Securing your .NET application software supply-chain

Niels Tanis

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
- Microsoft MVP – Developer Technologies

@nielstanis@infosec.exchange

Securing your .NET application software supply-chain

CODE EUROPE

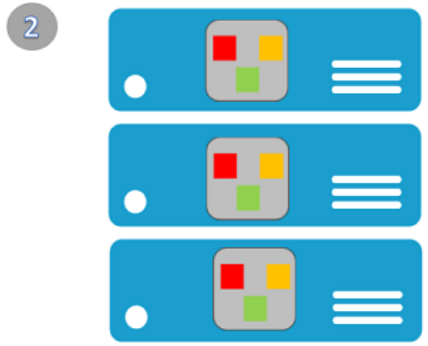
@nielstanis@infosec.exchange

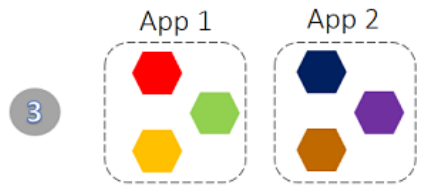
Agenda

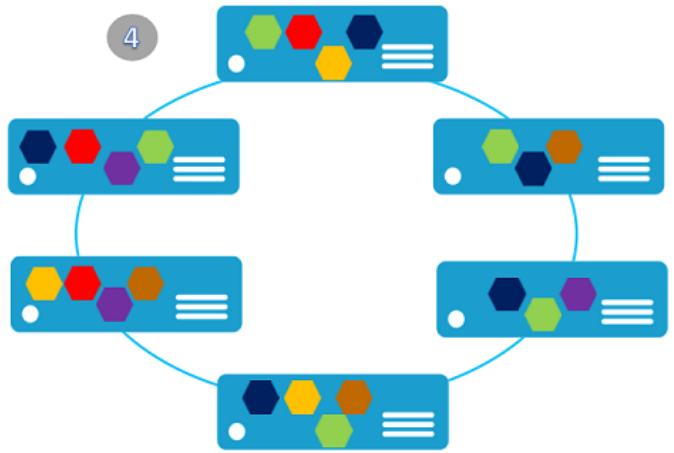
- Definition Software Supply-Chain
- Securing the Software Supply-Chain
  - Developer & Source
  - 3rd Party Libraries
  - Build & Release
- Conclusion and Q&A

Evolution in Software Architecture

- Monolith
- Microservices
- Serverless
  Cloud-Native





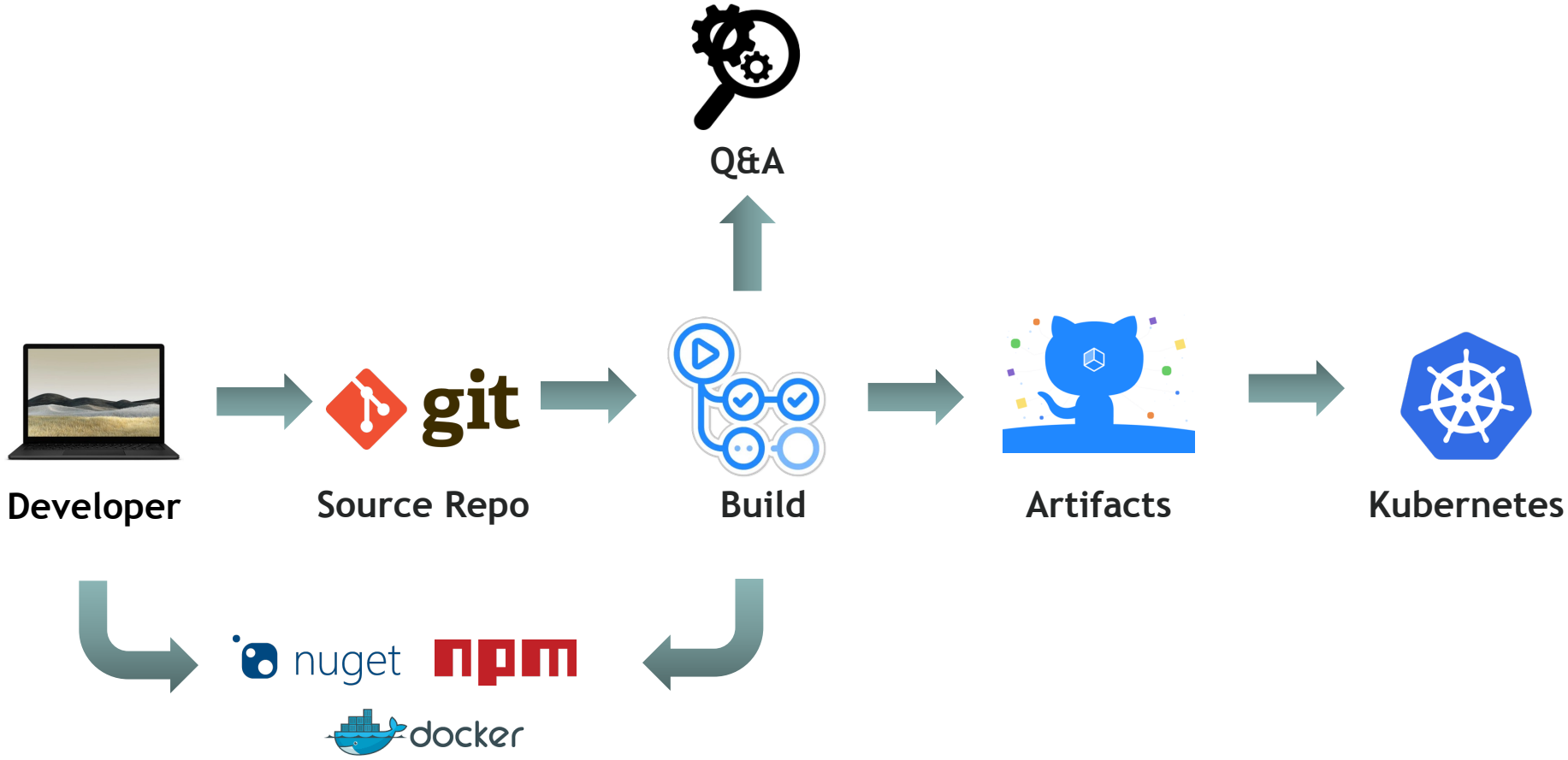


@nielstanis@infosec.exchange

What is a Supply Chain?

