

NDC { Porto }

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

Niels Tanis



0101
0101

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



NDC { Porto }

 @nielstanis@infosec.exchange

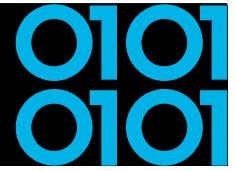


Agenda

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

NDC { Porto }

 @nielstanis@infosec.exchange



Agenda

- <https://github.com/nielstanis/ndcporto2023-supply/>
- <https://github.com/nielstanis/docgenerator>

NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

What is a Supply Chain?



NDC { Porto }

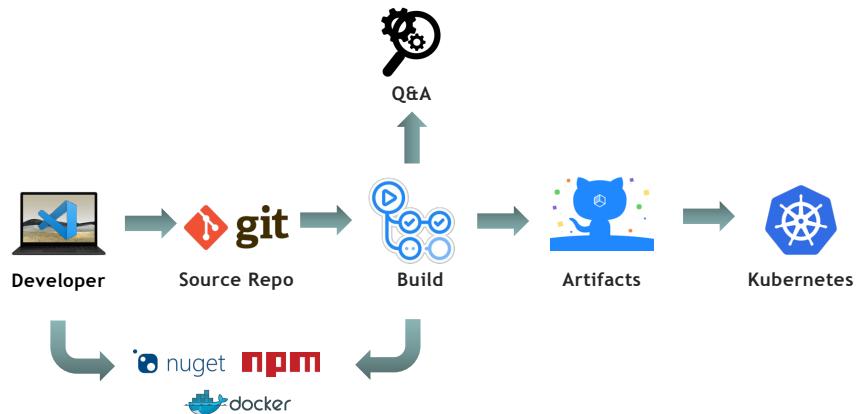
 @nielstanis@infosec.exchange

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?

