



# Securing your .NET application software supply-chain

# Niels Tanis

0101  
0101

# Who am I?

- Niels Tanis
- Sr. Principal Security Researcher
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - Research on static analysis for .NET apps
- Microsoft MVP - Developer Technologies



# Securing your .NET application software supply-chain

0101  
0101





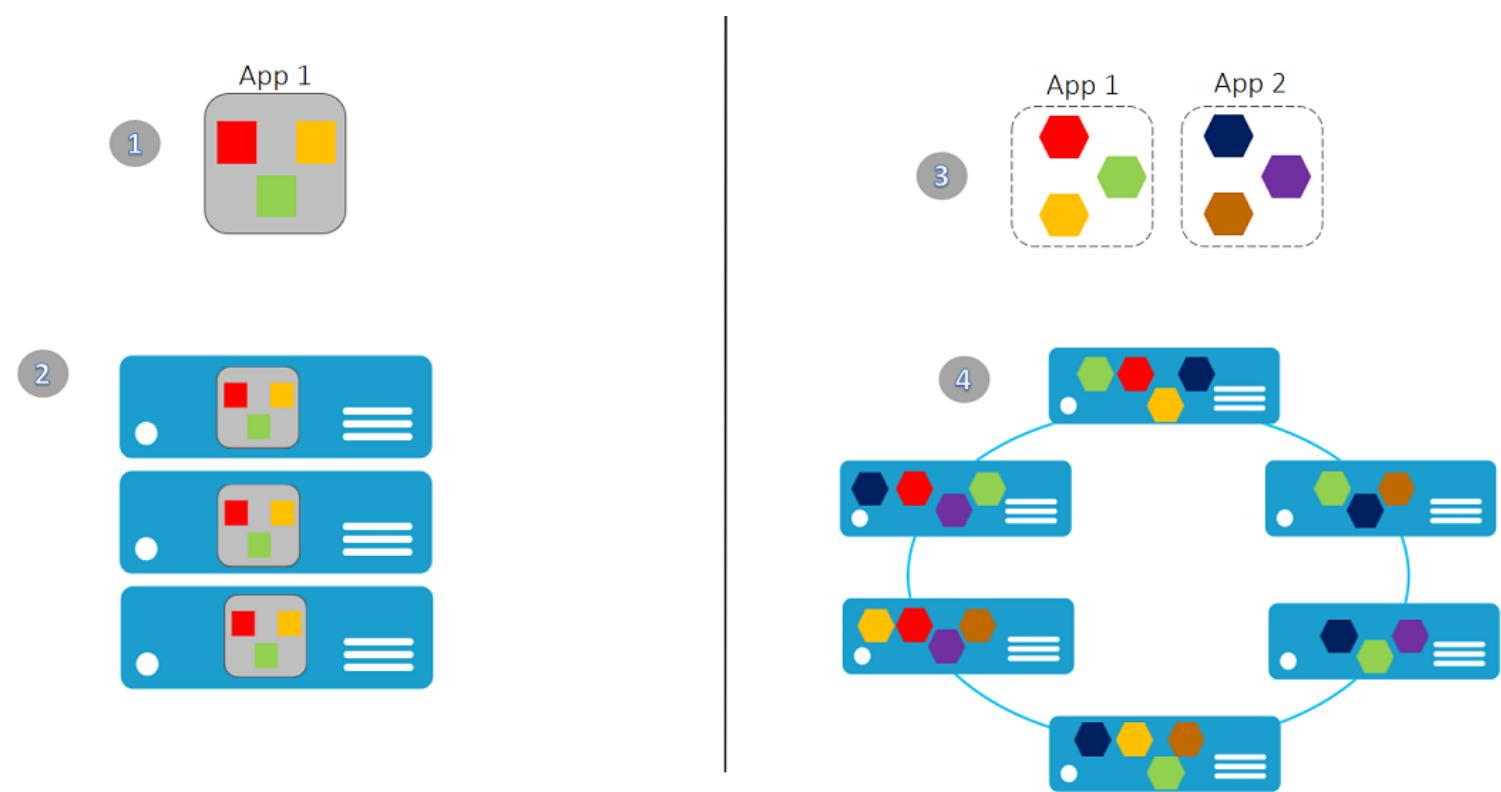
# Agenda

- 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

0101  
0101

- Monolith
- Microservices
- Serverless
- Cloud-Native



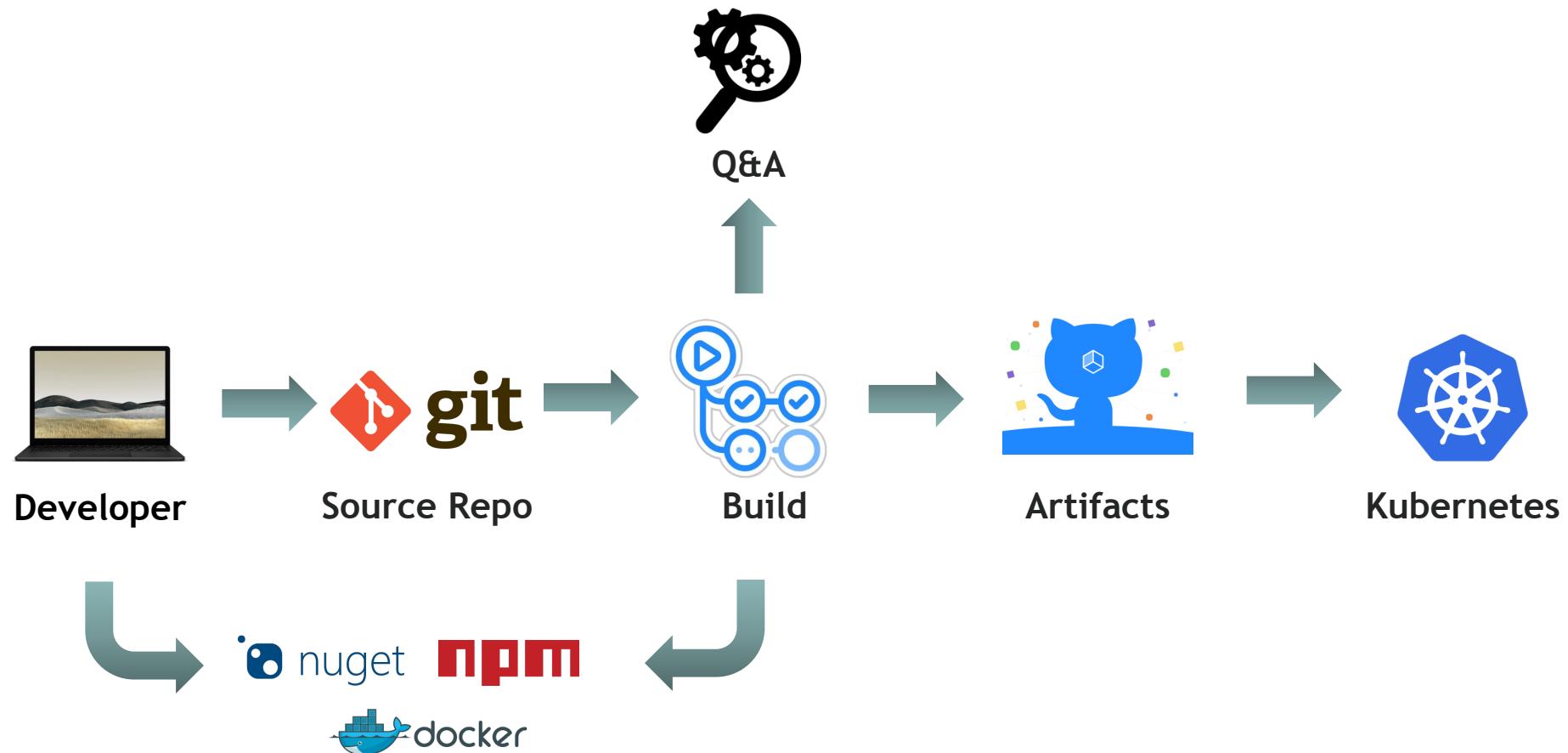
0101  
0101

# What is a Supply Chain?



# Software Supply Chain

0101  
0101



0101  
0101

# SolarWinds SunSpot

The screenshot shows a web browser window with the following details:

- Title Bar:** SUNSPOT Malware: A Technical Analysis
- Address Bar:** https://www.crowdstrike.com/blog/sunspot-malware-technical-analysis/
- Header:** CROWDSTRIKE | BLOG
- Navigation:** Featured, Recent, Videos, Categories, Start Free Trial
- Content:** Article title: SUNSPOT: An Implant in the Build Process (January 11, 2021, by CrowdStrike Intelligence Team, Research & Threat Intel). Below the title is a large, abstract sunburst graphic.
- Categories Sidebar:**
  - Endpoint & Cloud Security (374)
  - Engineering & Tech (71)
  - Executive Viewpoint (143)
  - From The Front Lines (186)
  - Identity Protection (28)
  - Observability & Log Management (71)
  - People & Culture (86)
  - Remote Workplace (20)
  - Research & Threat Intel (160)
  - Tech Center (149)
- Connect With Us:** Links to Twitter, Facebook, LinkedIn, YouTube, and RSS feed.

0101  
0101

# 3CX Supply Chain Attack

The screenshot shows a web browser window with the URL <https://www.wired.com/story/3cx-supply-chain-attack-times-two/>. The page is from the WIRED website, specifically the SECURITY section. The author is ANDY GREENBERG, and the date is APR 20, 2023 8:00 AM. The main headline reads "The Huge 3CX Breach Was Actually 2 Linked Supply Chain Attacks". Below the headline is a subtext: "The mass compromise of the VoIP firm's customers is the first confirmed incident where one software-supply-chain attack enabled another, researchers say." A red decorative graphic featuring a row of dominoes is at the bottom.