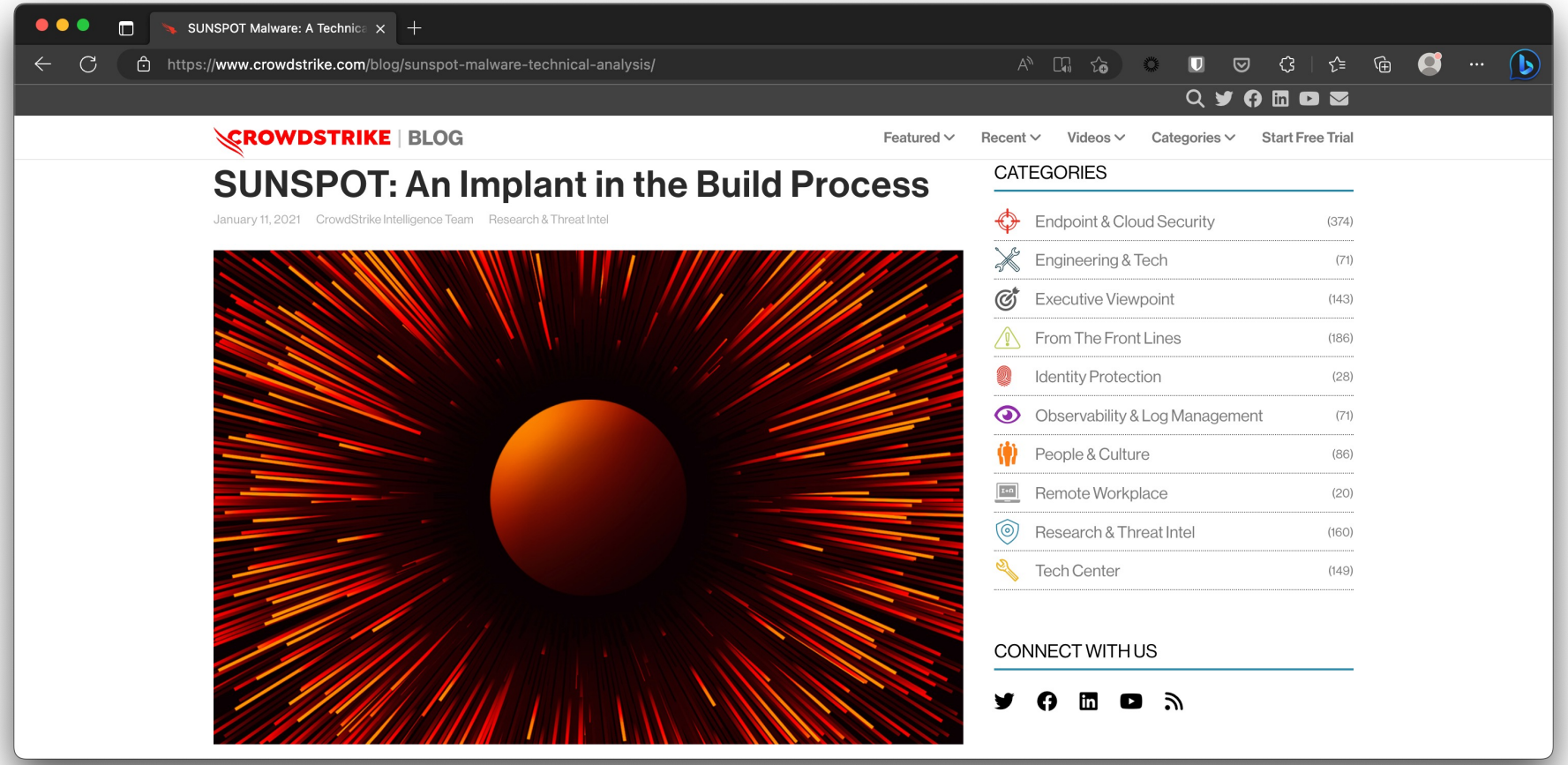
@nielstanis@infosec.exchange

Software Supply Chain



@nielstanis@infosec.exchange

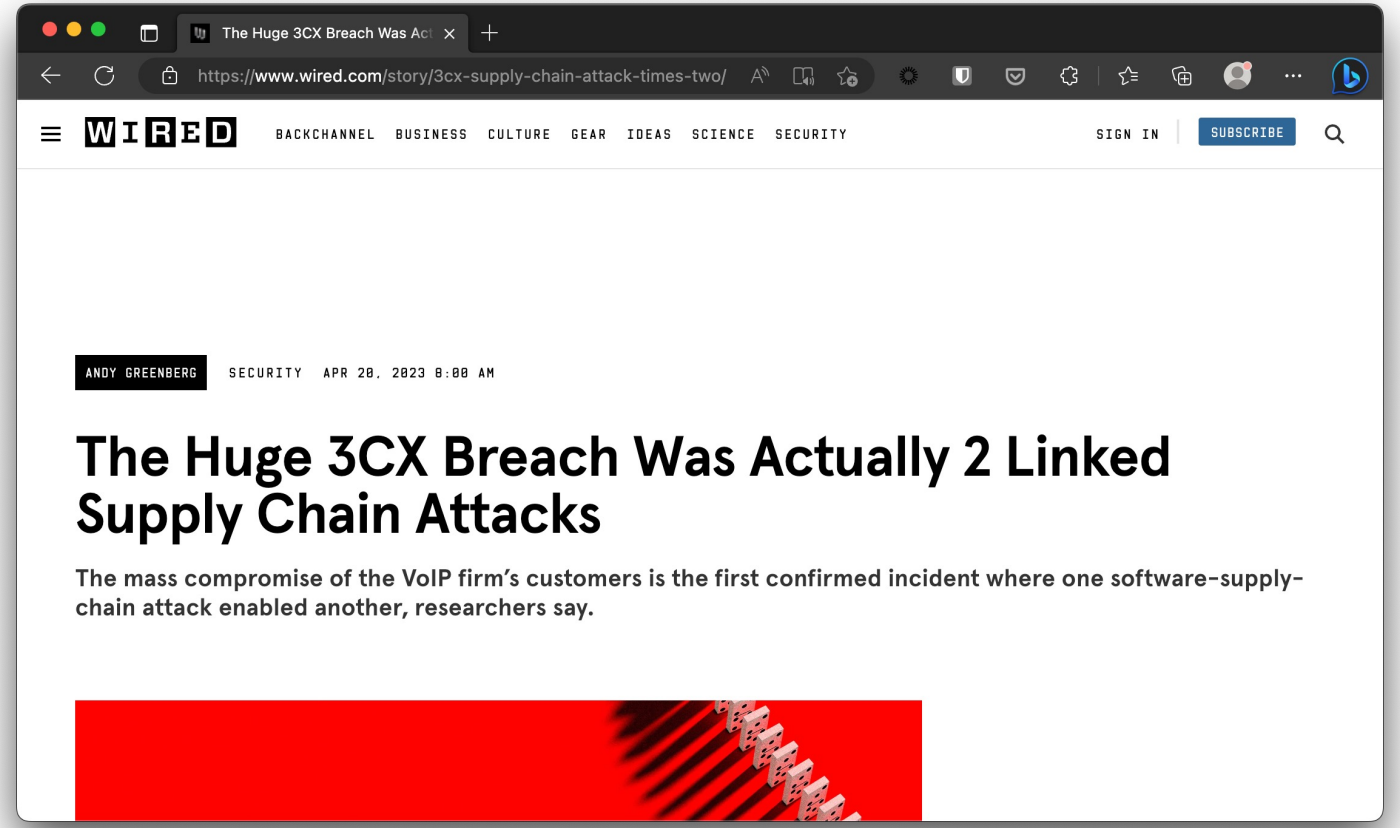
SolarWinds SunSpot



CODE EUROPE

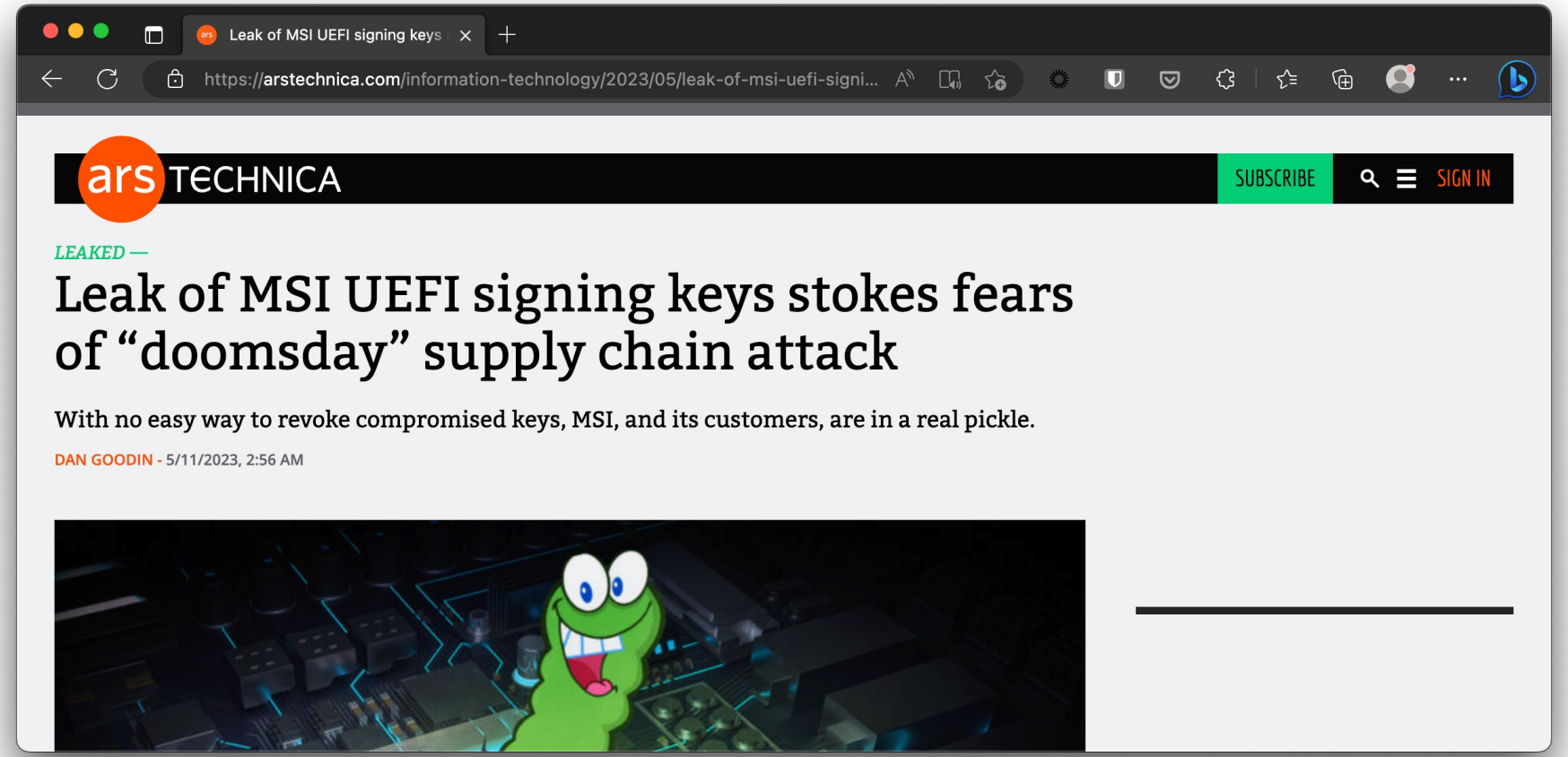
@nielstanis@infosec.exchange

3CX Supply Chain Attack

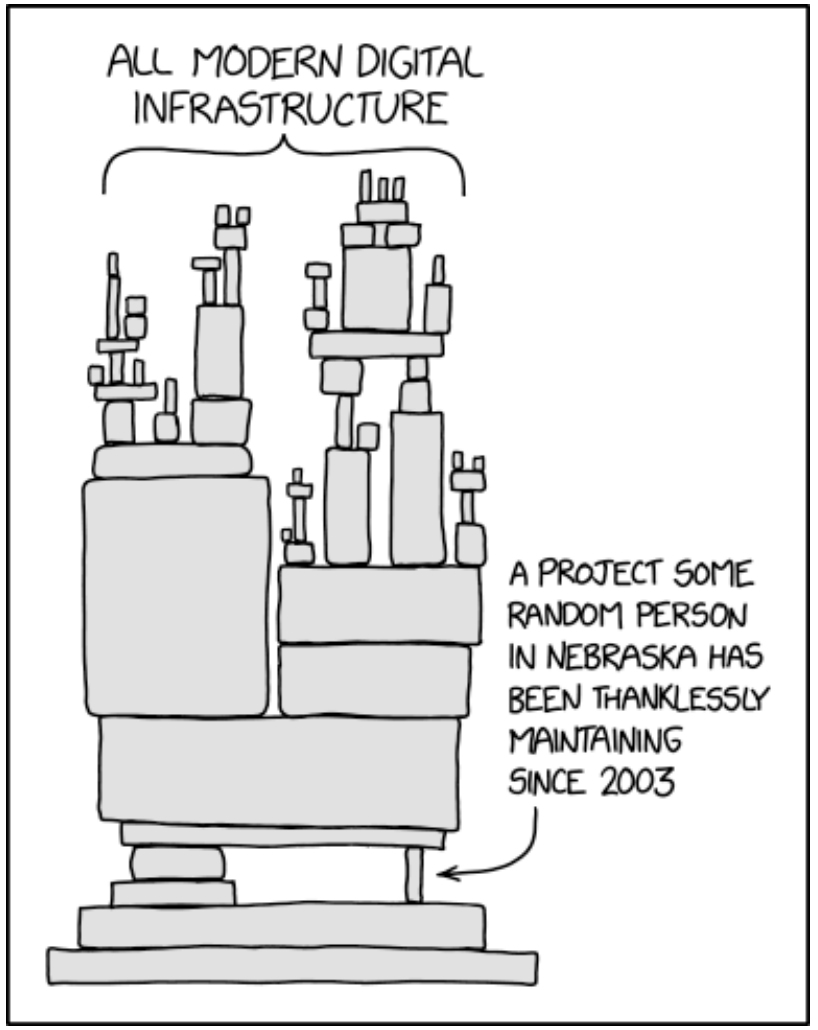


@nielstanis@infosec.exchange

MSI UEFI Signing Keys

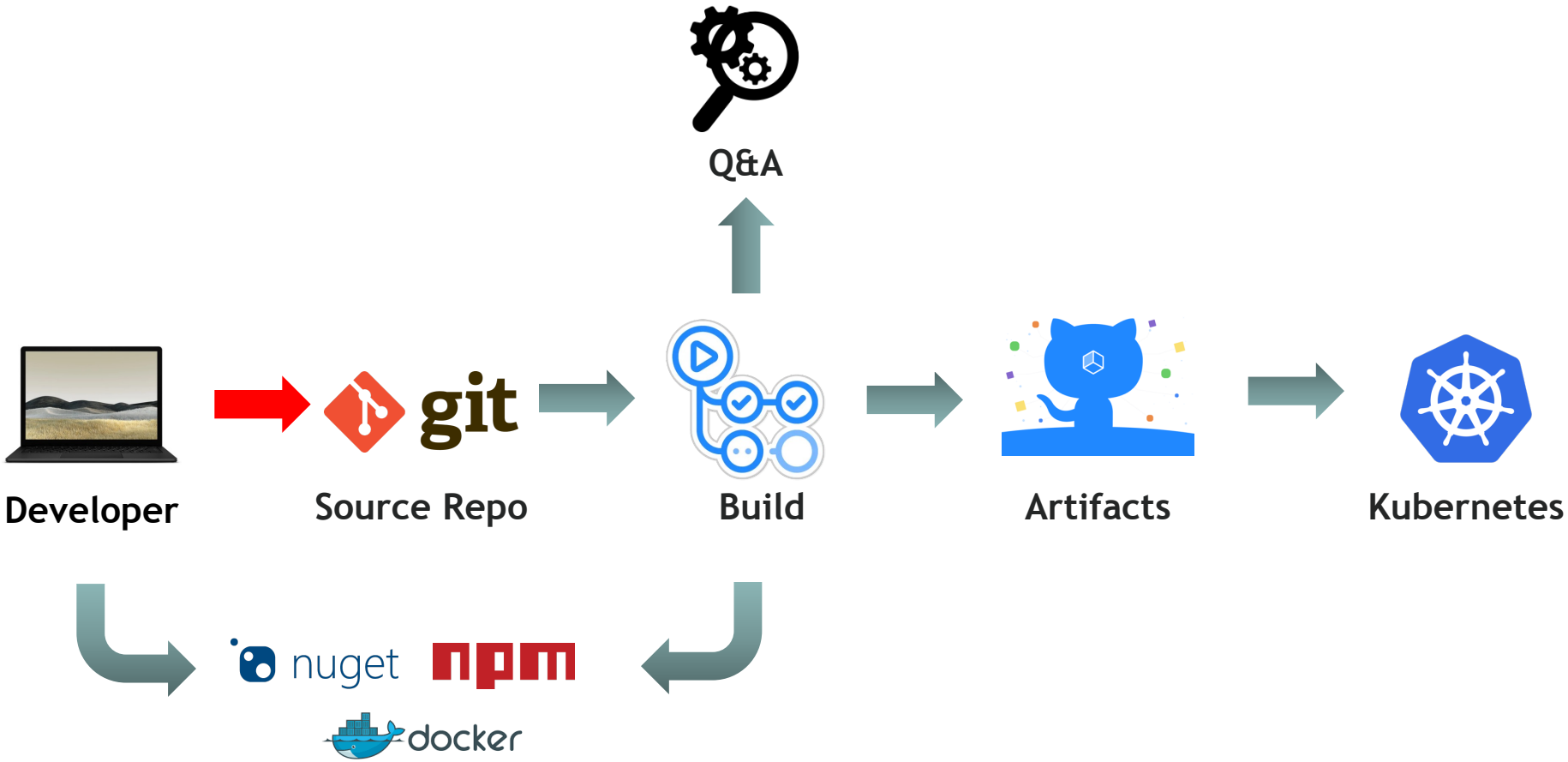


XKDC - Dependency



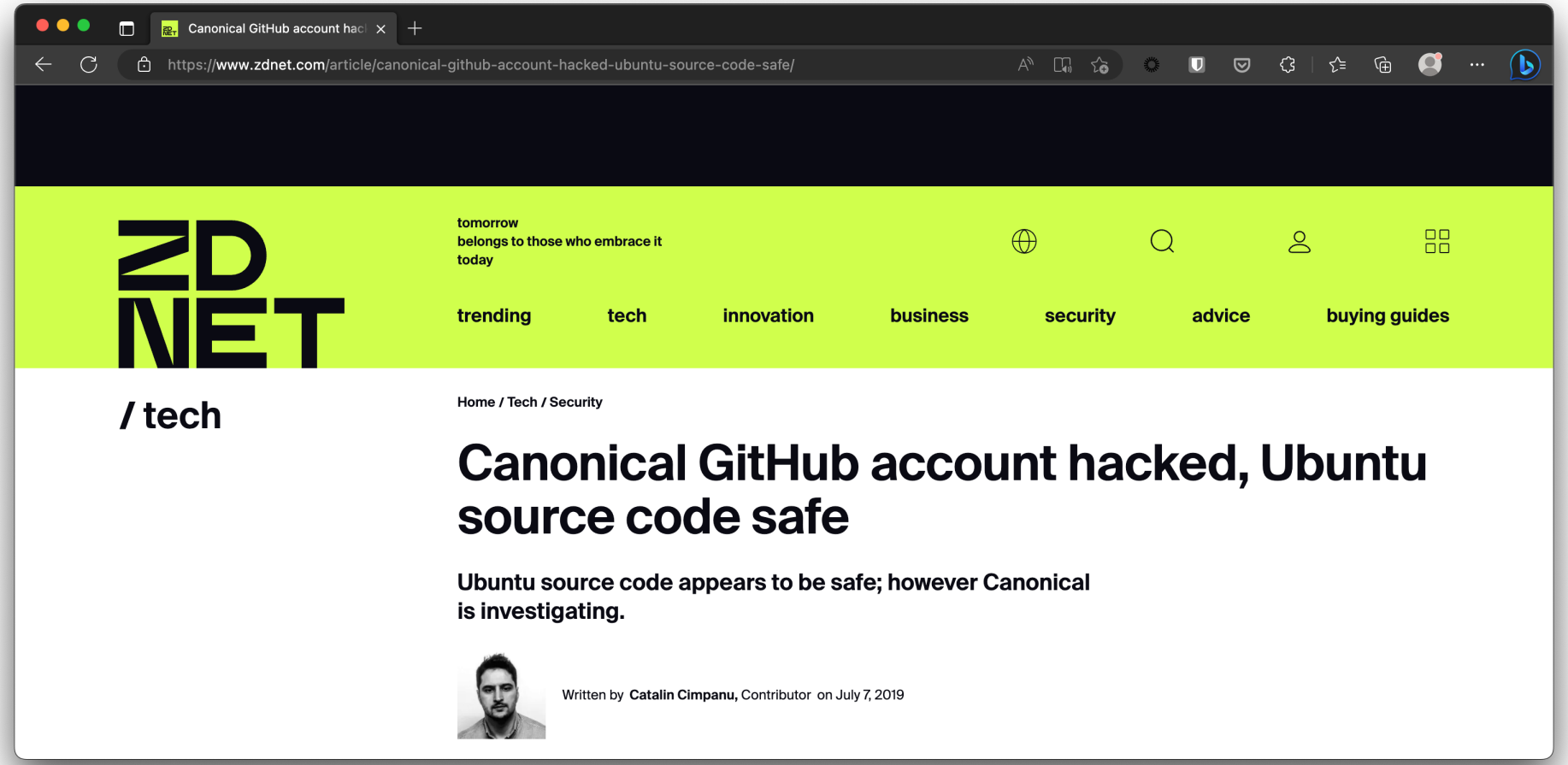
https://xkcd.com/2347/

Software Supply Chain

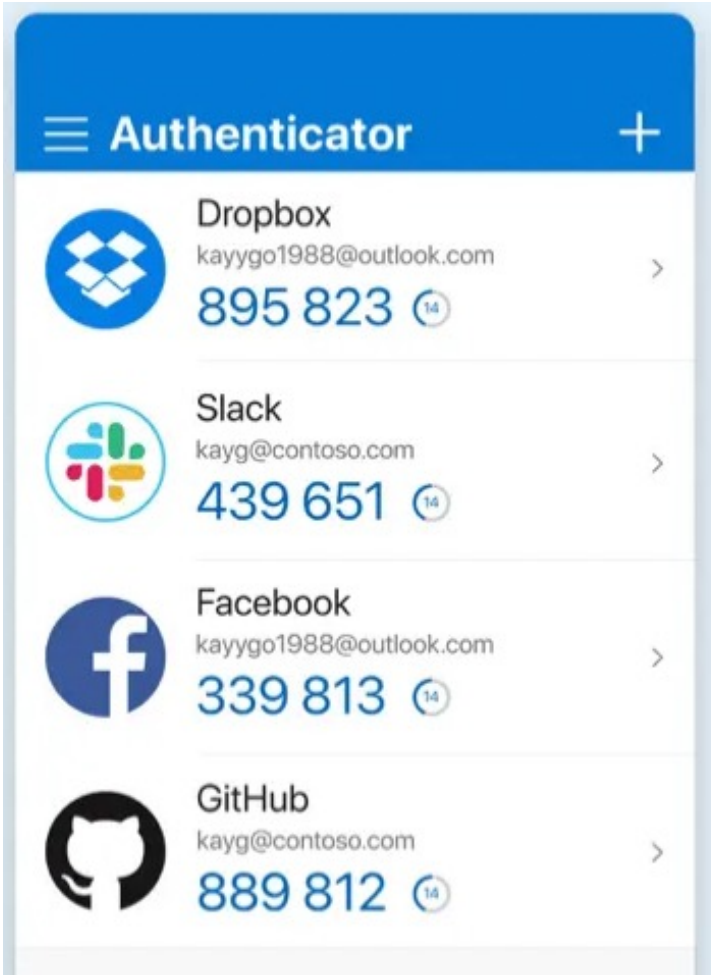


@nielstanis@infosec.exchange

GitHub account

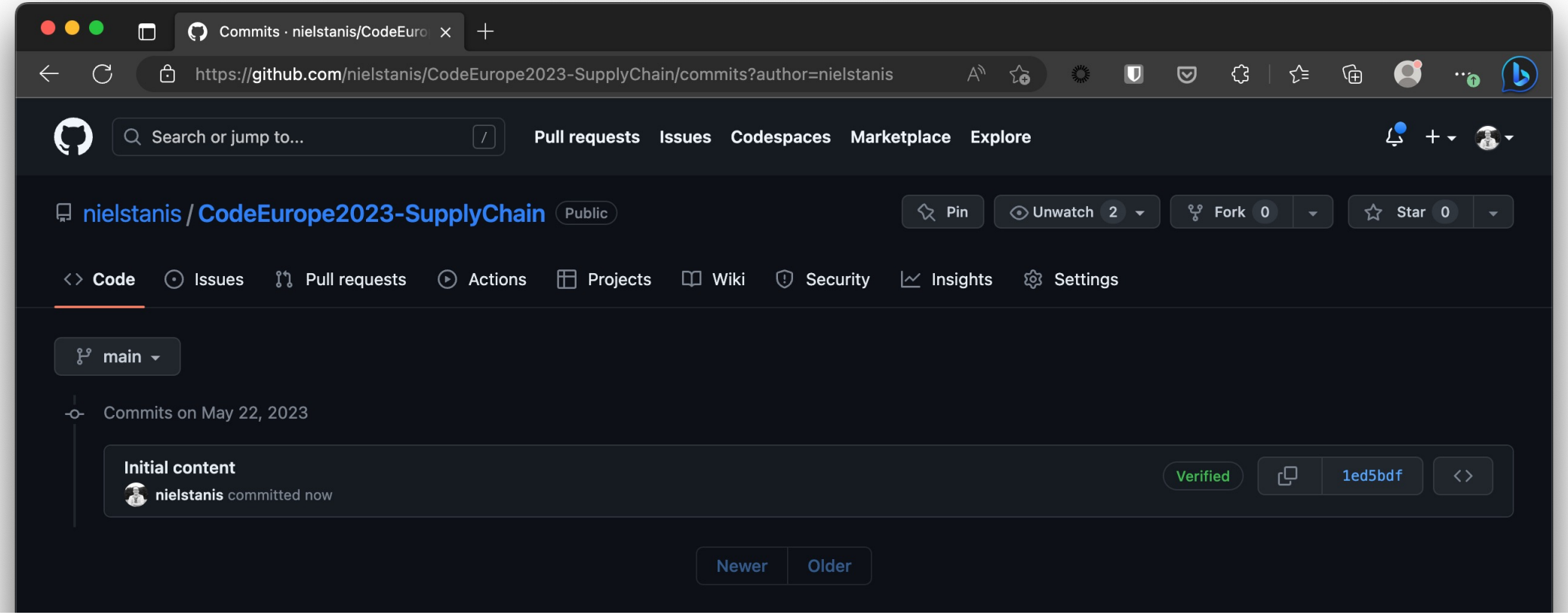


Use MFA on source-repository

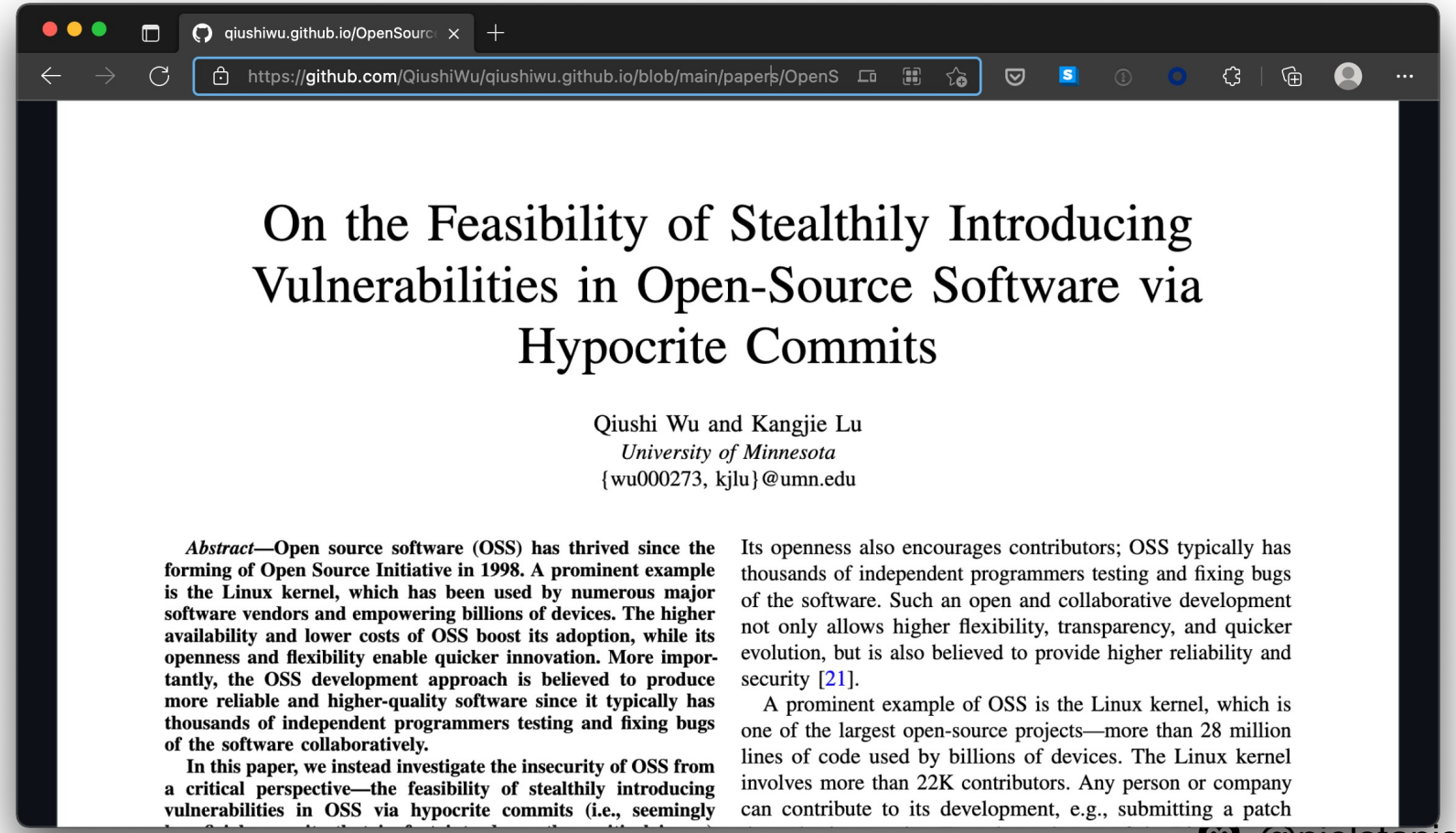


@nielstanis@infosec.exchange

GIT Commit Signing



Hypocrite Commits



On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits

Qiushi Wu and Kangjie Lu
University of Minnesota
{wu000273, kjlu}@umn.edu

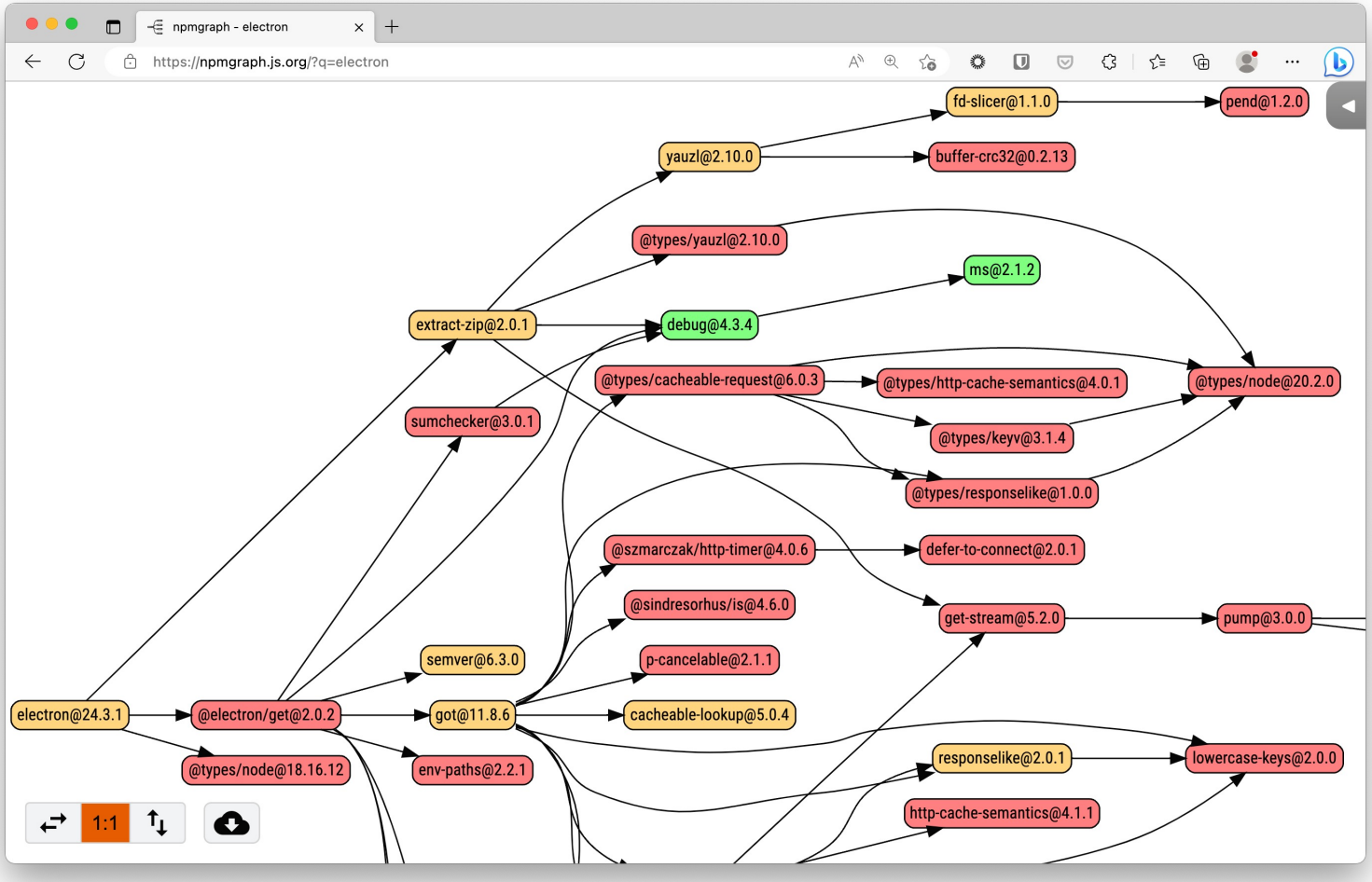
***Abstract*—Open source software (OSS) has thrived since the forming of Open Source Initiative in 1998. A prominent example is the Linux kernel, which has been used by numerous major software vendors and empowering billions of devices. The higher availability and lower costs of OSS boost its adoption, while its openness and flexibility enable quicker innovation. More importantly, the OSS development approach is believed to produce more reliable and higher-quality software since it typically has thousands of independent programmers testing and fixing bugs of the software collaboratively.**

In this paper, we instead investigate the insecurity of OSS from a critical perspective—the feasibility of stealthily introducing vulnerabilities in OSS via hypocrite commits (i.e., seemingly

Its openness also encourages contributors; OSS typically has thousands of independent programmers testing and fixing bugs of the software. Such an open and collaborative development not only allows higher flexibility, transparency, and quicker evolution, but is also believed to provide higher reliability and security [21].

