



Securing your .NET application software supply-chain

Niels Tanis



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



Securing your .NET application software supply-chain

0101
0101



@nielstanis



Agenda

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



@nielstanis

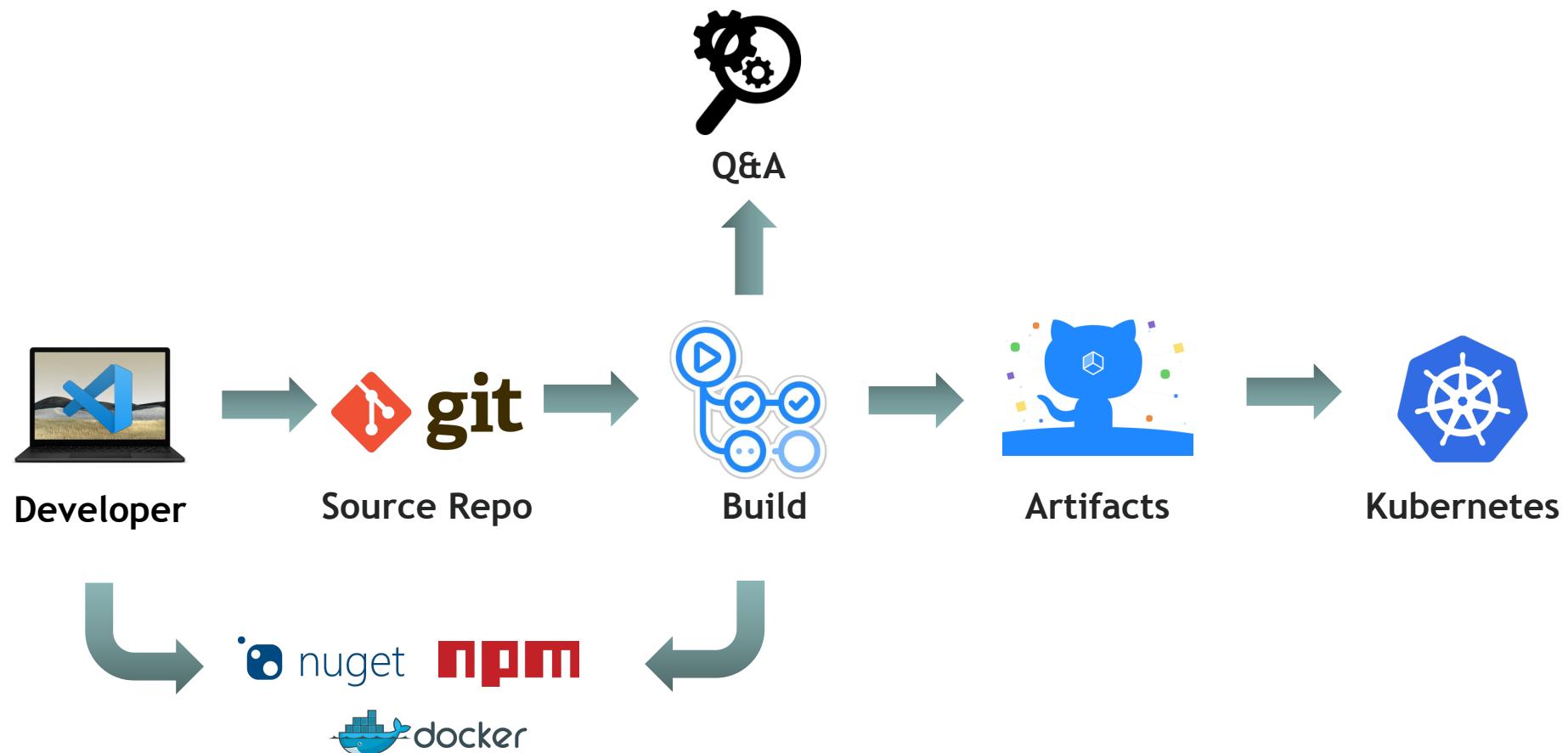
0101
0101

What is a Supply Chain?



Software Supply Chain

0101
0101



0101
0101

SolarWinds SunSpot

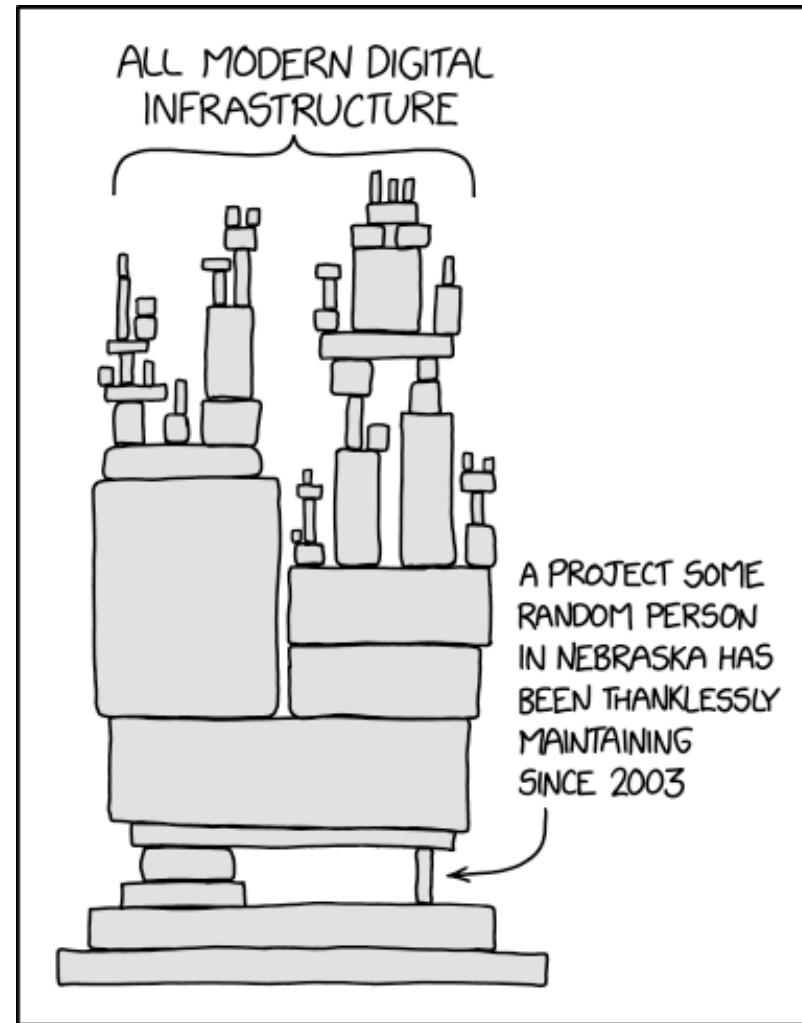
A screenshot of a web browser window showing a blog post from CrowdStrike. The title of the post is "SUNSPOT: An Implant in the Build Process". The post is dated January 11, 2021, and is authored by the CrowdStrike Intelligence Team under the Research & Threat Intel category. The background of the page features a large, stylized graphic of a sun with rays emanating from it.



@nielstanis

0101
0101

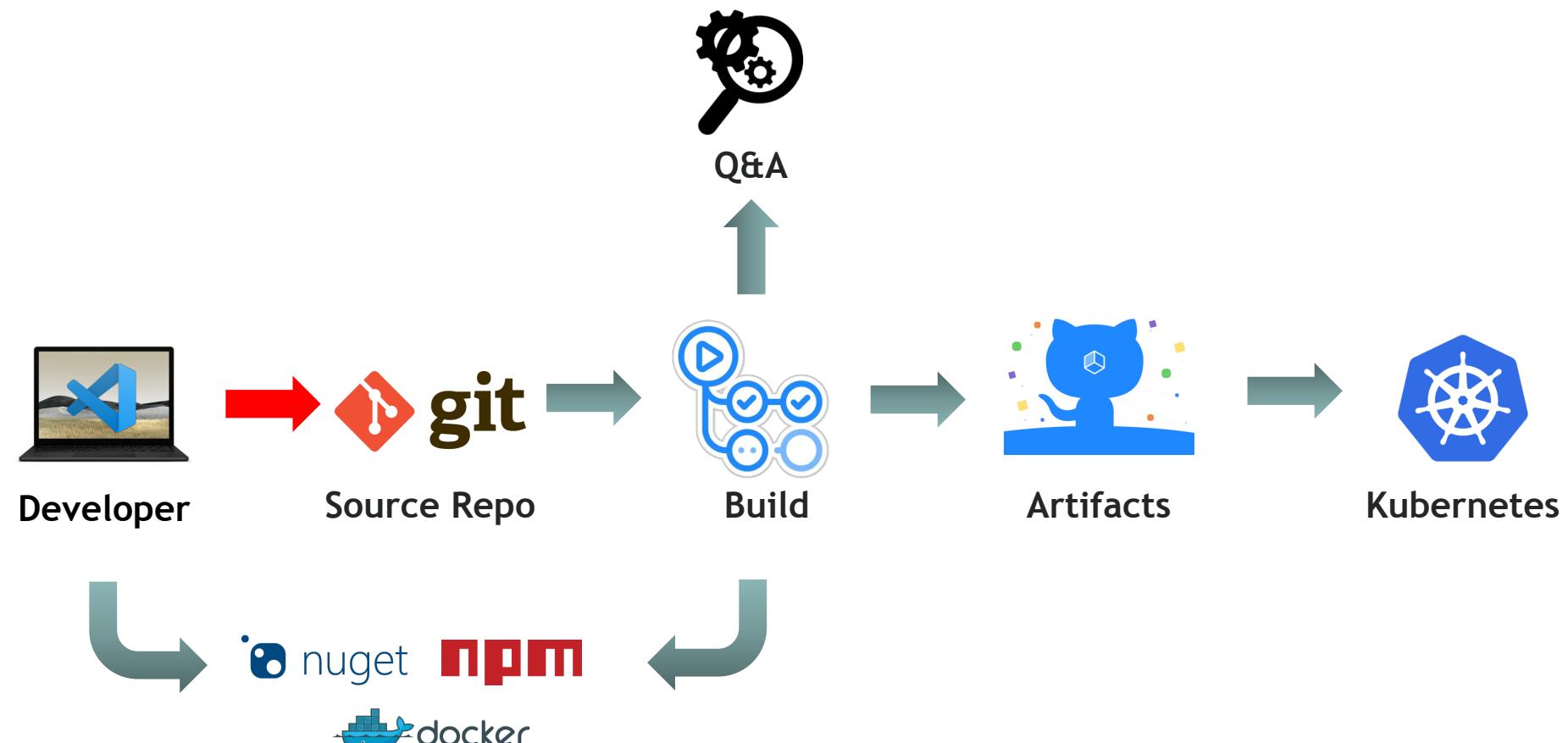
XKDC - Dependency



<https://xkcd.com/2347/>

Software Supply Chain

0101
0101



0101
0101

GitHub account

The screenshot shows a web browser window displaying a ZDNet article. The URL in the address bar is <https://www.zdnet.com/article/canonical-gith...>. The page title is "Canonical GitHub account hacked, Ubuntu source code safe". Below the title, a sub-headline reads "Ubuntu source code appears to be safe; however Canonical is investigating.". The author information indicates the article was written by Catalin Cimpanu for Zero Day on July 7, 2019, and includes a security topic tag. A small image of a GitHub interface for Canonical Ltd. is shown, featuring a repository named "CAN_GOT_HAXXD_10". To the right of the main article, there is a sidebar with a "RELATED" section and a link to another article titled "Why everyone should have this cheap security tool".

Canonical GitHub account hacked, Ubuntu source code safe

Ubuntu source code appears to be safe; however Canonical is investigating.