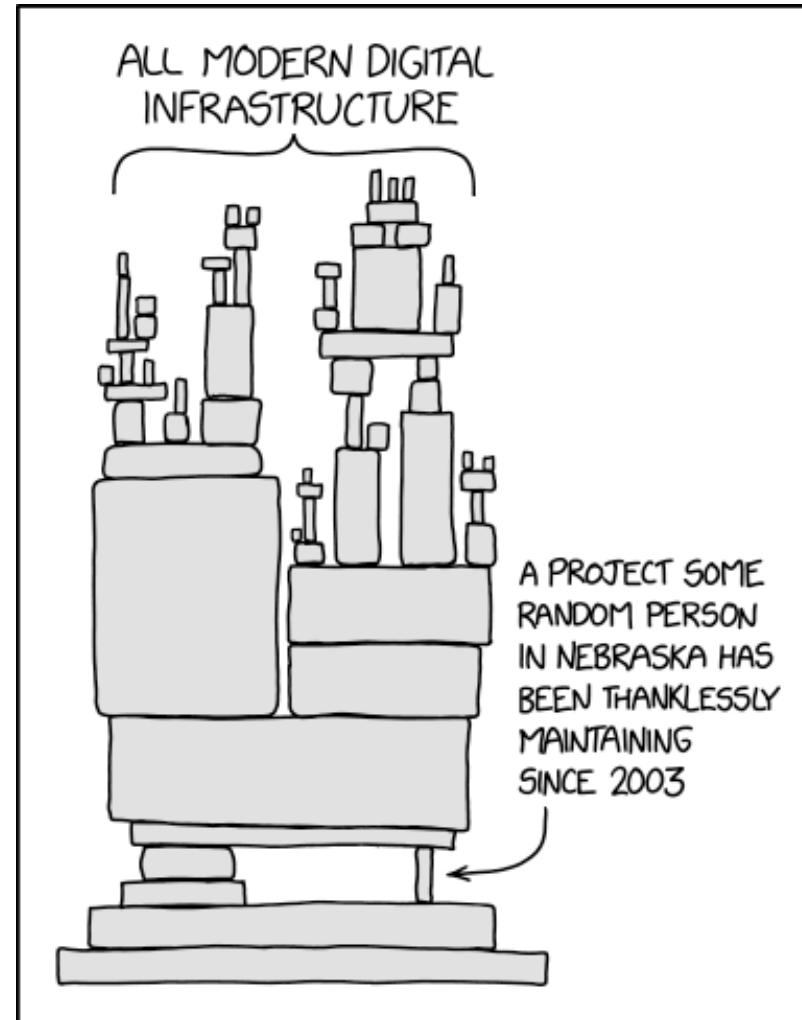
# MSI UEFI Signing Keys

0101  
0101

A screenshot of a web browser displaying an Ars Technica article. The title of the article is "Leak of MSI UEFI signing keys stokes fears of ‘doomsday’ supply chain attack". The article is by Dan Goodin and was published on May 11, 2023, at 2:56 AM. The Ars Technica logo is visible in the top left corner, and there are "SUBSCRIBE" and "SIGN IN" buttons in the top right corner. Below the title, there is a sub-headline: "With no easy way to revoke compromised keys, MSI, and its customers, are in a real pickle." A cartoon illustration of a green worm-like character with large eyes and a wide smile is shown on the right side of the article.

