

NDC { Sydney }

Securing your .NET application  
software supply-chain

Niels Tanis

VERACODE





# Who am I?

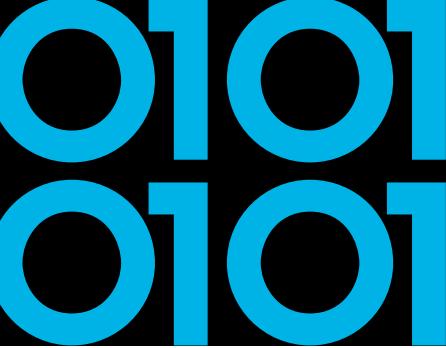
- Niels Tanis
- Principal Security Researcher @ Veracode
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - ISC<sup>2</sup> CSSLP
  - Research on static analysis for .NET apps



# Securing your .NET application software supply-chain

0101  
0101





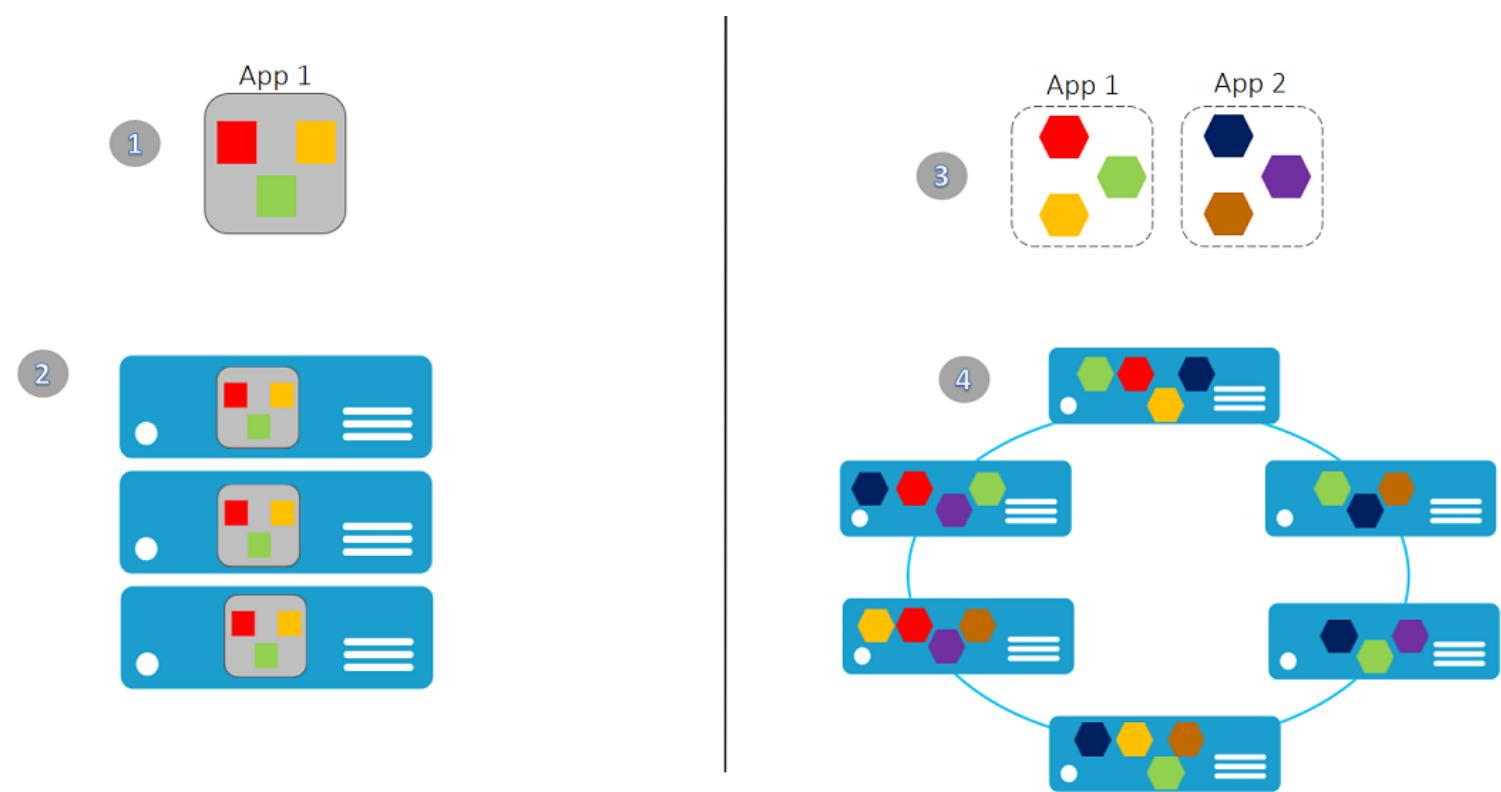
# Agenda

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



# Evolution in Software Architecture

- Monolith
- Microservices
- Serverless
- Cloud-Native



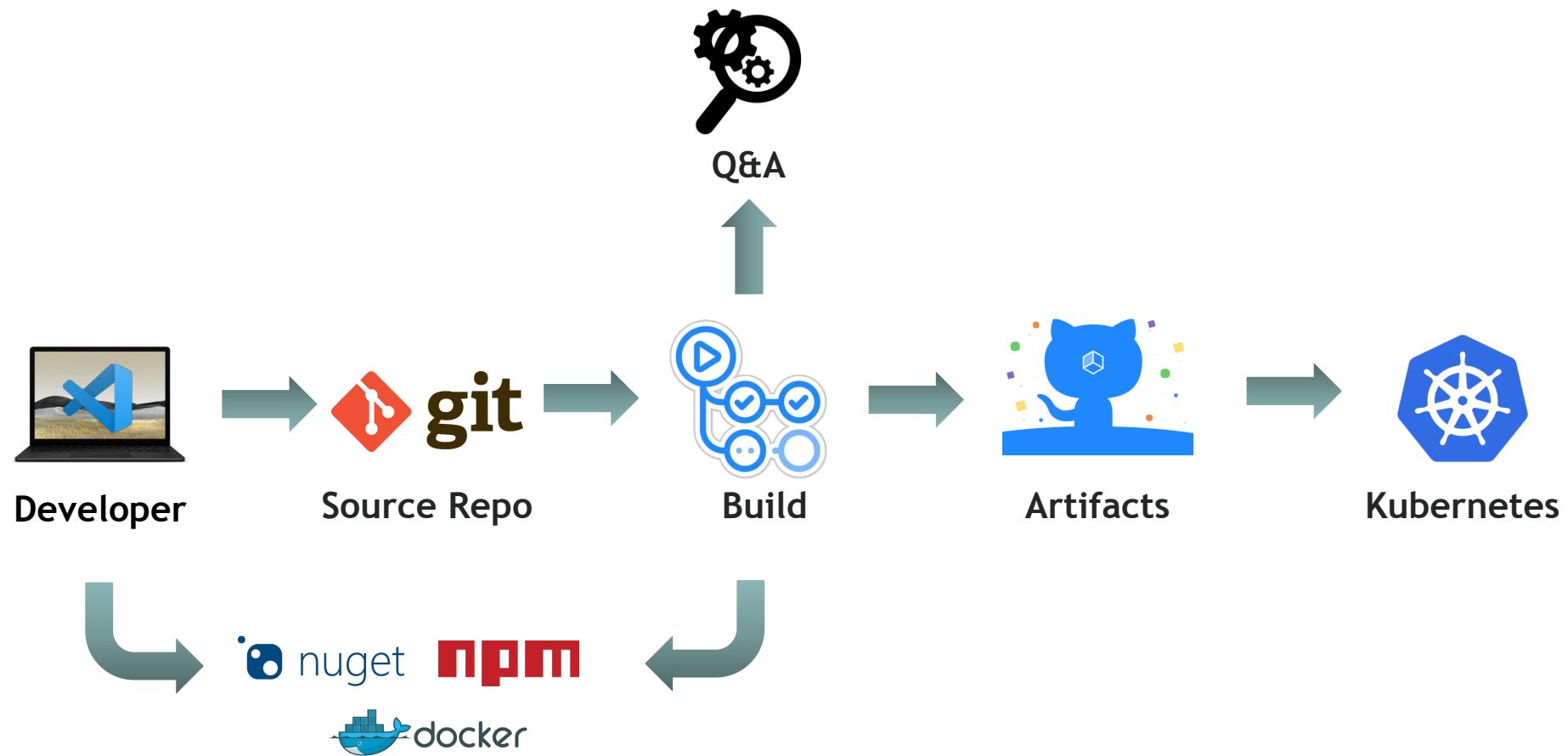
0101  
0101

# What is a Supply Chain?



# Software Supply Chain

0101  
0101



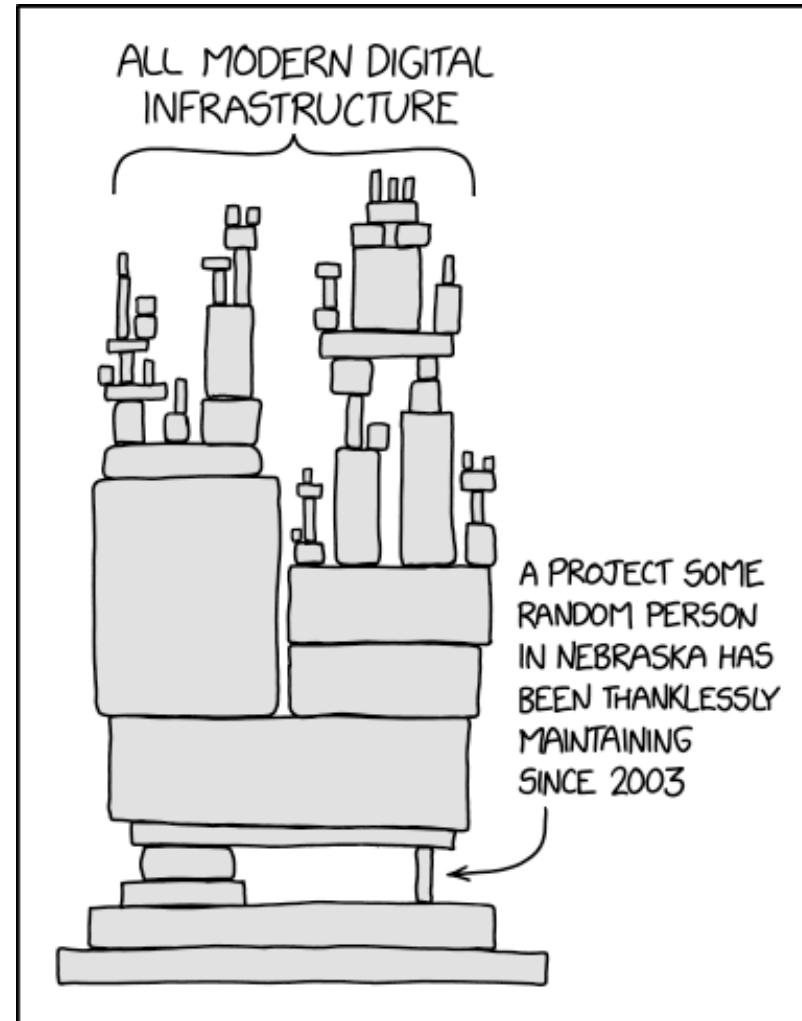
0101  
0101

# 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.

0101  
0101

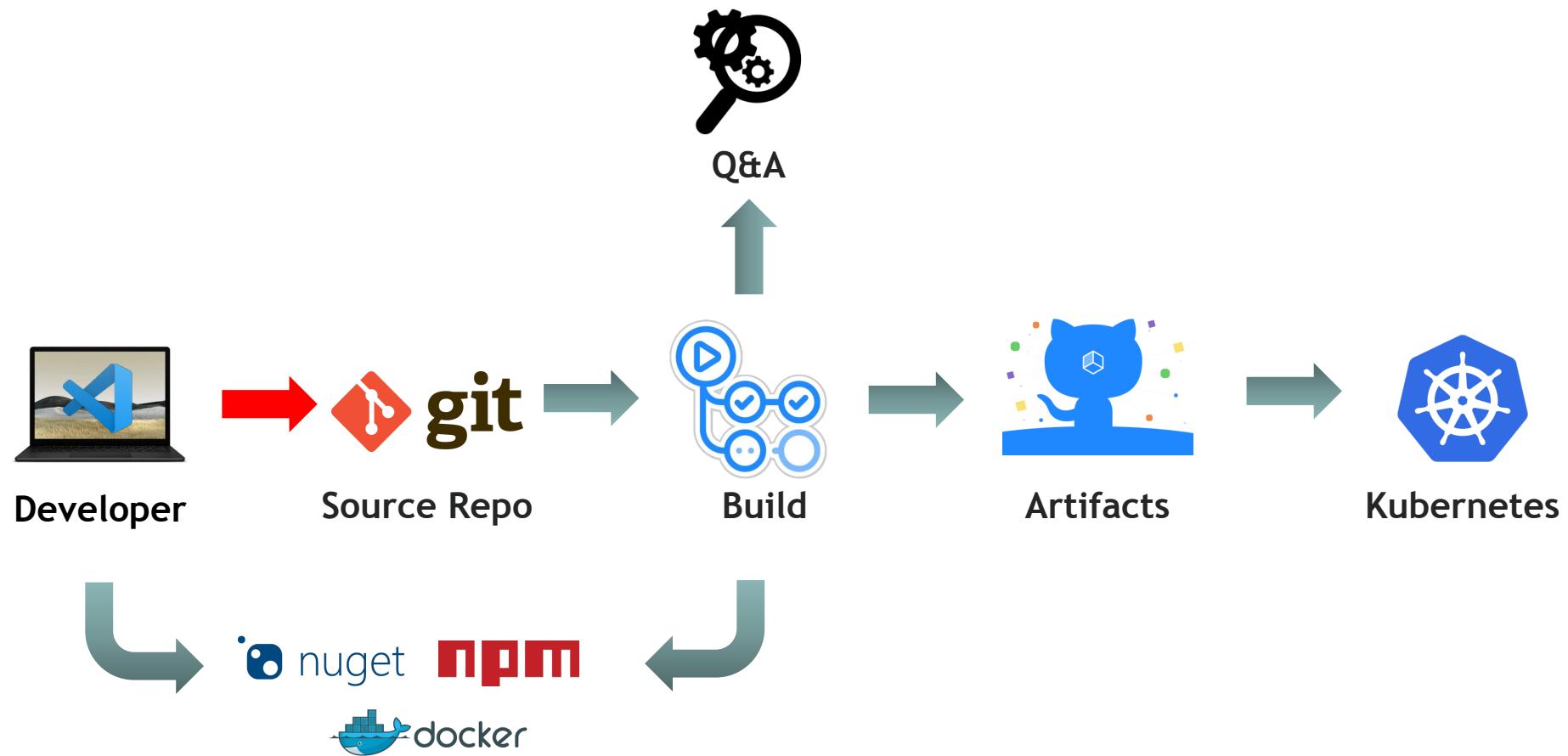
# XKDC - Dependency



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

# Software Supply Chain

0101  
0101



# GitHub account

0101  