0101
0101

Software Supply Chain

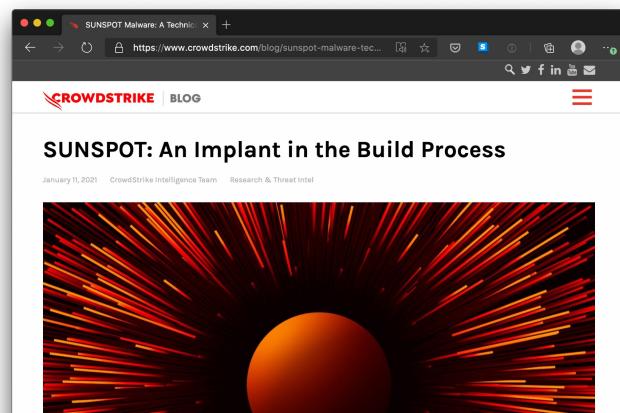


NDC { Porto }

@nielstanis@infosec.exchange

SolarWinds SunSpot

0101
0101



NDC { Porto }

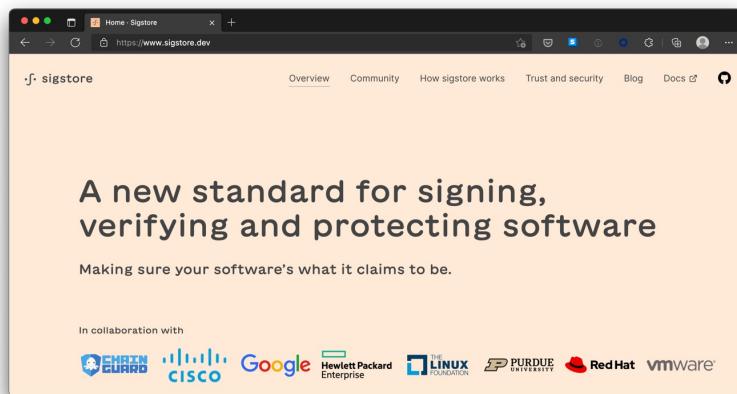
 @nielstanis@infosec.exchange

<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/>

Signing artifacts

0101
0101



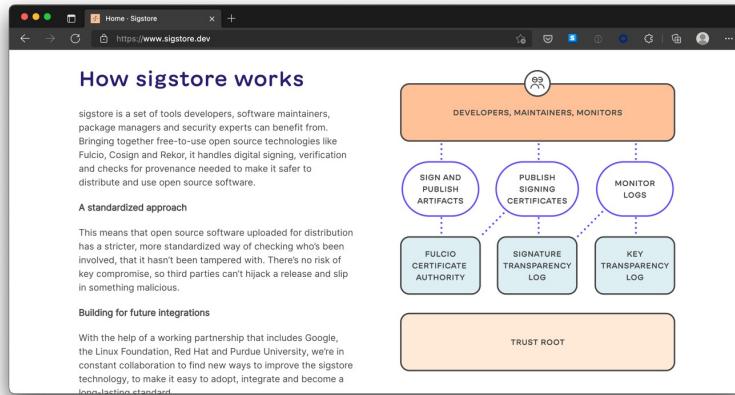
NDC { Porto }

@nielstanis@infosec.exchange

<https://sigstore.dev>

0101
0101

Signing artifacts



NDC { Porto }

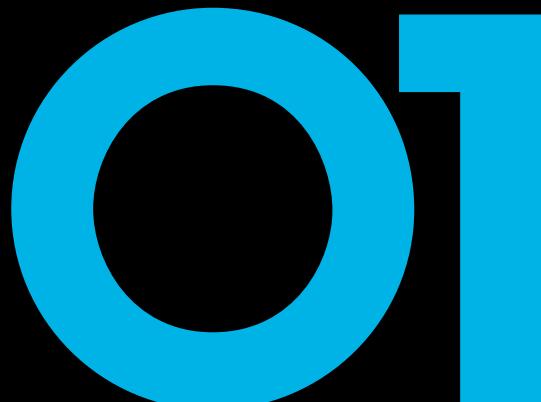
@nielstanis@infosec.exchange

<https://sigstore.dev>

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

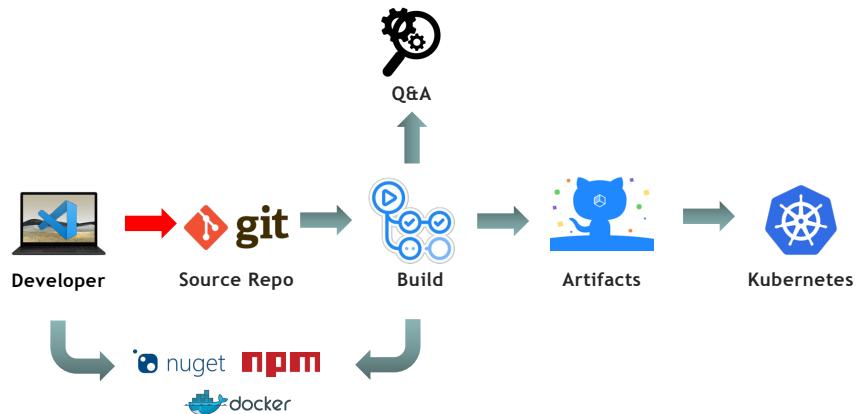
Lab 1

**Signing artifacts with
Sigstore**



0101
0101

Software Supply Chain



NDC { Porto }

@nielstanis@infosec.exchange

0101
0101

GIT Commit Signing

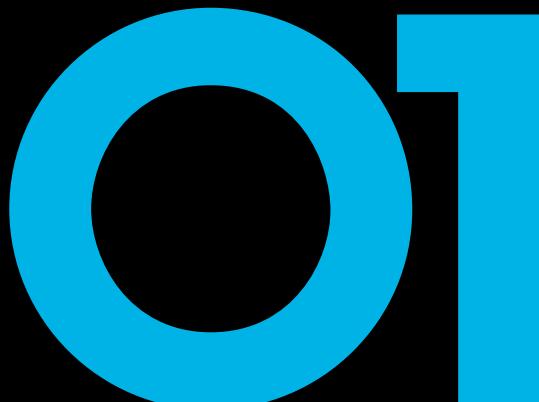
A screenshot of a GitHub repository page for 'nielstanis / ndcsydney2022-supplychain'. The repository is public. A recent commit, 'Initial content', was made by 'nielstanis' 5 minutes ago and is verified. The commit hash is 054a8a3b9c91e8cc6ec2e9f144d2b9b7dbc8628e. The commit message shows 12 changed files with 3,668 additions and 0 deletions. The repository structure includes .gitignore, .Cosign, and Demo.md. On the right, there is a sidebar with a profile picture and the email address @nielstanis@infosec.exchange.