0101  
0101

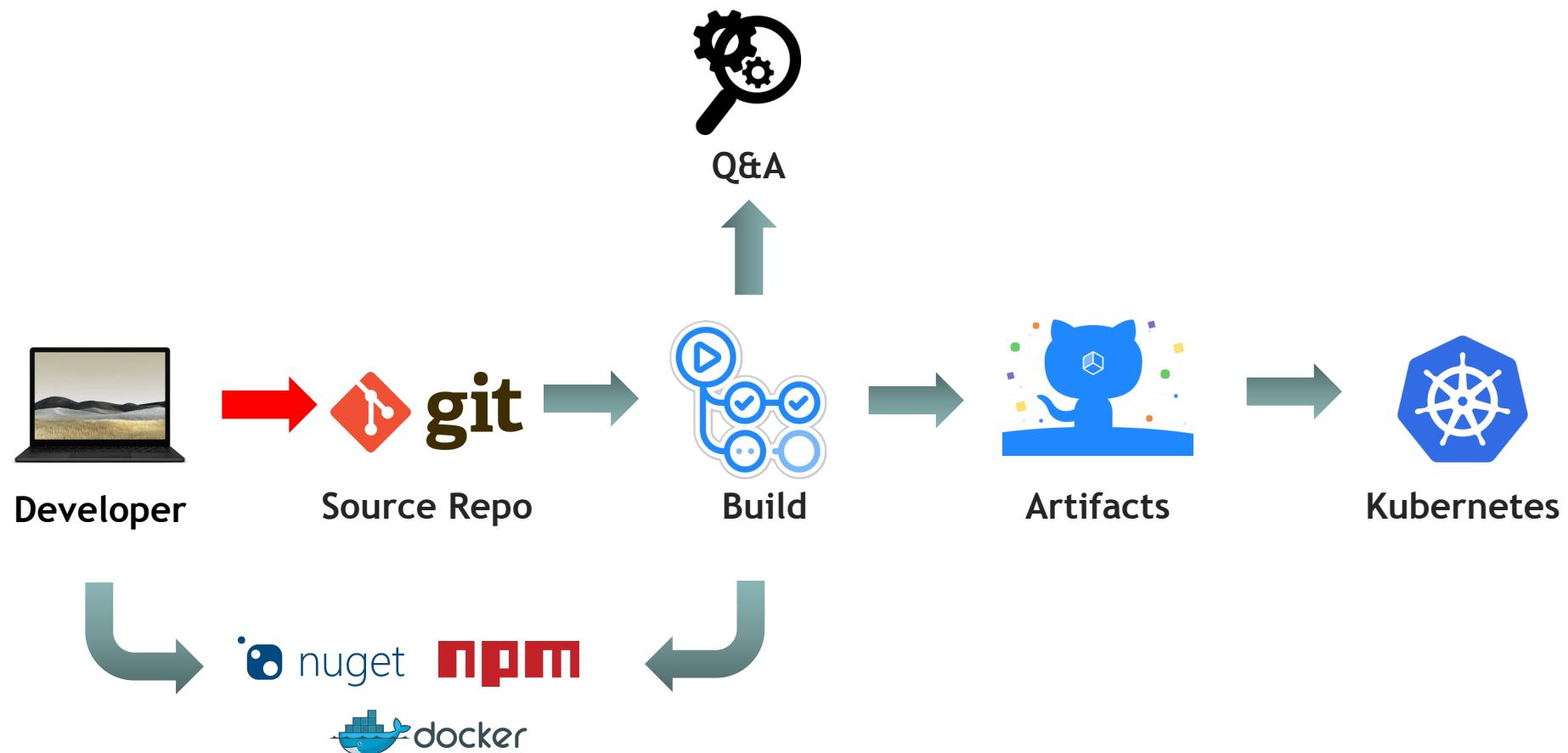
# XKDC - Dependency



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

# Software Supply Chain

0101  
0101



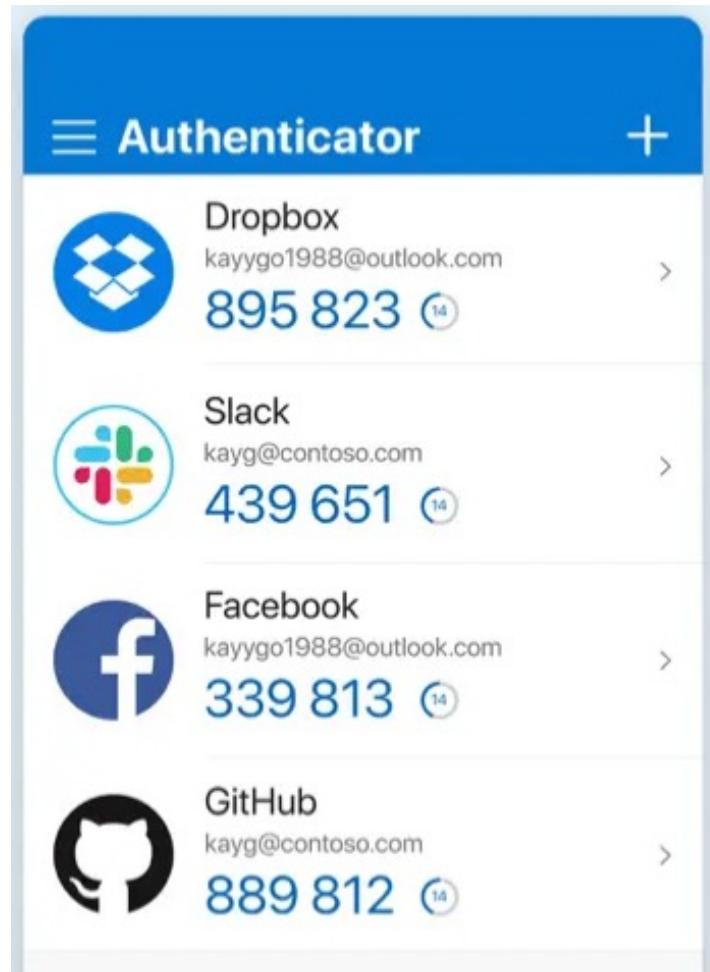
# GitHub account

0101  
0101

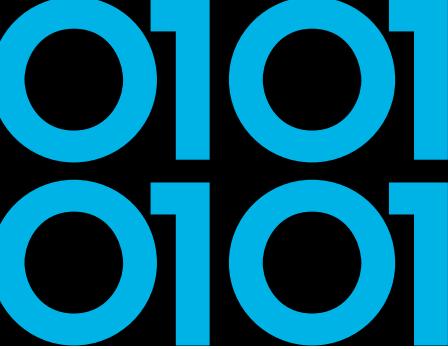
A screenshot of a web browser window displaying a ZDNet article. The URL in the address bar is <https://www.zdnet.com/article/canonical-github-account-hacked-ubuntu-source-code-safe/>. The page has a yellow header with the ZDNET logo and navigation links for trending, tech, innovation, business, security, advice, and buying guides. The main content area features a large headline: "Canonical GitHub account hacked, Ubuntu source code safe". Below the headline is a sub-headline: "Ubuntu source code appears to be safe; however Canonical is investigating." A small profile picture of a man and the text "Written by Catalin Cimpanu, Contributor on July 7, 2019" are also visible.

0101  
0101

# Use MFA on source-repository



# GIT Commit Signing



A screenshot of a GitHub commit page for the repository `nielstanis / CodeEurope2023-SupplyChain`. The commit was made by `nielstanis` on May 22, 2023, and is titled `Initial content`. The commit message includes the text `nielstanis committed now`. The commit is marked as `Verified`, and the SHA-1 hash is `1ed5bdf`. The GitHub interface shows standard navigation links like Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The commit is listed under the `main` branch.

