

Copenhagen  
*Developers Festival*

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



Copenhagen Developers Festival

 [@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)



## Agenda

- Introduction
- DocGenerator Library
- Signing artifacts
- Reproducible Builds
- Software Bill Of Materials (SBOM)
- Google SLSA
- Docker SBOM
- I got hacked!

Copenhagen Developers Festival

 @nielstanis@infosec.exchange

## Agenda



- <https://github.com/nielstanis/cphdevfest2023>
- <https://github.com/nielstanis/docgenerator>

Copenhagen Developers Festival

 @nielstanis@infosec.exchange

0101  
0101

## What is a Supply Chain?

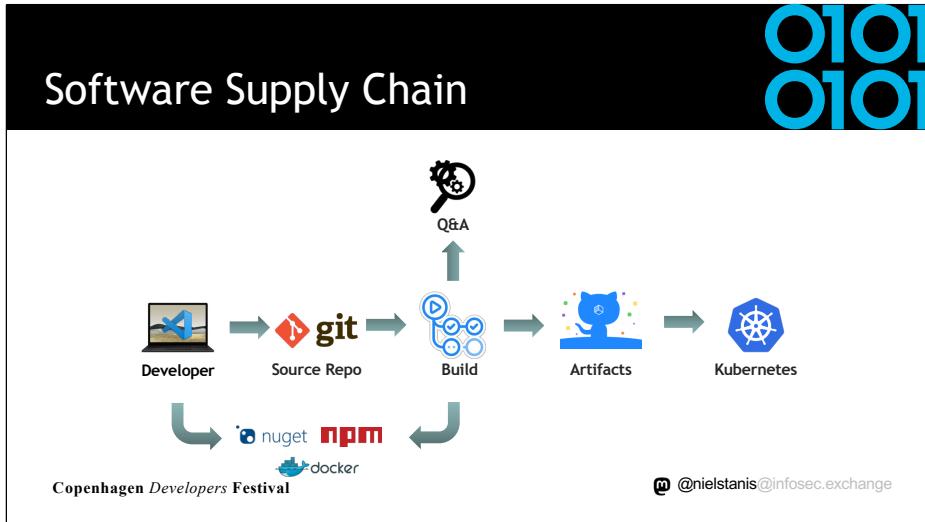


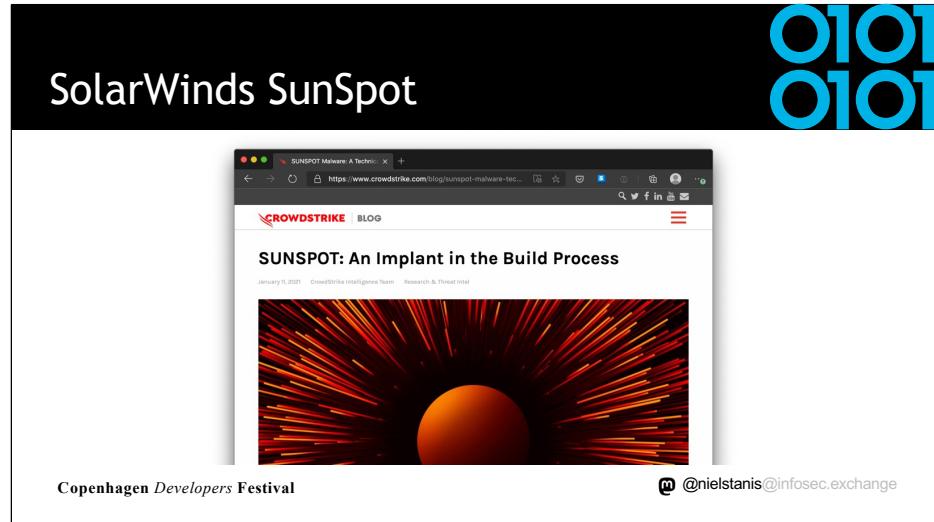
Copenhagen Developers Festival

@nielstanis@infosec.exchange

Image source:

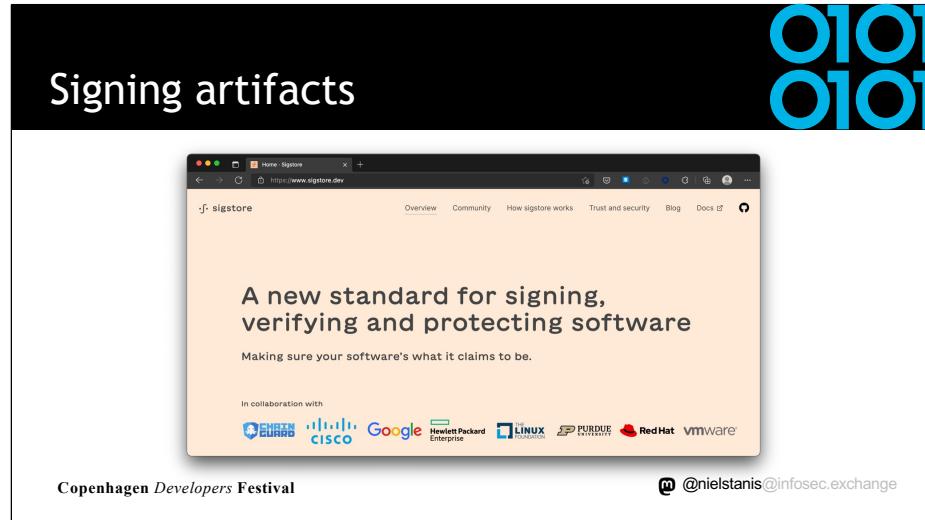
[https://www.wardsauto.com/sites/wardsauto.com/files/styles/article\\_featured\\_retina/public/Renault%20Kadjar%20assembly%20line%20-%20Palencia%20Spain-5\\_8.jpg?](https://www.wardsauto.com/sites/wardsauto.com/files/styles/article_featured_retina/public/Renault%20Kadjar%20assembly%20line%20-%20Palencia%20Spain-5_8.jpg?)





<https://www.crowdstrike.com/blog/sunspot-malware-technical-analysis/>

<https://www.microsoft.com/security/blog/2020/12/18/analyzing-solorigate-the-compromised-dll-file-that-started-a-sophisticated-cyberattack-and-how-microsoft-defender-helps-protect/>



<https://sigstore.dev>



## Signing artifacts

The screenshot shows a web browser displaying the [sigstore.dev](https://www.sigstore.dev) website. The main heading is "How sigstore works". Below it is a diagram illustrating the workflow:

```
graph TD; DEVS["DEVELOPERS, MAINTAINERS, MONITORS"] --> SIGN["SIGN AND PUBLISH ARTIFACTS"]; DEVS --> PUBLISH["PUBLISH SIGNING CERTIFICATES"]; DEVS --> MONITOR["MONITOR LOGS"]; SIGN --> FULCIO["FULCIO CERTIFICATE AUTHORITY"]; PUBLISH --> SIGLOG["SIGNATURE TRANSPARENCY LOG"]; MONITOR --> KEYLOG["KEY TRANSPARENCY LOG"]; FULCIO --- TRUST["TRUST ROOT"]; SIGLOG --- TRUST; KEYLOG --- TRUST;
```

The diagram shows "DEVELOPERS, MAINTAINERS, MONITORS" at the top, connected by dashed arrows to three circular boxes: "SIGN AND PUBLISH ARTIFACTS", "PUBLISH SIGNING CERTIFICATES", and "MONITOR LOGS". Each of these three boxes has a dashed arrow pointing down to one of three rectangular boxes: "FULCIO CERTIFICATE AUTHORITY", "SIGNATURE TRANSPARENCY LOG", and "KEY TRANSPARENCY LOG". Finally, all three of these boxes have dashed arrows pointing down to a large rectangular box labeled "TRUST ROOT".

