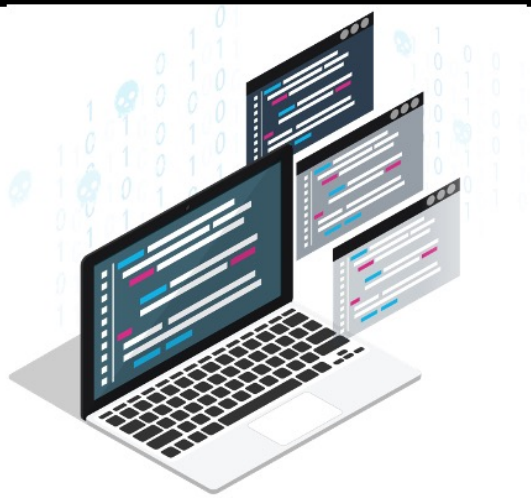

NDC { Oslo }
Securing your .NET application
software supply-chain
Niels Tanis
VERACODE
0101
0101

Who am I?

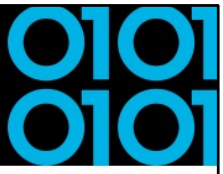
- Niels Tanis
  - Security Researcher @ Veracode
  - Background in .NET Development
  - Application Security Consultancy
  - Pen-testing & Ethical Hacking
  - ISC² CSSLP

Securing your .NET application software supply-chain



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

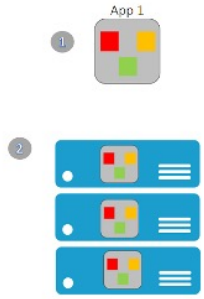
Agenda

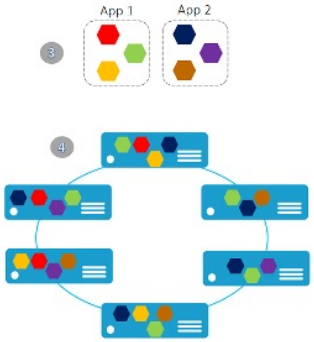


- Definition Software Supply-Chain
- Securing the Software Supply-Chain
  - Developer & Source
  - 3rd Party Libraries
  - Build & Release
- Conclusion and Q&A

Evolution in Software Architecture

- Monolith
- Microservices
- Serverless
  Cloud-Native

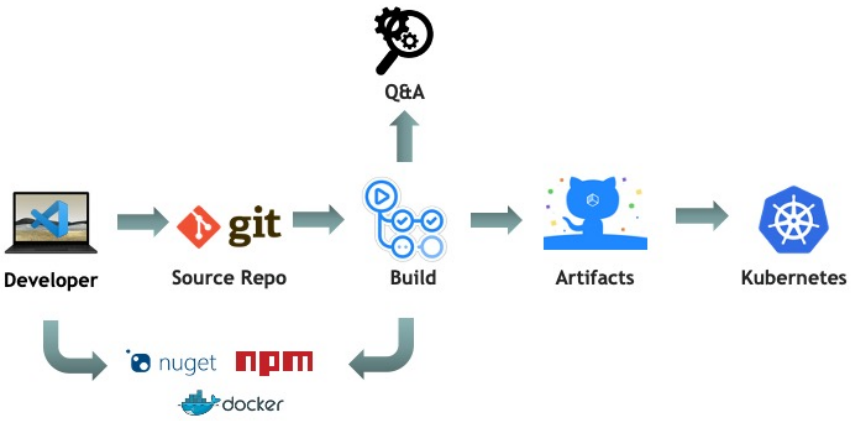




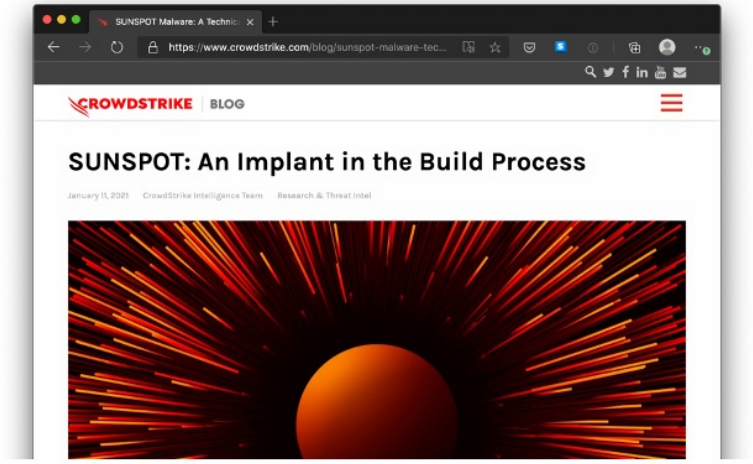
What is a Supply Chain?