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`

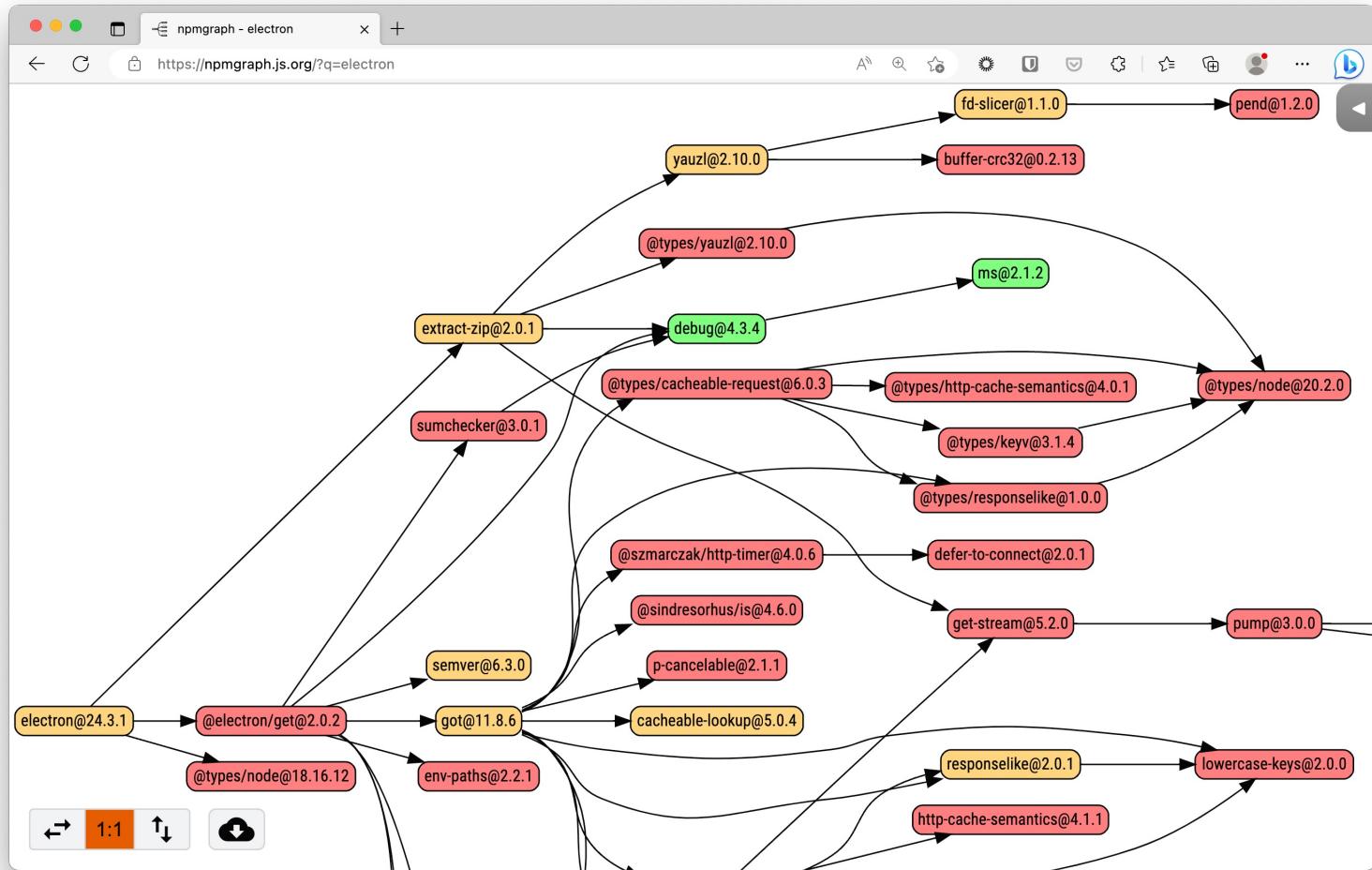
**Abstract:** Open source software (OSS) has thrived since the forming of Open Source Initiative in 1998. A prominent example is the Linux kernel, which has been used by numerous major software vendors and empowering billions of devices. The higher availability and lower costs of OSS boost its adoption,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

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].

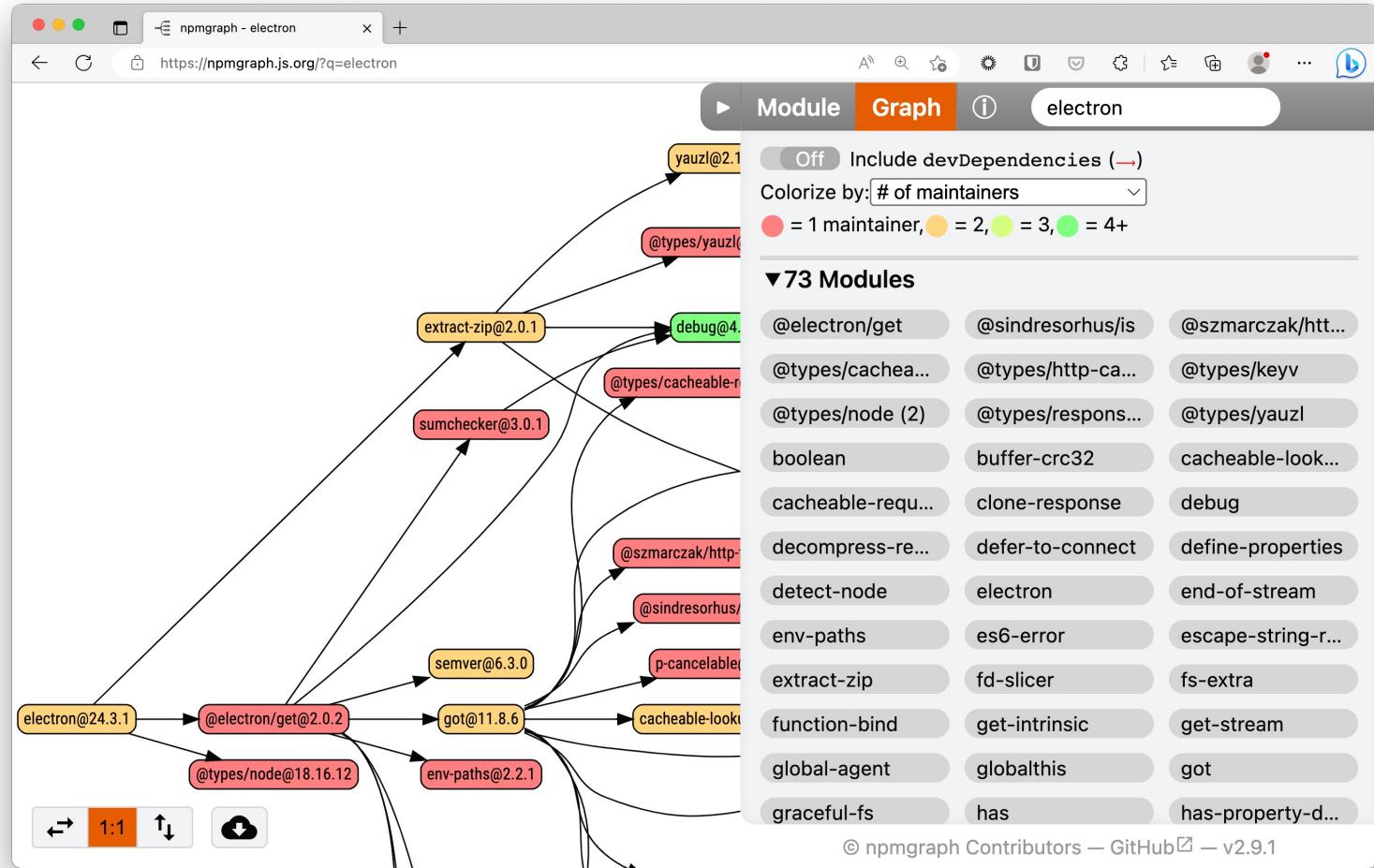
A prominent example of OSS is the Linux kernel, which is one of the largest open-source projects—more than 28 million lines of code used by billions of devices. The Linux kernel involves more than 22K contributors. Any person or company can contribute to its development, e.g., submitting a patch