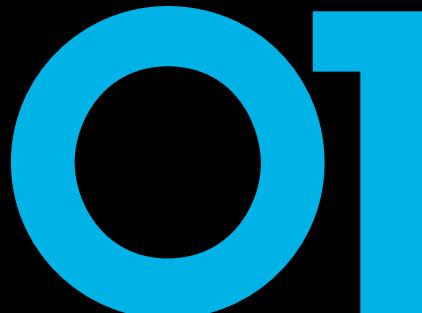
Below the diagram, there is explanatory text about a "standardized approach" and "building for future integrations". At the bottom left is the text "Copenhagen Developers Festival" and at the bottom right is the handle "@nielstanis @infosec.exchange".

<https://sigstore.dev>

Divider slide option  
please left justify the  
text. Delete Sub title  
if not needed. Delete  
presenter if not need

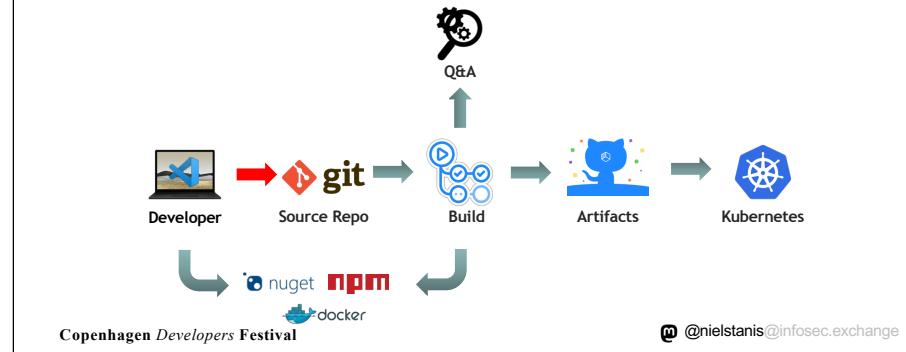
# Lab 1

Sigstore



0101  
0101

## Software Supply Chain





## GIT Commit Signing

The screenshot shows a GitHub repository page for 'nielstanis / ndcsydney2022-supplychain'. A recent commit by 'nielstanis' is displayed, showing a green 'Verified' badge next to the author name. The commit message is a long hex string. Below the commit, a file list shows '.gitignore' and 'Demo.md' files. The GitHub interface includes standard navigation bars like 'Code', 'Issues', and 'Actions'.

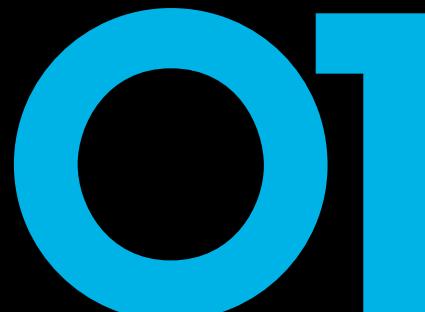
Copenhagen Developers Festival @nielstanis@infosec.exchange

<https://www.hanselman.com/blog/HowToSetupSignedGitCommitsWithAYubiKeyNEOAndGPGAndKeybaseOnWindows.aspx>

Divider slide option  
please left justify the  
text. Delete Sub title  
if not needed. Delete  
presenter if not need

## Lab 2

Setup repository, build  
and git signing with  
GitSign



# Reproducible/Deterministic Builds



Home

Contribute

[Documentation](#)

Tools

Who is involved?

News

Events

Talks

## Definitions

### When is a build reproducible?

A build is **reproducible** if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts.

The relevant attributes of the build environment, the build instructions and the source code as well as the expected reproducible artifacts are defined by the authors or distributors. The artifacts of a build are the parts of the build results that are the desired primary output.