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



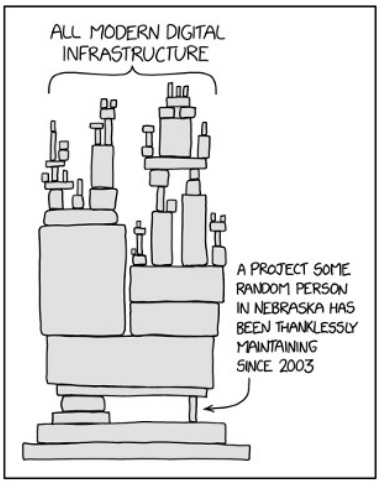
SolarWinds SunSpot



Kaseya

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

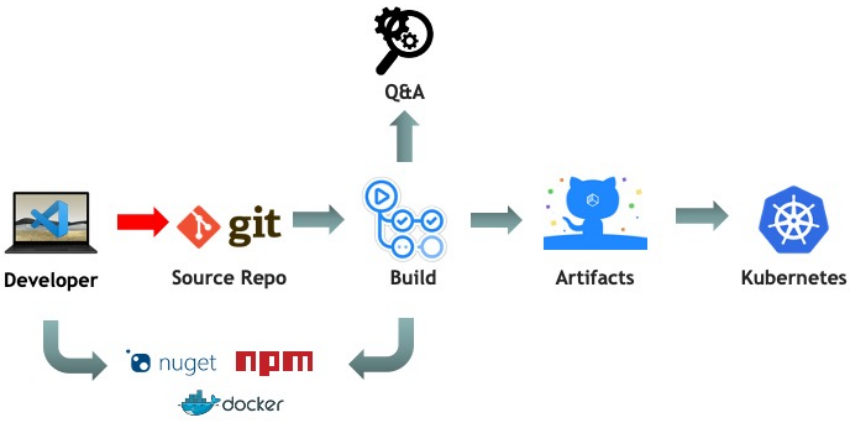
XKDC - Dependency



https://xkcd.com/2347/

https://xkcd.com/2347/

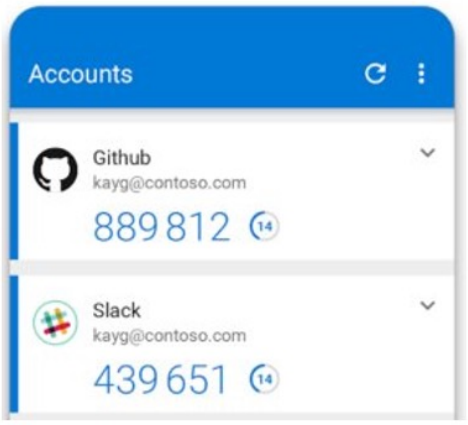
Software Supply Chain



GitHub account

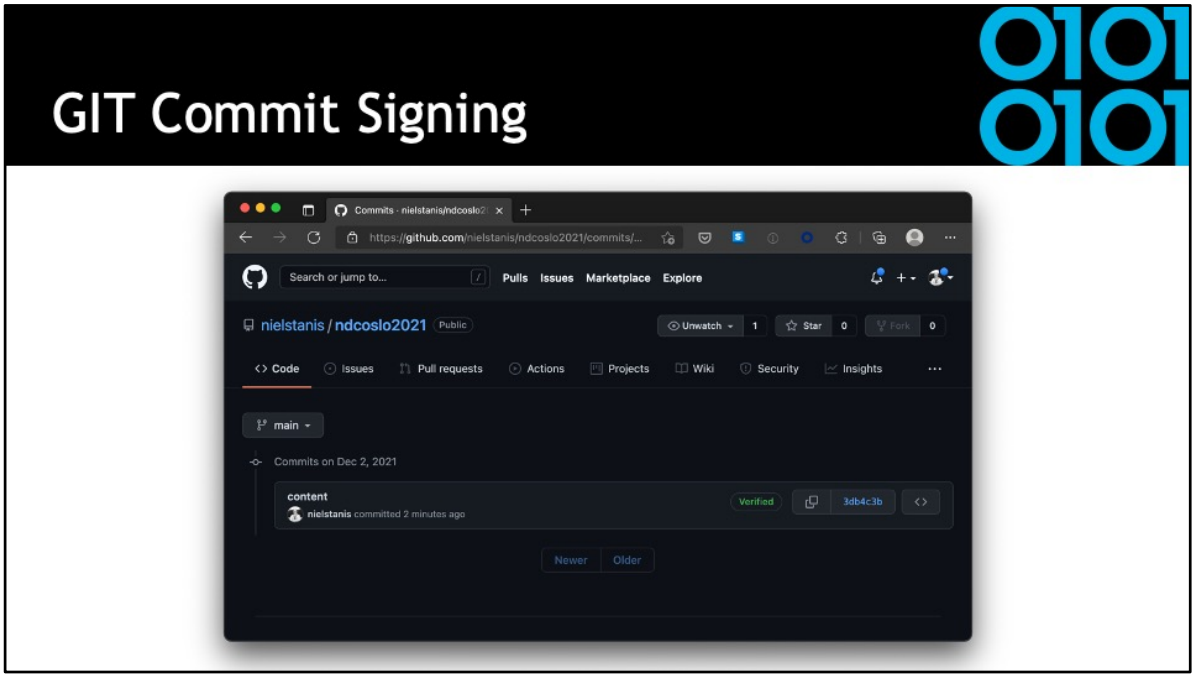
https://www.zdnet.com/article/canonical-github-account-hacked-ubuntu-source-code-safe/

Use MFA on source-repository



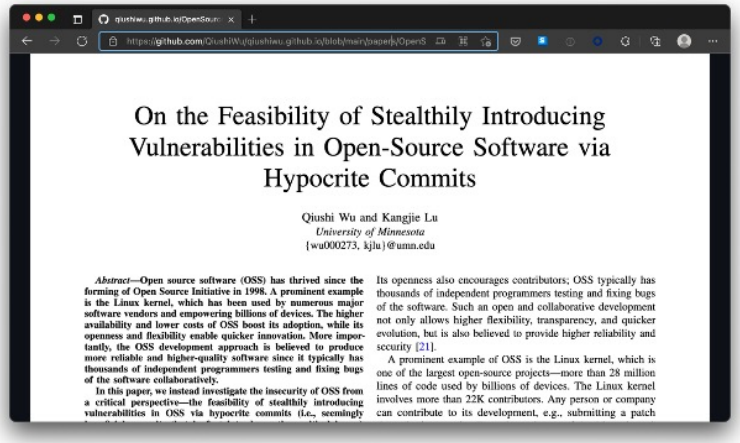
https://help.github.com/en/github/authenticating-to-github/configuring-two-factor-authentication

GIT Commit Signing



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

Octopus Scanner - NetBeans

May 28, 2020

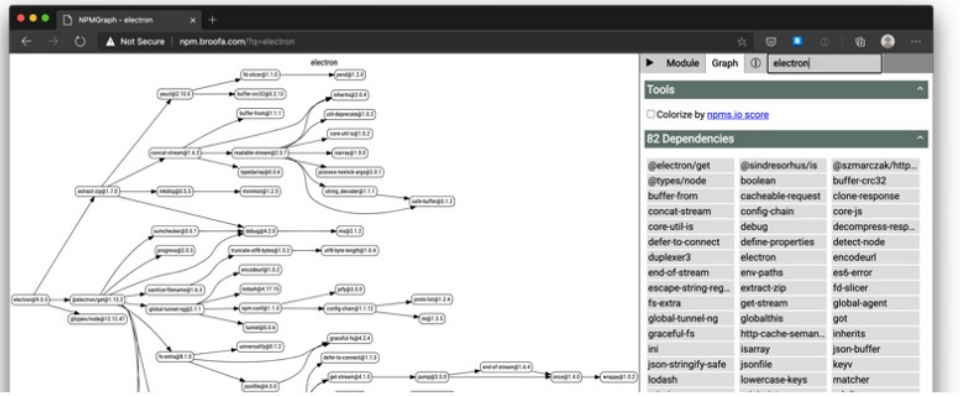
The Octopus Scanner Malware: Attacking the open source supply chain

