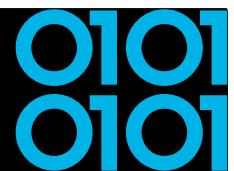


NDC { London }

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



NDC { London }

@nielstanis@infosec.exchange

Agenda

0101
0101

NDC { London }

 @nielstanis@infosec.exchange

0101
0101

What is a Supply Chain?



NDC { London }

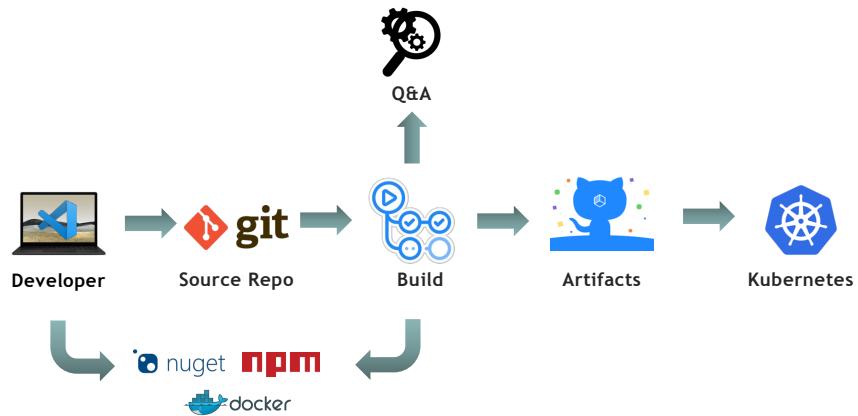
 @nielstanis@infosec.exchange

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

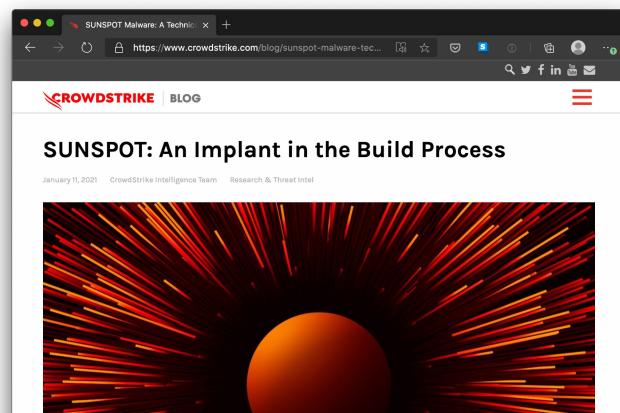


NDC { London }

@nielstanis@infosec.exchange

0101
0101

SolarWinds SunSpot



NDC { London }

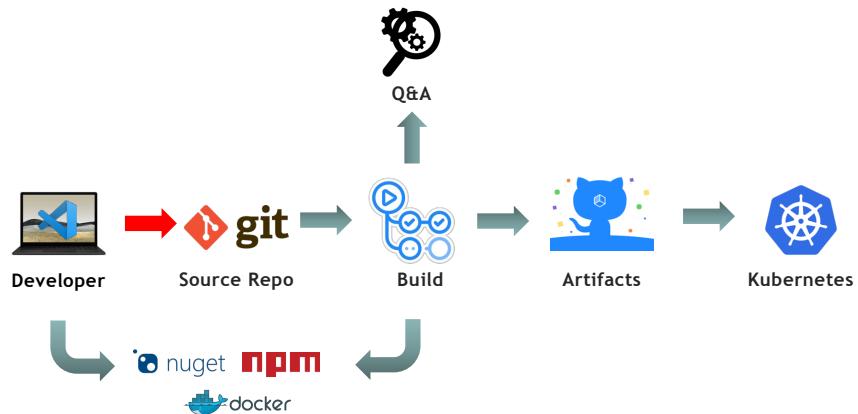
 @nielstanis@infosec.exchange

<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/>

0101
0101

Software Supply Chain

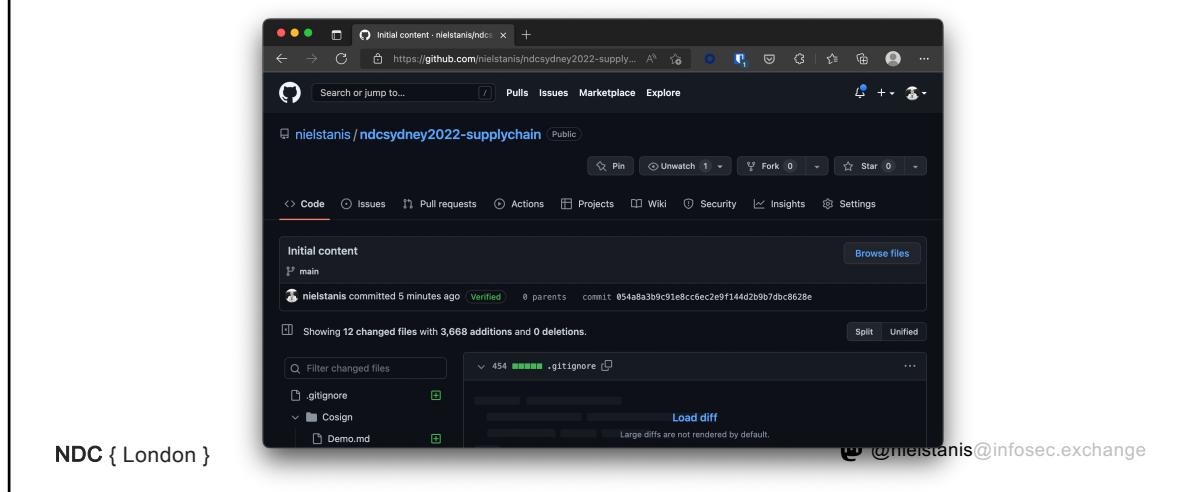


NDC { London }

@nielstanis@infosec.exchange

GIT Commit Signing

0101
0101

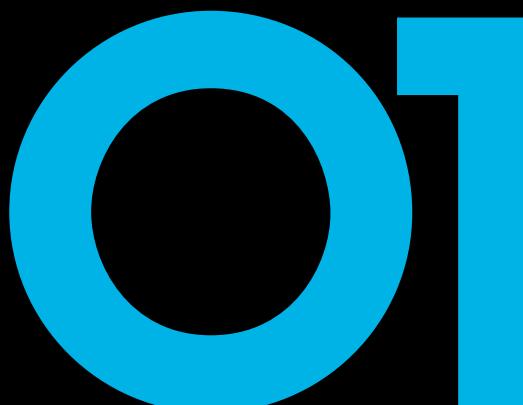


<https://www.hanselman.com/blog/HowToSetupSignedGitCommitsWithAYubiKeyNEOA ndGPGAndKeybaseOnWindows.aspx>

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

Lab 1

DocGenerator



Reproducible/Deterministic Builds

0101
0101

The screenshot shows a dark-themed website for 'Reproducible Builds'. At the top is a blue header bar with the 'Reproducible Builds' logo. Below it is a white sidebar containing links: Home, Contribute, Documentation (which is currently selected), Tools, Who is involved?, News, Events, and Talks. To the right of the sidebar, the main content area has a dark background. A large bold heading 'Definitions' is centered. Below it is a sub-section titled 'When is a build reproducible?'. The text defines a reproducible build as one where given the same source code, build environment, and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts. It also notes that relevant attributes like the build environment and source code are defined by authors or distributors.