Copenhagen Developers Festival

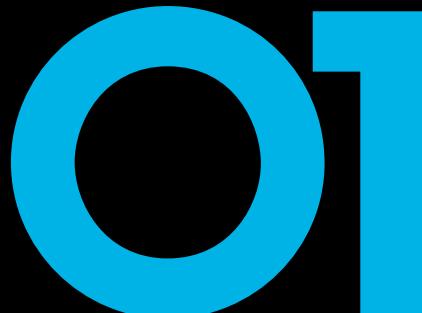
 [@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

<https://reproducible-builds.org/docs/definition/>

Divider slide option  
please left justify the  
text. Delete Sub title  
if not needed. Delete  
presenter if not need

# Lab 3

.NET reproducibility



## 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’

Copenhagen Developers Festival

@nielstanis@infosec.exchange

<https://blog.paranoidcoding.com/2016/04/05/deterministic-builds-in-roslyn.html>  
<https://github.com/dotnet/roslyn/blob/master/docs/compilers/Deterministic%20Inputs.md>  
<https://github.com/clairernovotny/DeterministicBuilds>



## Reproducible/Deterministic Builds

- DotNet.ReproducibleBuilds NuGet Package
  - MSBuild *ContinuousIntegrationBuild*
  - SourceLink
- Dotnet.ReproducibleBuilds.Isolated NuGet Package
  - Hermetic builds

Copenhagen Developers Festival

@nielstanis@infosec.exchange

<https://blog.paranoidcoding.com/2016/04/05/deterministic-builds-in-roslyn.html>

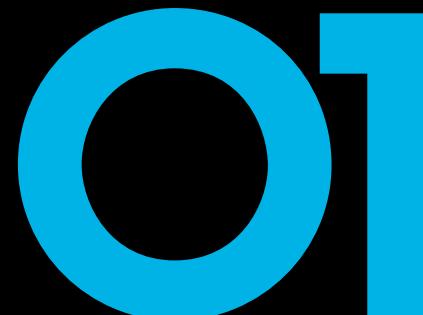
<https://github.com/dotnet/roslyn/blob/master/docs/compilers/Deterministic%20Inputs.md>

<https://devblogs.microsoft.com/dotnet/producing-packages-with-source-link/>

<https://github.com/dotnet/reproducible-builds>

# Lab 4

DotNet.Reproducible



Divider slide option  
please left justify the  
text. Delete Sub title  
if not needed. Delete  
presenter if not need



## 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`

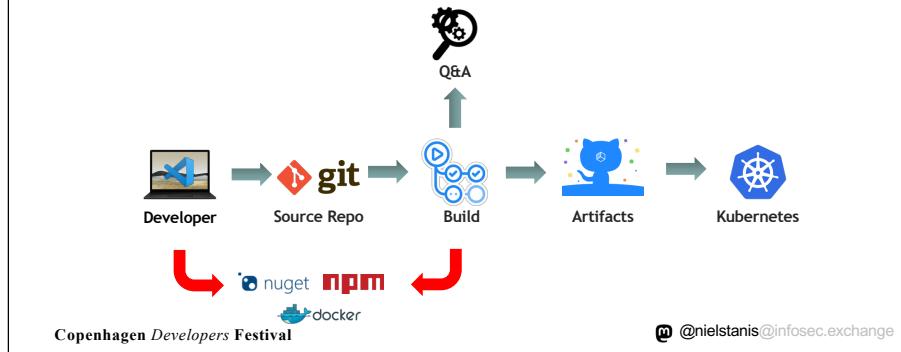
Copenhagen Developers Festival

 @nielstanis@infosec.exchange

<https://github.com/dotnet/designs/blob/main/accepted/2020/reproducible-builds.md>

0101  
0101

## 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."*



Copenhagen Developers Festival

@nielstanis@infosec.exchange

<https://info.veracode.com/fy22-state-of-software-security-v11-open-source-edition.html>



## Vulnerabilities in libraries

The screenshot shows a GitHub issue page for Microsoft Security Advisory CVE-2023-38180. The title is "Microsoft Security Advisory CVE-2023-38180: .NET Denial of Service Vulnerability". The issue was opened by rhonda 3 weeks ago and has 0 comments. The issue body contains the following text:

Microsoft is releasing this security advisory to provide information about a vulnerability in ASP.NET Core 2.1, .NET 4.0, and .NET 7.0. This advisory also provides guidance on what developers can do to update their applications to remove this vulnerability.