Alvaro Muñoz

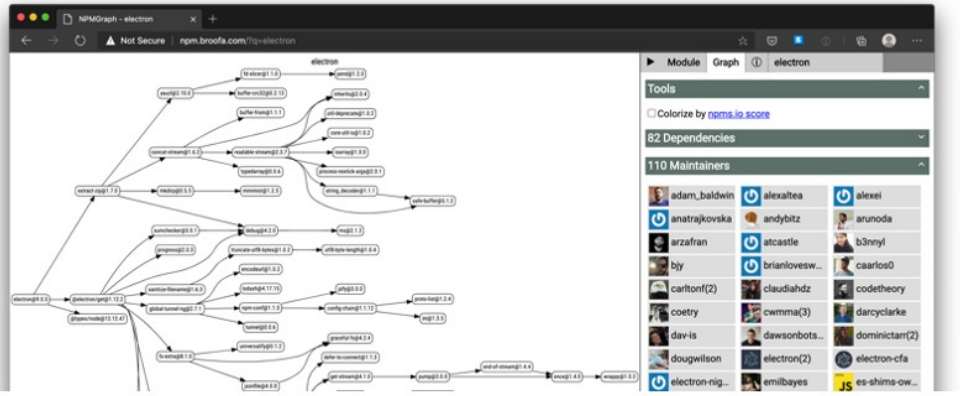
Securing the open source supply chain is an enormous task. It goes far beyond a security assessment or just patching for the latest CVEs. Supply chain security is about the integrity of the entire software development and delivery ecosystem. From the code commits themselves, to how they flow through the CI/CD pipeline, to the actual delivery of releases, there's the potential for loss of integrity and security concerns, throughout the entire lifecycle.

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

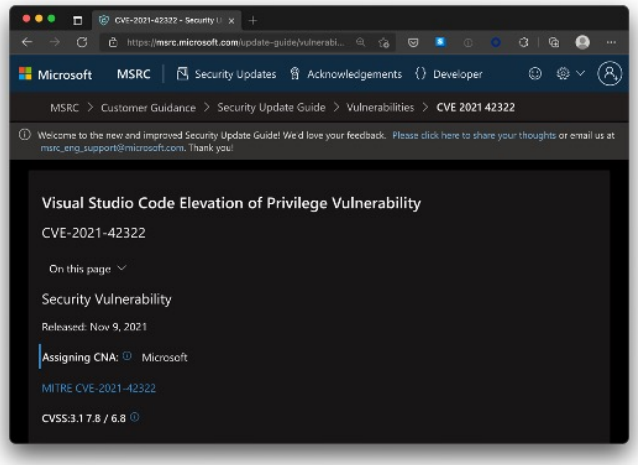
Visual Studio Code



Visual Studio Code



Visual Studio Code

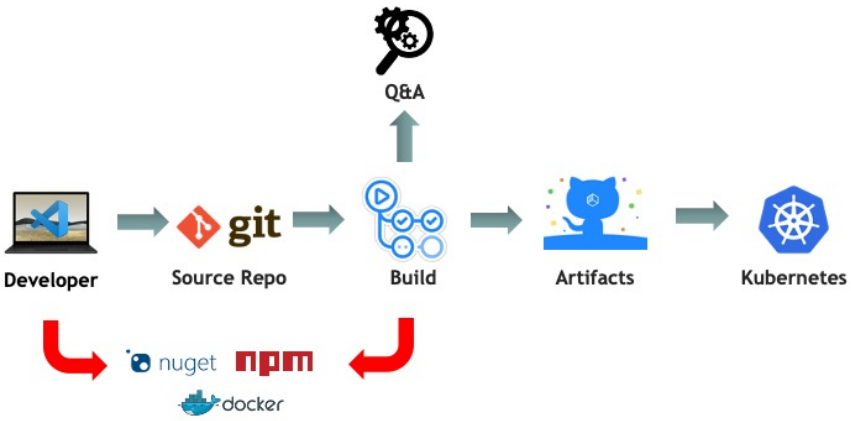


https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-42322

Visual Studio Code

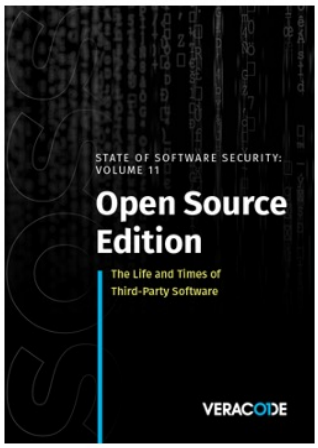
https://www.bleepingcomputer.com/news/security/heres-how-a-researcher-broke-into-microsoft-vs-codes-github/

3rd Party Libraries



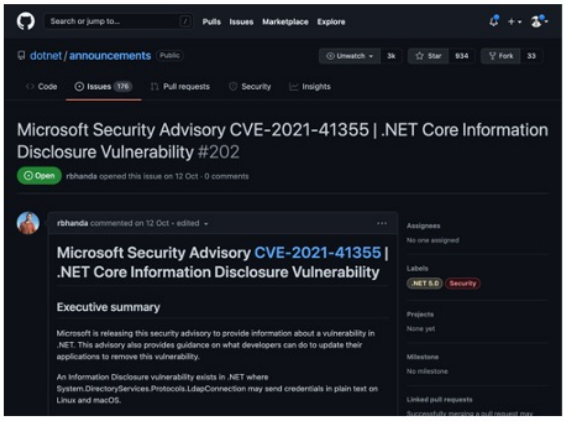
State Of Software Security v11 2021

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



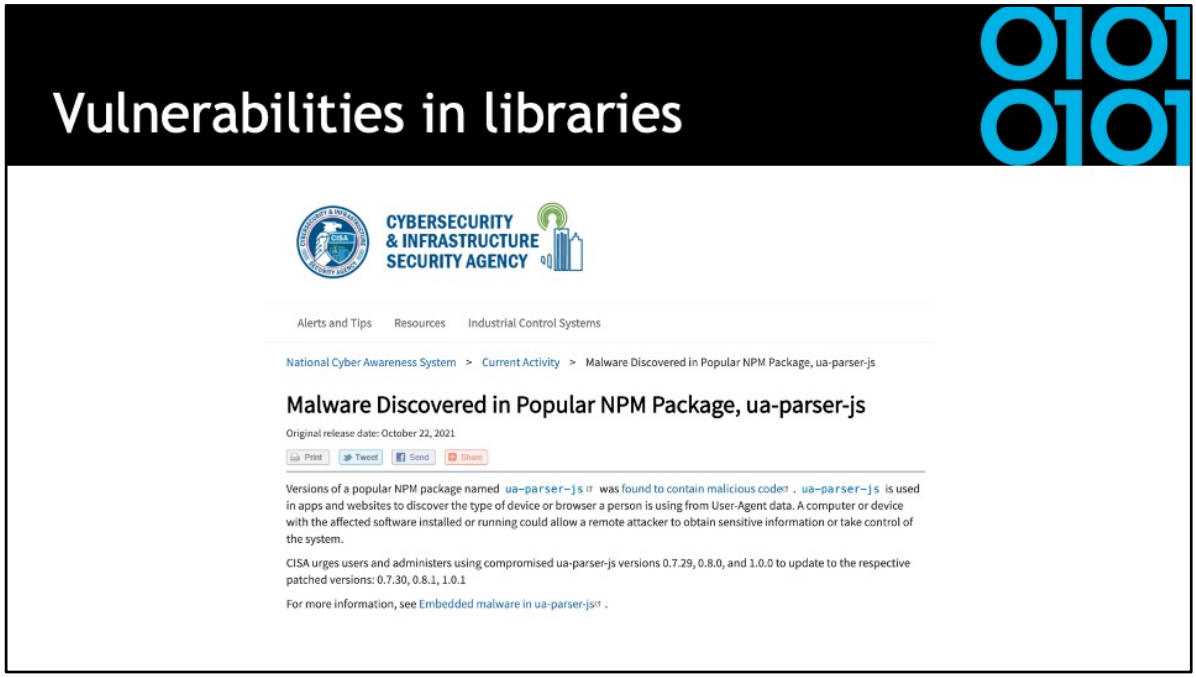
https://info.veracode.com/fy22-state-of-software-security-v11-open-source-edition.html

