
NDC { London }
Securing your .NET application software supply-chain
Niels Tanis
VERACODE
0101
0101

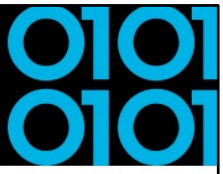
Who am I?

- Niels Tanis
- Principal Security Researcher @ Veracode
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - ISC² CSSLP
  - Research on static analysis for .NET apps

Picture is from Veracode report/site:

https://www.veracode.com/sites/default/files/pdf/resources/ipapers/everything-you-need-to-know-open-source-risk/index.html

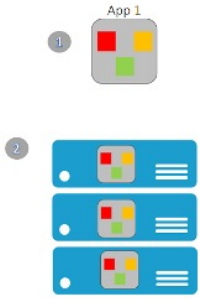
Agenda

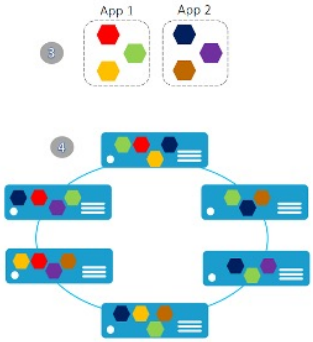


- Definition Software Supply-Chain
- Securing the Software Supply-Chain
  - Developer & Source
  - 3rd Party Libraries
  - Build & Release
- Conclusion and Q&A

Evolution in Software Architecture

- Monolith
- Microservices
- Serverless
  Cloud-Native





NDC { London }

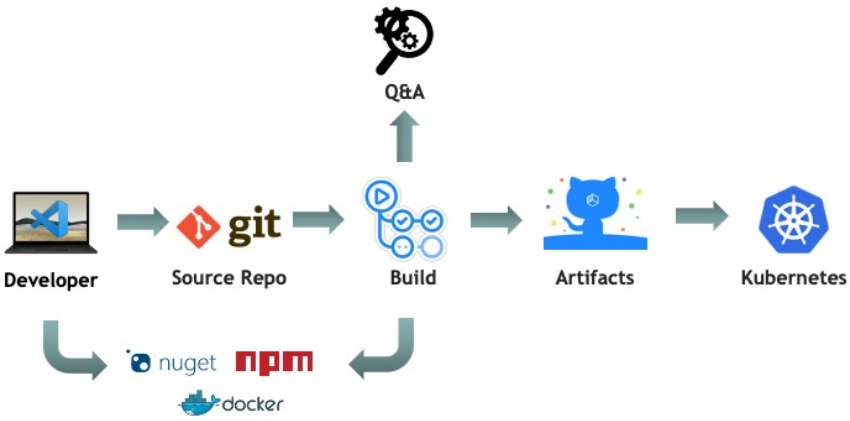
@nielstanis

What is a Supply Chain?

NDC { Istanis

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?

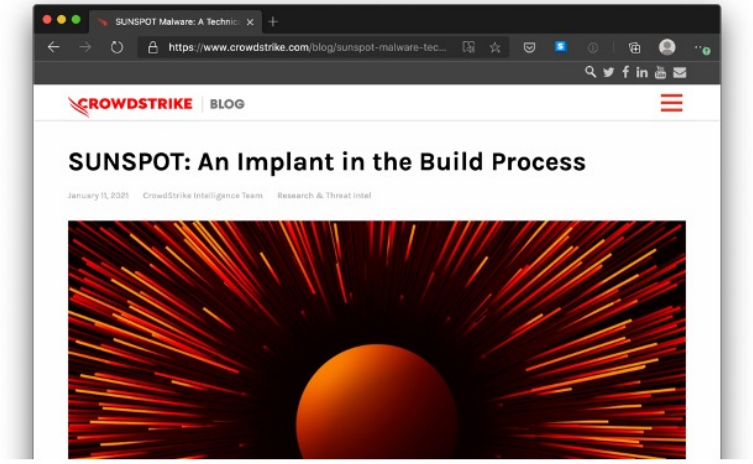
Software Supply Chain



NDC { London }

@nielstanis

SolarWinds SunSpot



NDC { London }

@nielstanis

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/

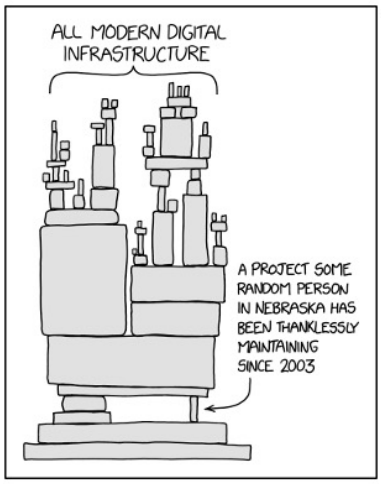
Kaseya

NDC { London }

@nielstanis

https://blog.sonatype.com/kaseya-ransomware-supply-chain

XKDC - Dependency



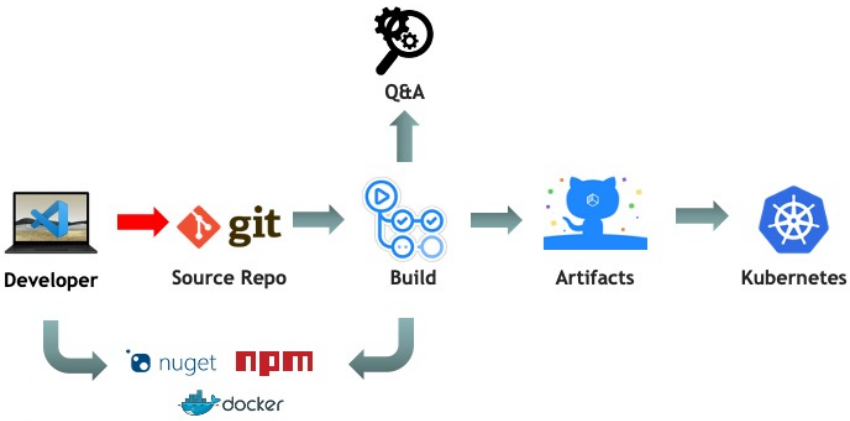
https://xkcd.com/2347/

NDC { London }

@nielstanis

https://xkcd.com/2347/

Software Supply Chain



NDC { London }

@nielstanis

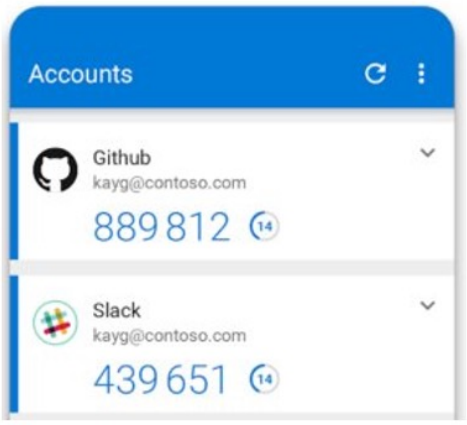
GitHub account

NDC { London }

@nielstanis

https://www.zdnet.com/article/canonical-github-account-hacked-ubuntu-source-code-safe/

Use MFA on source-repository

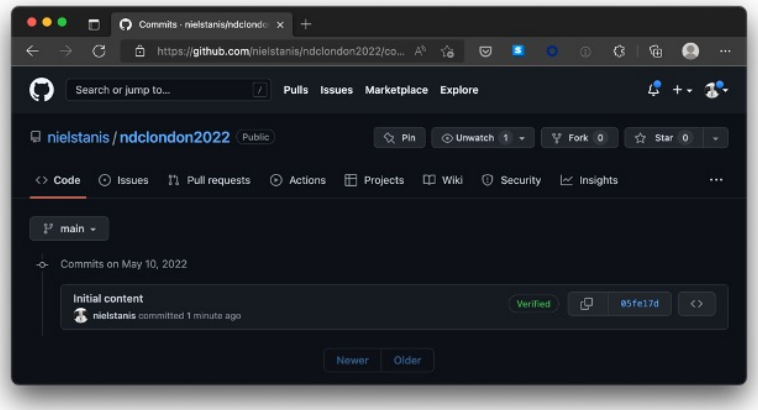


NDC { London }

@nielstanis

https://help.github.com/en/github/authenticating-to-github/configuring-two-factor-authentication