By Catalin Cimpanu for Zero Day | July 7, 2019 | Topic: Security

Canonical Ltd

CAN_GOT_HAXXD_10

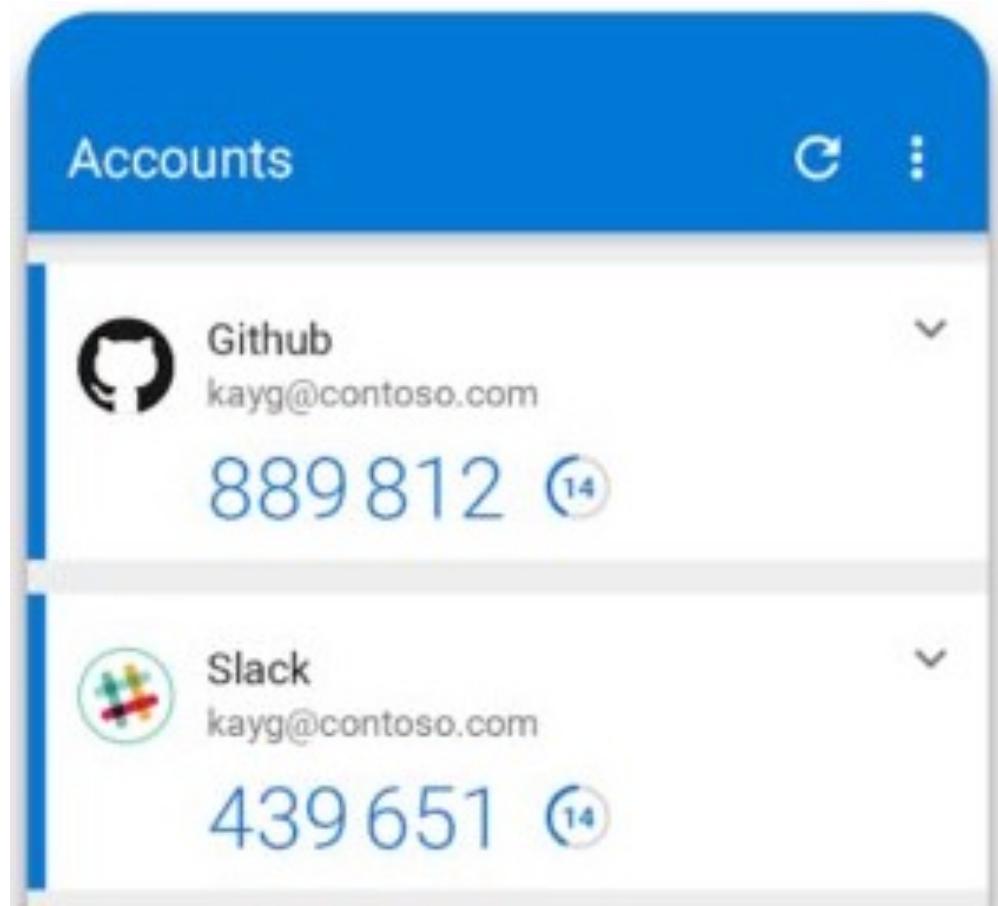
Why everyone should have this cheap security tool



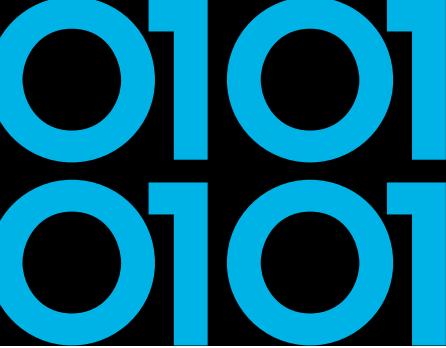
@nielstanis

0101
0101

Use MFA on source-repository



GIT Commit Signing



A screenshot of a GitHub commit page for the repository `nielstanis/FutureTech2022`. The commit was made by `nielstanis` on June 20, 2022. The commit message is "Initial content". The commit is marked as `Verified`, and the SHA-1 hash is `18a6600`. The GitHub interface shows standard navigation links like Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights.



@nielstanis

0101
0101

Hypocrite Commits

The screenshot shows a dark-themed web browser window. The address bar displays the URL <https://github.com/QiushiWu/qiushiwu.github.io/blob/main/papers/OpenS...>. The main content area of the browser shows a research paper with the following details:

Title: On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits

Authors: 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 legitimate commits that introduce security flaws). We propose a novel attack framework that can automatically identify and exploit such vulnerabilities in OSS projects. Our experiments show that our framework can successfully identify and exploit various types of vulnerabilities in real-world OSS projects, including the Linux kernel, Apache, and MySQL. This work highlights the potential risks of OSS and calls for more rigorous security measures in its development process.

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



@nielstanis



Octopus Scanner - NetBeans

May 28, 2020

The Octopus Scanner Malware: Attacking the open source supply chain



Alvaro Muñoz

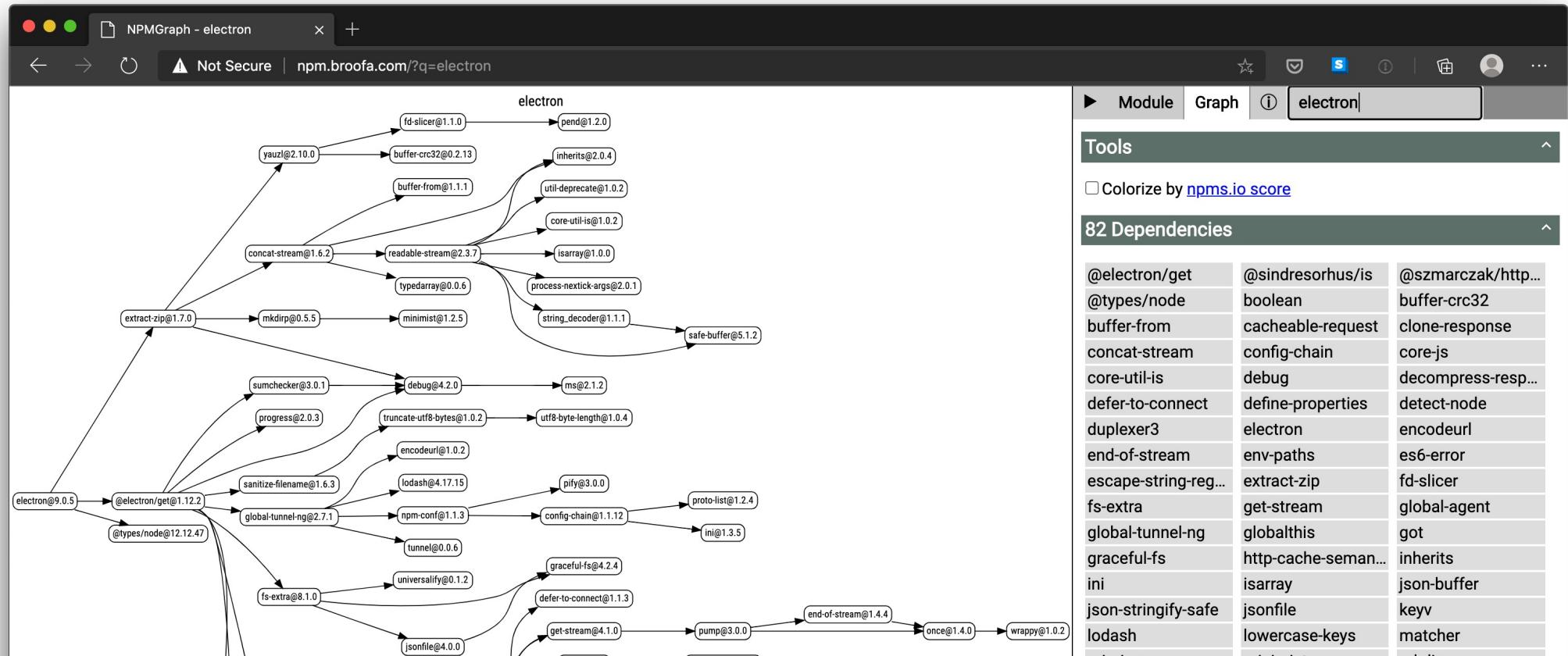
Securing the open source supply chain is an enormous task. It goes far beyond a security assessment or just patching for the latest CVEs. Supply chain security is about the integrity of the entire software development and delivery ecosystem. From the code commits themselves, to how they flow through 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.



@nielstanis

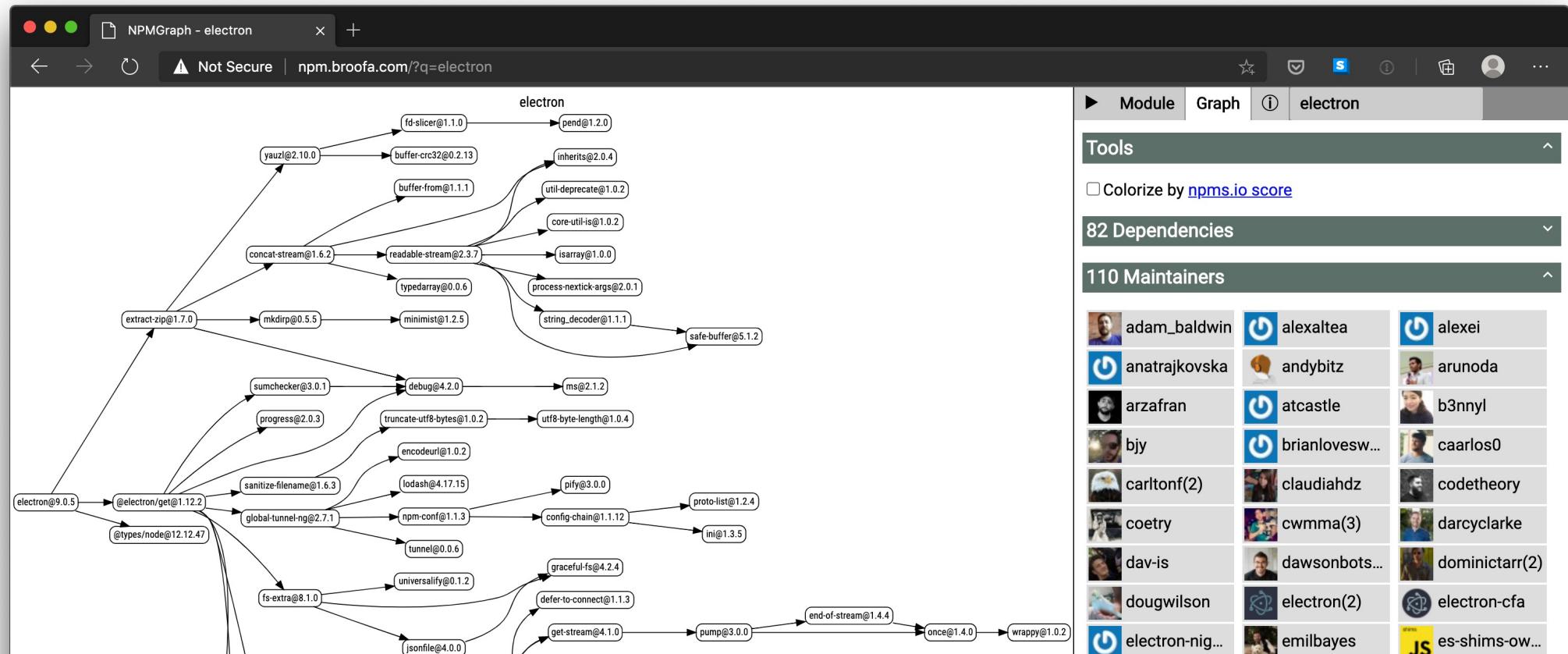
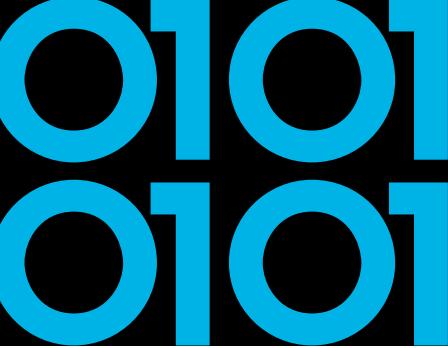
Visual Studio Code