<https://www.hanselman.com/blog/HowToSetupSignedGitCommitsWithAYubiKeyNEOA ndGPGAndKeybaseOnWindows.aspx>

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

Lab 2

Setup repository, build
and git signing with
GitSign



Reproducible/Deterministic Builds

0101
0101

The screenshot shows a dark-themed website for "Reproducible Builds". On the left is a sidebar with a blue header containing the "Reproducible Builds" logo. Below the header, the sidebar lists several menu items: Home, Contribute, Documentation (which is currently selected), Tools, Who is involved?, News, Events, and Talks. The main content area has a white background. At the top of this area, the word "Definitions" is centered in bold black font. Below it, the heading "When is a build reproducible?" is also in bold black font. To the right of this heading is a detailed explanation of what makes a build reproducible, mentioning source code, build environment, and build instructions.

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.

NDC { Porto }

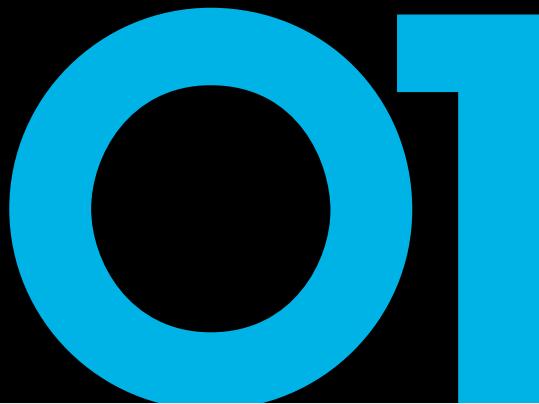
@nielstanis@infosec.exchange

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

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

Lab 3

.NET reproducibility





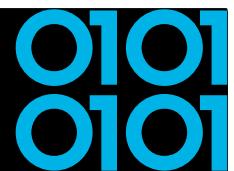
Reproducible/Deterministic Builds

- 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’

NDC { Porto }

@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

NDC { Porto }

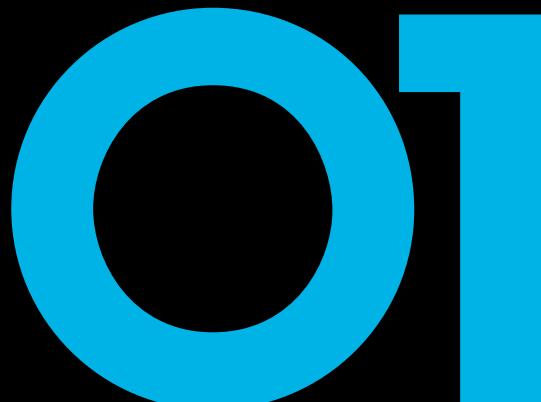
@nielstanis@infosec.exchange

<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>

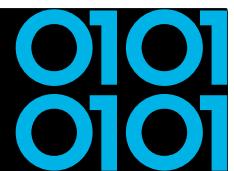
Divider slide option
please left justify the
text. Delete Sub title
not needed. Delete
presenter if not needed