GIT Commit Signing

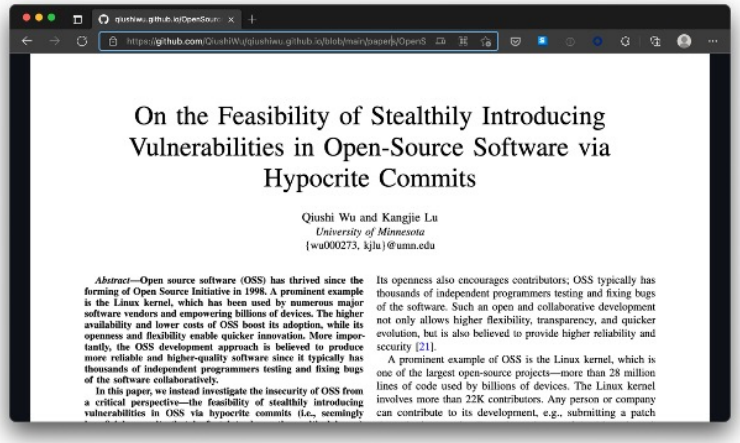


NDC { London }

@nielstanis

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

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

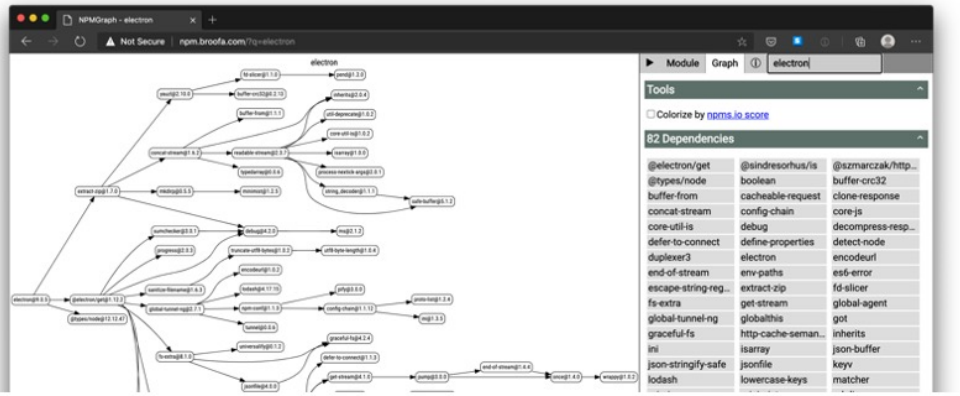
Its openness also encourages contributors; OSS typically has thousands of independent programmers testing and fixing bugs of the software. Such an open and collaborative development not only allows higher flexibility, transparency, and quicker evolution, but is also believed to provide higher reliability 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 contributoes. Any person or company can contribute to its development, e.g., submitting a patch

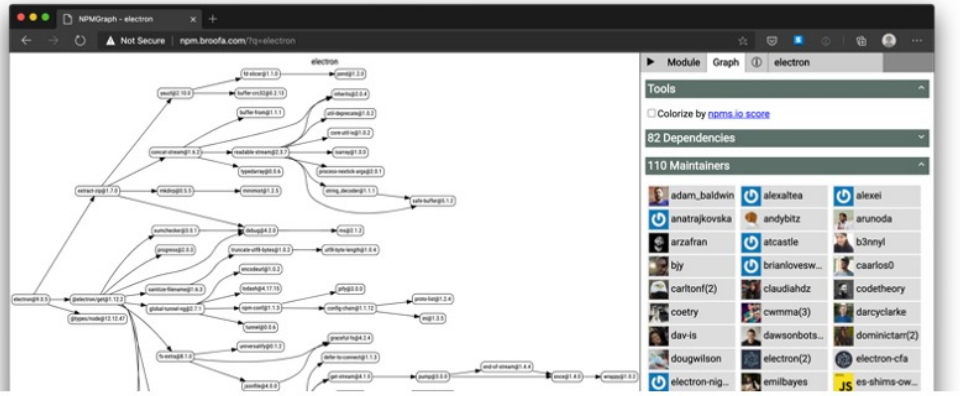
https://github.com/QiushiWu/qiushiwu.github.io/blob/main/papers/OpenSourceInsecurity.pdf

https://securitylab.github.com/research/octopus-scanner-malware-open-source-supply-chain

Visual Studio Code



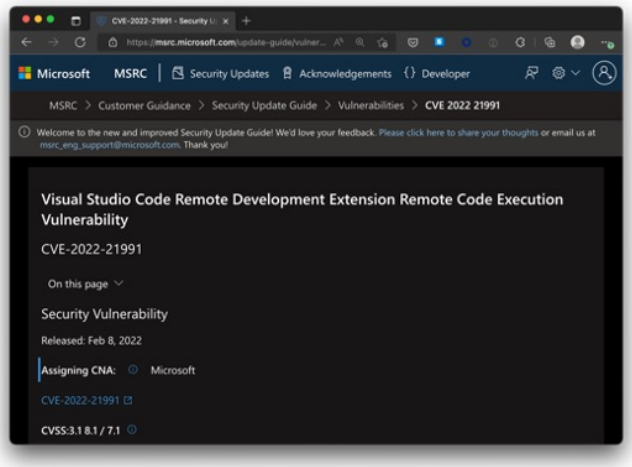
Visual Studio Code



NDC { London }

@nielstanis

Visual Studio Code



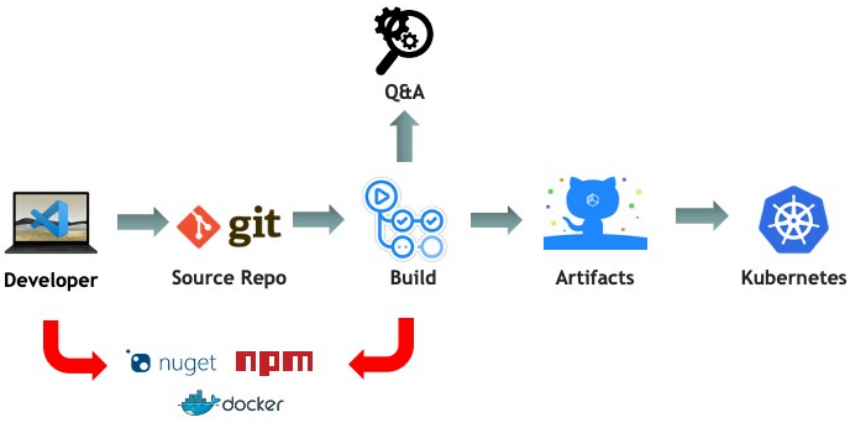
https://msrc.microsoft.com/update-guide/vulnerability/CVE-2022-21991

Visual Studio Code

NDC { London }

@nielstanis

3rd Party Libraries

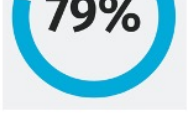


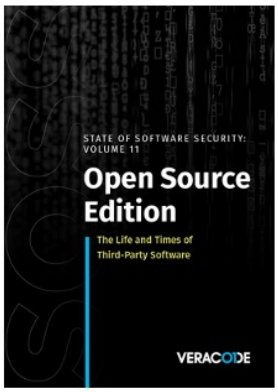
NDC { London }

@nielstanis

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."



