NDC { Sydney }

Securing your .NET application software supply-chain

Niels Tanis

Who am I?

- Niels Tanis
- Principal Security Researcher @ Veracode
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - ISC^2 CSSLP
  - Research on static analysis for .NET apps

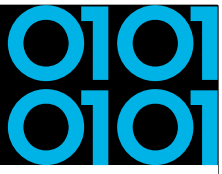
Securing your .NET application software supply-chain

NDC { Sydney }

@nielstanis

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

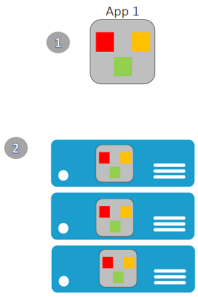
Agenda

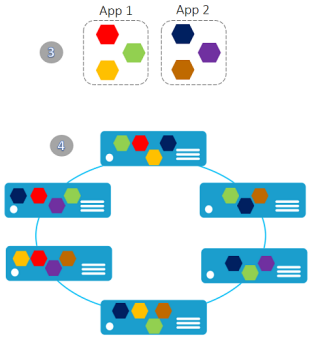


- Definition Software Supply-Chain
- Securing the Software Supply-Chain
  - Developer & Source
  - 3rd Party Libraries
  - Build & Release
- Conclusion and Q&A

Evolution in Software Architecture

- Monolith
- Microservices
- Serverless
  Cloud-Native

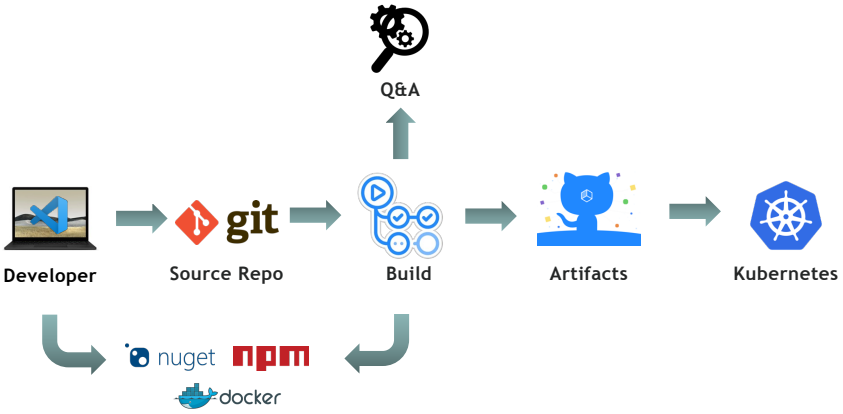




What is a Supply Chain?

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?

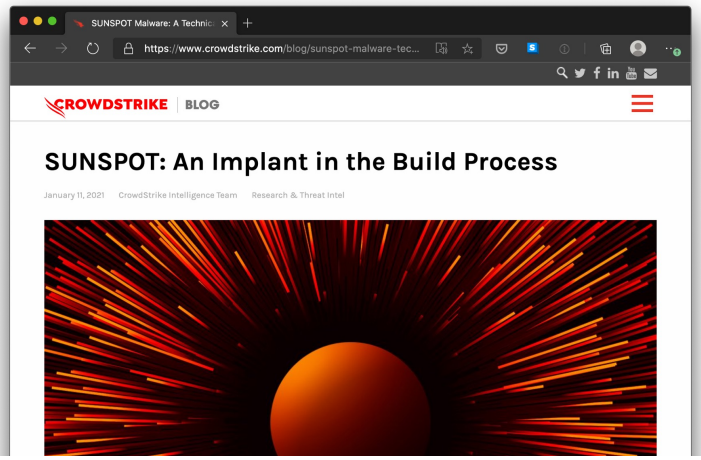
Software Supply Chain



NDC { Sydney }

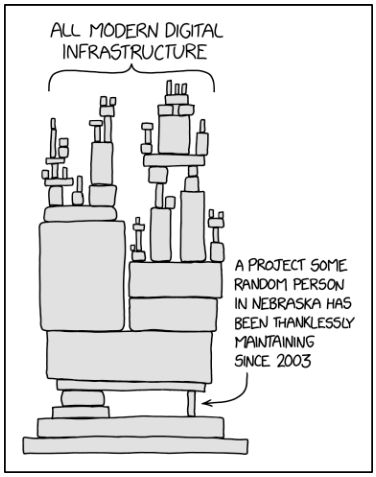
@nielstanis

SolarWinds SunSpot



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/

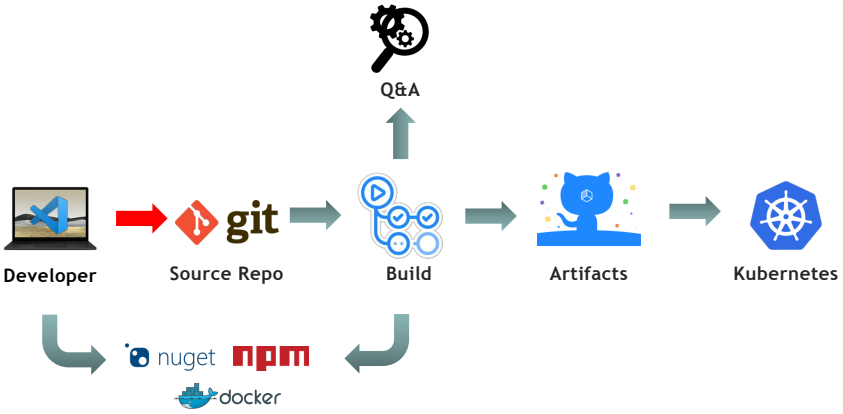
XKDC - Dependency



https://xkcd.com/2347/

https://xkcd.com/2347/

Software Supply Chain



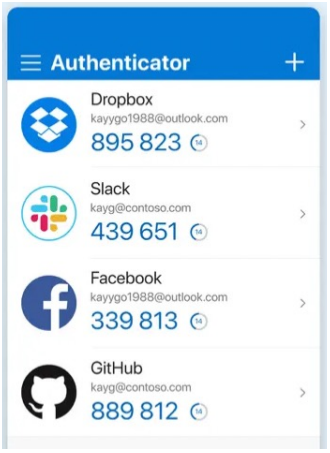
NDC { Sydney }

@nielstanis

GitHub account

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

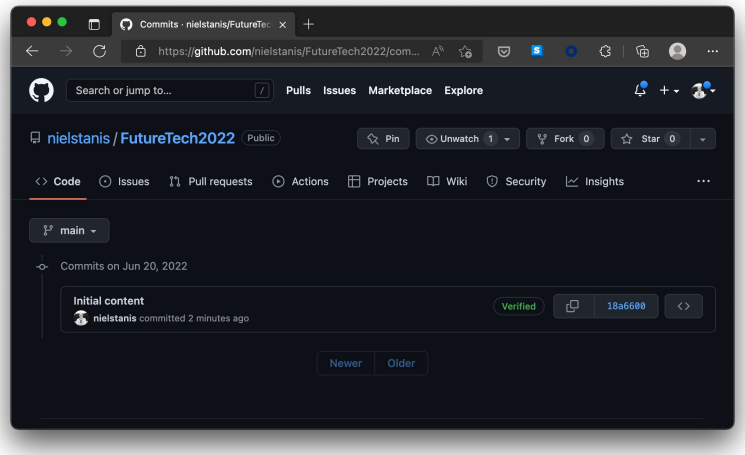
Use MFA on source-repository



Slide with larger he
when there is not a
be text heavy
The content text slid
bullets as preview, h
you do not need t
bullets. Text is pres
Trebuchet size 14
bullets may be rem
not needed.