0101
0101



@nielstanis

Visual Studio Code



0101
0101

Visual Studio Code

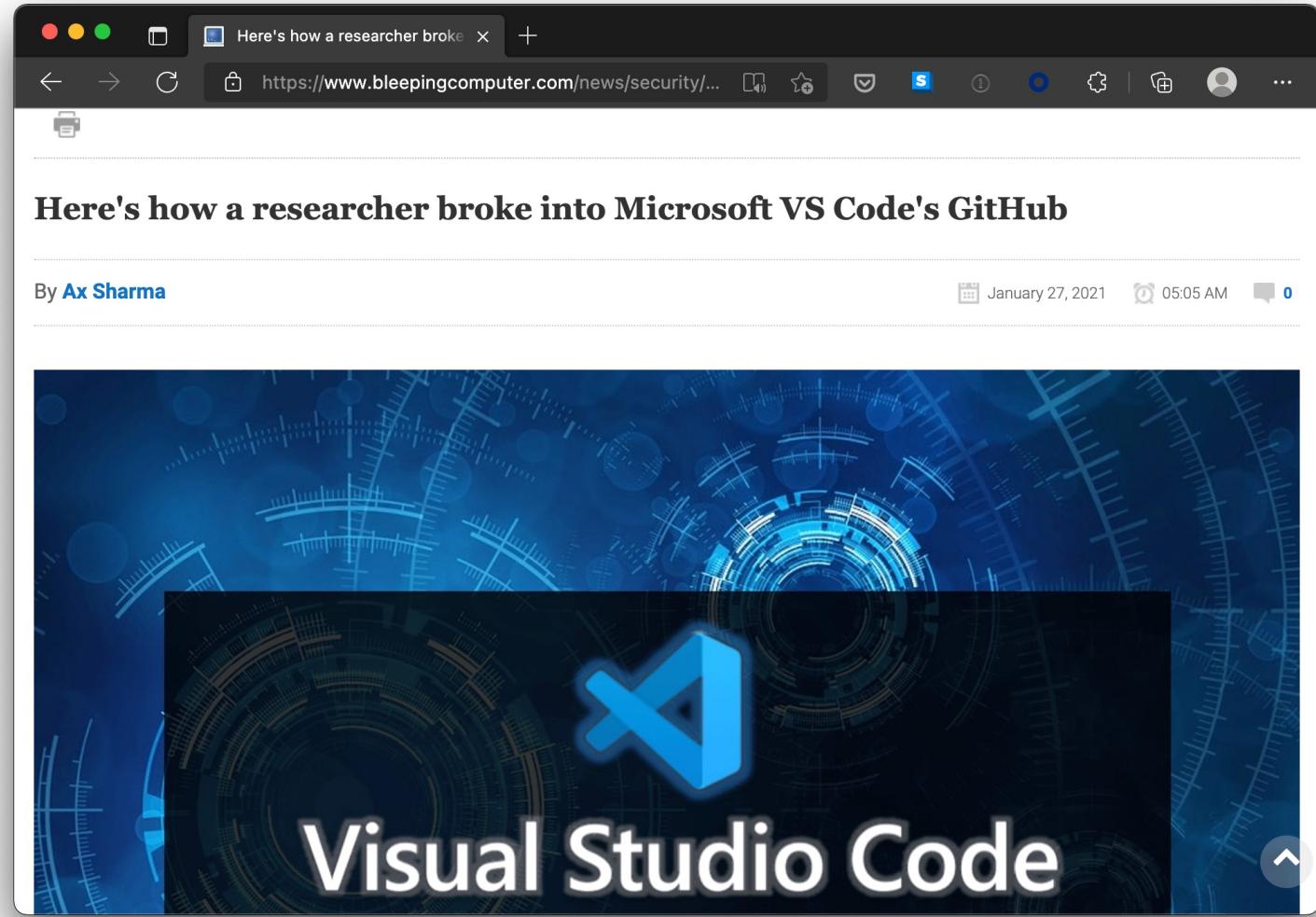
The screenshot shows a web browser displaying the Microsoft MSRC Security Update Guide. The URL in the address bar is <https://msrc.microsoft.com/update-guide/en-US/CVE-2022-30129>. The page title is "CVE-2022-30129 - Security Update". The main content area is titled "Visual Studio Code Remote Code Execution Vulnerability" and includes the identifier "CVE-2022-30129". Below the title, there's a "On this page" section with a dropdown arrow. The page also displays the release date "Released: May 10, 2022", the assigned CNA "Microsoft", the Mitre CVE ID "MitreCVE-2022-30129", and the CVSS score "CVSS:3.1 8.8 / 7.7". At the bottom right, there are "Expand all" and "Collapse all" buttons.



@nielstanis

0101
0101

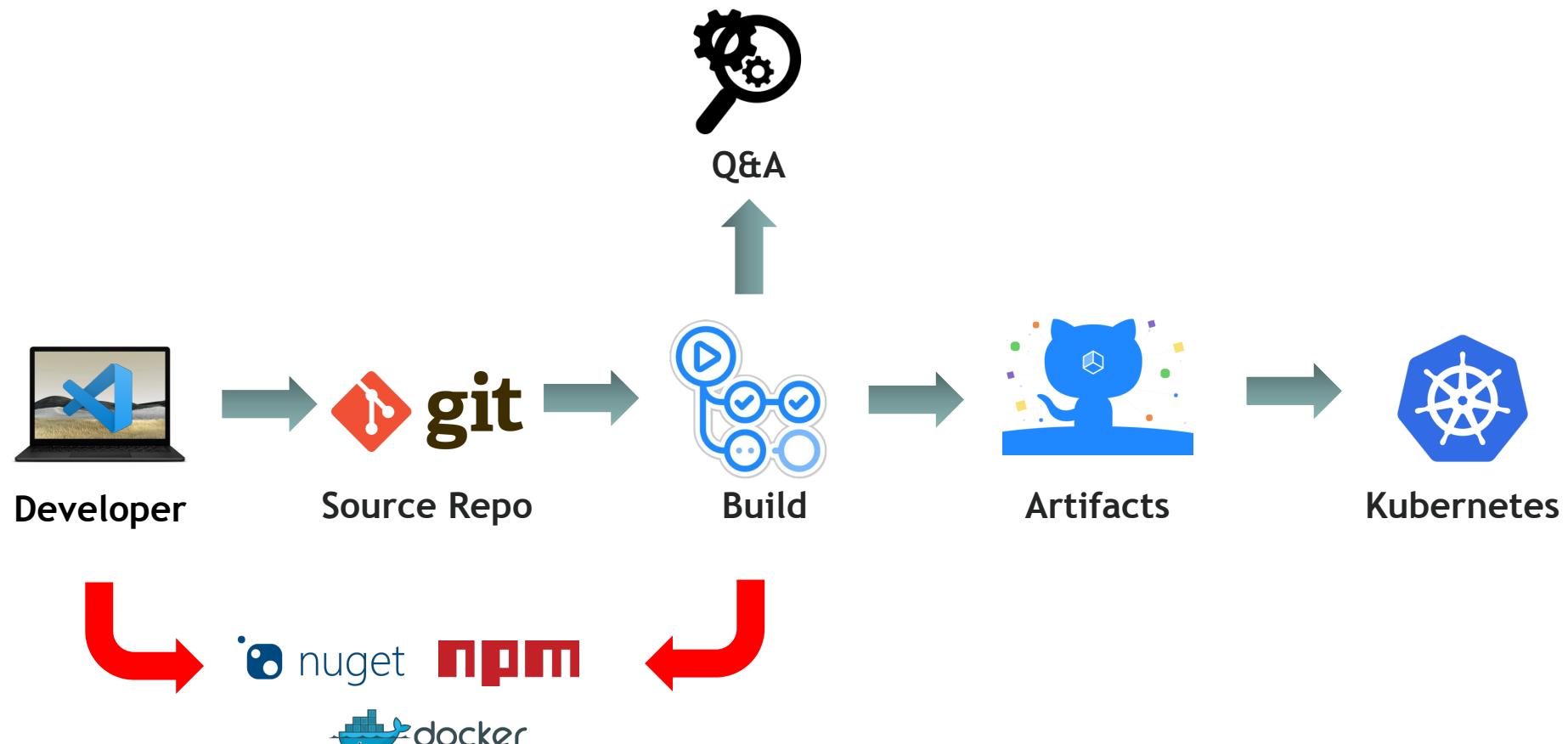
Visual Studio Code



@nielstanis

3rd Party Libraries

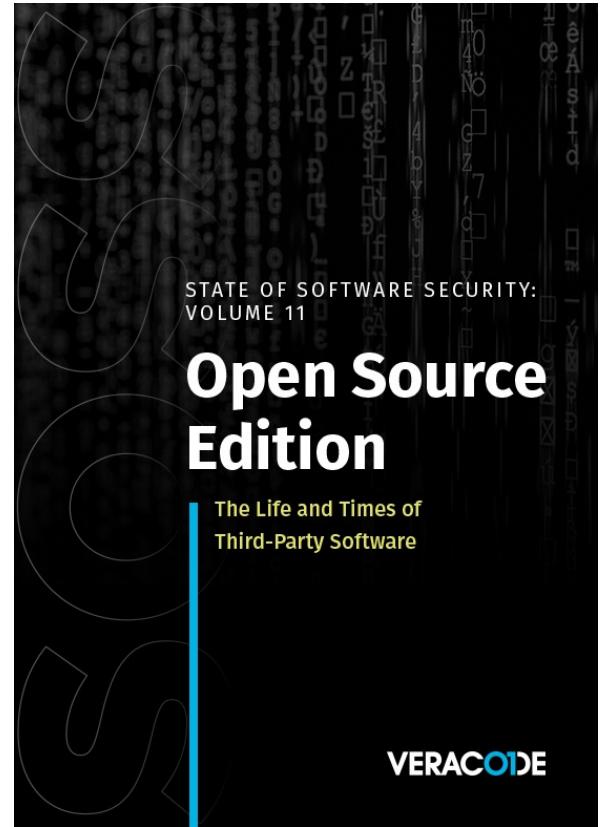
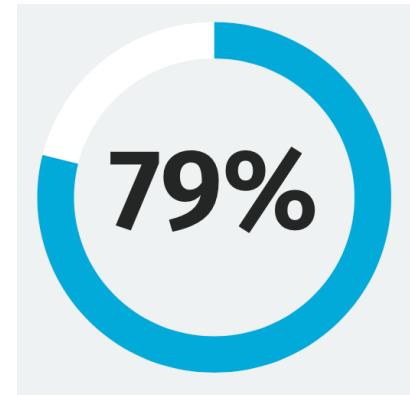
0101
0101



State Of Software Security v11 2021

0101
0101

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



@nielstanis

0101
0101

Vulnerabilities in libraries

The screenshot shows a GitHub issue page for the repository `dotnet/announcements`. The issue is titled "Microsoft Security Advisory CVE-2022-24512 | .NET Remote Code Execution Vulnerability #213". The issue was opened by `dcwhittaker` on March 8th, 2022, and has 0 comments. The issue body contains an executive summary and a discussion section. The executive summary states that Microsoft is releasing a security advisory for a vulnerability in .NET 6.0, .NET 5.0, and .NET Core 3.1. It describes a Remote Code Execution vulnerability in the .NET Double Parse routine where a stack buffer overrun occurs. The discussion section points to another issue at `dotnet/runtime#66348`. On the right side of the issue page, there are sections for Assignees (No one assigned), Labels (Monthly-Update, .NET Core 3.1, .NET 5.0, .NET 6.0, Patch-Tuesday, Security), Projects (None yet), and Milestone (No milestone).



@nielstanis

Vulnerabilities in libraries

0101
0101



Alerts and Tips Resources Industrial Control Systems

National Cyber Awareness System > Current Activity > Malware Discovered in Popular NPM Package, ua-parser-js

Malware Discovered in Popular NPM Package, ua-parser-js

Original release date: October 22, 2021

[Print](#) [Tweet](#) [Send](#) [Share](#)

Versions of a popular NPM package named [ua-parser-js](#) was found to contain malicious code. [ua-parser-js](#) is used in apps and websites to discover the type of device or browser a person is using from User-Agent data. A computer or device with the affected software installed or running could allow a remote attacker to obtain sensitive information or take control of the system.

CISA urges users and administrators using compromised ua-parser-js versions 0.7.29, 0.8.0, and 1.0.0 to update to the respective patched versions: 0.7.30, 0.8.1, 1.0.1

For more information, see [Embedded malware in ua-parser-js](#).



@nielstanis

0101
0101

Vulnerabilities in libraries

The image shows two side-by-side screenshots of a Mac OS X desktop environment. Both screenshots display a web browser window with a dark theme, showing GitHub blog posts.

Left Screenshot: The title is "GitHub's commitment to npm ecosystem security". It features a sub-section "Open Source Security". Below the title is a paragraph: "We're sharing details of recent incidents on the npm registry, our investigations, and how we're continuing to invest in the security of npm." A large, stylized blue illustration of a hand assembling a yellow puzzle piece is centered below the text. At the bottom, it says "Author" followed by a profile picture of Mike Hanley and the name "Mike Hanley", with the date "November 15, 2021" to the right.

Right Screenshot: The title is "Enrolling all npm publishers in enhanced login verification and next steps for two-factor authentication enforcement". It also has an "Open Source Security" section. Below the title is a paragraph: "Today we're introducing enhanced login verification to the npm registry, and we will begin a staged rollout to maintainers beginning Dec 7." The GitHub logo is at the top left, and the "Blog" section of the GitHub navigation bar is visible. The "Subscribe" button and search icon are also present.



@nielstanis

0101
0101

Dependency Confusion

A screenshot of a web browser window showing a Medium article. The title of the article is "Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies" by Alex Birsan. The article is described as having a 11 min read time. Below the title is a large image of colorful shipping containers stacked together against a blue sky.

Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies

The Story of a Novel Supply Chain Attack

Alex Birsan Feb 9 · 11 min read •

Twitter Facebook LinkedIn Share ...



