

Copenhagen *Developers Festival*

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

Niels Tanis

0101
0101

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





Agenda

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



Agenda

- <https://github.com/nielstanis/cphdevfest2023>
- <https://github.com/nielstanis/docgenerator>

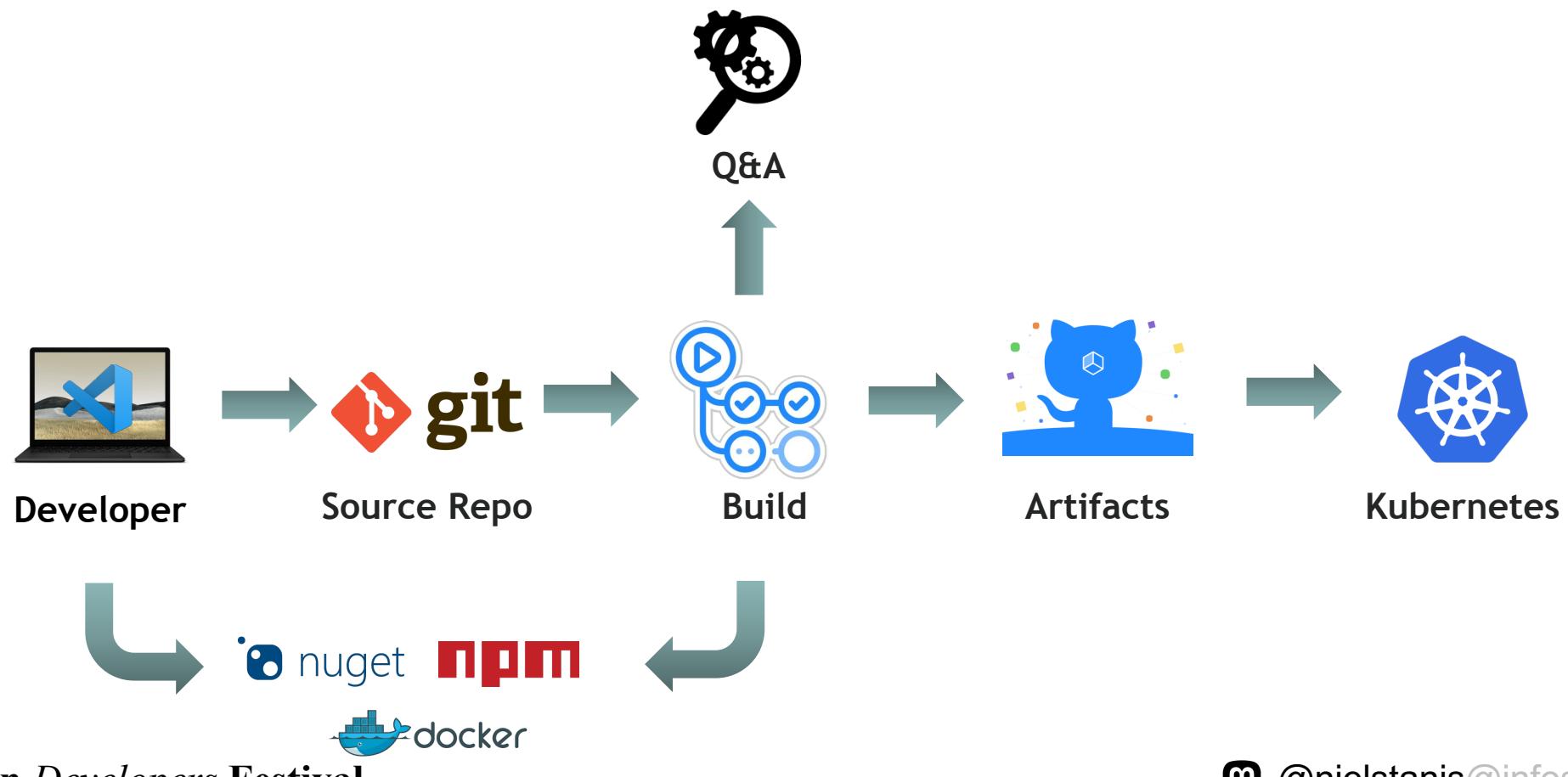
0101
0101

What is a Supply Chain?