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

GIT Commit Signing

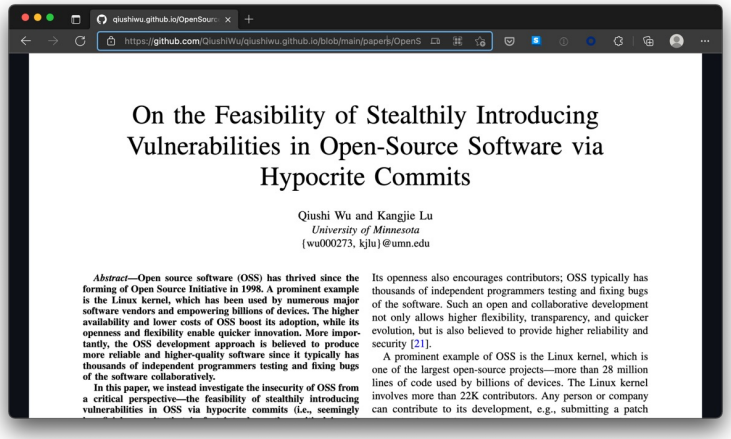


NDC { Sydney }

@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.**

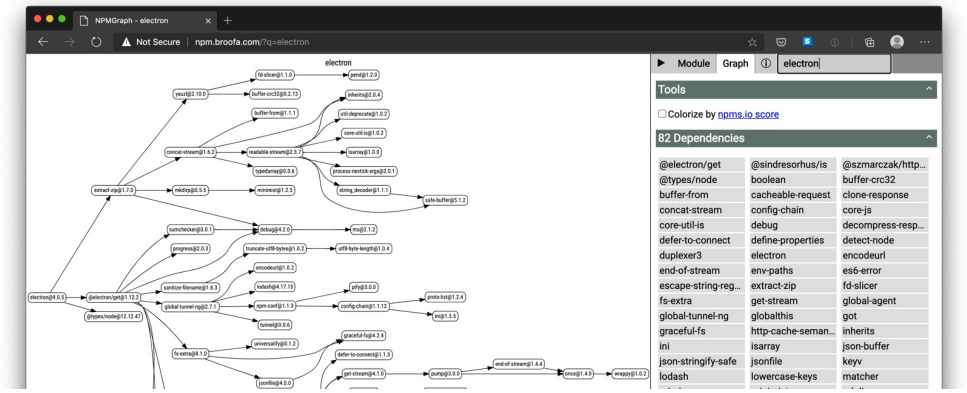
In this paper, we instead investigate the insecurity of OSS from a critical perspective—the feasibility of stealthily introducing vulnerabilities in OSS via hypocrite commits (i.e., seemingly

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].

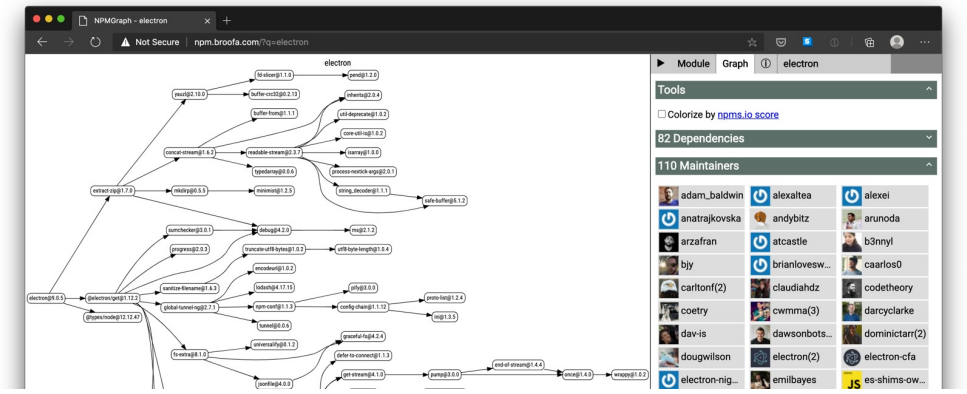
A prominent example of OSS is the Linux kernel, which is one of the largest open-source projects—more than 28 million lines of code used by billions of devices. The Linux kernel involves more than 22K contributors. Any person or company can contribute to its development, e.g., submitting a patch

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

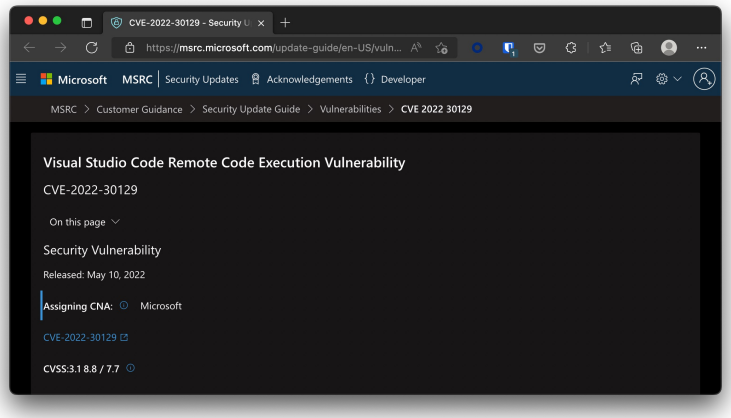
Visual Studio Code



Visual Studio Code



Visual Studio Code

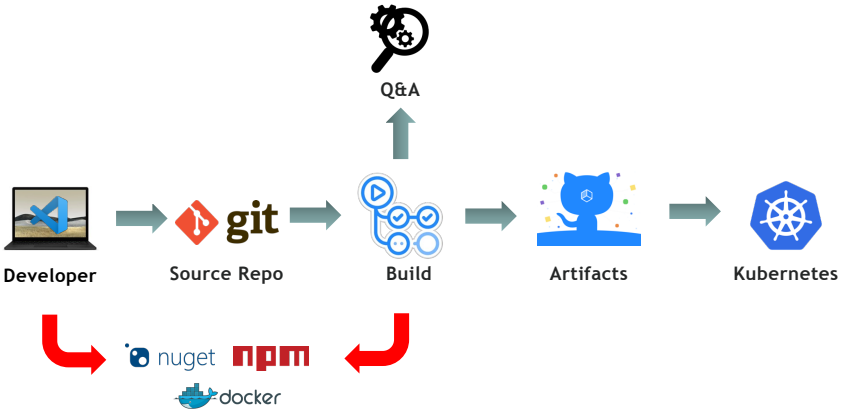


https://msrc.microsoft.com/update-guide/vulnerability/CVE-2022-30129

Visual Studio Code

https://www.bleepingcomputer.com/news/security/heres-how-a-researcher-broke-into-microsoft-vs-codes-github/