@nielstanis



Microsoft Whitepaper

3 ways to mitigate risk when using private package feeds

Secure Your Hybrid Software Supply Chain

An always-up-to-date version of this whitepaper is located at: <https://aka.ms/pkg-sec-wp>



@nielstanis



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





3rd Party Libraries

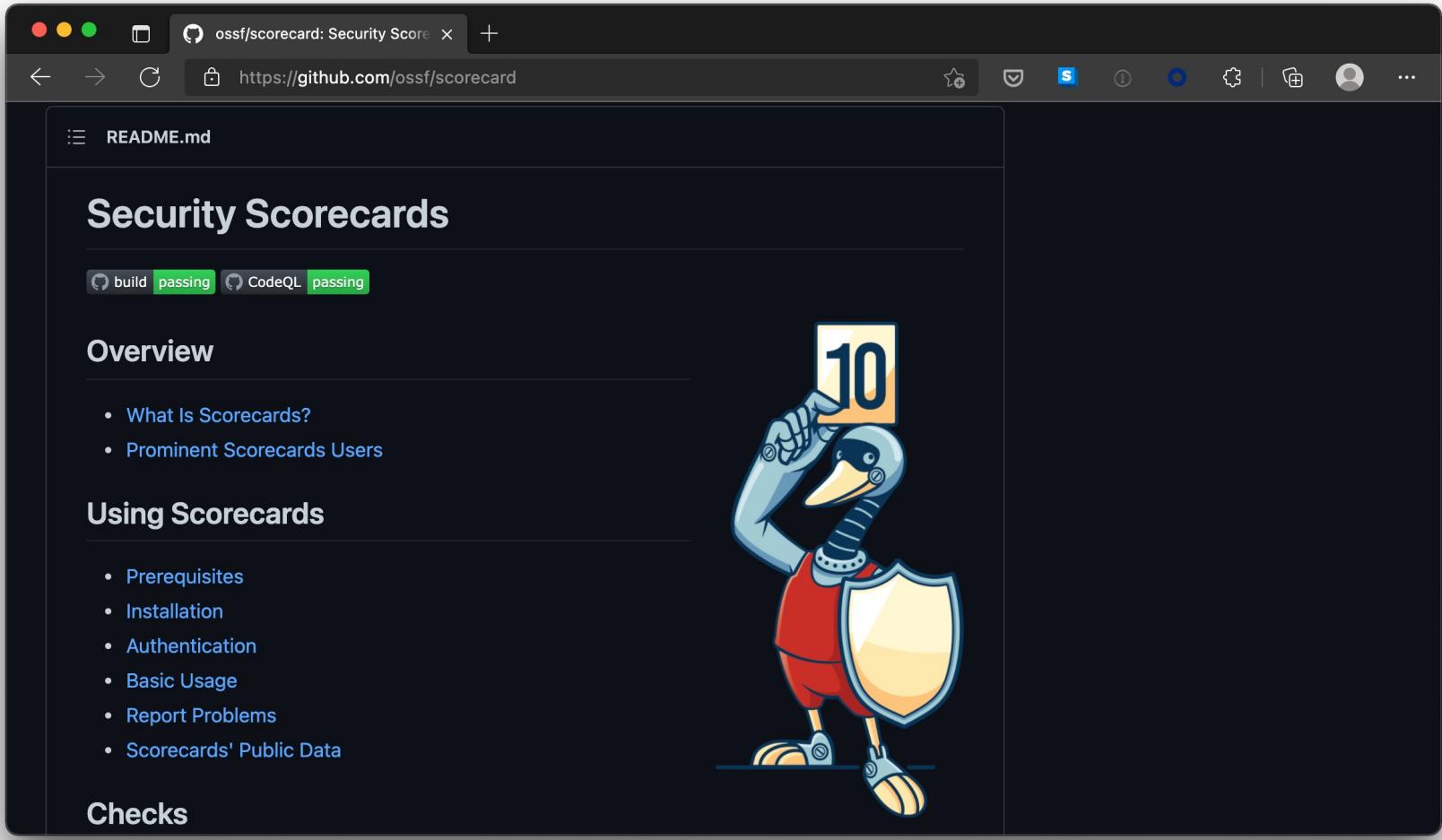
- Intent of library, know what's inside!
- Keep in mind that's a transitive list of dependencies
- Other talk 'Sandboxing .NET Assemblies' @ NDC Porto

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



0101
0101

Security Scorecards - OpenSSF



The screenshot shows a GitHub README page for the repository `ossf/scorecard`. The title is **Security Scorecards**. Below it, there are status indicators for build and CodeQL, both showing **passing**. The **Overview** section contains two bullet points:

- What Is Scorecards?
- Prominent Scorecards Users

 The **Using Scorecards** section lists eight items:

- Prerequisites
- Installation
- Authentication
- Basic Usage
- Report Problems
- Scorecards' Public Data

 At the bottom, there is a **Checks** section. To the right of the text, there is a cartoon illustration of a blue bird-like character wearing red pants and a yellow shield, holding up a yellow card with the number **10**.



@nielstanis

Deps.Dev by Google

0101
0101

The screenshot shows a web browser window for 'Open Source Insights' at <https://deps.dev>. The page has a dark background with a teal header bar. The header bar contains the text 'Open Source Insights is an experimental project by Google.' Below the header, the main navigation includes 'open/source/insights', 'About', 'Documentation', and 'Blog'. The main content area features a large heading 'Understand your dependencies' followed by a descriptive paragraph about dependency management. At the bottom of this section is a search bar with a magnifying glass icon, the placeholder 'Search for open source packages', a dropdown menu labeled 'All systems', and a blue 'Search' button. To the right of the search bar is a list of dependency statistics:

npm PACKAGES	1.80M
Go MODULES	717k
Maven ARTIFACTS	427k
PyPI PACKAGES	325k
Cargo CRATES	72k
NuGet PACKAGES	Coming soon



@nielstanis

Deps.Dev by Google



[electron/electron](#)

GitHub

:electron: Build cross-platform desktop apps with
JavaScript, HTML, and CSS

↗ 14k forks

★ 102k stars

OpenSSF scorecard

The Open Source Security Foundation is a cross-industry collaboration to improve the security of open source software (OSS). The [Scorecard](#) provides security health metrics for open source projects.

[View information about checks and how to fix failures.](#)

SCORE

5.8/10

Scorecard as of April 25, 2022.

▶ Code-Review	5/10
▶ Maintained	10/10
▶ CII-Best-Practices	0/10
▶ Vulnerabilities	10/10
▶ Dependency-Update-Tool	0/10
▶ Security-Policy	10/10
▶ Dangerous-Workflow	10/10
▶ Token-Permissions	0/10
▶ License	10/10
▶ Pinned-Dependencies	8/10
▶ Binary-Artifacts	10/10
▶ Fuzzing	0/10
▶ Signed-Releases	0/10

Project metadata as of May 7, 2022.



@nielstanis

0101
0101

Source Generators

A screenshot of a web browser window showing a blog post. The title of the post is ".NET 5, Source Generators, and Supply Chain Attacks". It is written by Mateusz Krzeszowiec and published on September 30, 2021. The post discusses IDEs and build infrastructure as targets for threat actors, mentioning XcodeGhost as an example. The browser interface includes a navigation bar with back, forward, and search buttons, as well as a toolbar with various icons.

.NET 5, Source Generators, and Supply Chain Attacks

By **Mateusz Krzeszowiec** | September 30, 2021

Secure Development

Share this article: [Twitter](#) [Facebook](#) [LinkedIn](#)

IDEs and build infrastructure are being a target of various threat actors since at least 2015 when XcodeGhost has been discovered

- <https://en.wikipedia.org/wiki/XcodeGhost> - malware-ridden Apple Xcode IDE that enabled attackers to plant malware in iOS applications built using it.



@nielstanis



Source Generators

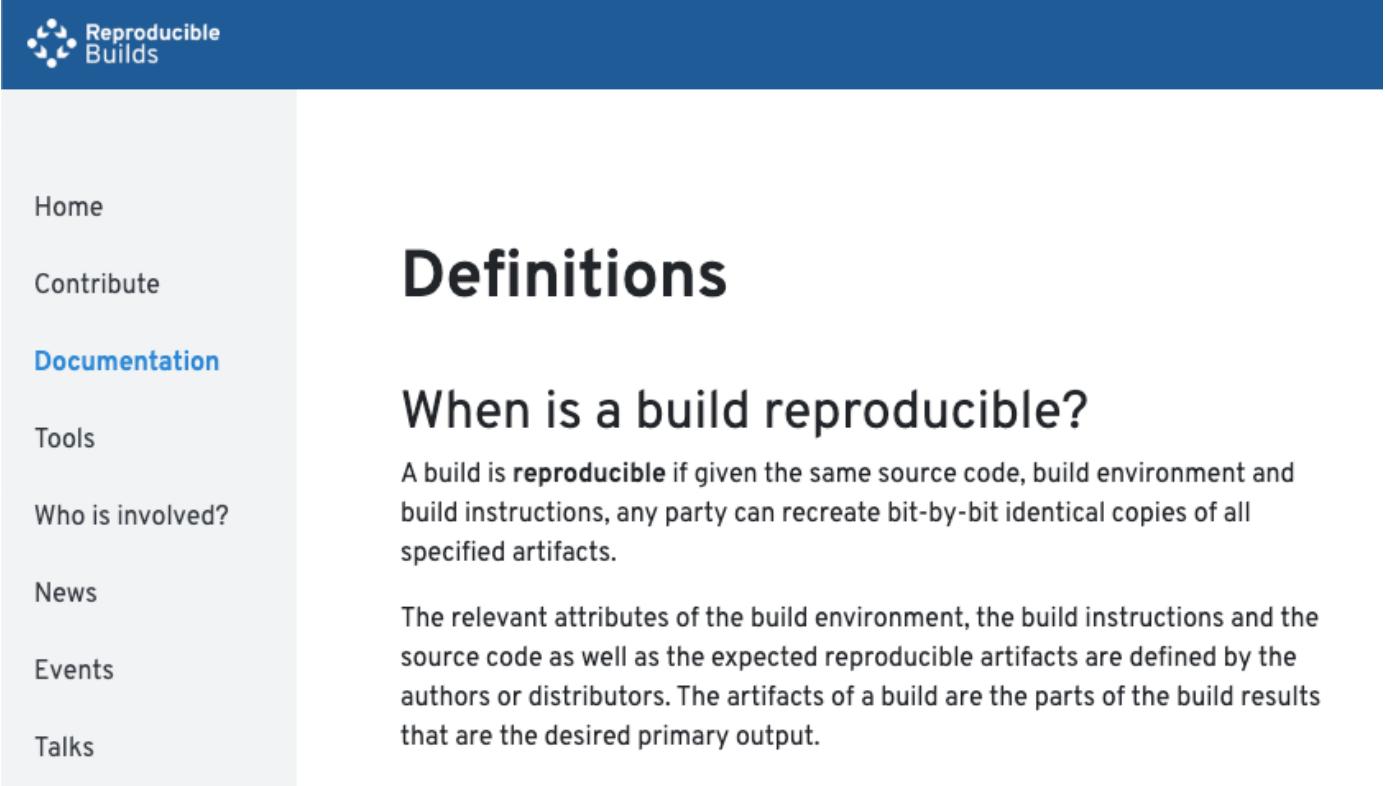
- Any 3rd party library can contain Source Generator!
- Consider disabling on project-level:

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





Reproducible/Deterministic Builds



The screenshot shows a navigation sidebar on the left with a blue header bar containing the "Reproducible Builds" logo. The sidebar includes links for Home, Contribute, Documentation (which is currently selected), Tools, Who is involved?, News, Events, and Talks. The main content area to the right features a large heading "Definitions" and a sub-section "When is a build reproducible?" with its definition.

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.



@nielstanis



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’