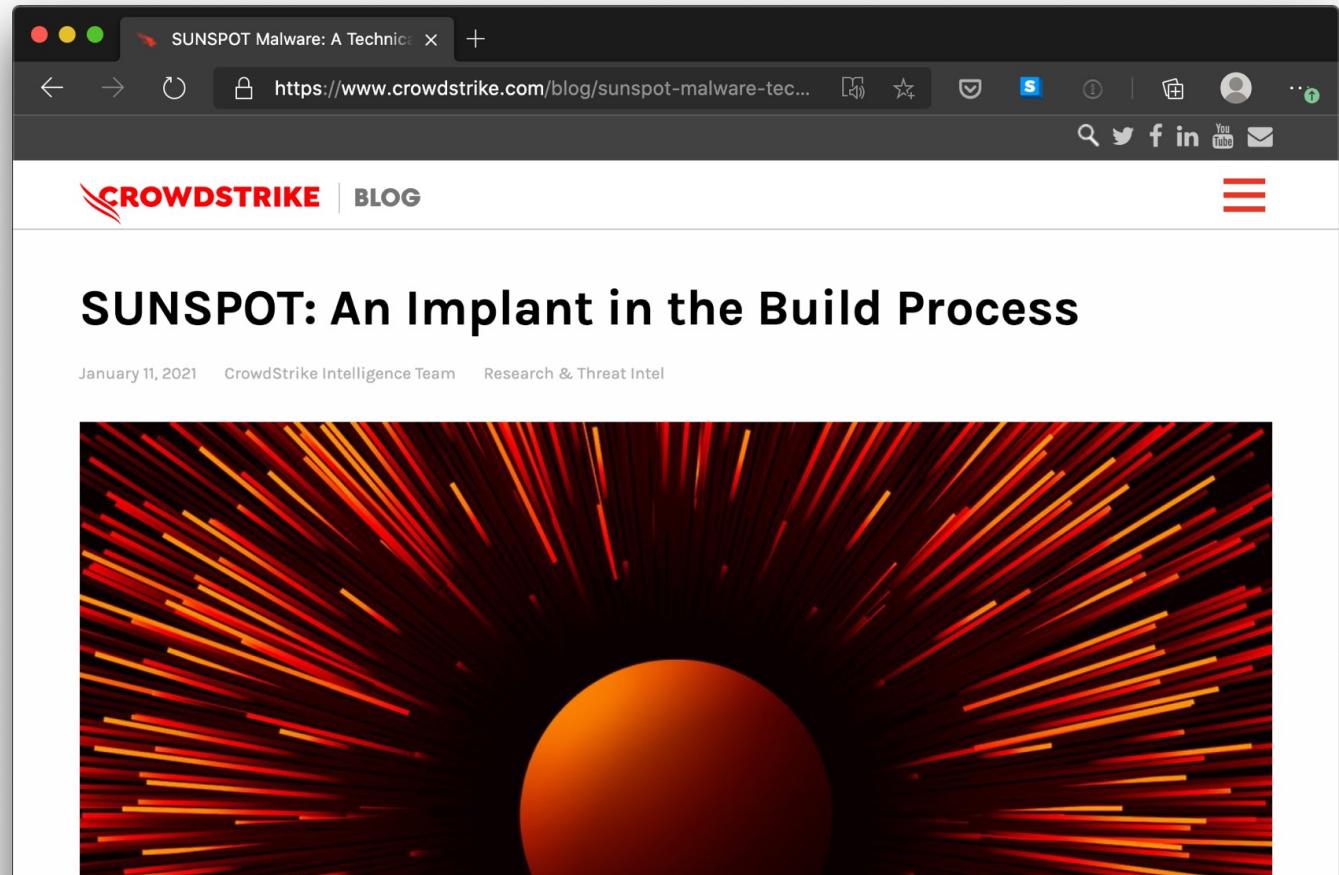
Software Supply Chain

0101
0101



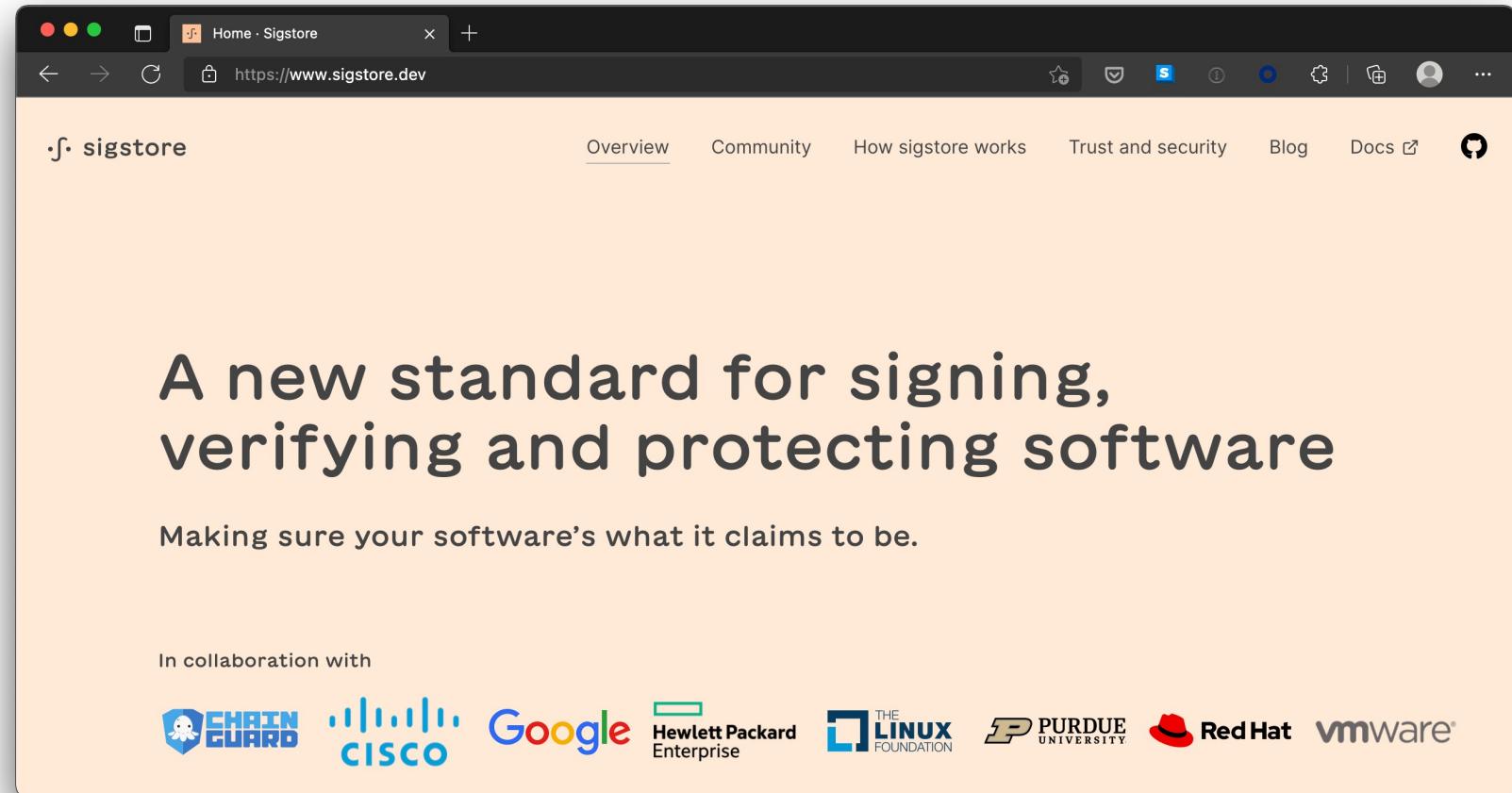
0101
0101

SolarWinds SunSpot



Signing artifacts

0101
0101



A screenshot of a web browser displaying the Sigstore website at <https://www.sigstore.dev>. The page has a light orange background. At the top left is the Sigstore logo (a stylized 'f'). To its right are navigation links: Overview (underlined), Community, How sigstore works, Trust and security, Blog, Docs, and a user icon. Below the navigation is a large heading: "A new standard for signing, verifying and protecting software". Underneath it is the subtext: "Making sure your software's what it claims to be.". At the bottom, there is a section titled "In collaboration with" featuring logos from various partners: ChainGuard, Cisco, Google, Hewlett Packard Enterprise, The Linux Foundation, Purdue University, Red Hat, and VMware.

0101
0101

Signing artifacts

The screenshot shows a web browser window for the Sigstore website (<https://www.sigstore.dev>). The main heading is "How sigstore works". Below it, there's a paragraph about how sigstore benefits developers, maintainers, and security experts by bringing together tools like Fulcio, Cosign, and Rekor. There are three sections: "A standardized approach" which explains the use of Fulcio for key transparency; "Building for future integrations" which mentions a partnership with Google, the Linux Foundation, Red Hat, and Purdue University; and "How sigstore works" which includes a flow diagram.

How sigstore works

sigstore is a set of tools developers, software maintainers, package managers and security experts can benefit from. Bringing together free-to-use open source technologies like Fulcio, Cosign and Rekor, it handles digital signing, verification and checks for provenance needed to make it safer to distribute and use open source software.

A standardized approach

This means that open source software uploaded for distribution has a stricter, more standardized way of checking who's been involved, that it hasn't been tampered with. There's no risk of key compromise, so third parties can't hijack a release and slip in something malicious.

Building for future integrations

With the help of a working partnership that includes Google, the Linux Foundation, Red Hat and Purdue University, we're in constant collaboration to find new ways to improve the sigstore technology, to make it easy to adopt, integrate and become a long-lasting standard.

DEVELOPERS, MAINTAINERS, MONITORS

SIGN AND PUBLISH ARTIFACTS

PUBLISH SIGNING CERTIFICATES

MONITOR LOGS

FULCIO CERTIFICATE AUTHORITY

SIGNATURE TRANSPARENCY LOG

KEY TRANSPARENCY LOG

TRUST ROOT

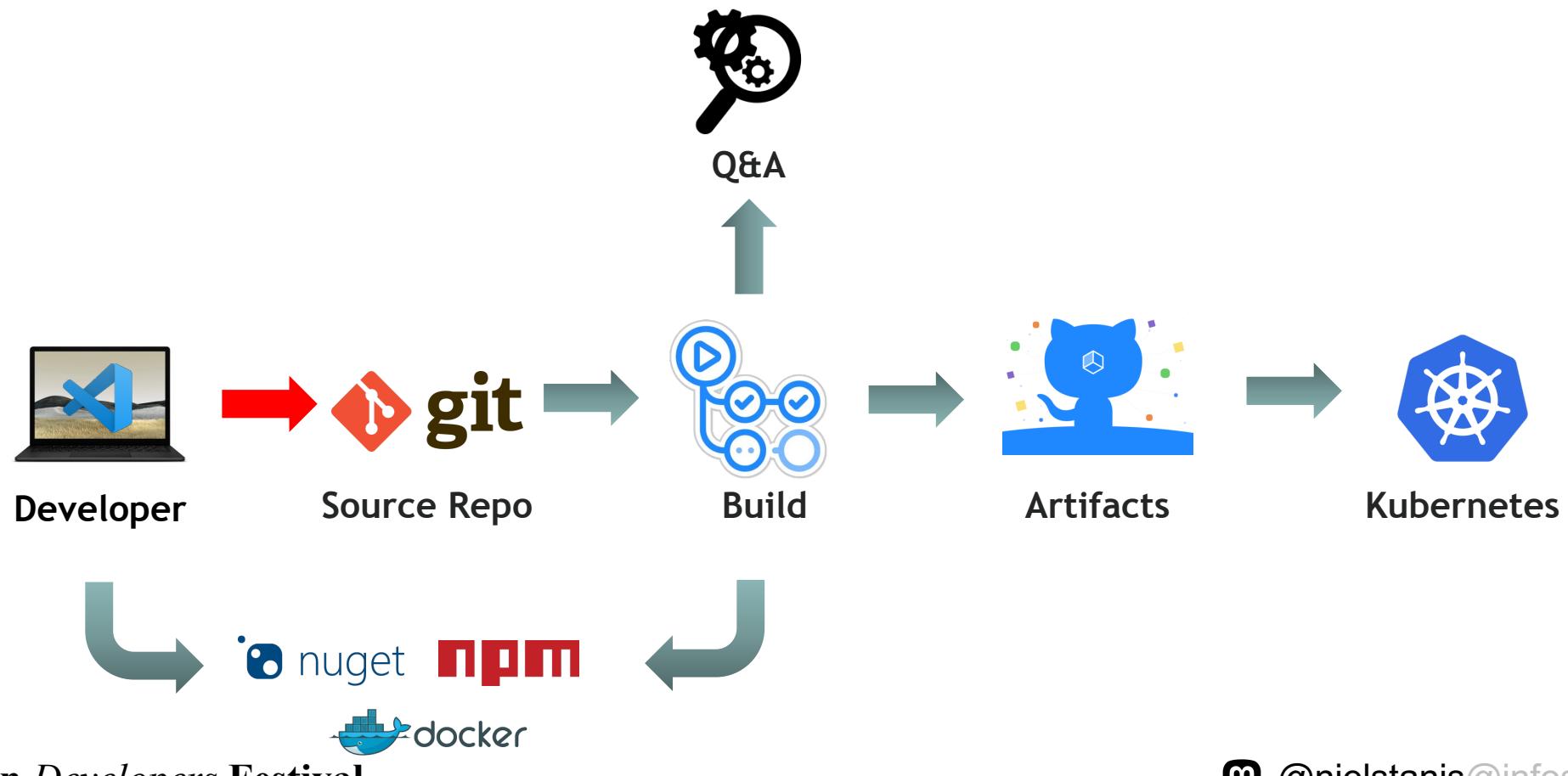
Lab 1

Signing artifacts with
Sigstore

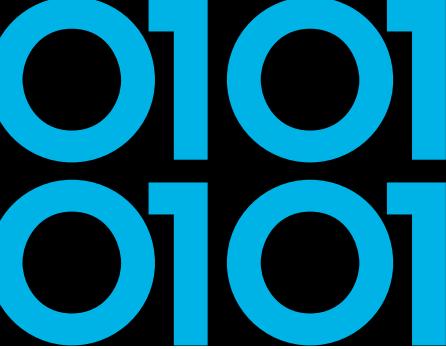


Software Supply Chain

0101
0101



GIT Commit Signing



A screenshot of a GitHub repository page titled "Initial content · nielstanis/ndcsydney2022-supplychain". The repository is public. The main navigation bar includes "Pulls", "Issues", "Marketplace", "Explore", and "Code" (which is underlined). Below the navigation, there are buttons for "Pin", "Unwatch 1", "Fork 0", and "Star 0". The "Code" section shows a commit from "nielstanis" committed 5 minutes ago, which is verified. The commit hash is 054a8a3b9c91e8cc6ec2e9f144d2b9b7dbc8628e. It shows 12 changed files with 3,668 additions and 0 deletions. A file named ".gitignore" is expanded, showing its contents. A "Load diff" button is present at the bottom of the diff view.