3rd Party Libraries

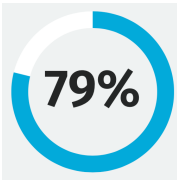


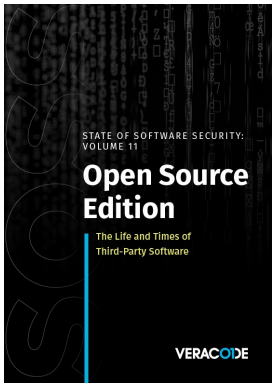
NDC { Sydney }

@nielstanis

State Of Software Security v11 2021

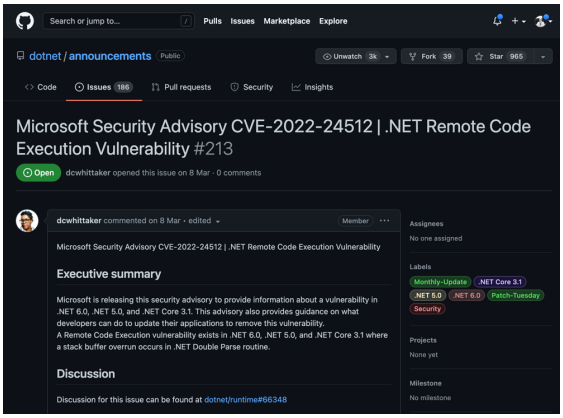
”Despite this dynamic landscape, 79 percent of the time, developers never update third-party libraries after including them in a codebase.”





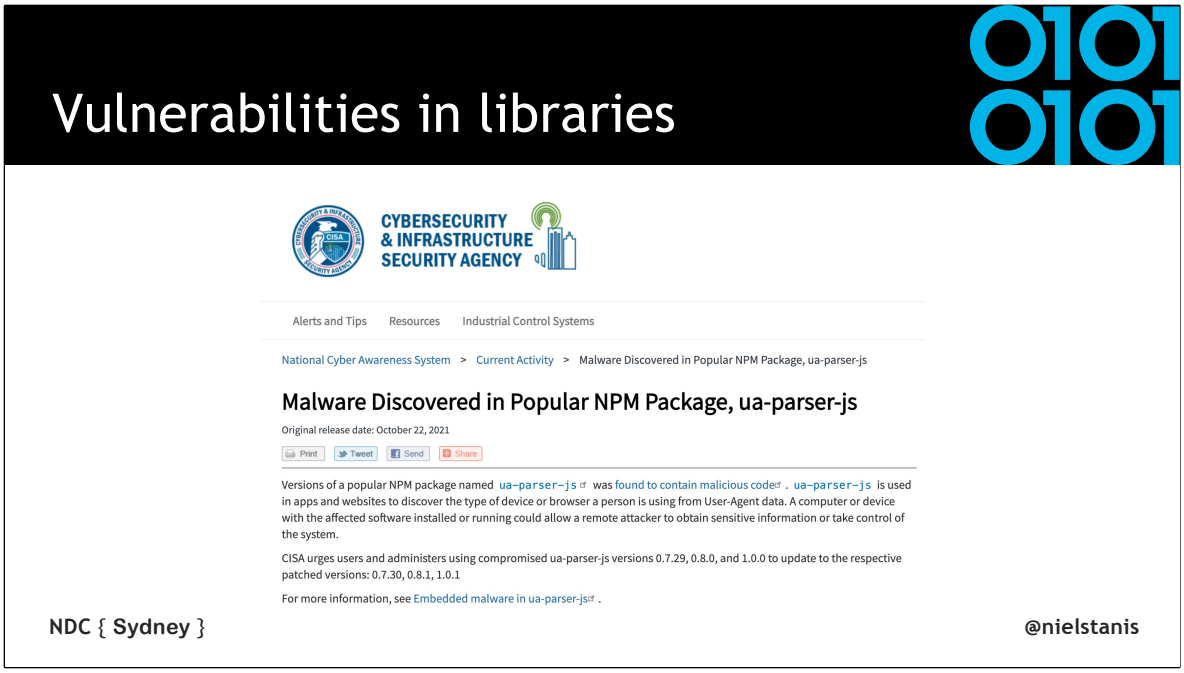
https://info.veracode.com/fy22-state-of-software-security-v11-open-source-edition.html

Vulnerabilities in libraries



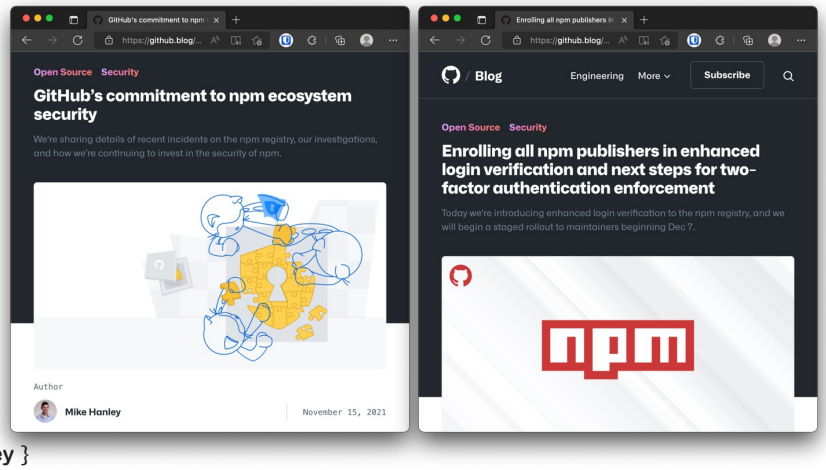
https://github.com/dotnet/announcements/issues/213

Vulnerabilities in libraries



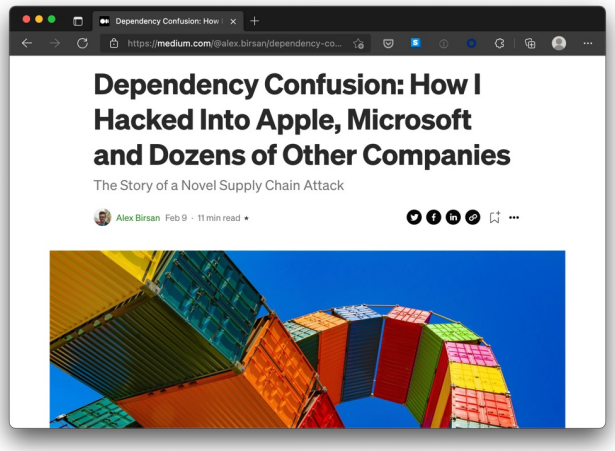
https://us-cert.cisa.gov/ncas/current-activity/2021/10/22/malware-discovered-popular-npm-package-ua-parser-js
https://portswigger.net/daily-swig/popular-npm-package-ua-parser-js-poisoned-with-cryptomining-password-stealing-malware