Vulnerabilities in libraries



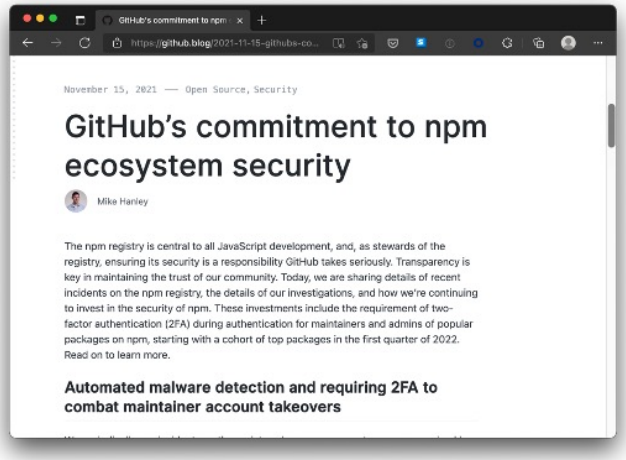
https://github.com/dotnet/announcements/issues/202

Vulnerabilities in libraries



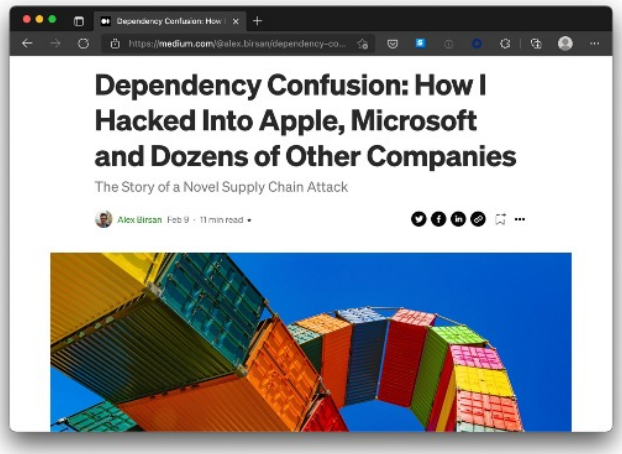
https://us-cert.cisa.gov/ncas/current-activity/2021/10/22/malware-discovered-popular-npm-package-ua-parser-js
https://portswigger.net/daily-swig/popular-npm-package-ua-parser-js-poisoned-with-cryptomining-password-stealing-malware

Vulnerabilities in libraries



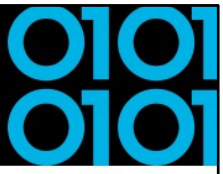
https://github.blog/2021-11-15-githubs-commitment-to-npm-ecosystem-security/

Dependency Confusion



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

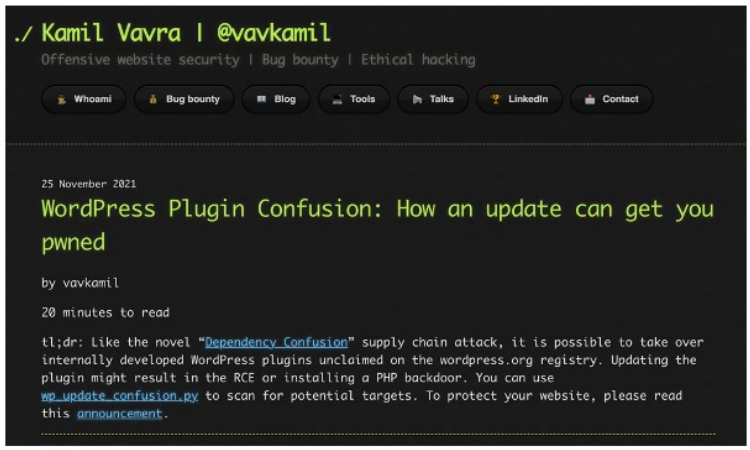
Dependency Confusion - NuGet



- 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

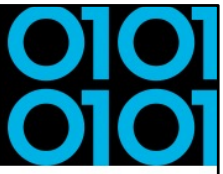
https://azure.microsoft.com/nl-nl/resources/3-ways-to-mitigate-risk-using-private-package-feeds/
https://azure.microsoft.com/mediahandler/files/resourcefiles/3-ways-to-mitigate-risk-using-private-package-feeds/3%20Ways%20to%20Mitigate%20Risk%20When%20Using%20Private%20Package%20Feeds%20-%20v1.0.pdf

WordPress Plugin Confusion



https://vavkamil.cz/2021/11/25/wordpress-plugin-confusion-update-can-get-you-pwned/

