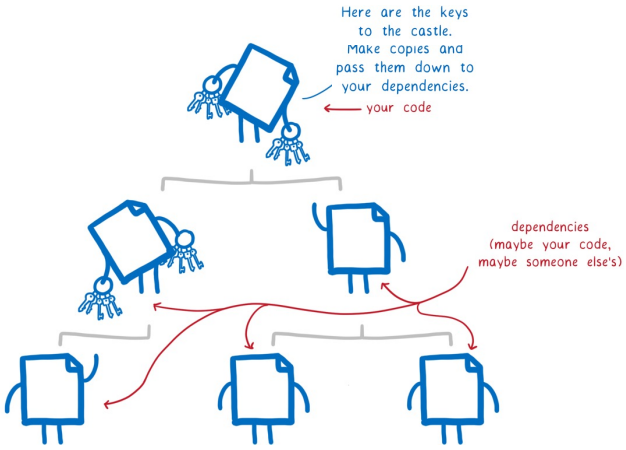
64%
of applications have flaws in **first-party code**

70%
of applications have flaws in **third-party code**

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

https://www.veracode.com/wp-content/uploads/2025/02/State-of-Software-Security-2025.pdf

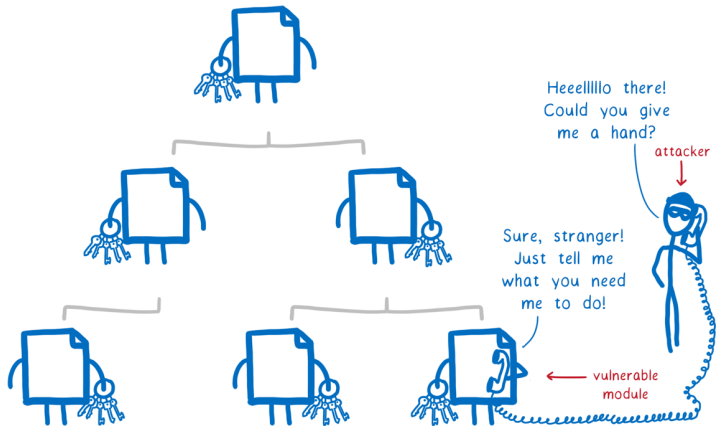
Average codebase composition



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

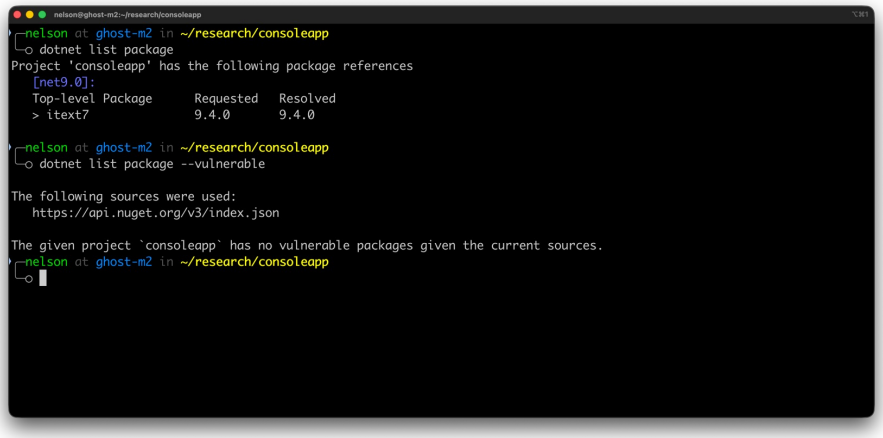
https://hacks.mozilla.org/2019/11/announcing-the-bytecode-alliance/

Vulnerable Assembly



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

DotNet CLI


nelson@ghost-m2:~/research/consoleapp
nelson at ghost-m2 in ~/research/consoleapp
dotnet list package
Project 'consoleapp' has the following package references
[net9.0]:
Top-level Package Requested Resolved
> itext7 9.4.0 9.4.0
nelson at ghost-m2 in ~/research/consoleapp
dotnet list package --vulnerable
The following sources were used:
https://api.nuget.org/v3/index.json
The given project `consoleapp` has no vulnerable packages given the current sources.
nelson at ghost-m2 in ~/research/consoleapp