0101

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 header includes the ZDNet logo, a search bar, and navigation links for AFRICA, UK, ITALY, SPAIN, MORE, EDITION: EU, NEWSLETTERS, ALL WRITERS, and a user profile icon.

A banner at the top of the article reads: "MUST READ: Chinese hackers could steal data now and crack it with quantum computers later, warns report".

## 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

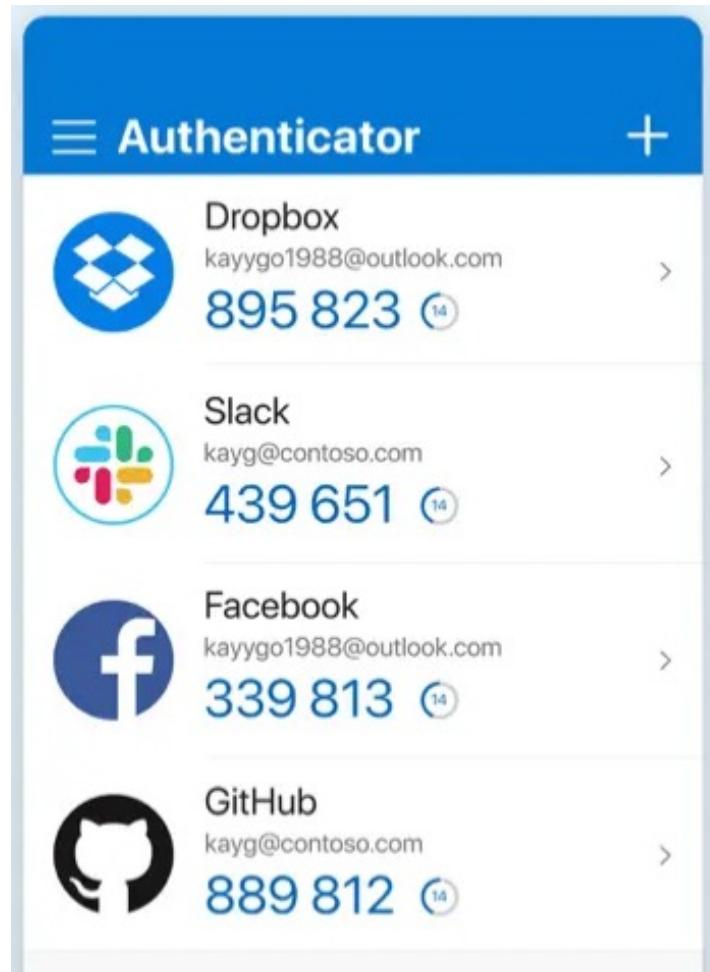
The main content of the article features a screenshot of the Canonical Ltd GitHub repository page. The repository name is "CAN\_GOT\_HAXXED\_10". The page shows basic repository statistics: 0 stars, 0 forks, and 10 commits, last updated 20 hours ago. It also displays the "Top languages" (Python, Go, Shell, C++) and "Most used topics" (ubuntu, canonical, ceon, vmmesites).

**RELATED**

The bottom of the article includes a sidebar with a link to "Why everyone should have this cheap security tool" and a small image of a physical security device.

