

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

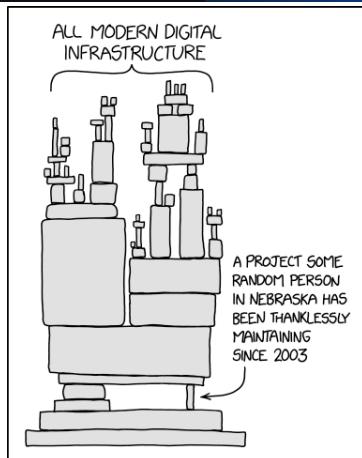


NDC { Manchester }

 @niels.fennec.dev

 @nielstanis@infosec.exchange

Modern Application Architecture XKCD 2347



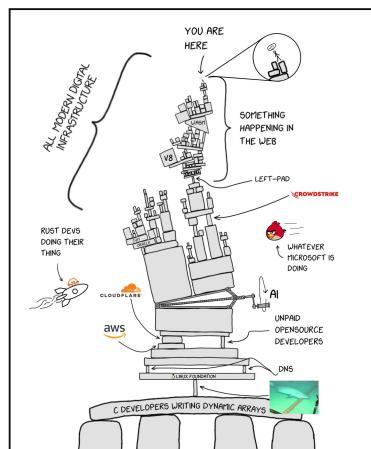
NDC { Manchester }

@niels.fennec.dev

@nielstanis@infosec.exchange

<https://xkcd.com/2347/>

You are here



NDC { Manchester }

@niels.fennec.dev

@nielstanis@infosec.exchange

<https://xkcd.com/2347/>

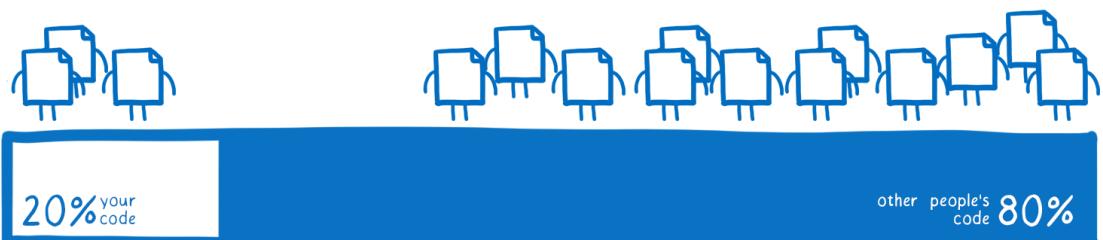
Agenda

- 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
 - Feedback & sharing
 - Future and idea's
- Conclusion and Q&A