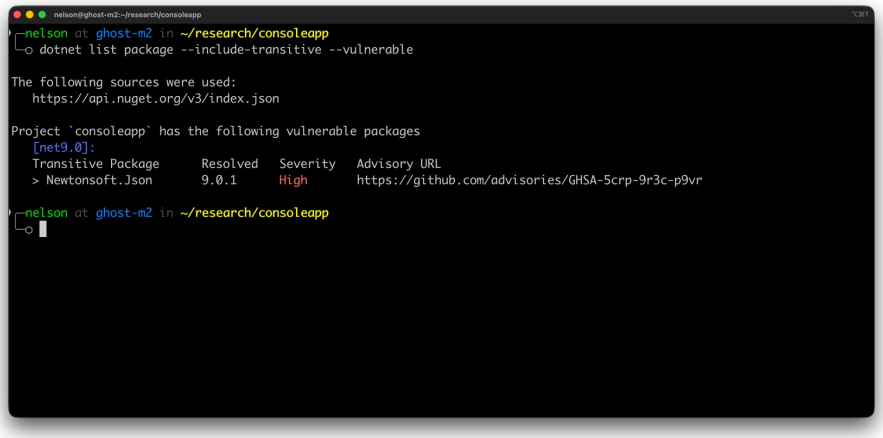
DotNet CLI

```
nelson@ghost-m2:~/research/consoleapp
nelson at ghost-m2 in ~/research/consoleapp
└─○ dotnet list package --include-transitive
Project 'consoleapp' has the following package references
   [net9.0]:
   Top-level Package      Requested   Resolved
   > itext7               9.4.0       9.4.0

   Transitive Package                                                Resolved
   > itext                                                           9.4.0
   > itext.commons                                                   9.4.0
   > 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
   > Microsoft.NETCore.Platforms                                     1.1.1
   > Microsoft.NETCore.Targets                                       1.1.3
   > Microsoft.Win32.Primitives                                      4.3.0
   > Microsoft.Win32.Registry                                        4.3.0
   > Newtonsoft.Json                                                 9.0.1
```