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 { London }

@nielstanis@infosec.exchange

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



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 { London }

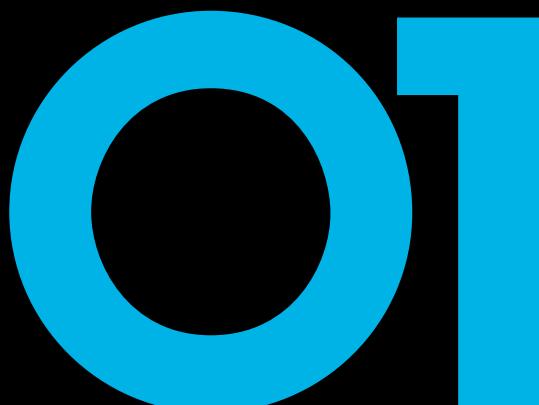
@nielstanis@infosec.exchange

<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>

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

Lab 2

.NET 7 reproducible





Reproducible/Deterministic Builds

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

NDC { London }

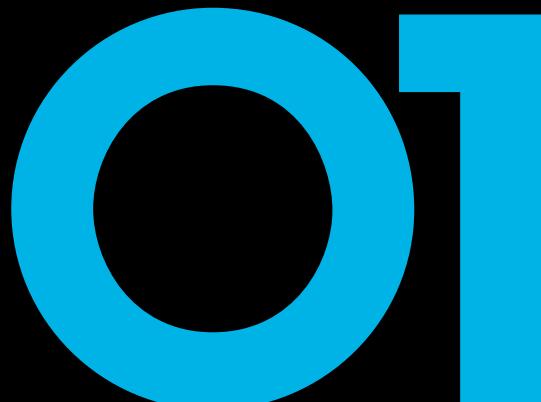
@nielstanis@infosec.exchange

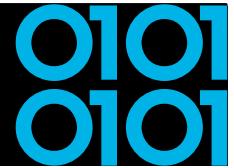
<https://blog.paranoidcoding.com/2016/04/05/deterministic-builds-in-roslyn.html>
<https://github.com/dotnet/roslyn/blob/master/docs/compilers/Deterministic%20Inputs.md>
<https://devblogs.microsoft.com/dotnet/producing-packages-with-source-link/>
<https://github.com/dotnet/reproducible-builds>

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

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`

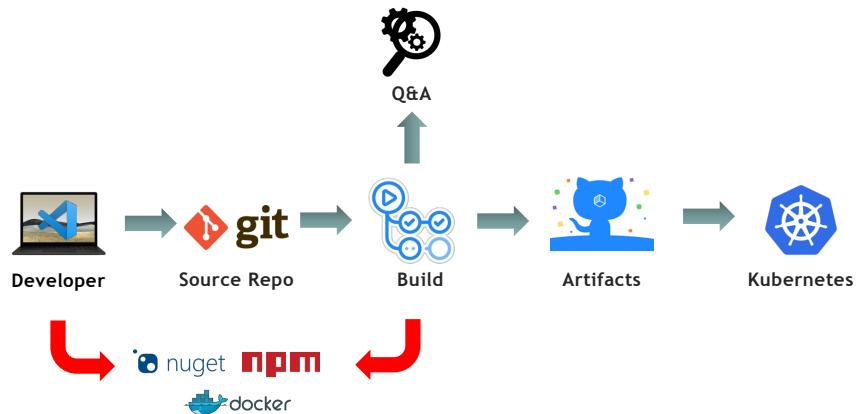
NDC { London }

@nielstanis@infosec.exchange

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

0101
0101

3rd Party Libraries



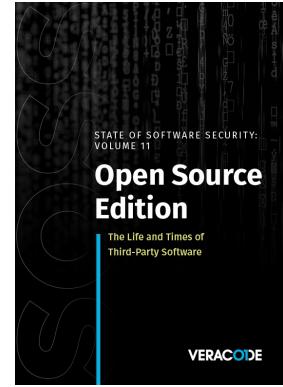
NDC { London }

@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 { London }

@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 the repository 'dotnet/announcements'. The issue is titled 'Microsoft Security Advisory CVE-2022-24512 | .NET Remote Code Execution Vulnerability #213'. The issue is marked as 'Open' and was created by 'dcwhittaker' on March 8. It has 0 comments. The issue details the discovery of a remote code execution vulnerability in .NET 6.0, .NET 5.0, and .NET Core 3.1 due to a stack buffer overrun in the .NET Double Parse routine. The issue is assigned to 'No one assigned' and has labels for 'Monthly-Update', '.NET Core 3.1', '.NET 5.0', '.NET 6.0', 'Patch-Tuesday', and 'Security'. The discussion section notes that the issue can be found at [dotnet/runtime#66348](https://github.com/dotnet/runtime#66348).