NDC { Manchester }

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

Average codebase composition



NDC { Manchester }

 @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**



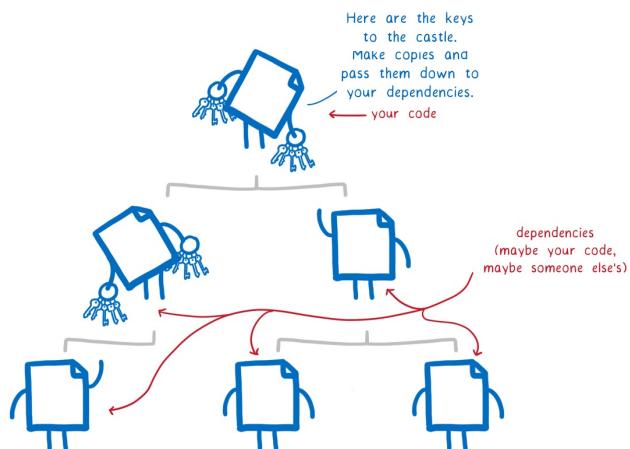
NDC { Manchester }

@niels.fennec.dev

@nielstanis@infosec.exchange

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

Average codebase composition



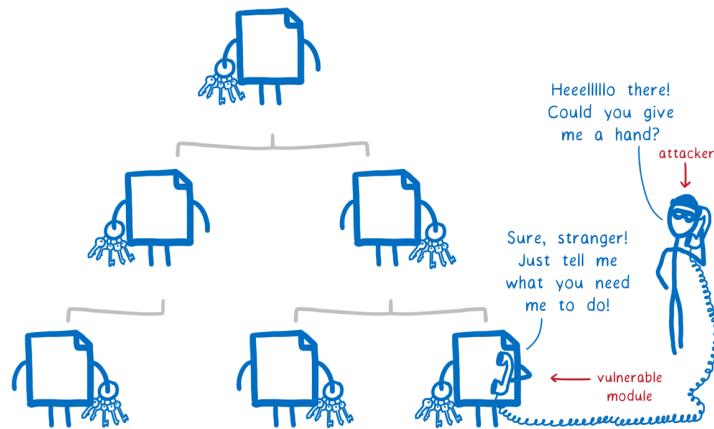
NDC { Manchester }

@niels.fennec.dev

@nielstanis@infosec.exchange

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

Vulnerable Assembly



NDC { Manchester }

 @niels.fennec.dev

 @nielstanis@infosec.exchange

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

DotNet CLI

```
nelson at ghost-m2 in ~/research/consoleapp
└─o 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
└─o 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.
└─o █
```

NDC { Manchester }

 @niels.fennec.dev

 @nielstanis@infosec.exchange

DotNet CLI

```
nelson at ghost-m2 in ~/research/consoleapp
└─o 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.common           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
```

NDC { Manchester }

 @niels.fennec.dev

 @nielstanis@infosec.exchange

DotNet CLI

```
nelson at ghost-m2 in ~/research/consoleapp
└─$ dotnet list package --include-transitive --vulnerable

The following sources were used:
  https://api.nuget.org/v3/index.json

Project `consoleapp` has the following vulnerable packages
[net9.0]:
  Transitive Package      Resolved   Severity   Advisory URL
  > Newtonsoft.Json        9.0.1      High       https://github.com/advisories/GHSA-5crp-9r3c-p9vr

nelson at ghost-m2 in ~/research/consoleapp
└─$
```

NDC { Manchester }

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

DotNet CLI - .NET 10

```
nelson at ghost-m2 in ~/research/consoleapp
└─o dotnet --version
  10.0.100
'nelson at ghost-m2 in ~/research/consoleapp
└─o dotnet build
Restore succeeded with 1 warning(s) in 0.3s
  /Users/nelson/Research/consoleapp/consoleapp.csproj : warning NU1903: Package 'Newtonsoft.Json' 9.0.1 has a known high
severity vulnerability, https://github.com/advisories/GHSA-5crp-9r3c-p9vr
  consoleapp net10.0 succeeded with 1 warning(s) (1.5s) → bin/Debug/net10.0/consoleapp.dll
  /Users/nelson/Research/consoleapp/consoleapp.csproj : warning NU1903: Package 'Newtonsoft.Json' 9.0.1 has a known high
severity vulnerability, https://github.com/advisories/GHSA-5crp-9r3c-p9vr

Build succeeded with 2 warning(s) in 2.0s
'nelson at ghost-m2 in ~/research/consoleapp
└─o █
```