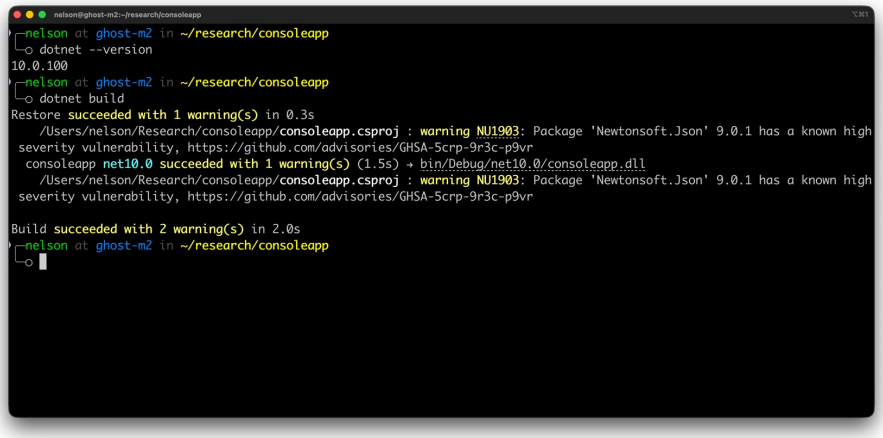

@niels.fennec.dev @nielstanis@infosec.exchange

DotNet CLI

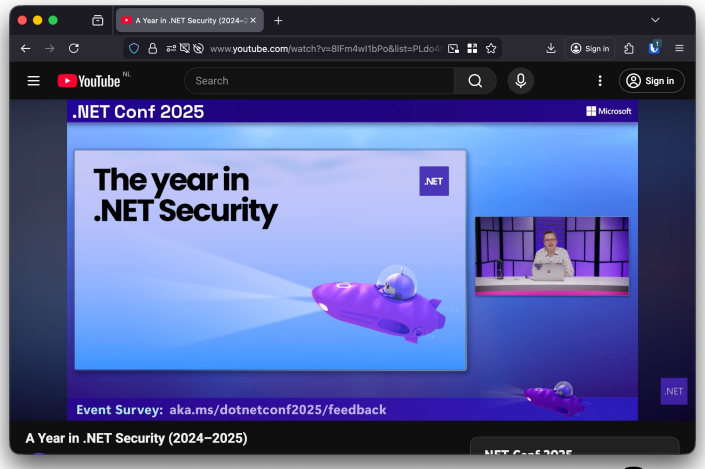


@niels.fennec.dev @nielstanis@infosec.exchange

DotNet CLI - .NET 10



.NET Conf 2025 – The year in .NET Security

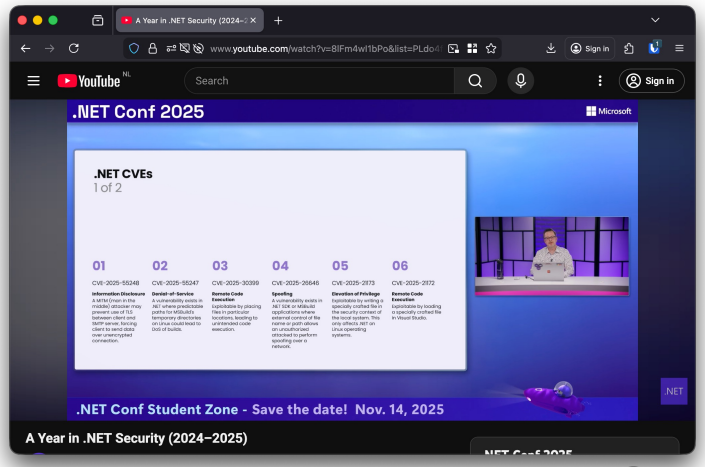


NDC { Manchester }

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

https://youtu.be/8lFm4wI1bPo?si=UJUwARMqY1NF_PO0

.NET Conf 2025 – The year in .NET Security

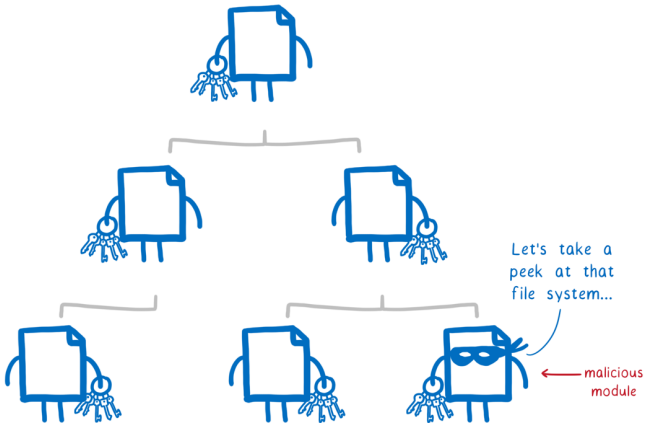


NDC { Manchester }

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

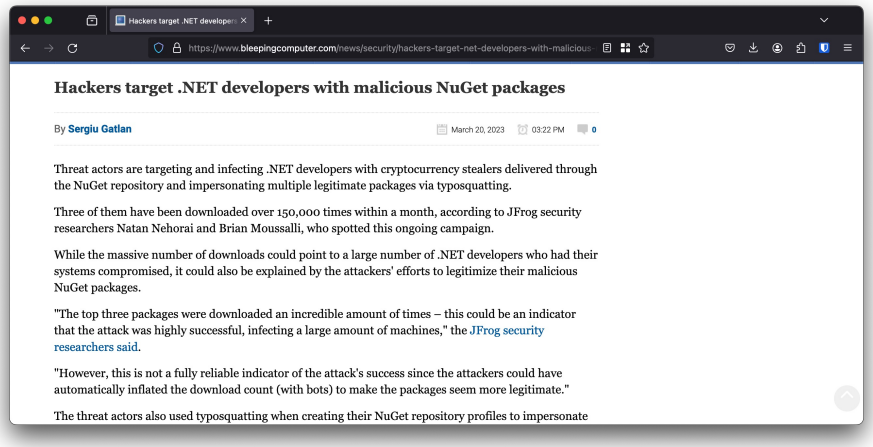
https://youtu.be/8lFm4wI1bPo?si=UJUwARMqY1NF_PO0

Malicious Assembly



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

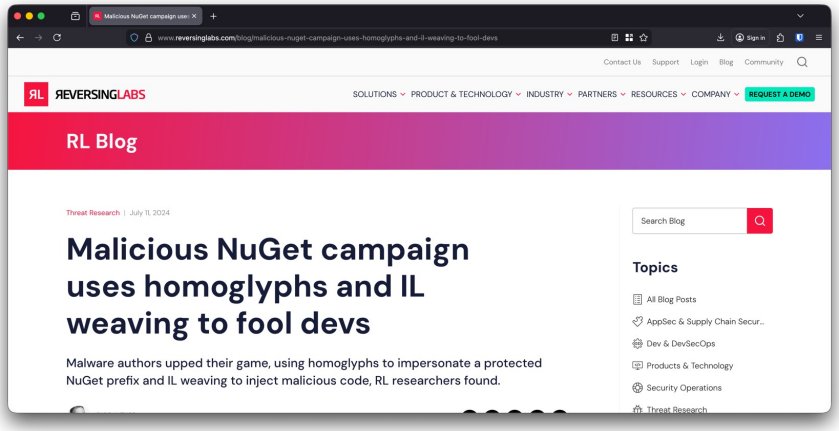
Malicious Package



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

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

Malicious Package



NDC { Manchester }

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

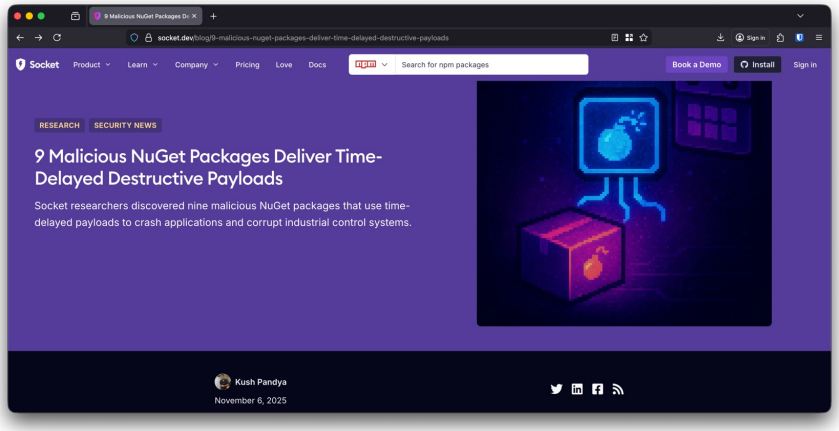
https://www.reversinglabs.com/blog/malicious-nuget-campaign-uses-homoglyphs-and-il-weaving-to-fool-devs

Key Attack Techniques

IL Weaving: Attackers used Intermediary Language weaving to patch legitimate .NET binaries and inject malicious module initializers reversinglabs. This made detection harder compared to earlier PowerShell-based approaches because the malicious code was embedded in compiled DLL files rather than plaintext scripts.