A vulnerability exists in Kestrel where, on detecting a potentially malicious client, Kestrel will sometimes fail to disconnect it, resulting in denial of service.

Labels: Security

Assignees: No one assigned

Projects: None yet

Milestone: No milestone

At the bottom of the screenshot, there is a footer with the text "Copenhagen Developers Festival" and a Twitter handle "@nielstanis@infosec.exchange".

<https://github.com/dotnet/announcements/issues/269>

## Automotive Industry

O1O1  
O1O1



Copenhagen Developers Festival

@nielstanis@infosec.exchange

O1O1  
O1O1

## 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

Copenhagen Developers Festival

@nielstanis@infosec.exchange

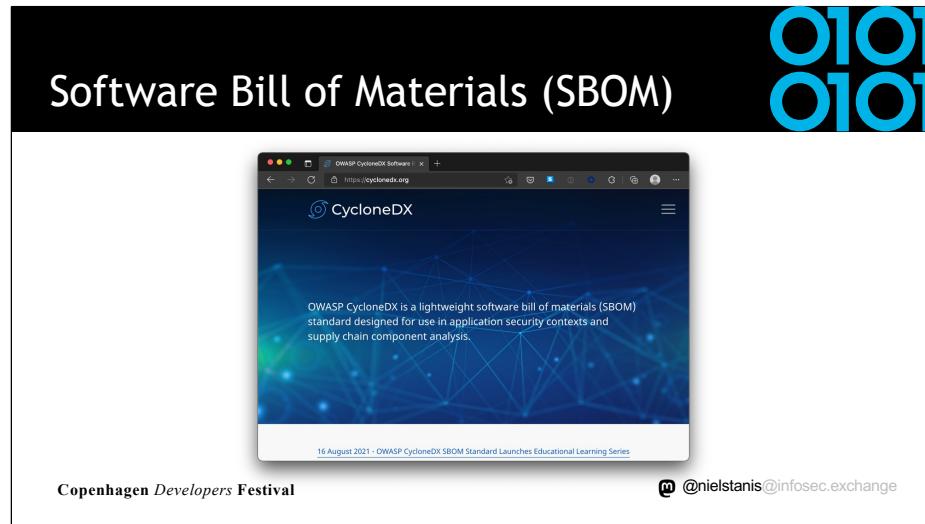


## Software Bill of Materials (SBOM)

- 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?

Copenhagen Developers Festival

 @nielstanis@infosec.exchange



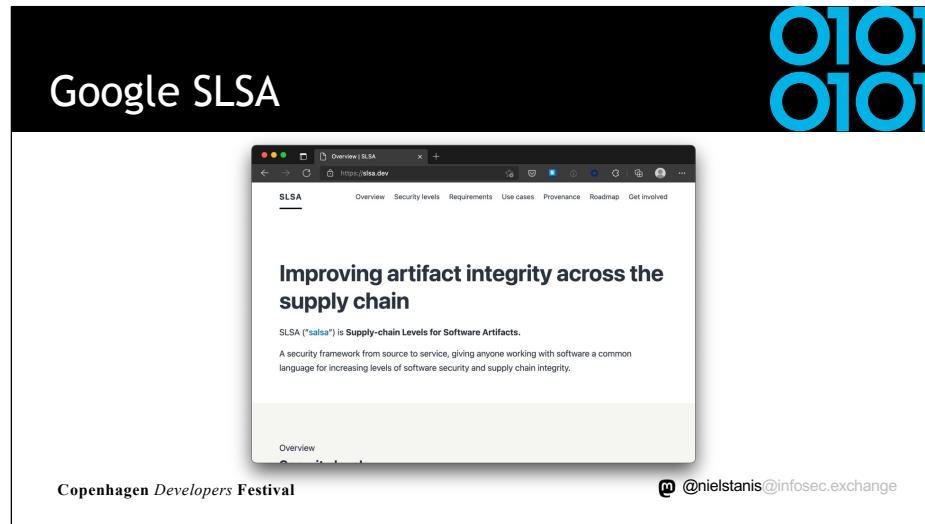
<https://cyclonedx.org>

Divider slide option  
please left justify the  
text. Delete Sub title  
if not needed. Delete  
presenter if not need

# Lab 5

CycloneDX .NET  
Any vulnerabilities?