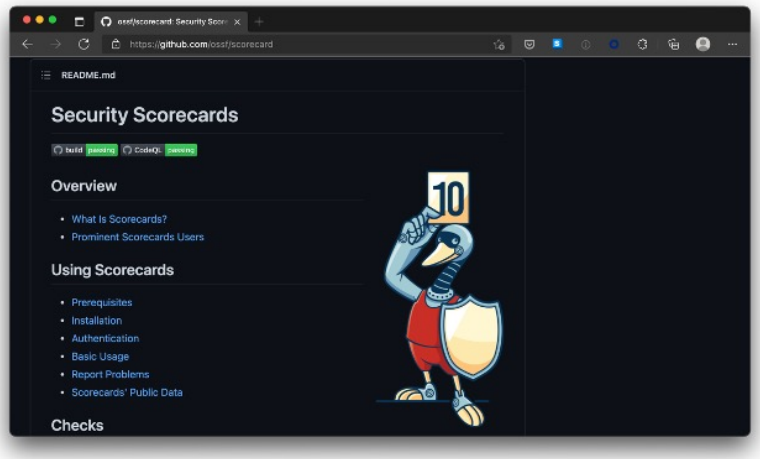
3rd Party Libraries



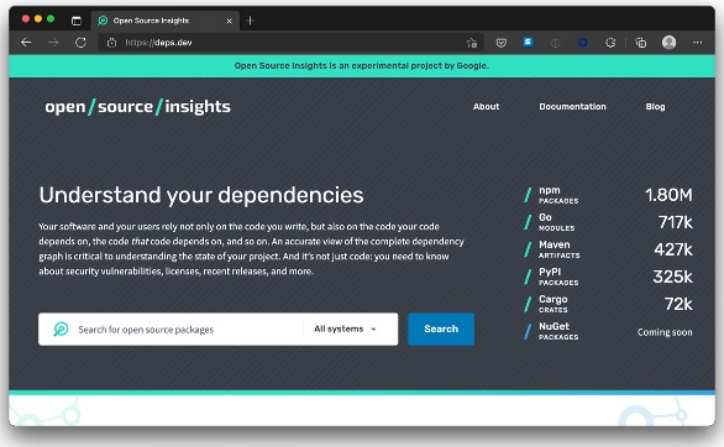
- Intent of library, know what's inside!
- Keep in mind that's a transitive list of dependencies
- Look out for talk on 'Sandboxing .NET Assemblies' I gave in November.

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

Security Scorecards - OpenSSF

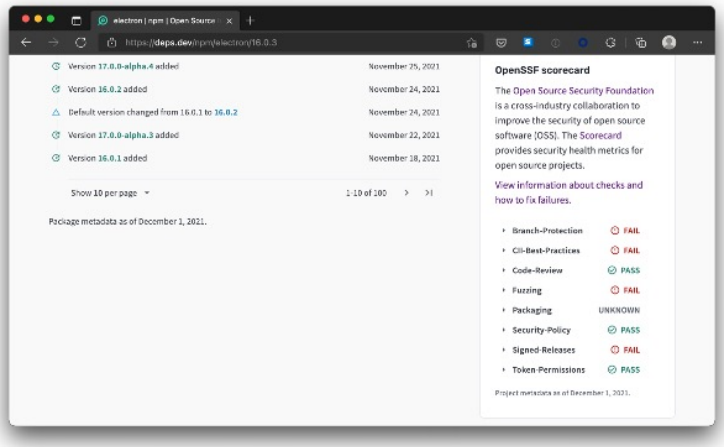


Deps.Dev by Google



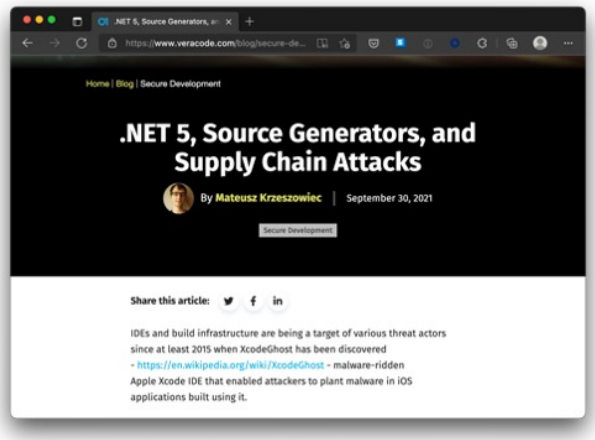
https://deps.dev/

Deps.Dev by Google



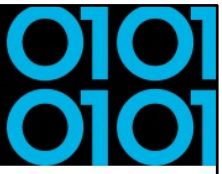
https://deps.dev/

Source Generators



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

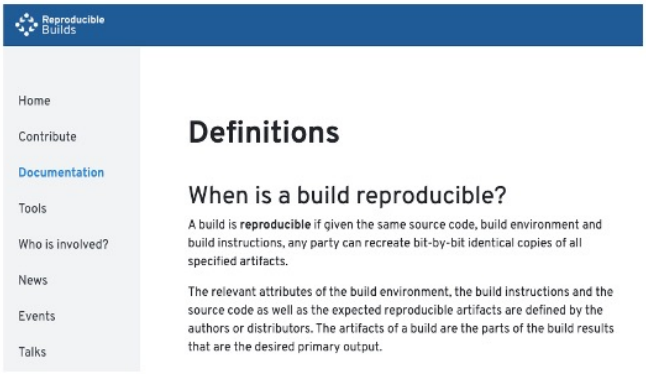
Source Generators



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

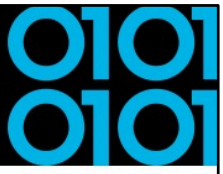
```
<Target Name="DisableAnalyzers"
        BeforeTargets="CoreCompile">
  <ItemGroup>
    <Analyzer Remove="@(Analyzer)" />
  </ItemGroup>
</Target>
```

Reproducible/Deterministic Builds



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

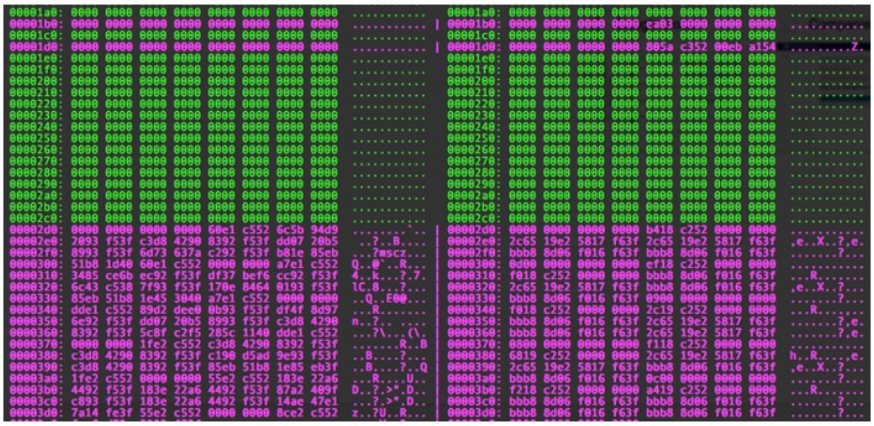
Reproducible/Deterministic Builds



- 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'

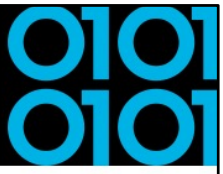
https://blog.paranoidcoding.com/2016/04/05/deterministic-builds-in-roslyn.html
https://github.com/dotnet/roslyn/blob/master/docs/compilers/Deterministic%20Inputs.md
https://github.com/clairernovotny/DeterministicBuilds