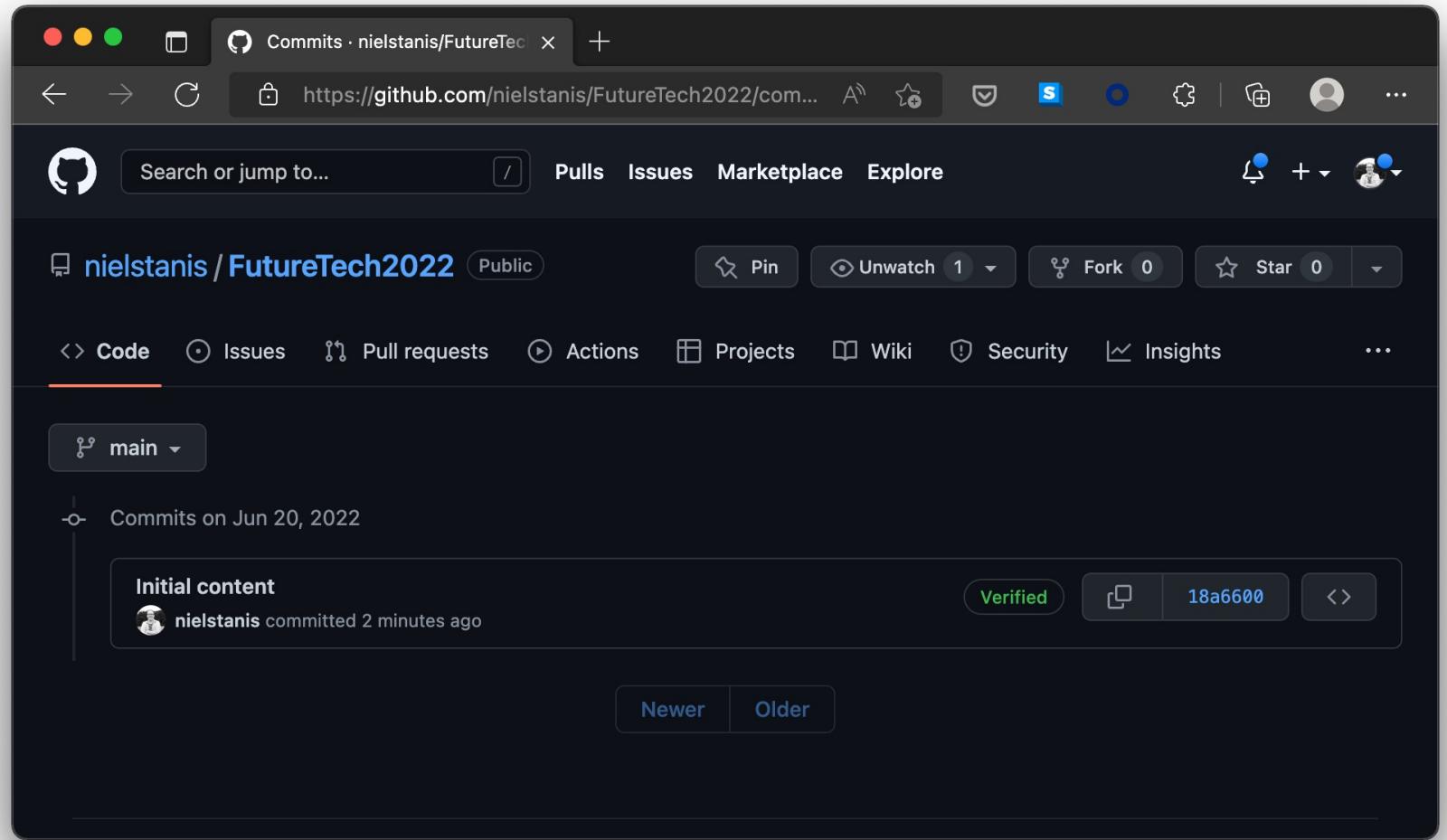
0101  
0101

# Use MFA on source-repository



# GIT Commit Signing

0101  
0101



# Hypocrite Commits

0101  
0101

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`

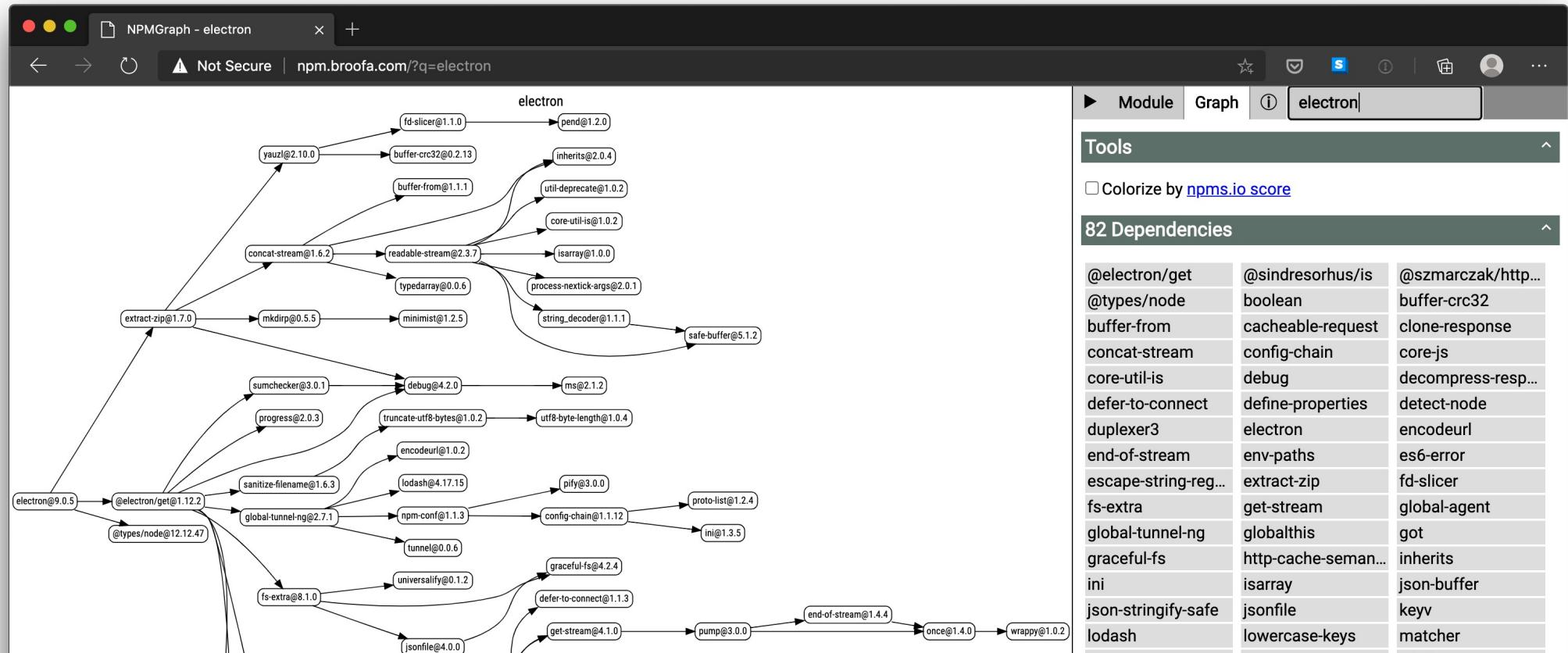
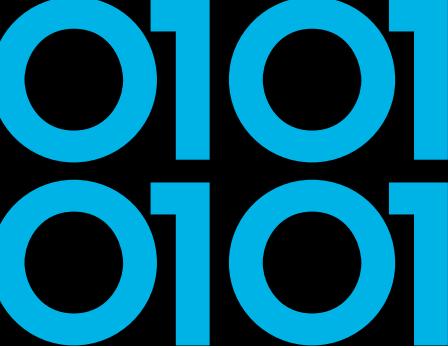
**Abstract:** Open source software (OSS) has thrived since the forming of Open Source Initiative in 1998. A prominent example is the Linux kernel, which has been used by numerous major software vendors and empowering billions of devices. The higher availability and lower costs of OSS boost its adoption, 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 analysis and defense mechanisms.

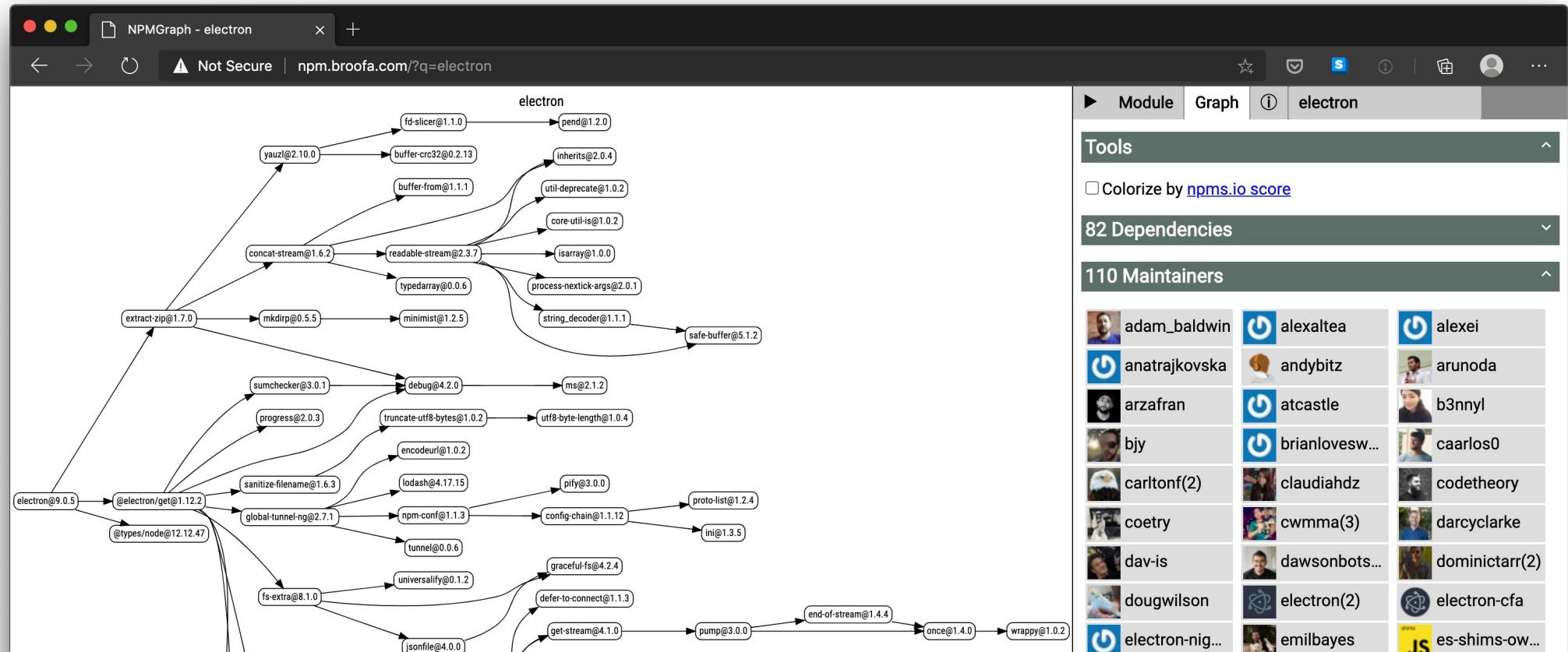
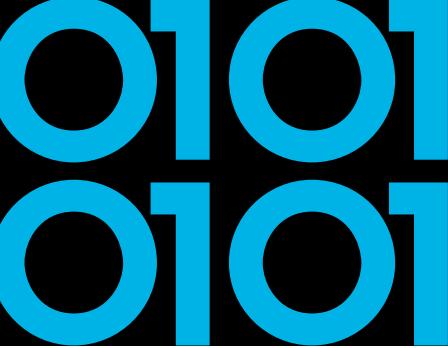
**Its openness also encourages contributors; OSS typically has thousands of independent programmers testing and fixing bugs of the software. Such an open and collaborative development not only allows higher flexibility, transparency, and quicker evolution, but is also believed to provide higher reliability 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

# Visual Studio Code



# Visual Studio Code



0101  
0101

# Visual Studio Code

CVE-2022-30129 - Security Update

https://msrc.microsoft.com/update-guide/en-US/vuln...

Microsoft MSRC | Security Updates Acknowledgements Developer

MSRC > Customer Guidance > Security Update Guide > Vulnerabilities > CVE 2022 30129

## Visual Studio Code Remote Code Execution Vulnerability

CVE-2022-30129

On this page ▾

### Security Vulnerability

Released: May 10, 2022

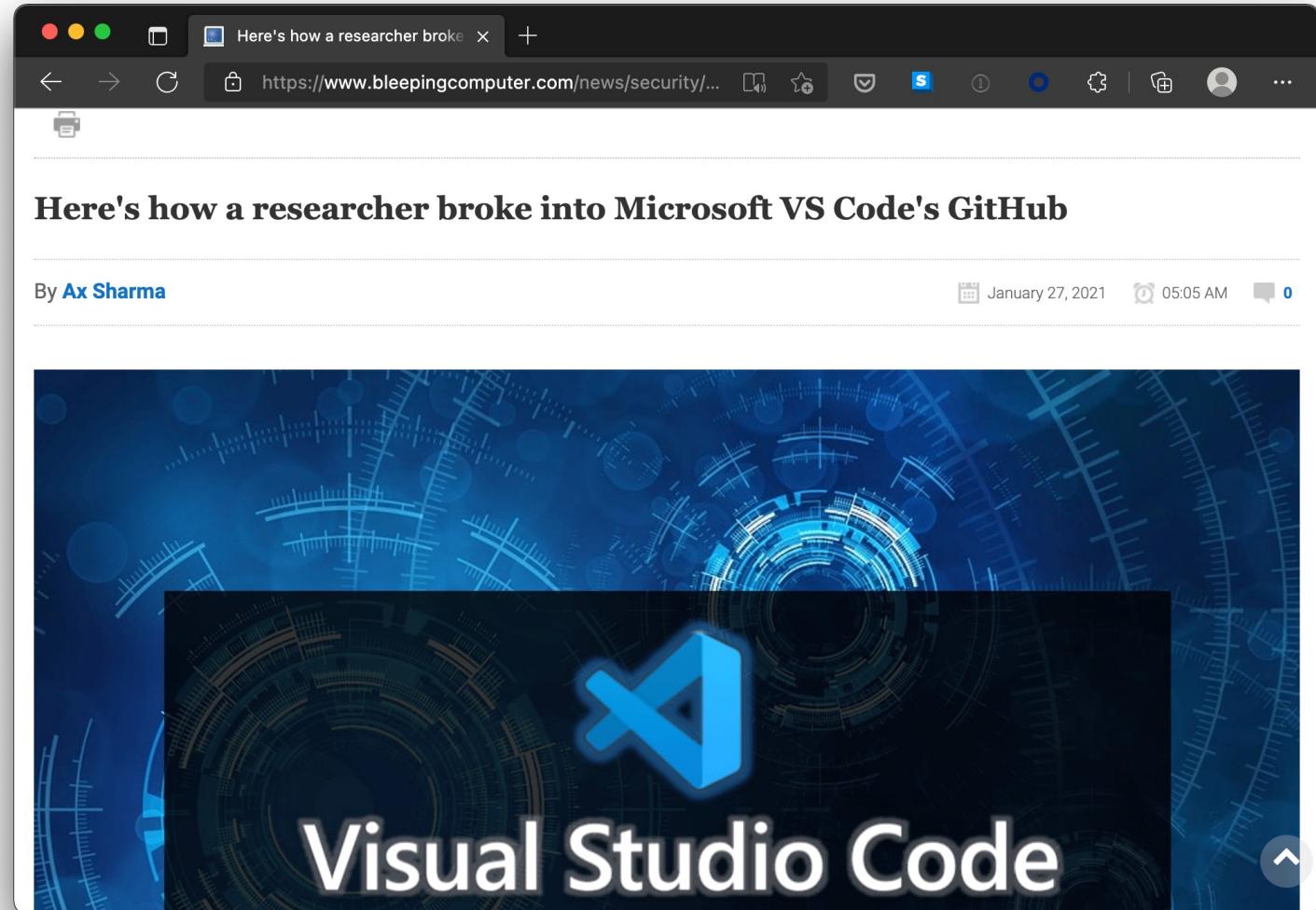
Assigning CNA: Microsoft

[CVE-2022-30129](#)