Lab 2

Setup repository, build
and git signing with
GitSign





Reproducible/Deterministic Builds



The screenshot shows a dark blue header bar with the "Reproducible Builds" logo on the left. Below it is a light gray sidebar containing links: Home, Contribute, Documentation (which is highlighted in blue), Tools, Who is involved?, News, Events, and 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.

Lab 3

.NET reproducibility





Reproducible/Deterministic Builds

- Roslyn v1.1 started supporting some kind of determinism on how items are emitted
- Given same inputs, the compiled output will always be deterministic
- Inputs can be found in Roslyn compiler docs
‘Deterministic Inputs’



Reproducible/Deterministic Builds

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

Lab 4

DotNet.Reproducible

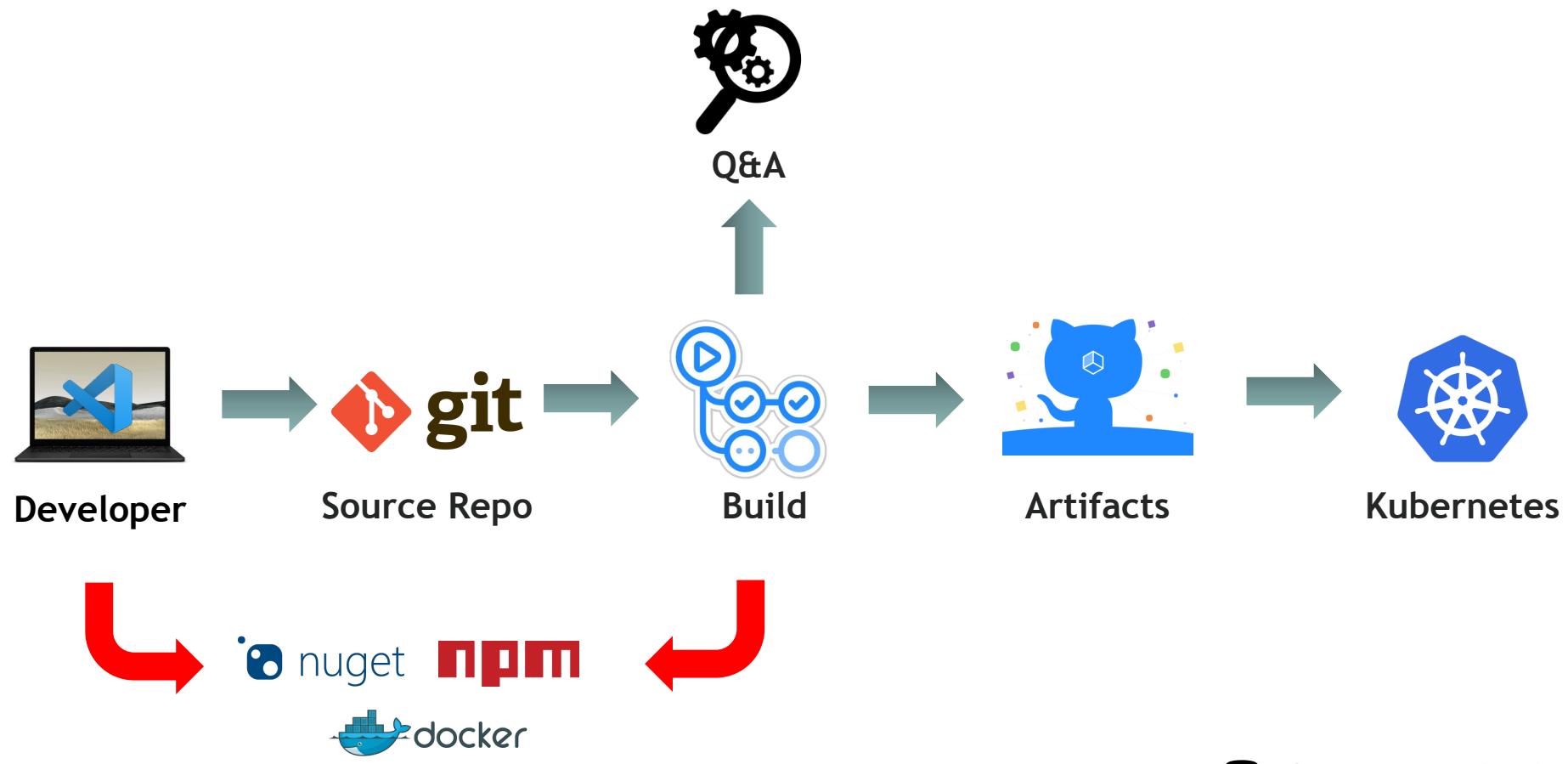




Reproducible Build Validation

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

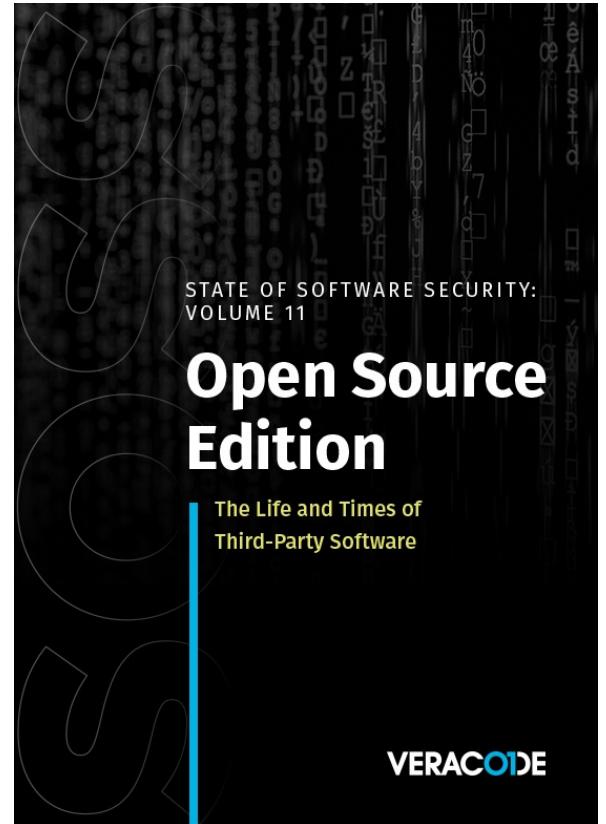
3rd Party Libraries



State Of Software Security v11 2021

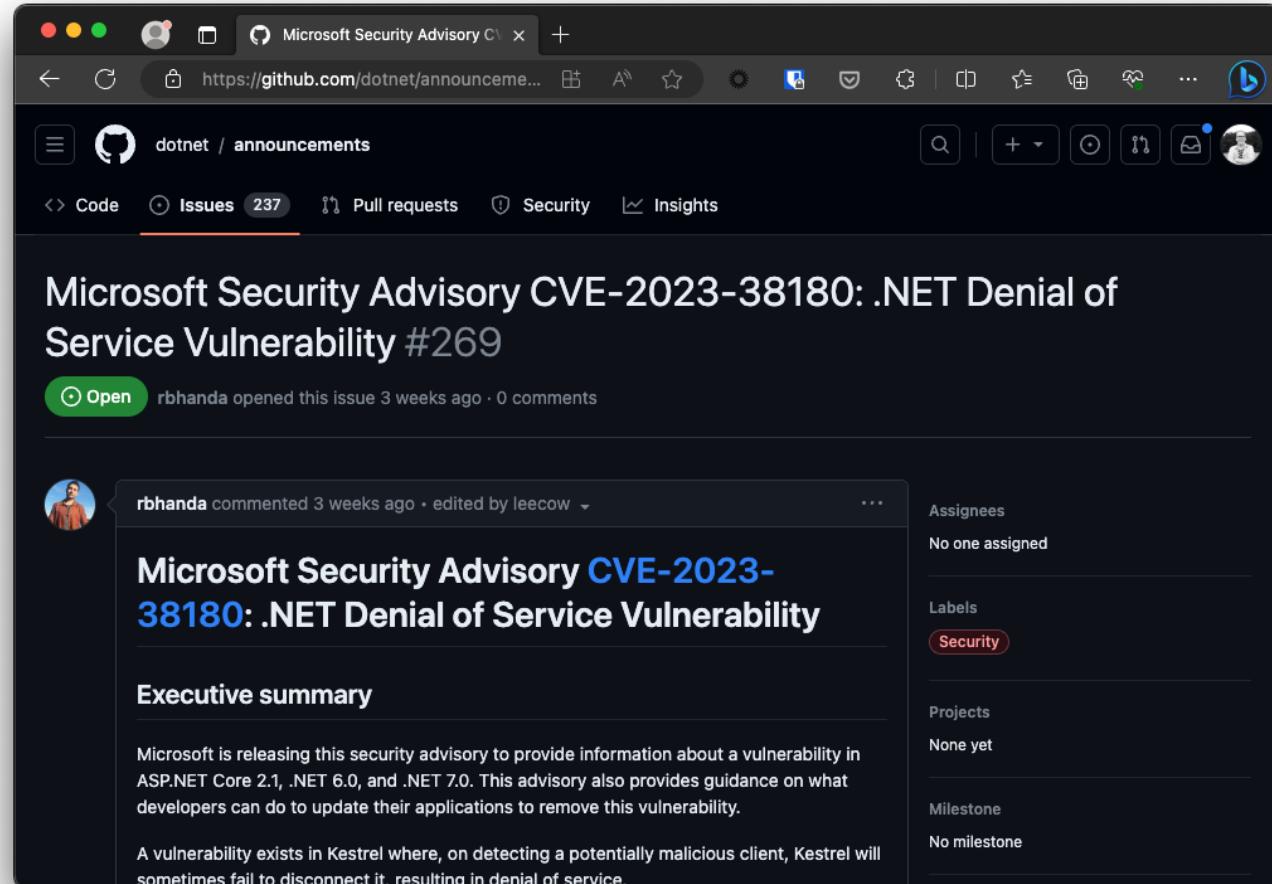


*“Despite this dynamic landscape,
79 percent of the time, developers
never update third-party libraries after
including them in a codebase.”*



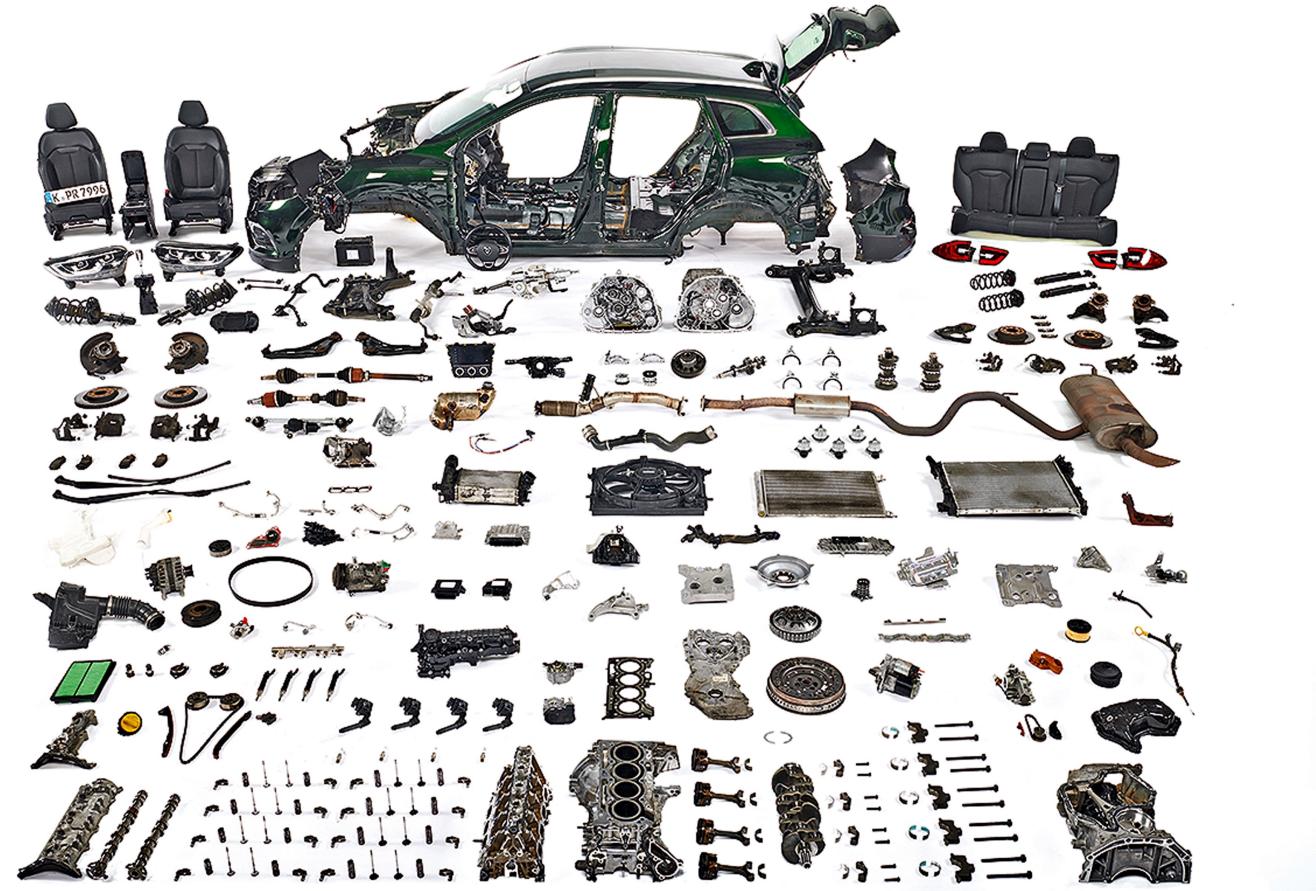
0101
0101

Vulnerabilities in libraries



0101
0101

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

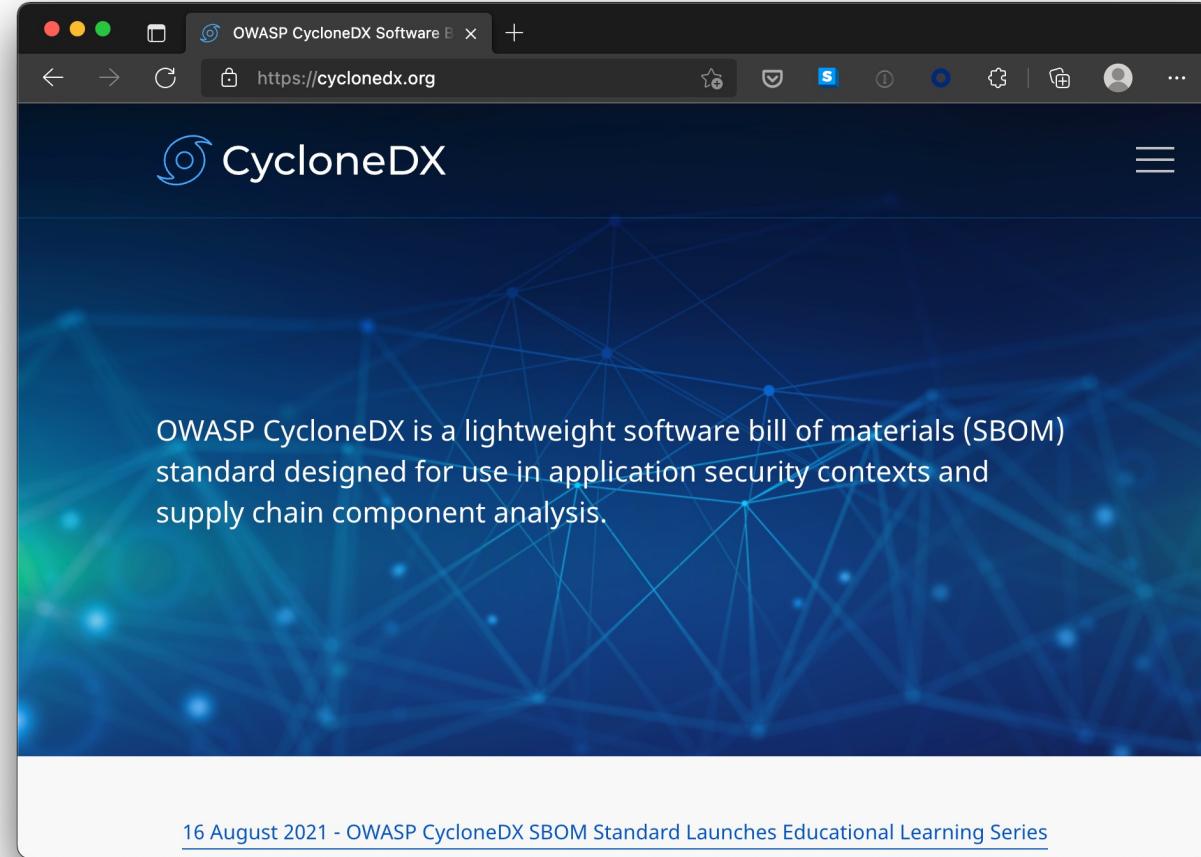


Software Bill of Materials (SBOM)

- 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?