Vulnerabilities in libraries



https://github.blog/2021-11-15-githubs-commitment-to-npm-ecosystem-security/
https://github.blog/2021-12-07-enrolling-npm-publishers-enhanced-login-verification-two-factor-authentication-enforcement/

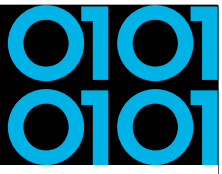
Dependency Confusion



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

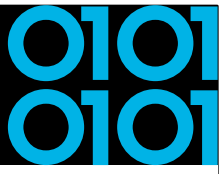
Dependency Confusion - NuGet



- Use single private repository
- Azure Artifacts can help manage and control upstream
- If your company has public packages consider registering prefix
- Lock packages with config

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

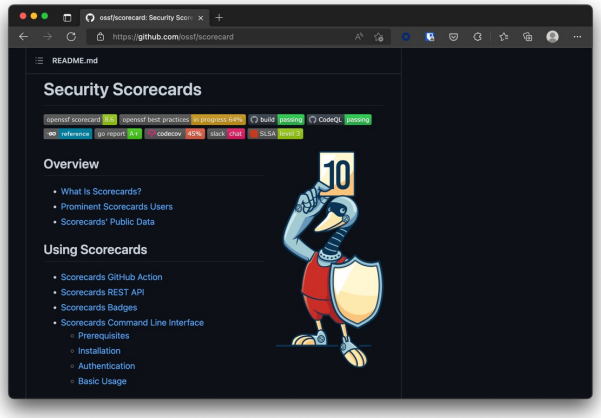


- Intent of library, know what's inside!
- Keep in mind that's a transitive list of dependencies
- My talk 'Sandboxing .NET Assemblies' Tomorrow @ 3PM in Room 4

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

Sandboxing .NET assemblies for fun, profit and of course security! - Niels Tanis - NDC Porto 2022 - YouTube