A prominent example of OSS is the Linux kernel, which is one of the largest open-source projects—more than 28 million lines of code used by billions of devices. The Linux kernel involves more than 22K contributors. Any person or company can contribute to its development, e.g., submitting a patch

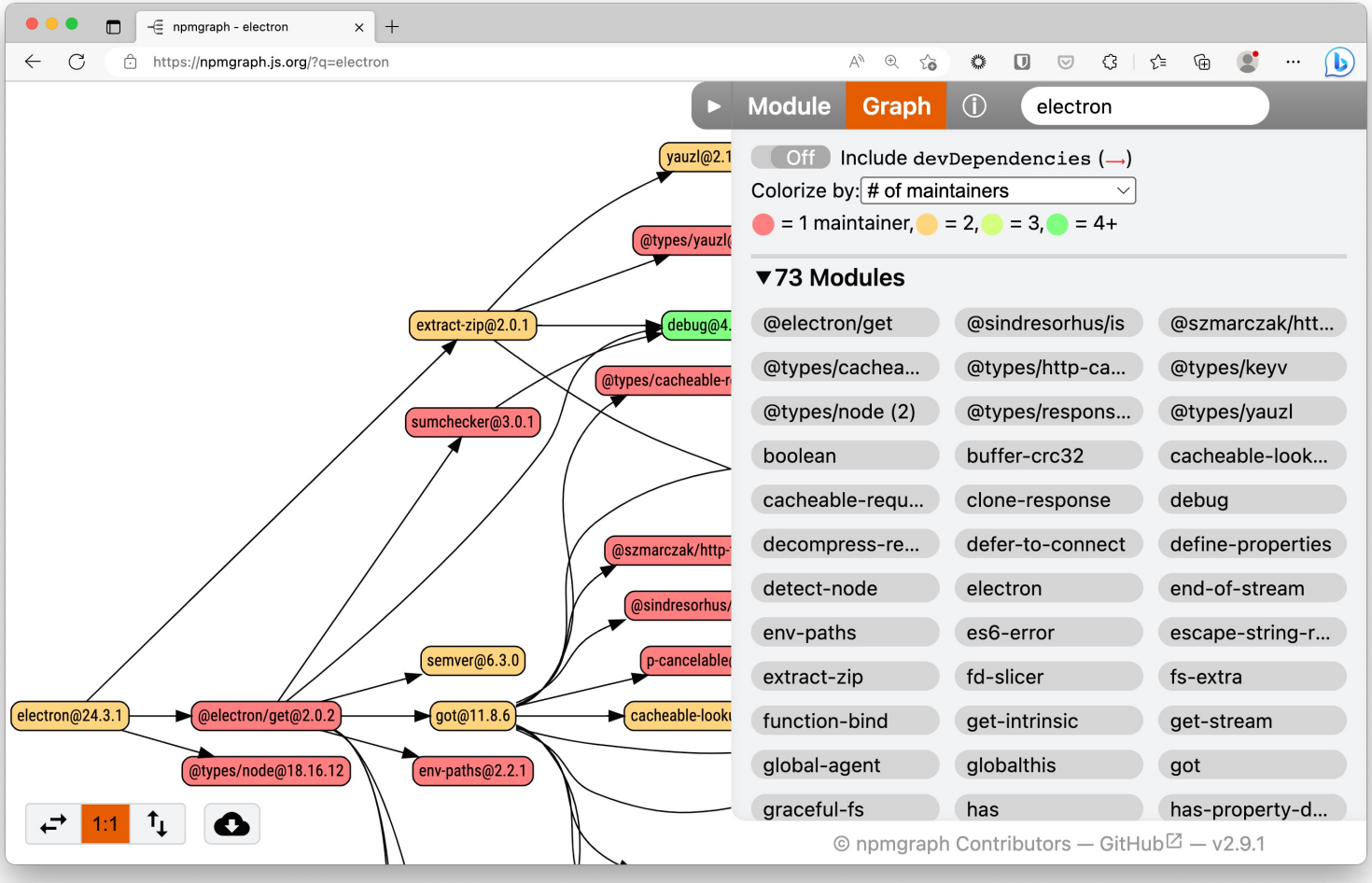
Visual Studio Code



CODE EUROPE

@nielstanis@infosec.exchange

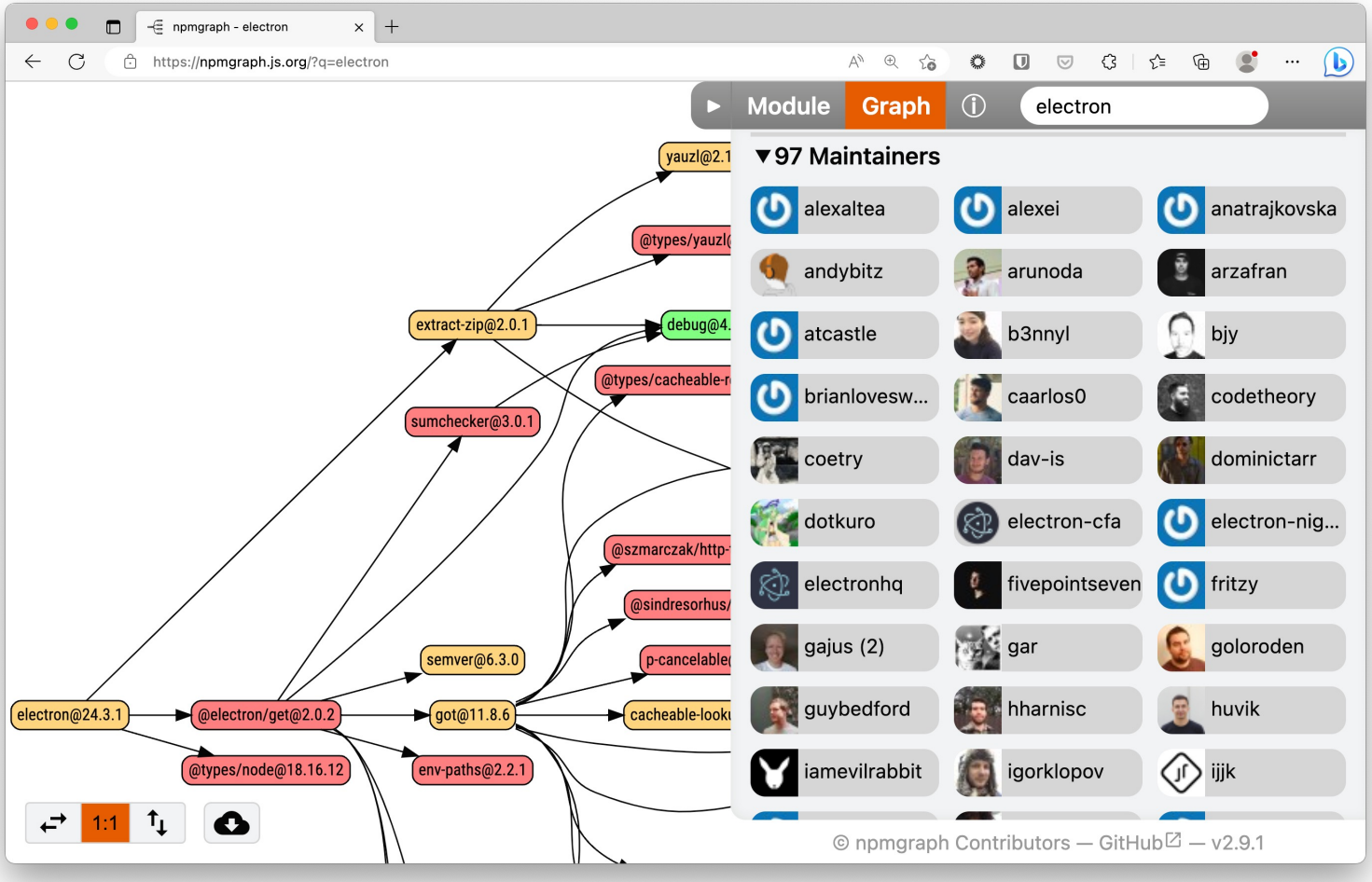
Visual Studio Code



CODE EUROPE

@nielstanis@infosec.exchange

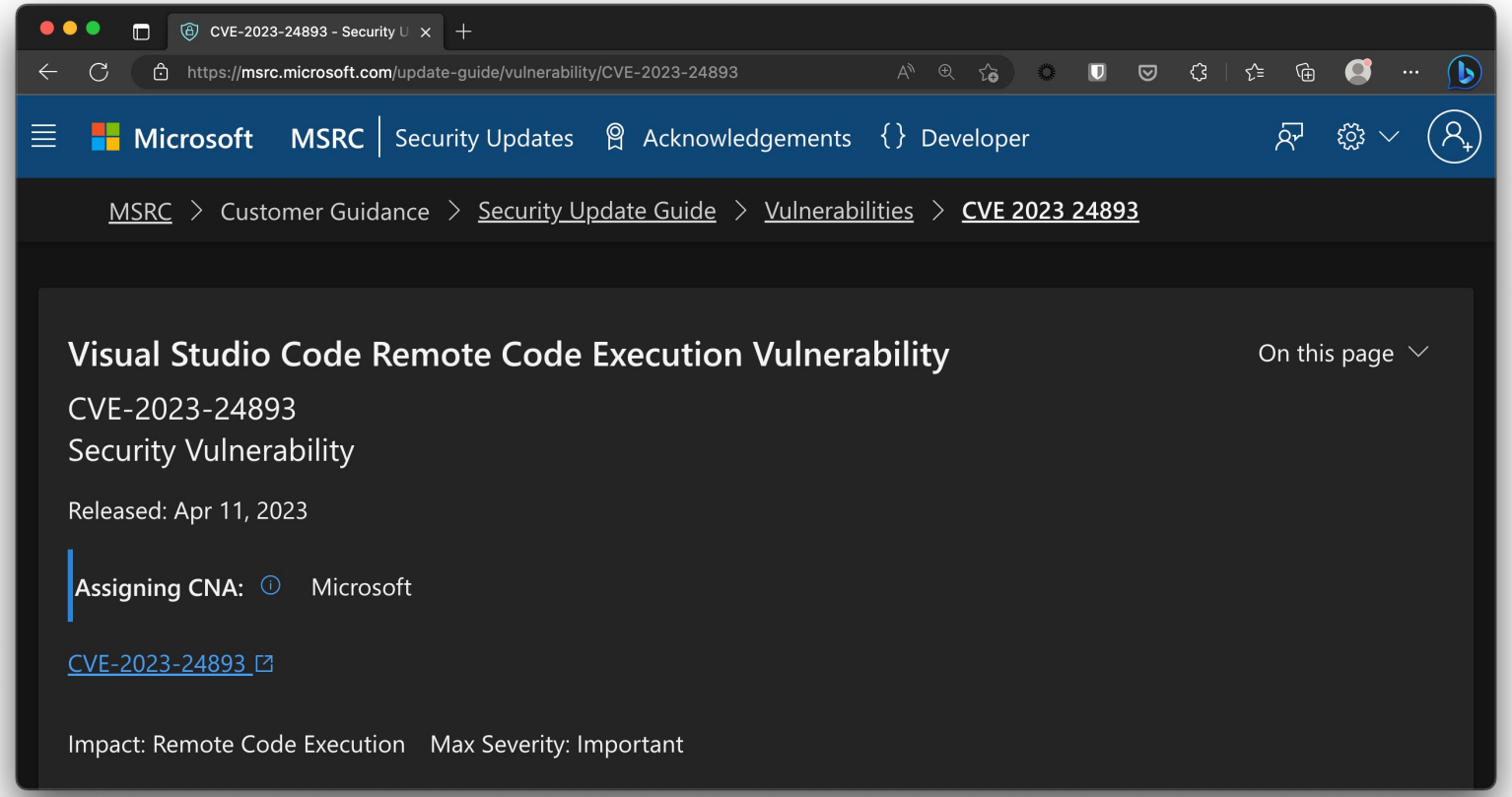
Visual Studio Code



CODE EUROPE

@nielstanis@infosec.exchange

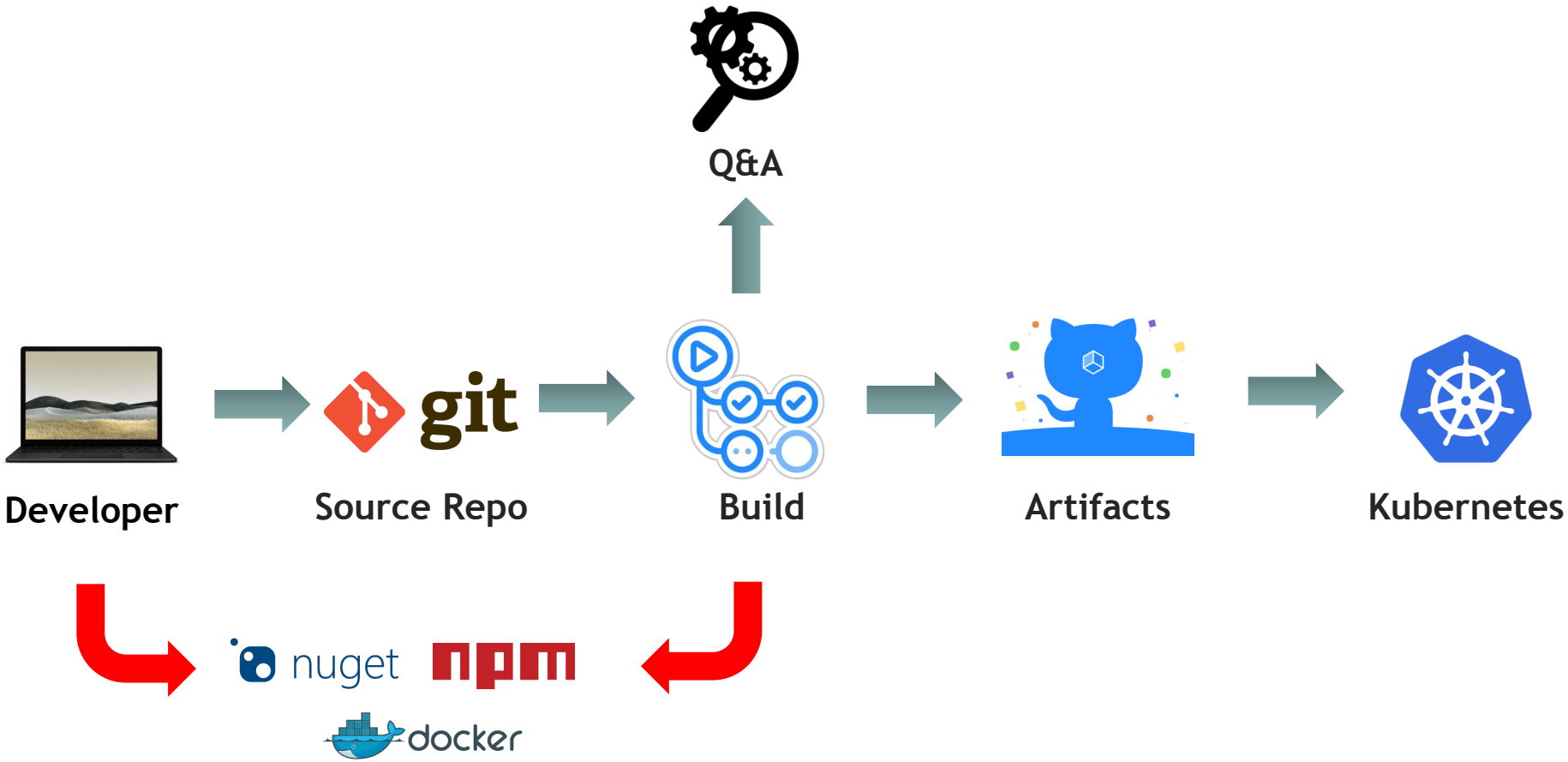
Visual Studio Code



@nielstanis@infosec.exchange

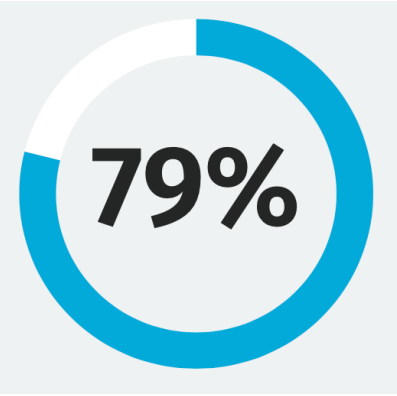
Visual Studio Code

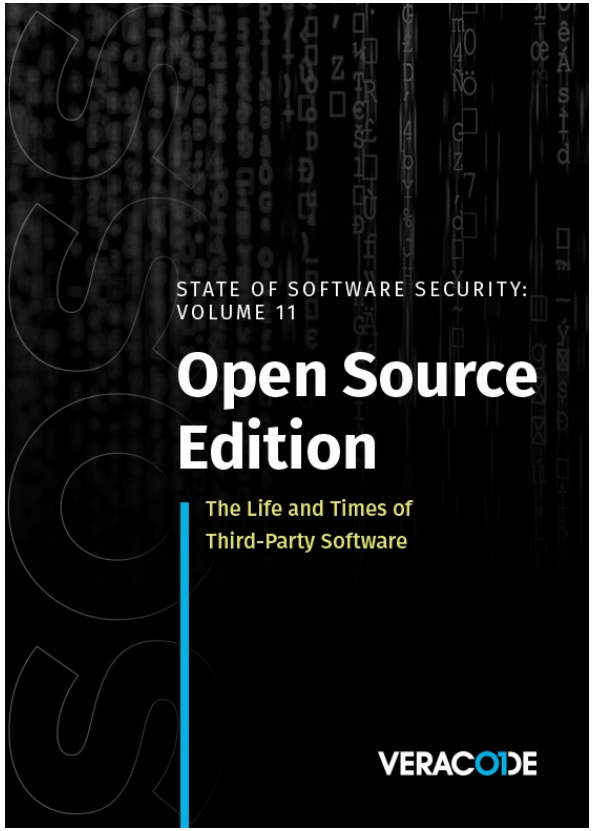
3rd Party Libraries



State Of Software Security v11 2021

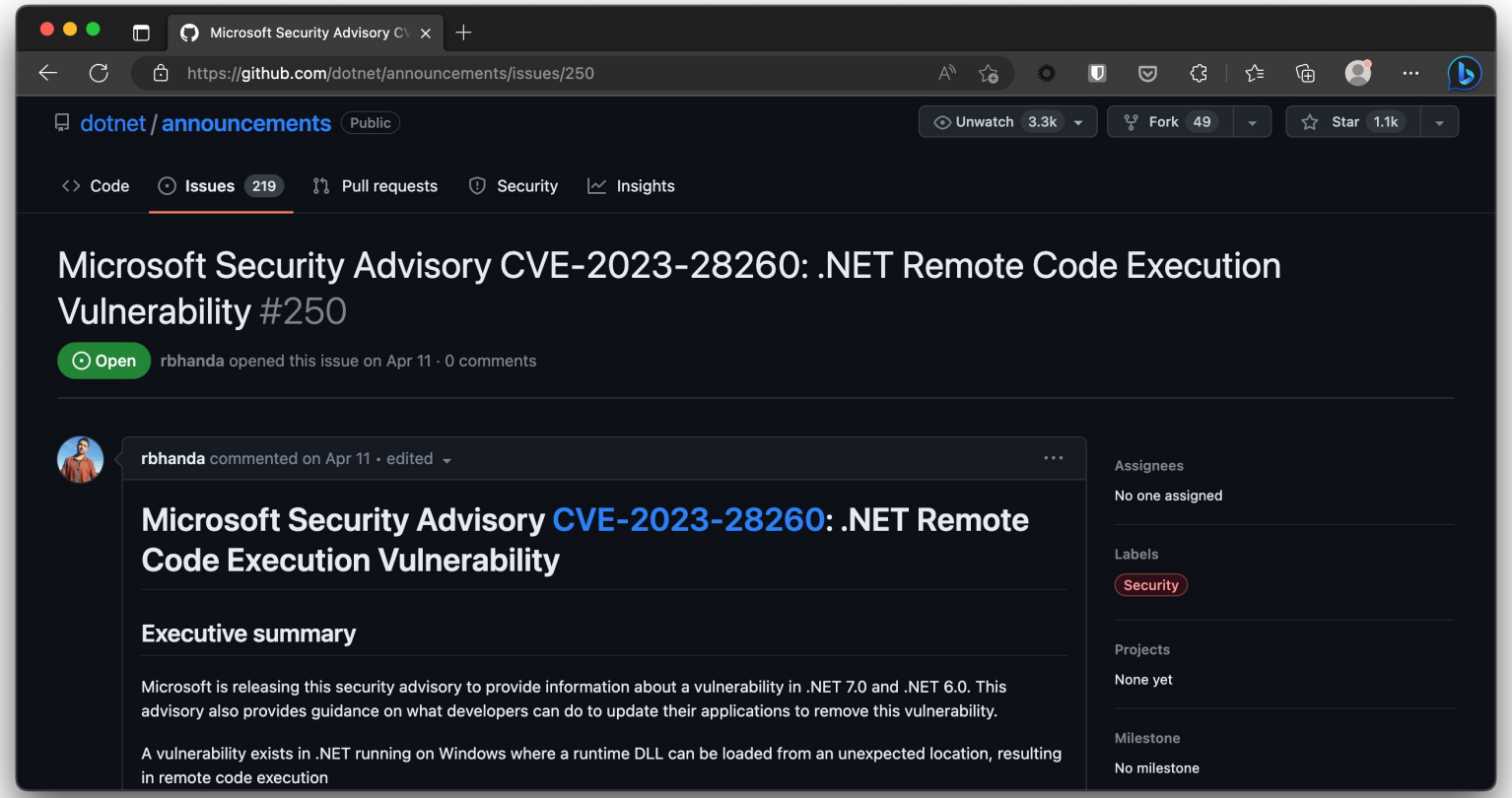
”Despite this dynamic landscape, 79 percent of the time, developers never update third-party libraries after including them in a codebase.”