Homoglyphs for Typosquatting: The threat actors exploited homoglyphs—characters with identical visual appearance but different logical representations—to bypass NuGet's reserved prefix security feature reversinglabs. For example, they created a malicious package named "Guna.UI3.Winforms" (using Armenian and Cyrillic characters) to impersonate the legitimate "Guna.UI2.WinForms" package, which has a protected prefix.

Malicious Package



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

https://socket.dev/blog/9-malicious-nuget-packages-deliver-time-delayed-destructive-payloads

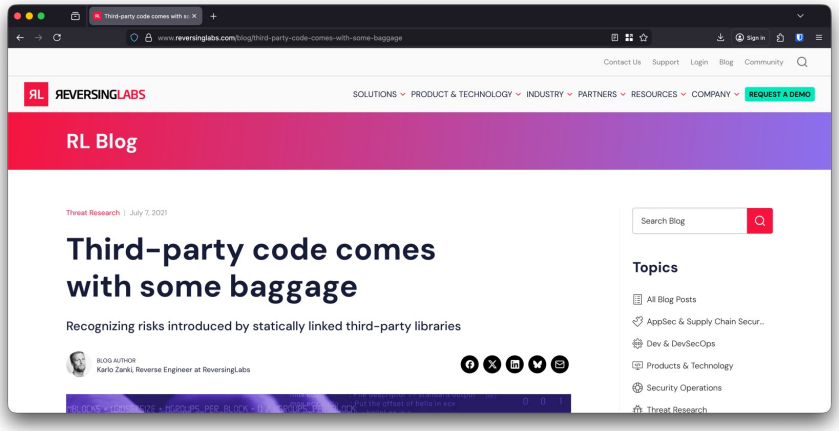
Malicious Package

- Single publisher called **shanhai666**
- 99% code is valid & functional focusing on databases and industrial PLC's, malicious extension methods
- Delayed logic for Aug '27 & Nov '28
- 20% of database queries are terminated
- Sharp7Extend (typo squatting) for PLC, immediate activation mimics hardware failures

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

https://socket.dev/blog/9-malicious-nuget-packages-deliver-time-delayed-destructive-payloads

Do you know what's inside?



NDC { Manchester }

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

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

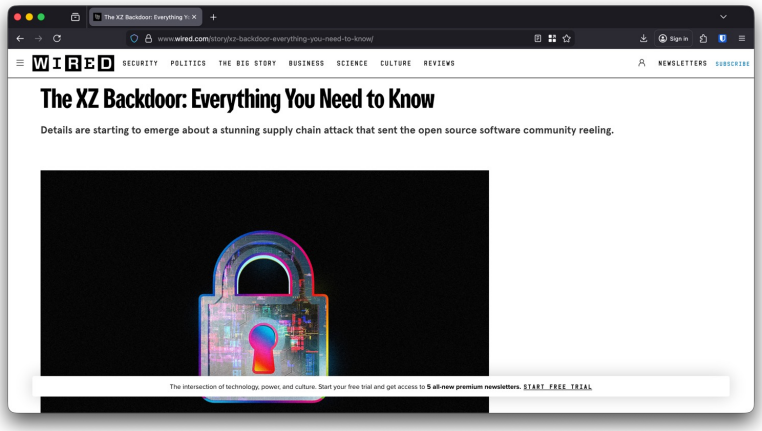
Do you know what's inside?

- Unmanaged components interop
- Precompiled version of 7Zip, WinSCP and PuTTYgen
- Zlib compression library
  - Only source code distributed
  - Used in medical appliances
  - Severe vulnerabilities since 2005!

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

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

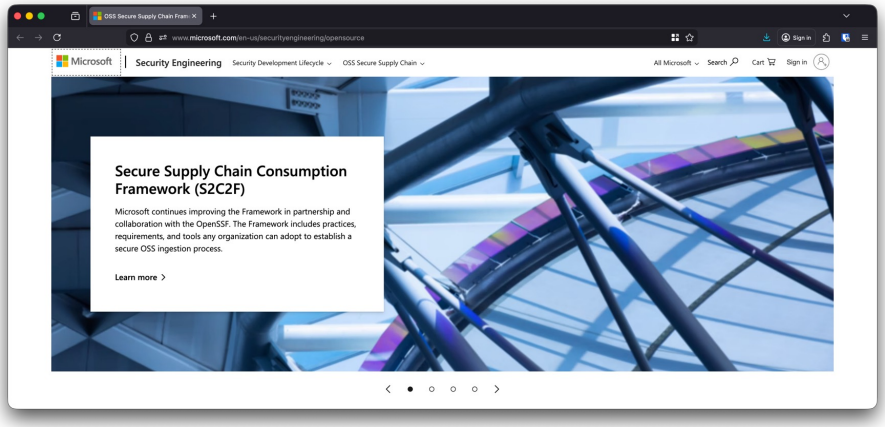
Do you know what's inside? – xz utils



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

https://www.wired.com/story/xz-backdoor-everything-you-need-to-know/

Secure Supply Chain Consumption Framework (S2C2F)



