NDC { London }

Securing your .NET application software supply-chain the practical approach! (workshop)

Niels Tanis

Who am I?

- Niels Tanis
- Sr. Principal Security Researcher @ Veracode
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - Research on static analysis for .NET apps

@nielstanis@infosec.exchange

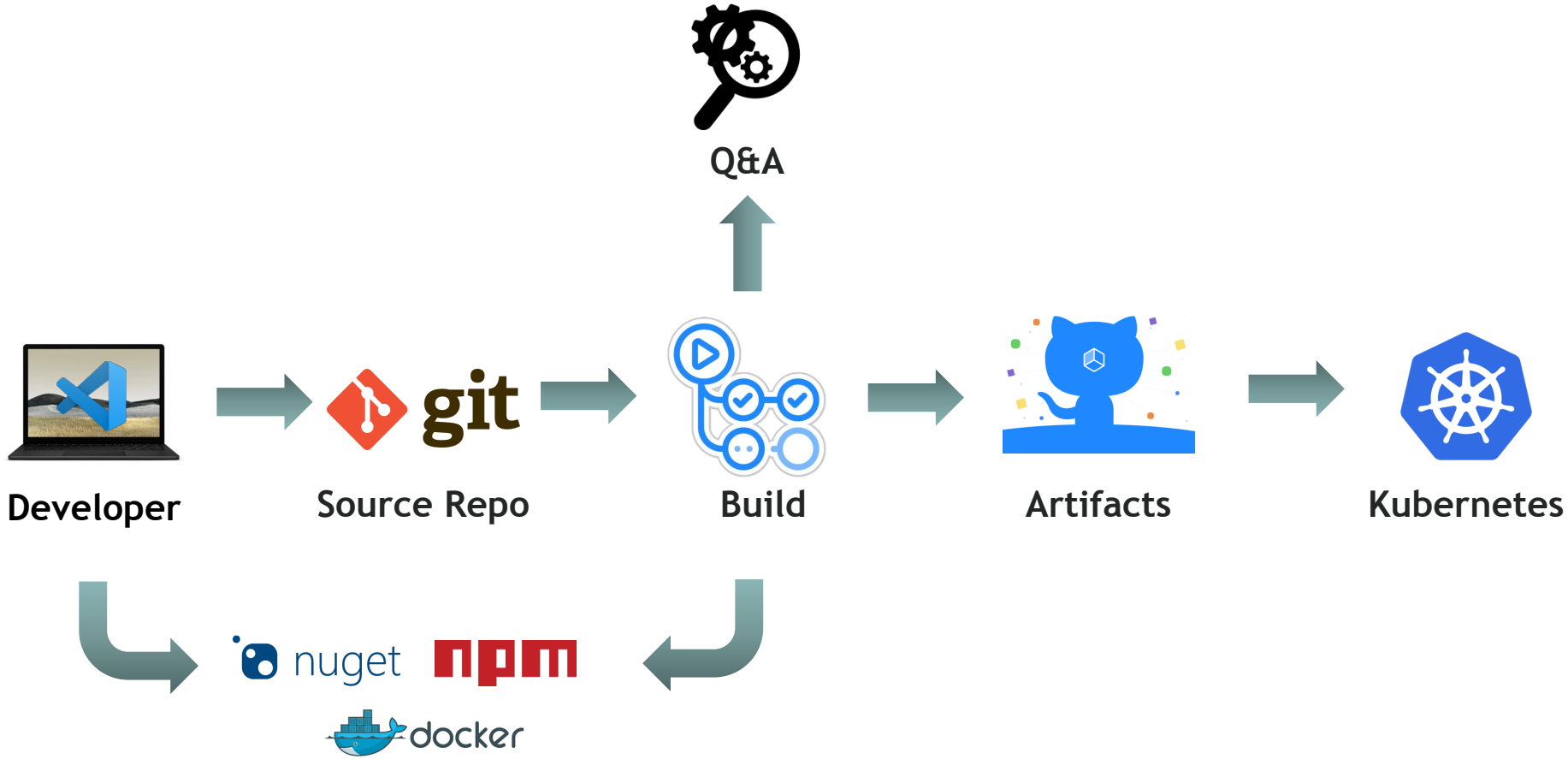
Agenda

@nielstanis@infosec.exchange

What is a Supply Chain?