@nielstanis@infosec.exchange

Vulnerabilities in libraries



Vulnerabilities in libraries

CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

Alerts and Tips | Resources | Industrial Control Systems

National Cyber Awareness System > Current Activity > Malware Discovered in Popular NPM Package, ua-parser-js

Malware Discovered in Popular NPM Package, ua-parser-js

Original release date: October 22, 2021

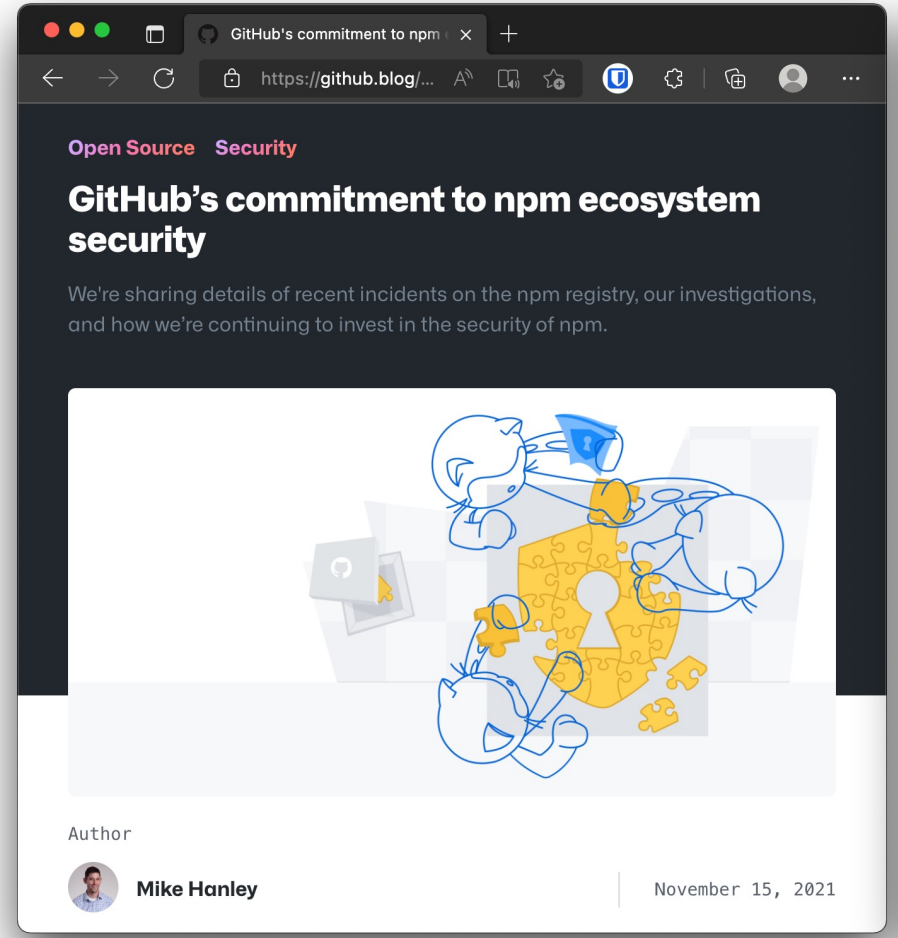
Print | Tweet | Send | Share

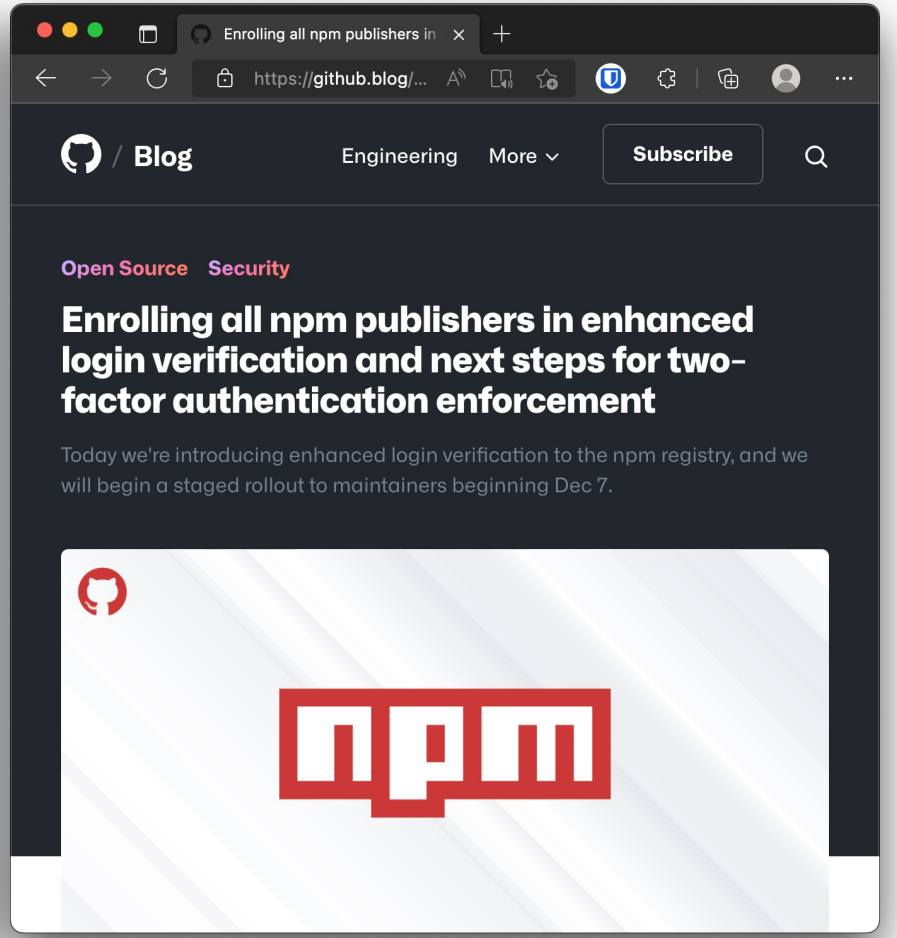
Versions of a popular NPM package named `ua-parser-js` was found to contain malicious code. `ua-parser-js` is used in apps and websites to discover the type of device or browser a person is using from User-Agent data. A computer or device with the affected software installed or running could allow a remote attacker to obtain sensitive information or take control of the system.

CISA urges users and administers using compromised ua-parser-js versions 0.7.29, 0.8.0, and 1.0.0 to update to the respective patched versions: 0.7.30, 0.8.1, 1.0.1

For more information, see Embedded malware in ua-parser-js.

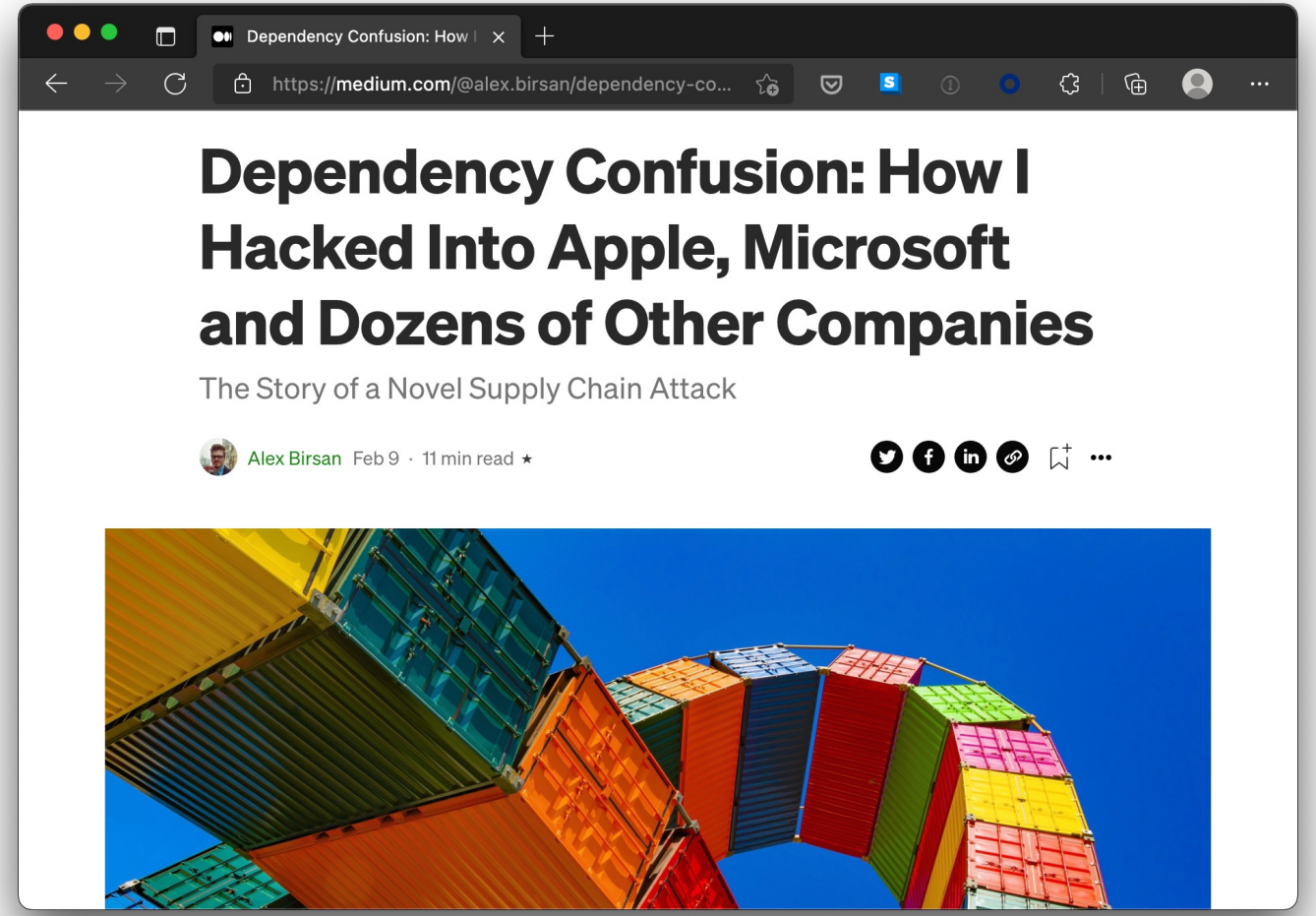
Vulnerabilities in libraries





@nielstanis@infosec.exchange

Dependency Confusion



Microsoft Whitepaper

3 ways to mitigate risk when using private package feeds

Secure Your Hybrid Software Supply Chain

An always-up-to-date version of this whitepaper is located at: https://aka.ms/pkg-sec-wp

Dependency Confusion - NuGet

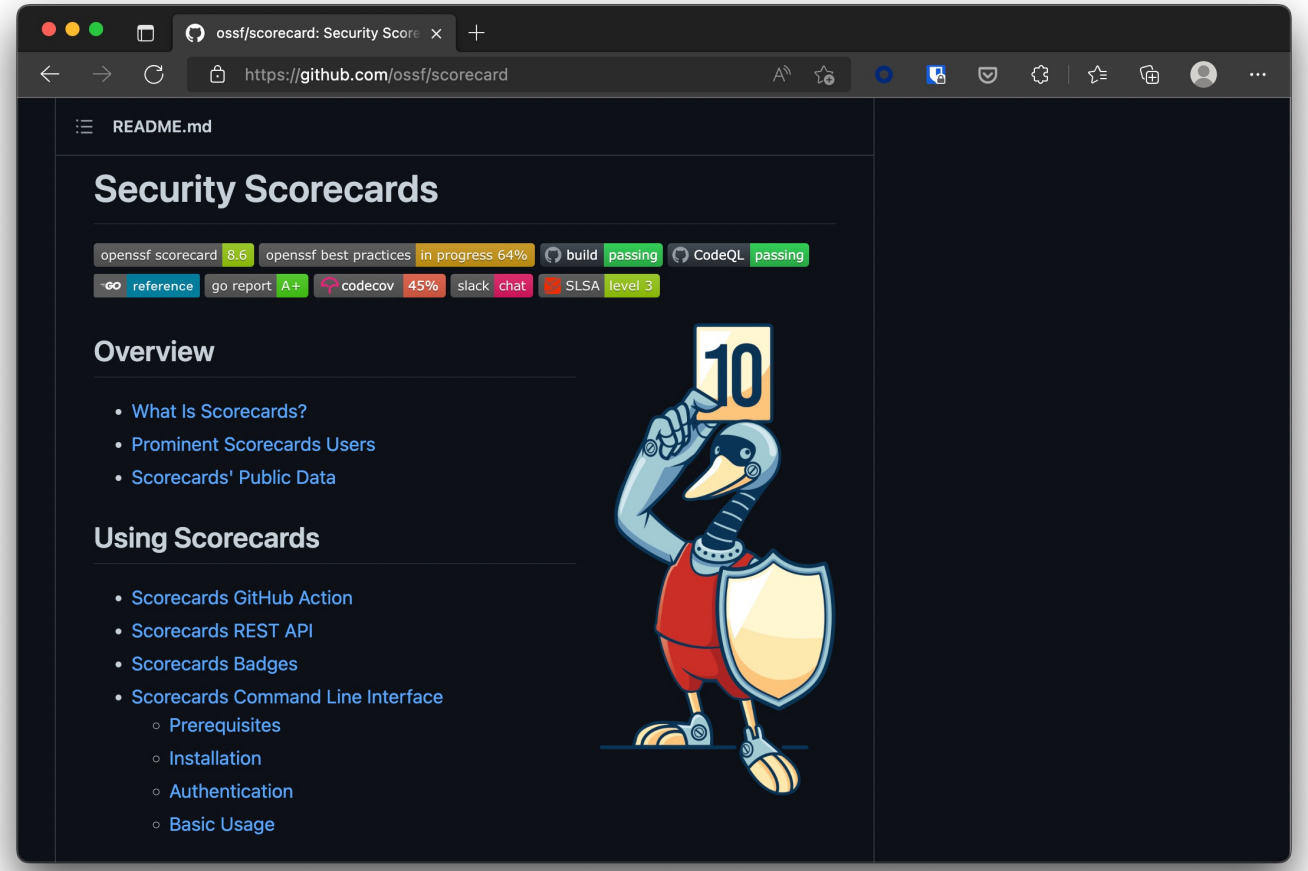
- Use single private repository
- Azure Artifacts can help manage and control upstream
- If your company has public packages consider registering prefix
- Lock packages with config

3rd Party Libraries

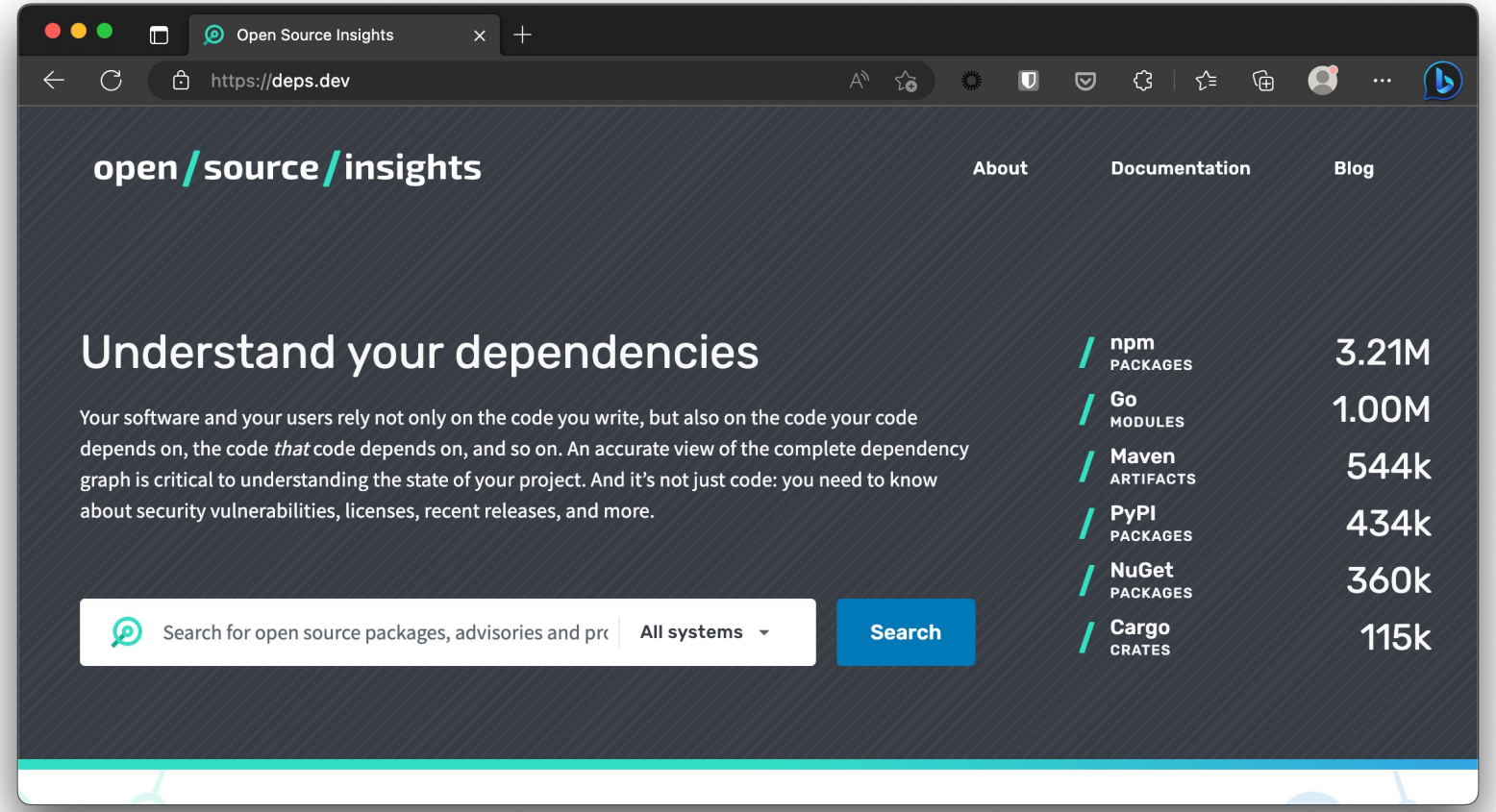
- Intent of library, know what's inside!
- Keep in mind that's a transitive list of dependencies
- My talk 'Sandboxing .NET Assemblies'

- Open Source Security Foundation - OpenSSF
- Security Scorecards - Security health metrics for Open Source

Security Scorecards - OpenSSF



Deps.Dev by Google



@nielstanis@infosec.exchange

Deps.Dev by Google

OpenSSF scorecard

The Open Source Security Foundation is a cross-industry collaboration to improve the security of open source software (OSS). The Scorecard provides security health metrics for open source projects.

View information about checks and how to fix failures.

SCORE

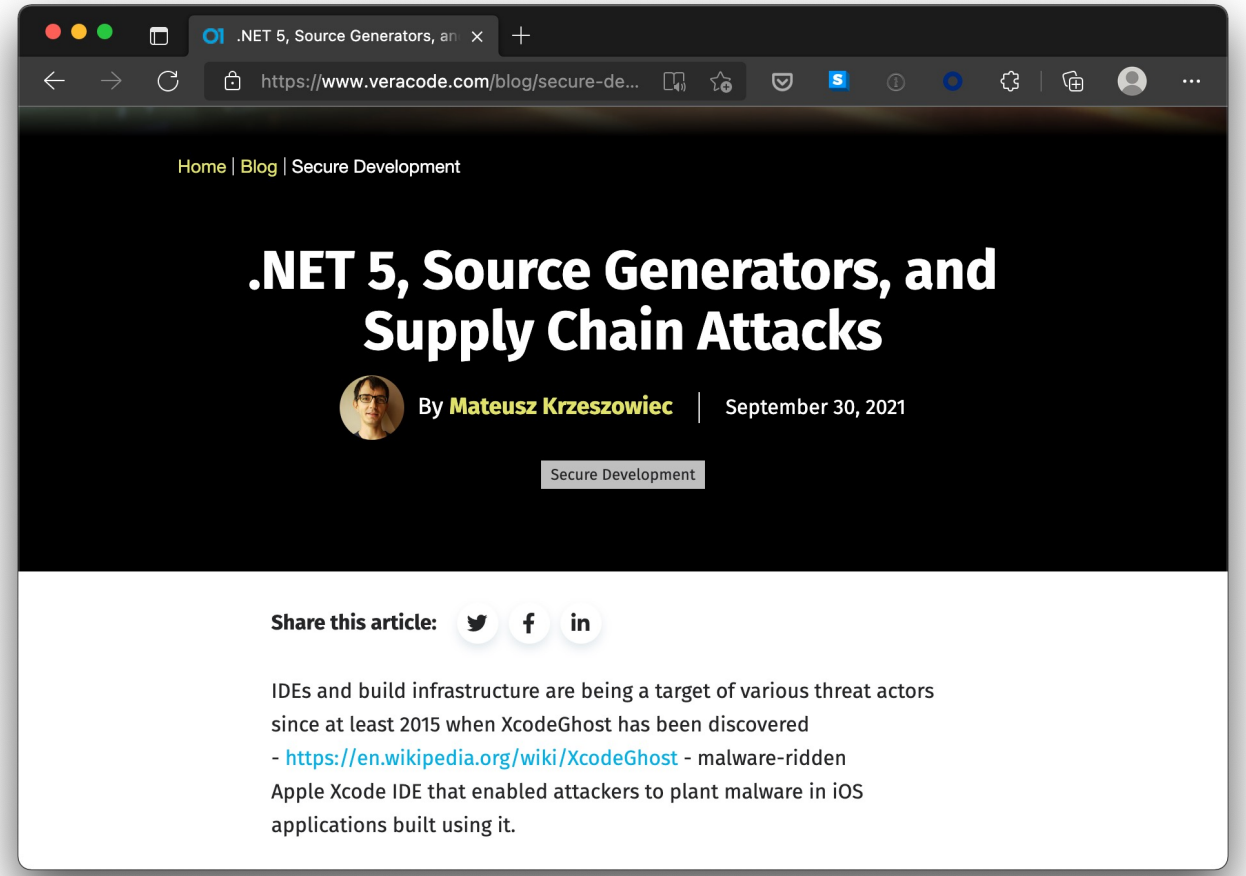
6.6/10

Scorecard as of September 12, 2022.