0101
0101

Reproducible Build .NET6

00001a0:	0000 0000 0000 0000 0000 0000 0000 0000	00001a0:	0000 0000 0000 0000 0000 0000 0000 0000
00001b0:	0000 0000 0000 0000 0000 0000 0000 0000	00001b0:	0000 0000 0000 0000 0000 ea03 0000 0000 0000
00001c0:	0000 0000 0000 0000 0000 0000 0000 0000	00001c0:	0000 0000 0000 0000 0000 0000 0000 0000
00001d0:	0000 0000 0000 0000 0000 0000 0000 0000	00001d0:	0000 0000 0000 0000 0000 805a c352 00eb a154
00001e0:	0000 0000 0000 0000 0000 0000 0000 0000	00001e0:	0000 0000 0000 0000 0000 0000 0000 0000
00001f0:	0000 0000 0000 0000 0000 0000 0000 0000	00001f0:	0000 0000 0000 0000 0000 0000 0000 0000
0000200:	0000 0000 0000 0000 0000 0000 0000 0000	0000200:	0000 0000 0000 0000 0000 0000 0000 0000
0000210:	0000 0000 0000 0000 0000 0000 0000 0000	0000210:	0000 0000 0000 0000 0000 0000 0000 0000
0000220:	0000 0000 0000 0000 0000 0000 0000 0000	0000220:	0000 0000 0000 0000 0000 0000 0000 0000
0000230:	0000 0000 0000 0000 0000 0000 0000 0000	0000230:	0000 0000 0000 0000 0000 0000 0000 0000
0000240:	0000 0000 0000 0000 0000 0000 0000 0000	0000240:	0000 0000 0000 0000 0000 0000 0000 0000
0000250:	0000 0000 0000 0000 0000 0000 0000 0000	0000250:	0000 0000 0000 0000 0000 0000 0000 0000
0000260:	0000 0000 0000 0000 0000 0000 0000 0000	0000260:	0000 0000 0000 0000 0000 0000 0000 0000
0000270:	0000 0000 0000 0000 0000 0000 0000 0000	0000270:	0000 0000 0000 0000 0000 0000 0000 0000
0000280:	0000 0000 0000 0000 0000 0000 0000 0000	0000280:	0000 0000 0000 0000 0000 0000 0000 0000
0000290:	0000 0000 0000 0000 0000 0000 0000 0000	0000290:	0000 0000 0000 0000 0000 0000 0000 0000
00002a0:	0000 0000 0000 0000 0000 0000 0000 0000	00002a0:	0000 0000 0000 0000 0000 0000 0000 0000
00002b0:	0000 0000 0000 0000 0000 0000 0000 0000	00002b0:	0000 0000 0000 0000 0000 0000 0000 0000
00002c0:	0000 0000 0000 0000 0000 0000 0000 0000	00002c0:	0000 0000 0000 0000 0000 0000 0000 0000
00002d0:	0000 0000 0000 60e1 c552 6c5b 94d9 . . . ? . B . .	00002d0:	0000 0000 0000 b418 c252 0000 0000 . . . X . ? . e .
00002e0:	2093 f53f c3d8 4290 8392 f53f dd07 20b5 . . ? . B . .	00002e0:	2c65 19e2 5817 f63f 2c65 19e2 5817 f63f , e . X . ? . e .
00002f0:	8993 f53f 6d73 637a c292 f53f b81e 85eb . . ? mscz . .	00002f0:	bbb8 8d06 f016 f63f bbb8 8d06 f016 f63f . . . ? . .
0000300:	51b8 1d40 60e1 c552 0000 a7e1 c552 Q . @ . R . .	0000300:	0d00 0000 0000 ef18 c252 0000 0000
0000310:	3485 ce6b ec92 f53f df37 bef6 cc92 f53f 4 . k . ? . 7 .	0000310:	f018 c252 0000 0000 bbb8 8d06 f016 f63f . . . R . .
0000320:	6c43 c538 7f93 f53f 170e 8464 0193 f53f 1C . 8 . ? .	0000320:	2c65 19e2 5817 f63f bbb8 8d06 f016 f63f , e . X . ? .
0000330:	85eb 51b8 1e45 3040 a7e1 c552 0000 0000 . . Q . E00 . .	0000330:	bbb8 8d06 f016 f63f 0900 0000 0000 0000 ? . .
0000340:	dde1 c552 89d2 dee0 0b93 f53f df4f 8d97 . . R . . .	0000340:	f018 c252 0000 0000 2c19 c252 0000 0000 . . . R . . .
0000350:	6e92 f53f dd07 20b5 8993 f53f c3d8 4290 n . ? . .	0000350:	bbb8 8d06 f016 f63f 2c65 19e2 5817 f63f ? , e .
0000360:	8392 f53f 5c8f c2f5 285c 1140 dde1 c552 . . ? \ . (\ .	0000360:	bbb8 8d06 f016 f63f 2c65 19e2 5817 f63f ? , e .
0000370:	0000 0000 1fe2 c552 c3d8 4290 8392 f53f . . . R . B .	0000370:	0800 0000 0000 f118 c252 0000 0000
0000380:	c3d8 4290 8392 f53f c190 d5ad 9e93 f53f . . B . . . ? .	0000380:	6819 c252 0000 0000 2c65 19e2 5817 f63f h . R . . . e .
0000390:	c3d8 4290 8392 f53f 85eb 51b8 1e85 eb3f . . B . . . ? . Q	0000390:	2c65 19e2 5817 f63f bbb8 8d06 f016 f63f , e . X . ? .
00003a0:	1fe2 c552 0000 0000 55e2 c552 183e 22a6 . . R . . U . .	00003a0:	bbb8 8d06 f016 f63f 0c00 0000 0000 0000 ? . .
00003b0:	4492 f53f 183e 22a6 4492 f53f 87a2 409f D . ? . > . D . .	00003b0:	f218 c252 0000 0000 a419 c252 0000 0000 . . . R . . .
00003c0:	c893 f53f 183e 22a6 4492 f53f 14ae 47e1 . . ? . > . D . .	00003c0:	bbb8 8d06 f016 f63f bbb8 8d06 f016 f63f . . . ? . .
00003d0:	7a14 fe3f 55e2 c552 0000 0000 8ce2 c552 z . ? U . R . .	00003d0:	bbb8 8d06 f016 f63f bbb8 8d06 f016 f63f . . . ? . .

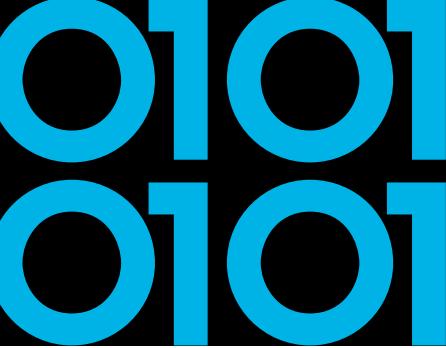




Reproducible/Deterministic Builds

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





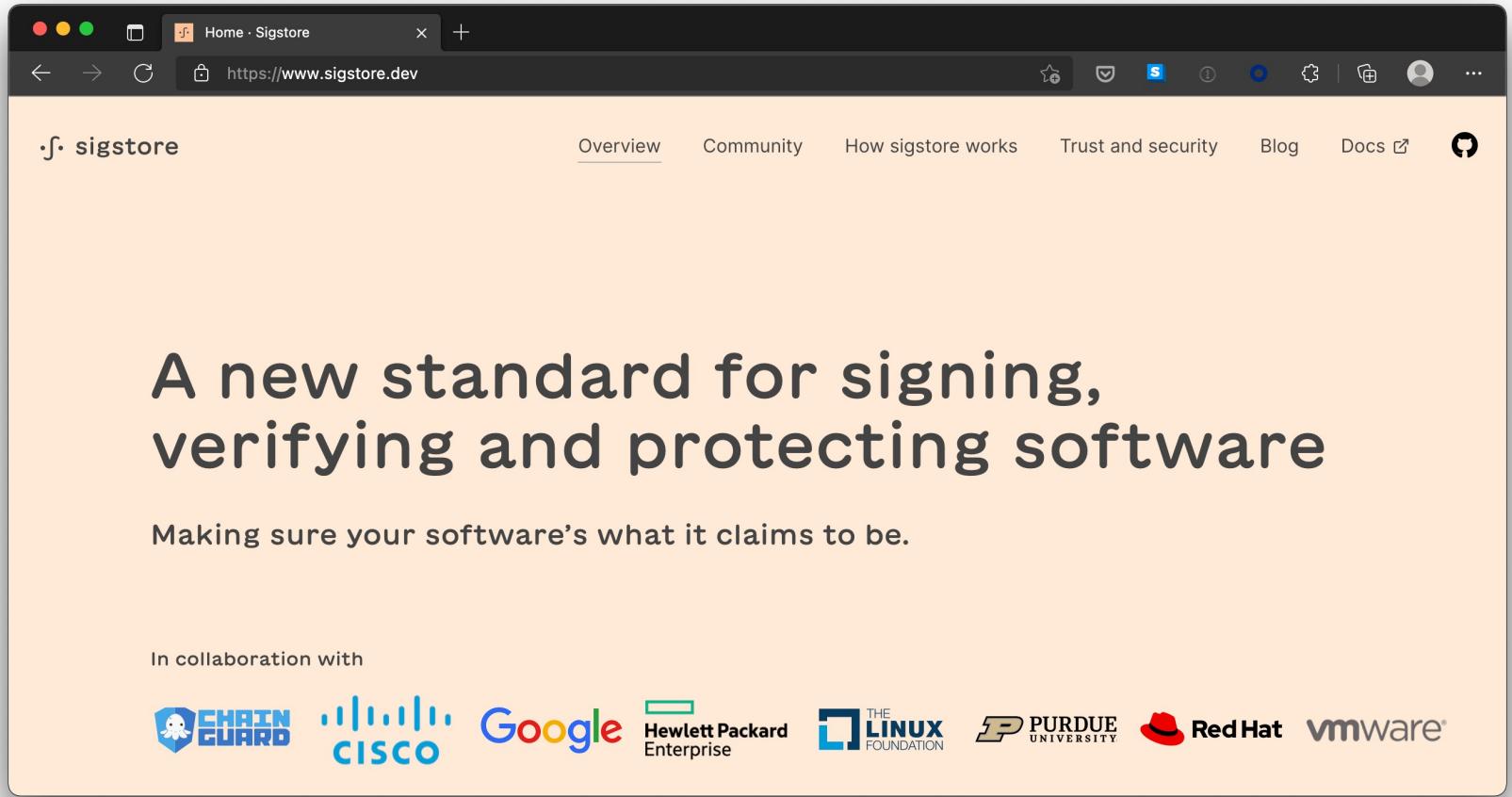
Reproducible Build Validation

- 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`



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's 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.

Home · Sigstore

https://www.sigstore.dev

sigstore

Overview Community How sigstore works Trust and security Blog Docs

A new standard for signing,
verifying and protecting software

Making sure your software's what it claims to be.

In collaboration with

CHAIN GUARD CISCO Google Hewlett Packard Enterprise THE LINUX FOUNDATION PURDUE UNIVERSITY Red Hat vmware



@nielstanis

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 section about the standardized approach and another about building for future integrations. To the right, a flowchart illustrates the process:

```
graph TD; Developers["DEVELOPERS, MAINTAINERS, MONITORS"] --> Sign["SIGN AND PUBLISH ARTIFACTS"]; Developers --> Publish["PUBLISH SIGNING CERTIFICATES"]; Developers --> Monitor["MONITOR LOGS"]; Sign --> Fulcio["FULCIO CERTIFICATE AUTHORITY"]; Publish --> SignatureLog["SIGNATURE TRANSPARENCY LOG"]; Monitor --> KeyLog["KEY TRANSPARENCY LOG"]; Fulcio --- TRUST_ROOT["TRUST ROOT"]; SignatureLog --- TRUST_ROOT; KeyLog --- TRUST_ROOT;
```

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.