<https://slsa.dev>



## 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

Copenhagen Developers Festival

 @nielstanis@infosec.exchange

<https://slsa.dev>

## 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.

Copenhagen Developers Festival

 [@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

<https://slsa.dev>

## 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

Copenhagen Developers Festival

 @nielstanis@infosec.exchange

<https://security.googleblog.com/2022/04/improving-software-supply-chain.html>

SLSA3 Generator GitHub Actions

General availability of SLSA3 Generic Generator for GitHub Actions

by Ian Lewis, Laurent Simon, Asra Ali  
29 Aug 2022

A few months ago Google and GitHub announced the release of a Go builder that would help software developers and consumers more easily verify the origins of software by using verification files known as provenance. Since then, the SLSA community has been working to enable provenance generation for other projects that may use any number of languages or build tools. Today, we're pleased to announce that we're adding a new tool to generate similar provenance documents for projects developed in any programming language, while keeping your existing building workflows.

Copenhagen Developers Festival      @nielstanis@infosec.exchange

<https://slsa.dev/blog/2022/08/slsa-github-workflows-generic-ga>

# Lab 6

SLSA Level 3 Provenance

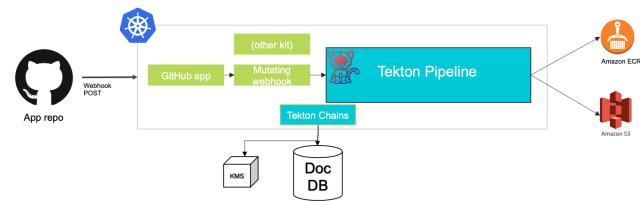


Divider slide option  
please left justify the  
text. Delete Sub title  
if not needed. Delete  
presenter if not need



## SolarWinds Project Trebuchet

### Pipeline With Attestations



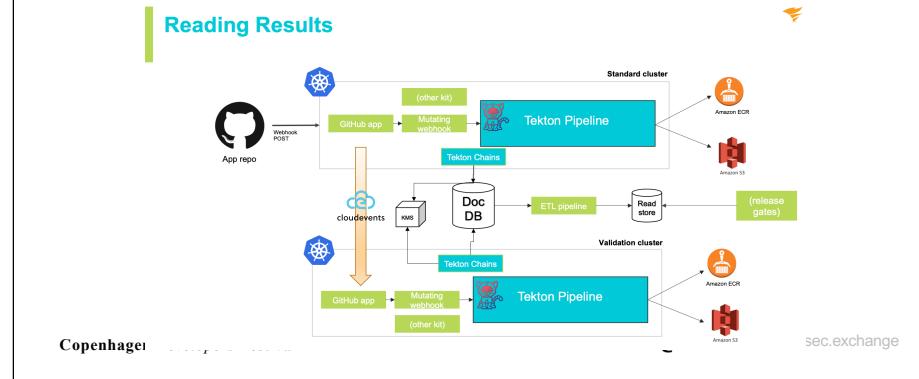
Copenhagen Developers Festival

@nielstanis@infosec.exchange

[https://static.sched.com/hosted\\_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf](https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf)  
<https://www.youtube.com/watch?v=1-tMRxqMwTQ>



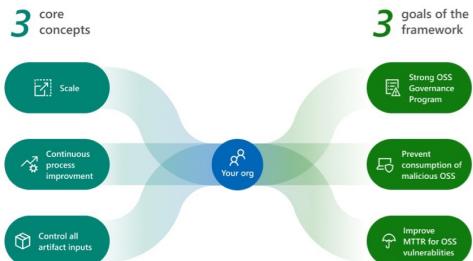
## SolarWinds Project Trebuchet



[https://static.sched.com/hosted\\_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf](https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf)  
<https://www.youtube.com/watch?v=1-tMRxqMwTQ>