CVSS:3.1 8.8 / 7.7

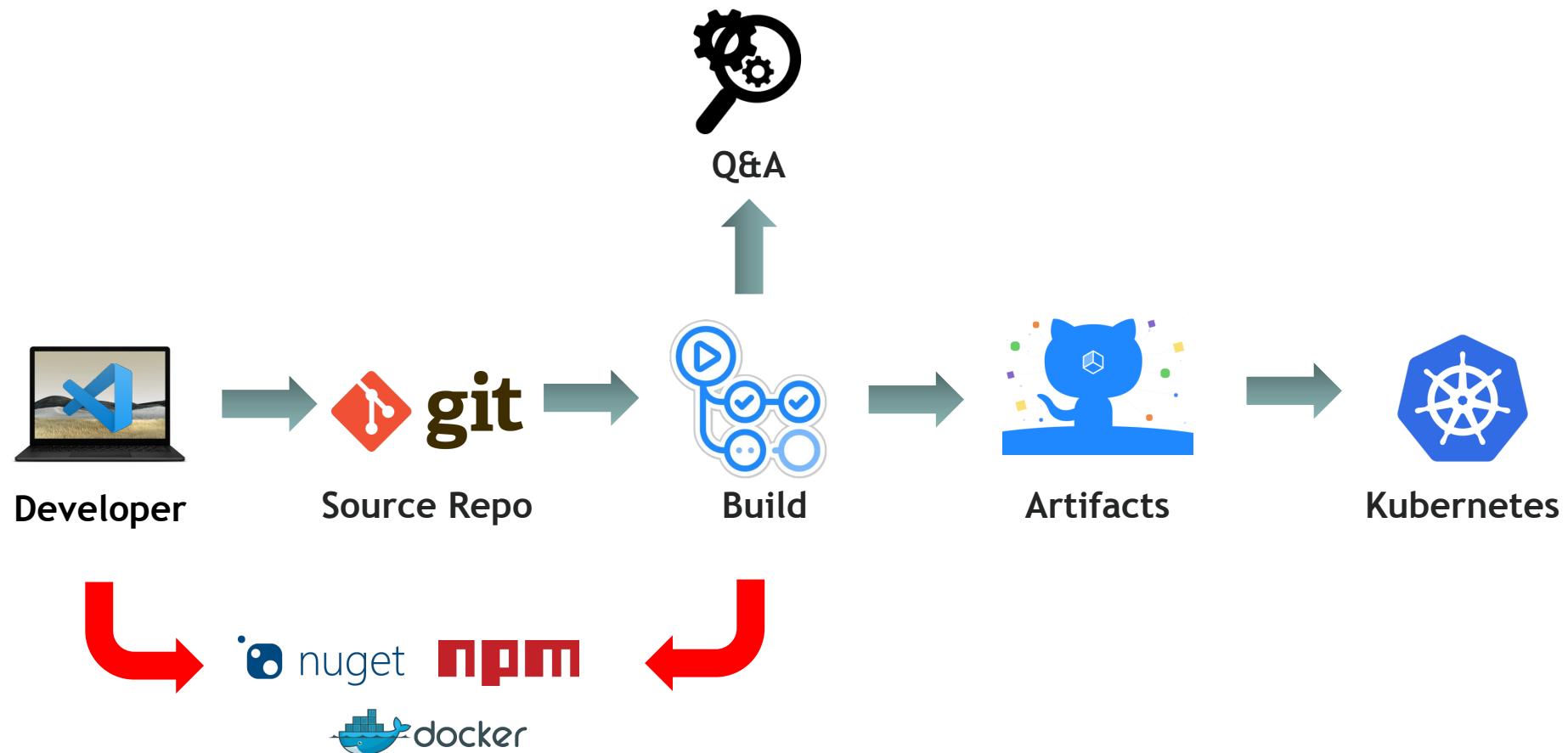
0101  
0101

# Visual Studio Code



# 3rd Party Libraries

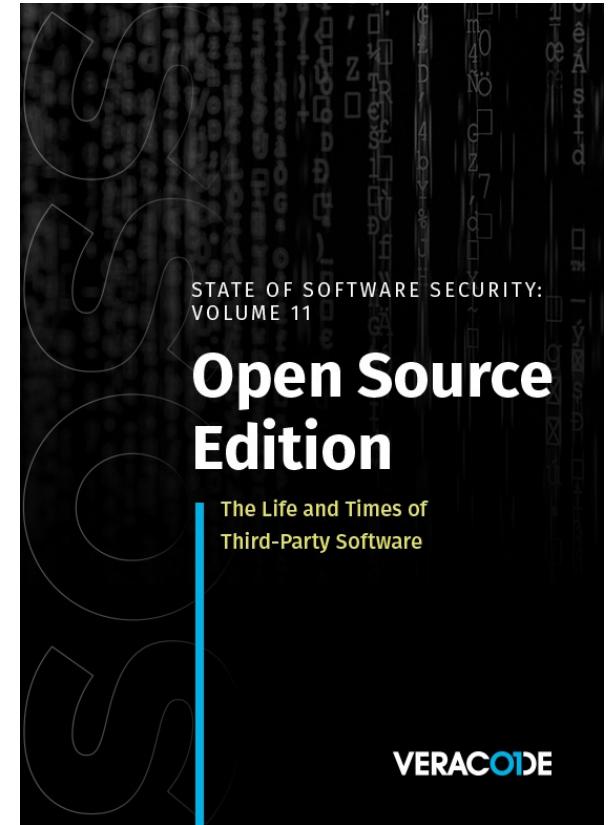
0101  
0101



# 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

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 Remote Code Execution vulnerability in .NET 6.0, .NET 5.0, and .NET Core 3.1. It describes a stack buffer overrun in the .NET Double Parse routine. 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).

# 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



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](#).

0101  
0101

# Vulnerabilities in libraries

The image shows two side-by-side screenshots of a Mac OS X browser window displaying GitHub blog posts. Both posts are under the 'Open Source' and 'Security' categories.

**Post 1: GitHub's commitment to npm ecosystem security**

**Post 2: Enrolling all npm publishers in enhanced login verification and next steps for two-factor authentication enforcement**

Both posts include a large blue cartoon illustration of a character working on a yellow puzzle piece, symbolizing security and assembly. The second post also features the red GitHub logo and the red 'npm' logo.

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. Below the title is a subtitle "The Story of a Novel Supply Chain Attack". The author's profile picture and name are on the left, along with the publication date "Feb 9 · 11 min read". On the right are social sharing icons for Twitter, Facebook, LinkedIn, and others. Below the article title is a large image of colorful shipping containers stacked together against a blue sky.



# 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>



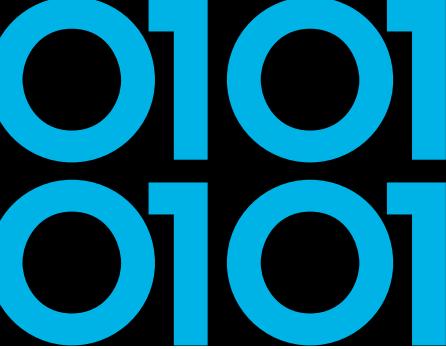
# Dependency Confusion - NuGet

- 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
- My talk 'Sandboxing .NET Assemblies' Tomorrow @ 3PM in Room 4
  
- Open Source Security Foundation - OpenSSF
- Security Scorecards - Security health metrics for Open Source

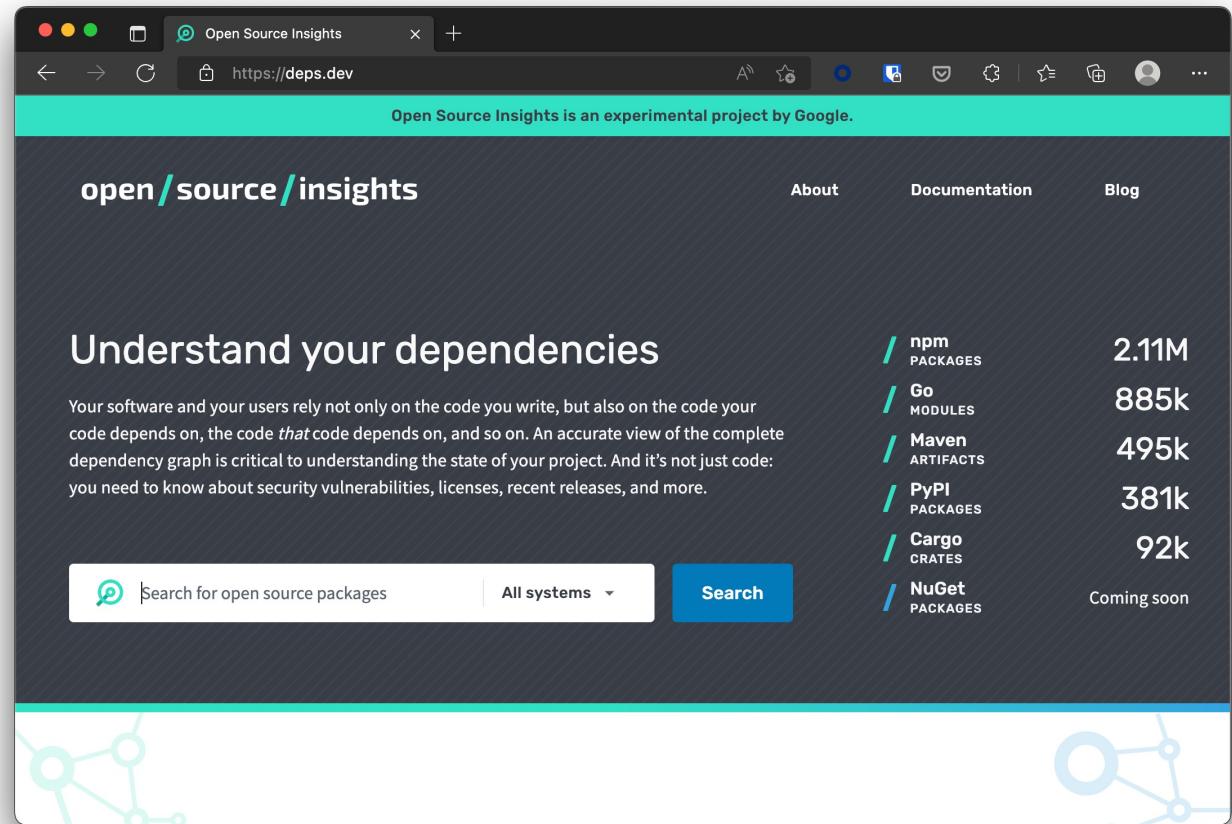


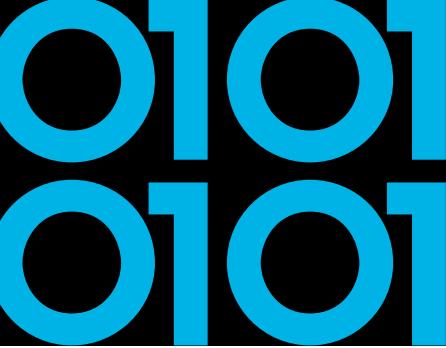
# Security Scorecards - OpenSSF

The screenshot shows the GitHub README page for the `ossf/scorecard` repository. The page has a dark theme. At the top, there's a navigation bar with icons for back, forward, search, and other GitHub features. The URL `https://github.com/ossf/scorecard` is visible. Below the navigation is a header with the file name `README.md`. The main content starts with a section titled **Security Scorecards**. Below this, there are several status badges: `openssf scorecard 8.6`, `openssf best practices in progress 64%`, `build passing`, `CodeQL passing`, `reference`, `go report A+`, `codecov 45%`, `slack chat`, and `SLSA level 3`. To the right of the badges is a large, stylized cartoon character of a bird-like creature wearing a red vest and holding up a yellow sign with the number **10**. The page also contains sections for **Overview** and **Using Scorecards**, each with a list of links.

0101  
0101

# Deps.Dev by Google





# Deps.Dev by Google

## 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

# 6.6/10

Scorecard as of September 12, 2022.