Lab 4

DotNet.Reproducible



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`

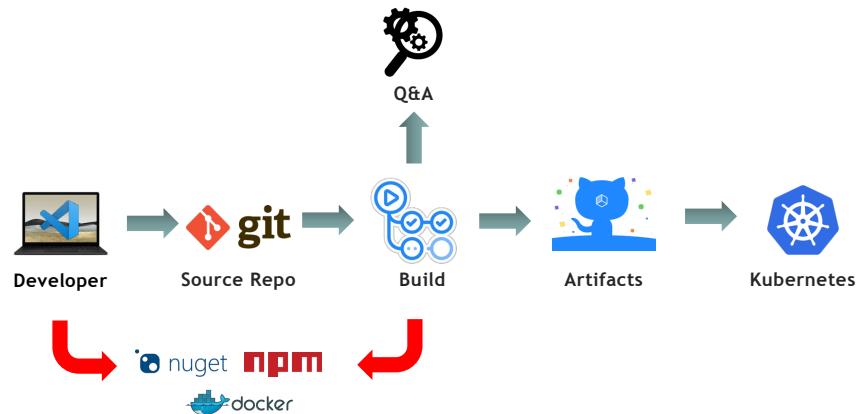
NDC { Porto }

@nielstanis@infosec.exchange

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

0101
0101

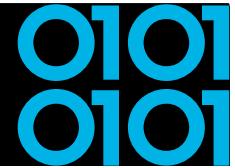
3rd Party Libraries



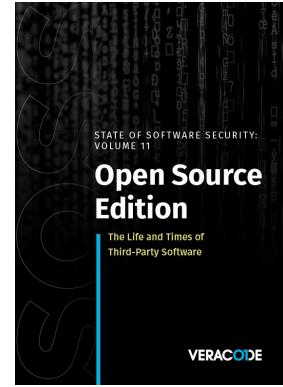
NDC { Porto }

@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."



NDC { Porto }

@nielstanis@infosec.exchange

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

0101
0101

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 rbanda 3 weeks ago and has 0 comments. The body of the issue contains the Microsoft Security Advisory text, which states: "Microsoft is releasing this security advisory to provide information about a vulnerability in ASP.NET Core 2.1, .NET 6.0, 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." The right sidebar shows project details: Assignees (No one assigned), Labels (Security), Projects (None yet), and Milestone (No milestone).

NDC { Porto }

@nielstanis@infosec.exchange

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

0101
0101

Automotive Industry



NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

Car Supply Chain



Tata Steel Factory

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

Bosch Factory

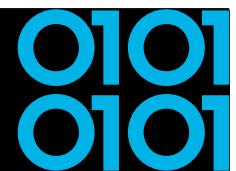
- Steel Batch #1234 Tata
- ECE-R90 Tested/Certified
 - Serie #45678
- Used by Ford, Volkswagen and Renault

Renault Manufacturing

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

NDC { Porto }

 @nielstanis@infosec.exchange



Software Bill of Materials (SBOM)

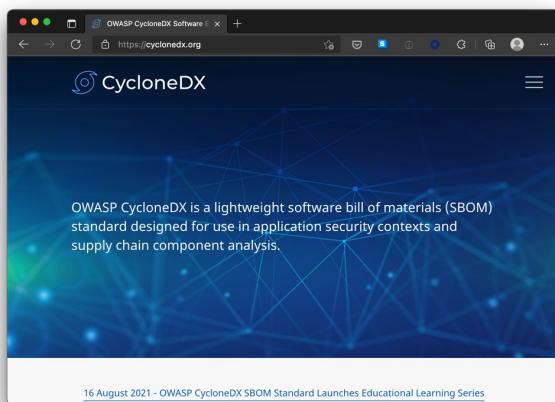
- Industry standard of describing the software
 - Producer Identity - Who Created it?
 - Product Identity - What's the product?
 - Integrity - Is the project unaltered?
 - Licensing - How can the project be used?
 - Creation - How was the product created? Process meets requirements?
 - Materials - How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

NDC { Porto }

@nielstanis@infosec.exchange

Software Bill of Materials (SBOM)

0101
0101



NDC { Porto }

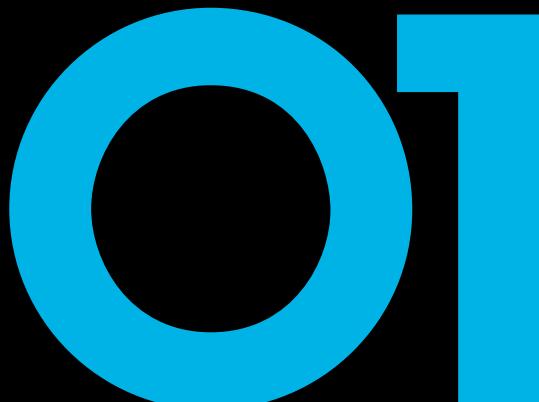
 @nielstanis@infosec.exchange

<https://cyclonedx.org>

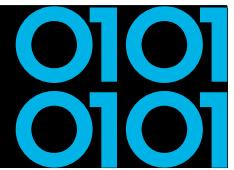
Divider slide option
please left justify text. Delete Sub title if not needed. Delete presenter if not needed

Lab 5

CycloneDX .NET
Any vulnerabilities?