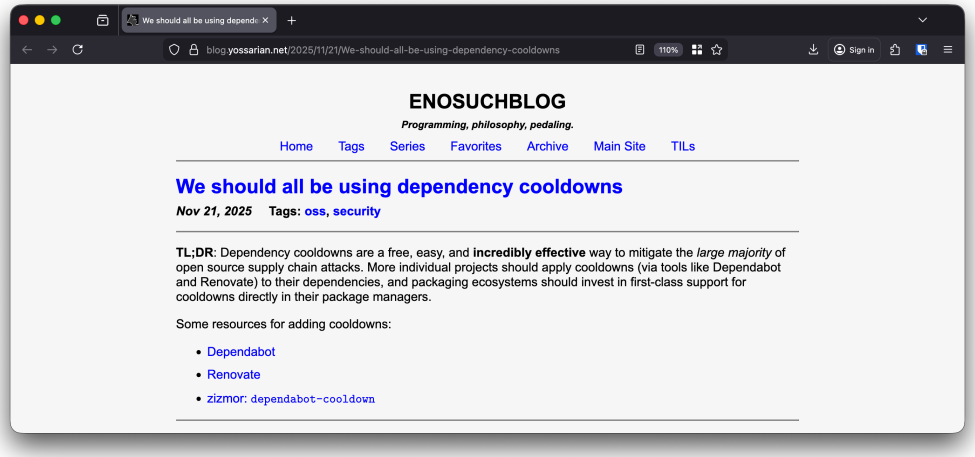
Secure Supply Chain Consumption Framework (S2C2F)

Level 1	Level 2	Level 3	Level 4
Minimum OSS Governance Program	**Secure Consumption and Improved MTTR**	**Malware Defense and Zero-Day Detection**	**Advanced Threat Defense**
• 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]	• 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]	• 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]	• 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]

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

https://github.com/ossf/s2c2f/blob/main/specification/framework.md

Hot take; dependency cooldowns



NDC { Manchester }

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

https://blog.yossarian.net/2025/11/21/We-should-all-be-using-dependency-cooldowns

What's inside? (Audit)

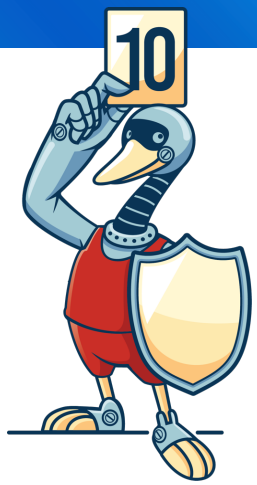
NDC { Manchester }

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

https://scorecard.dev/viewer/?uri=github.com%2FJamesNK%2FNewtonsoft.Json

Open SSF Security Scorecard

- Set Automated Security Checks
- Scoring System 0-10
- Security Best Practices
- Ease of Use
- Community Support



Maintenance

- Vulnerabilities (**High**)
  - Does the project have unfixed vulnerabilities?
- Maintained (**High**)
  - Is the project at least 90 days old, and maintained?
- Security Policy (**Medium**)
  - Does project contain security policy?

Maintenance

- License (Low)
  - Does the project declare a licence?
- CII Best Practices (Low)
  - Does the project have a CII Best Practices Badge?
- Dependency Update Tool (High)
  - Does the project use tools to help update its dependencies e.g. Dependabot, RenovateBot?

Continous Testing

- Continuous Integration Tests (Low)
  - Does the project run tests in CI?
- Fuzzing (Medium)
  - Does the project use fuzzing tools, e.g. OSS-Fuzz?
- Static Analysis (Medium)
  - Does the project use static code analysis tools?

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

Source Risk Assessment

- Binary Artifacts (**High**)
  - Does repository contain binaries/compiled artifacts?
- Branch Protection (**High**)
  - GitHub best practices branch protection
- Dangerous Workflow (**Critical**)
  - GitHub best practices for Workflow Actions

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

Source Risk Assessment

- Code Review (High)
  - Does the project require code review before code is merged?
- Contributors (Low)
  - Does the project have contributors from multiple organizations?

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

Build Risk Assessment

- Pinned Dependencies (Medium)
  - Does the project declare and pin dependencies?
- Token Permission (High)
  - Does the project declare GitHub workflow tokens as read only?