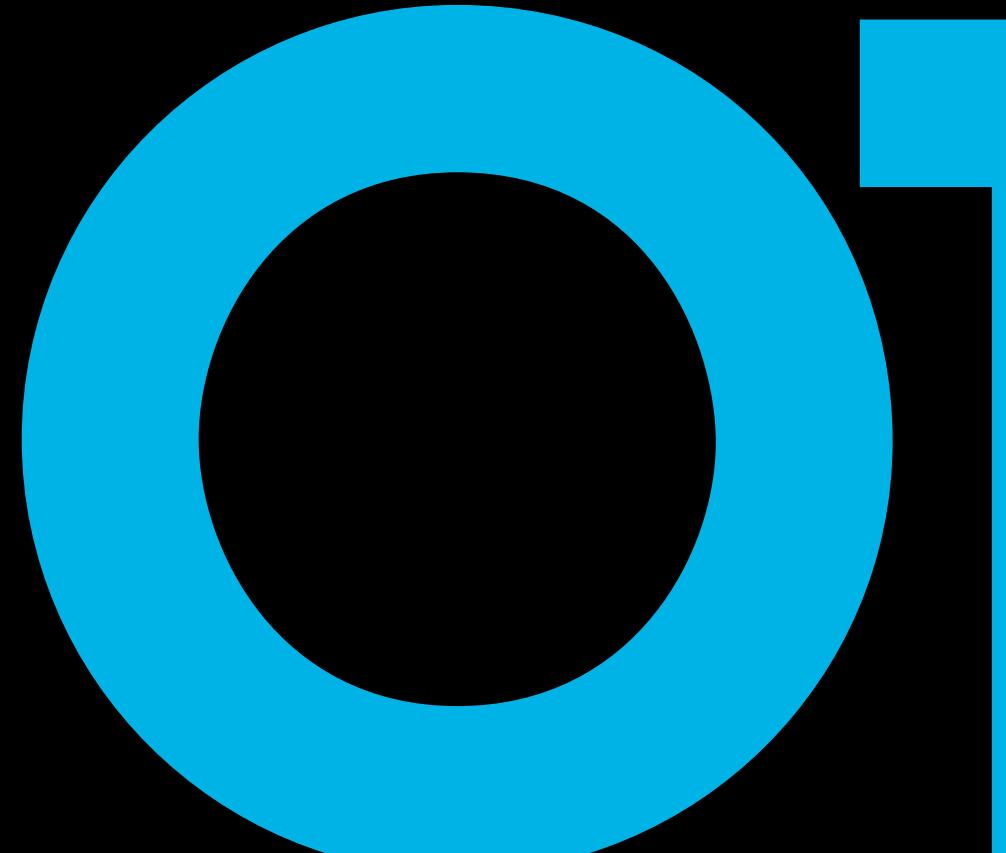


Software Bill of Materials (SBOM)



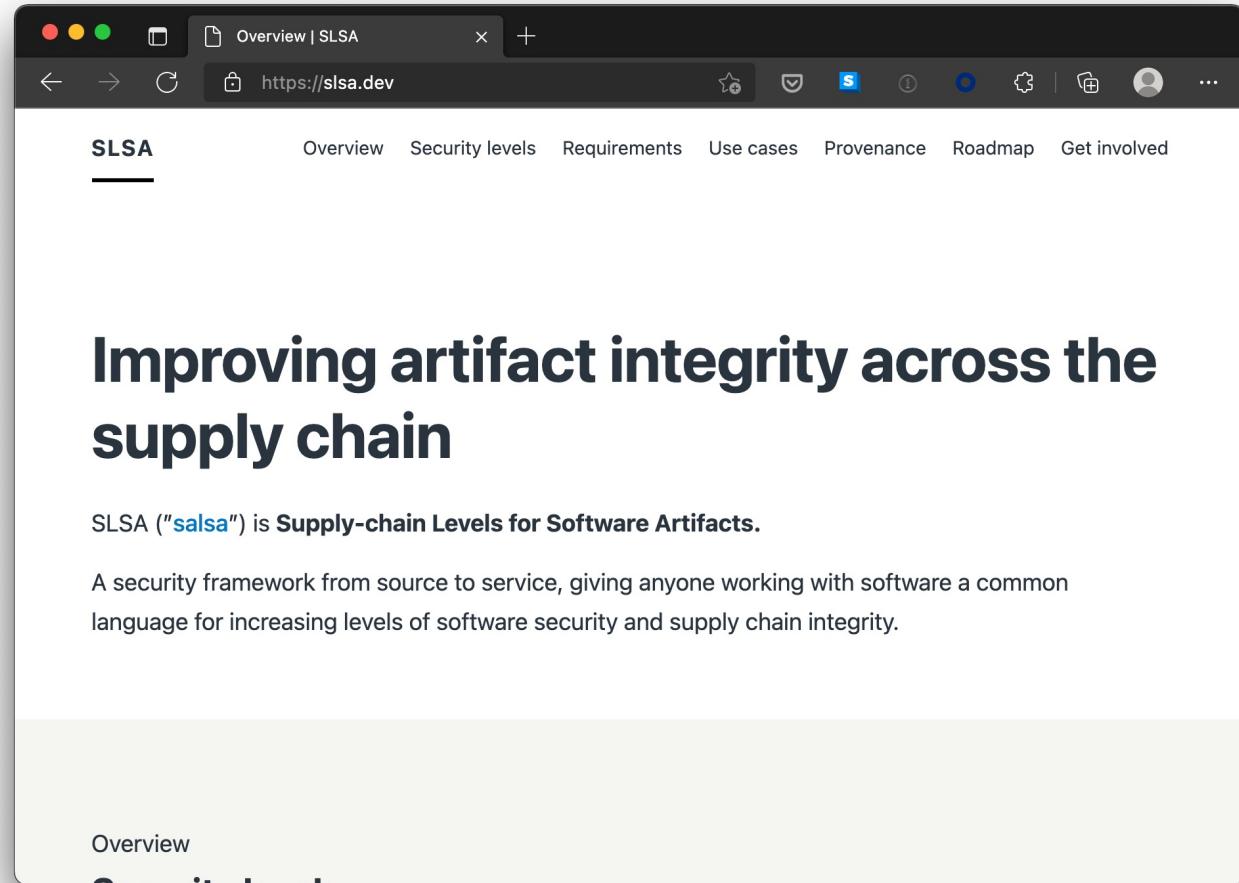
Lab 5

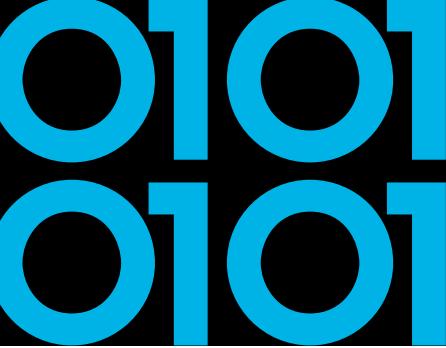
CycloneDX .NET
Any vulnerabilities?



0101
0101

Google SLSA

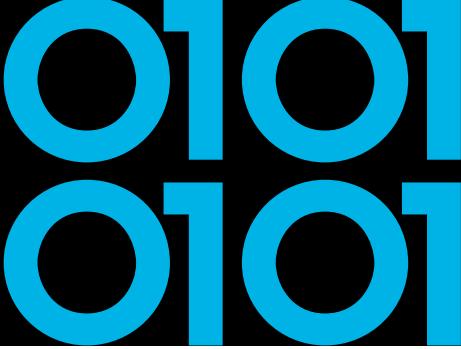




Google SLSA Levels

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

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.



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

SLSA3 Generator GitHub Actions



A screenshot of 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 release of a Go builder and the addition of a new tool for generating provenance documents for projects developed in any programming language, while keeping existing building workflows.