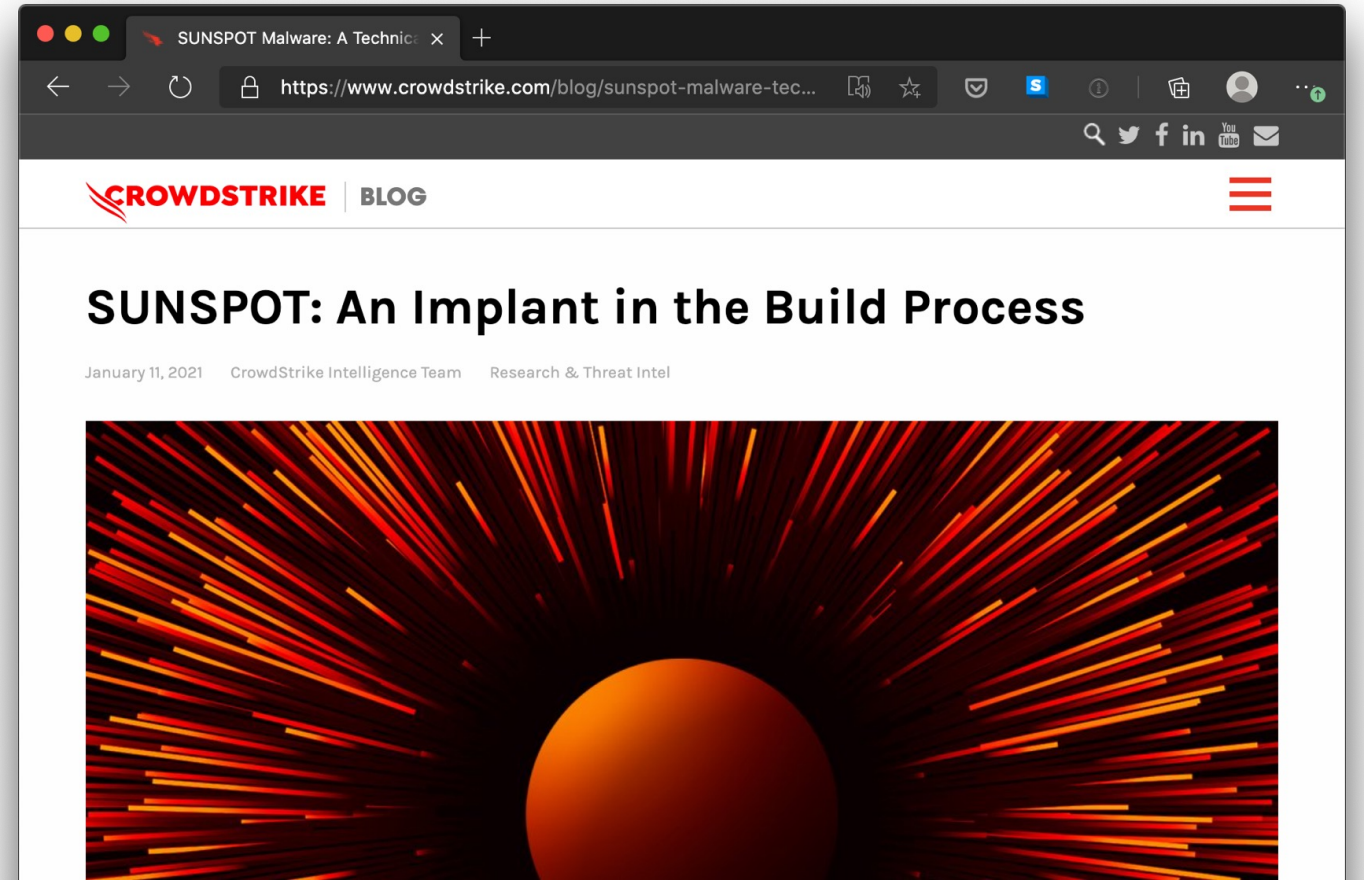
@nielstanis@infosec.exchange

Software Supply Chain



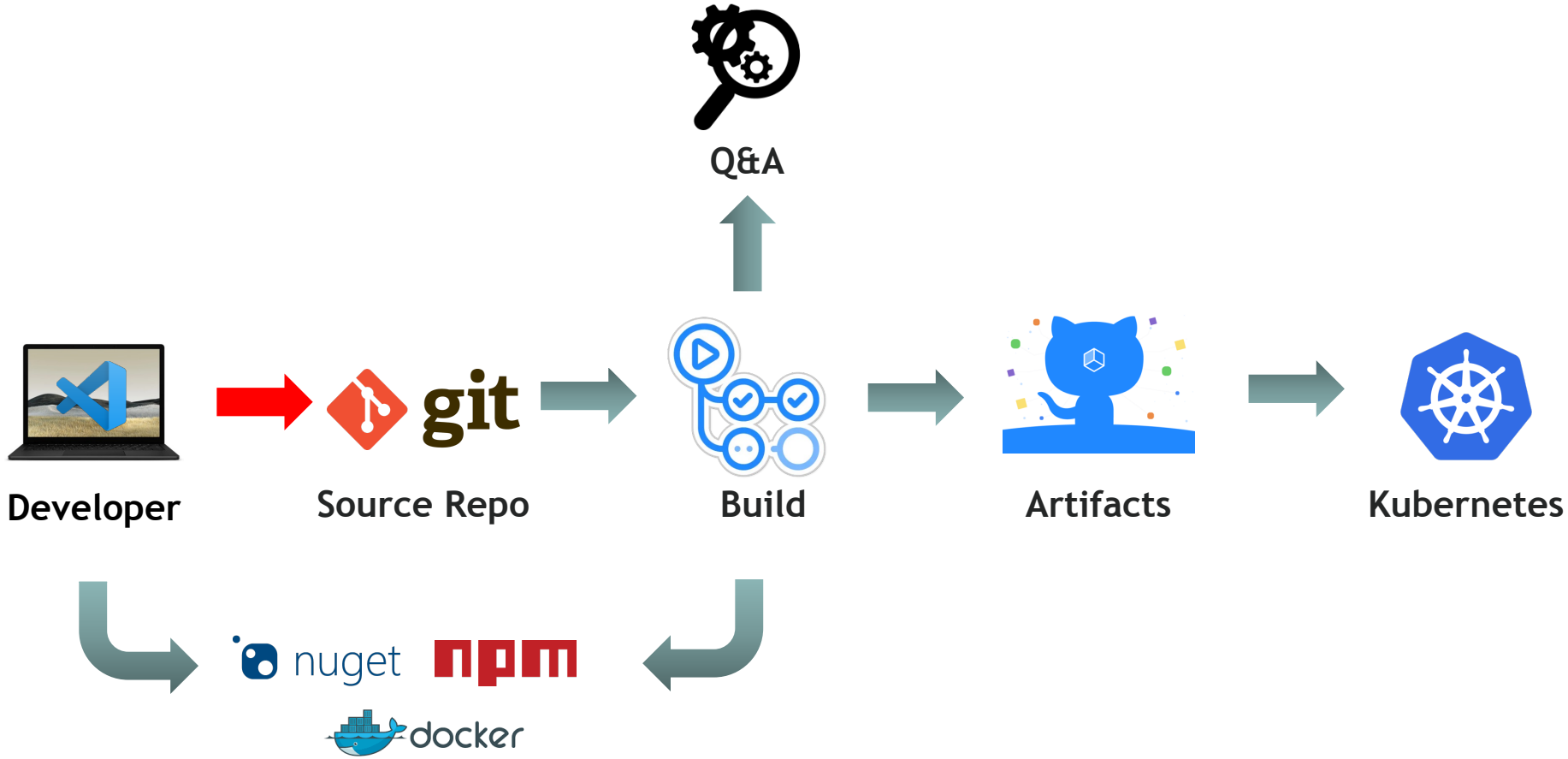
@nielstanis@infosec.exchange

SolarWinds SunSpot



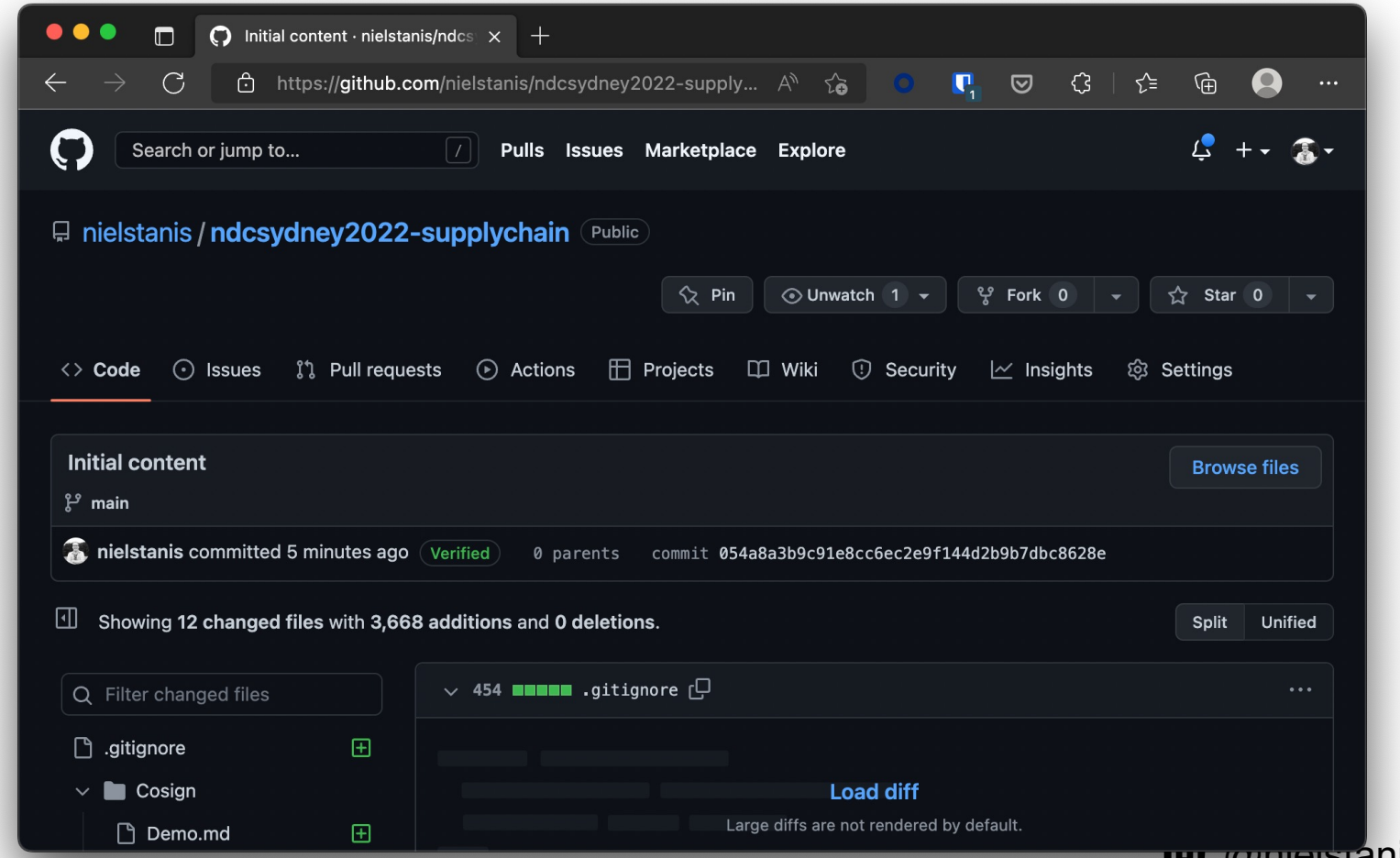
@nielstanis@infosec.exchange

Software Supply Chain