Reproducible Build .NET6



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

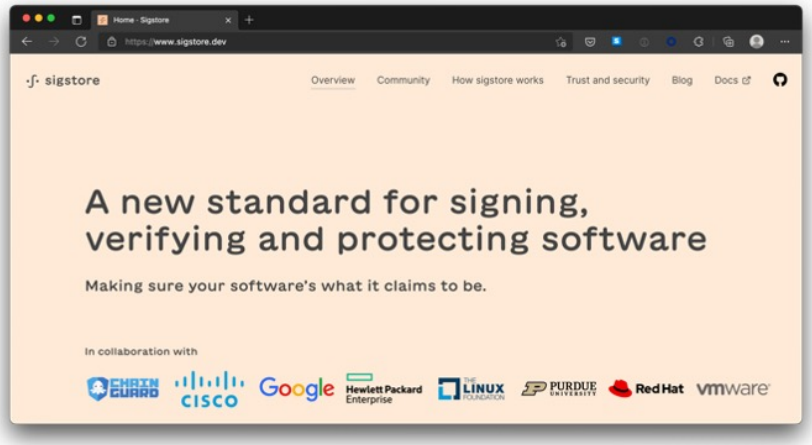
Reproducible/Deterministic Builds



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

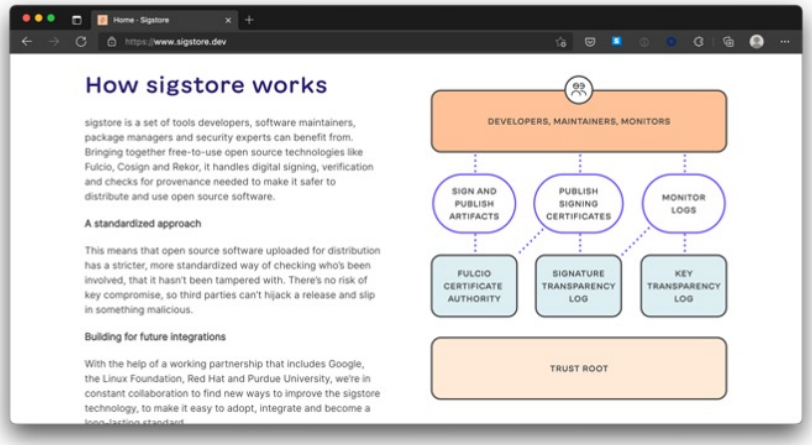
https://blog.paranoidcoding.com/2016/04/05/deterministic-builds-in-roslyn.html
https://github.com/dotnet/roslyn/blob/master/docs/compilers/Deterministic%20Inputs.md
https://devblogs.microsoft.com/dotnet/producing-packages-with-source-link/
https://github.com/dotnet/reproducible-builds

Signing artifacts



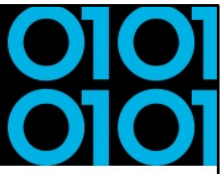
https://sigstore.dev

Signing artifacts



https://sigstore.dev

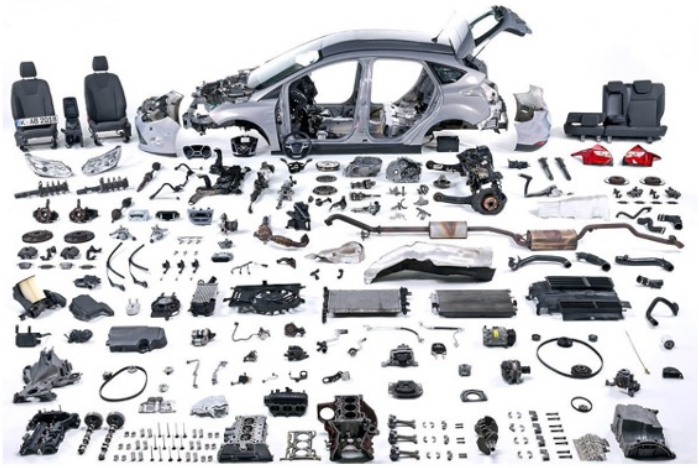
Signing artifacts



- Cosign can do keyless signing based on OpenID Connect
- GitHub Actions have released OpenID Connect support
- Demo time!

https://sigstore.dev

Automotive Industry



Car Supply Chain

Tata Steel Factory

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

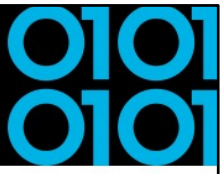
Bosch Factory

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

Renault Manufacturing

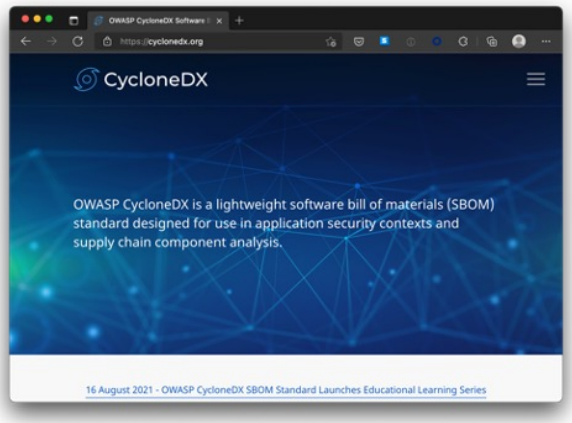
- Bosch Disk #45678
- Bosal Exhaust #RE9876
- Goodyear Tires #GY8877
- Kadjar VIN 1234567890

Software Bill of Materials (SBOM)



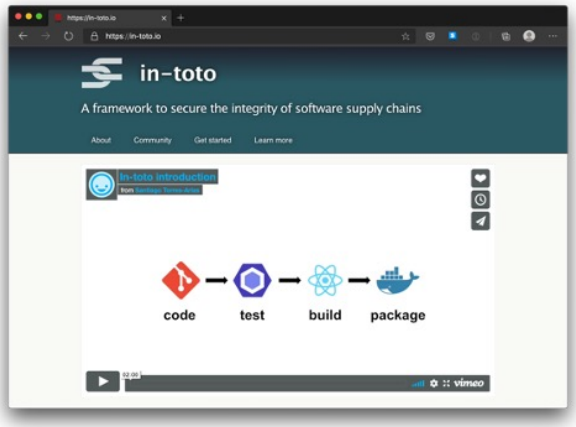
- Industry standard of describing the software
  - Producer Identity – Who Created it?
  - Product Identity – What's the product?
  - Integrity – Is the project unaltered?
  - Licensing – How can the project be used?
  - Creation – How was the product created? Process meets requirements?
  - Materials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

Software Bill of Materials (SBOM)

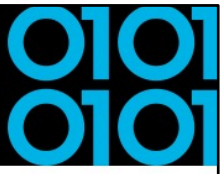


https://cyclonedx.org

In-toto


in-toto
A framework to secure the integrity of software supply chains
About
Community
Get started
Learn more
in-toto introduction
code
test
build
package
vimeo