NDC { Manchester }

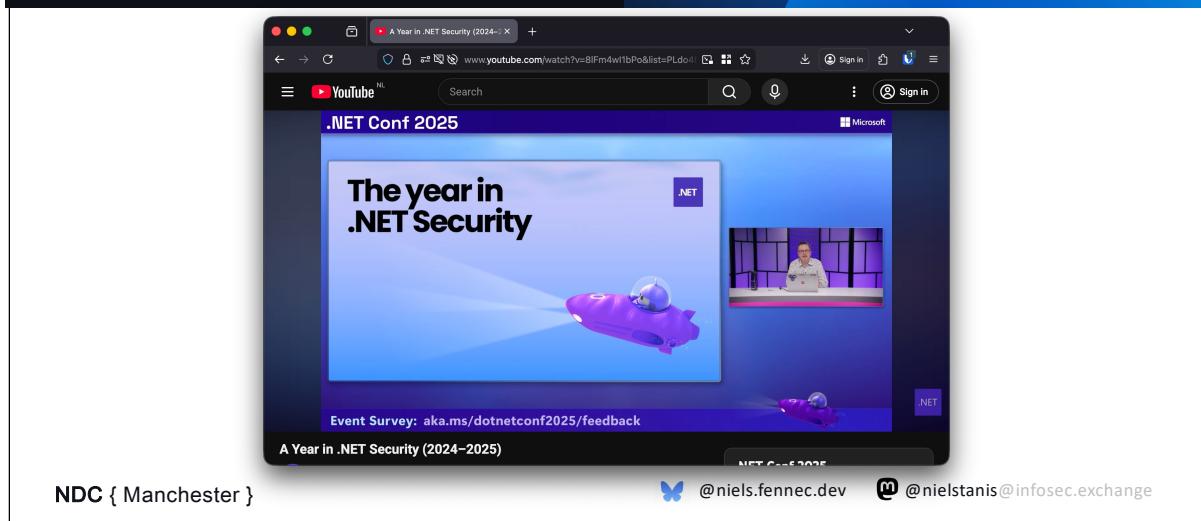


@niels.fennec.dev



@nielstanis@infosec.exchange

.NET Conf 2025 – The year in .NET Security



NDC { Manchester }



@niels.fennec.dev



@nielstanis@infosec.exchange

https://youtu.be/8lFm4wl1bPo?si=UJUwARMqY1NF_PO0

.NET Conf 2025 – The year in .NET Security

The screenshot shows a YouTube video player with the title ".NET Conf 2025". The video content displays a slide titled ".NET CVEs" with the subtitle "1 of 2". The slide lists six CVEs numbered 01 to 06, each with a brief description:

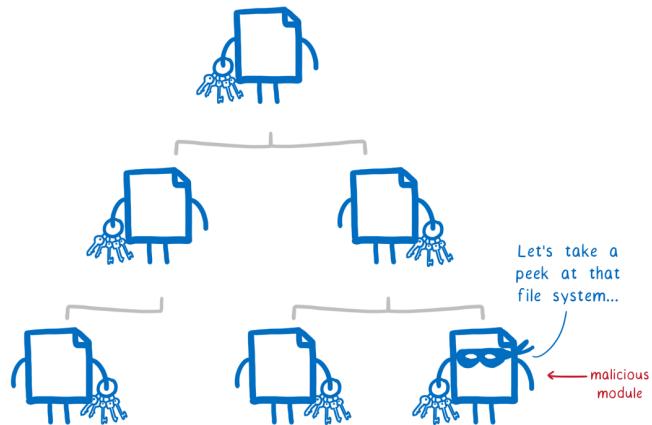
- 01** CV-2025-8548 **Information Disclosure**: A .NET API in the `System.Diagnostics` assembly can be used to read memory from other .NET processes.
- 02** CV-2025-8547 **Denial-of-Service**: A vulnerability exists in the .NET runtime when parsing assembly files. This can lead to denial-of-service conditions.
- 03** CV-2025-3099 **Remote Code Execution**: A vulnerability exists in the .NET runtime when parsing assembly files. This can lead to remote code execution.
- 04** CV-2025-2648 **Spawning**: A vulnerability exists in the .NET runtime when spawning processes. This can lead to uncontrolled process creation and spawning over a mutex.
- 05** CV-2025-2913 **Execution of Privilege Escalation**: Exploitation by writing to memory controlled by another user can result in privilege escalation.
- 06** CV-2025-2912 **Remote Code Execution**: A vulnerability exists in the .NET runtime when reading memory from other processes. This can lead to remote code execution.

Below the slide, there is a call-to-action: ".NET Conf Student Zone - Save the date! Nov. 14, 2025". At the bottom of the slide, it says "A Year in .NET Security (2024–2025)".