Security Scorecards - OpenSSF

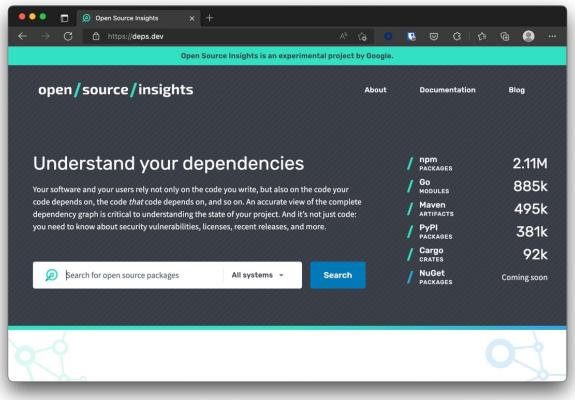


NDC { Sydney }

@nielstanis

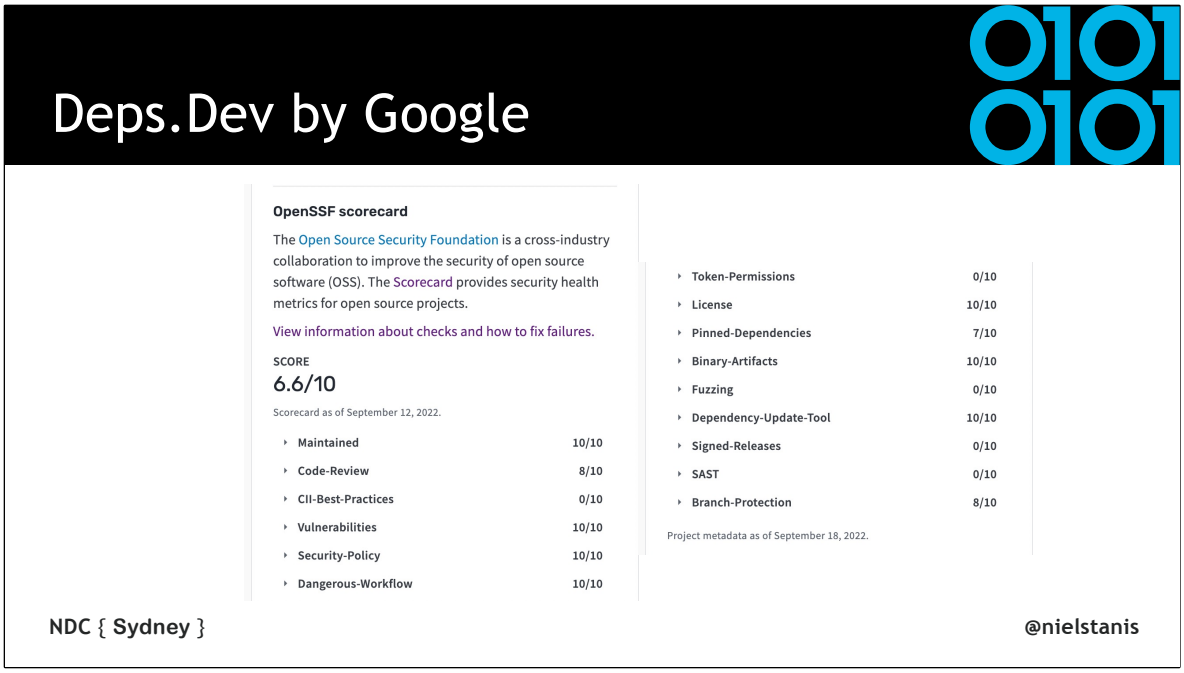
https://github.com/ossf/scorecard

Deps.Dev by Google



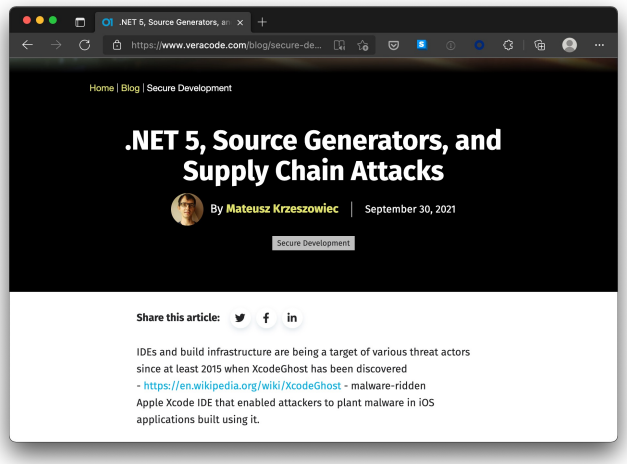
https://deps.dev/

Deps.Dev by Google



https://deps.dev/npm/electron

Source Generators

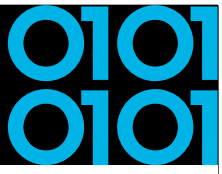


NDC { Sydney }

@nielstanis

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

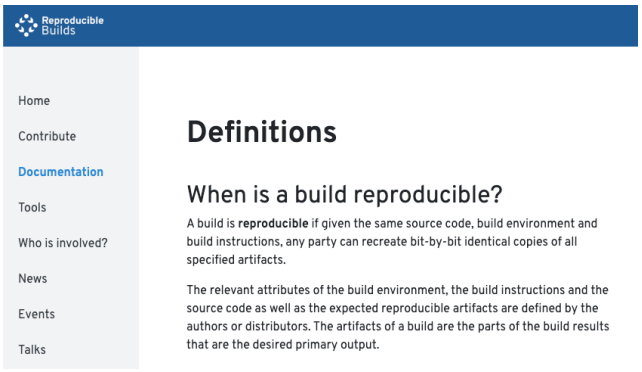
Source Generators



- 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

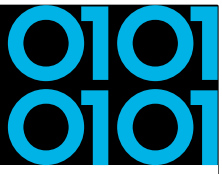


NDC { Sydney }

@nielstanis

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

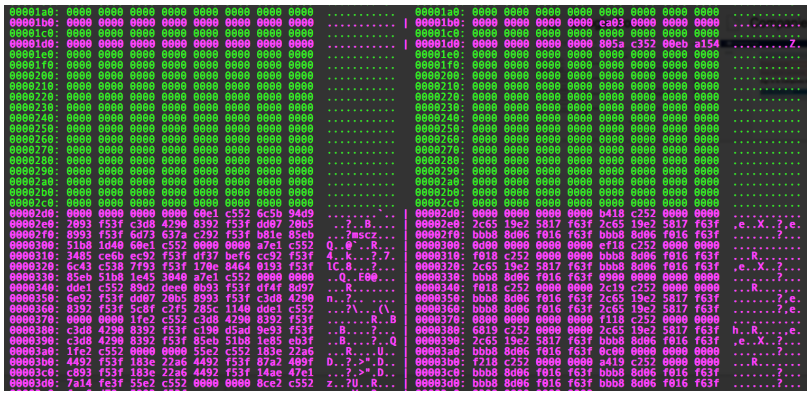
Reproducible/Deterministic Builds



- 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'

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

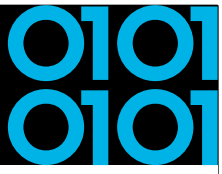


NDC { Sydney }

@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

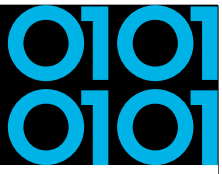
Reproducible/Deterministic Builds



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

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

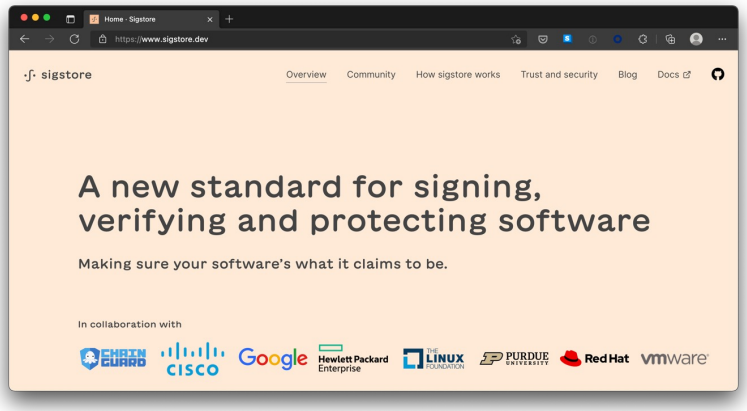
Reproducible Build Validation



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

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

Signing artifacts

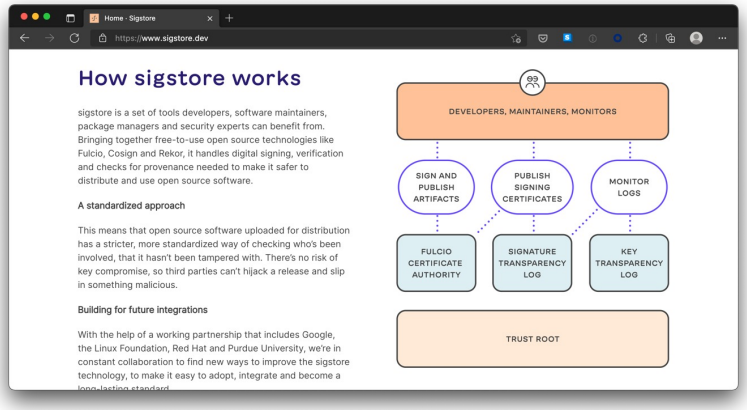


NDC { Sydney }

@nielstanis

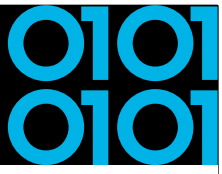
https://sigstore.dev

Signing artifacts



https://sigstore.dev

Signing artifacts



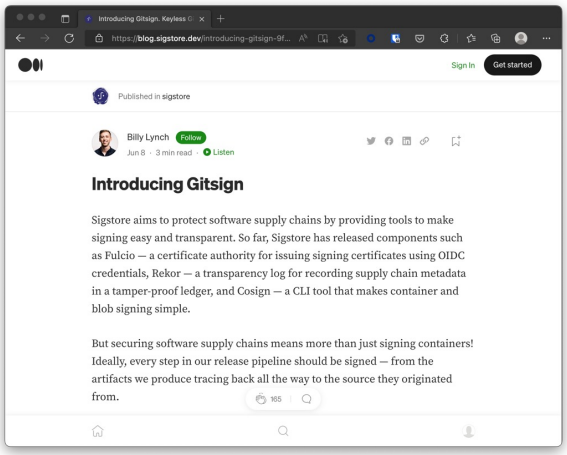
- Cosign can be used for signing files like binaries, packages and Docker images
- It can do keyless signing based on OpenID Connect
- GitHub Actions have released OpenID Connect support since end 2021