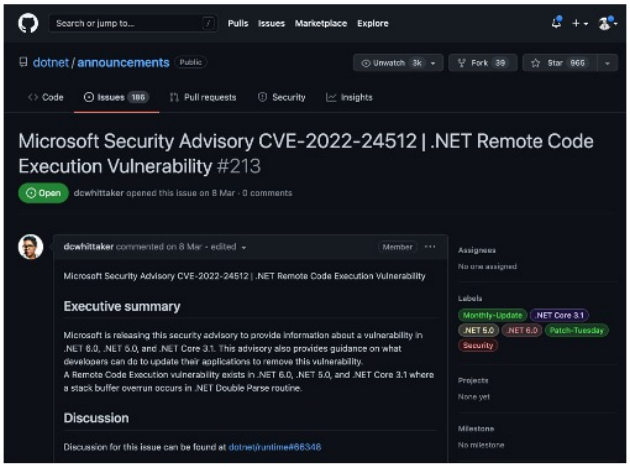


NDC { London }

@nielstanis

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

Vulnerabilities in libraries

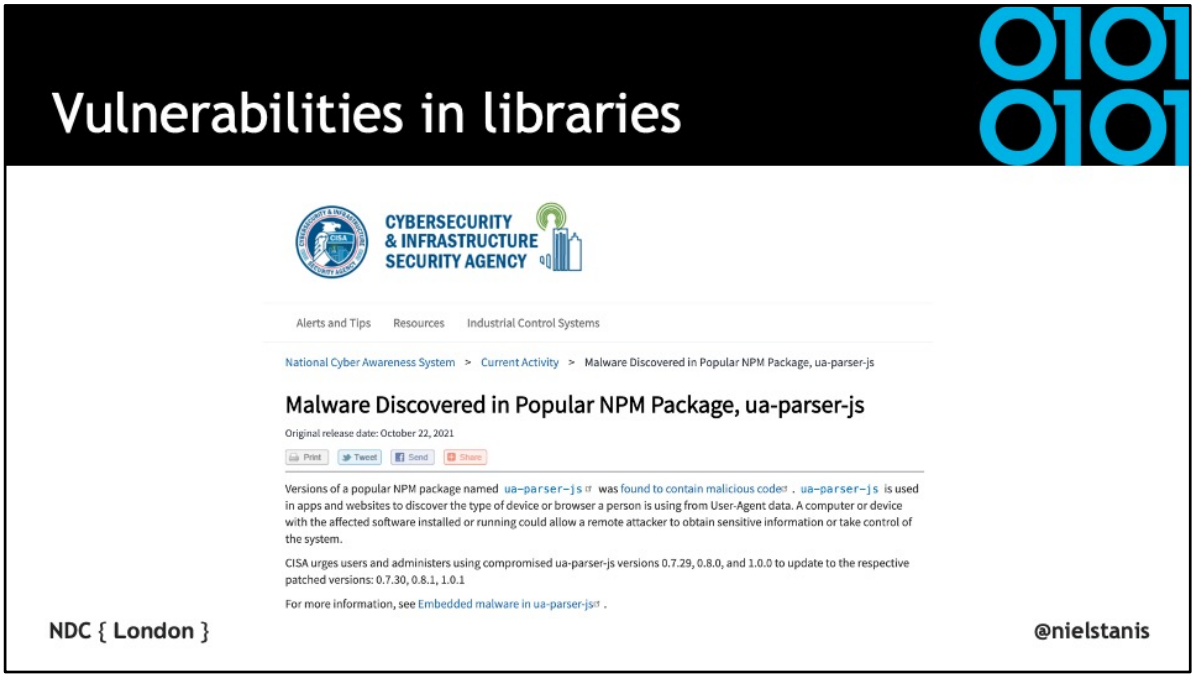


NDC { London }

@nielstanis

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

Vulnerabilities in libraries

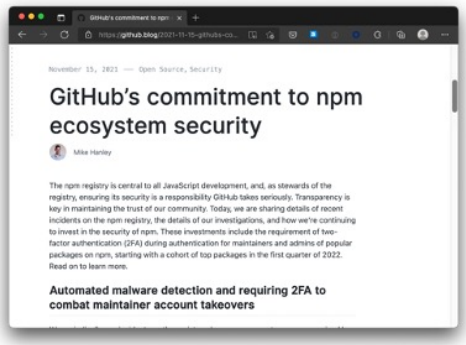


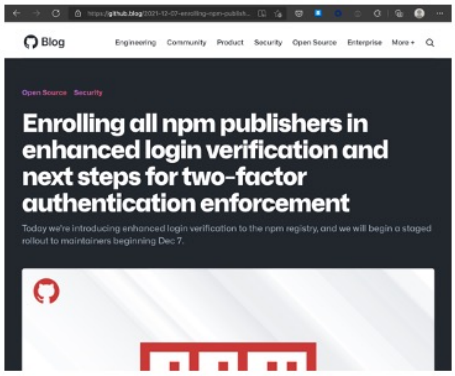
NDC { London }

@nielstanis

https://us-cert.cisa.gov/ncas/current-activity/2021/10/22/malware-discovered-popular-npm-package-ua-parser-js
https://portswigger.net/daily-swig/popular-npm-package-ua-parser-js-poisoned-with-cryptomining-password-stealing-malware

Vulnerabilities in libraries



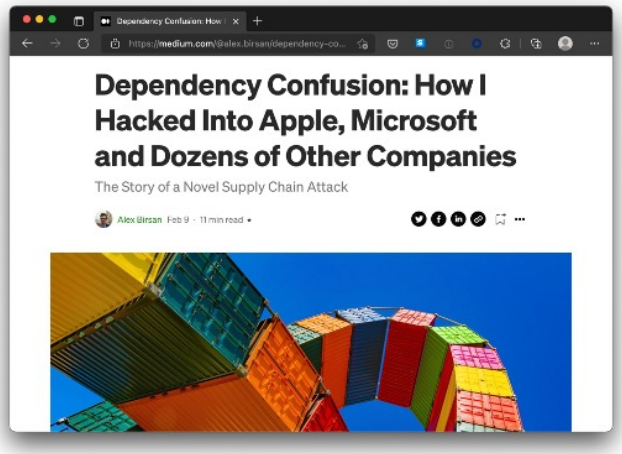


NDC { London }

@nielstanis

https://github.blog/2021-11-15-githubs-commitment-to-npm-ecosystem-security/

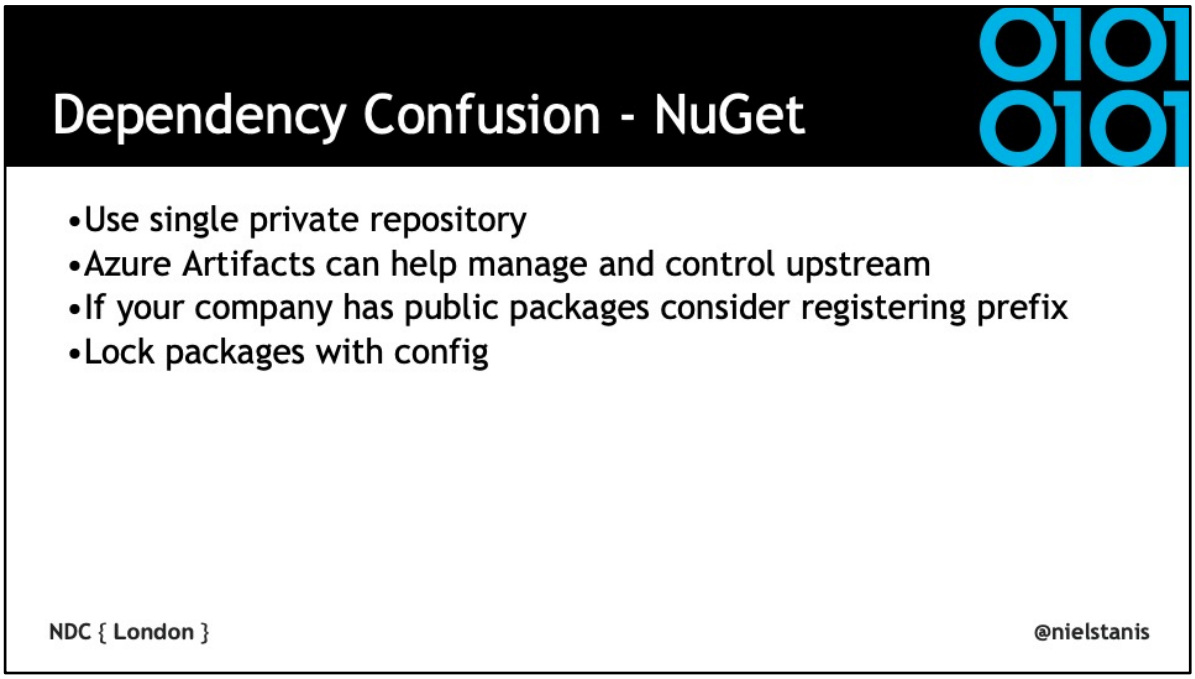
Dependency Confusion



NDC { London }

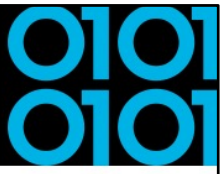
@nielstanis

https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610



https://azure.microsoft.com/nl-nl/resources/3-ways-to-mitigate-risk-using-private-package-feeds/
https://azure.microsoft.com/mediahandler/files/resourcefiles/3-ways-to-mitigate-risk-using-private-package-feeds/3%20Ways%20to%20Mitigate%20Risk%20When%20Using%20Private%20Package%20Feeds%20-%20v1.0.pdf

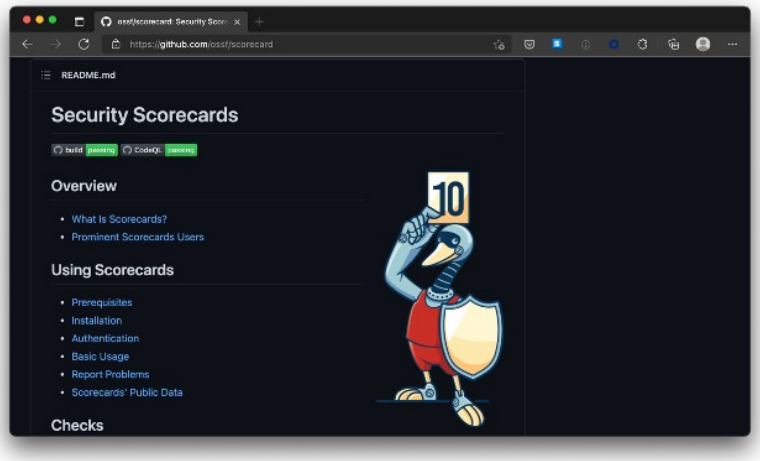
3rd Party Libraries



- Intent of library, know what's inside!
- Keep in mind that's a transitive list of dependencies
- Look out for talk on 'Sandboxing .NET Assemblies' I gave two weeks ago at NDC Porto!