At the bottom of the screen, there are two social media handles: [@niels.fennec.dev](#) and [@nielstanis@infosec.exchange](#).

https://youtu.be/8lFm4wl1bPo?si=UJUwARMqY1NF_PO0

Malicious Assembly



NDC { Manchester }

@niels.fennec.dev

@nielstanis@infosec.exchange

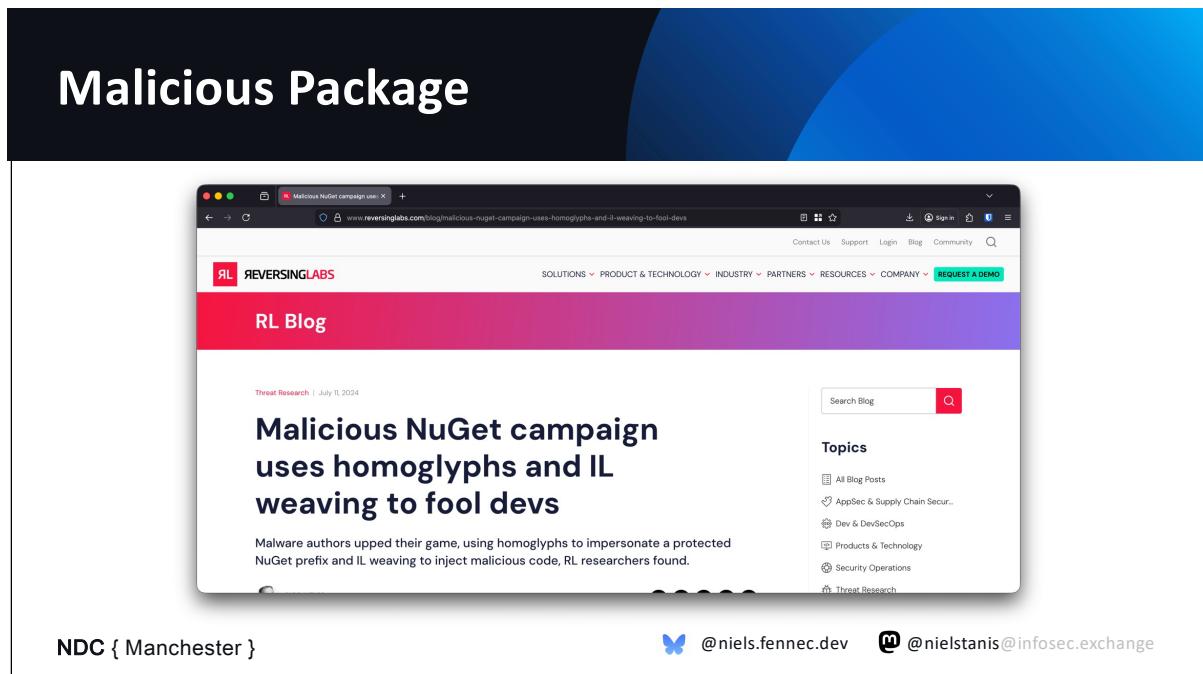
<https://hacks.mozilla.org/2019/11/announcing-the-bytecode-alliance/>

Malicious Package

The screenshot shows a web browser window with a dark blue header and a white content area. The title bar says "Hackers target .NET developers x +". The address bar shows the URL "https://www.bleepingcomputer.com/news/security/hackers-target-net-developers-with-malicious-nuget-packages//". The main content is an article titled "Hackers target .NET developers with malicious NuGet packages" by Sergiu Gatlan, published on March 20, 2023, at 03:22 PM. The article discusses threat actors targeting .NET developers with cryptocurrency stealers delivered through the NuGet repository and impersonating multiple legitimate packages via typosquatting. It mentions three packages downloaded over 150,000 times. Researchers Natan Nehorai and Brian Moussalli spotted the campaign. The text quotes JFrog security researchers and notes that while the massive number of downloads could point to a large number of .NET developers, it could also be explained by attackers' efforts to legitimize their malicious NuGet packages. The article concludes with a note about typosquatting used in NuGet repository profiles.

NDC { Manchester }  @niels.fennec.dev  @nielstanis@infosec.exchange

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



<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

The screenshot shows a blog post from socket.dev. The title is "9 Malicious NuGet Packages Deliver Time-Delayed Destructive Payloads". The post discusses nine malicious NuGet packages that use time-delayed payloads to crash applications and corrupt industrial control systems. The author is Kush Pandya, and the date is November 6, 2025. The post includes a sidebar for research and security news, and a search bar for npm packages. There are social media sharing icons at the bottom.

NDC { Manchester } @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

NDC { Manchester }

 @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?