# Visual Studio Code



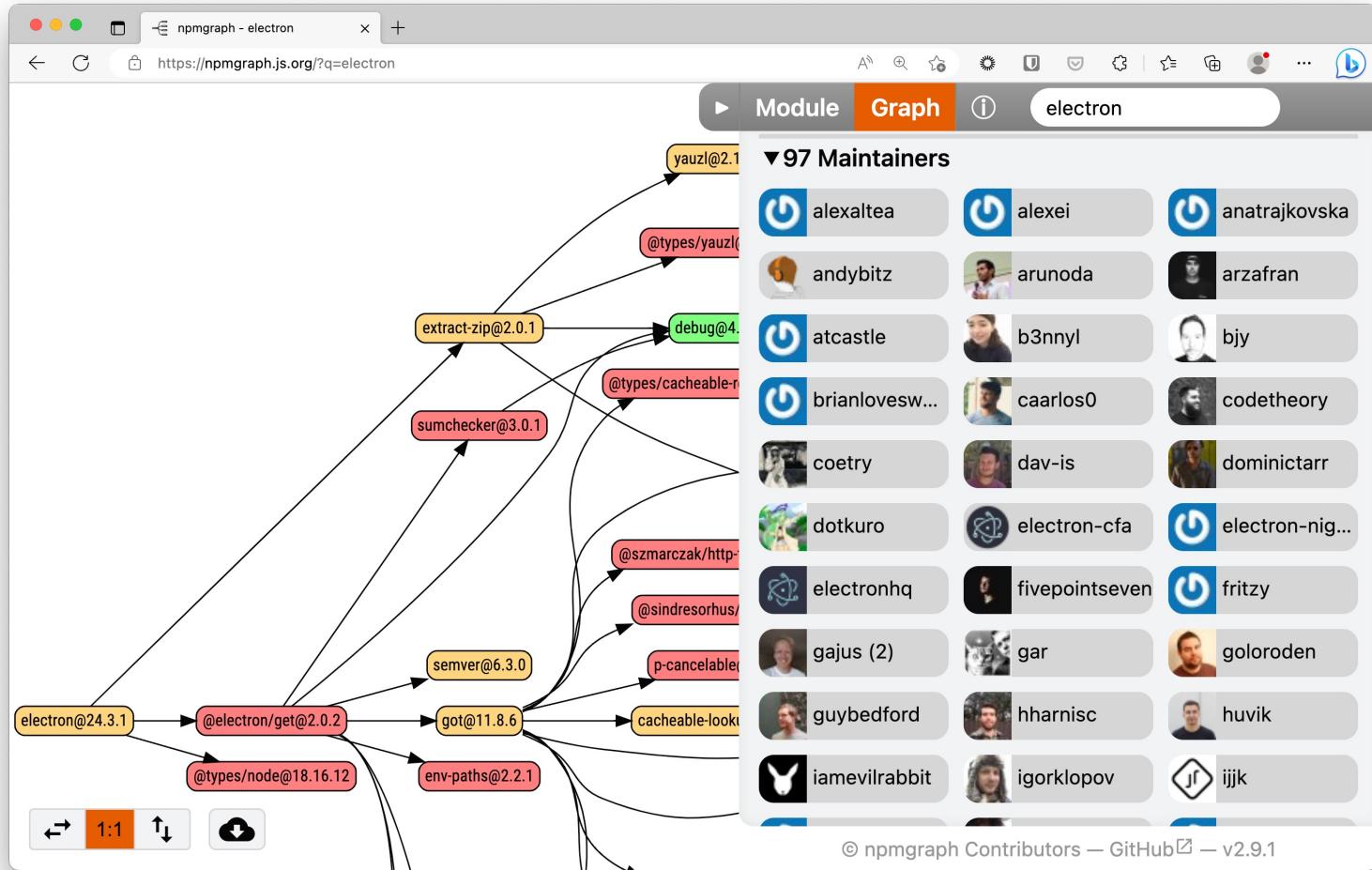
0101  
0101

# Visual Studio Code



0101  
0101

# Visual Studio Code



0101  
0101

# Visual Studio Code

CVE-2023-24893 - Security Update Guide

https://msrc.microsoft.com/update-guide/vulnerability/CVE-2023-24893

Microsoft | MSRC | Security Updates | Acknowledgements | Developer

MSRC > Customer Guidance > Security Update Guide > Vulnerabilities > CVE 2023 24893