Maintained	10/10
Code-Review	8/10
CII-Best-Practices	0/10
Vulnerabilities	10/10
Security-Policy	10/10
Dangerous-Workflow	10/10
Token-Permissions	0/10
License	10/10
Pinned-Dependencies	7/10
Binary-Artifacts	10/10
Fuzzing	0/10
Dependency-Update-Tool	10/10
Signed-Releases	0/10
SAST	0/10
Branch-Protection	8/10

Project metadata as of September 18, 2022.

Source Generators

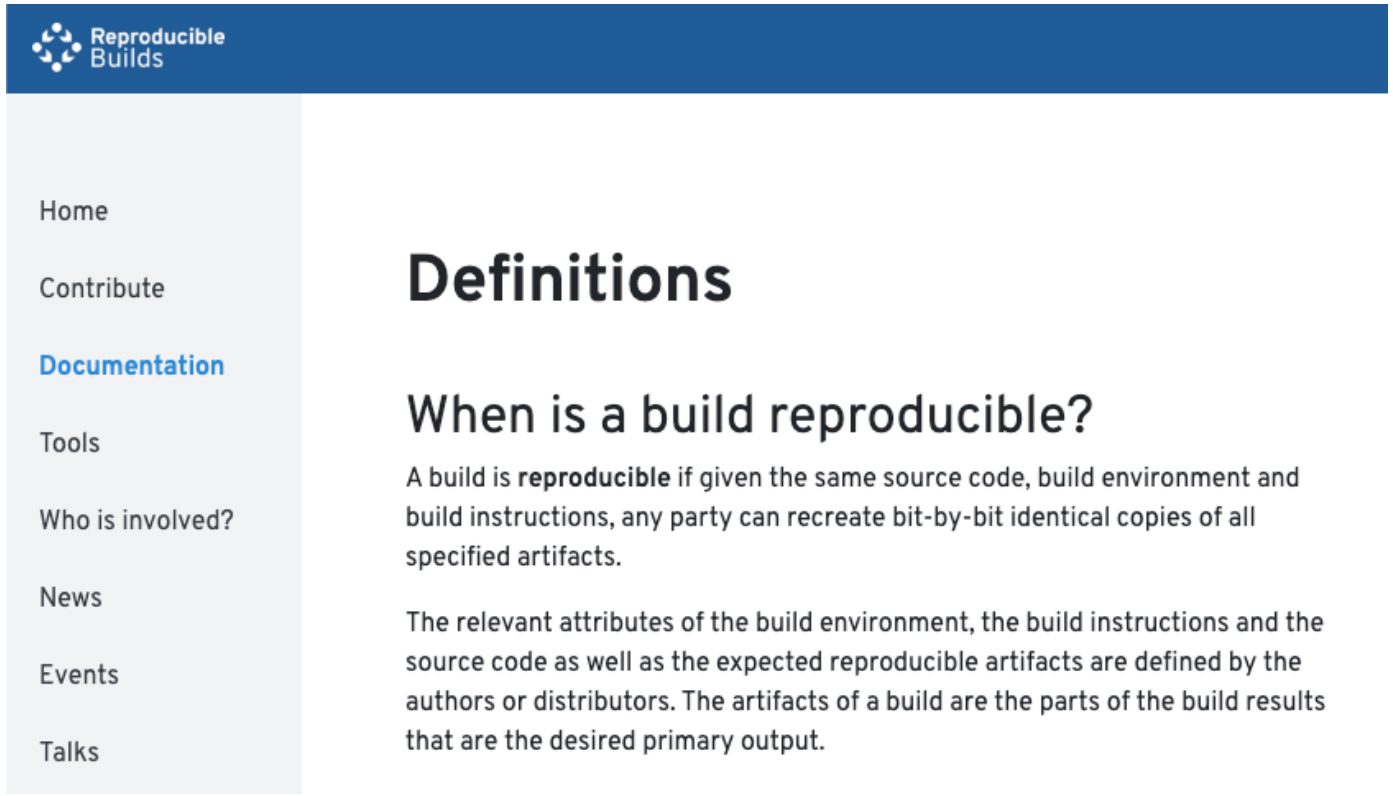


Source Generators

- Any 3rd party library can contain Source Generator!
- Consider disabling on project-level:

```
<Target Name="DisableAnalyzers"
        BeforeTargets="CoreCompile">
  <ItemGroup>
    <Analyzer Remove="@(Analyzer)" />
  </ItemGroup>
</Target>
```

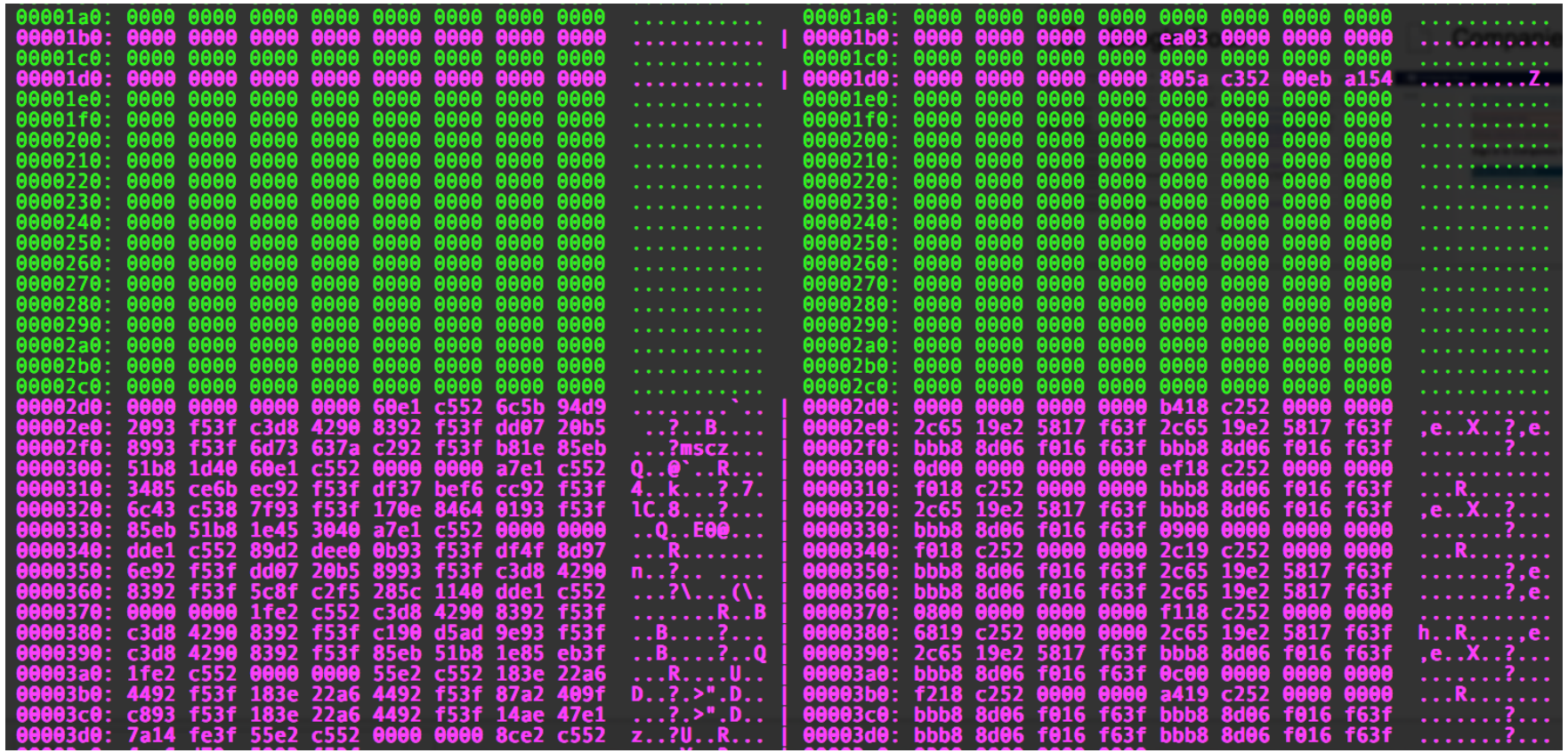
Reproducible/Deterministic Builds



Reproducible/Deterministic Builds

- Roslyn v1.1 started supporting some kind of determinism on how items are emitted
- Given same inputs, the compiled output will always be deterministic
- Inputs can be found in Roslyn compiler docs ‘Deterministic Inputs’

Reproducible Build .NET6



Reproducible/Deterministic Builds

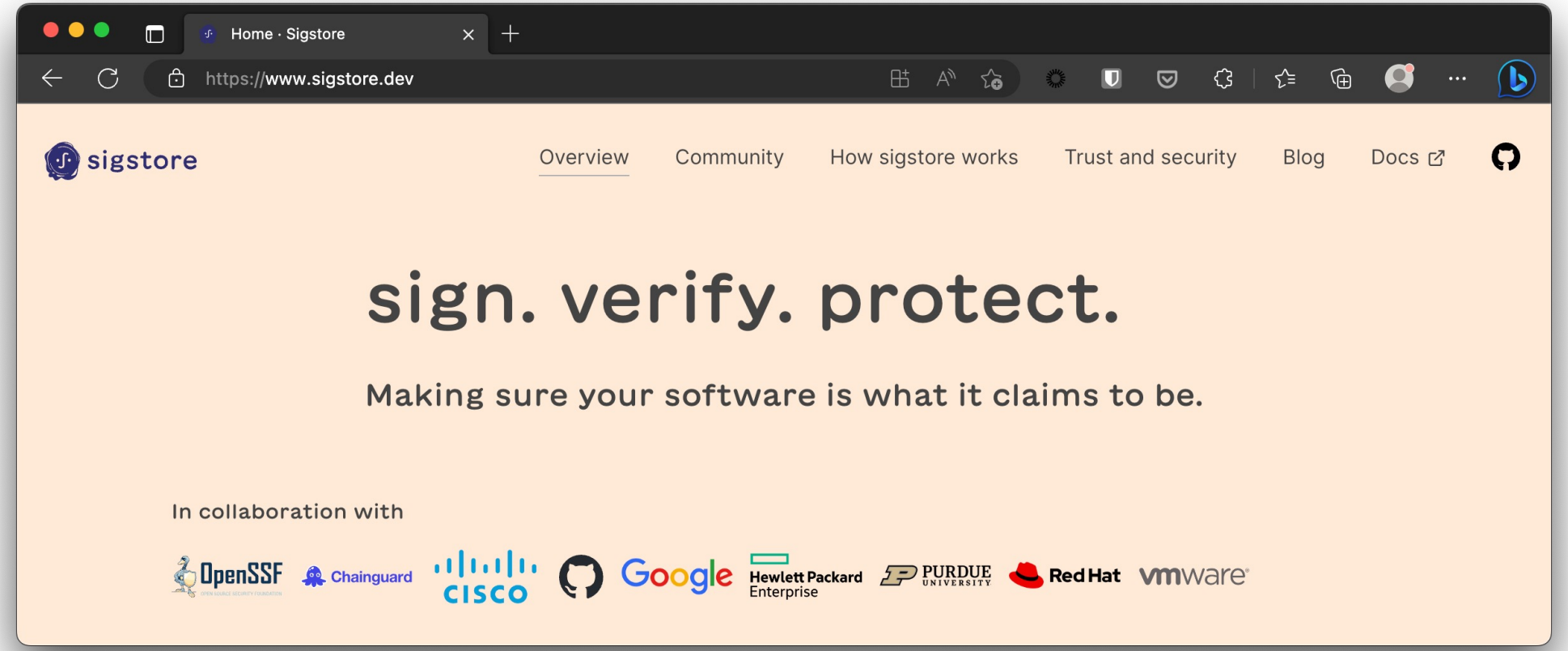
- DotNet.Reproducible NuGet Package
  - MSBuild *ContinuousIntegrationBuild*
  - SourceLink

- Dotnet.Reproducible.Isolated NuGet Package
  - Hermetic builds

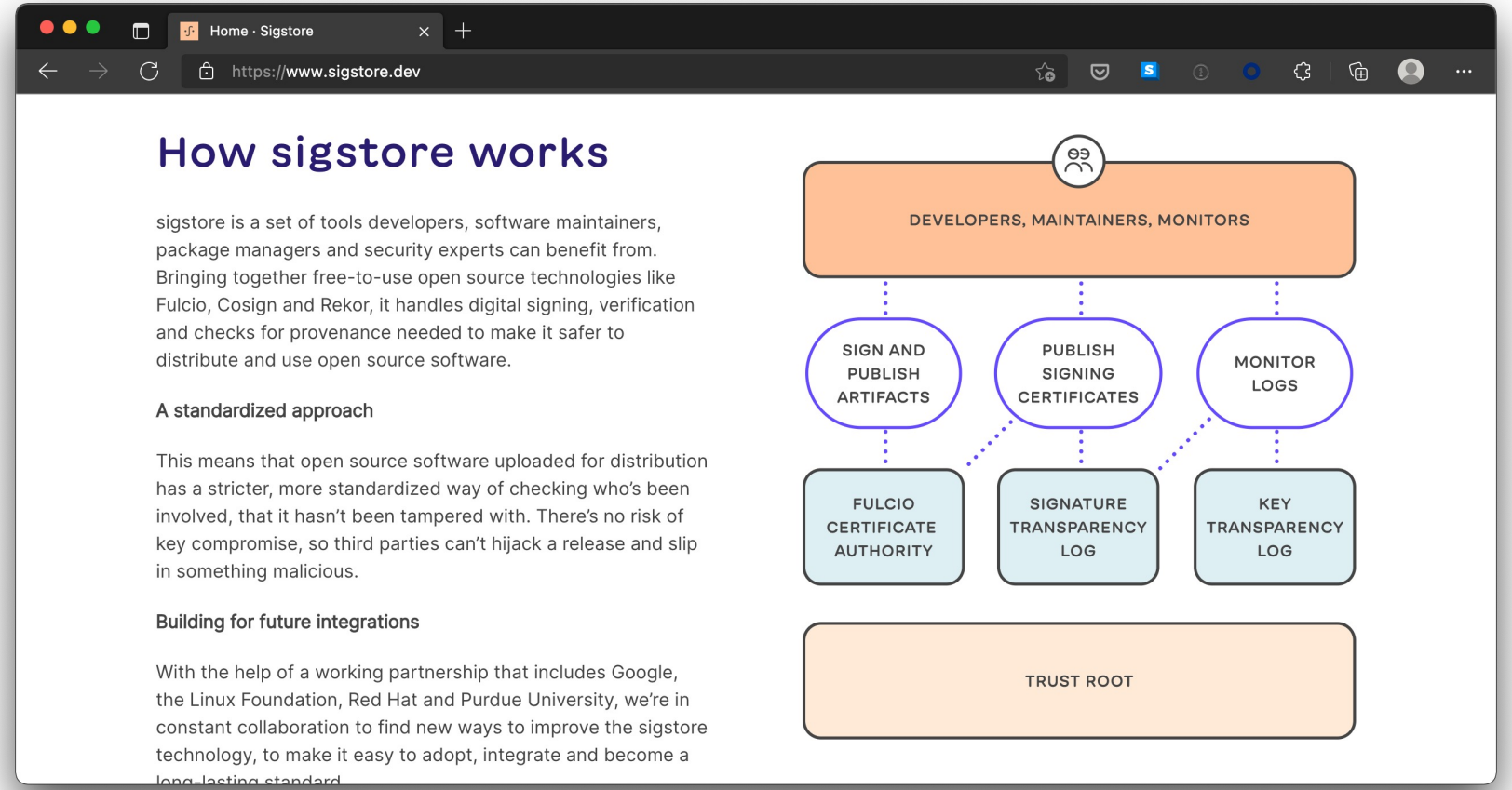
Reproducible Build Validation

- Design to validate NuGet packages & .NET binaries
  - Does linked source code match binaries?
  - Ability to rebuild reproducible based on given inputs
  - .NET CLI Validate tool `dotnet validate`

Signing artifacts



Signing artifacts



Signing artifacts

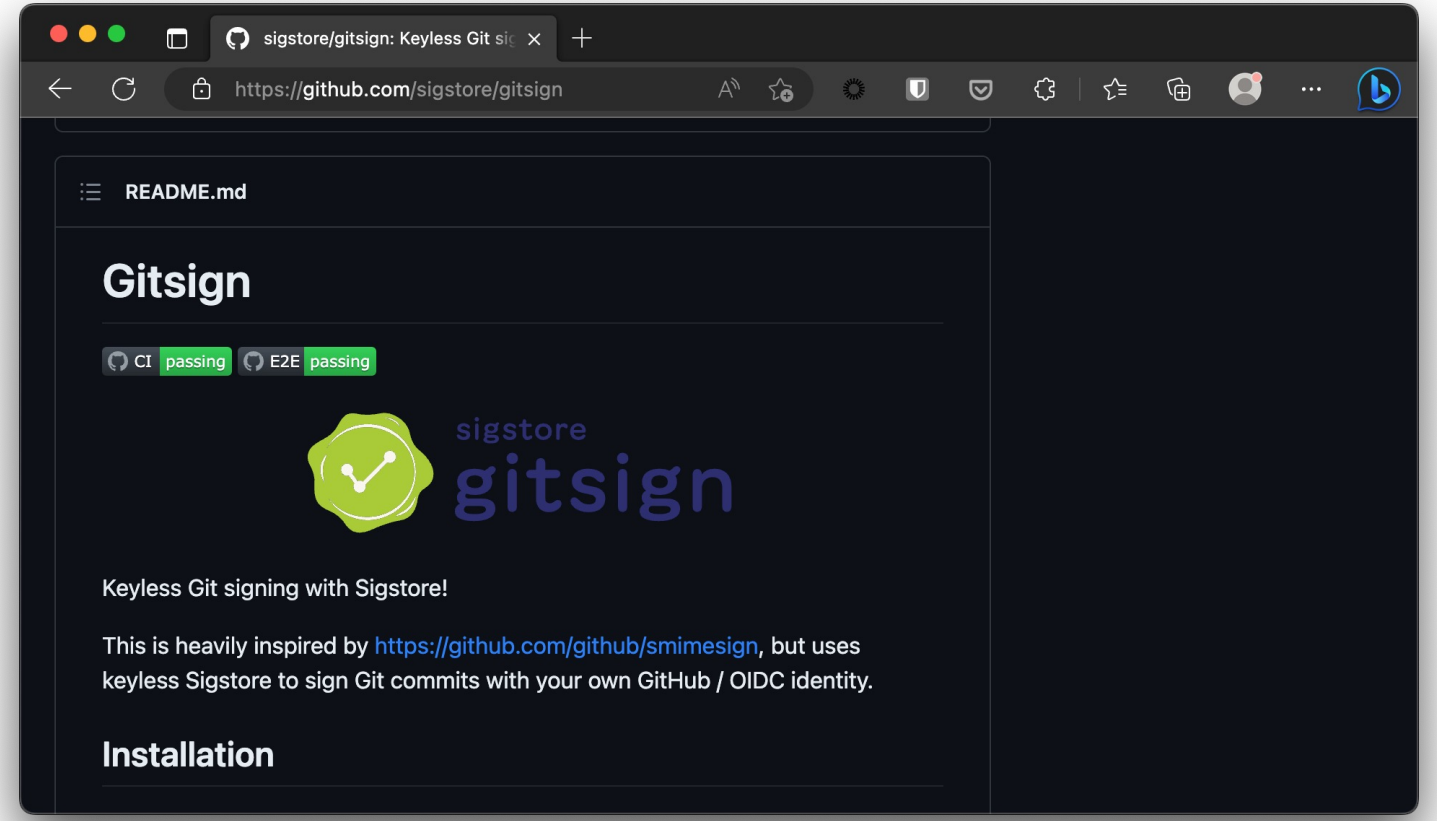
- Cosign can be used for signing files like binaries, packages and Docker images
- It can do keyless signing based on OpenID Connect
- GitHub Actions have released OpenID Connect support since end 2021