@nielstanis@infosec.exchange

GIT Commit Signing

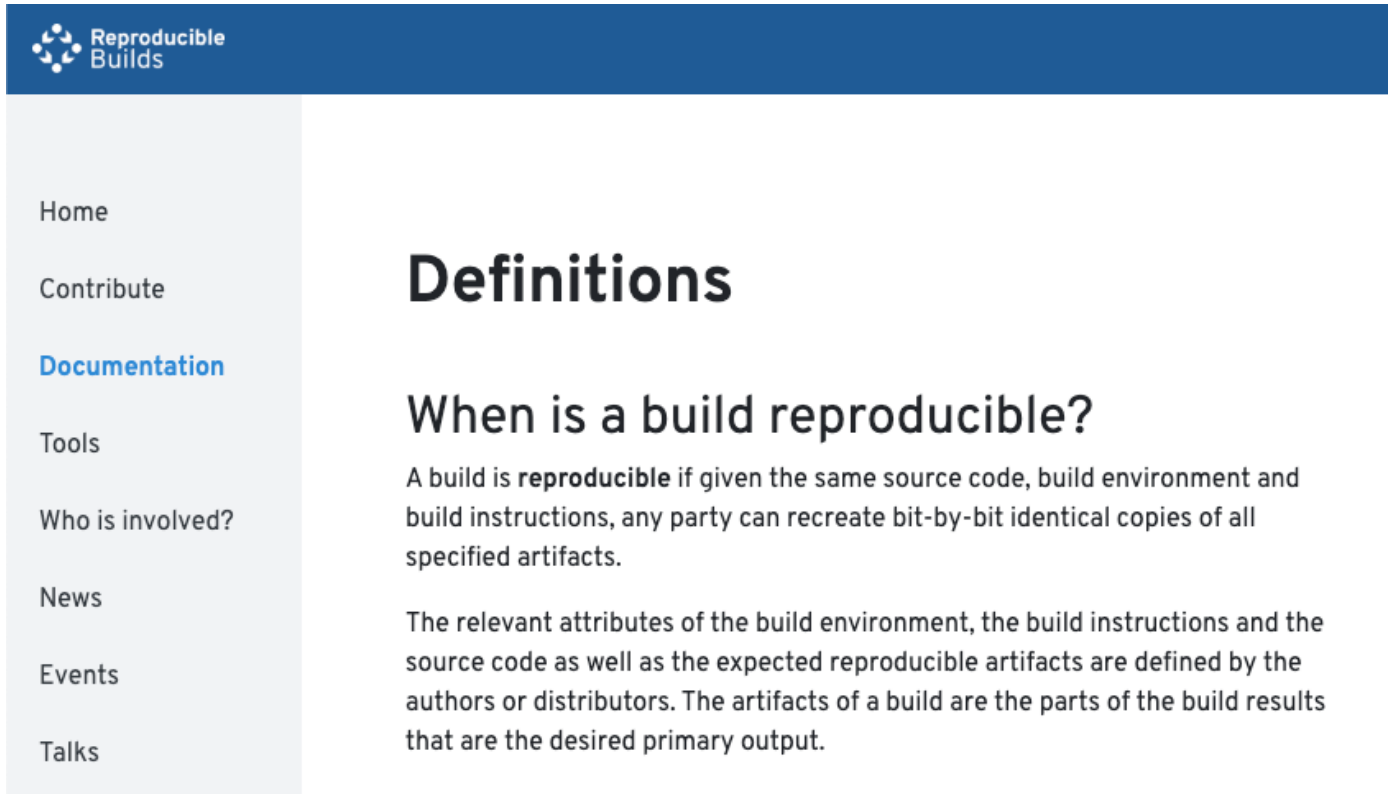


@nielstanis@infosec.exchange

Lab 1

DocGenerator

Reproducible/Deterministic Builds



@nielstanis@infosec.exchange

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’