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

Security Scorecards - OpenSSF

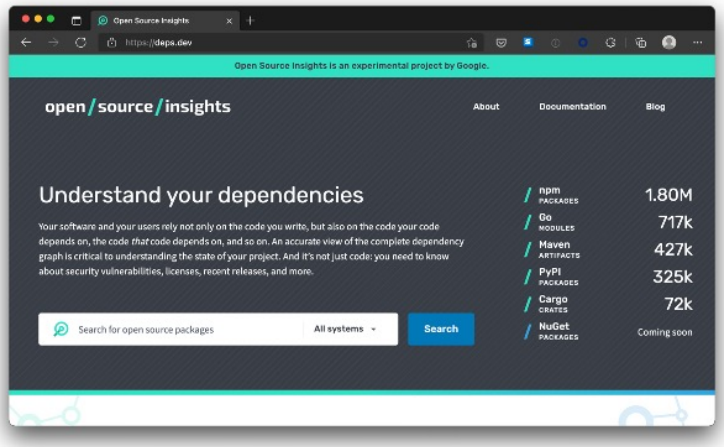


NDC { London }

@nielstanis

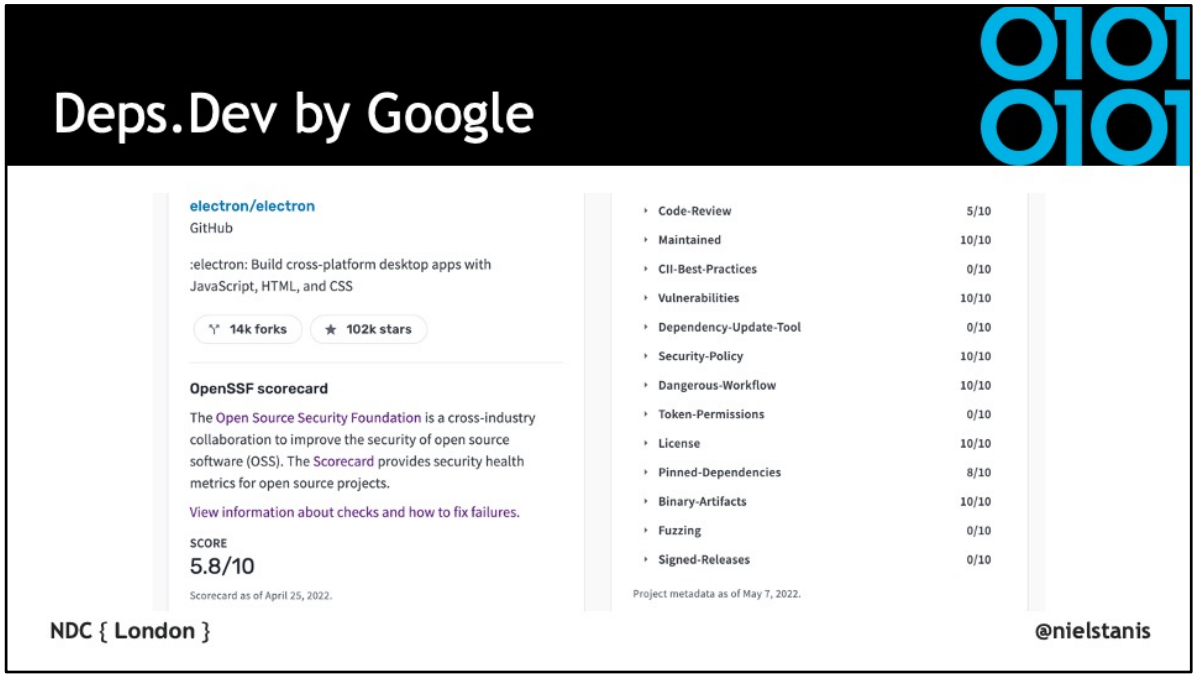
https://github.com/ossf/scorecard

Deps.Dev by Google



https://deps.dev/

Deps.Dev by Google

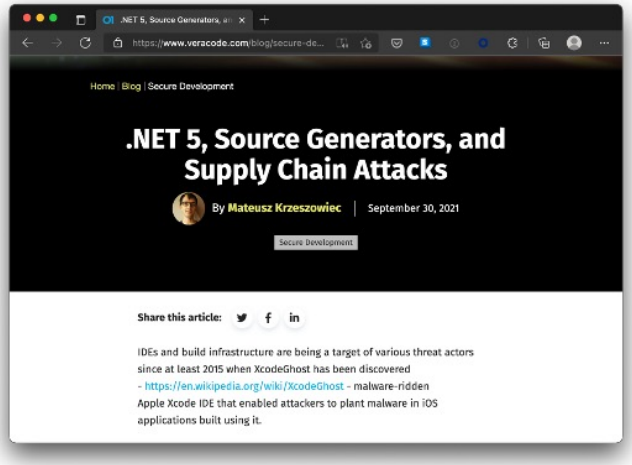


NDC { London }

@nielstanis

https://deps.dev/npm/electron

Source Generators

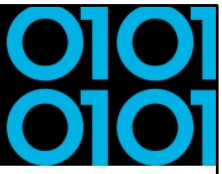


NDC { London }

@nielstanis

https://www.veracode.com/blog/secure-development/net-5-source-generators-and-supply-chain-attacks