The screenshot shows a blog post on the ReversingLabs website. The title of the post is "Third-party code comes with some baggage". The post discusses recognizing risks introduced by statically linked third-party libraries. The author is Karlo Zarki, a Reverse Engineer at ReversingLabs. The post has 0 comments and 0 likes. On the right side of the page, there is a sidebar with a search bar and a "Topics" section listing categories like All Blog Posts, AppSec & Supply Chain Secur..., Dev & DevSecOps, Products & Technology, Security Operations, and Threat Research. At the bottom of the page, there are social media sharing icons and two Twitter handles: @niels.fennec.dev and @nielstanis@infosec.exchange.

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

Do you know what's inside?

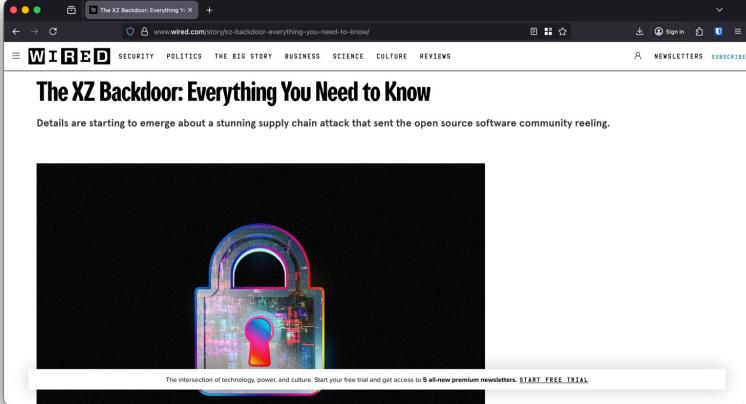
- Unmanaged components interop
- Precompiled version of 7Zip, WinSCP and PuTTYgen
- Zlib compression library
 - Only source code distributed
 - Used in medical appliances
- Severe vulnerabilities since 2005!

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? – xz utils

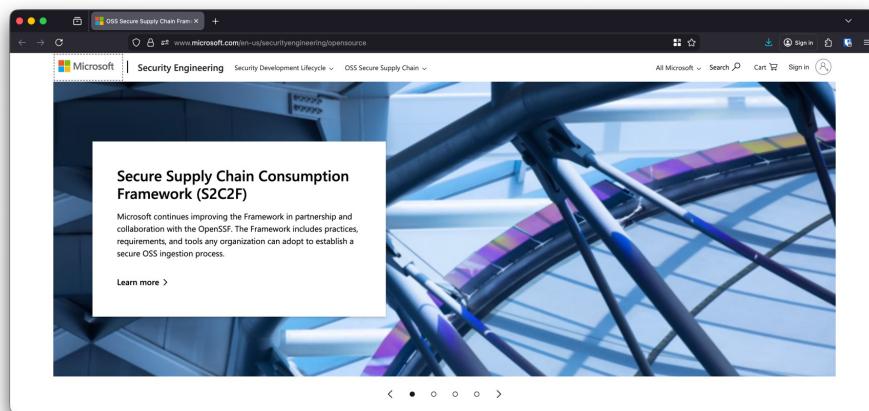


The screenshot shows a browser window with the URL www.wired.com/story/xz-backdoor-everything-you-need-to-know/. The page is titled "The XZ Backdoor: Everything You Need to Know". Below the title, it says "Details are starting to emerge about a stunning supply chain attack that sent the open source software community reeling." A large image of a glowing padlock is centered on the page. At the bottom, there is a footer bar with the text "The intersection of technology, power, and culture. Start your free trial and get access to 5 all-new premium newsletters. [START FREE TRIAL](#)".