In-Toto – Demo - 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

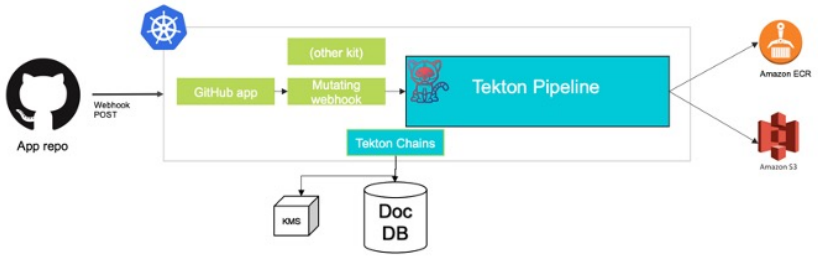
https://in-toto.io/

In-Toto – Demo

https://in-toto.io/

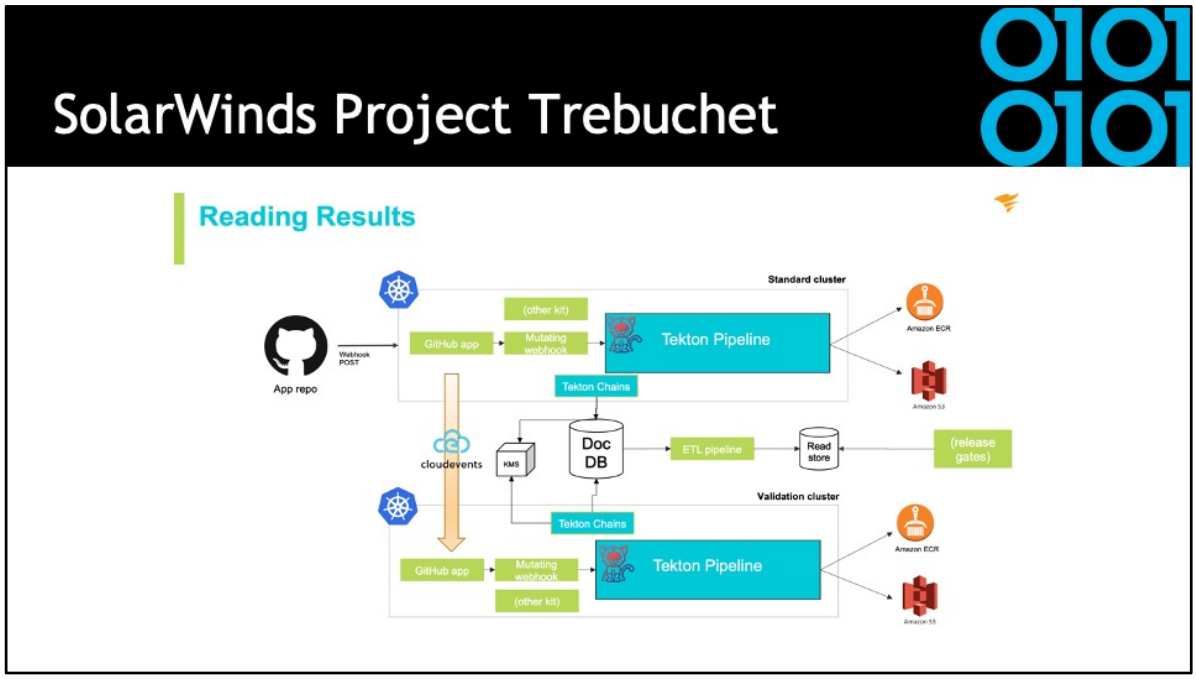
SolarWinds Project Trebuchet

Pipeline With Attestations



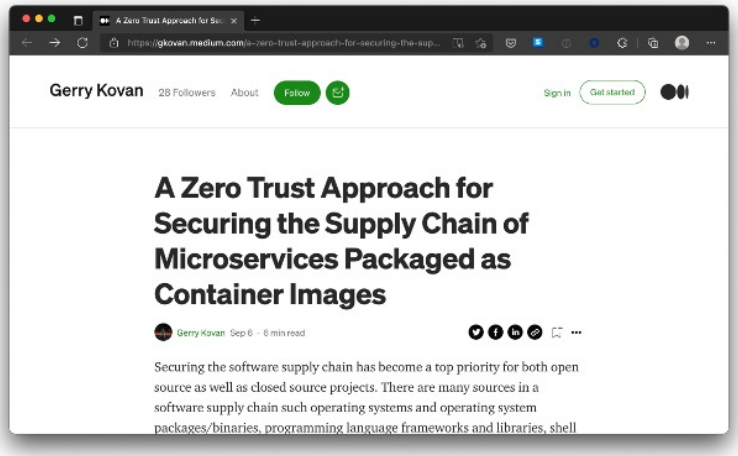
https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf
https://www.youtube.com/watch?v=1-tMRxqMwTQ

SolarWinds Project Trebuchet



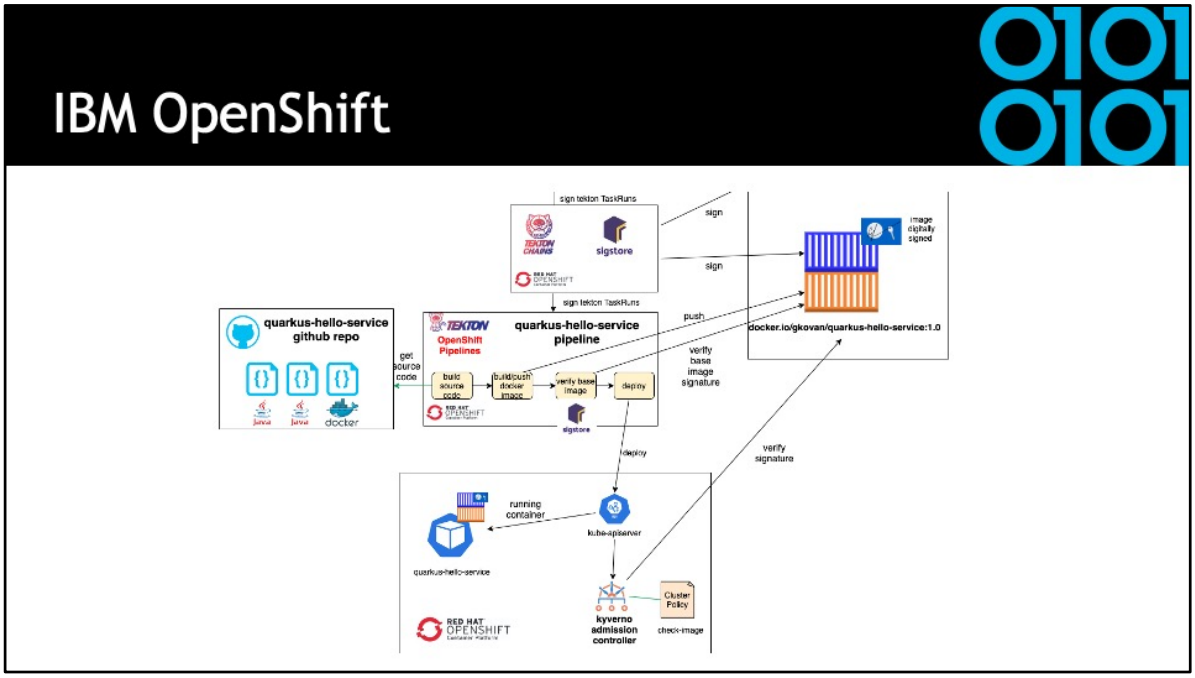
https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf
https://www.youtube.com/watch?v=1-tMRxqMwTQ