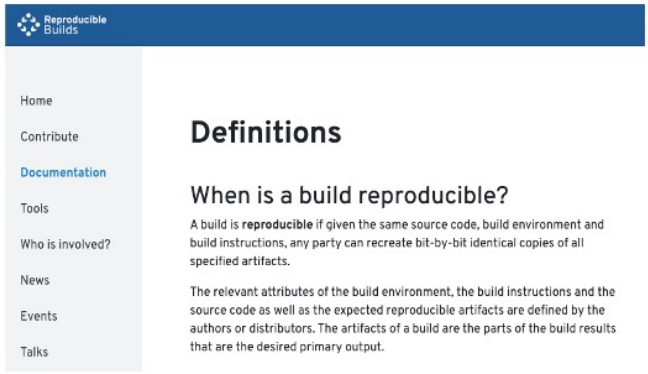
Source Generators



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

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

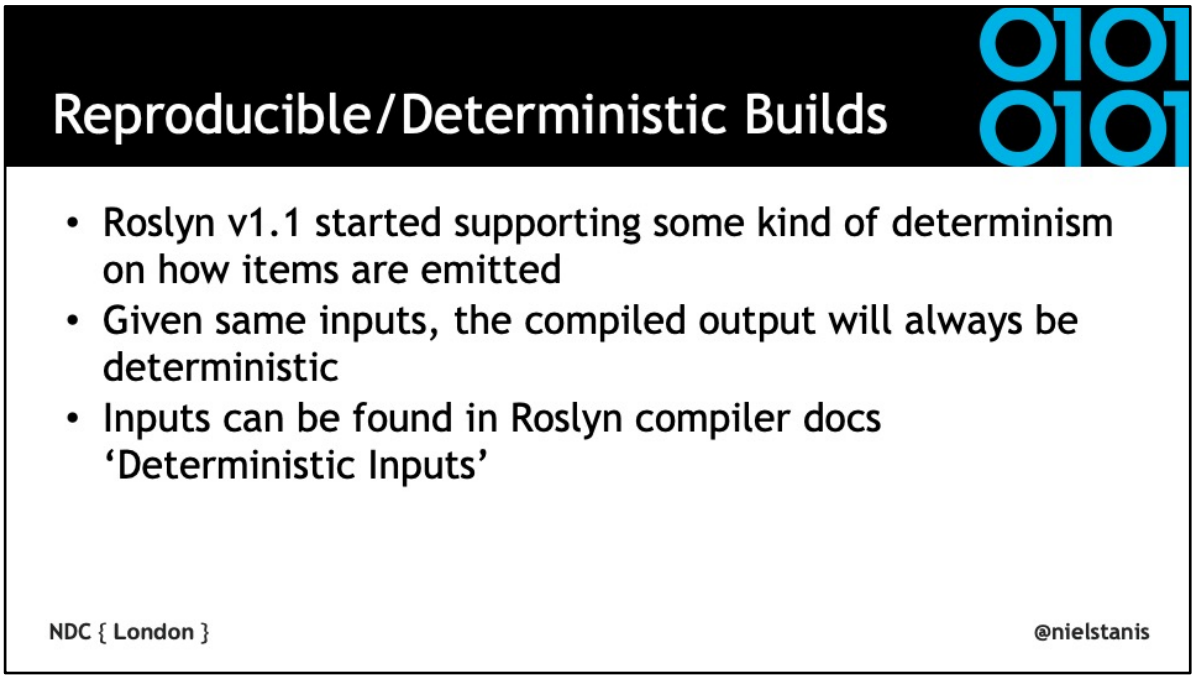
Reproducible/Deterministic Builds



NDC { London }

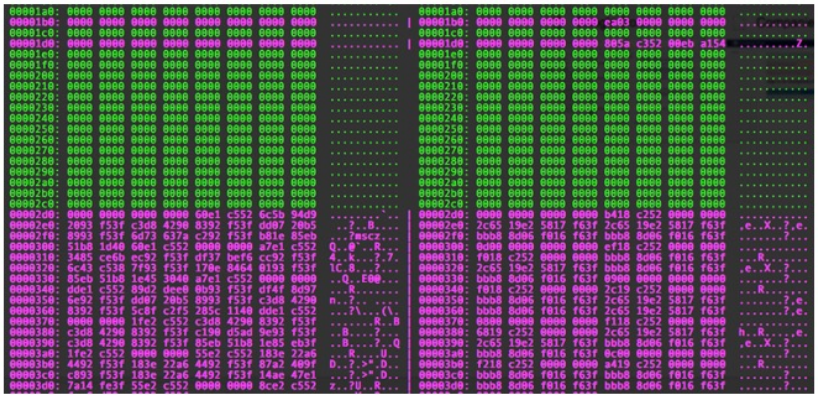
@nielstanis

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



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

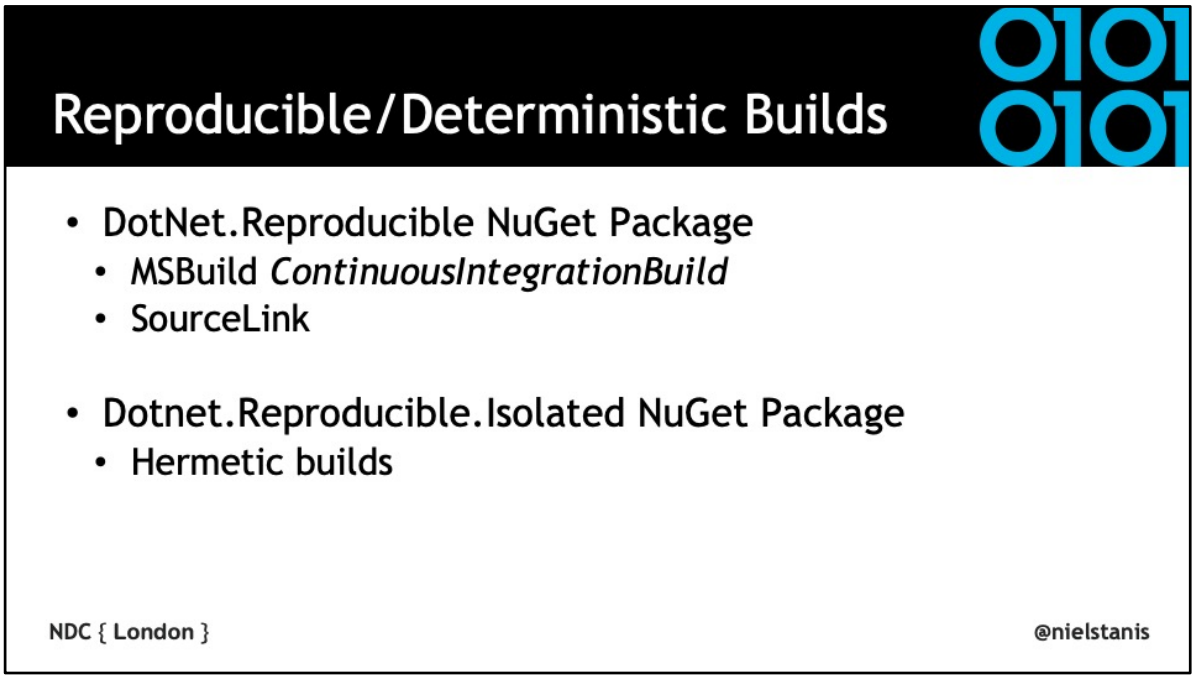
Reproducible Build .NET6



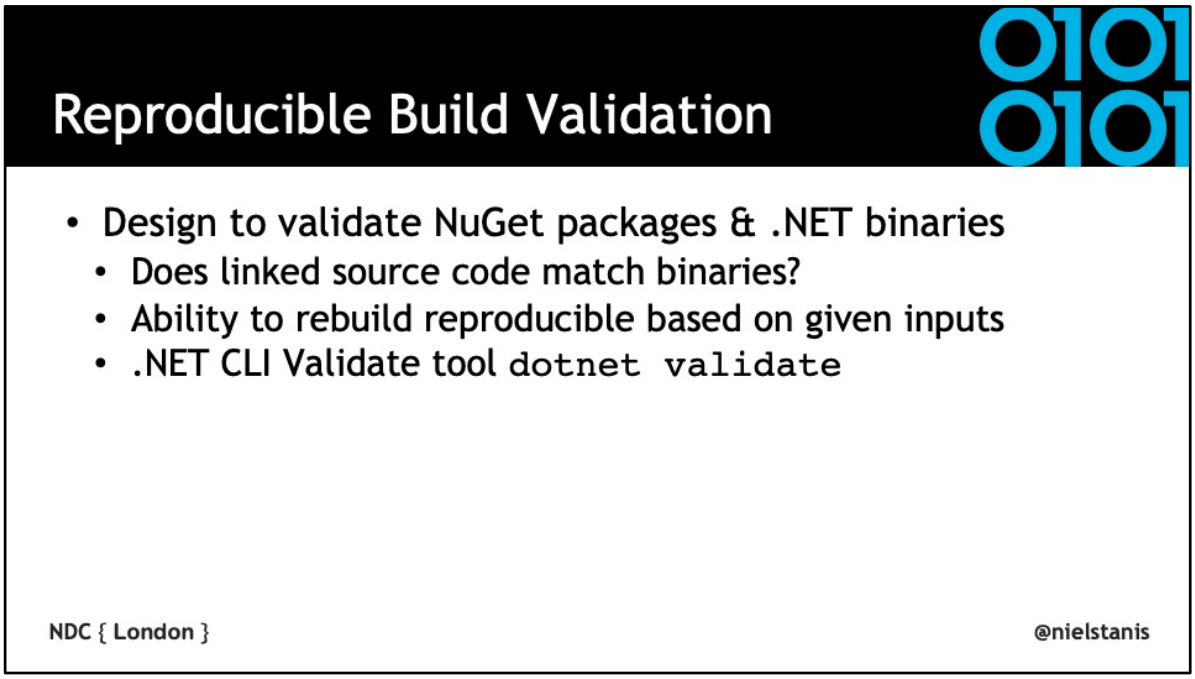
NDC { London }

@nielstanis

https://www.tabsoverspaces.com/233662-changing-paths-in-pdb-files-for-source-files-and-pdb-file-path-in-dll-as-well
<DebugOutput> in CSPROJ demo