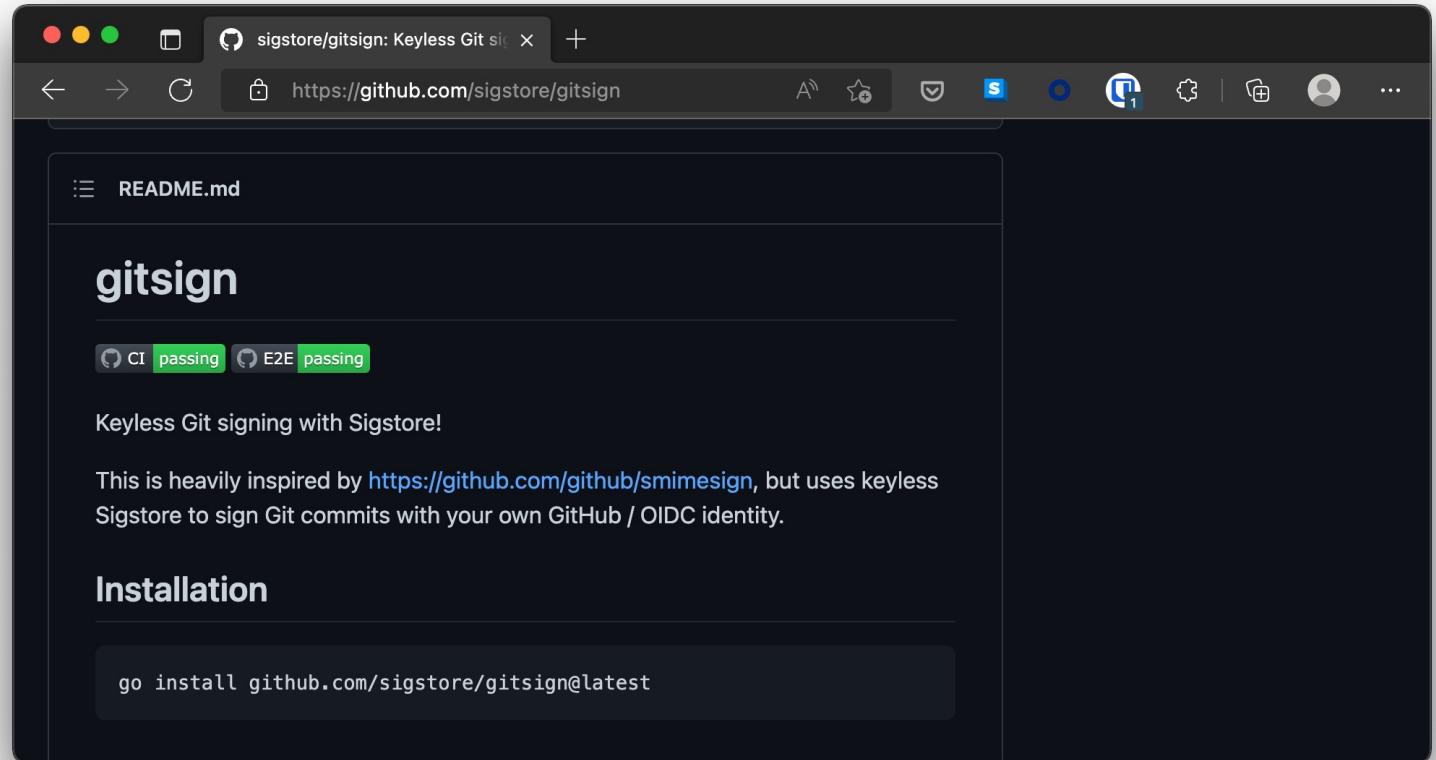
Signing artifacts

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



0101
0101

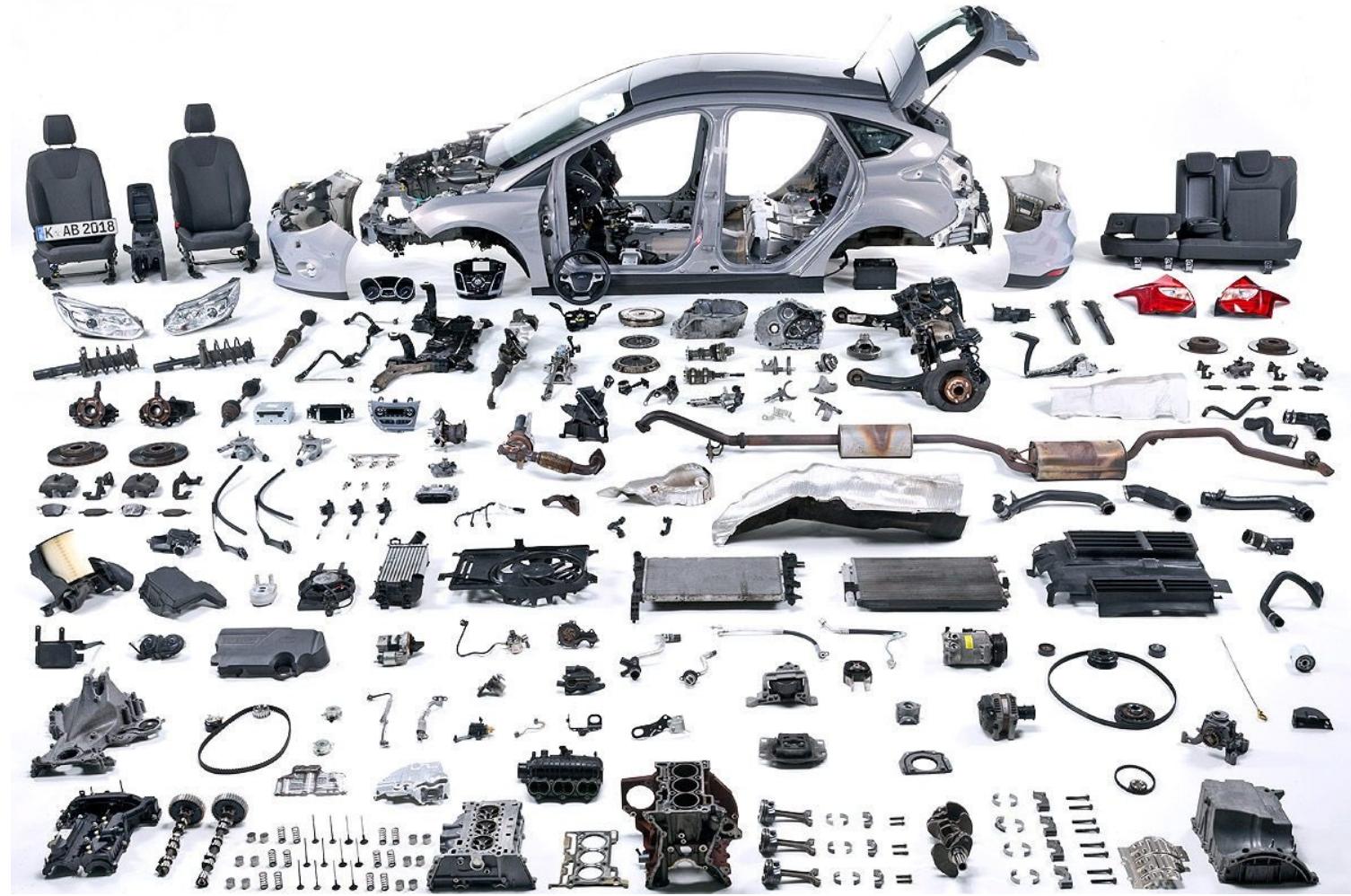
Git Commit Signing Sigstore GitSign



@nielstanis

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



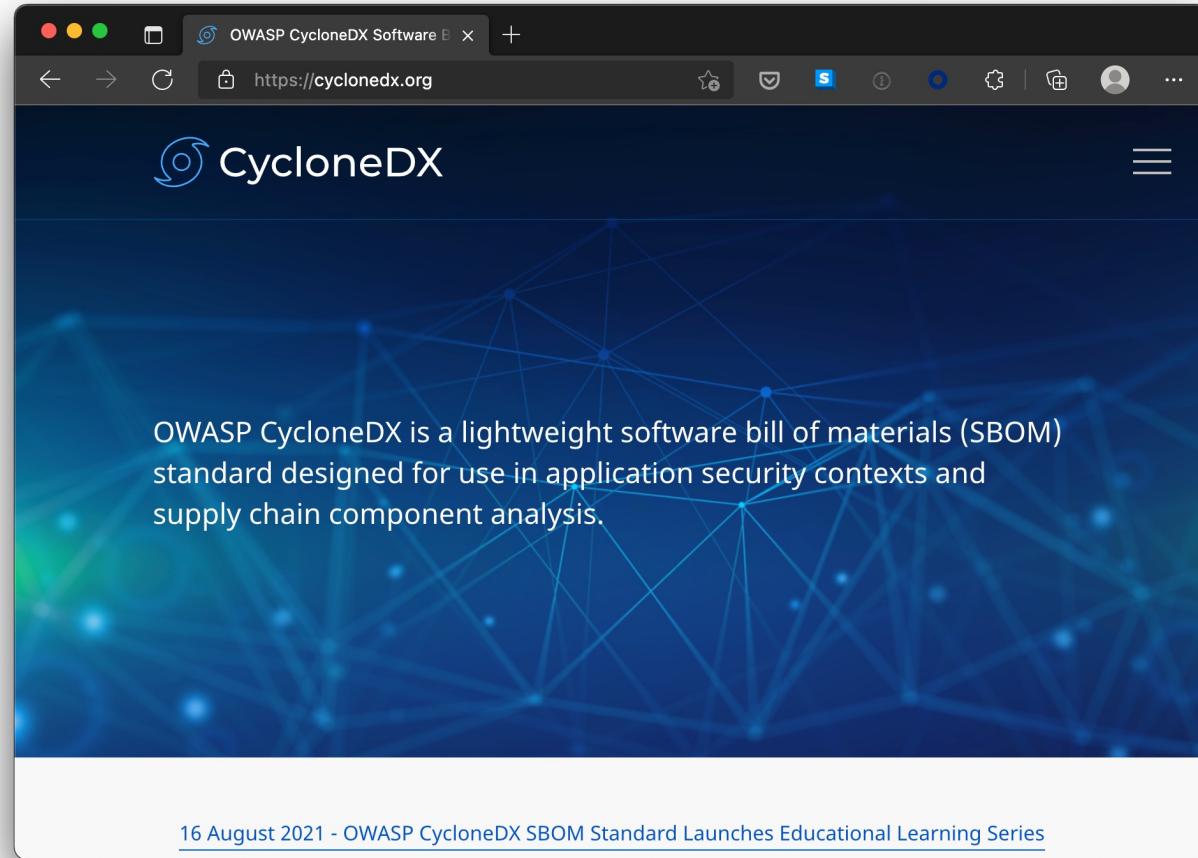
Software Bill of Materials (SBOM)

- 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?



0101
0101

Software Bill of Materials (SBOM)



@nielstanis

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". The author is listed as JUSTIN CORMACK, dated Apr 7 2022. The post content discusses Docker's first step in making container images more visible, specifically mentioning the experimental docker sbom CLI command. To the right of the post, there are sections for "Post Tags" and "Categories", each with a list of related topics.

Join us for DockerCon on May 9-10th. Preview the agenda and register today.

docker Products Developers Pricing Blog About Us Partners 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

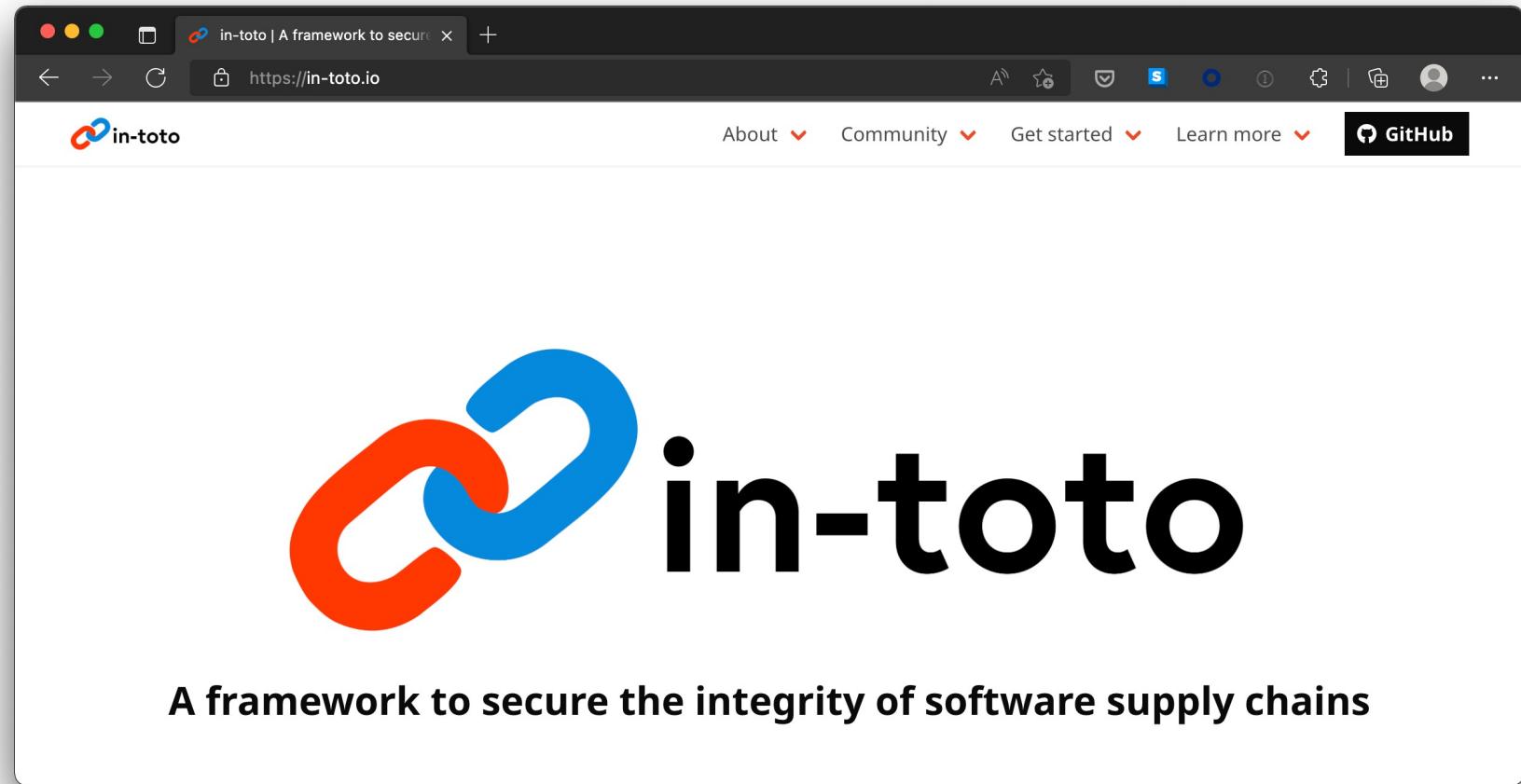
- Community
- Company
- Engineering
- Newsletters
- Products



@nielstanis

In-toto

0101
0101



@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



0101
0101

In-Toto - Demo

In-Toto - Demo - Terminology



Niels Tanis

- **Functionaries** that are identified by public key our supply chain.
Niels (Project-Owner), Aimee (Developer) and Noud (Packager)
- **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