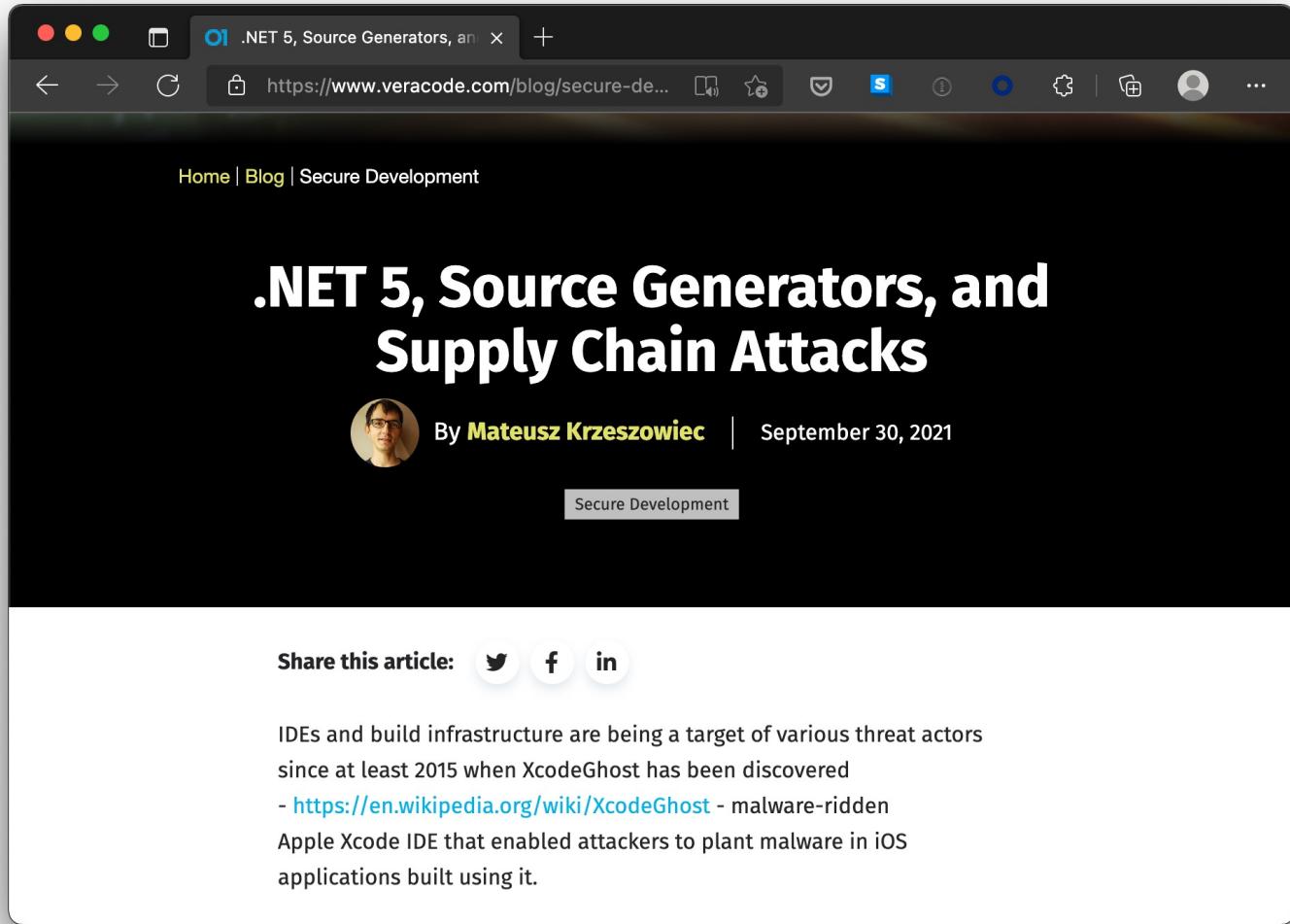
‣ Maintained	10/10
‣ Code-Review	8/10
‣ CII-Best-Practices	0/10
‣ Vulnerabilities	10/10
‣ Security-Policy	10/10
‣ Dangerous-Workflow	10/10

‣ Token-Permissions	0/10
‣ License	10/10
‣ Pinned-Dependencies	7/10
‣ Binary-Artifacts	10/10
‣ Fuzzing	0/10
‣ Dependency-Update-Tool	10/10
‣ Signed-Releases	0/10
‣ SAST	0/10
‣ Branch-Protection	8/10

Project metadata as of September 18, 2022.

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

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.



# 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 bar for 'Reproducible Builds'. The top bar is dark blue with white text. Below it is a light gray sidebar with a dark blue header containing the 'Reproducible Builds' logo and name. The sidebar lists several menu items: Home, Contribute, Documentation (which is bolded), Tools, Who is involved?, News, Events, and Talks.

## Definitions

### When is a build reproducible?

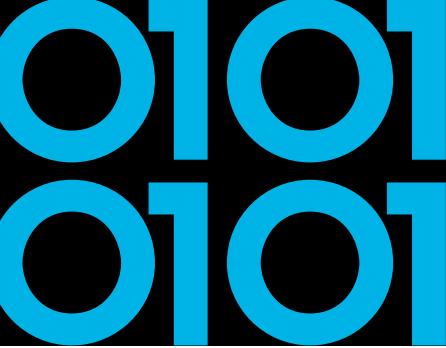
A build is **reproducible** if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts.

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.



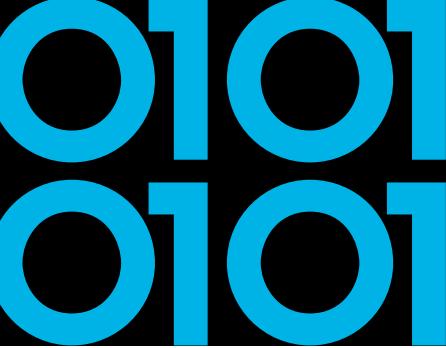
# Reproducible/Deterministic Builds

- 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’



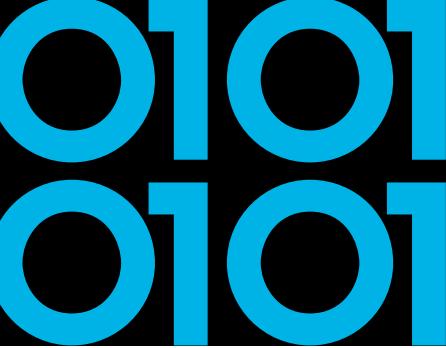
# Reproducible Build .NET6

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

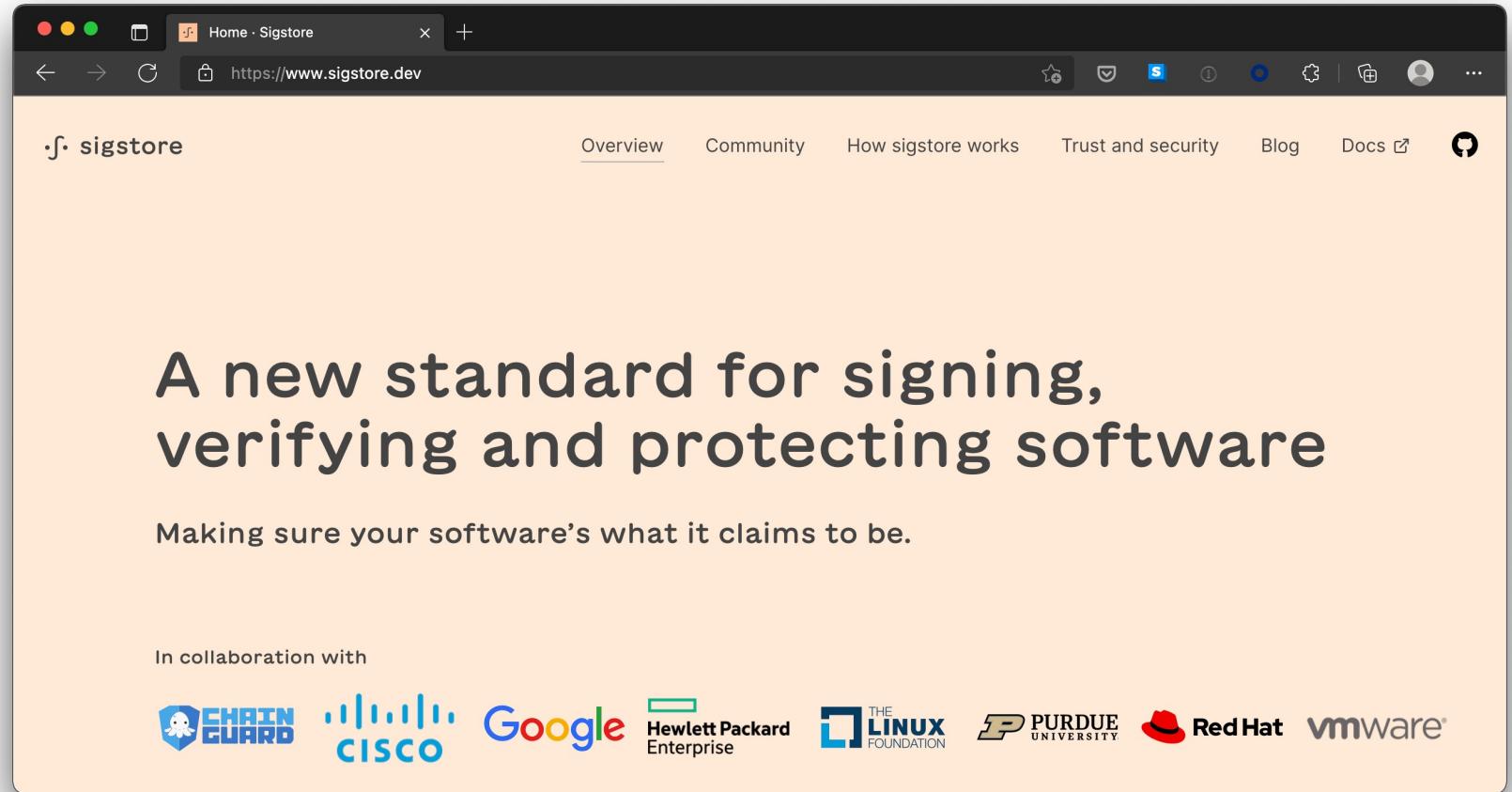


# Reproducible Build Validation

- 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, there's a navigation bar with links for Overview, Community, How sigstore works, Trust and security, Blog, Docs, and a user icon. Below the navigation, the main heading reads "A new standard for signing, verifying and protecting software" in large, bold, dark gray font. Underneath it, a subtitle says "Making sure your software's what it claims to be." At the bottom, there's a section titled "In collaboration with" featuring logos for ChainGuard, Cisco, Google, Hewlett Packard Enterprise, The Linux Foundation, Purdue University, Red Hat, and VMware.

Home · Sigstore

https://www.sigstore.dev

sigstore

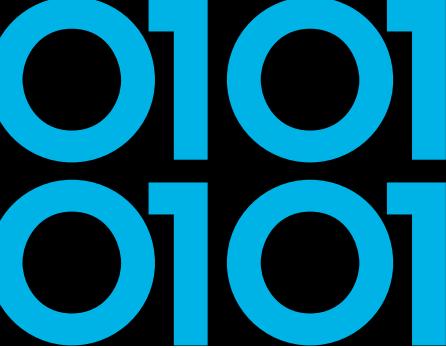
Overview    Community    How sigstore works    Trust and security    Blog    Docs

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



# Signing artifacts

## 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

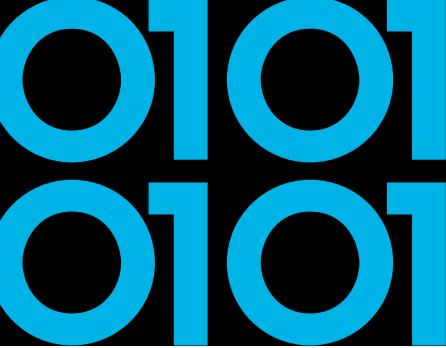
This means that open source software uploaded for distribution has a stricter, more standardized way of checking who's been involved, that it hasn't been tampered with. There's no risk of key compromise, so third parties can't hijack a release and slip in something malicious.

### 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.

```
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;
```

The diagram illustrates the sigstore workflow. At the top, a box labeled "DEVELOPERS, MAINTAINERS, MONITORS" contains three circular nodes: "SIGN AND PUBLISH ARTIFACTS", "PUBLISH SIGNING CERTIFICATES", and "MONITOR LOGS". Dotted lines connect each of these nodes to three corresponding rectangular boxes below: "FULCIO CERTIFICATE AUTHORITY", "SIGNATURE TRANSPARENCY LOG", and "KEY TRANSPARENCY LOG". Finally, dotted lines connect each of these three boxes to a bottom box labeled "TRUST ROOT".