NDC { Manchester }  @niels.fennec.dev  @nielstanis@infosec.exchange

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

Secure Supply Chain Consumption Framework (S2C2F)



NDC { Manchester }

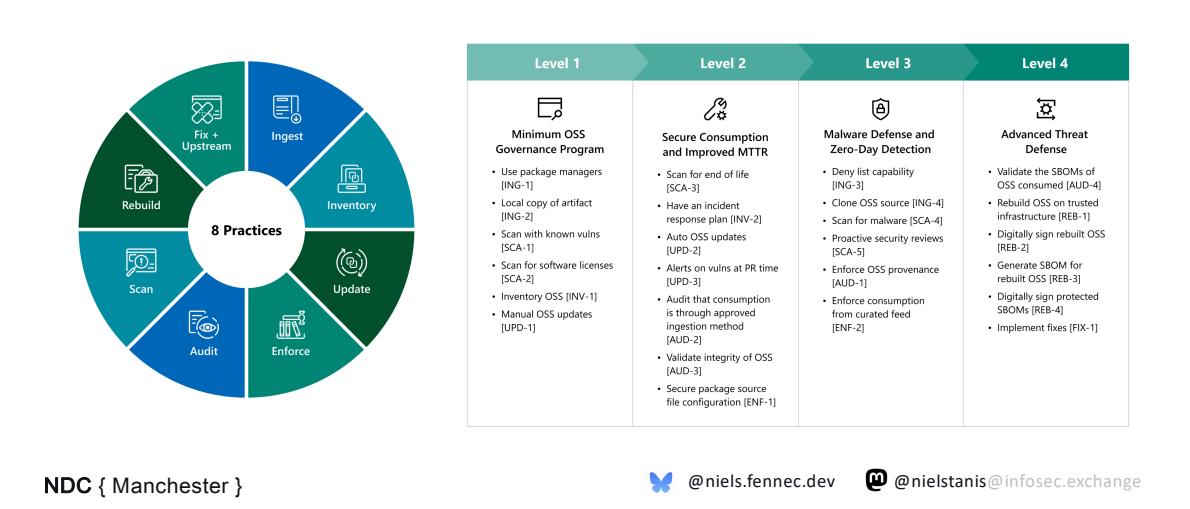


@niels.fennec.dev

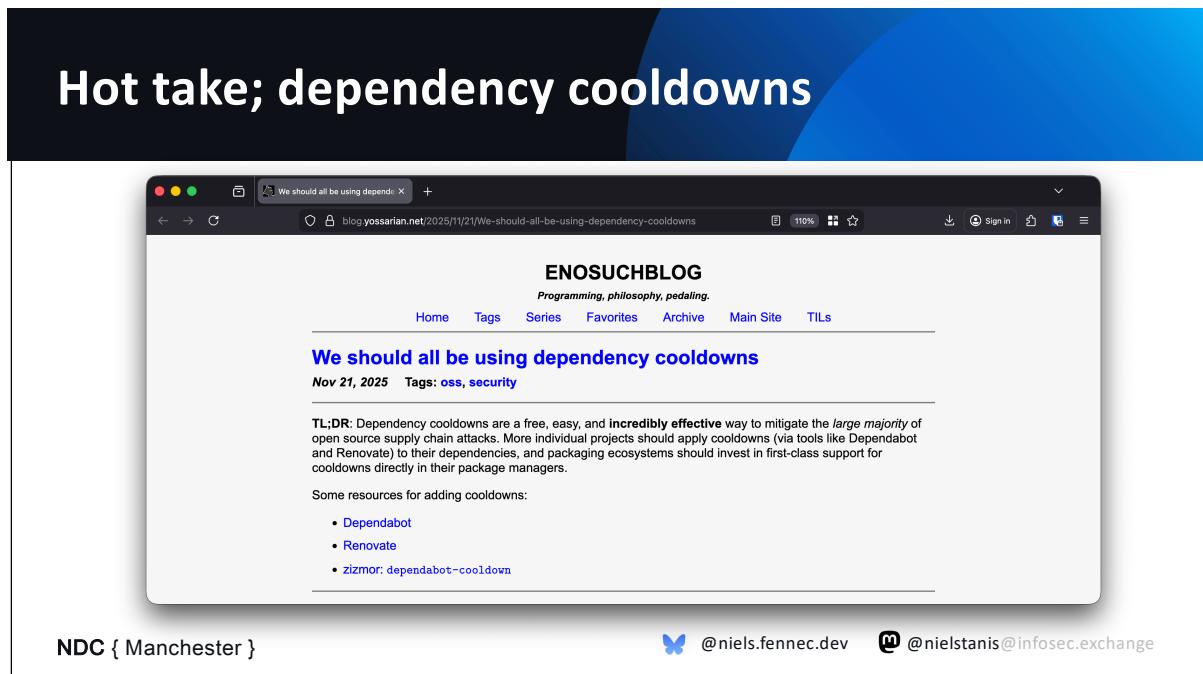


@nielstanis@infosec.exchange

Secure Supply Chain Consumption Framework (S2C2F)