Google SLSA

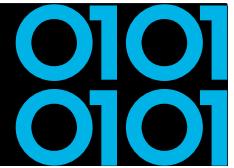


The screenshot shows a web browser window with the URL <https://slsa.dev>. The page has a dark header with the text "Google SLSA" and the binary logo. Below the header is a navigation bar with links: Overview, Security levels, Requirements, Use cases, Provenance, Roadmap, and Get involved. The main content area features a large heading "Improving artifact integrity across the supply chain". Below the heading, there is a brief description: "SLSA ("salsa") is Supply-chain Levels for Software Artifacts. A security framework from source to service, giving anyone working with software a common language for increasing levels of software security and supply chain integrity." At the bottom of the main content area is a "Overview" button.

NDC { Porto }

@nielstanis@infosec.exchange

<https://slsa.dev>



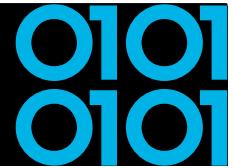
Google SLSA Levels

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

NDC { Porto }

@nielstanis@infosec.exchange

<https://slsa.dev>



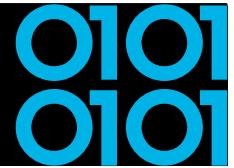
Google SLSA Levels

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.

NDC { Porto }

 @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

NDC { Porto }

 @nielstanis@infosec.exchange

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

The screenshot shows a web browser displaying a blog post from the SLSA website. The title of the post is "General availability of SLSA3 Generic Generator for GitHub Actions". The post is authored by Ian Lewis, Laurent Simon, and Asra Ali, and was published on 29 Aug 2022. The content discusses the addition of a new tool to generate similar provenance documents for projects developed in any programming language, while keeping existing building workflows. The SLSA logo, consisting of the letters "SLSA" in white inside a blue hexagon, is visible in the top right corner of the slide.

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.

NDC { Porto }

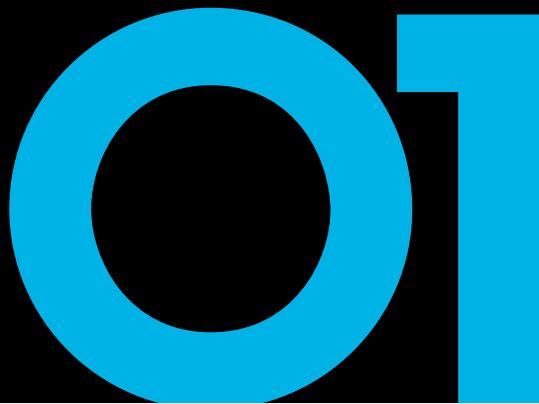
[@nielstanis@infosec.exchange](https://slsa.dev/blog/2022/08/slsa-github-workflows-generic-ga)

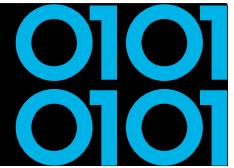
<https://slsa.dev/blog/2022/08/slsa-github-workflows-generic-ga>