# Signing artifacts

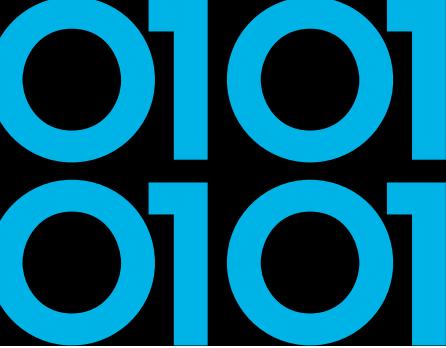
- Cosign can be used for signing files like binaries, packages and Docker images
- It can do keyless signing based on OpenID Connect
- GitHub Actions have released OpenID Connect support since end 2021

0101  
0101

# Cosign Keyless Signing



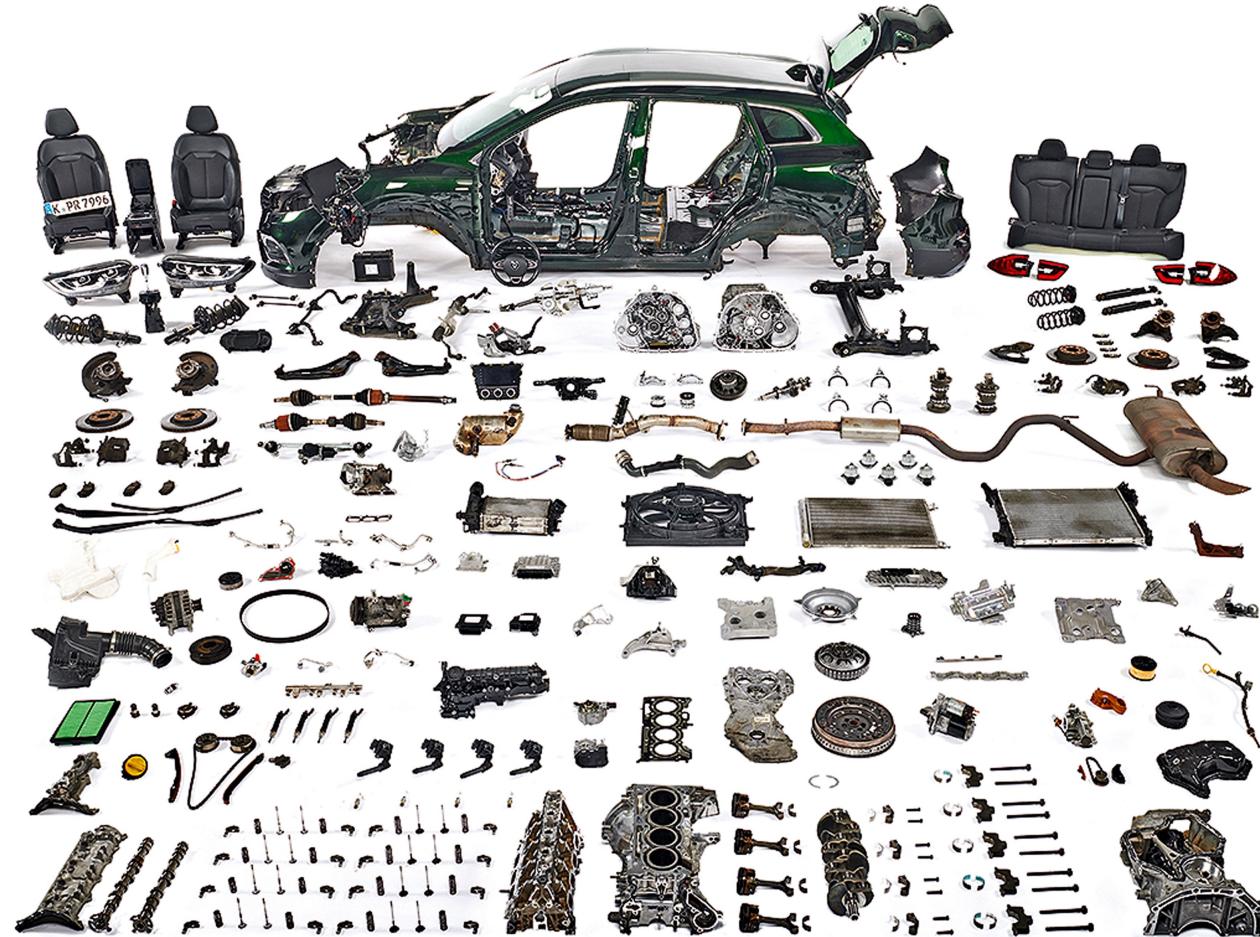
# Git Commit Signing Sigstore GitSign



The screenshot shows a web browser window with the URL <https://blog.sigstore.dev/introducing-gitsign-9f...>. The page title is "Introducing Gitsign". The author is listed as "Billy Lynch" with a "Follow" button. The post was published on "Jun 8 · 3 min read · Listen". The main content starts with "Introducing Gitsign". It explains that Sigstore aims to protect software supply chains by providing tools to make signing easy and transparent. It mentions components like Fulcio, Rekor, and Cosign. The text continues with "But securing software supply chains means more than just signing containers! Ideally, every step in our release pipeline should be signed — from the artifacts we produce tracing back all the way to the source they originated from." At the bottom, there are navigation icons for home, search, and user profile.

0101  
0101

# Automotive Industry



0101  
0101

# 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

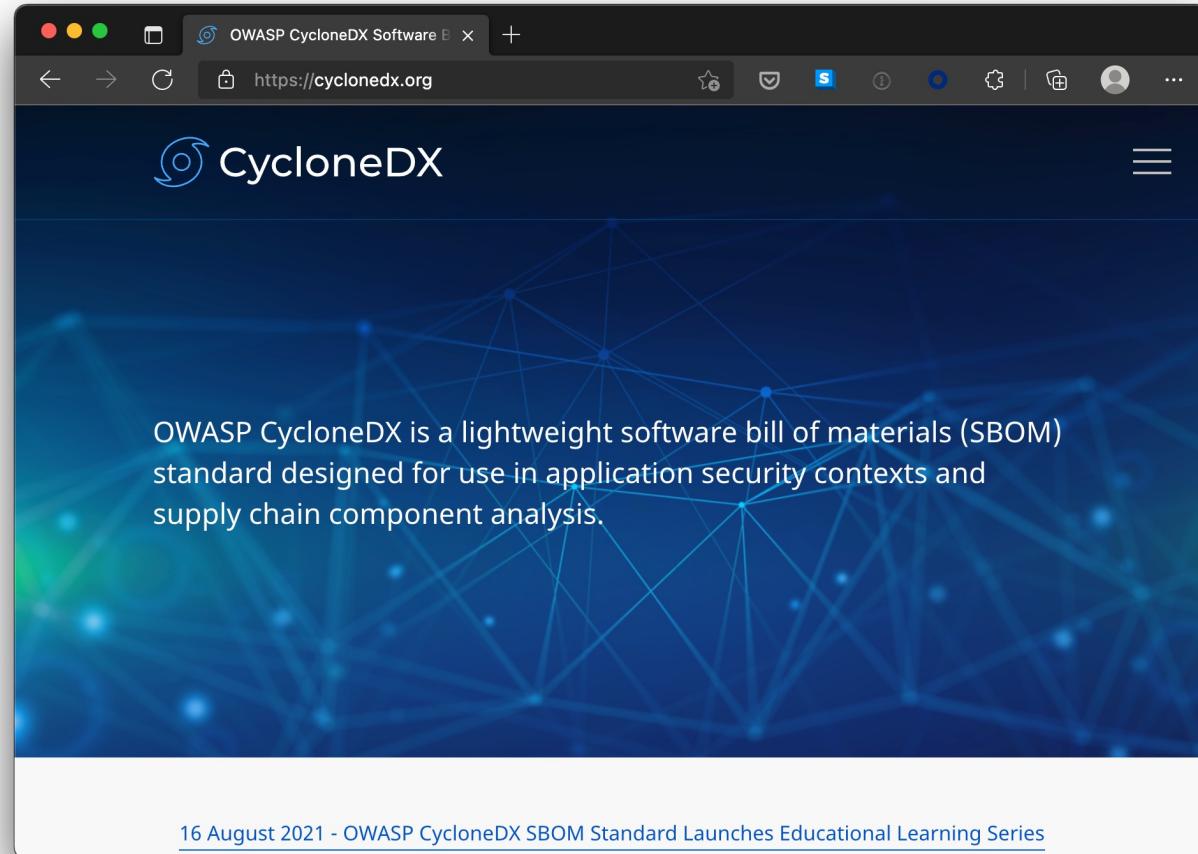


# Software Bill of Materials (SBOM)

- 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)



# 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](#).

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

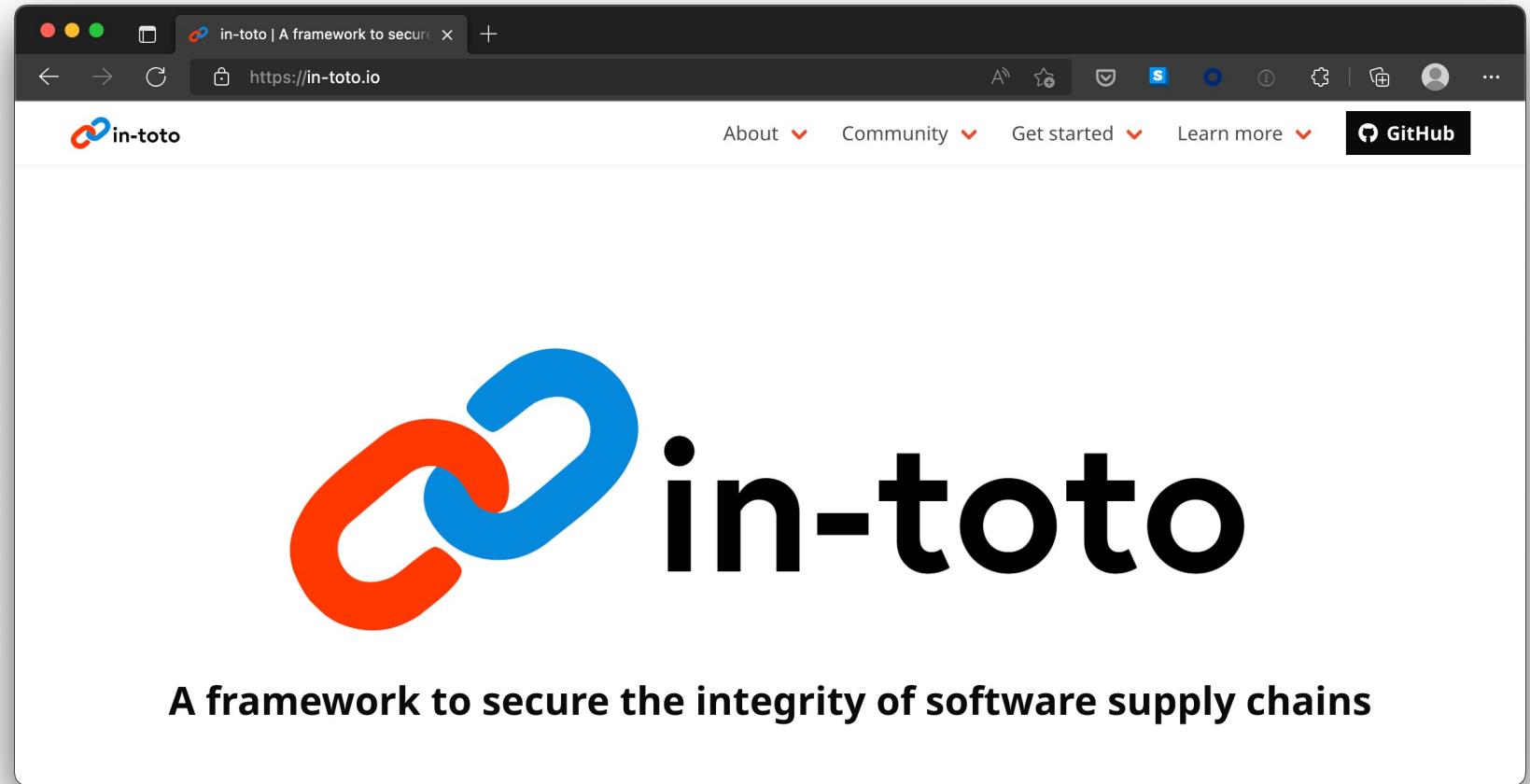
- # docker
- # Docker images
- # sbom
- # Secure Software Supply Chain