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

Build Risk Assessment

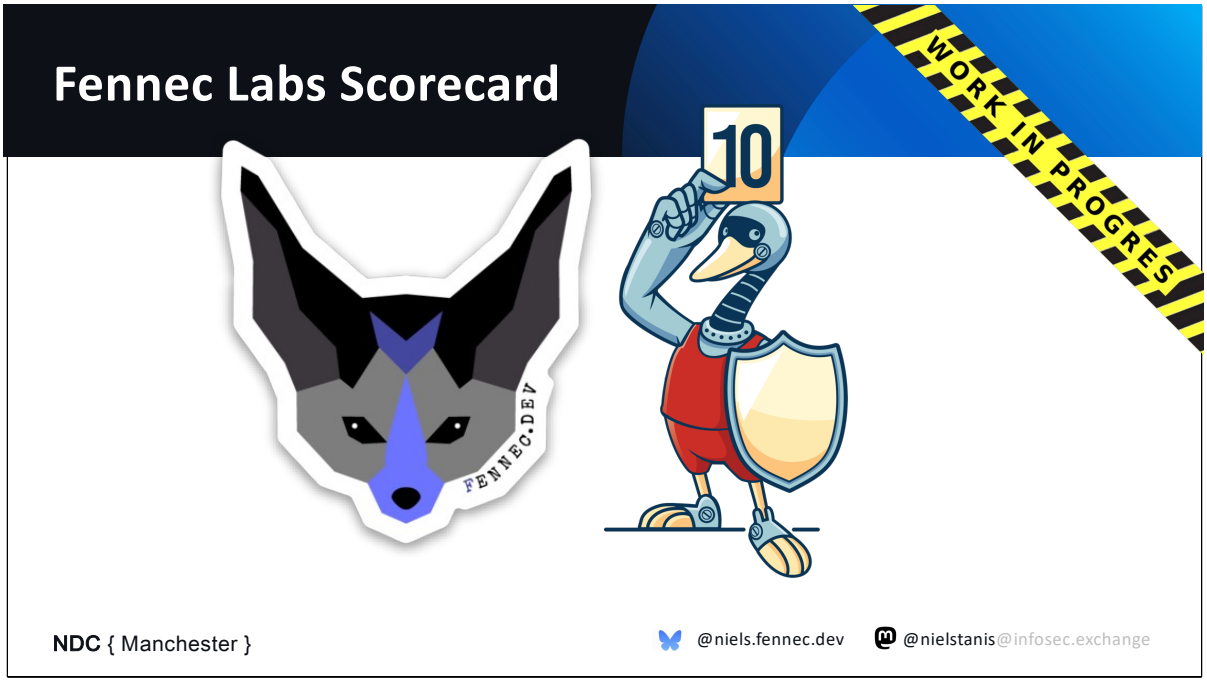
- Packaging (Medium)
  - Does the project build and publish official packages from CI/CD?
- Signed Releases (High)
  - Does the project cryptographically sign releases?

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

Demo OpenSSF Scorecard

```
Running checks
```


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

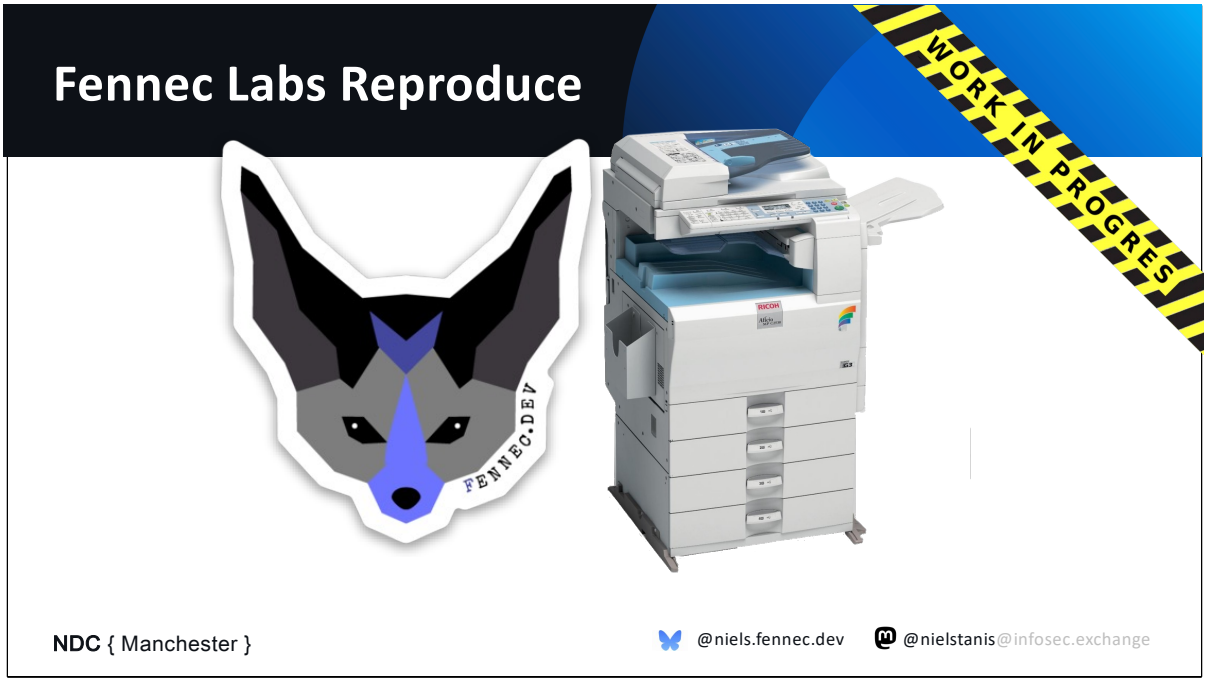


https://scorecard.dev/viewer/?uri=github.com%2FJamesNK%2FNewtonsoft.Json

.NET Reproducibility

- Reproducible builds
  - Independent path from source to binary code.
- Roslyn Compiler Deterministic Compilation
- How reproducible is a simple console app?