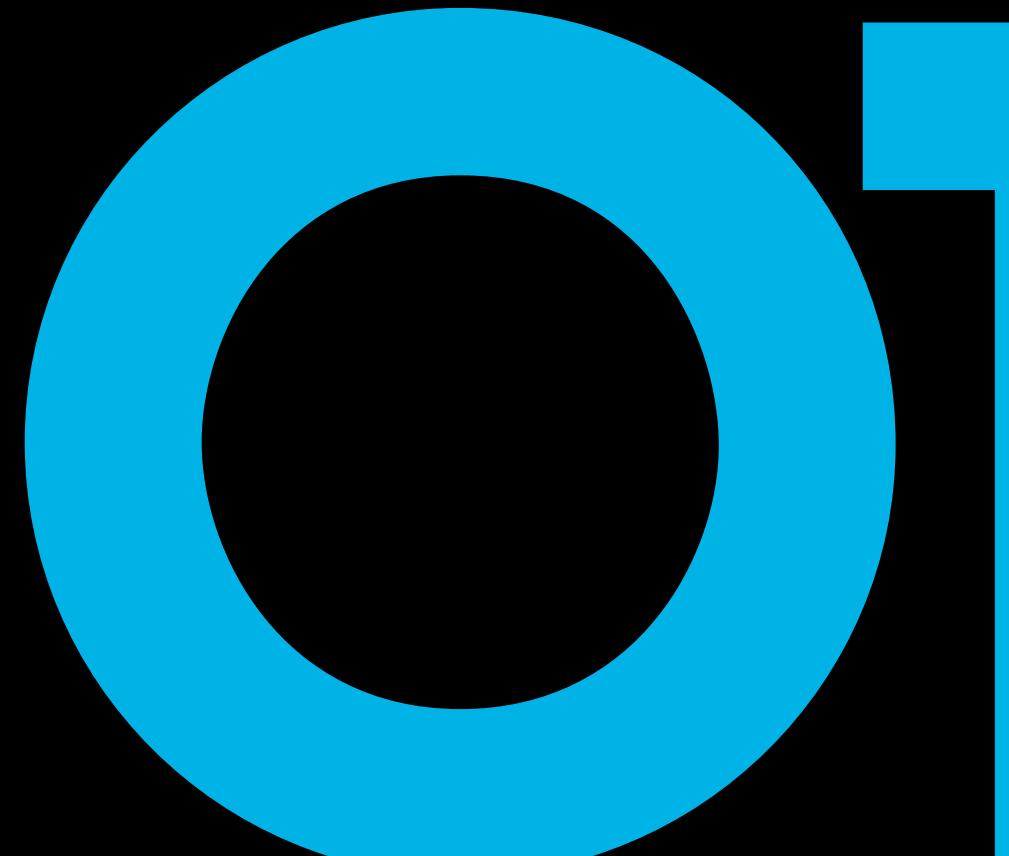
General availability of SLSA3 Generic Generator for GitHub Actions

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.