https://sigstore.dev

Cosign Keyless Signing

https://sigstore.dev
https://blog.sigstore.dev/introducing-gitsign-9fd3f1b682aa

Git Commit Signing Sigstore GitSign

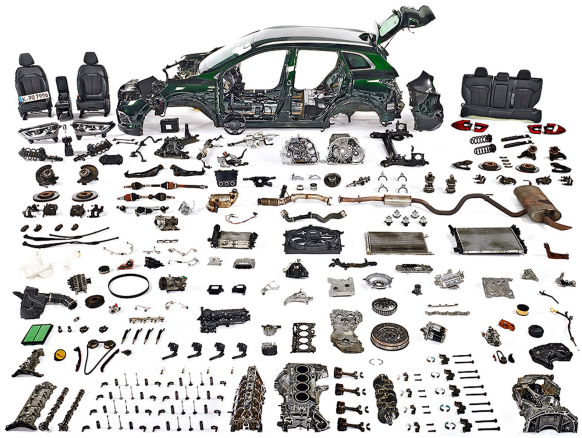


NDC { Sydney }

@nielstanis

https://sigstore.dev
https://blog.sigstore.dev/introducing-gitsign-9fd3f1b682aa

Automotive Industry



Car Supply Chain

Tata Steel Factory

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

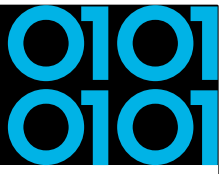
Bosch Factory

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

Renault Manufacturing

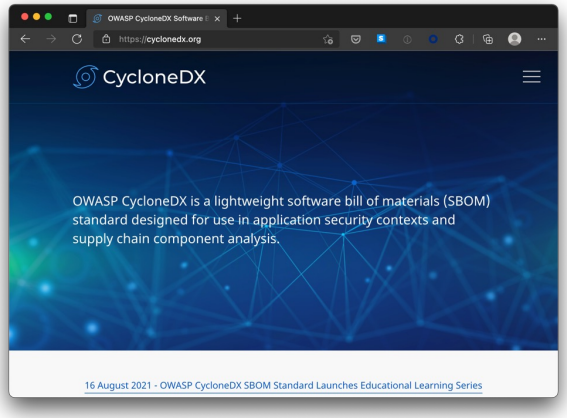
- Bosch Disk #45678
- Bosal Exhaust #RE9876
- Goodyear Tires #GY8877
- Kadjar VIN 1234567890

Software Bill of Materials (SBOM)



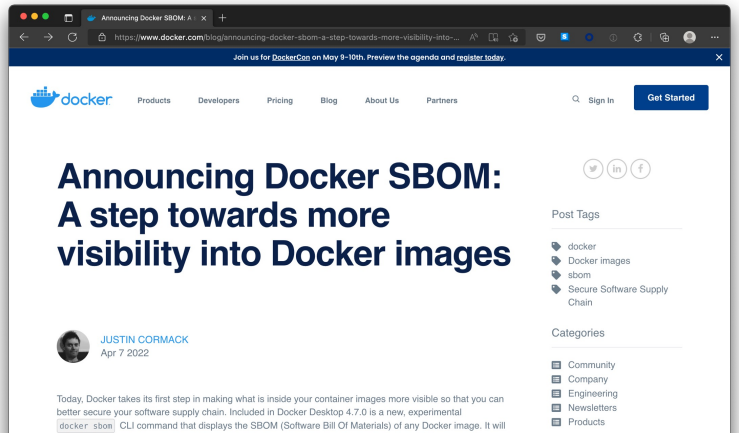
- Industry standard of describing the software
  - Producer Identity – Who Created it?
  - Product Identity – What's the product?
  - Integrity – Is the project unaltered?
  - Licensing – How can the project be used?
  - Creation – How was the product created? Process meets requirements?
  - Materials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

Software Bill of Materials (SBOM)



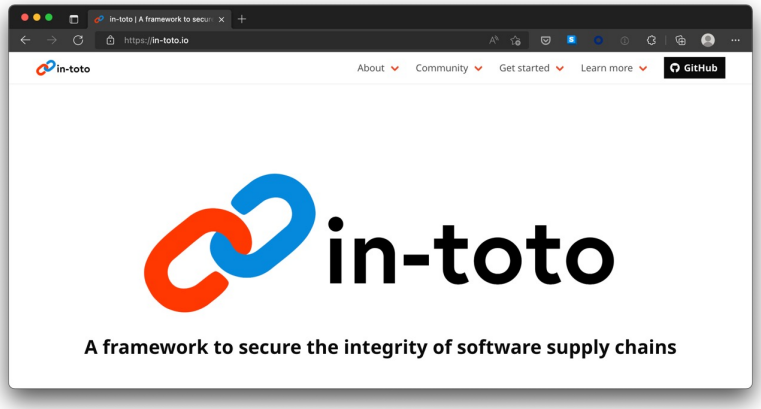
https://cyclonedx.org

Docker SBOM

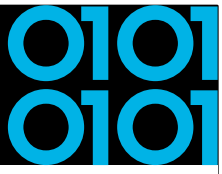


https://www.docker.com/blog/announcing-docker-sbom-a-step-towards-more-visibility-into-docker-images/

In-toto


in-toto | A framework to secure
https://in-toto.io
in-toto
About
Community
Get started
Learn more
GitHub
in-toto
A framework to secure the integrity of software supply chains