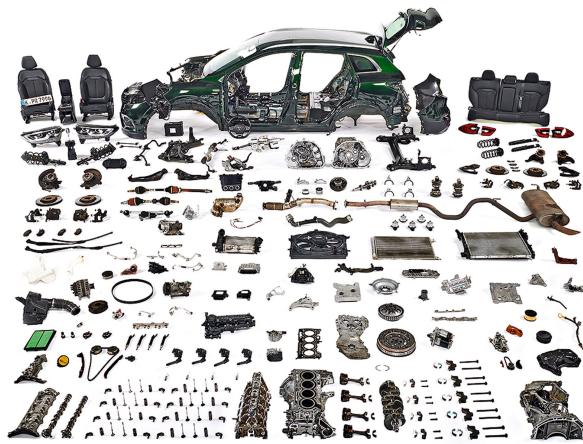
NDC { London }

@nielstanis@infosec.exchange

<https://github.com/dotnet/announcements/issues/213>

0101
0101

Automotive Industry



mark@marknismail.com @infosec.exchange

NDC { London }

0101
0101

Car Supply Chain



Tata Steel Factory

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

Bosch Factory

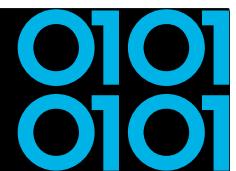
- Steel Batch #1234 Tata
- ECE-R90 Tested/Certified
 - Serie #45678
- Used by Ford, Volkswagen and Renault

Renault Manufacturing

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

NDC { London }

 @nielstanis@infosec.exchange



Software Bill of Materials (SBOM)

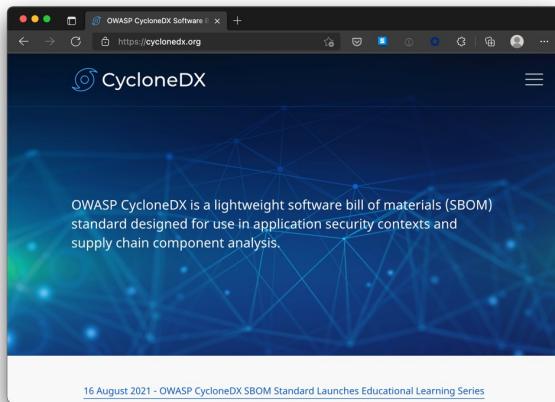
- Industry standard of describing the software
 - Producer Identity - Who Created it?
 - Product Identity - What's the product?
 - Integrity - Is the project unaltered?
 - Licensing - How can the project be used?
 - Creation - How was the product created? Process meets requirements?
 - Materials - How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

NDC { London }

@nielstanis@infosec.exchange

Software Bill of Materials (SBOM)

0101
0101



NDC { London }

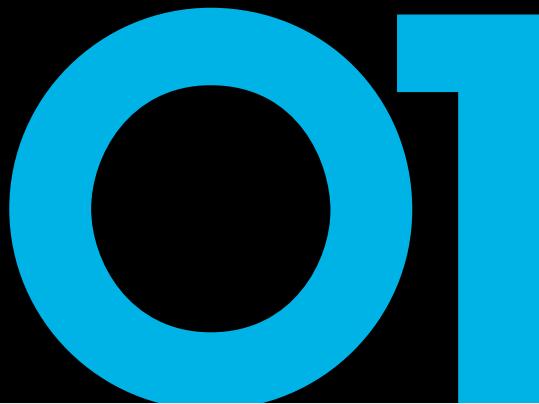
@nielstanis@infosec.exchange

<https://cyclonedx.org>

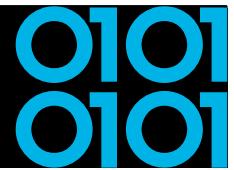
Divider slide option
please left justify the
text. Delete Sub title
not needed. Delete
presenter if not needed

Lab 4

CycloneDX .NET



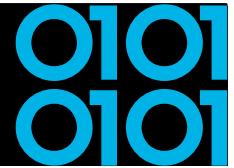
Google SLSA



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 SLSA logo. Below the header is a large blue section containing the text "Improving artifact integrity across the supply chain". Underneath this, there is a smaller text block explaining what SLSA is and its purpose. At the bottom of the page, there is a footer with links and social media icons.