IBM OpenShift



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

IBM OpenShift



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

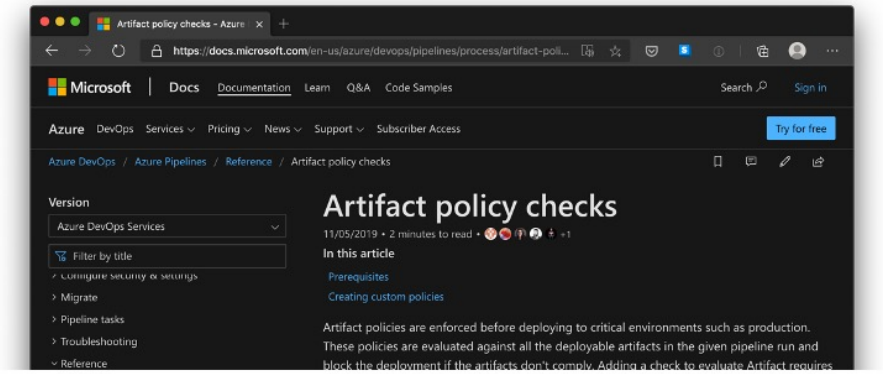
Grafeas and Kritis by Google

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

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

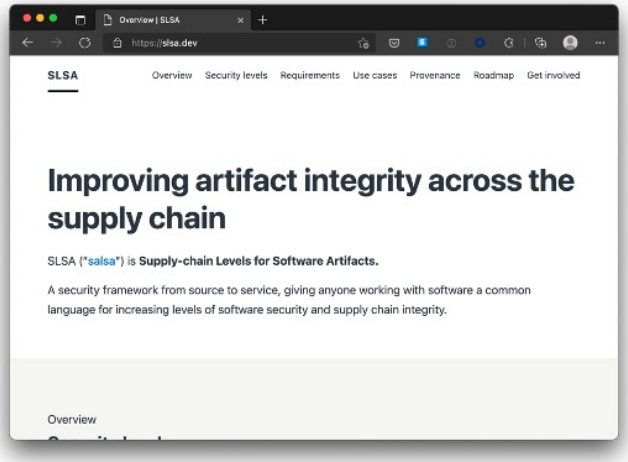
https://www.infoq.com/presentations/supply-grafeas-kritis/
https://www.youtube.com/watch?v=hOzH3mOApjs
https://www.youtube.com/watch?v=05zN-YQxEAM

Azure Pipelines Artifact Policy



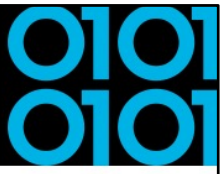
https://devblogs.microsoft.com/devops/secure-software-supply-chain-with-azure-pipelines-artifact-policies/
https://docs.microsoft.com/en-us/azure/devops/pipelines/process/artifact-policy?view=azure-devops

Google SLSA

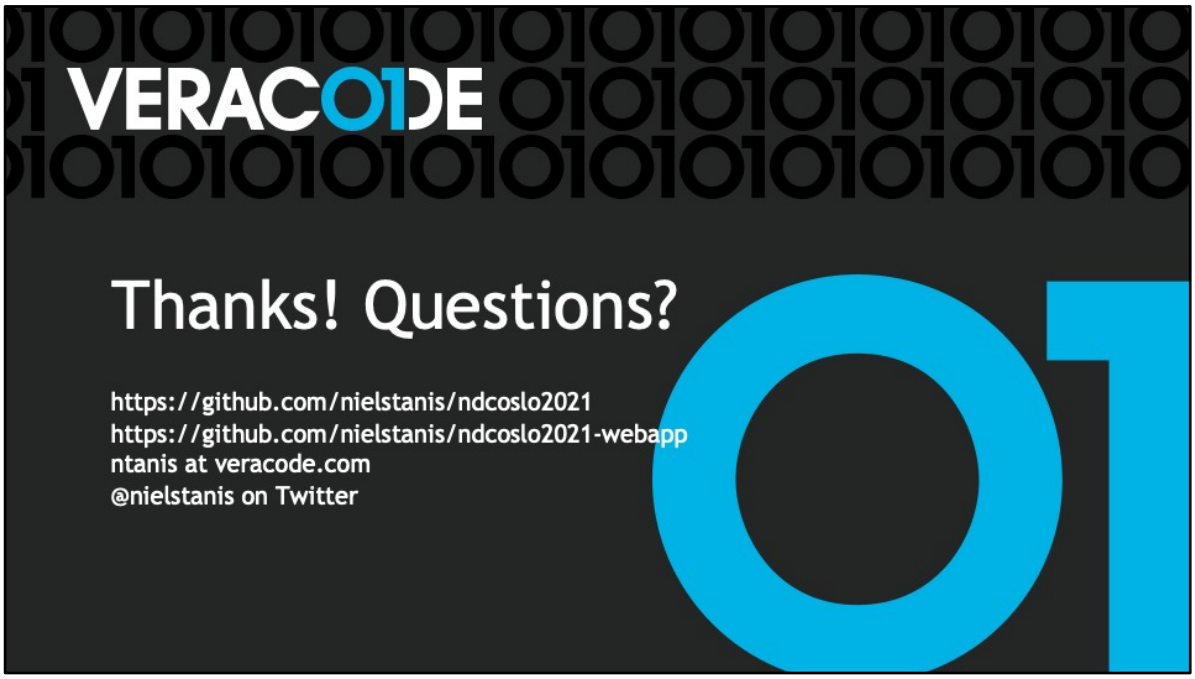


https://slsa.dev

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.
- Integrate security into your software lifecycle.
- Start working on creating SBOM’s and see how SLSA can fit into your process.


VERACODE
Thanks! Questions?
https://github.com/nielstanis/ndcoslo2021
https://github.com/nielstanis/ndcoslo2021-webapp
ntanis at veracode.com
@nielstanis on Twitter