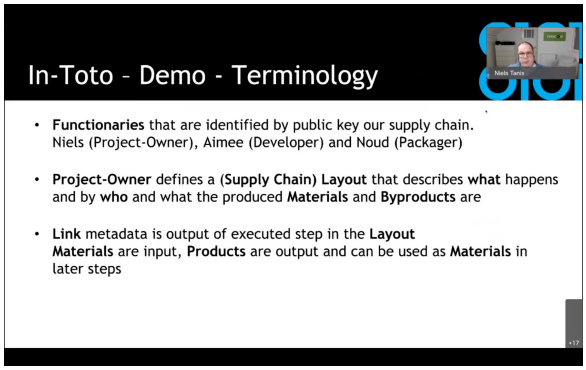
In-Toto - Terminology



- **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

https://in-toto.io/

In-Toto – Demo

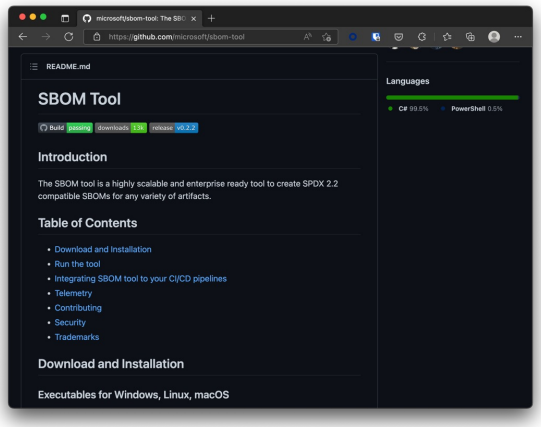


NDC { Sydney }

@nielstanis

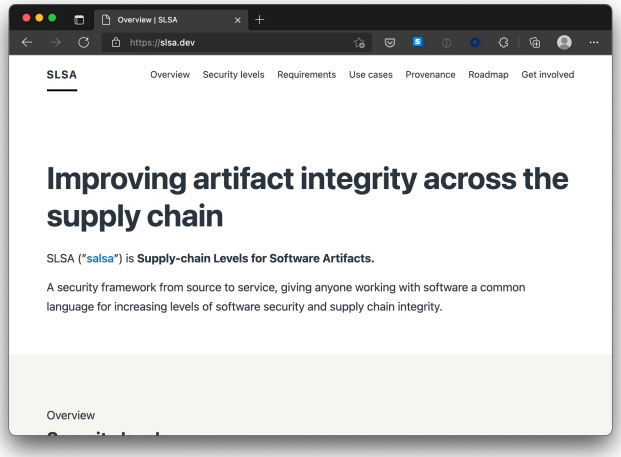
https://youtu.be/fYCfB7MZPh4?t=2777

Microsoft SBOM Tool



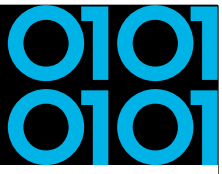
https://github.com/microsoft/sbom-tool

Google SLSA



https://slsa.dev

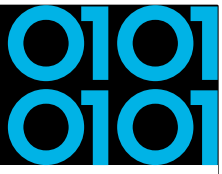
Google SLSA Levels



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

https://slsa.dev

SLSA GitHub Action



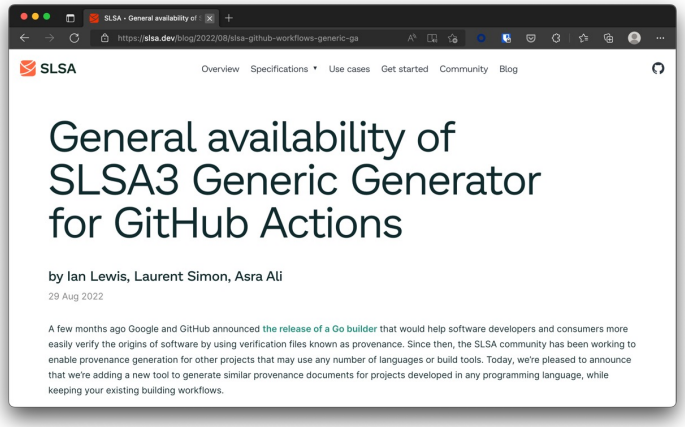
- Released April 2022

- 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

SLSA3 Generator GitHub Actions

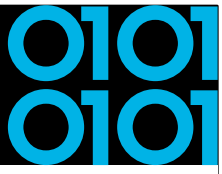


NDC { Sydney }

@nielstanis

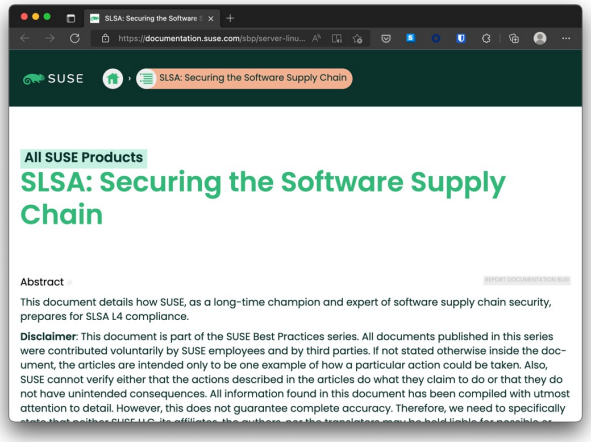
https://slsa.dev/blog/2022/08/slsa-github-workflows-generic-ga

MyAwesomePDFComponent Demo



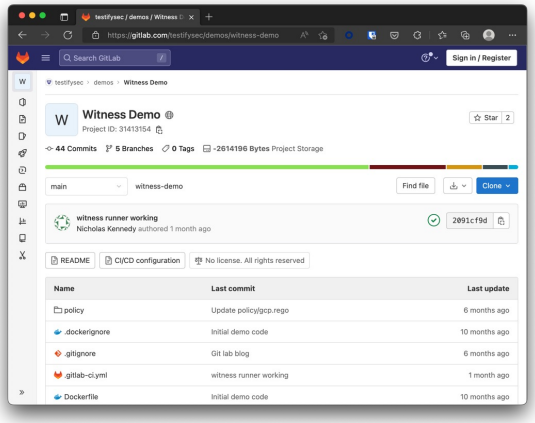
- A NuGet Package build in GitHub Actions
- CycloneDX SBOM
- Sigstore Keyless Signing
- SLSA Level 3 Provenance
- SBOM data, know what?

SUSE SLSA Level 4



https://documentation.suse.com/sbp/server-linux/html/SBP-SLSA4/index.html

Witness & GitLab Attestator

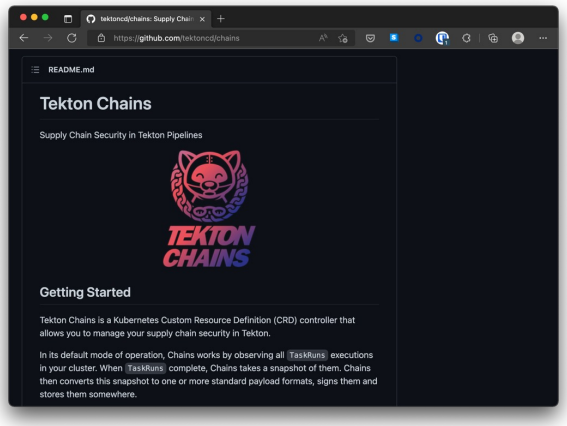


NDC { Sydney }

@nielstanis

https://gitlab.com/testifysec/demos/witness-demo
https://github.com/testifysec/witness