Cosign Keyless Signing

CODE EUROPE

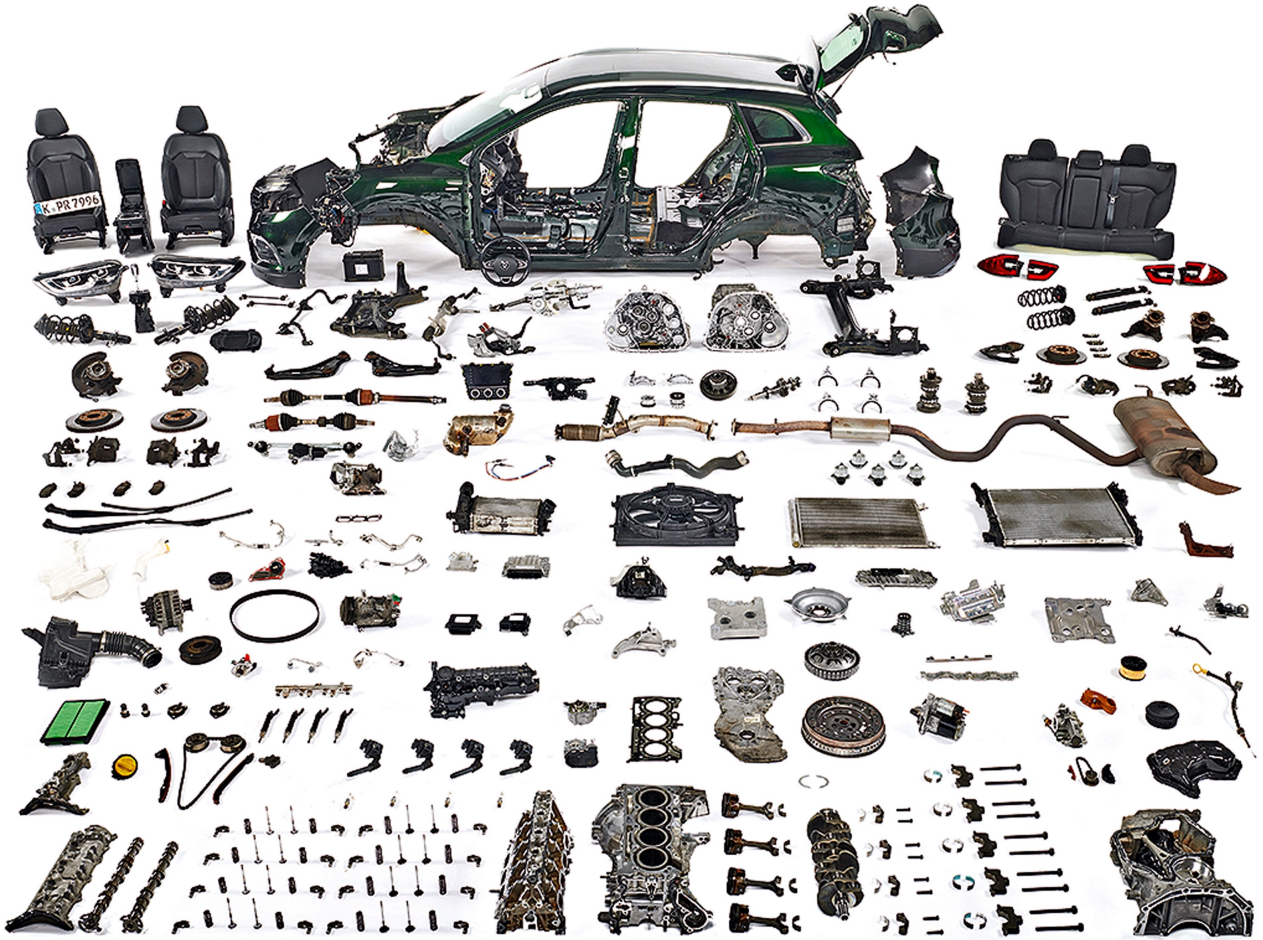
anis@infosec.exchange

Git Commit Signing Sigstore GitSign



@nielstanis@infosec.exchange

Automotive Industry



@nielstanis@infosec.exchange

Car Supply Chain

Tata Steel Factory

- Iron Ore from Sweden
- ISO 6892-1 Tested/Certified
- Batch #1234

Bosch Factory

- Steel Batch #1234 Tata
- ECE-R90 Tested/Certified
- Serie #45678
- Used by Ford, Volkswagen and Renault

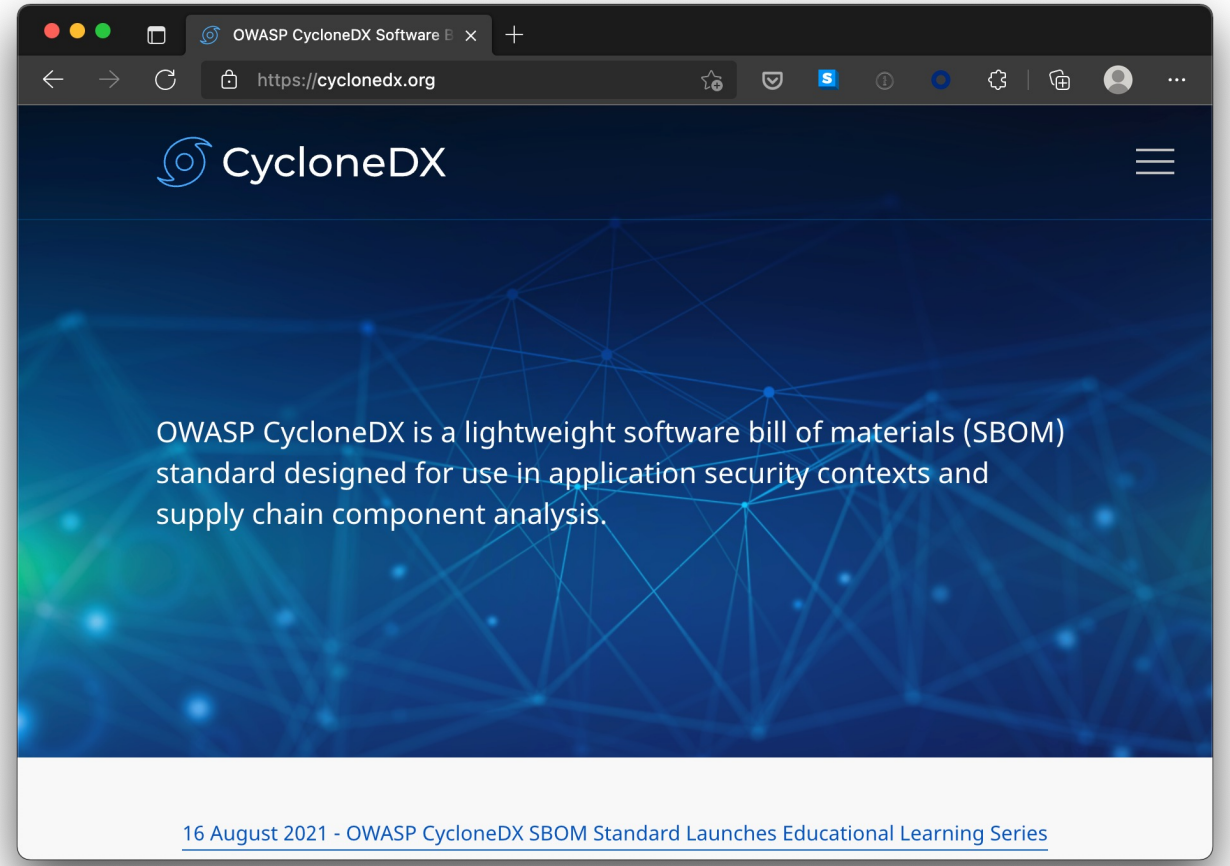
Renault Manufacturing

- Bosch Disk #45678
- Bosal Exhaust #RE9876
- Goodyear Tires #GY8877
- Kadjar VIN 1234567890

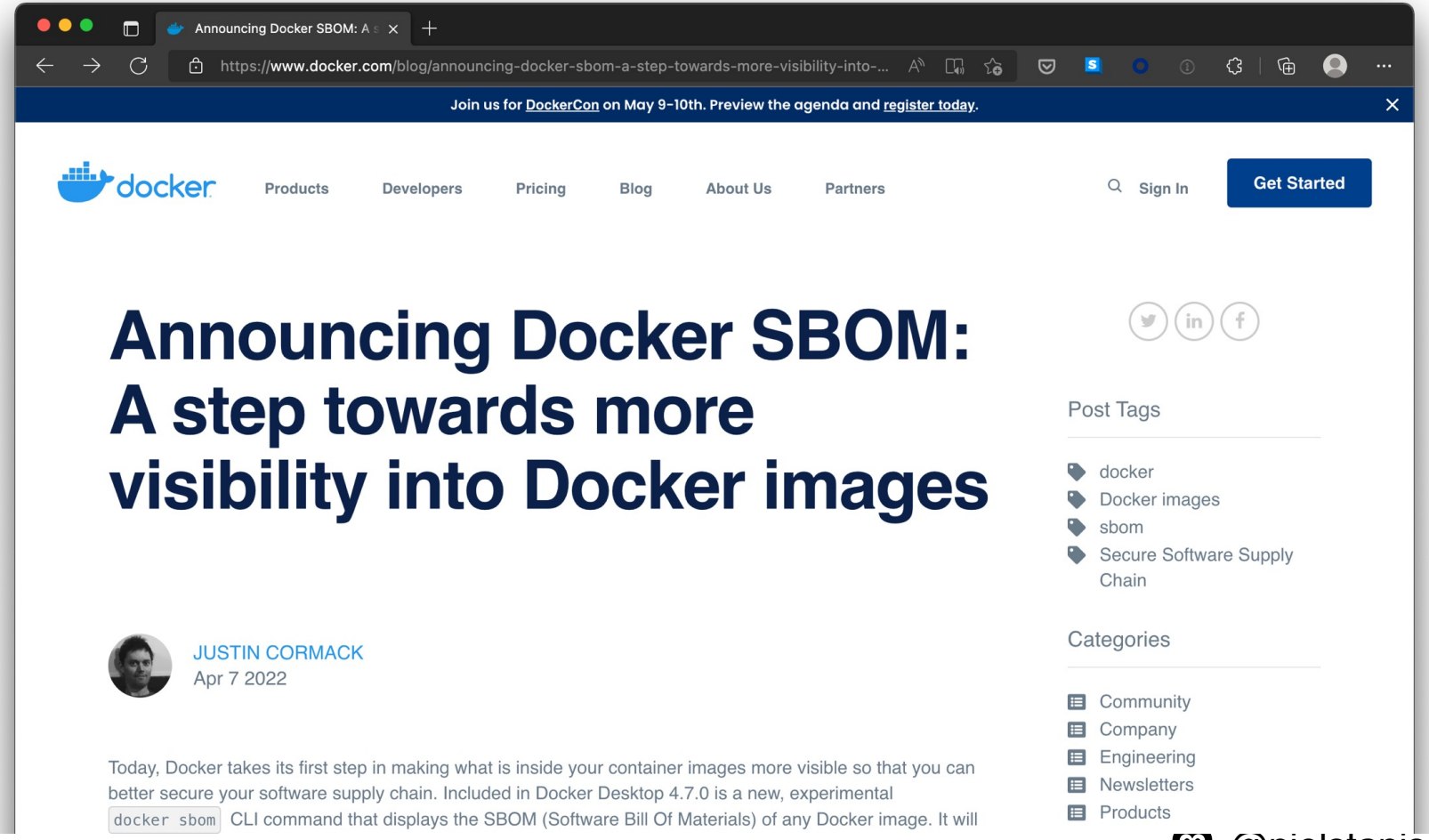
Software Bill of Materials (SBOM)

- Industry standard of describing the software
  - Producer Identity – Who Created it?
  - Product Identity – What's the product?
  - Integrity – Is the project unaltered?
  - Licensing – How can the project be used?
  - Creation – How was the product created? Process meets requirements?
  - Materials - How was the product created? Materials/Source used?

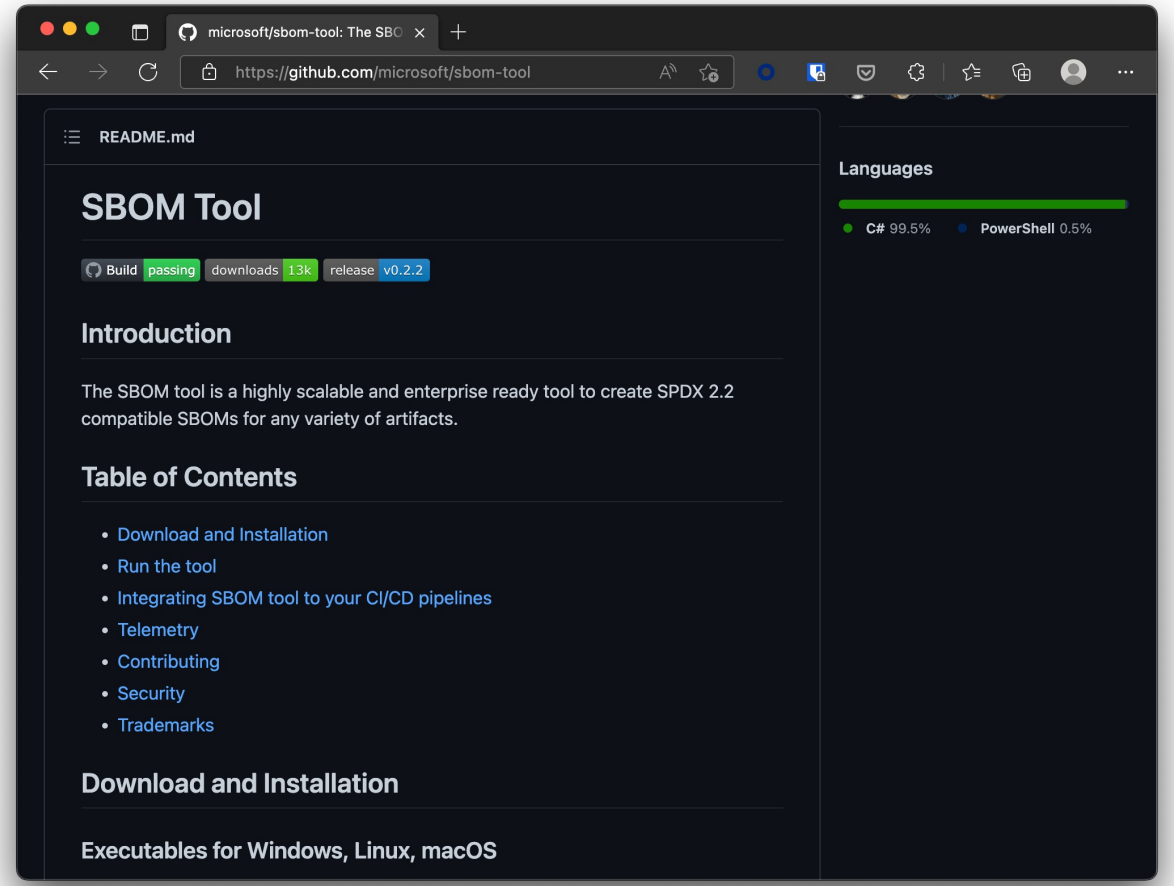
Software Bill of Materials (SBOM)



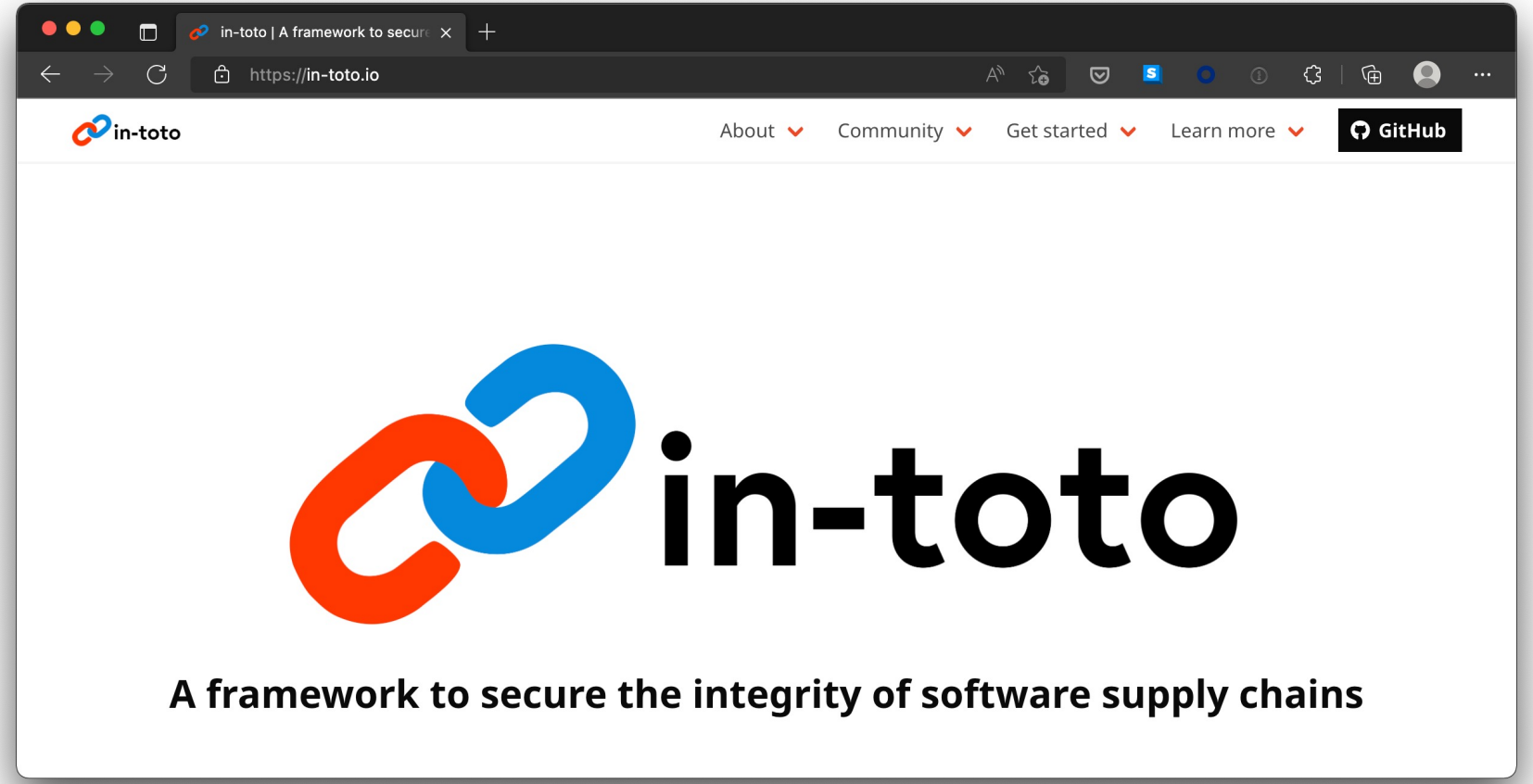
Docker SBOM


Announcing Docker SBOM: A
https://www.docker.com/blog/announcing-docker-sbom-a-step-towards-more-visibility-into-...
Join us for DockerCon on May 9-10th. Preview the agenda and register today.
docker
Products
Developers
Pricing
Blog
About Us
Partners
Sign In
Get Started
Announcing Docker SBOM: A step towards more visibility into Docker images
JUSTIN CORMACK
Apr 7 2022
Today, Docker takes its first step in making what is inside your container images more visible so that you can better secure your software supply chain. Included in Docker Desktop 4.7.0 is a new, experimental
docker sbom
CLI command that displays the SBOM (Software Bill Of Materials) of any Docker image. It will
Post Tags
docker
Docker images
sbom
Secure Software Supply Chain
Categories
Community
Company
Engineering
Newsletters
Products