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

Lab 6

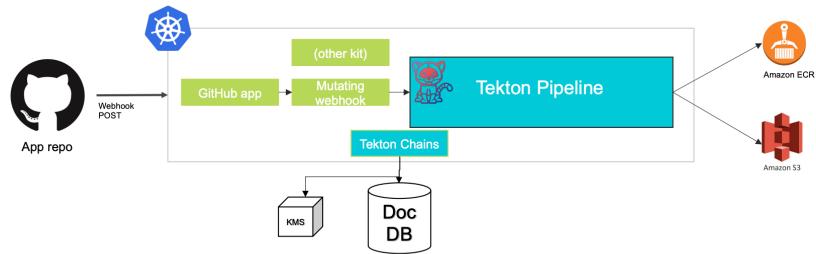
SLSA Level 3 Provenance





SolarWinds Project Trebuchet

Pipeline With Attestations

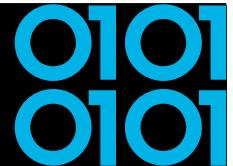


NDC { Porto }

@nielstanis@infosec.exchange

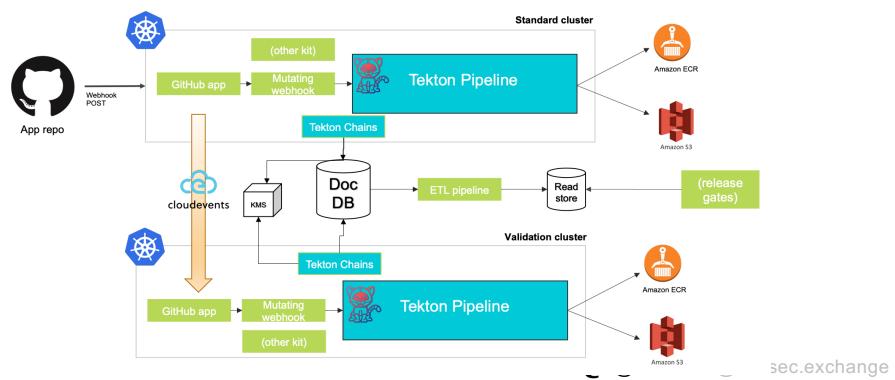
https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf

<https://www.youtube.com/watch?v=1-tMRxqMwTQ>



SolarWinds Project Trebuchet

Reading Results



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

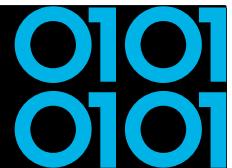


NDC { Porto }

@nielstanis@infosec.exchange

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

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.

NDC { Porto }

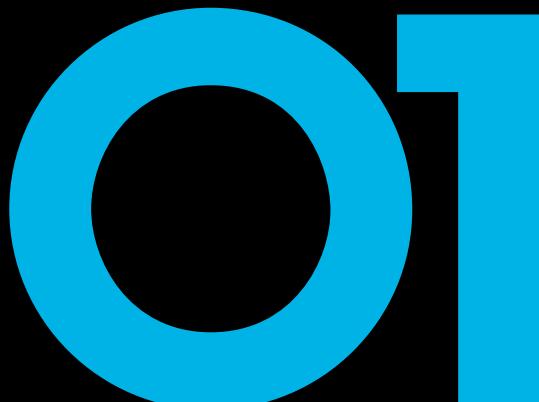
 @nielstanis@infosec.exchange

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

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

Lab 7

DocGenerator on
ASP.NET Core MVC &
Docker



The screenshot shows a blog post titled "Announcing Docker SBOM: A step towards more visibility into Docker images" by Justin Cormack. The post was published on April 7, 2022. The Docker logo is visible in the top right corner of the browser window. The URL in the address bar is <https://www.docker.com/blog/announcing-docker-sbom-a-step-towards-more-visibility-into-docker-images/>. The Docker logo is also present in the top right corner of the blog post header.