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



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

What's inside? (Audit)

The image shows two product labels side-by-side. On the left is a can of Club-Mate Carbonated Yerba-Mate Soft Drink. The label features a logo of a man in a hat, the brand name 'CLUB-MATE' in large letters, and the product description 'CARBONATED YERBA-MATE SOFT DRINK'. It also states 'NO ARTIFICIAL ADDITIVES.', 'VEGAN.', and 'SERVE CHILLED.' Below this is the website 'www.clubmateusa.com'. At the bottom, it includes import information: 'IMPORTED BY CM USA, LLC NEW YORK, NY 10001', 'PRODUCT OF GERMANY', 'SUGAR FREE', '500 ml (16.9 fl. oz.)', and a refund policy: 'REFUND EX-CA REDEMPTION VALUE 500 ml (16.9 fl. oz.)'.

Nutrition Facts

Serving Size	8 fl. oz. (240 mL)	Servings Per Container	2
Amount Per Serving		% Daily Value*	
Calories	80	0%	
Total Fat	0g	0%	
Sodium	75mg	3%	
Total Carbohydrate	16g	5%	
Sugars	16g		
Protein	0g		
Calcium	2%	Iron	0%

*Percent daily values are based on a 2,000 calorie diet.

48mg caffeine per 8 fl. oz. serving.

Contains: carbonated water, glucose-fructose syrup, sugar, yerba mate extract (0.4%), citric acid, caffeine, natural flavors, caramel color.

The right image shows a hand holding a glass bottle of Mate-Mate. The label features the brand name 'MATE MATE' in large letters, with the tagline 'FÜR WACHE TAGE & LANGE NÄCHTE, SO FRISCH & MUNTER WIE DU'. Below this, it says 'ERHÖHTER KOFFEINGEHALT! FÜR KINDER UND SCHWANGERE ODER STILLENDE FRAUEN NICHT EMPFOHLEN (30 mgC / 100 mL)'. The label also indicates it is 'VEGAN - GLUTENFREI / MIT AGAVENDICKSAFT'. At the bottom, it shows the barcode '4 260310 553382', the expiration date '19.10.25 AB', and the manufacturer information 'Thomas Henry GmbH, Bessemerstraße 22, 22103 Hamburg'.

NDC { Manchester } @niels.fennec.dev @nielstanis@infosec.exchange

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

OpenSSF Security Scorecard

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



NDC { Manchester }

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

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?

NDC { Manchester }



@niels.fennec.dev



@nielstanis@infosec.exchange

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?

NDC { Manchester }

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

Continuous 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?

NDC { Manchester }



@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

NDC { Manchester }

 @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?

NDC { Manchester }

 @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?

NDC { Manchester }

 @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?

NDC { Manchester }

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

Demo OpenSSF Scorecard

Running checks

NDC { Manchester }

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

Fennec Labs Scorecard



NDC { Manchester }

 @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?

NDC { Manchester }



@niels.fennec.dev



@nielstanis@infosec.exchange

Fennec Labs Reproduce



NDC { Manchester }

 @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

The screenshot shows the Microsoft Application Inspector interface. At the top, there's a navigation bar with tabs: Overview, Summary, Features, and About. Below the navigation bar is a section titled "Application Features". A descriptive text explains that this section reports major characteristics of the application or its primary features organized by customizable Feature Groups. It also mentions that clicking highlighted icons will show additional details or expand a feature group for more information. A note states that if a specific feature was found in source code, a rule name link is shown on the right, and a disabled icon indicates a not found status.