@nielstanis@infosec.exchange

Lab 2

.NET 7 reproducible

Reproducible/Deterministic Builds

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

- Dotnet.Reproducible.Isolated NuGet Package
  - Hermetic builds

@nielstanis@infosec.exchange

Lab 3

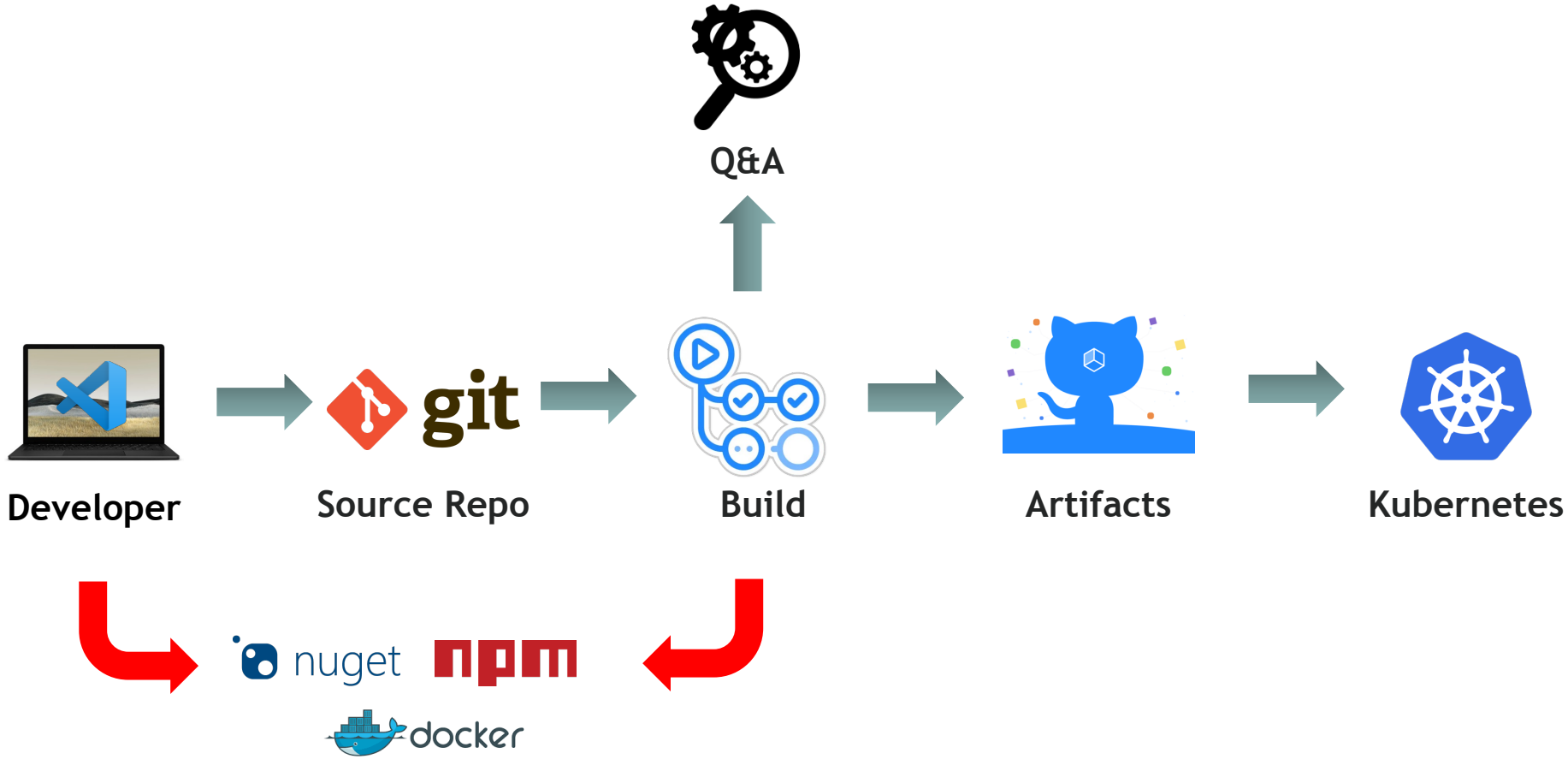
DotNet.Reproducible
NuGet Package

Reproducible Build Validation

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

@nielstanis@infosec.exchange

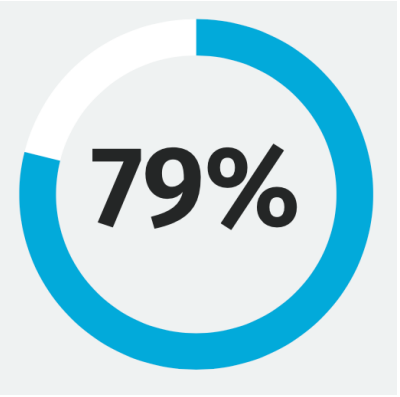
3rd Party Libraries

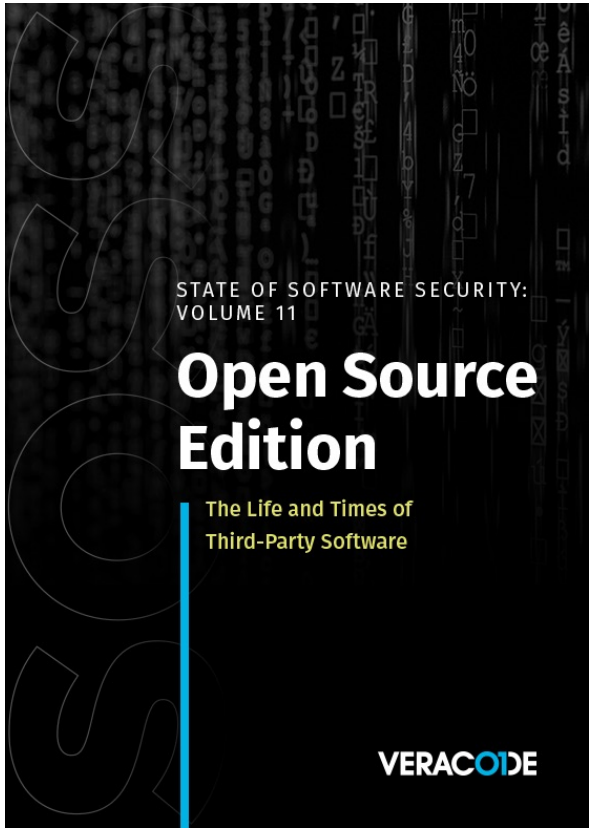


@nielstanis@infosec.exchange

State Of Software Security v11 2021

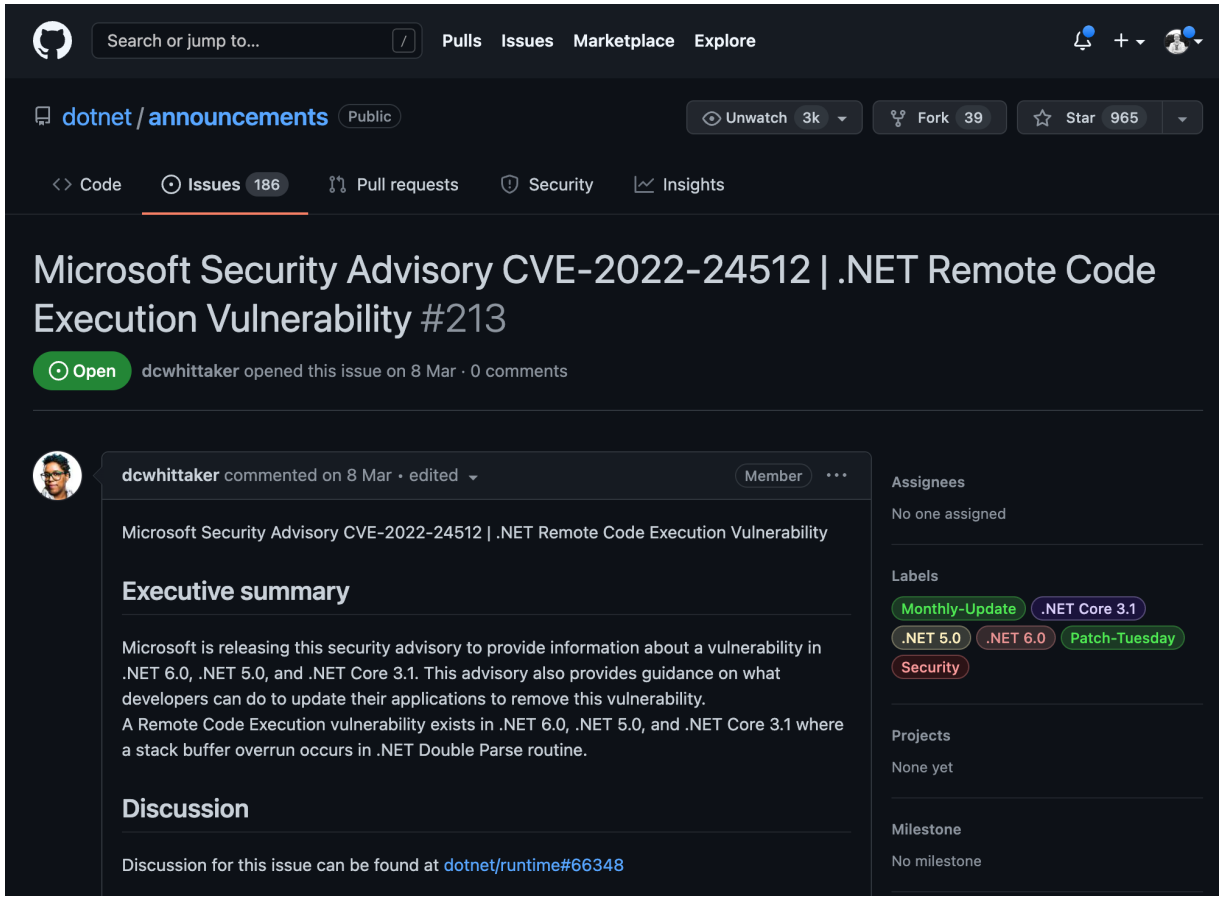
”Despite this dynamic landscape, 79 percent of the time, developers never update third-party libraries after including them in a codebase.”