Docker SBOM

0101
0101

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

NDC { Porto }

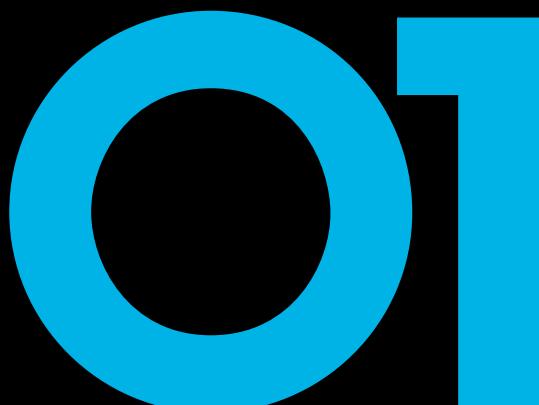
@nielstanis@infosec.exchange

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

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

Lab 8

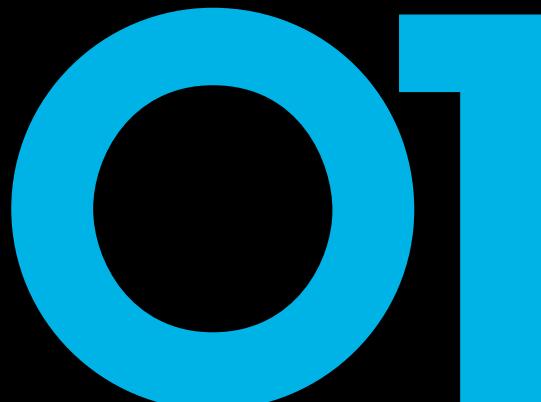
Docker SBOM with
Anchore Syft



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

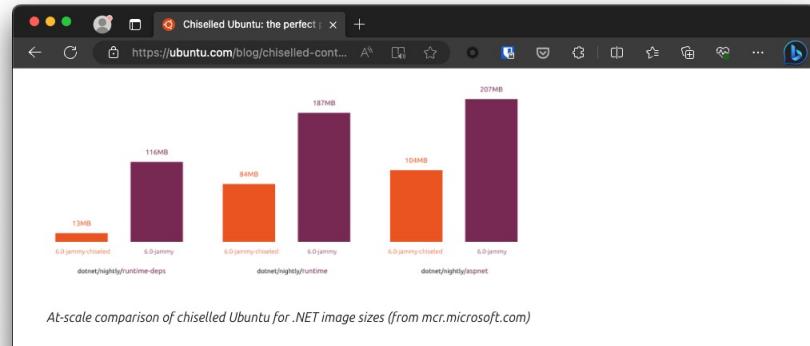
Lab 9

Cosign on Docker in
GitHub Actions



Ubuntu Chiseled - Distroless Linux

0101
0101



NDC { Porto }

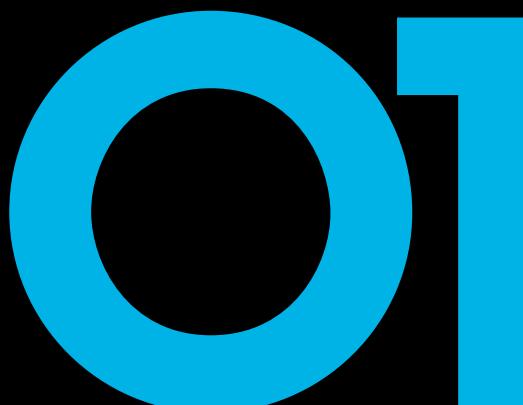
 @nielstanis@infosec.exchange

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

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

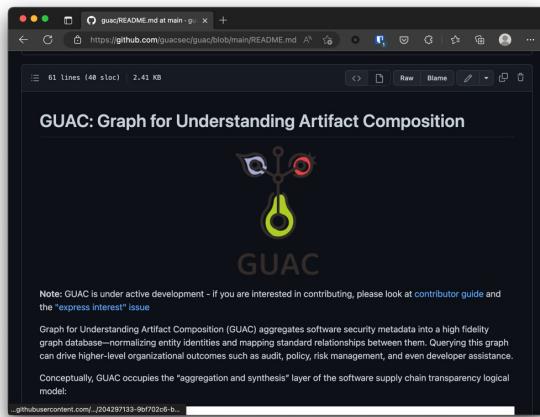
Lab 10

You got HACKED!



GUAC: Graph for Understanding Artifact Composition

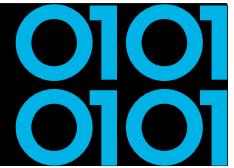
0101
0101



NDC { Porto }

 @nielstanis@infosec.exchange

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

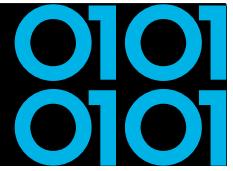


Conclusion

- 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.

NDC { Porto }

@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!

NDC { Porto }

@nielstanis@infosec.exchange

01010101010101010101010101010101
01 VERACODE 010101010101010101010101
01010101010101010101010101010101

Thanks!

<https://github.com/nielstanis/ndcporto2023-supply>
ntanis at veracode.com
@nielstanis@infosec.exchange on Mastodon