Feature Groups

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

Associated Rules

- Name (click to view search)
- Authentication: Microsoft (Identity)
- Authentication: General
- Authentication: (OAuth)

NDC { Manchester }

@niels.fennec.dev

@nielstanis@infosec.exchange

<https://github.com/microsoft/ApplicationInspector>

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

NDC { Manchester }

 @niels.fennec.dev

 @nielstanis@infosec.exchange

<https://github.com/microsoft/ApplicationInspector>

Fennec Labs Compare



NDC { Manchester }

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

<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?



NDC { Manchester }

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

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

Fuzzing .NET & SharpFuzz

The screenshot shows a web browser window with a dark blue header bar. The title bar reads "Five years of fuzzing .NET with SharpFuzz". The main content area is white and displays a blog post by "Nemanja Mijailovic's Blog". The title of the post is "Five years of fuzzing .NET with SharpFuzz" and it was published on "Jul 23, 2023". The post content discusses the history and evolution of SharpFuzz, mentioning its creation five years ago, its coverage-guided nature, and its ability to fuzz .NET Core base-class library. It also notes improvements like working with libFuzzer, Windows, and .NET Framework. The author expresses appreciation for the developments and decides to write an anniversary blog post. At the bottom of the post, there is a small note about the current capabilities of SharpFuzz. Below the post, there are social media icons and handles for NDC Manchester, @niels.fennec.dev, and @nielstanis@infosec.exchange.

NDC { Manchester }

Five years of fuzzing .NET with SharpFuzz

Jul 23, 2023

It's been almost five years since I created [SharpFuzz](#), the only .NET coverage-guided fuzzer. I already have a blog post on how it works, what it can do for you, and what bugs it found, so check it out if this is the first time you hear about SharpFuzz:

[SharpFuzz: Bringing the power of afl-fuzz to .NET platform](#)

A lot of interesting things have happened since then. SharpFuzz now works with libFuzzer, Windows, and .NET Framework. And it can finally fuzz the .NET Core base-class library! The whole fuzzing process has been dramatically simplified, too.

Not many people are aware of all these developments, so I decided to write this anniversary blog post and showcase everything SharpFuzz is currently capable of.

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

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

Fuzzing .NET & SharpFuzz

The screenshot shows a web browser window with a dark blue header bar. The title bar reads "Five years of fuzzing .NET with SharpFuzz". The main content area has a white background and features a section titled "Trophies". Below the title, there is a paragraph of text followed by a bulleted list of bugs found by SharpFuzz. Another paragraph follows, and at the bottom, there is a note about security vulnerabilities. At the very bottom of the page, there are social media links for NDC Manchester.

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.

NDC { Manchester } <https://mijailovic.net/2023/07/23/sharpfuzz-anniversary/> [@niels.fennec.dev](https://twitter.com/nielsfennec) [@niestanis@infosec.exchange](https://twitter.com/niestanis)

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) { }
    });
}
```

NDC { Manchester }

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

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

Fuzzomatic: Using AI to Fuzz Rust

The screenshot shows a web browser window with the URL <https://research.kudelskisecurity.com/2023/12/07/introducing-fuzzomatic-using-ai-to-automatically-fuzz-rust-projects-from-scratch/>. The page title is "Introducing Fuzzomatic: Using AI to Fuzz Rust". The main content discusses how Fuzzomatic works, mentioning libFuzzer and cargo-fuzz as backends, and how it uses AI and deterministic techniques. It also explains the OpenAI API used for generating and fixing fuzz targets. A code snippet in Rust is shown:

```
1  #[no_main]
2
3  extern crate libfuzzer_sys;
4
5  use mylib::mylib;
6  use libfuzzer_sys::fuzz_target;
7
8  fuzz_target!(data: &[u8]) {
9      // Fuzzed code goes here
10     if let Ok(input) = str::from_utf8(data) {
11         mylib::target_function(input);
12     }
13 }
```

A note below the code explains that it needs to be compiled into an executable and describes the use of the `fuzz_target!` macro. At the bottom right of the browser window, there are social media sharing options for Comment, Reblog, and 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



NDC { Manchester }

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

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.

NDC { Manchester }

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

Thank you!



- ntanis at Veracode.com
- <https://github.com/niestanis/ndcmanchester2025>
- Questions?

NDC { Manchester }

 @niels.fennec.dev  @niestanis@infosec.exchange