Lab 6

SLSA Level 3 Provenance

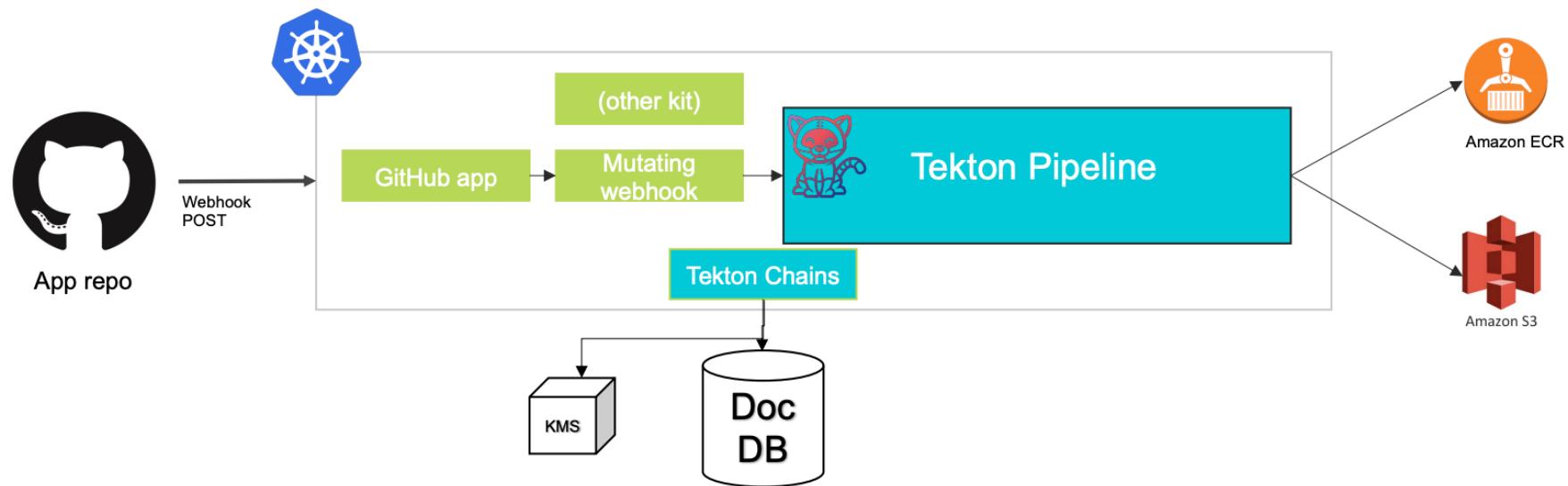




SolarWinds Project Trebuchet



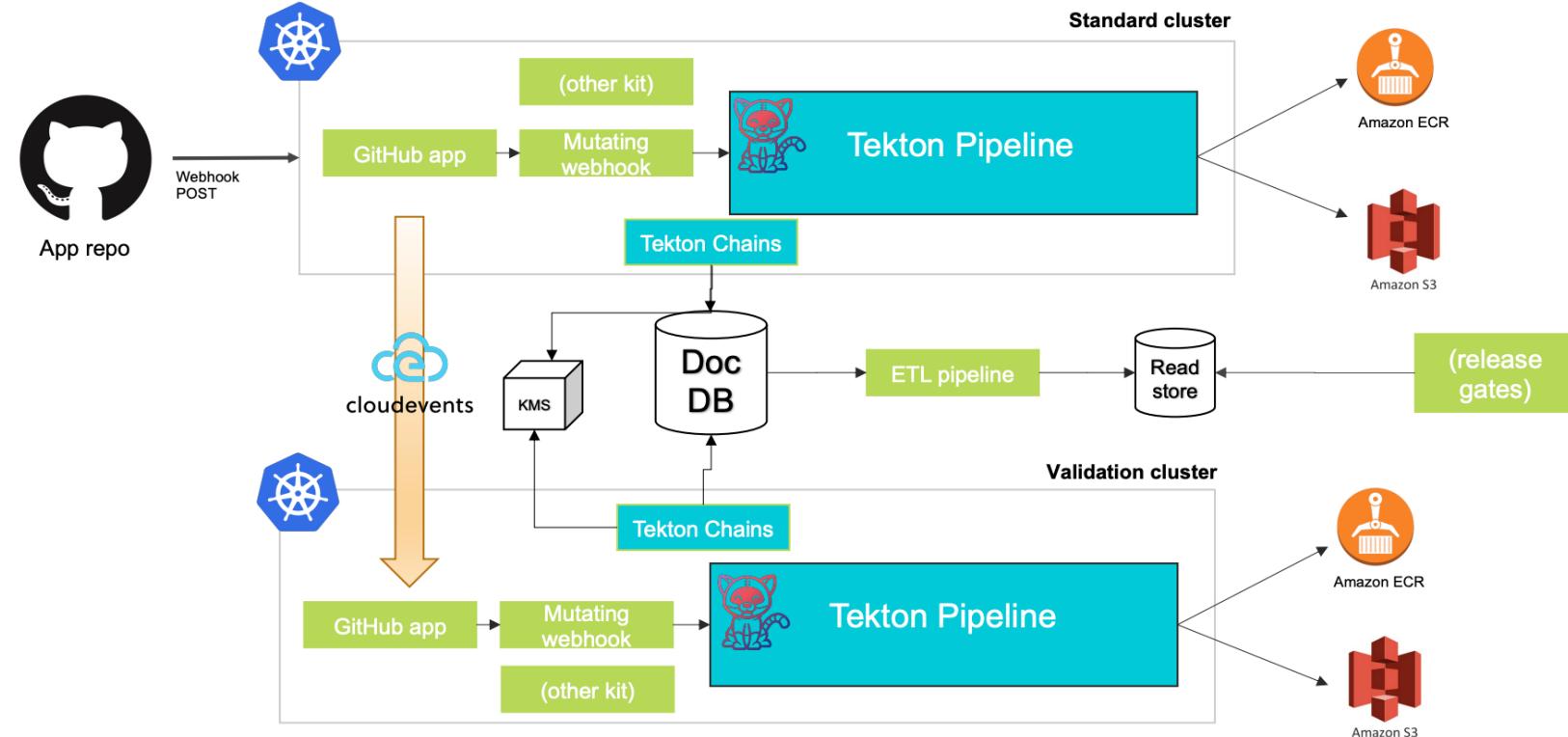
Pipeline With Attestations



SolarWinds Project Trebuchet

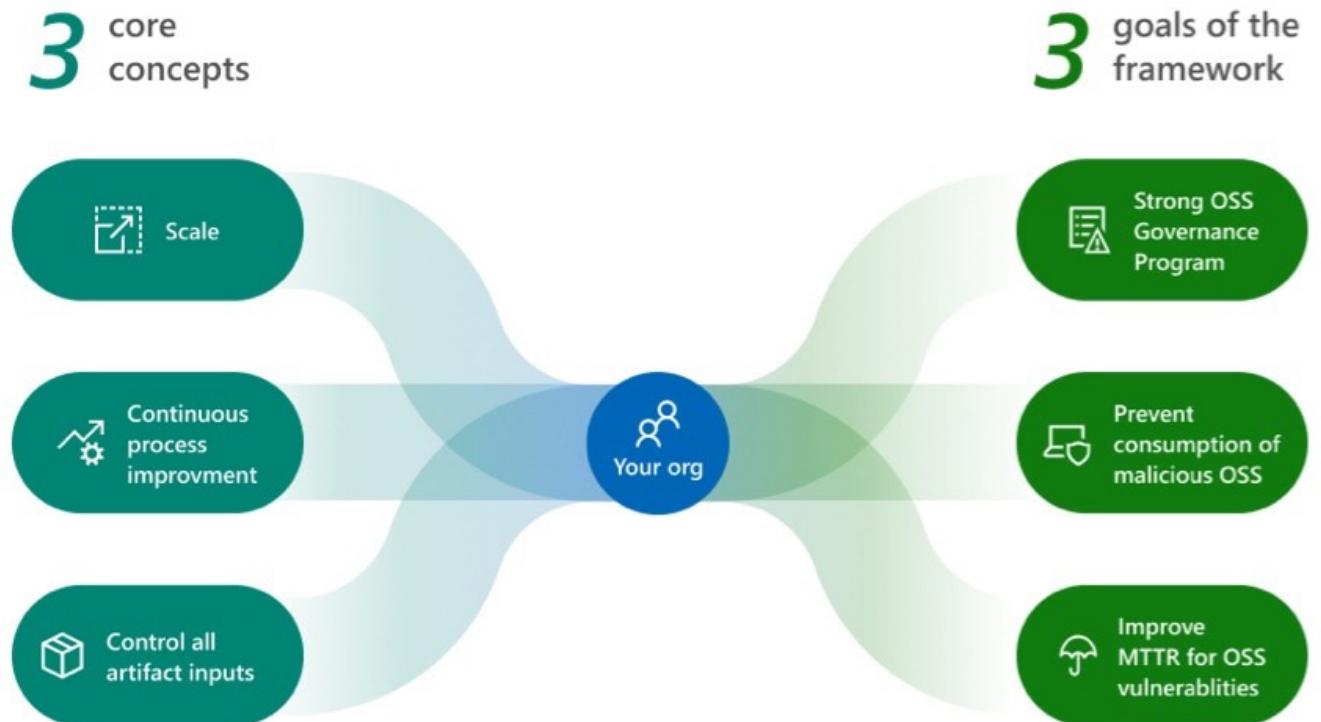


Reading Results

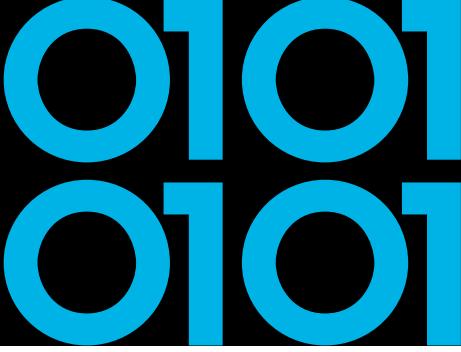


Secure Supply Chain Consumption Framework (S2C2F)

0101
0101



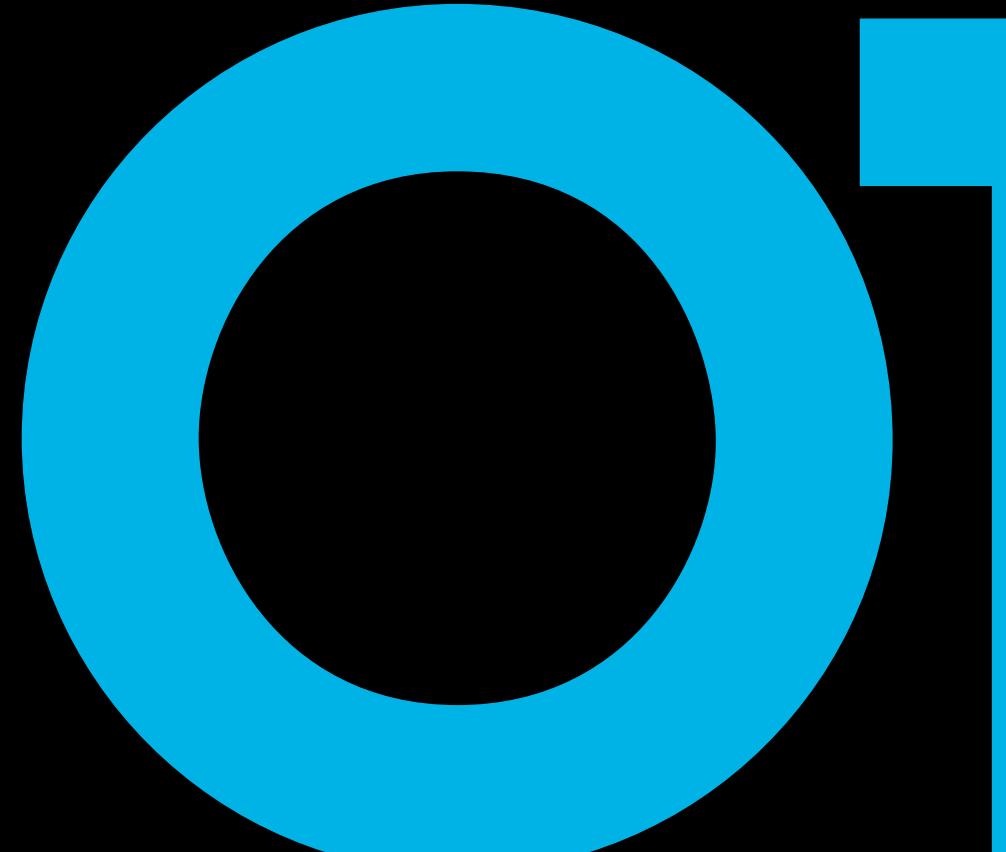
Secure Supply Chain Consumption Framework Practices



1. Ingest It - I can ship any existing asset if external OSS sources are compromised or unavailable
2. Scan It - I know if any OSS artifact in my pipeline has vulnerabilities or malware
3. Inventory It - I know where OSS artifacts are deployed in production.
4. Update It - I can deploy updated external artifacts soon after an update becomes publicly available.
5. Audit It - I can prove that every OSS artifact in production has a full chain-of-custody from the original artifact source and is consumed through the official supply chain
6. Enforce It - I can rely on secure and trusted OSS consumption within my organization.
7. Rebuild It - I can rebuild from source code every OSS artifact I'm deploying.

Lab 7

DocGenerator on
ASP.NET Core MVC &
Docker



Docker SBOM

0101