# Secure Supply Chain Consumption Framework (S2C2F)

O1O1  
O1O1



Copenhagen Developers Festival

@nielstanis@infosec.exchange

<https://www.microsoft.com/en-us/securityengineering/opensource/ossscframeworkguide>

## Secure Supply Chain Consumption Framework Practices



1. Ingest It - I can ship any existing asset if external OSS sources are compromised or unavailable
2. Scan It - I know if any OSS artifact in my pipeline has vulnerabilities or malware
3. Inventory It - I know where OSS artifacts are deployed in production.
4. Update It - I can deploy updated external artifacts soon after an update becomes publicly available.
5. Audit It - I can prove that every OSS artifact in production has a full chain-of-custody from the original artifact source and is consumed through the official supply chain
6. Enforce It - I can rely on secure and trusted OSS consumption within my organization.
7. Rebuild It - I can rebuild from source code every OSS artifact I'm deploying.

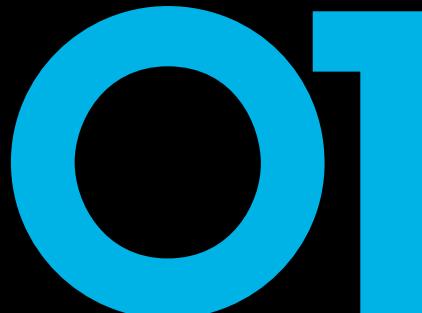
Copenhagen Developers Festival

@nielstanis@infosec.exchange

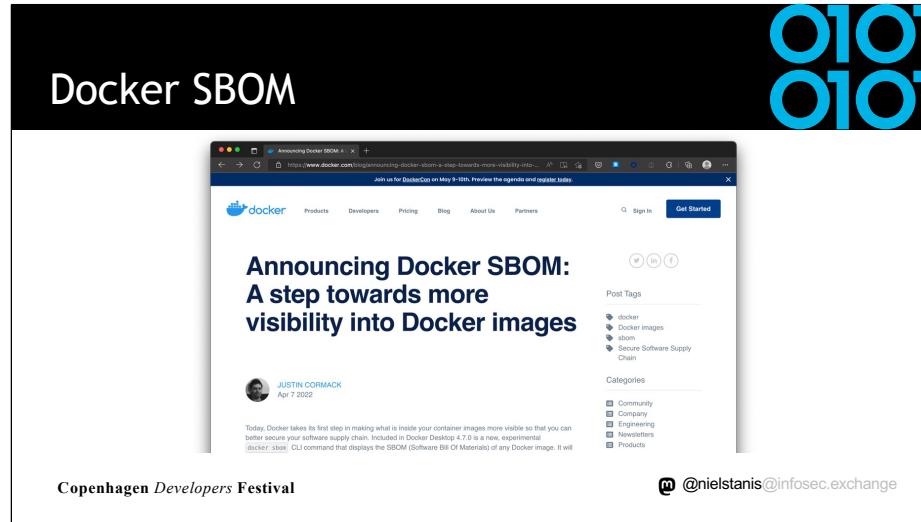
<https://www.microsoft.com/en-us/securityengineering/opensource/ossscframeworkguide>

# Lab 7

DocGenerator on  
ASP.NET Core MVC &  
Docker



Divider slide option  
please left justify the  
text. Delete Sub title  
if not needed. Delete  
presenter if not need

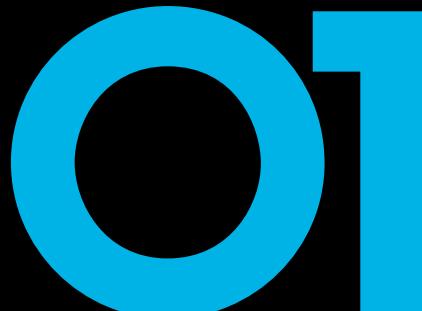


<https://www.docker.com/blog/announcing-docker-sbom-a-step-towards-more-visibility-into-docker-images/>

Divider slide option  
please left justify the  
text. Delete Sub title  
if not needed. Delete  
presenter if not need