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



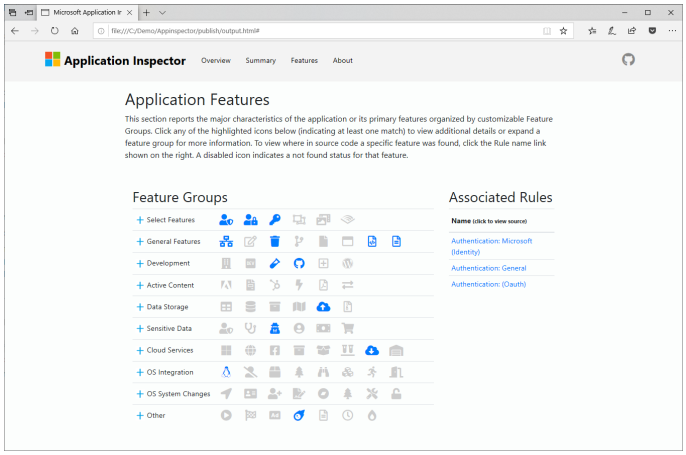
https://scorecard.dev/viewer/?uri=github.com%2FJamesNK%2FNewtonsoft.Json

Fennec Labs Instrument

NDC { Manchester }

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

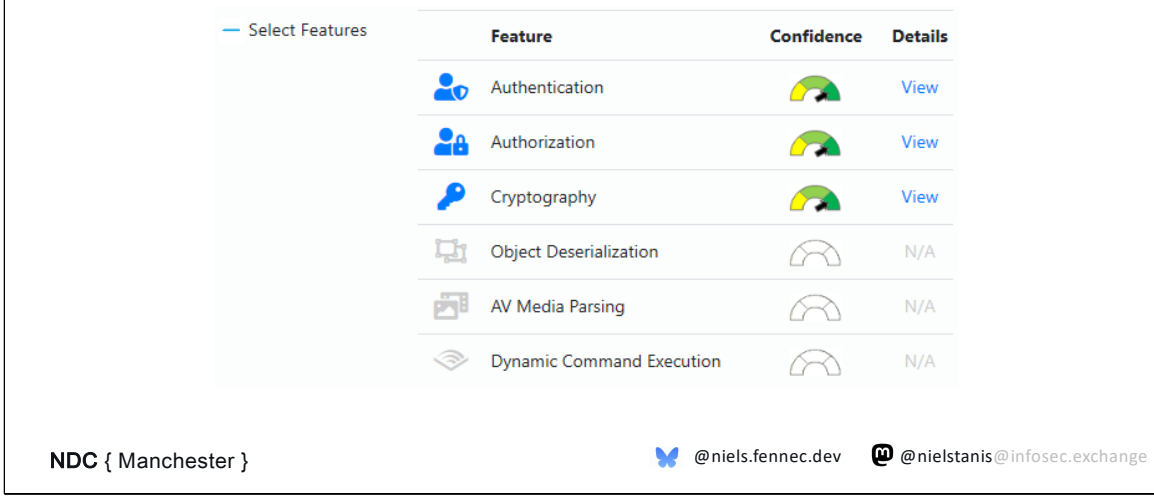
Application Inspector



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

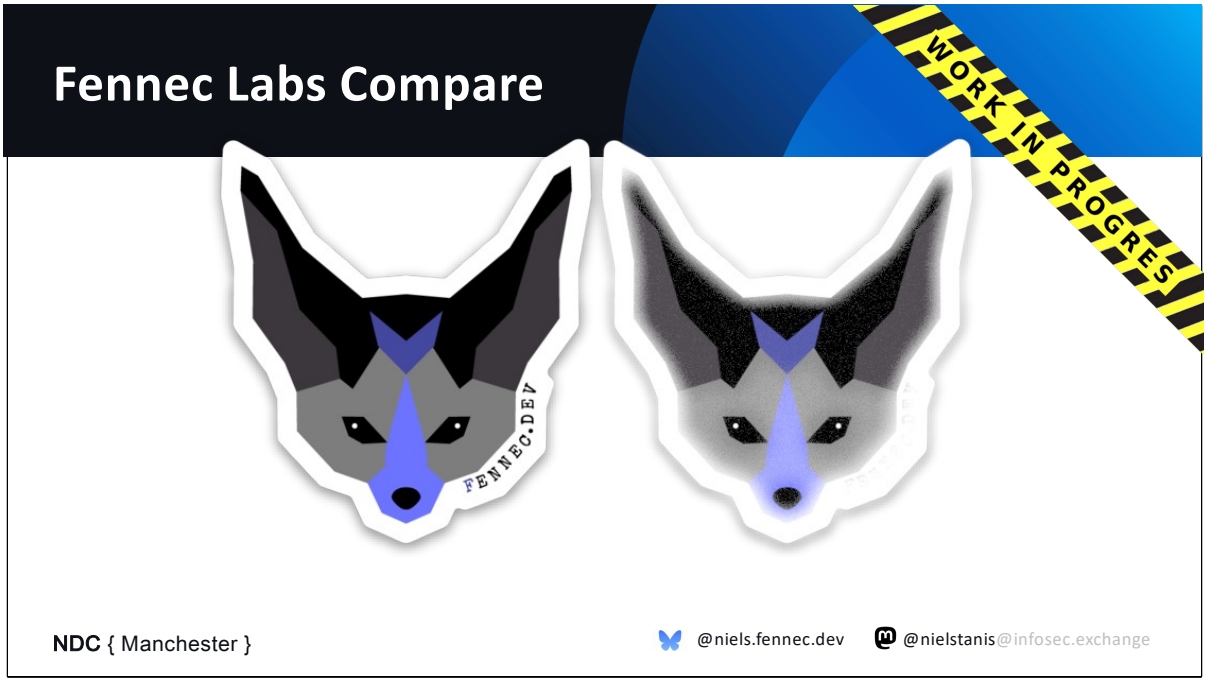
https://github.com/microsoft/ApplicationInspector