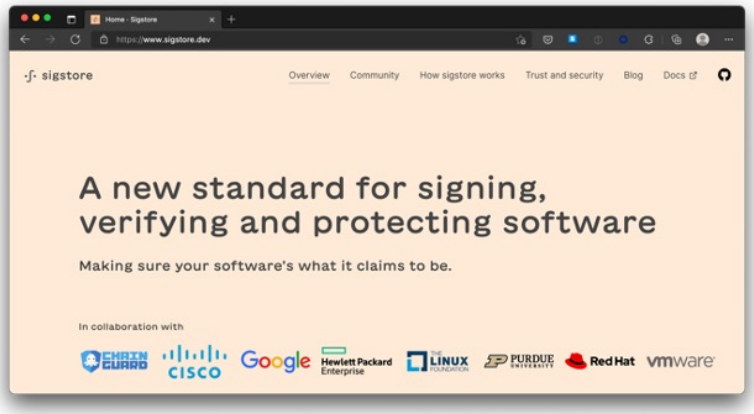


https://blog.paranoidcoding.com/2016/04/05/deterministic-builds-in-roslyn.html
https://github.com/dotnet/roslyn/blob/master/docs/compilers/Deterministic%20Inputs.md
https://devblogs.microsoft.com/dotnet/producing-packages-with-source-link/
https://github.com/dotnet/reproducible-builds



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

Signing artifacts

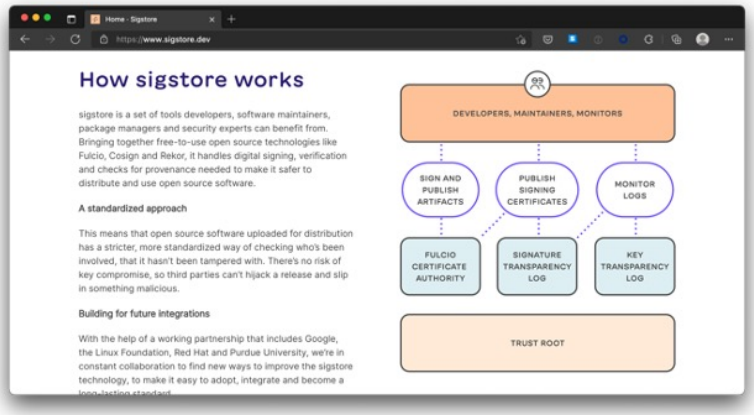


NDC { London }

@nielstanis

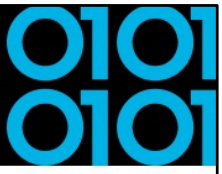
https://sigstore.dev

Signing artifacts



https://sigstore.dev

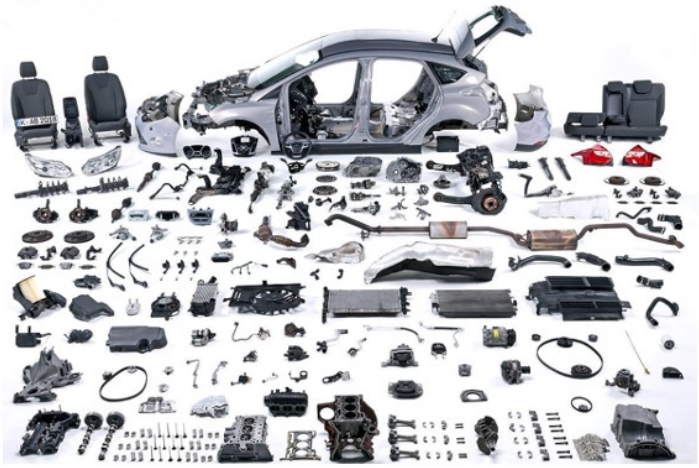
Signing artifacts



- Cosign can be used for sigining Docker images
- It can do keyless signing based on OpenID Connect
- GitHub Actions have released OpenID Connect support since end 2021

https://sigstore.dev

Automotive Industry



NDC { London }

@nielstanis

Car Supply Chain

Tata Steel Factory

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

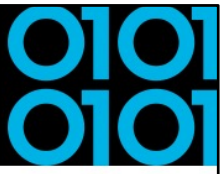
Bosch Factory

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

Renault Manufacturing

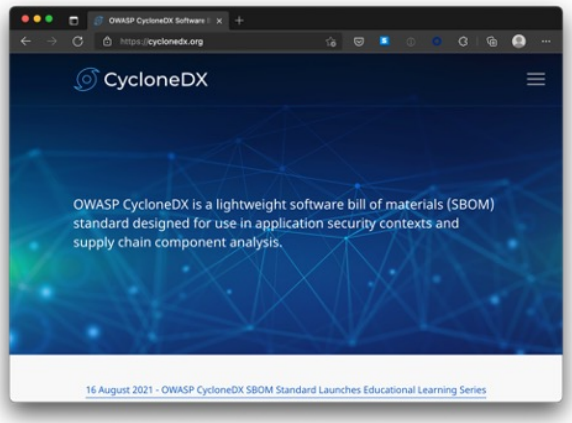
- Bosch Disk #45678
- Bosal Exhaust #RE9876
- Goodyear Tires #GY8877
- Kadjar VIN 1234567890

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)

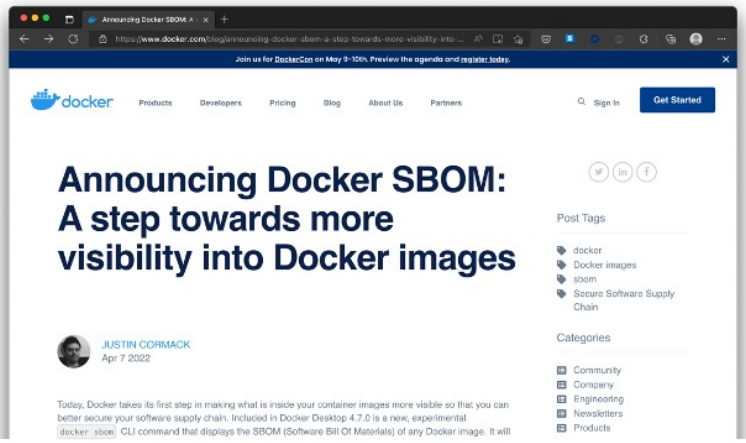


NDC { London }

@nielstanis

https://cyclonedx.org

Docker SBOM

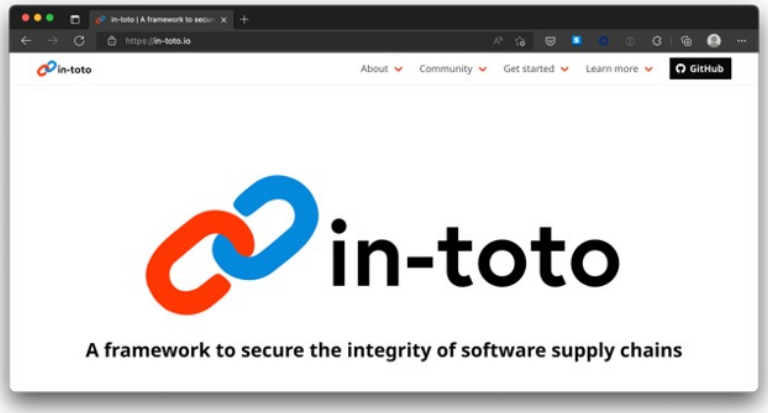


NDC { London }

@nielstanis

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

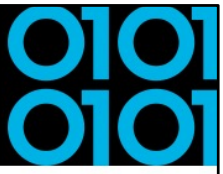
In-toto



NDC { London }

@nielstanis

In-Toto - Terminology

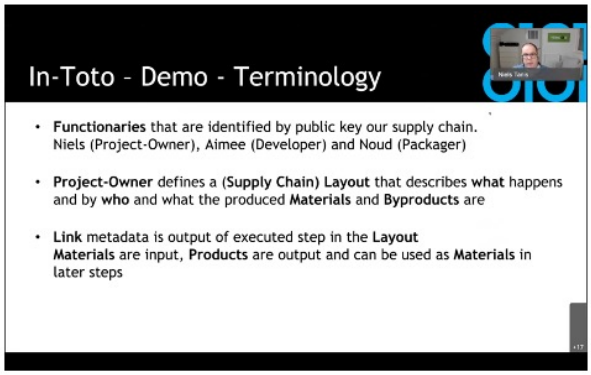


- **Functionaries** that are identified by public key our supply chain.
  For example, the Project-Owner, Developer, and Release Manager
- **Project-Owner** defines a **(Supply Chain) Layout** that describes **what** happens and by **who** and what the produced **Materials** and **Byproducts** are.
- **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