+17

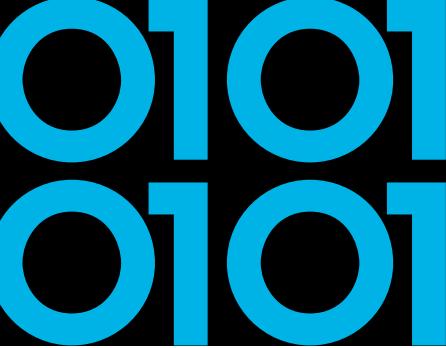
0101
0101

MyAwesomeWebApp Demo

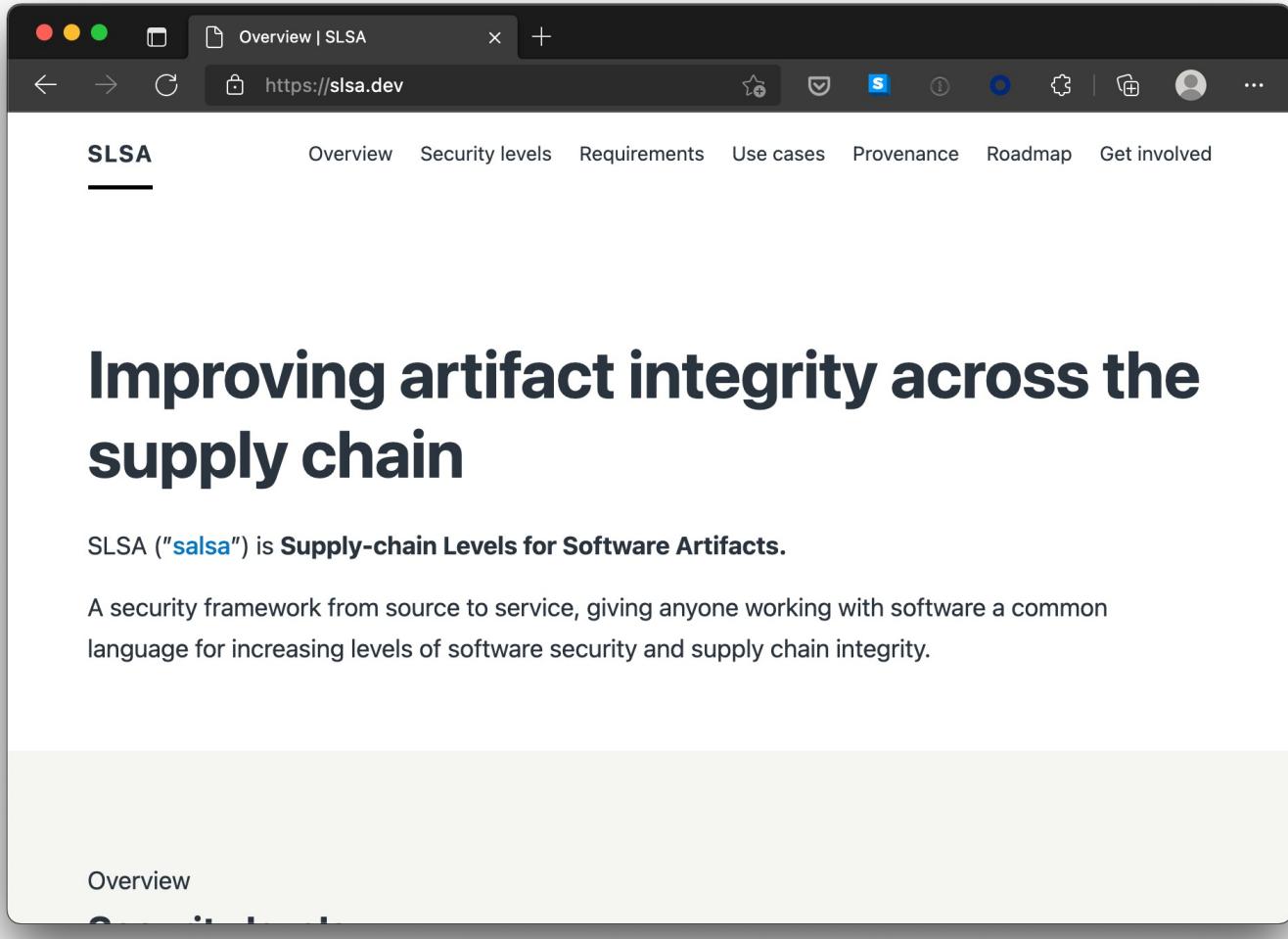
- GitHub Actions
- In-toto
- Sigstore Cosign
- Docker SBOM with Syft



@nielstanis



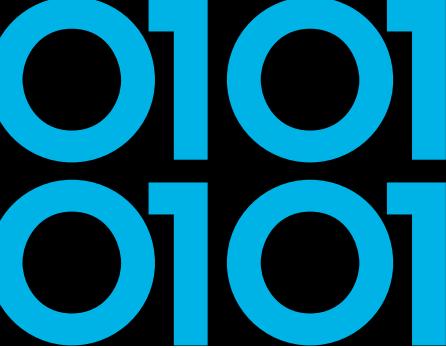
Google SLSA



The screenshot shows a web browser window displaying the official SLSA website at <https://slsa.dev>. The page has a dark header with a light gray navigation bar below it. The main content area features a large heading: "Improving artifact integrity across the supply chain". Below the heading, a definition of SLSA is provided: "SLSA ("salsa") is Supply-chain Levels for Software Artifacts." A descriptive paragraph follows: "A security framework from source to service, giving anyone working with software a common language for increasing levels of software security and supply chain integrity." At the bottom of the page, there is a navigation menu with links to "Overview", "Security levels", "Requirements", "Use cases", "Provenance", "Roadmap", and "Get involved".



@nielstanis



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





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



@nielstanis

0101
0101

SUSE SLSA Level 4

The screenshot shows a web browser window with the title bar "SLSA: Securing the Software Supply Chain". The address bar displays the URL <https://documentation.suse.com/sbp/server-linu...>. The page content is a white card with a dark header. The header includes the SUSE logo and a breadcrumb navigation: "SLSA: Securing the Software Supply Chain". Below the header, the text "All SUSE Products" is displayed in a green box. The main title "SLSA: Securing the Software Supply Chain" is prominently shown in large green text. Below the title, there is an "Abstract" section with a link "#", a "REPORT DOCUMENTATION BUG" button, and a paragraph of text explaining the document's purpose. A "Disclaimer" section follows, containing detailed legal disclaimers.

All SUSE Products

SLSA: Securing the Software Supply Chain

Abstract # REPORT DOCUMENTATION BUG

This document details how SUSE, as a long-time champion and expert of software supply chain security, prepares for SLSA L4 compliance.

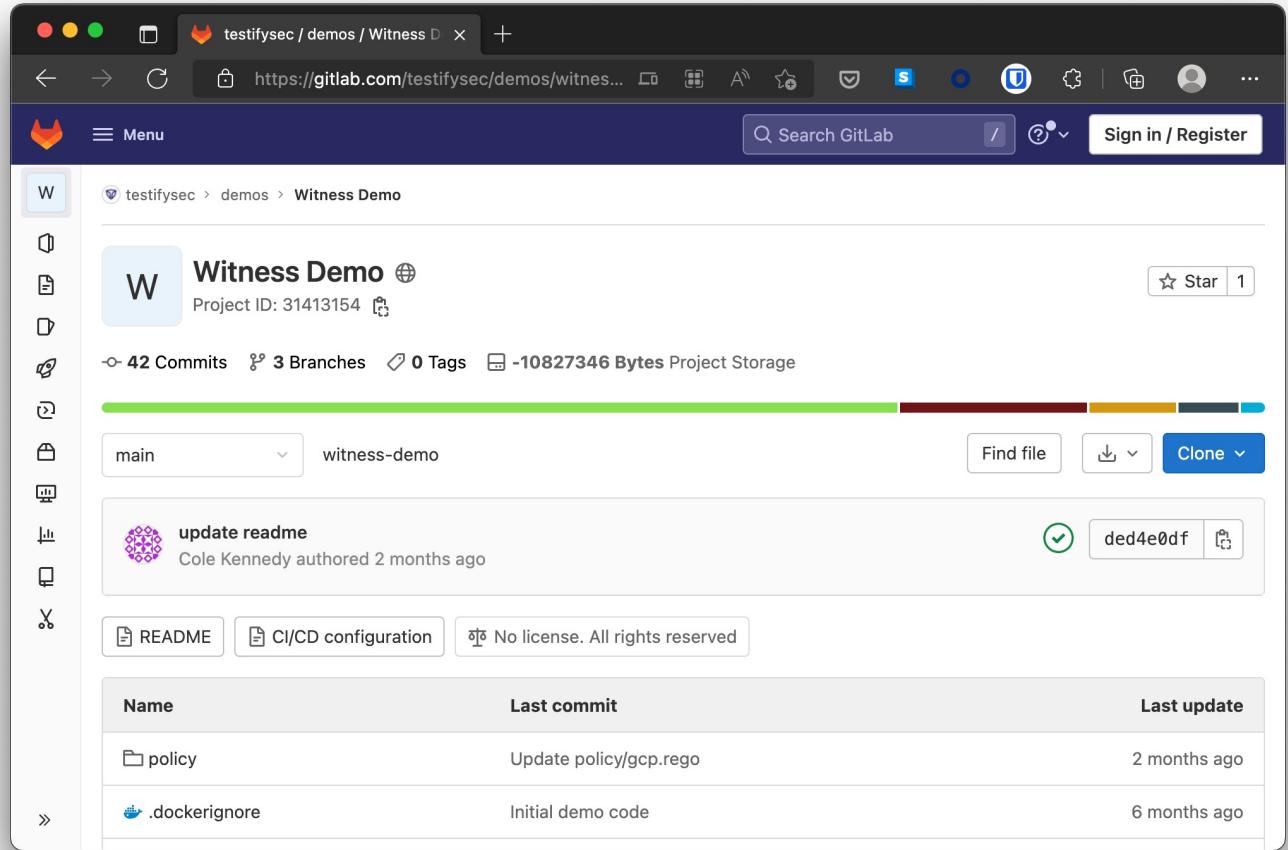
Disclaimer: This document is part of the SUSE Best Practices series. All documents published in this series were contributed voluntarily by SUSE employees and by third parties. If not stated otherwise inside the document, the articles are intended only to be one example of how a particular action could be taken. Also, SUSE cannot verify either that the actions described in the articles do what they claim to do or that they do not have unintended consequences. All information found in this document has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Therefore, we need to specifically state that neither SUSE LLC, its affiliates, the authors, nor the translators may be held liable for possible or