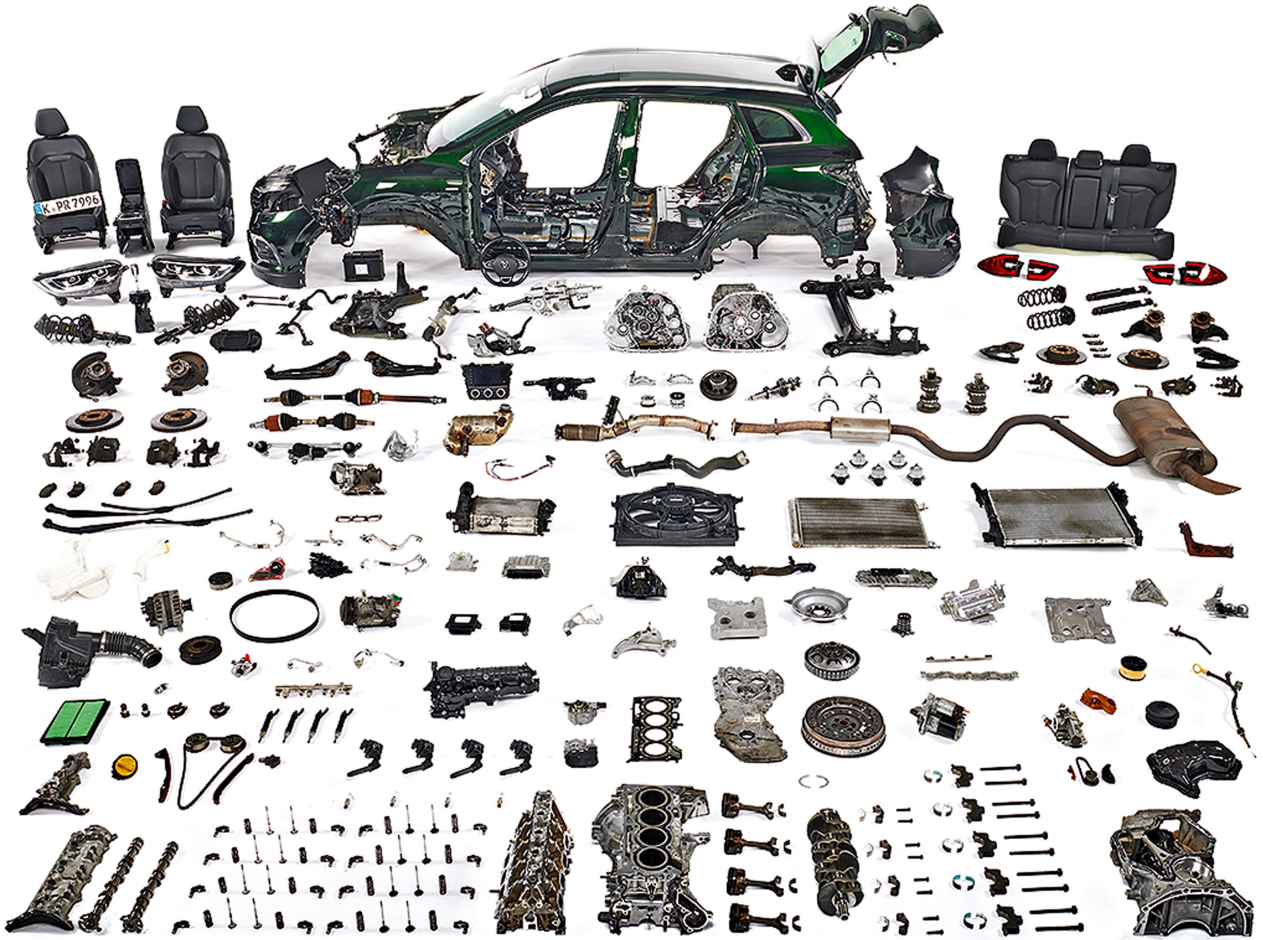
@nielstanis@infosec.exchange

Vulnerabilities in libraries



@nielstanis@infosec.exchange

Automotive Industry



@nicolasnis@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

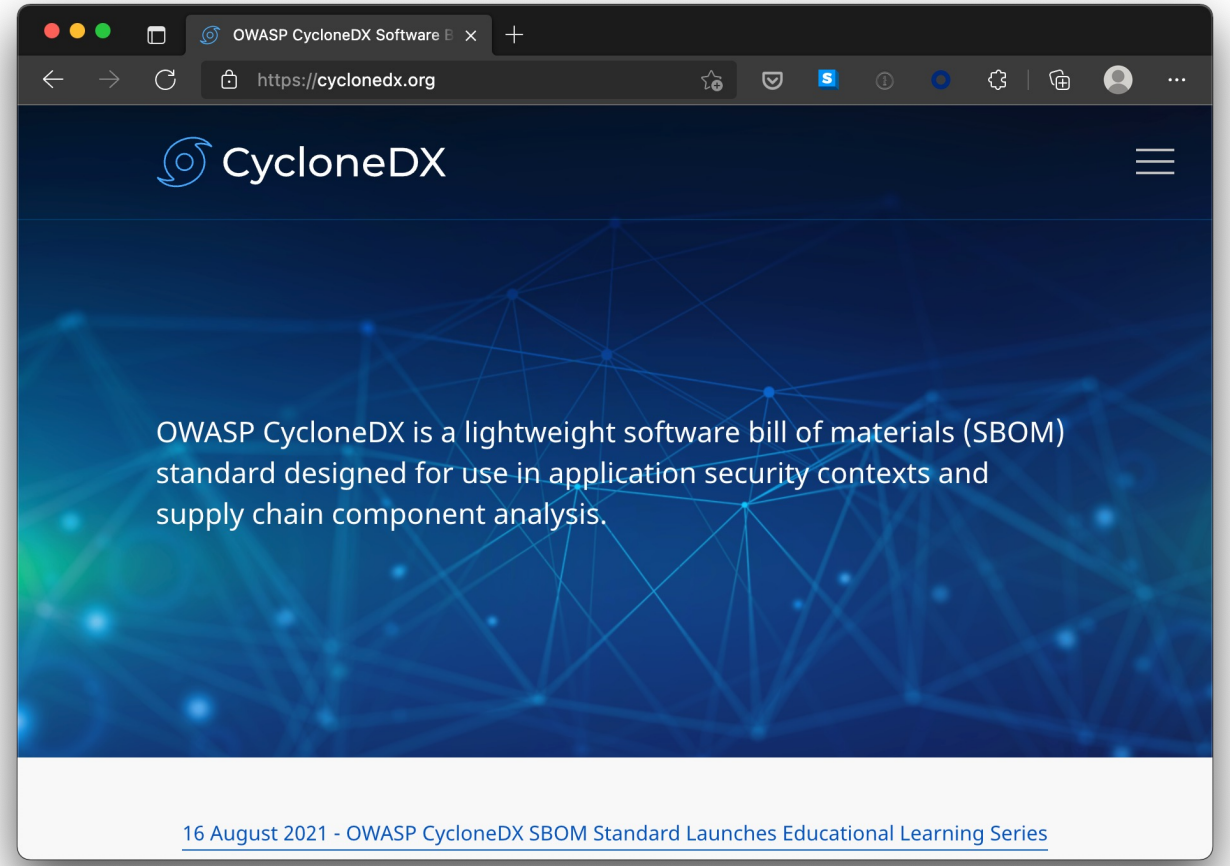
@nielstanis@infosec.exchange

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?

@nielstanis@infosec.exchange

Software Bill of Materials (SBOM)

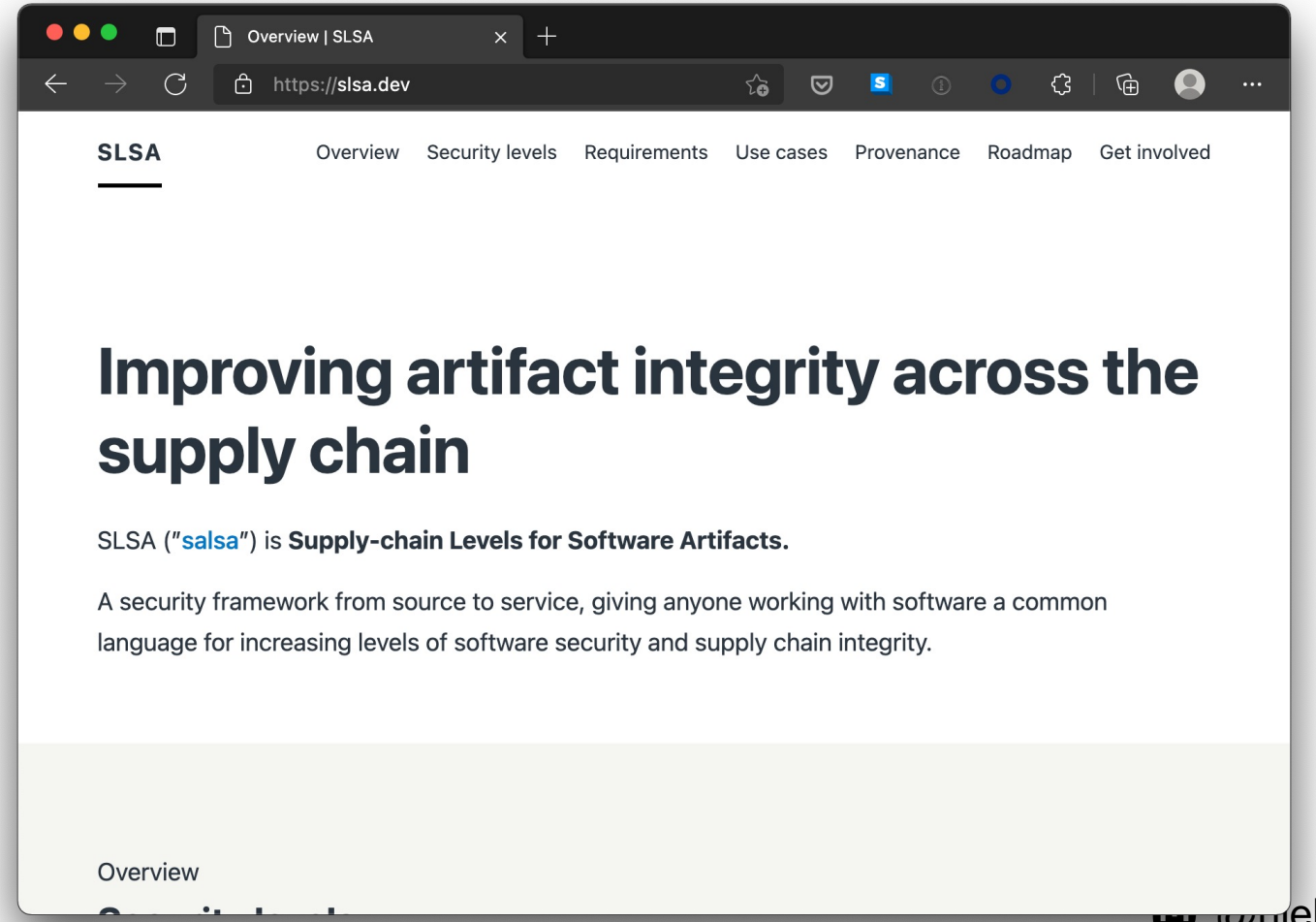


@nielstanis@infosec.exchange

Lab 4

CycloneDX .NET

Google SLSA



@nielstanis@infosec.exchange

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

@nielstanis@infosec.exchange

Google SLSA Levels

1. The build process must be fully scripted/automated and generate provenance.
2. Requires using version control and a hosted build service that generates authenticated provenance.
3. The source and build platforms meet specific standards to guarantee the auditability of the source and the integrity of the provenance respectively.
4. Requires two-person review of all changes and a hermetic, reproducible build process.