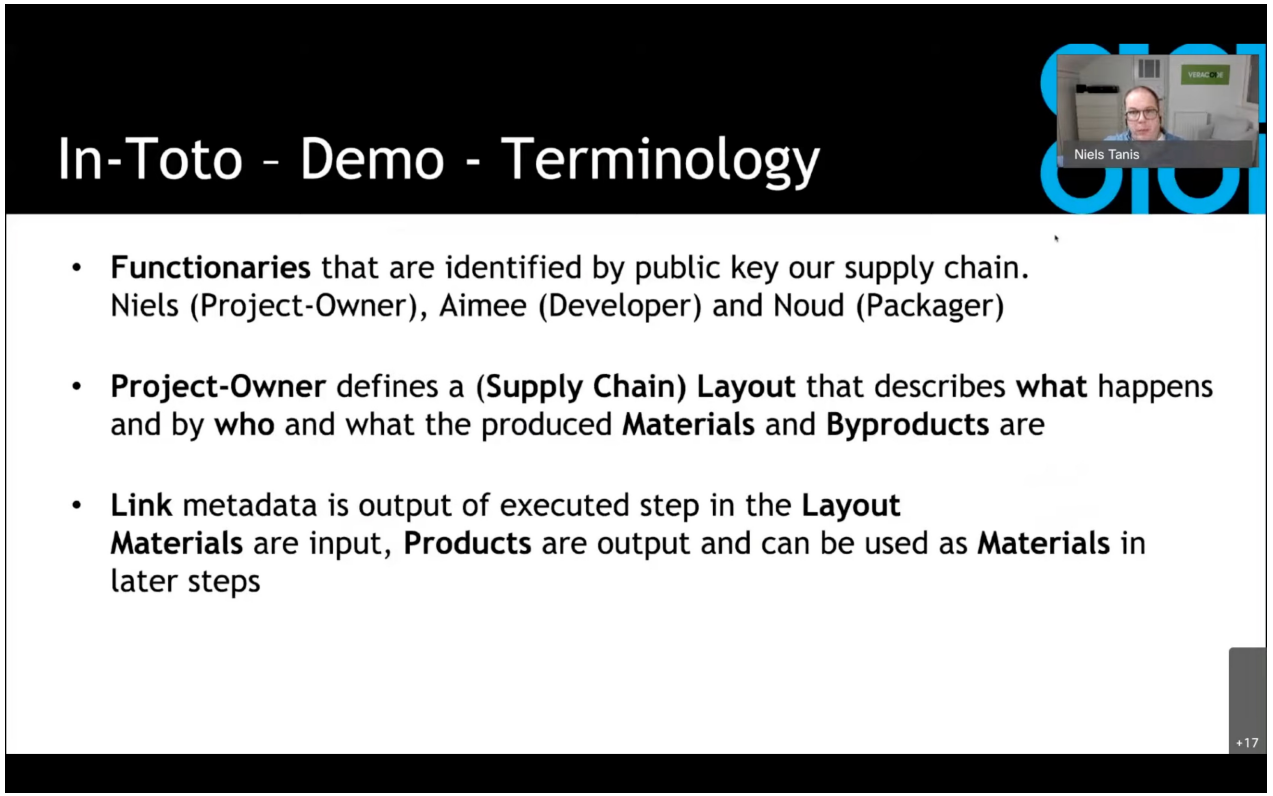
Microsoft SBOM Tool



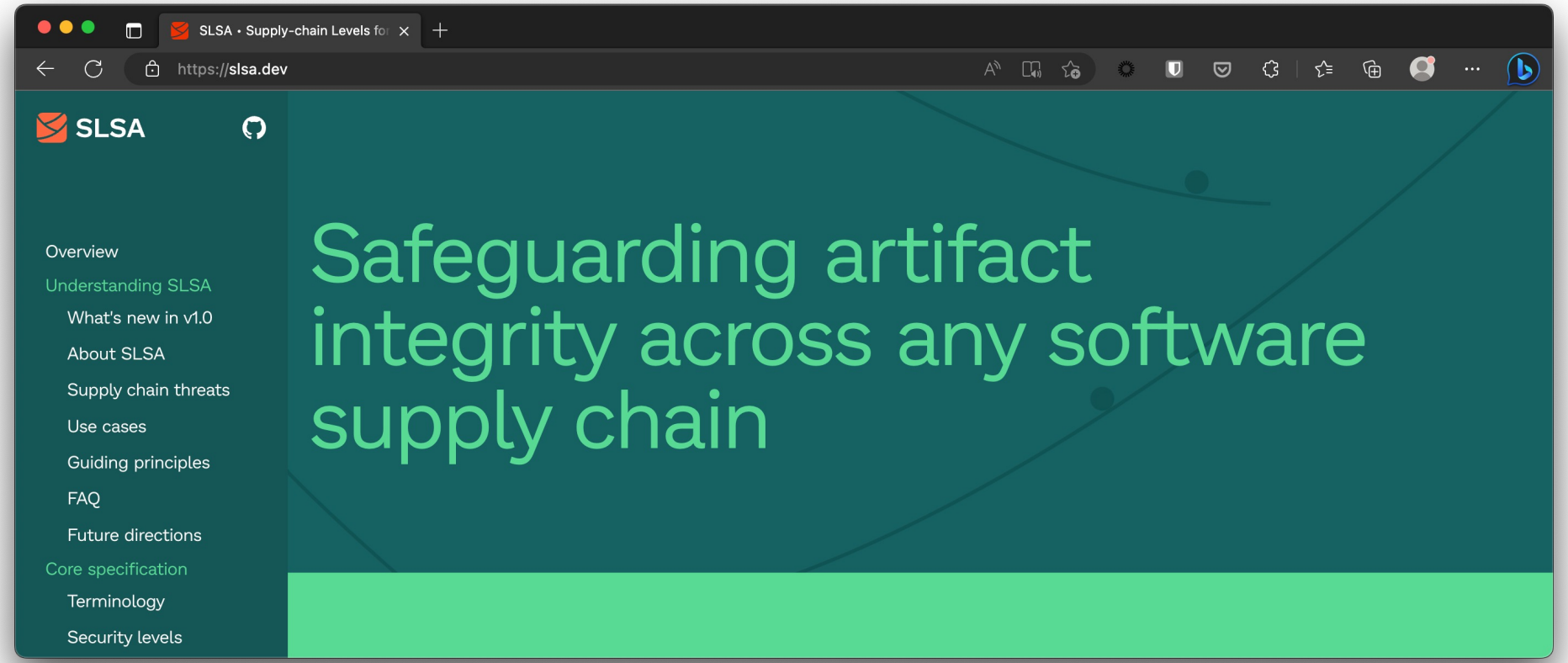
In-toto


in-toto | A framework to secure
https://in-toto.io
in-toto
About
Community
Get started
Learn more
GitHub
in-toto
A framework to secure the integrity of software supply chains

In-Toto - Demo


In-Toto - Demo - Terminology
Niels Tanis
• **Functionaries** that are identified by public key our supply chain. Niels (Project-Owner), Aimee (Developer) and Noud (Packager)
• **Project-Owner** defines a (**Supply Chain**) **Layout** that describes **what** happens and by **who** and what the produced **Materials** and **Byproducts** are
• **Link** metadata is output of executed step in the **Layout** **Materials** are input, **Products** are output and can be used as **Materials** in later steps
+17

Google SLSA


SLSA • Supply-chain Levels fo
https://slsa.dev
SLSA
Overview
Understanding SLSA
What's new in v1.0
About SLSA
Supply chain threats
Use cases
Guiding principles
FAQ
Future directions
Core specification
Terminology
Security levels
Safeguarding artifact integrity across any software supply chain

Google SLSA Levels

Level	Description	Example
1	Documentation of the build process	Unsigned provenance
2	Tamper resistance of the build service	Hosted source/build, signed provenance
3	Extra resistance to specific threats	Security controls on host, non-falsifiable provenance
4	Highest levels of confidence and trust	Two-party review + hermetic builds

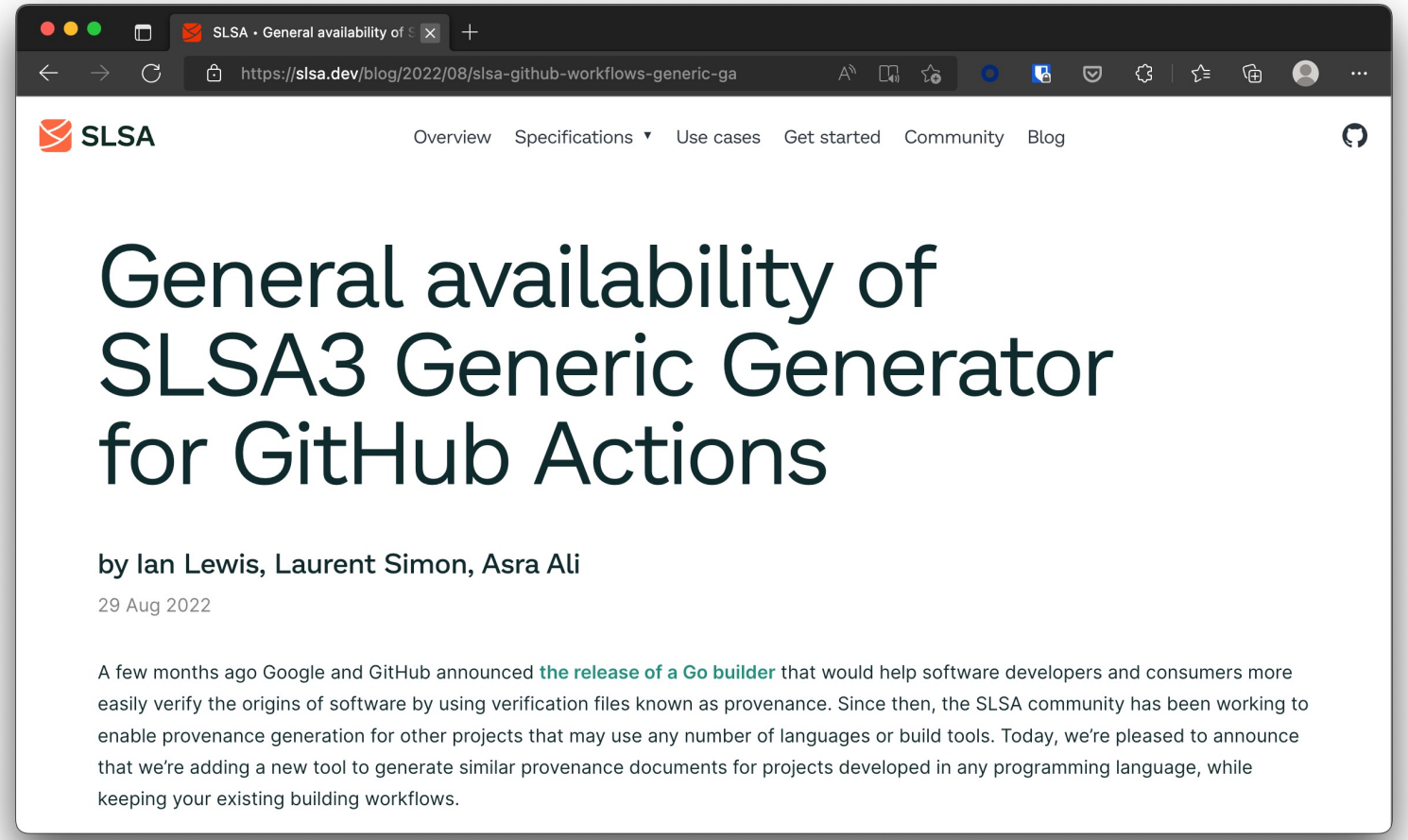
SLSA GitHub Action

- Released April 2022

- SLSA level 2 provenance generator in GitHub Action
- SLSA level 3+ provenance generator for Go binaries

- GitHub Hosted Runner
- Uses SigStore to do keyless signing with GitHub OIDC
- Verifier included

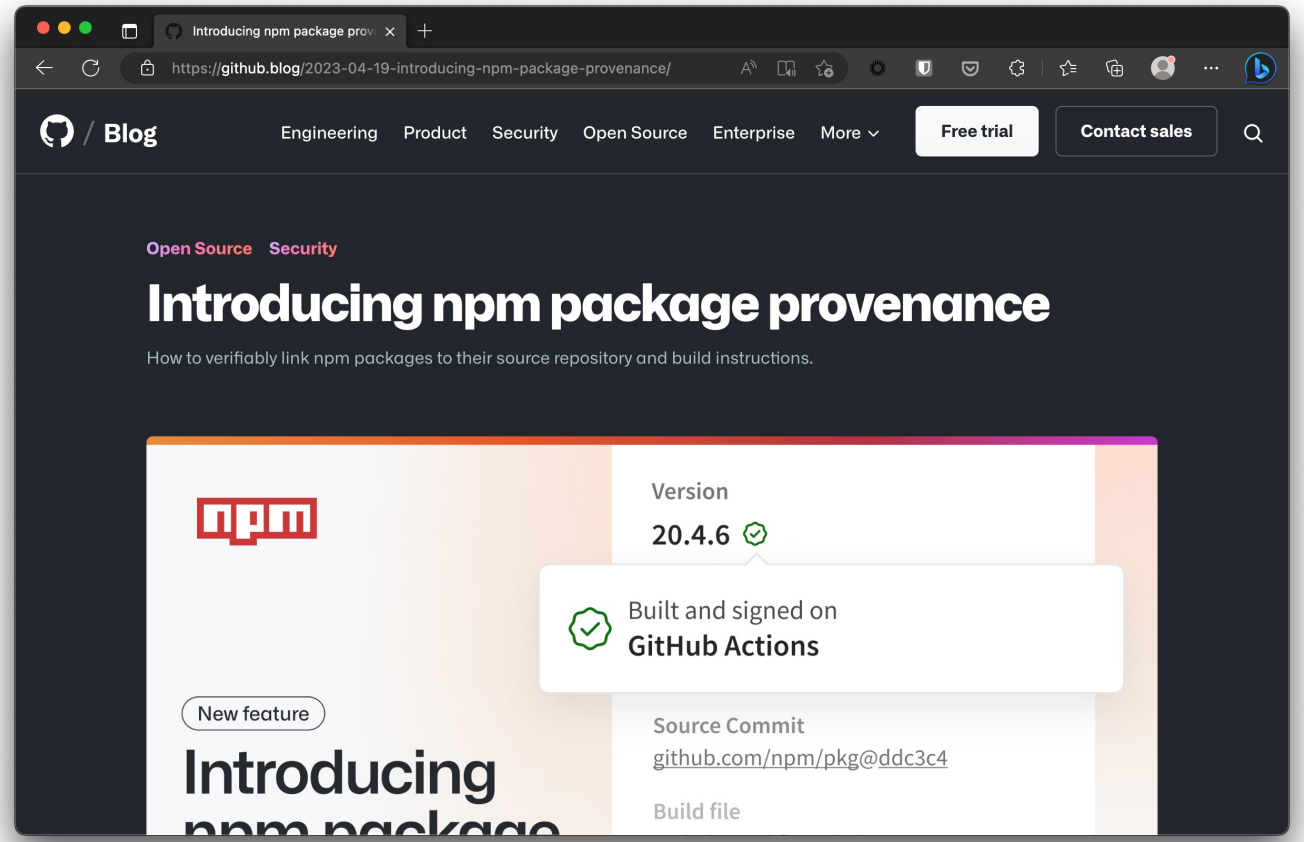
SLSA3 Generator GitHub Actions



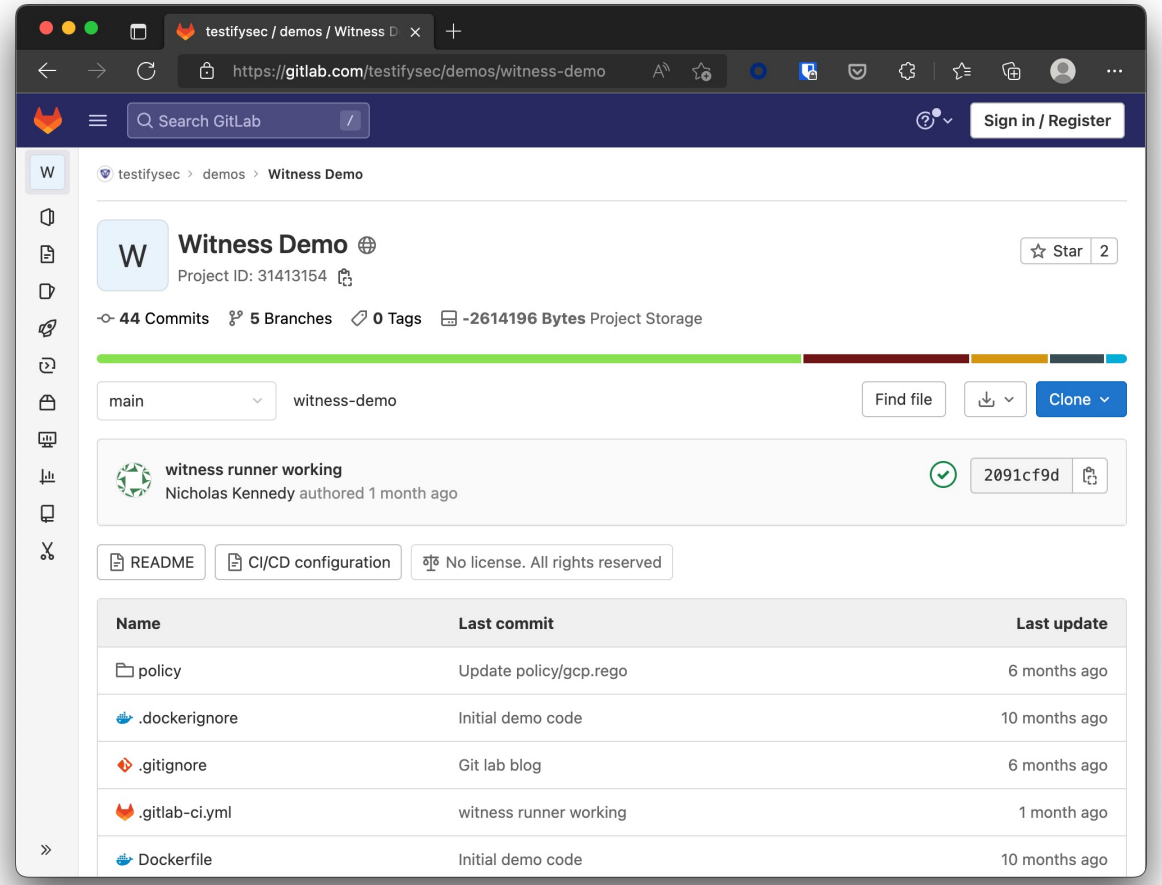
MyAwesomePDFComponent Demo

- A NuGet Package build in GitHub Actions
- CycloneDX SBOM
- Sigstore Keyless Signing
- SLSA Level 3 Provenance
- SBOM data, now what?