<https://slsa.dev>

Google SLSA Levels

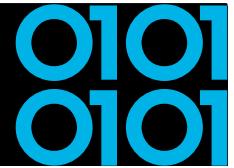


| Level | Description | Example |
|-------|--|---|
| 1 | Documentation of the build process | Unsigned provenance |
| 2 | Tamper resistance of the build service | Hosted source/build, signed provenance |
| 3 | Extra resistance to specific threats | Security controls on host, non-falsifiable provenance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

NDC { London }

 @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 { London }

 @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 { London }

@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 { London }

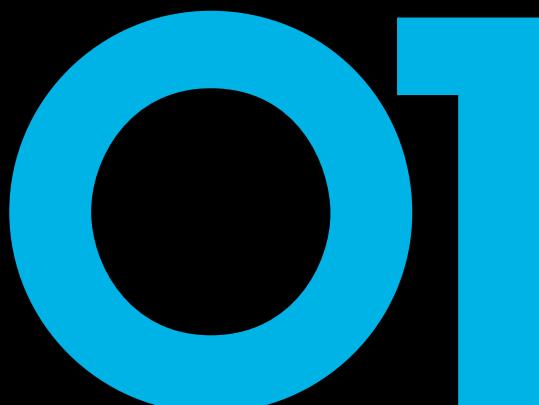
@nielstanis@infosec.exchange

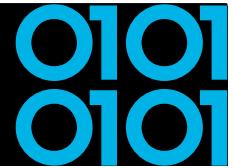
<https://slsa.dev/blog/2022/08/slsa-github-workflows-generic-ga>

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

Lab 5

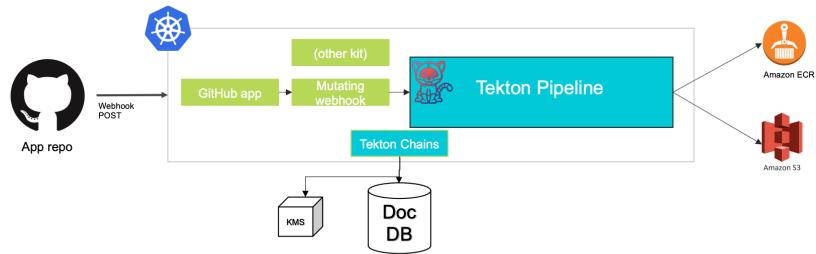
SLSA Level 3 Provenance





SolarWinds Project Trebuchet

Pipeline With Attestations



NDC { London }

@nielstanis@infosec.exchange

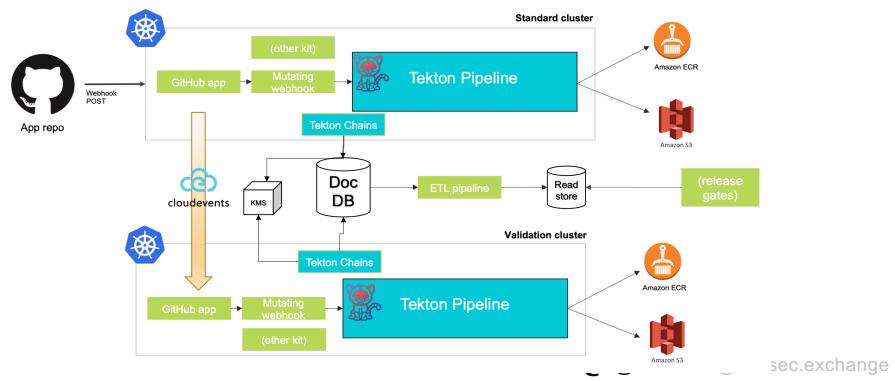
https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf

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



SolarWinds Project Trebuchet

Reading Results



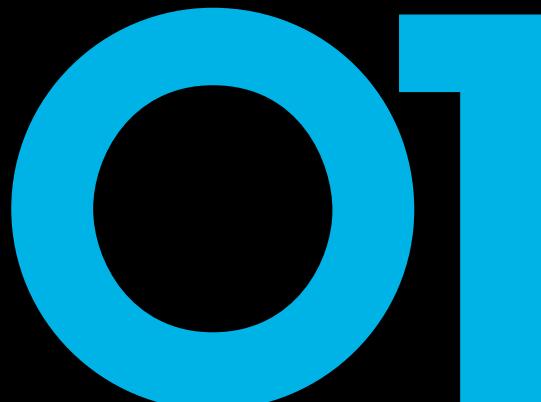
https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf

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

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

Lab 6

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 discusses the introduction of the docker-sbom command in Docker Desktop 4.7.0, which displays the Software Bill Of Materials (SBOM) for any Docker image. The Docker logo is visible in the top right corner of the browser window.

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

NDC { London }

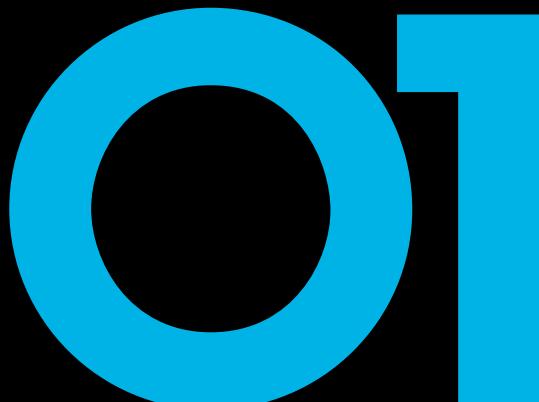
@nielstanis@infosec.exchange

<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

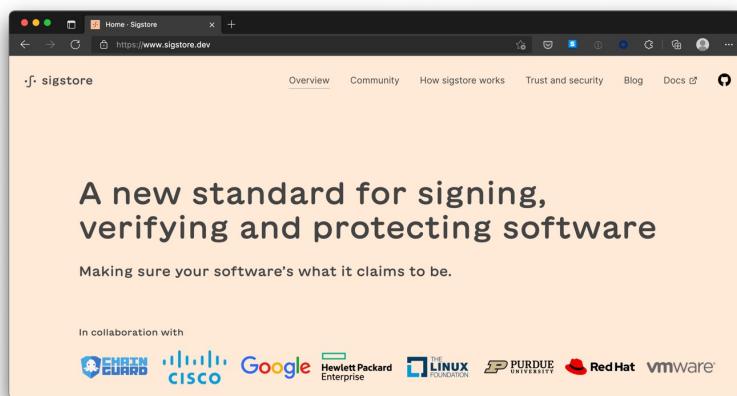
Lab 7

Docker SBOM with Anchor
and Syft



Signing artifacts

0101
0101



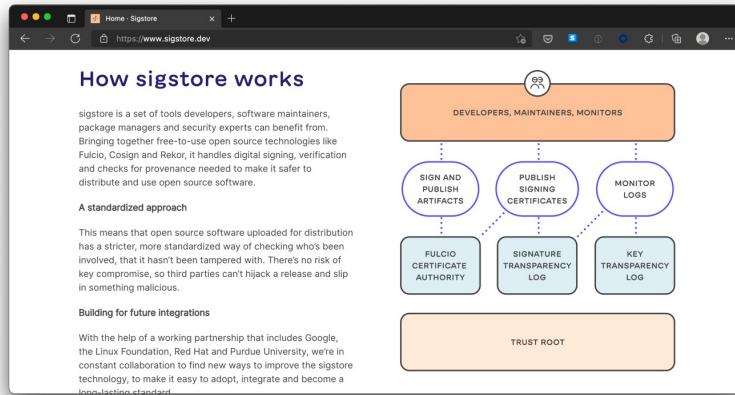
NDC { London }

@nielstanis@infosec.exchange

<https://sigstore.dev>

0101
0101

Signing artifacts



NDC { London }

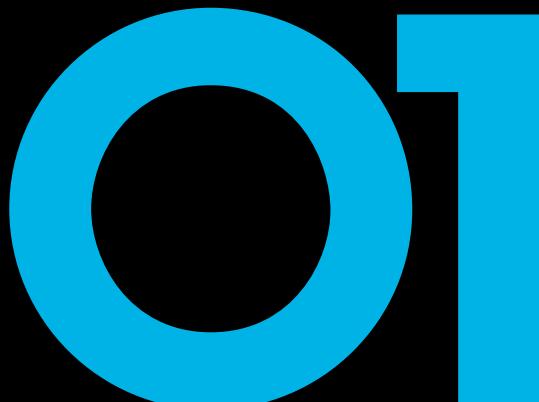
@nielstanis@infosec.exchange