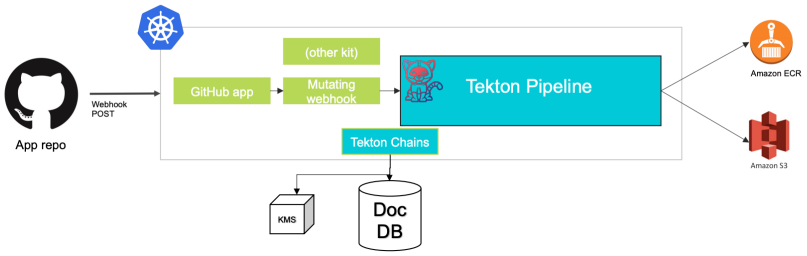
Open Shift - Tekton - Tekton Chains



https://github.com/tektoncd/chains

SolarWinds Project Trebuchet

Pipeline With Attestations

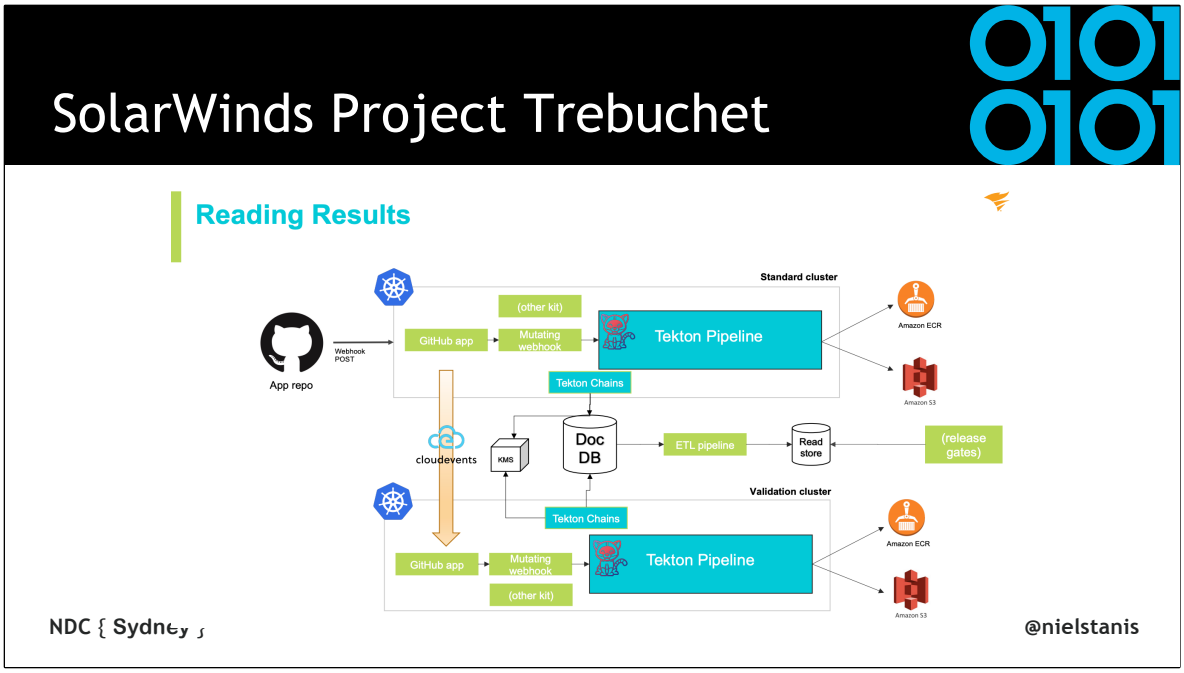


NDC { Sydney }

@nielstanis

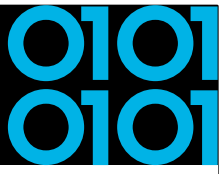
https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf
https://www.youtube.com/watch?v=1-tMRxqMwTQ

SolarWinds Project Trebuchet



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

Grafeas and Kritis by Google

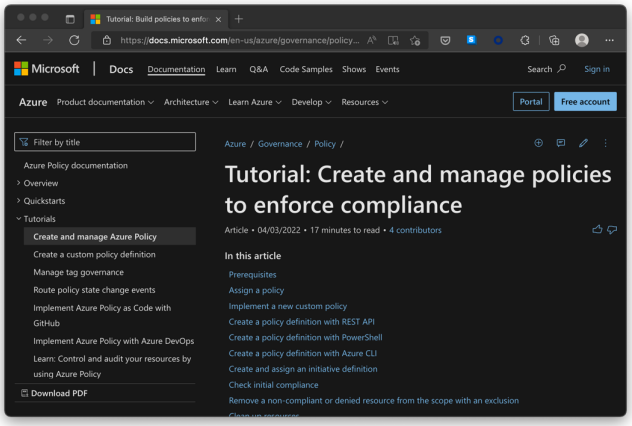


- Google released in 2019
- Grafeas – Component Metadata API
  - Container Analysis API on Google Cloud Platform
- Kritis – Deployment Authorization for Kubernetes Apps
  - Binary Authorization on Google Cloud Platform

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

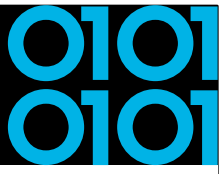
https://www.infoq.com/presentations/supply-grafeas-kritis/
https://www.youtube.com/watch?v=hOzH3mOApjs
https://www.youtube.com/watch?v=05zN-YQxEAM

Azure Policy



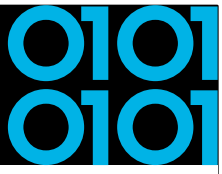
https://devblogs.microsoft.com/devops/secure-software-supply-chain-with-azure-pipelines-artifact-policies/
https://docs.microsoft.com/en-us/azure/devops/pipelines/process/artifact-policy?view=azure-devops

Conclusion

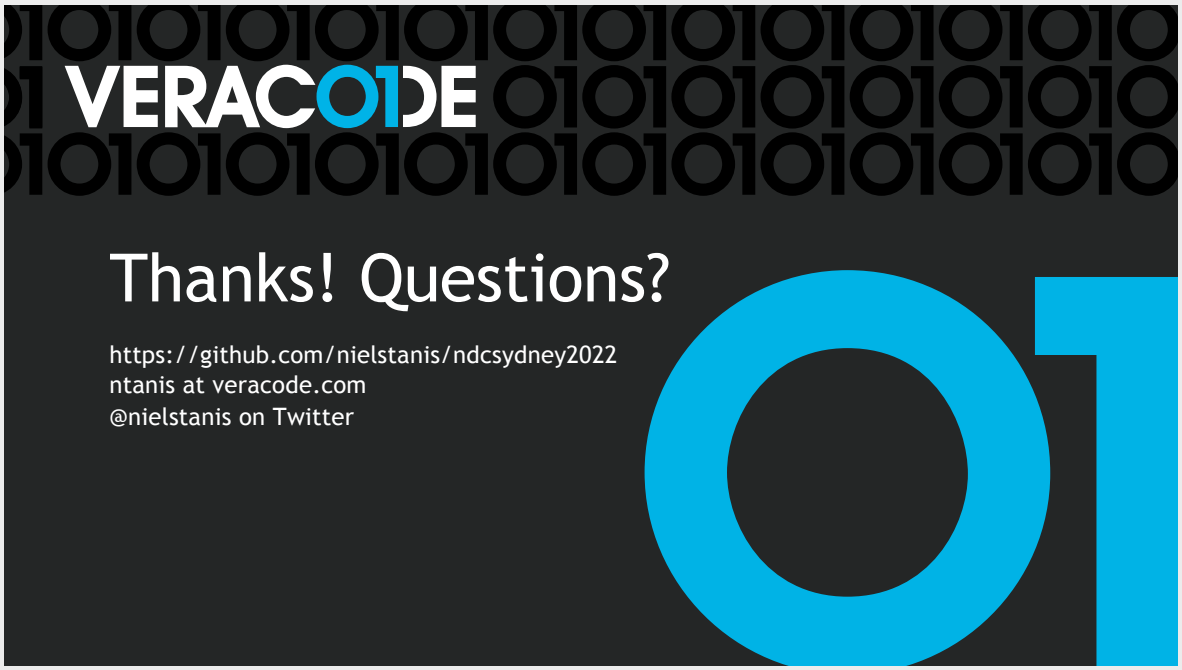


- It’s not how it’s more a matter of when!
- Be aware of your used software supply chain(s).
- Know what you’re using and pulling into projects.

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/ndcsydney2022
ntanis at veracode.com
@nielstanis on Twitter