Categories

- Community
- Company
- Engineering
- Newsletters
- Products

# In-toto

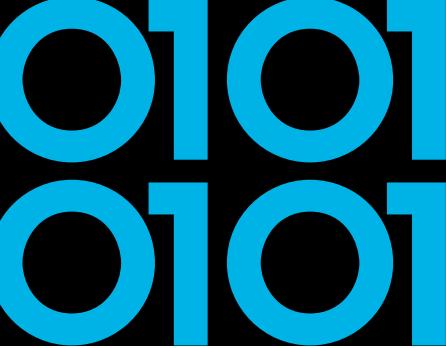
0101  
0101





# 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



# In-Toto - Demo



## In-Toto - Demo - Terminology

- **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

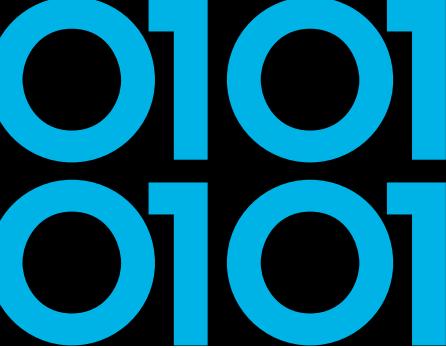
# Microsoft SBOM Tool

0101  
0101

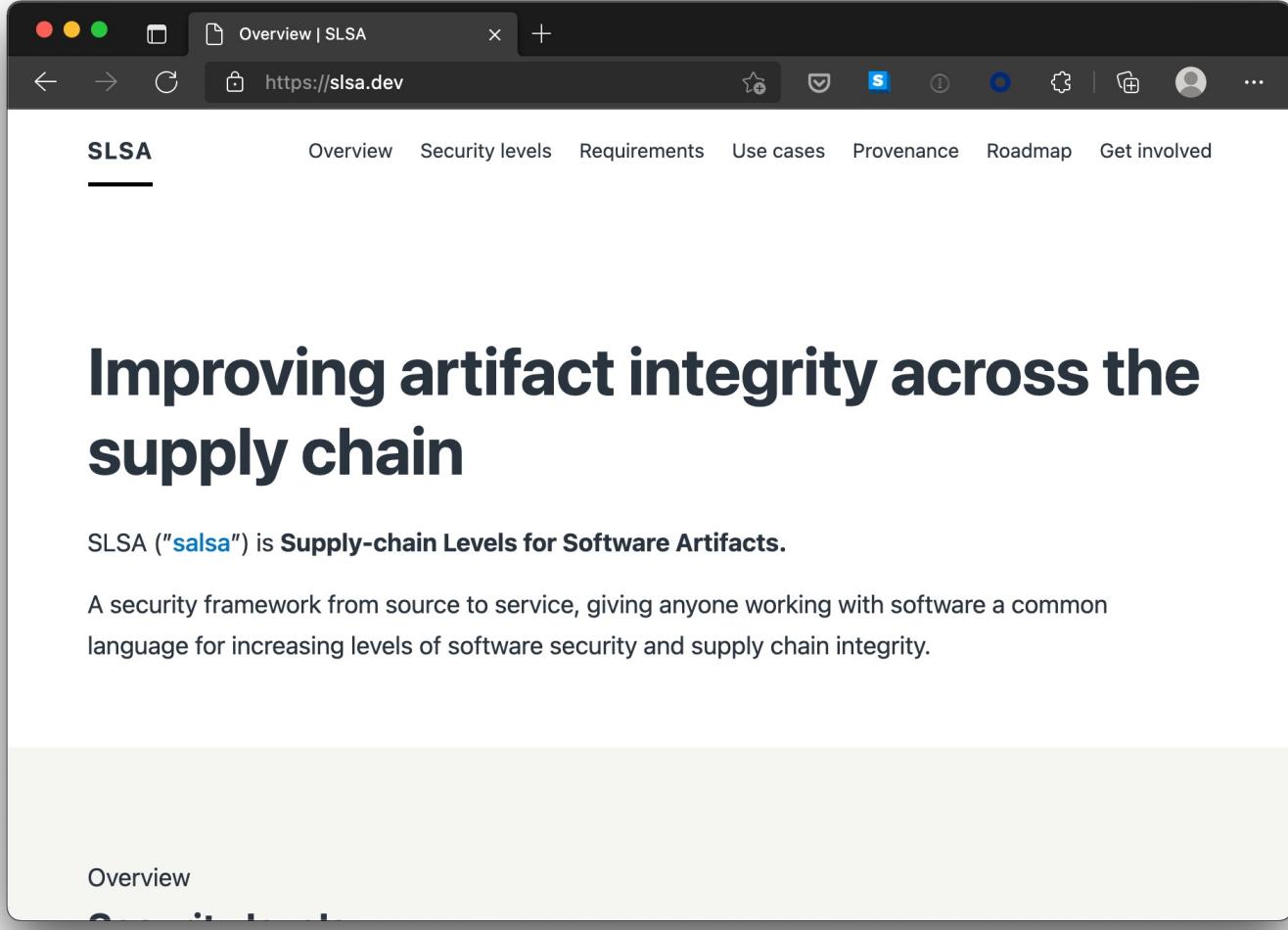
A screenshot of a web browser displaying the GitHub README page for the Microsoft SBOM Tool. The page has a dark theme. At the top, there's a navigation bar with icons for back, forward, search, and refresh, followed by the URL <https://github.com/microsoft/sbom-tool>. Below the URL is a toolbar with various icons. The main content area starts with a header titled "SBOM Tool". Below the header are several status indicators: "Build passing", "downloads 13k", and "release v0.2.2". A "Languages" section shows a chart with two entries: "C# 99.5%" and "PowerShell 0.5%". The "Introduction" section contains the text: "The SBOM tool is a highly scalable and enterprise ready tool to create SPDX 2.2 compatible SBOMs for any variety of artifacts." The "Table of Contents" section lists the following items:

- Download and Installation
- Run the tool
- Integrating SBOM tool to your CI/CD pipelines
- Telemetry
- Contributing
- Security
- Trademarks

The "Download and Installation" section includes a note: "Executables for Windows, Linux, macOS".



# Google SLSA



The screenshot shows a web browser window displaying the SLSA website at <https://slsa.dev>. The page title is "Overview | SLSA". The navigation bar includes links for SLSA, Overview, Security levels, Requirements, Use cases, Provenance, Roadmap, and Get involved. The main content 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 "Overview" section with some text and a "Continue reading" button.

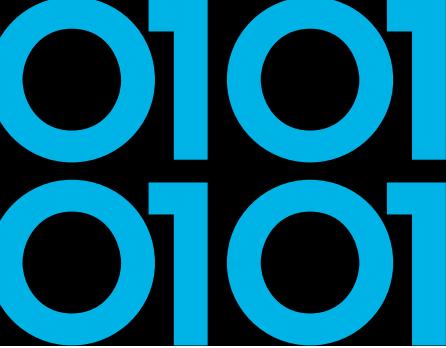
Improving artifact integrity across the supply chain

SLSA ("salsa") is **Supply-chain Levels for Software Artifacts**.

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.

Overview

Continue reading



# 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



# SLSA3 Generator GitHub Actions

The screenshot shows a web browser window with the URL <https://slsa.dev/blog/2022/08/slsa-github-workflows-generic-ga>. The page title is "General availability of SLSA3 Generic Generator for GitHub Actions" by Ian Lewis, Laurent Simon, Asra Ali, posted on 29 Aug 2022. The text 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.

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.



# MyAwesomePDFComponent Demo

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

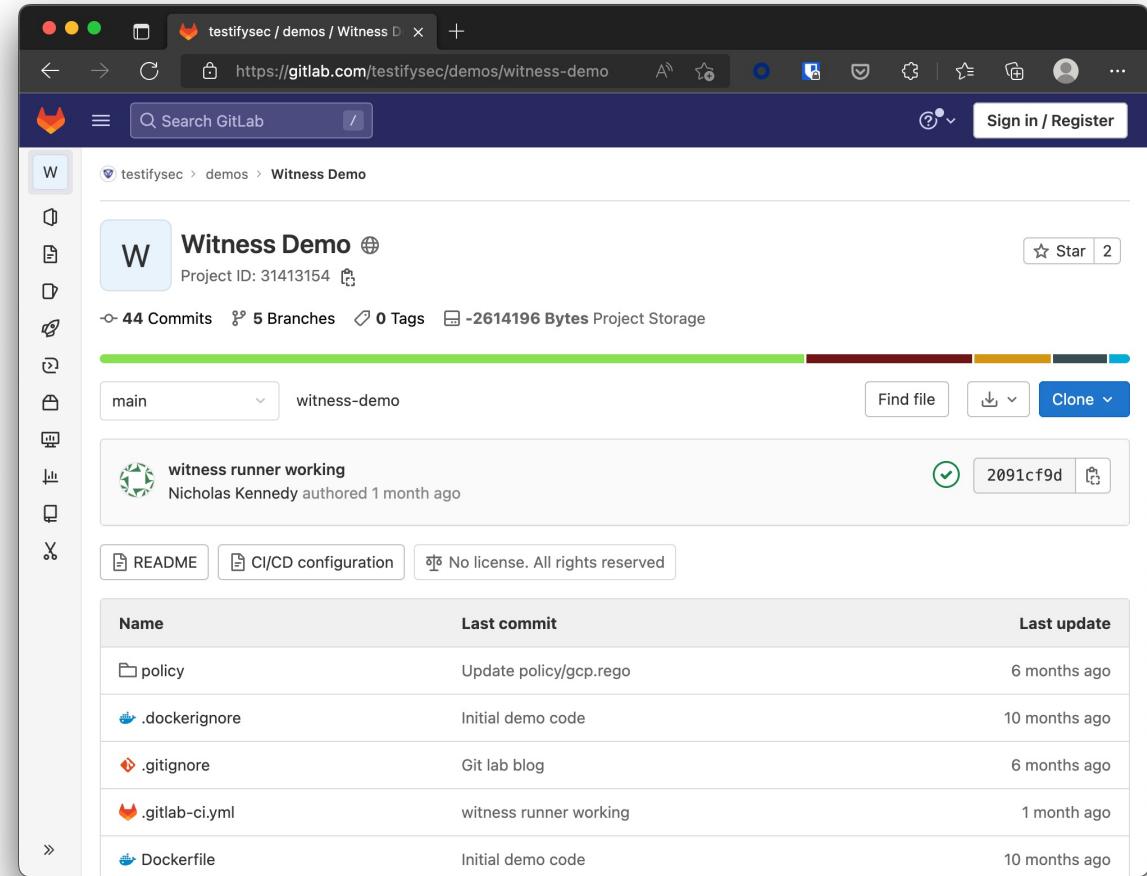
0101  
0101

# 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 titled "All SUSE Products" and "SLSA: Securing the Software Supply Chain". Below the title, there is an "Abstract" section which states: "This document details how SUSE, as a long-time champion and expert of software supply chain security, prepares for SLSA L4 compliance." A "Disclaimer" section follows, stating: "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". There is also a "REPORT DOCUMENTATION BUG" link.