NDC { London } @nielstanis

https://in-toto.io/

In-Toto – Demo



NDC { London }

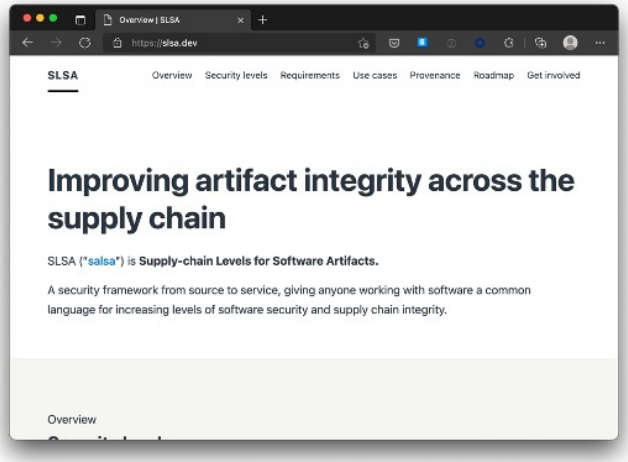
@nielstanis

https://youtu.be/fYCfB7MZPh4?t=2777

MyAwesomeWebApp Demo

- CycloneDX
- In-Toto
- Sigstore
- Docker SBOM

Google SLSA

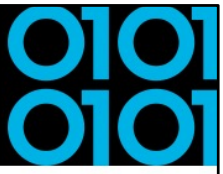


NDC { London }

@nielstanis

https://slsa.dev

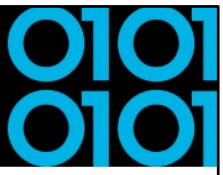
Google SLSA Levels



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.

https://slsa.dev

SLSA GitHub Action

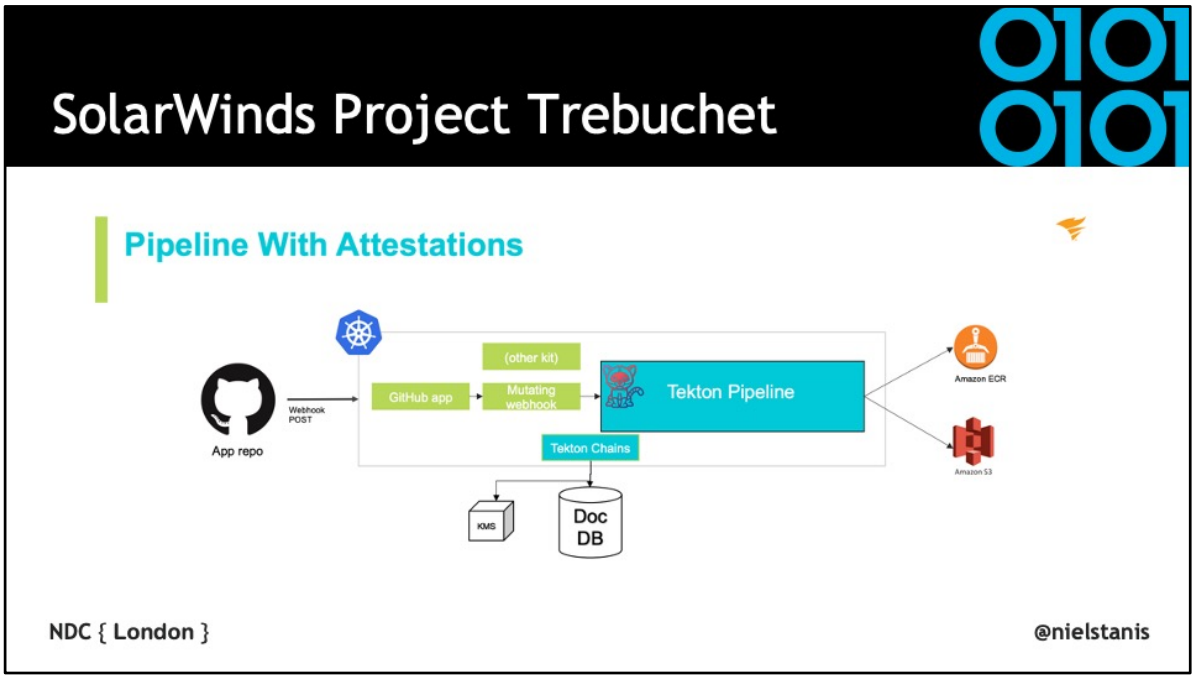


- Released April 2022

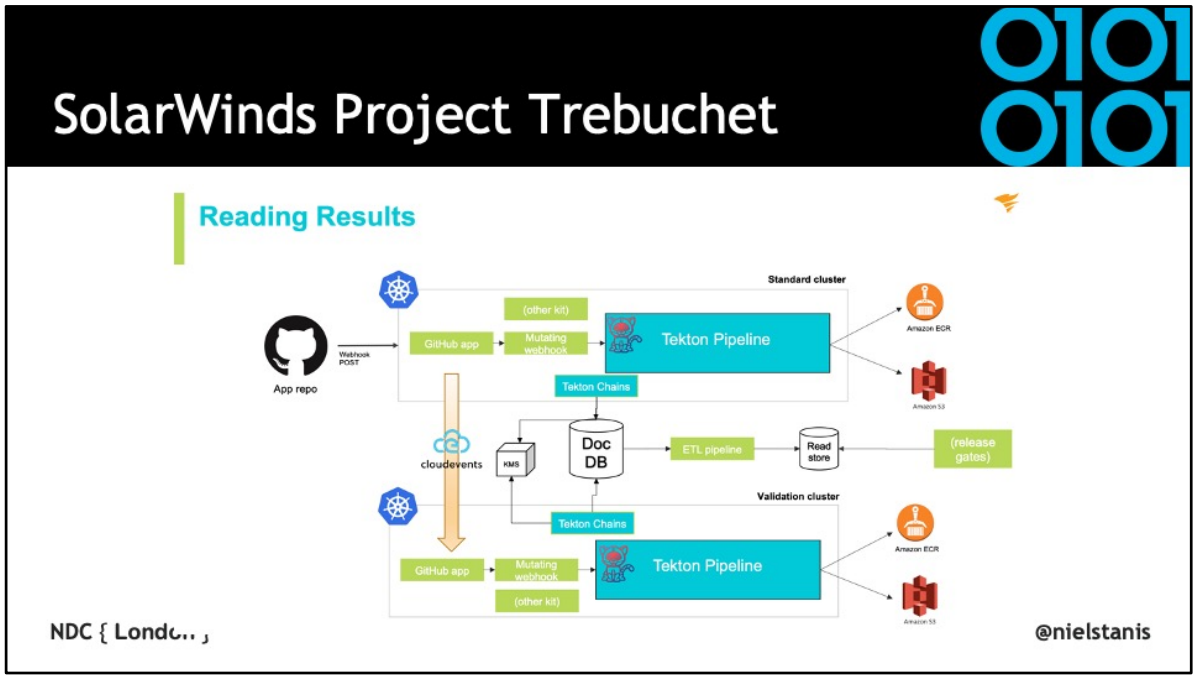
- SLSA level 2 provenance generator in GitHub Action
- SLSA level 3+ provenance generator for Go binaries

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

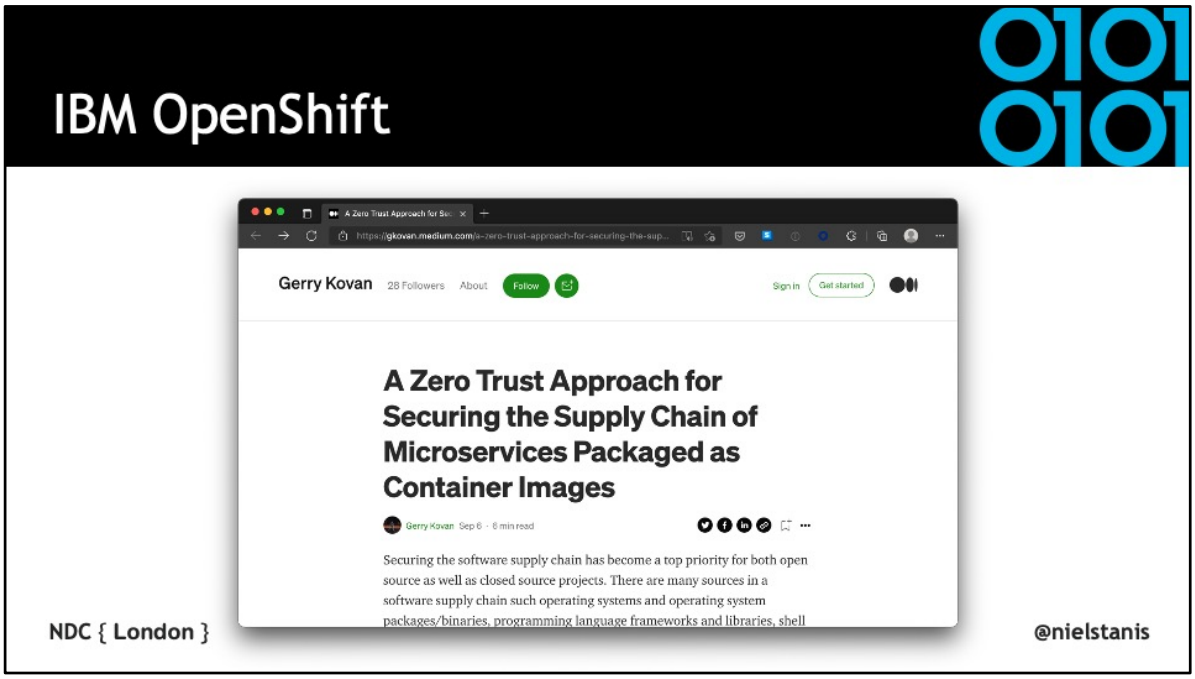
https://security.googleblog.com/2022/04/improving-software-supply-chain.html