@nielstanis@infosec.exchange

SLSA GitHub Action

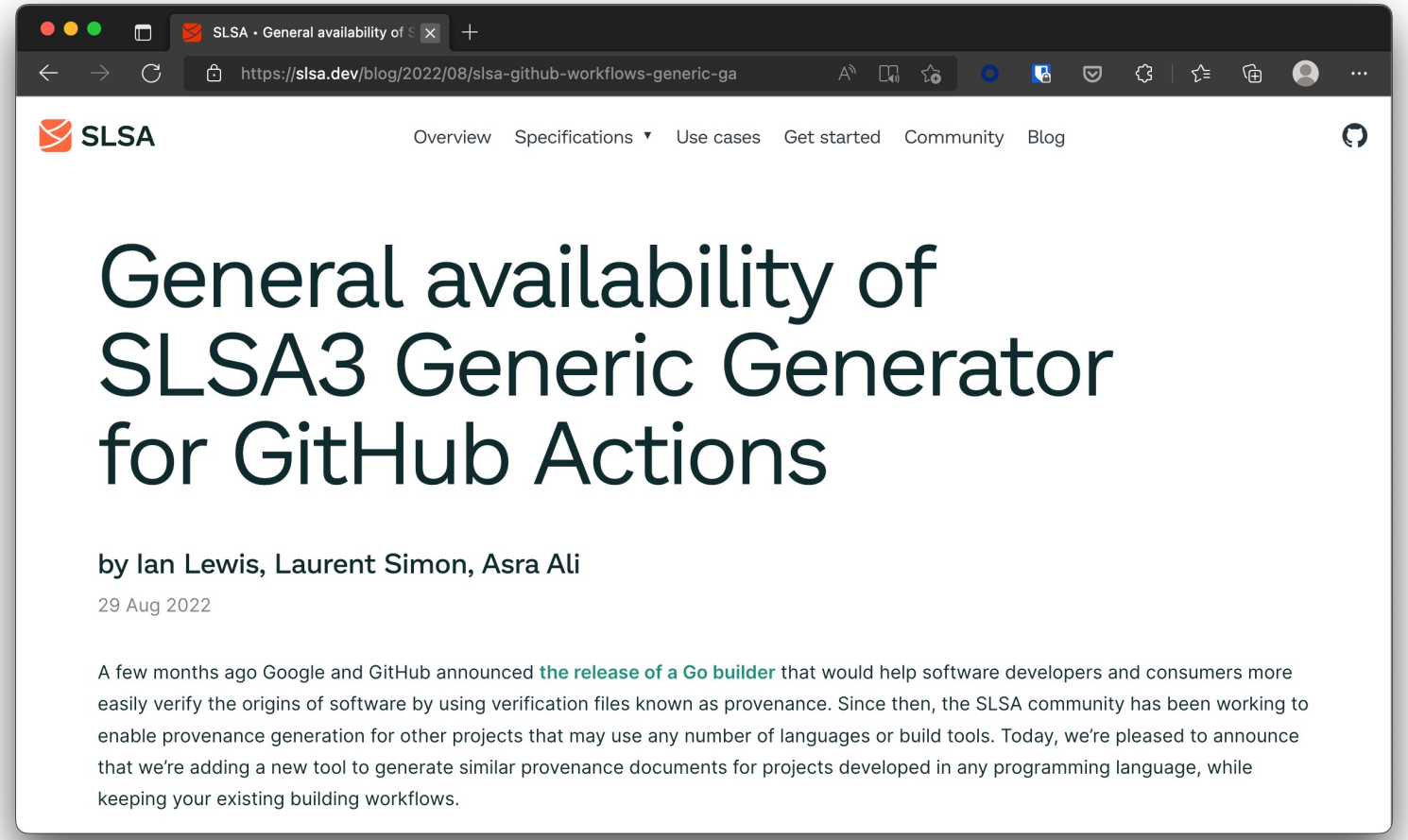
- Released April 2022

- 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

@nielstanis@infosec.exchange

SLSA3 Generator GitHub Actions



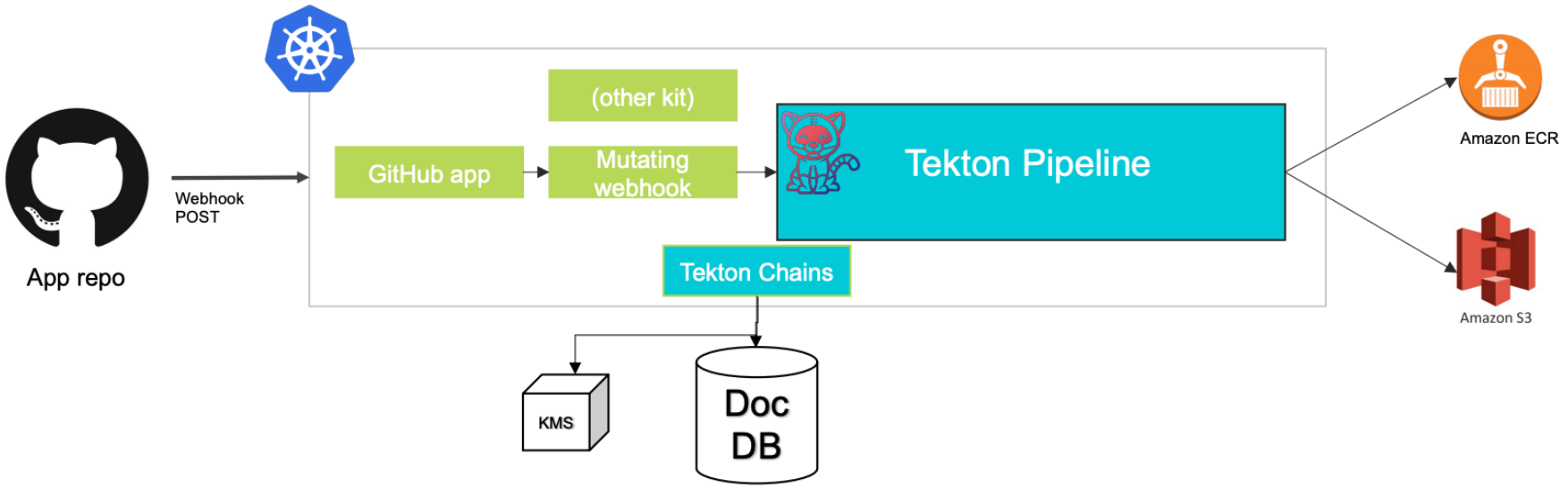
@nielstanis@infosec.exchange

Lab 5

SLSA Level 3 Provenance

SolarWinds Project Trebuchet

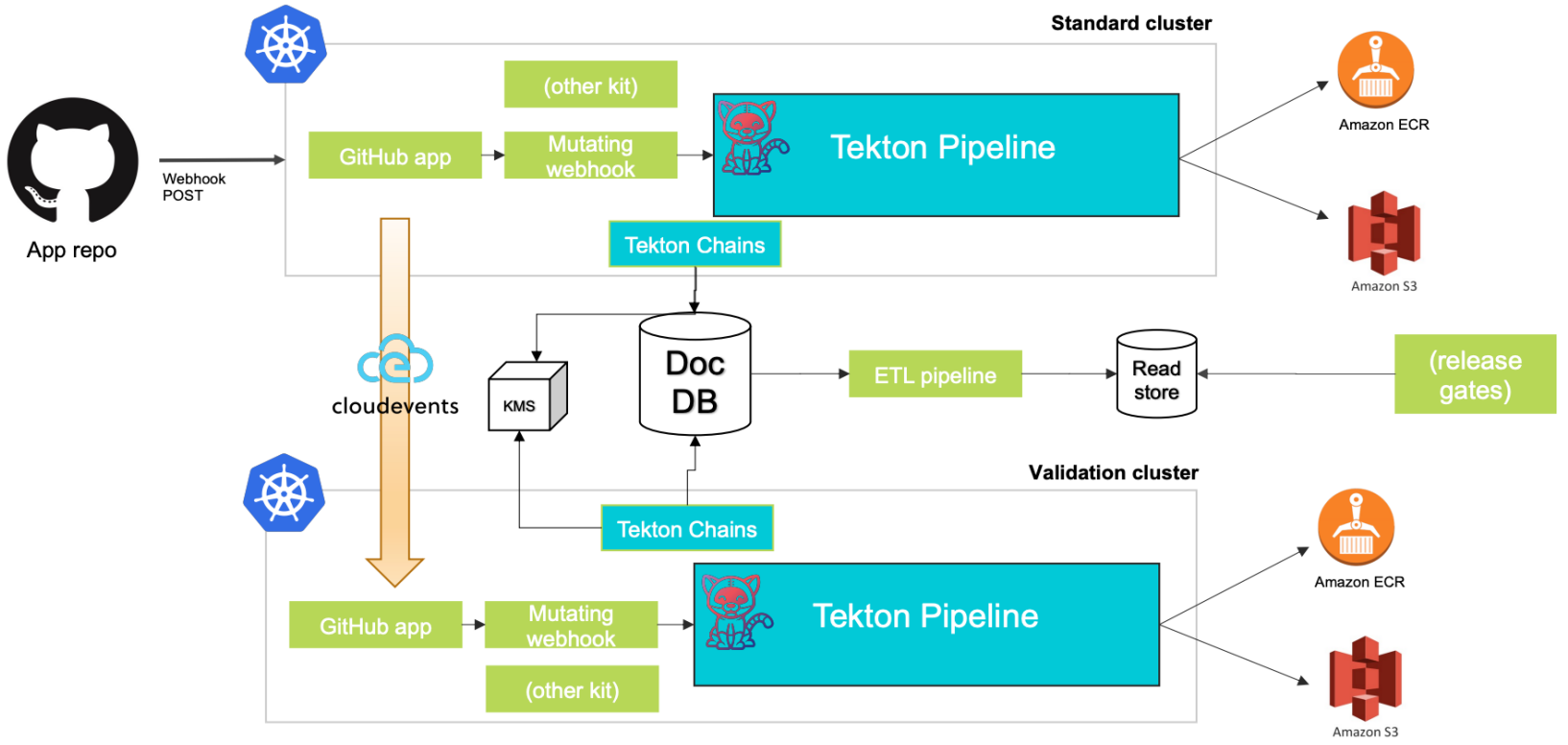
Pipeline With Attestations



@nielstanis@infosec.exchange

SolarWinds Project Trebuchet

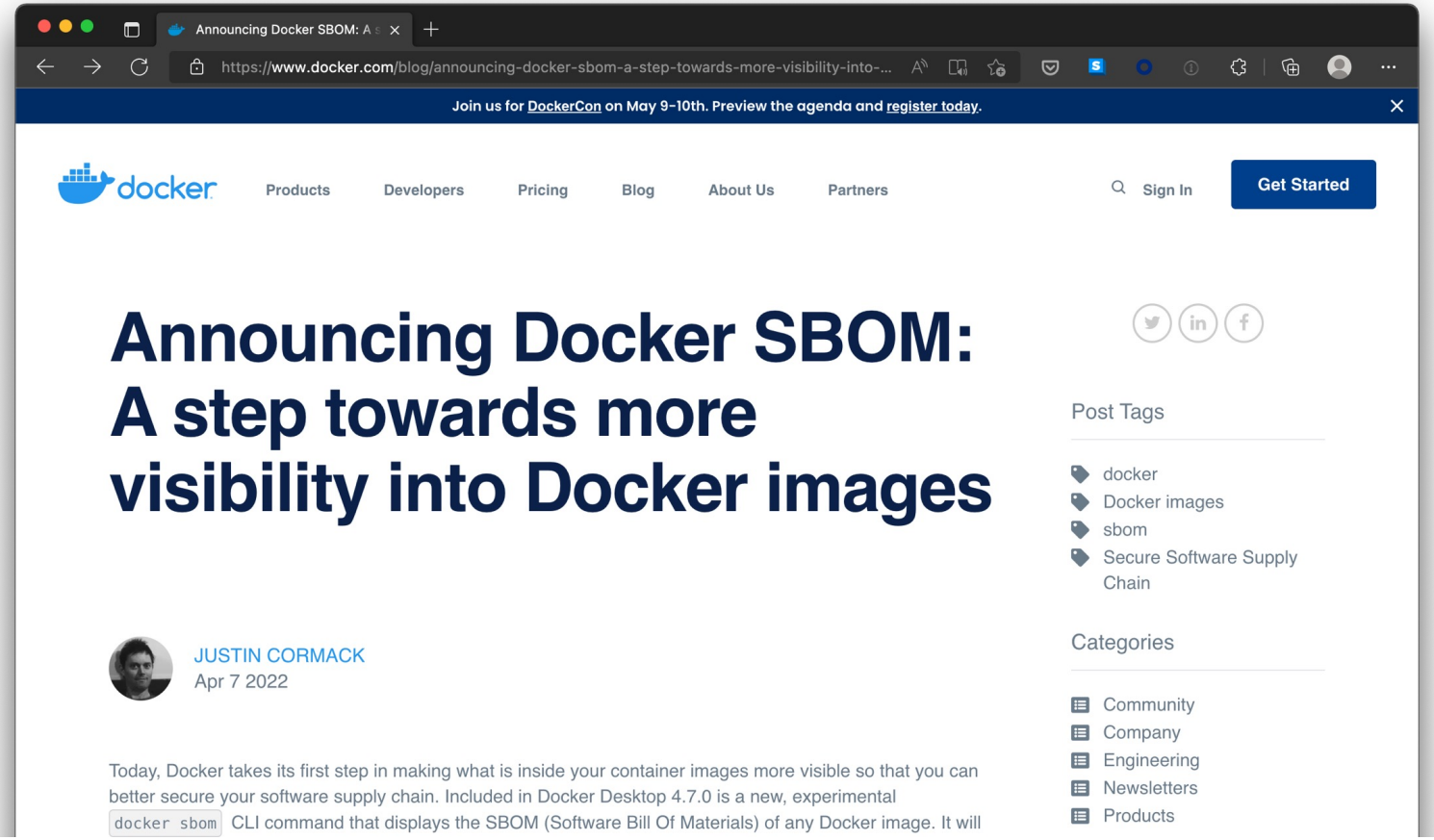
Reading Results