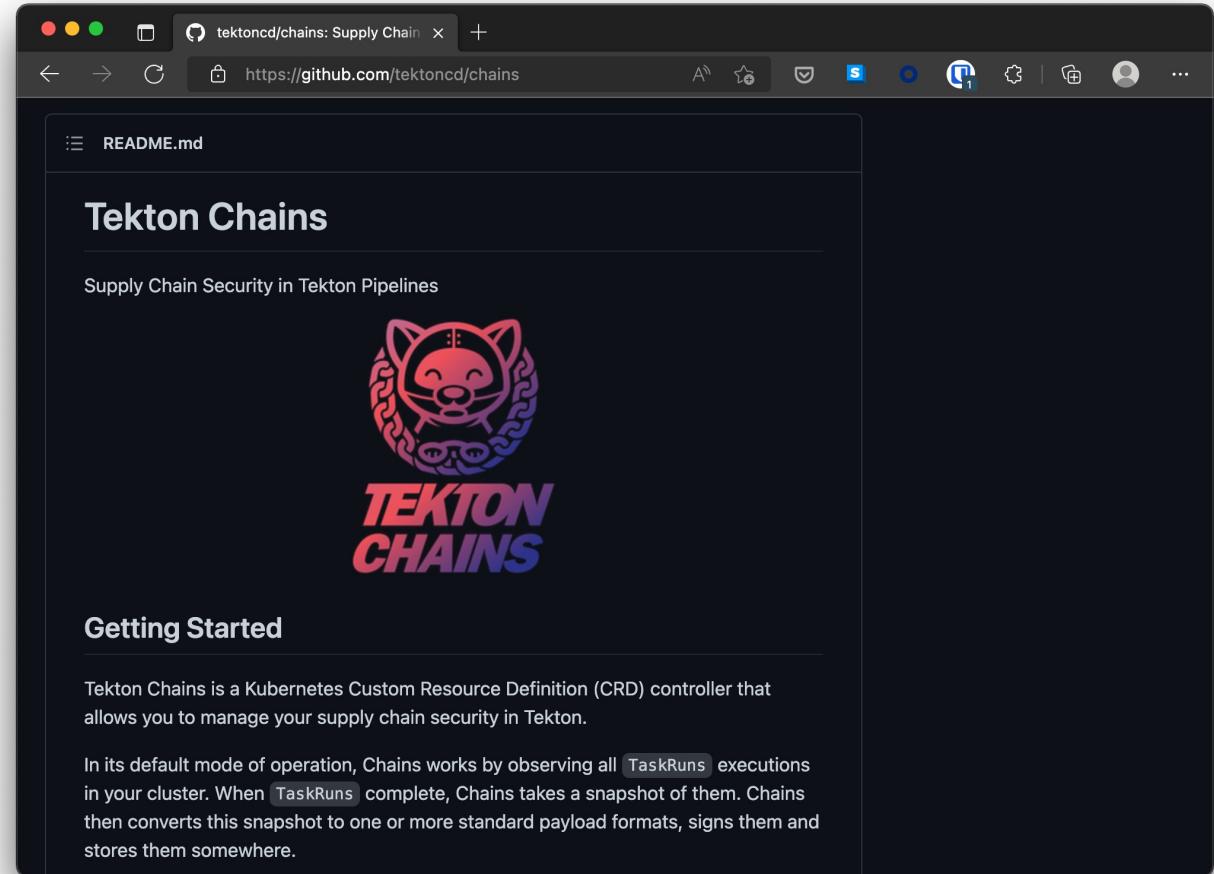
0101  
0101

# Witness & GitLab Attestator



0101  
0101

# Open Shift - Tekton - Tekton Chains

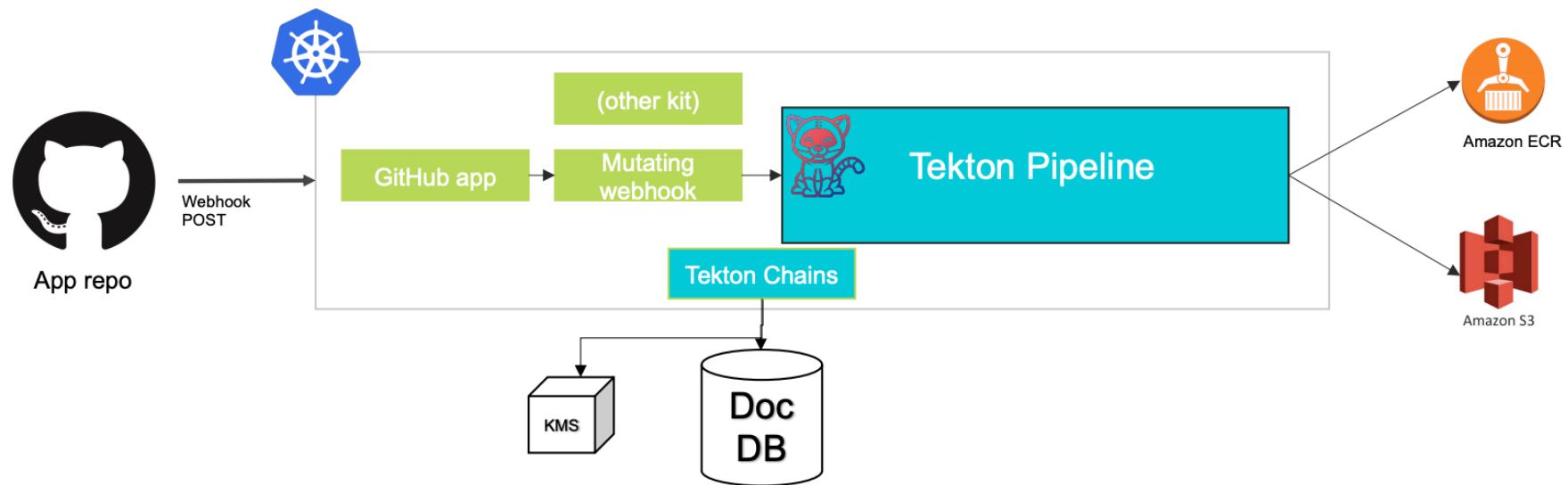




# SolarWinds Project Trebuchet



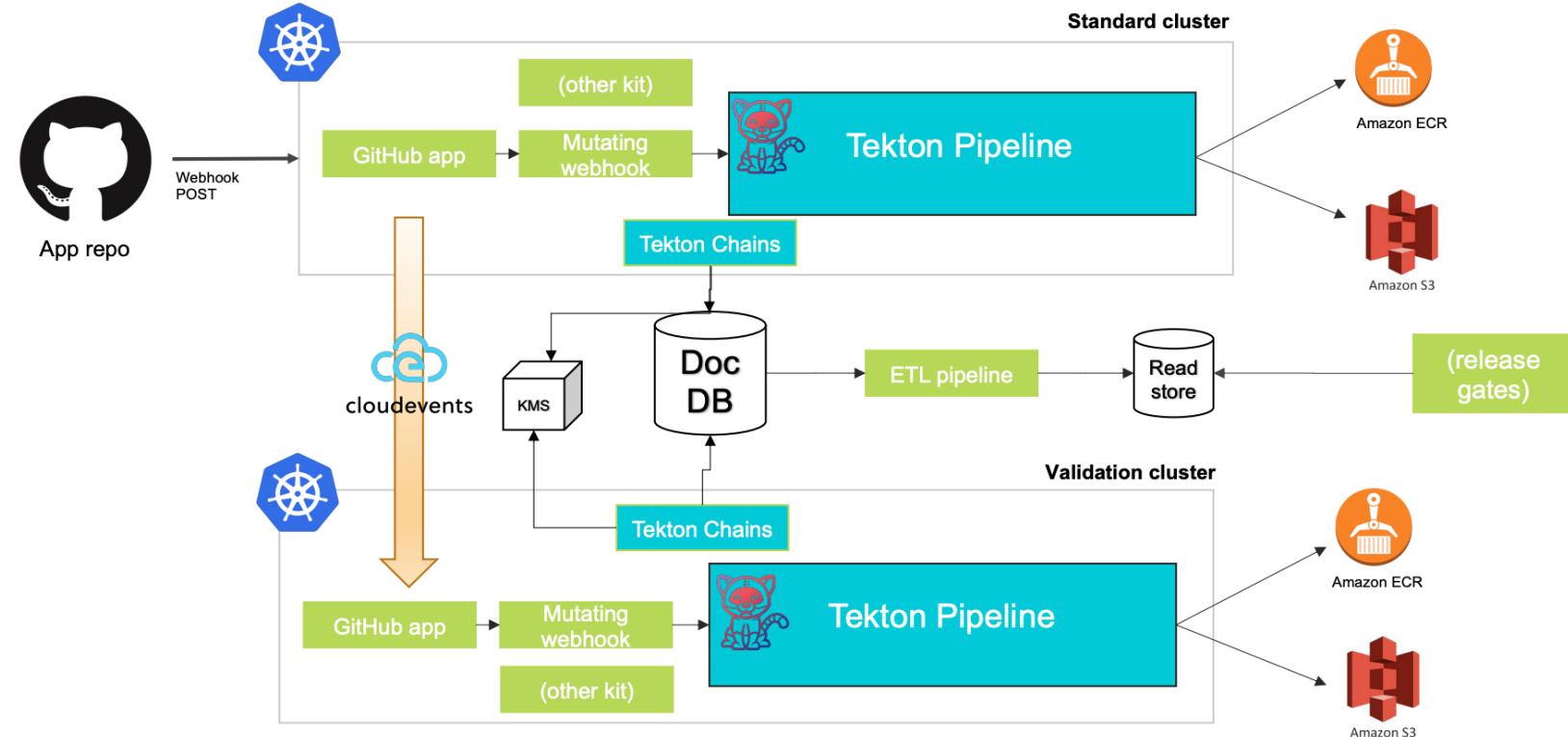
## Pipeline With Attestations

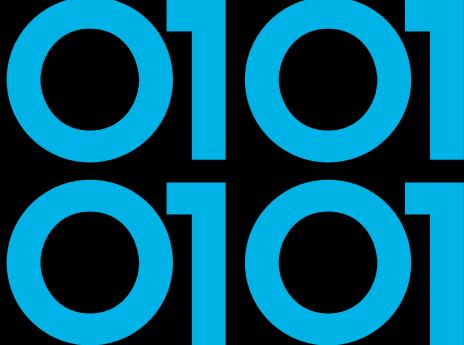


# SolarWinds Project Trebuchet



## Reading Results



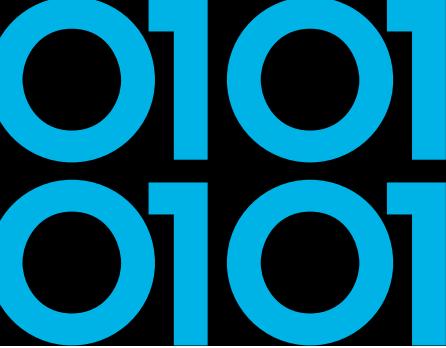


# Grafeas and Kritis by Google

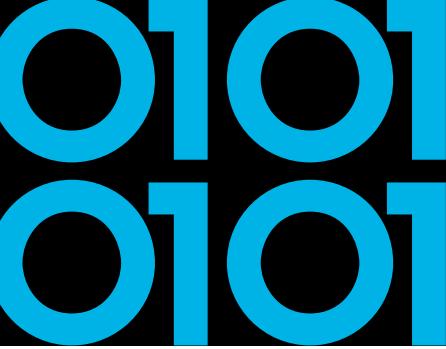
- Google released in 2019
- Grafeas - Component Metadata API
  - Container Analysis API on Google Cloud Platform
- Kritis - Deployment Authorization for Kubernetes Apps
  - Binary Authorization on Google Cloud Platform



# Azure Policy

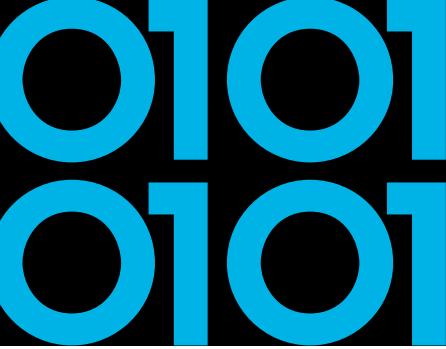


The screenshot shows a Microsoft Docs page titled "Tutorial: Build policies to enforce compliance". The page is part of the "Azure Policy documentation" section, under the "Tutorials" category. The main content area features the title "Tutorial: Create and manage policies to enforce compliance" in large, bold, white font. Below the title, there's a summary: "Article • 04/03/2022 • 17 minutes to read • 4 contributors". The left sidebar contains a navigation tree with links like "Create a custom policy definition", "Manage tag governance", and "Implement Azure Policy as Code with GitHub". A "Download PDF" button is also present in the sidebar.



# 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