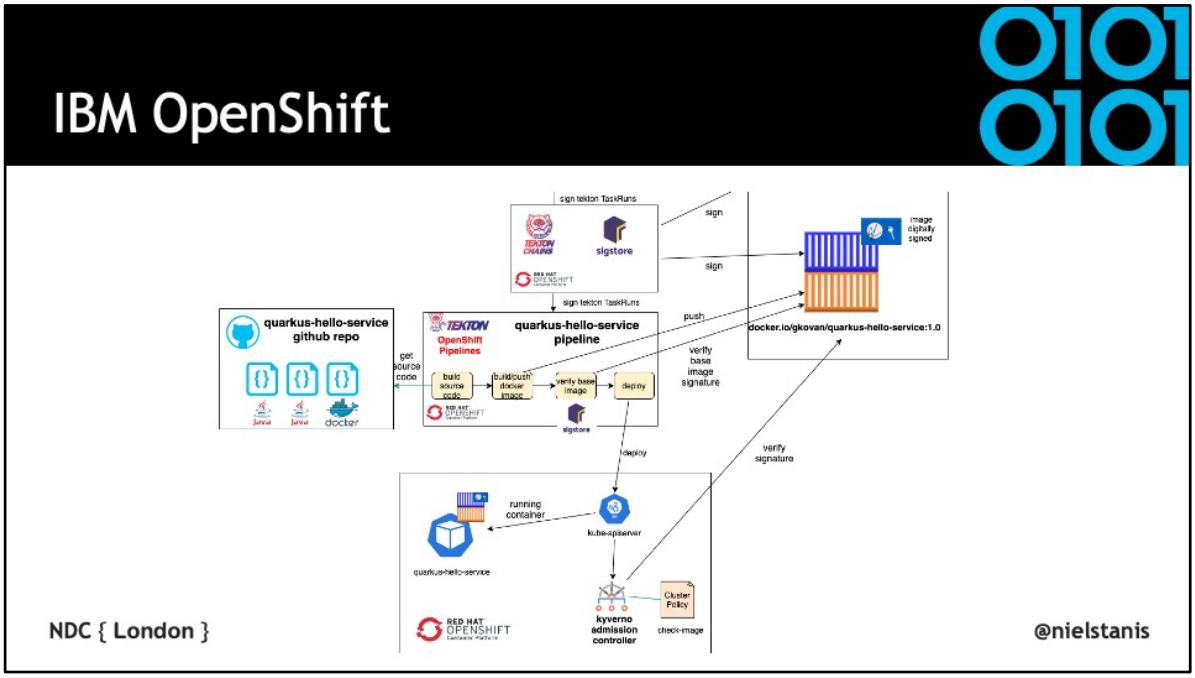
https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf
https://www.youtube.com/watch?v=1-tMRxqMwTQ



https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf
https://www.youtube.com/watch?v=1-tMRxqMwTQ



https://gkovan.medium.com/a-zero-trust-approach-for-securing-the-supply-chain-of-microservices-packaged-as-container-images-89d2f5b7293b



https://gkovan.medium.com/a-zero-trust-approach-for-securing-the-supply-chain-of-microservices-packaged-as-container-images-89d2f5b7293b

Grafeas and Kritis by Google

- Grafeas – Component Metadata API
  - Container Analysis API on Google Cloud Platform
- Kritis – Deployment Authorization for Kubernetes Apps
  - Binary Authorization on Google Cloud Platform

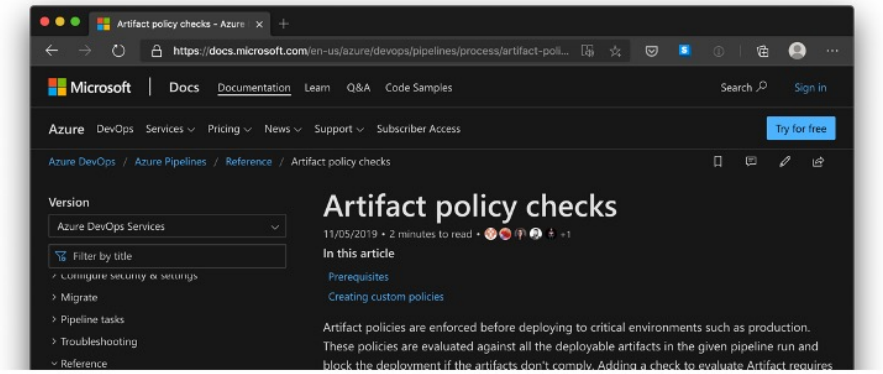
NDC { London }

@nielstanis

https://grafeas.io/
https://github.com/grafeas/kritis/blob/master/docs/binary-authorization.md

https://www.infoq.com/presentations/supply-grafeas-kritis/
https://www.youtube.com/watch?v=hOzH3mOApjs
https://www.youtube.com/watch?v=05zN-YQxEAM

Azure Pipelines Artifact Policy

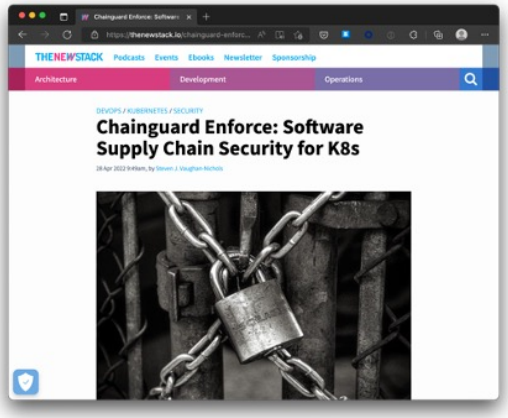


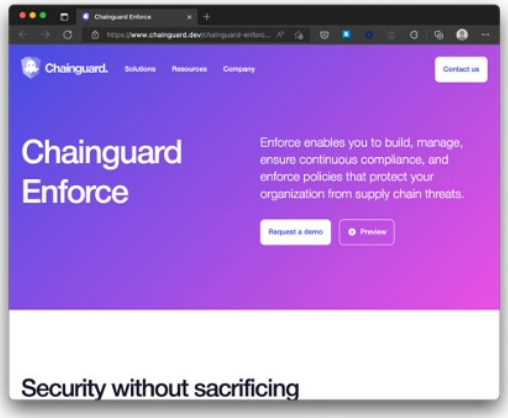
NDC { London }

@nielstanis

https://devblogs.microsoft.com/devops/secure-software-supply-chain-with-azure-pipelines-artifact-policies/
https://docs.microsoft.com/en-us/azure/devops/pipelines/process/artifact-policy?view=azure-devops

Chainguard Enforce



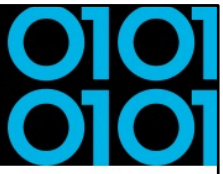


NDC { London }

@nielstanis

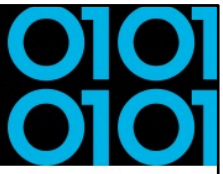
https://www.chainguard.dev/chainguard-enforce
https://thenewstack.io/chainguard-enforce-software-supply-chain-security-for-k8s/

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.

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!


VERACODE
Thanks! Questions?
https://github.com/nielstanis/ndclondon2022
ntanis at veracode.com
@nielstanis on Twitter