0101

The screenshot shows a web browser displaying a Docker blog post. The title of the post is "Announcing Docker SBOM: A step towards more visibility into Docker images". Below the title, it says "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". On the right side of the post, there is a sidebar with "Post Tags" and "Categories". The "Post Tags" section includes "docker", "Docker images", "sbom", and "Secure Software Supply Chain". The "Categories" section includes "Community", "Company", "Engineering", "Newsletters", and "Products". The Docker logo is visible at the top left of the page.

Join us for [DockerCon](#) on May 9-10th. Preview the agenda and [register today](#).

Get Started

Announcing Docker SBOM: A step towards more visibility into Docker images

JUSTIN CORMACK
Apr 7 2022

Today, Docker takes its first step in making what is inside your container images more visible so that you can better secure your software supply chain. Included in Docker Desktop 4.7.0 is a new, experimental `docker sbom` CLI command that displays the SBOM (Software Bill Of Materials) of any Docker image. It will

Post Tags

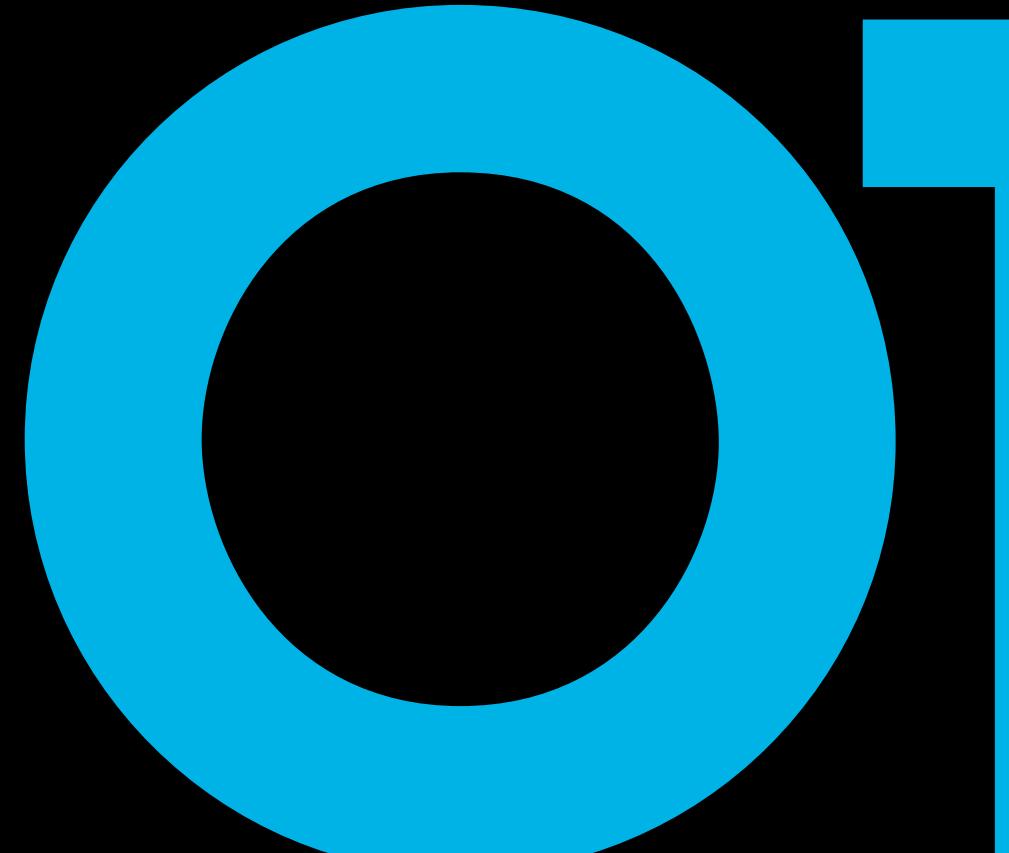
- # docker
- # Docker images
- # sbom
- # Secure Software Supply Chain