@nielstanis

0101
0101

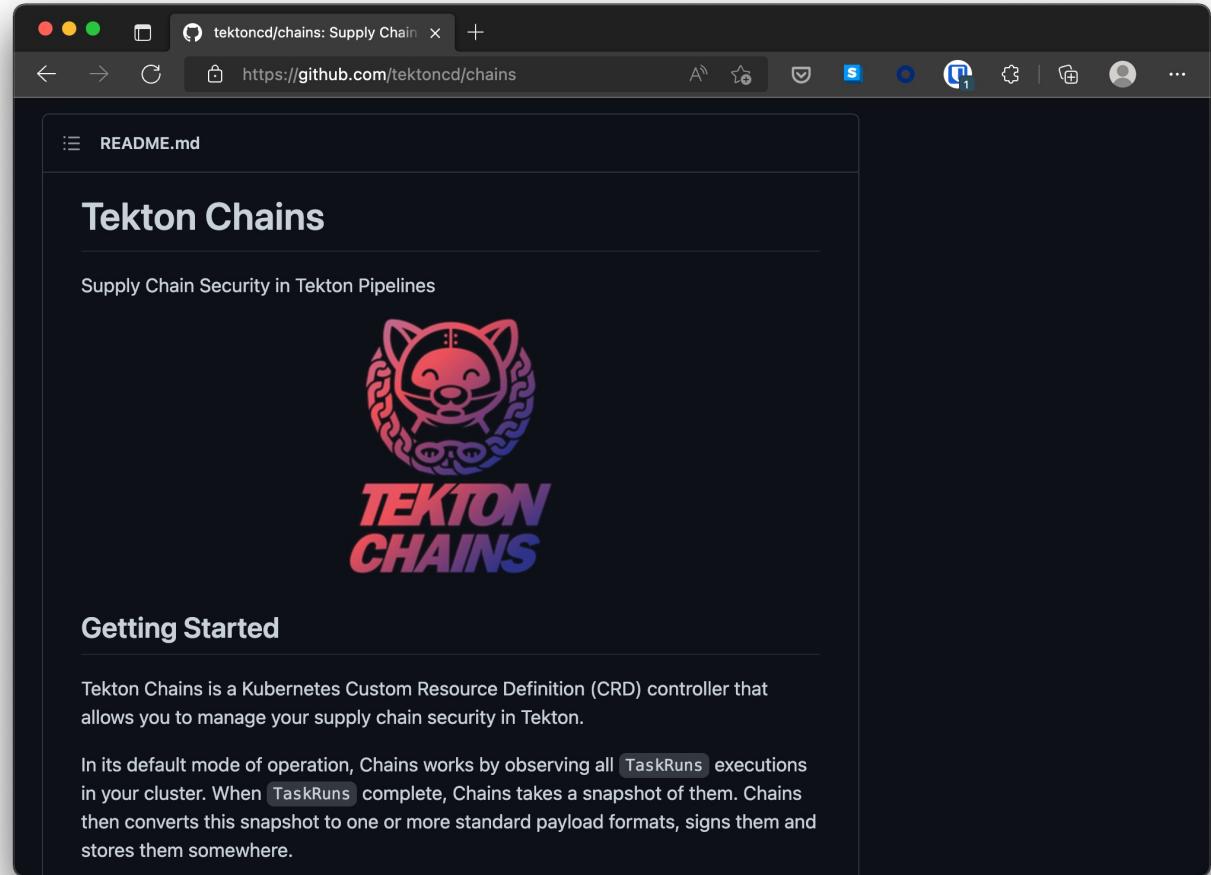
Witness & GitLab Attestator



@nielstanis

0101
0101

Open Shift - Tekton - Tekton Chains



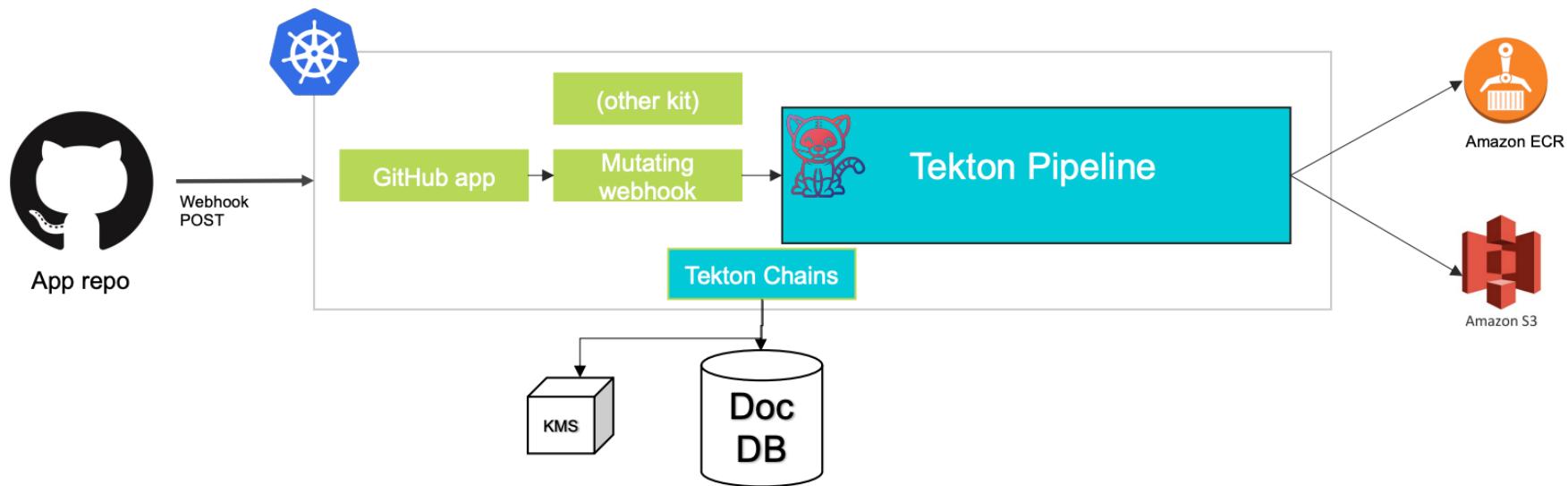
@nielstanis



SolarWinds Project Trebuchet



Pipeline With Attestations

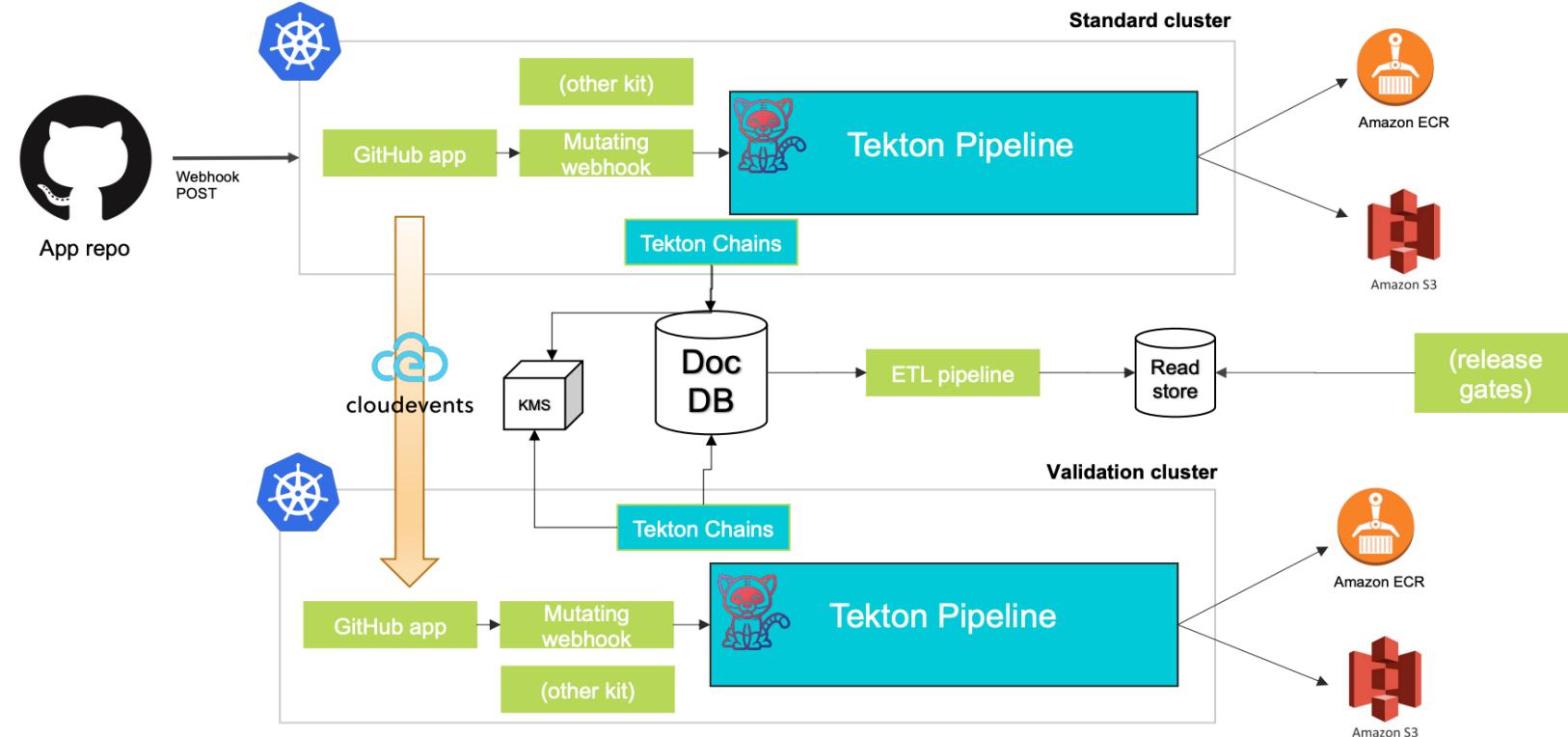


@nielstanis

SolarWinds Project Trebuchet



Reading Results



IBM OpenShift

0101
0101

A screenshot of a web browser displaying a Medium article. The article title is "A Zero Trust Approach for Securing the Supply Chain of Microservices Packaged as Container Images" by Gerry Kovan. The article summary states: "Securing the software supply chain has become a top priority for both open source as well as closed source projects. There are many sources in a software supply chain such operating systems and operating system packages/binaries, programming language frameworks and libraries, shell". The Medium interface shows 28 followers, a "Follow" button, and social sharing icons for Twitter, Facebook, LinkedIn, and Email.

A Zero Trust Approach for Securing the Supply Chain of Microservices Packaged as Container Images

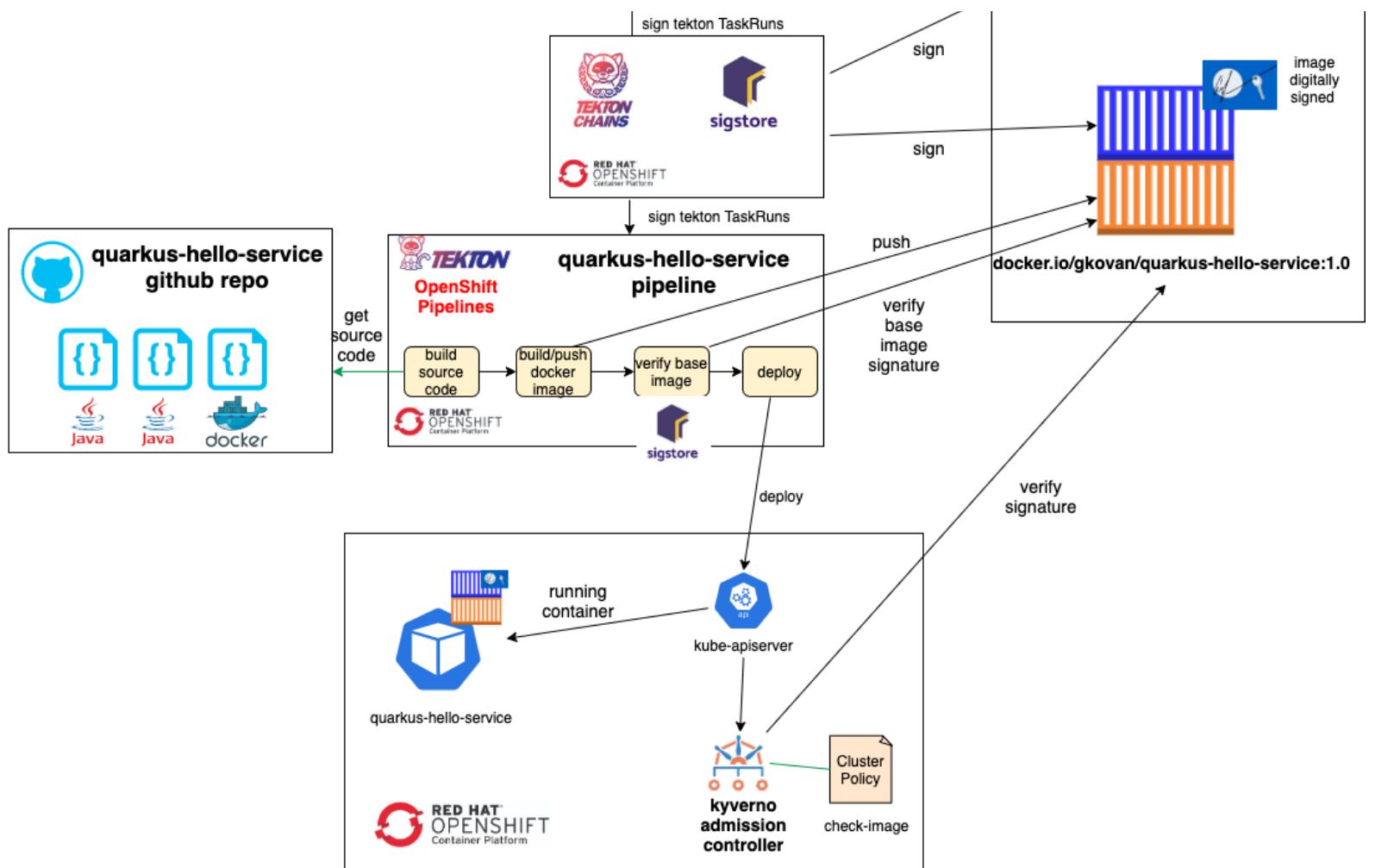
Gerry Kovan · Sep 6 · 6 min read

Securing the software supply chain has become a top priority for both open source as well as closed source projects. There are many sources in a software supply chain such operating systems and operating system packages/binaries, programming language frameworks and libraries, shell



@nielstanis

IBM OpenShift





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



@nielstanis

Azure Policy

0101
0101

The screenshot shows a Microsoft Docs page titled "Tutorial: Build policies to enforce compliance". The page is part of the "Azure Policy documentation" section. The main content area features a large title "Tutorial: Create and manage policies to enforce compliance" with a subtitle "Article • 04/03/2022 • 17 minutes to read • 4 contributors". Below the title is a "In this article" section listing various sub-topics. On the left side, there is a sidebar with a navigation menu for "Azure Policy documentation" including "Overview", "Quickstarts", "Tutorials" (which is expanded to show "Create and manage Azure Policy" as the current section), and several other sub-options like "Create a custom policy definition" and "Manage tag governance". At the bottom of the sidebar, there is a "Download PDF" button.



@nielstanis

Chainguard Enforce

0101
0101

A screenshot of a web browser displaying an article from THE NEW STACK. The title of the article is "Chainguard Enforce: Software Supply Chain Security for K8s". The article is dated 28 Apr 2022 9:49am, by Steven J. Vaughan-Nichols. Below the title is a large black and white photograph of a metal cage with a padlock and chains. A small blue icon with a checkmark is in the bottom left corner.

A screenshot of a web browser displaying the Chainguard Enforce product landing page. The page has a purple header with the Chainguard logo and navigation links for Solutions, Resources, and Company. The main heading is "Chainguard Enforce". To the right, there is a text block: "Enforce enables you to build, manage, ensure continuous compliance, and enforce policies that protect your organization from supply chain threats." Below this are two buttons: "Request a demo" and "Preview". At the bottom, there is a call-to-action: "Security without sacrificing".



@nielstanis

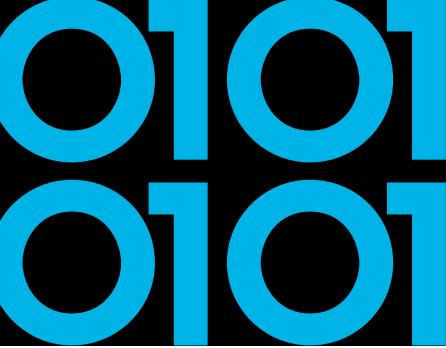
0101
0101

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.



@nielstanis



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/futuretech2022>

ntanis at veracode.com

@nielstanis on Twitter