Lab 6

DocGenerator on
ASP.NET Core MVC &
Docker

Docker SBOM

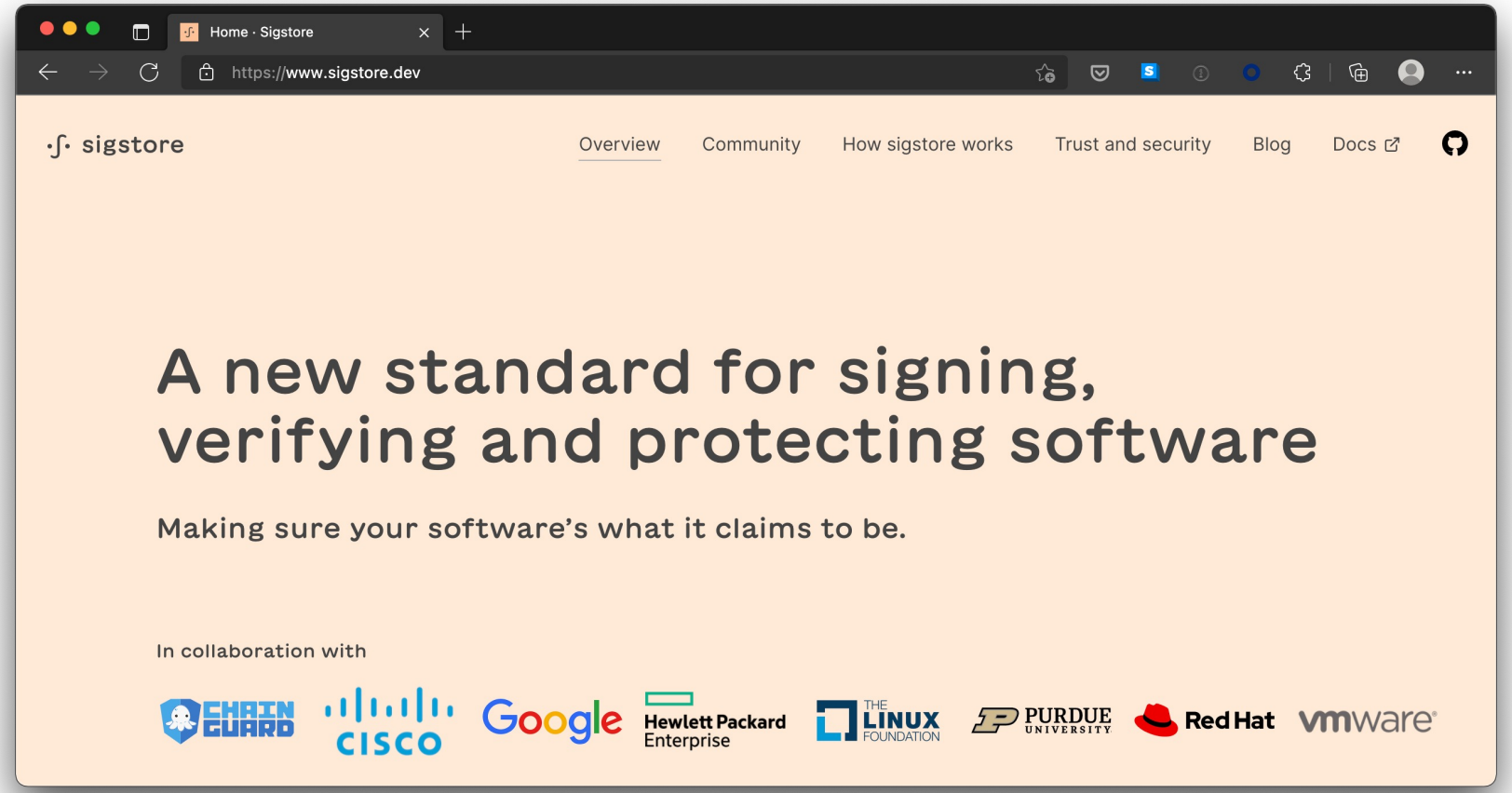


@nielstanis@infosec.exchange

Lab 7

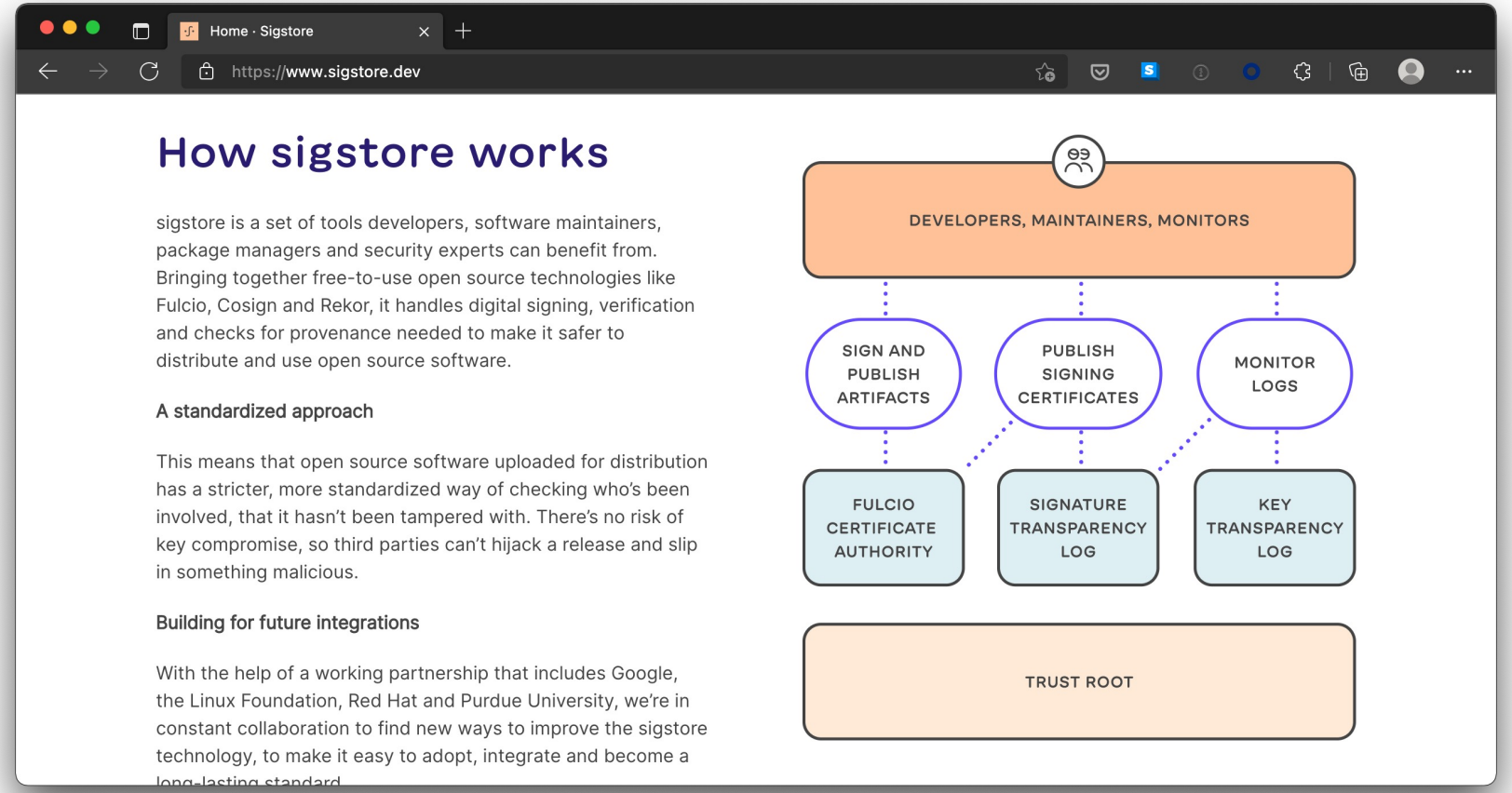
Docker SBOM with Anchor and Syft

Signing artifacts



@nielstanis@infosec.exchange

Signing artifacts



@nielstanis@infosec.exchange

Lab 8

Keyless Signing artifact with cosign

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

@nielstanis@infosec.exchange

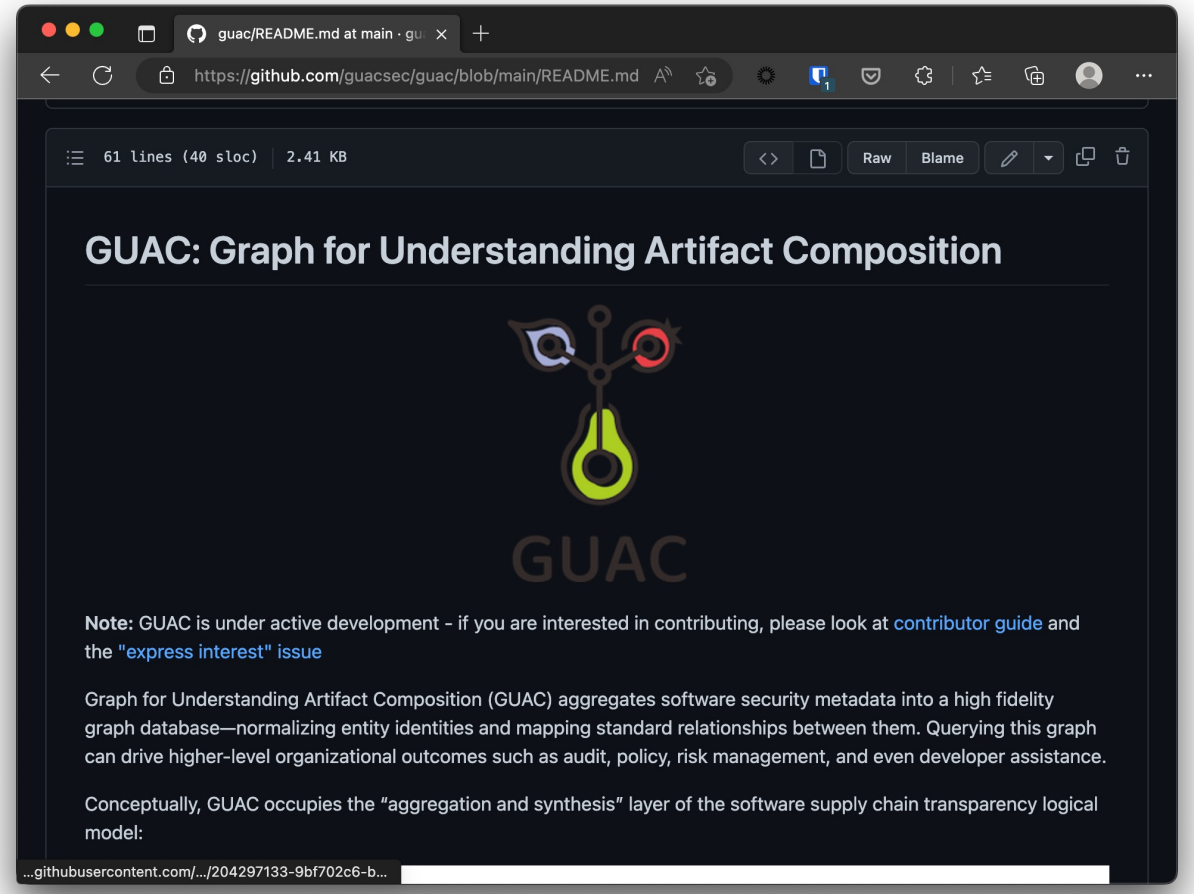
Lab 9

Cosign on Docker in
GitHub Actions

Lab 10

You got HACKED!

GUAC: Graph for Understanding Artifact Composition



@nielstanis@infosec.exchange

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.

@nielstanis@infosec.exchange

Conclusion

- Integrate security into your software lifecycle.
- Start working on creating SBOM's and see how SLSA can fit into your process.
- Try to work with SBOM output and use it!

@nielstanis@infosec.exchange

VERACODE

Thanks! Questions?

https://github.com/nielstanis/ndclondon2023-supplychainws

ntanis at veracode.com
@nielstanis@infosec.exchange on Mastodon