Application Inspector



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

https://github.com/microsoft/ApplicationInspector



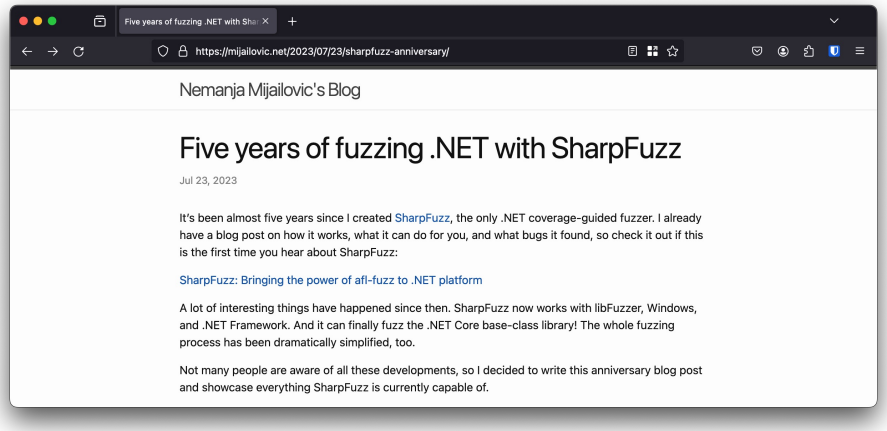
https://scorecard.dev/viewer/?uri=github.com%2FJamesNK%2FNewtonsoft.Json

Future Idea - Fuzzing

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

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

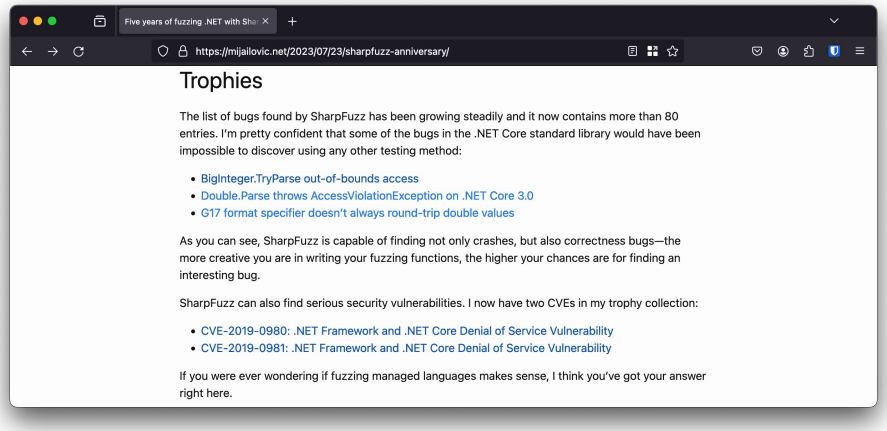
Fuzzing .NET & SharpFuzz



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

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

Fuzzing .NET & SharpFuzz



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

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

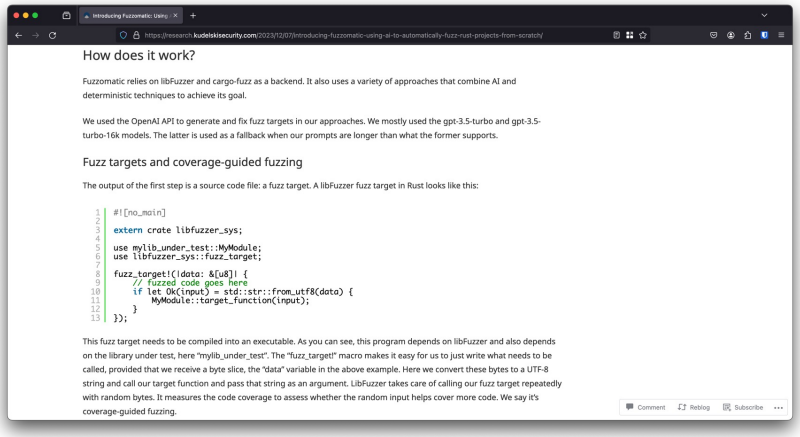
Fuzzing .NET – Jil JSON Serializer

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


@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.

Comment Reblog Subscribe

NDC { Manchester }

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

https://research.kudelskisecurity.com/2023/12/07/introducing-fuzzomatic-using-ai-to-automatically-fuzz-rust-projects-from-scratch/

Future Idea – Dashboard & Feedback

- Local dashboard for project
- Remote dashboard for community
- Ability to do review and share security related data of project
- MCP server with embedded prompt resources to help security review
- Roslyn analyzer that can help guide the developer

Conclusion

- Doing software security is hard and Software Security Professionals don't scale!
- Package eco-systems have been wild west lately
- Goal is to have Fennec Labs help with reviewing and sharing security details for all audiences
- Allowing everyone to make better informed decisions and can act if needed.

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

Thank you!

- ntanis at Veracode.com
- https://github.com/nielstanis/ndcmanchester2025
- Questions?

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