<https://sigstore.dev>

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

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

NDC { London }

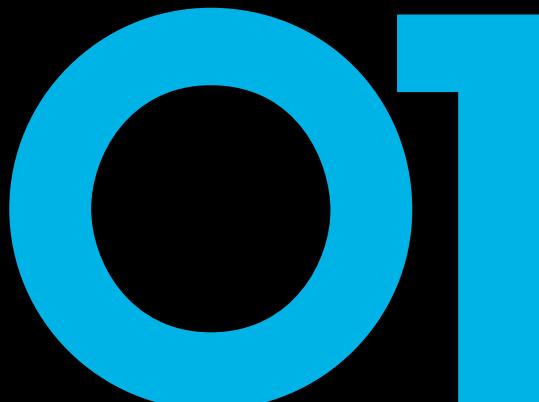
@nielstanis@infosec.exchange

<https://sigstore.dev>

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

Lab 9

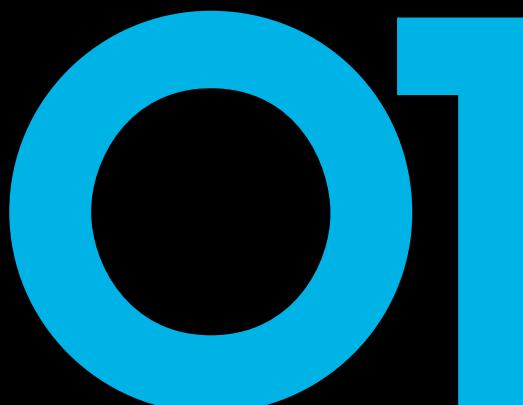
Cosign on Docker in
GitHub Actions



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

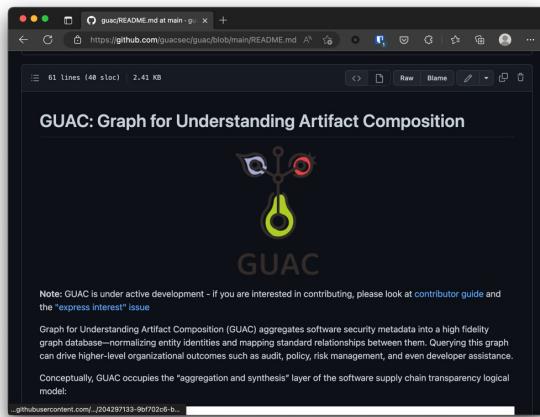
Lab 10

You got HACKED!



GUAC: Graph for Understanding Artifact Composition

0101
0101



NDC { London }

@nielstanis@infosec.exchange

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

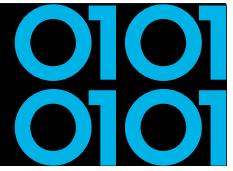


Conclusion

- 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 { London }

@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 { London }

@nielstanis@infosec.exchange

01010101010101010101010101010101
01 VERACODE 010101010101010101010101
01010101010101010101010101010101

Thanks! Questions?

<https://github.com/niestanis/ndclondon2023-supplychainws>
niestanis at veracode.com
@niestanis@infosec.exchange on Mastodon