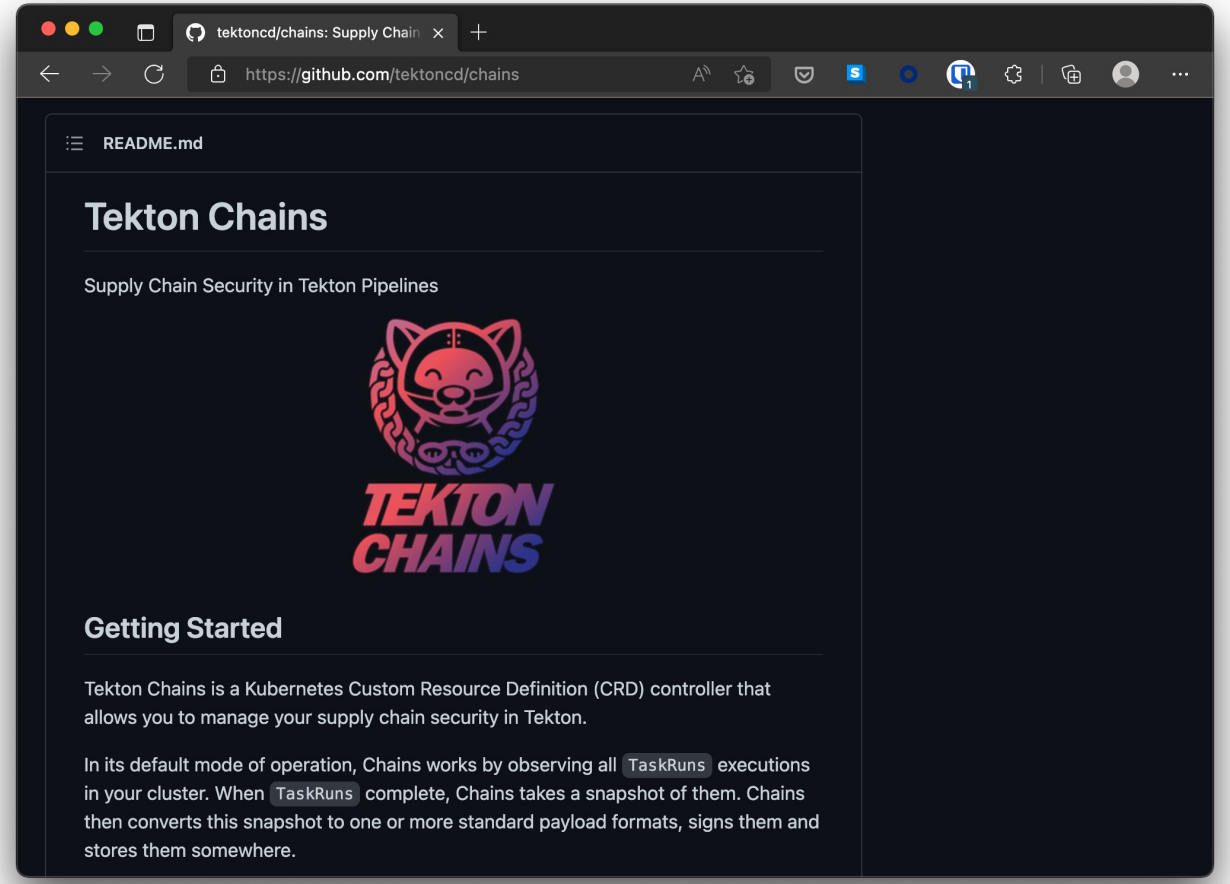
NPM Package SLSA Provenance



Witness & GitLab Attestator



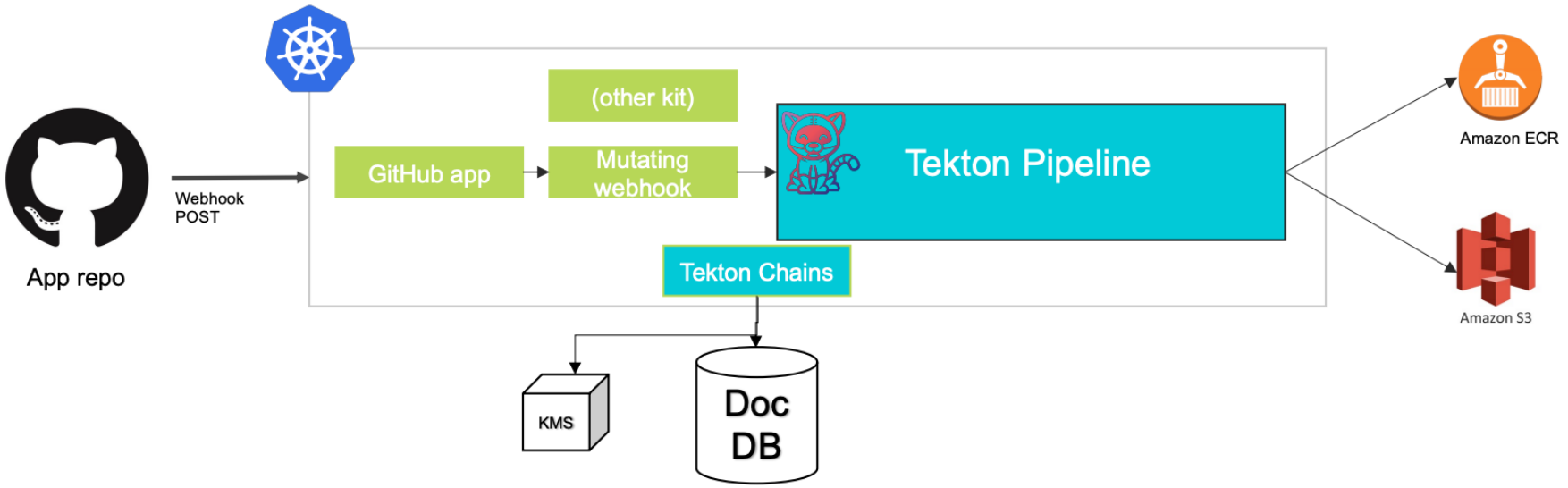
Open Shift - Tekton - Tekton Chains



@nielstanis@infosec.exchange

SolarWinds Project Trebuchet

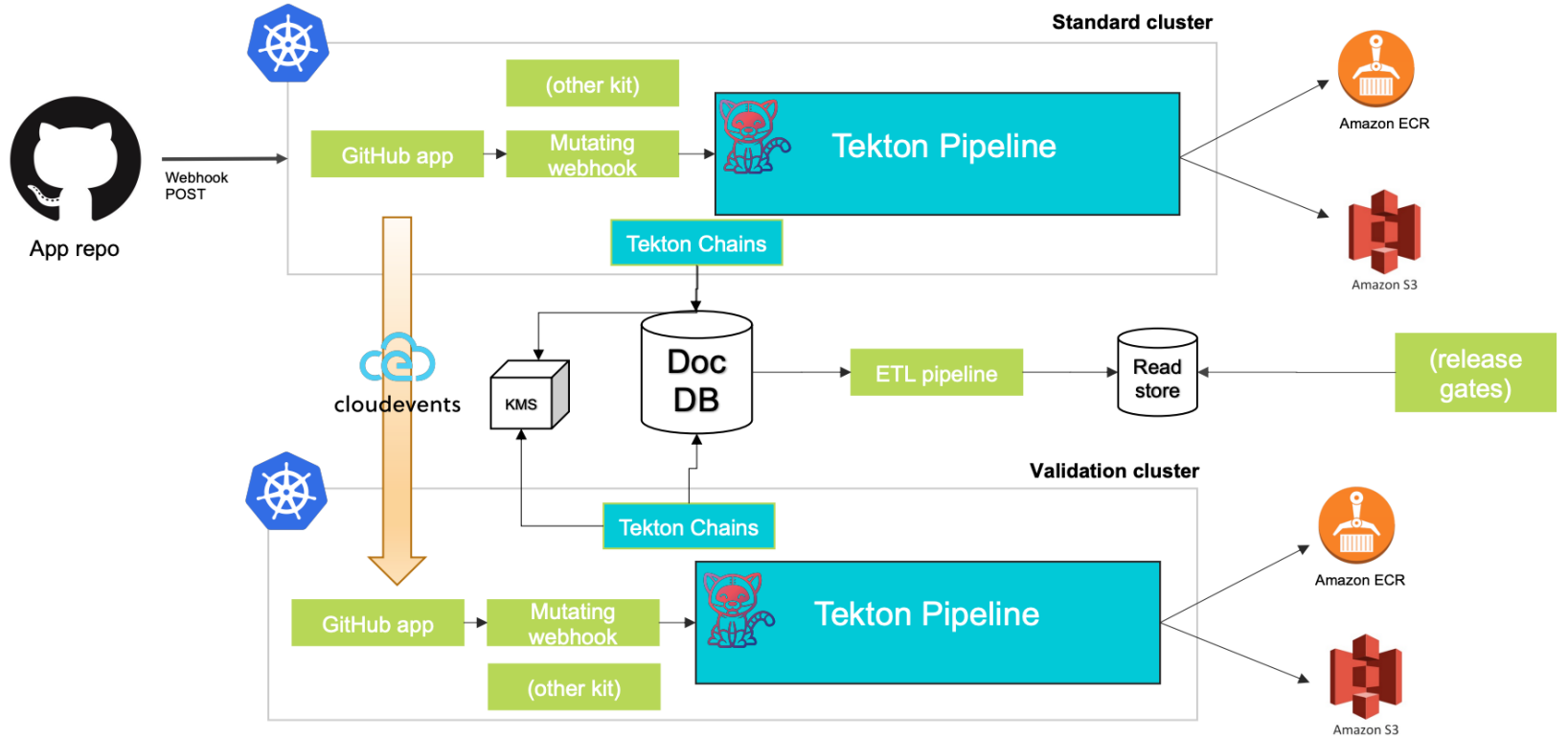
Pipeline With Attestations



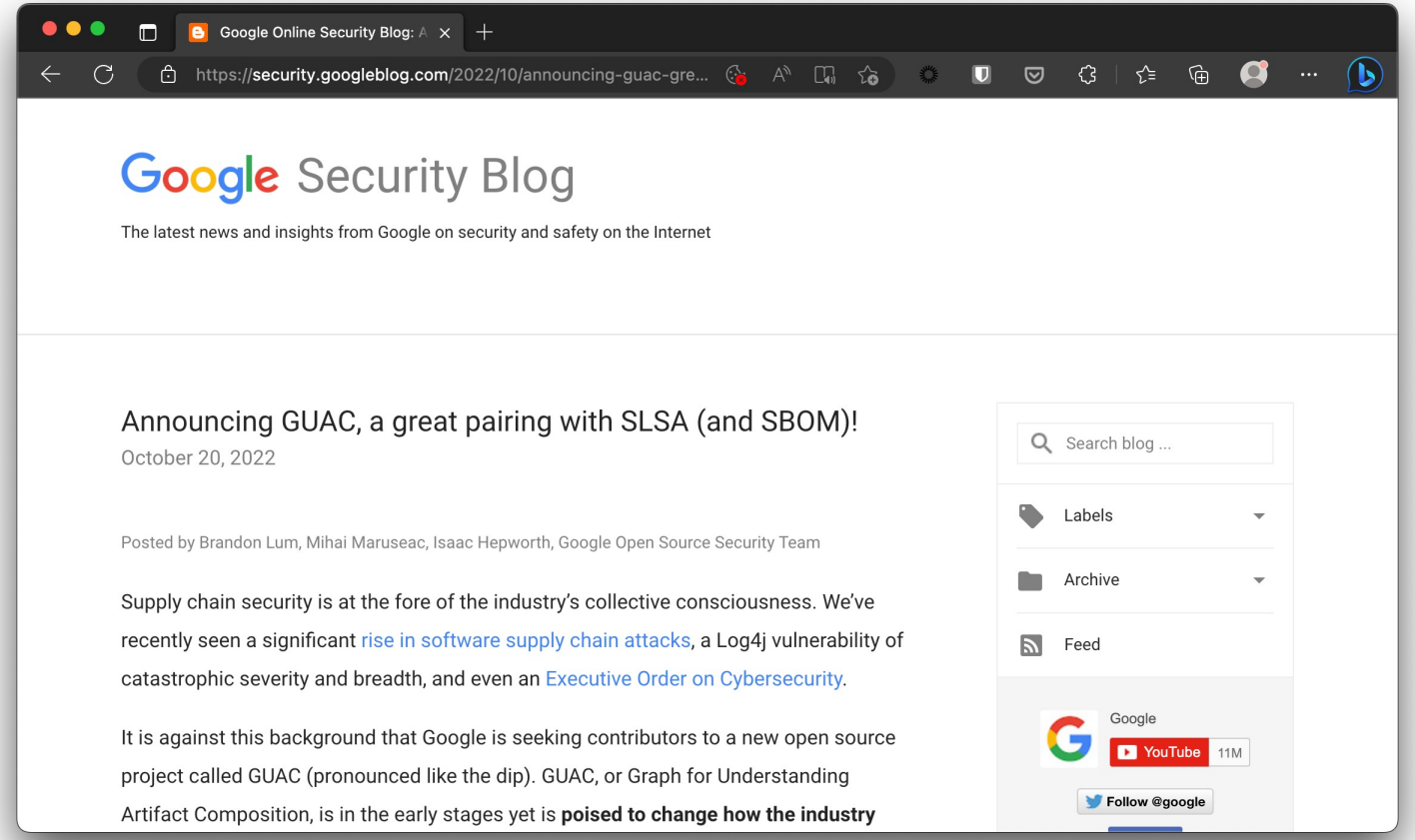
@nielstanis@infosec.exchange

SolarWinds Project Trebuchet

Reading Results



Google Graph for Understanding Artifact Composition (GUAC)



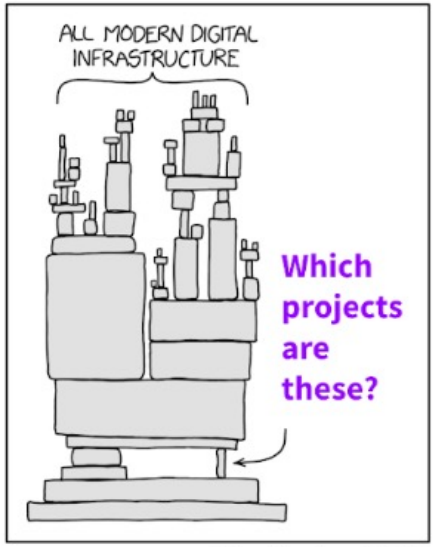
CODE EUROPE

@nielstanis@infosec.exchange

Graph for Understanding Artifact Composition

Proactive

How do I prevent large scale supply chain compromises?



https://xkcd.com/2347/

Preventive

Have I taken the right safeguards?

When deciding to use and deploy software, are there sufficient security checks and approvals?

SLSA

aqua trivy

grype

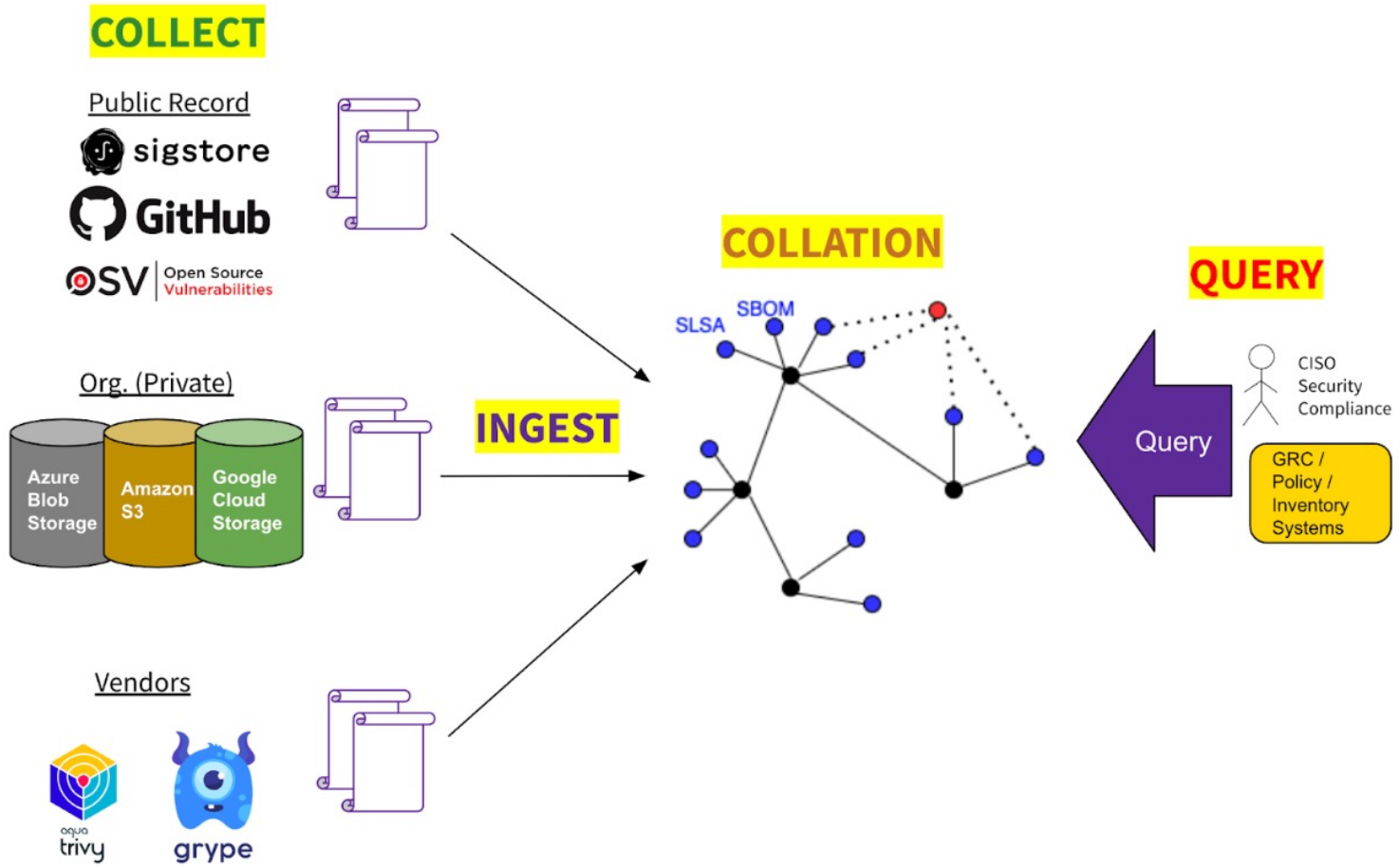
Reactive

HOW AM I AFFECTED???

A vulnerability or supply chain compromise is discovered!

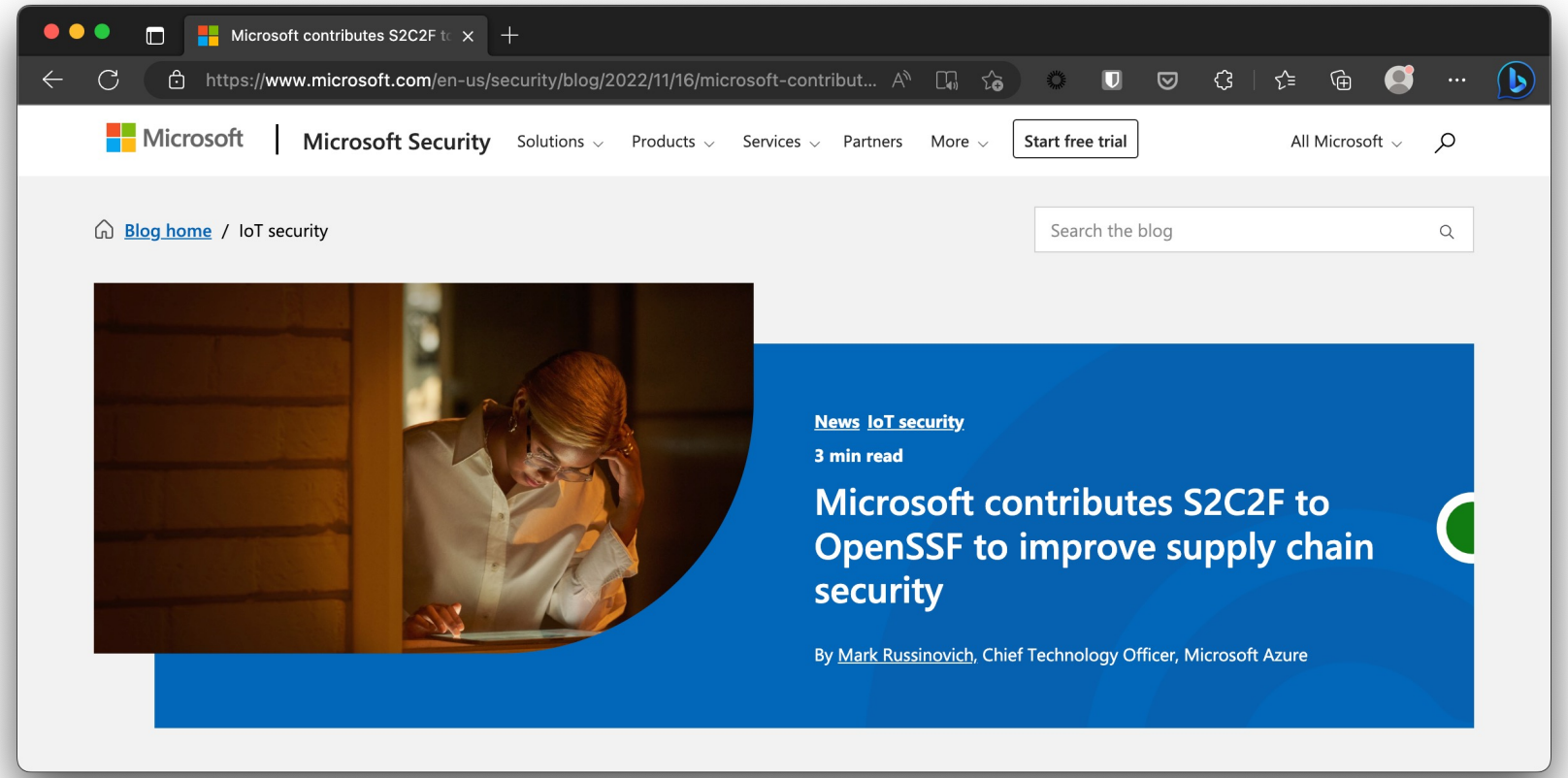
+ Codecov, Solarwinds compromises

Graph for Understanding Artifact Composition



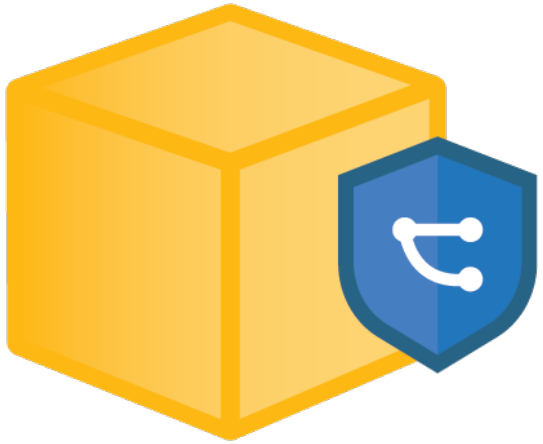
@nielstanis@infosec.exchange

Secure Supply Chain Consumption Framework (S2C2F)



Secure Supply Chain Consumption Framework (S2C2F)

- Provide a strong OSS governance program
- Improve the Mean Time To Remediate (MTTR) for resolving known vulnerabilities in OSS
- Prevent the consumption of compromised and malicious OSS packages



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 • Local copy of artifact • Scan with known vulns • Scan for software licenses • Inventory OSS • Manual OSS updates	• Scan for end life • Have an incident response plan • Auto OSS updates • Alert on vulns at PR time • Audit that consumption is through the approved ingestion method • Validate integrity of OSS • Secure package source file configuration	• Deny list capability • Clone OSS source • Scan for malware • Proactive security reviews • Enforce OSS provenance • Enforce consumption from curated feed	• Validate the SBOMs of OSS consumed • Rebuild OSS on trusted infrastructure • Digitally sign rebuilt OSS • Generate SBOM for rebuilt OSS • Digitally sign protected SBOMs • Implement fixes

Conclusion

- It’s not how it’s more a matter of when!
- Be aware of your used software supply chain(s).
- Know what you’re using and pulling into projects.
- Integrate security into your software lifecycle.

Conclusion

- Start working on creating SBOM's and see how SLSA can fit into your process
- Look how S2C2F can help you on your projects where you consume open-source
- Work smart with SBOM/provenance outputs!

Questions?

- https://github.com/nielstanis/codeeurope2023-supplychain
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- https://blog.fennec.dev
- Dziękuję! Thank you!