Categories

- Community
- Company
- Engineering
- Newsletters
- Products

Lab 8

Docker SBOM with
Anchore Syft



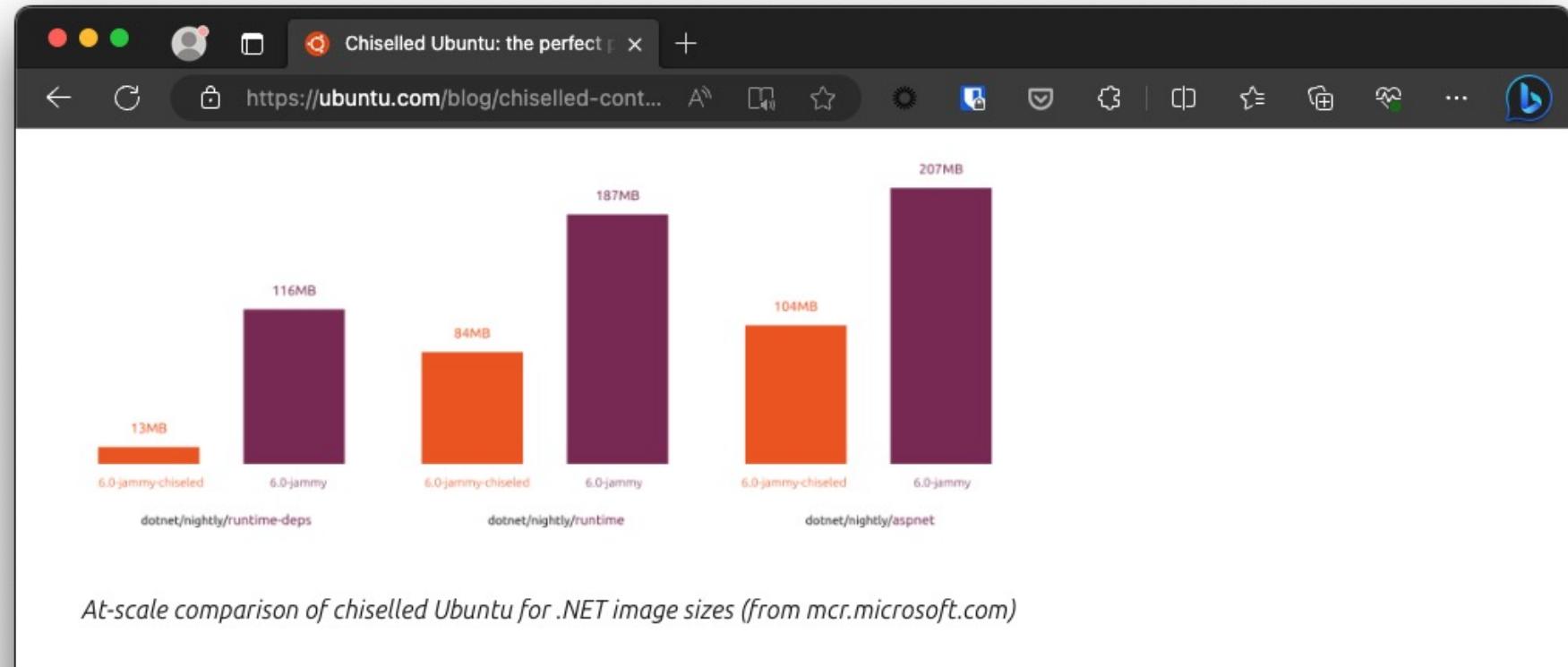
Lab 9

Cosign on Docker in
GitHub Actions





Ubuntu Chiseled - Distroless Linux



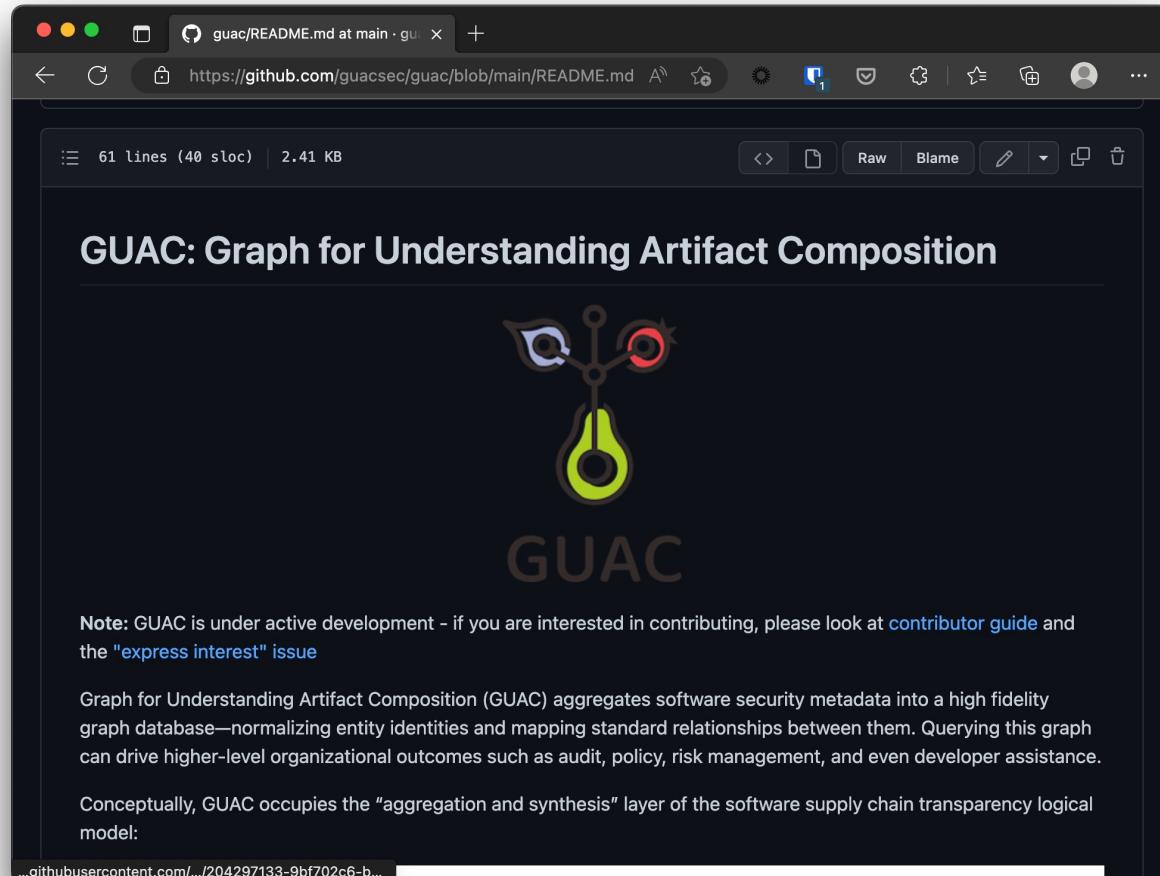
Lab 10

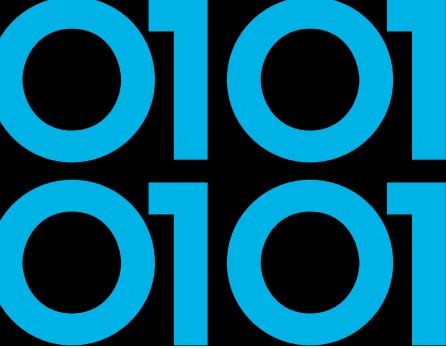
You got HACKED!



GUAC: Graph for Understanding Artifact Composition

0101
0101





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!

<https://github.com/nielstanis/cphdevfest2023>

ntanis at veracode.com

@nielstanis@infosec.exchange on Mastodon