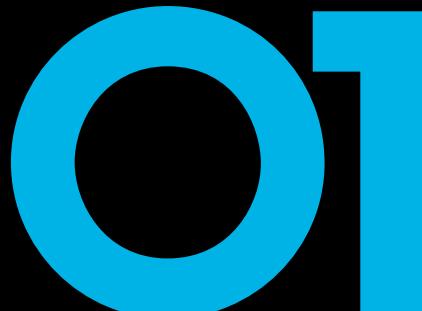
# Lab 8

Docker SBOM with  
Anchore Syft



# Lab 9

Cosign on Docker in  
GitHub Actions



Divider slide option  
please left justify the  
text. Delete Sub title  
if not needed. Delete  
presenter if not need

# Ubuntu Chiseled - Distroless Linux



A screenshot of a web browser showing a bar chart comparing the sizes of chiseled Ubuntu .NET images. The chart shows five categories: `dotnet/highcpu-deps`, `dotnet/highcpu`, `dotnet/lightcpu`, `dotnet/lightcpu-deps`, and `dotnet/lightcpu-interpret`. The sizes are 13MB, 116MB, 94MB, 124MB, and 207MB respectively.

At-scale comparison of chiseled Ubuntu for .NET image sizes (from mcr.microsoft.com)

Copenhagen Developers Festival

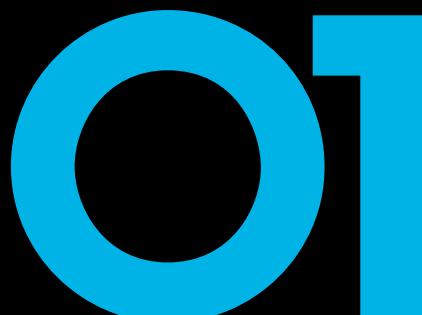
@nielstanis@infosec.exchange

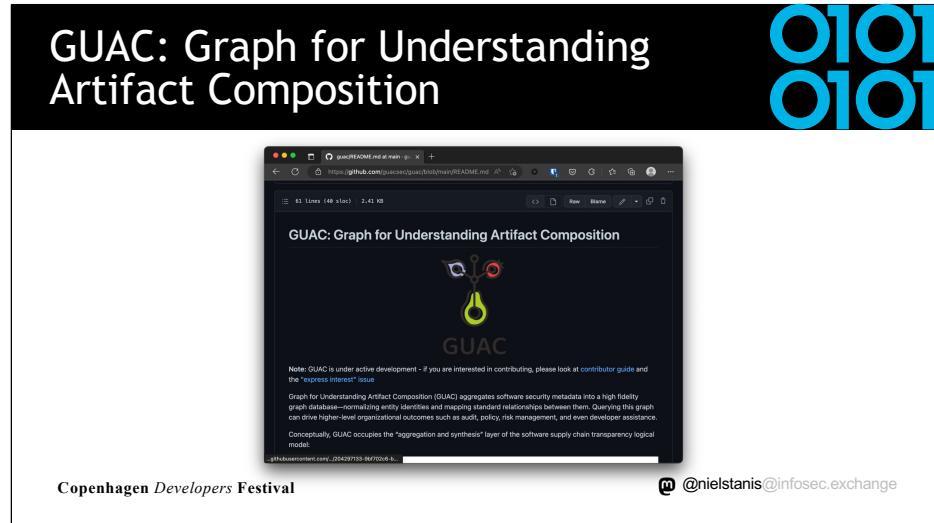
<https://ubuntu.com/blog/chiselled-containers-perfect-gift-cloud-applications>

Divider slide option  
please left justify the  
text. Delete Sub title  
if not needed. Delete  
presenter if not need

# Lab 10

You got HACKED!





<https://github.com/guacsec/guac>

## 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.

Copenhagen Developers Festival

 @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!

Copenhagen Developers Festival

 @nielstanis@infosec.exchange

VERACODE

Thanks!

<https://github.com/nielstanis/cphdevfest2023>  
ntanis at veracode.com  
@nielstanis@infosec.exchange on Mastodon