## Visual Studio Code Remote Code Execution Vulnerability

CVE-2023-24893  
Security Vulnerability

Released: Apr 11, 2023

Assigning CNA: Microsoft

[CVE-2023-24893](#)

Impact: Remote Code Execution Max Severity: Important

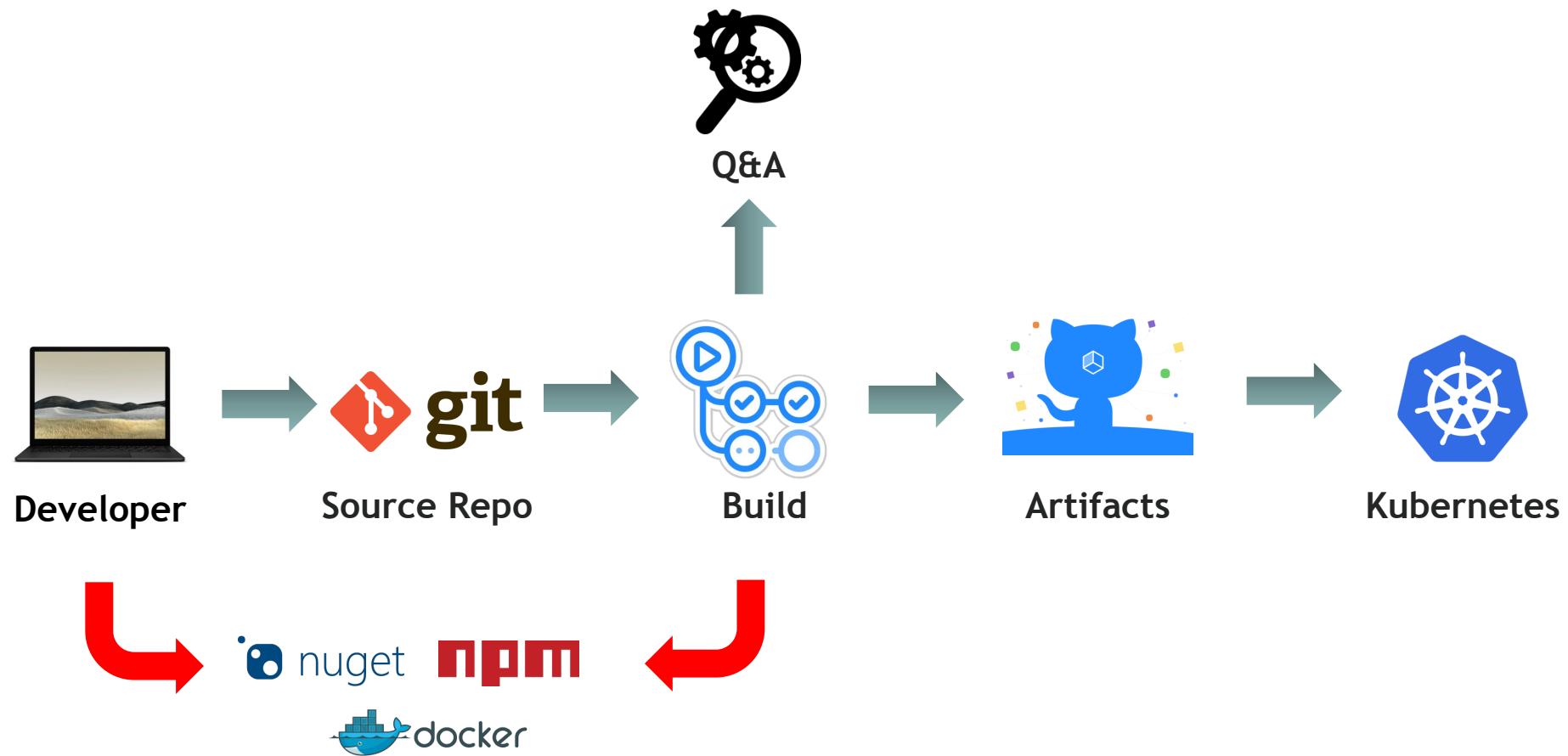
0101  
0101

# Visual Studio Code



# 3rd Party Libraries

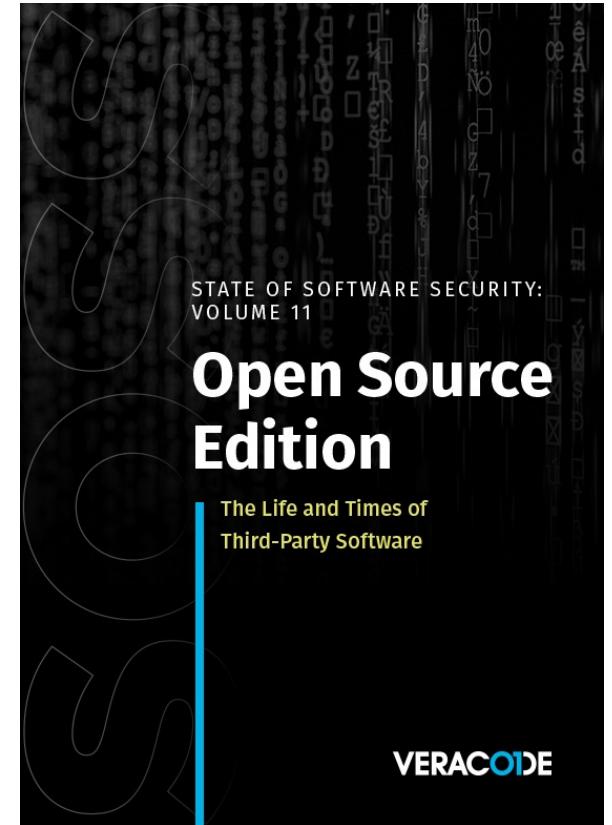
0101  
0101



# State Of Software Security v11 2021

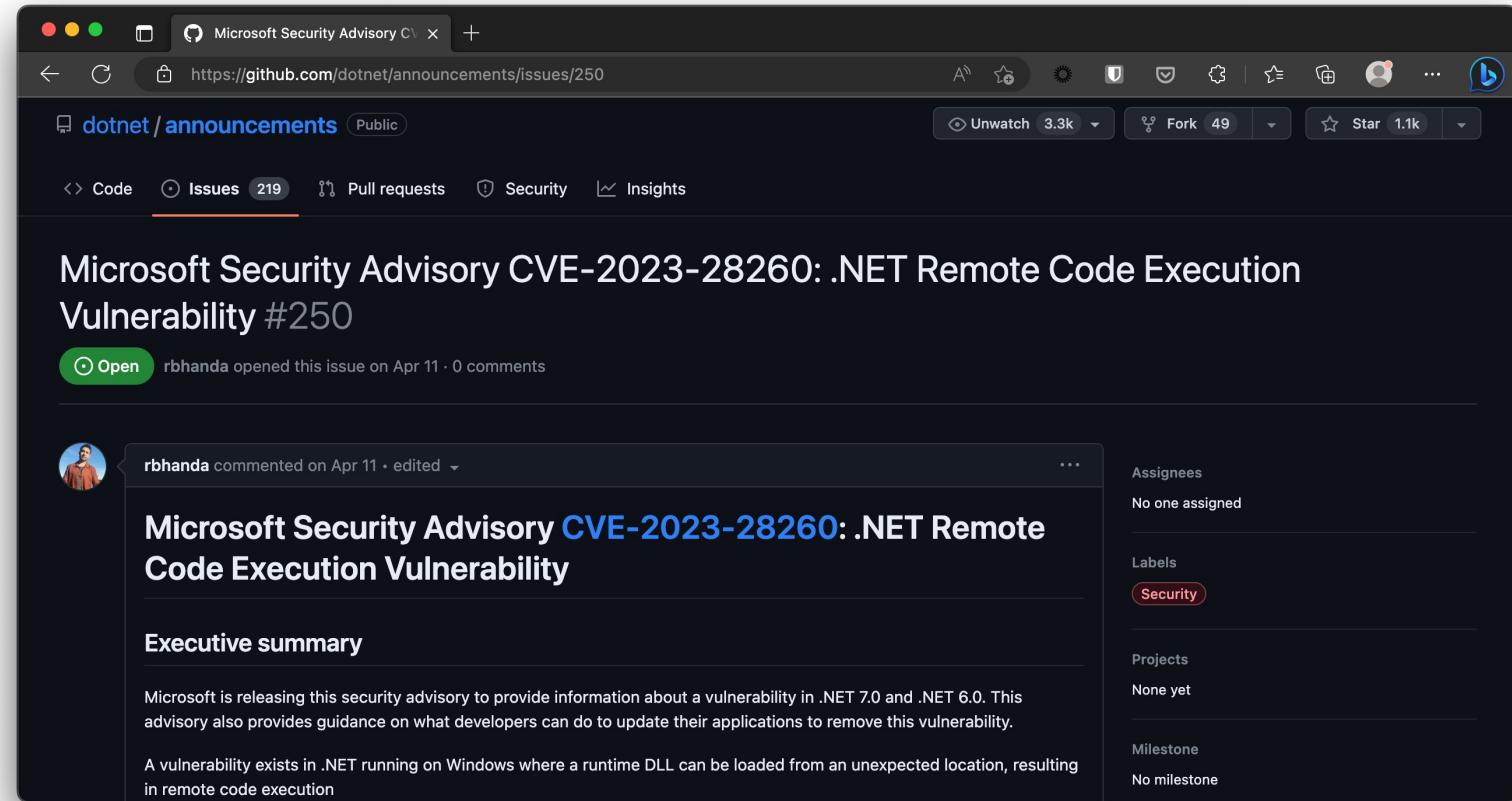
0101  
0101

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



0101  
0101

# Vulnerabilities in libraries



# 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 mention recent incidents on the npm registry, investigations, and future security investments. The second post specifically discusses the introduction of enhanced login verification for npm publishers starting December 7th.

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 at the bottom left, along with the date "Feb 9" and a note "11 min read". To 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

0101  
0101

## 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>



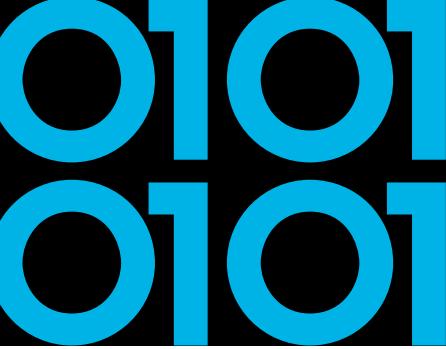
# Dependency Confusion - NuGet

- 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

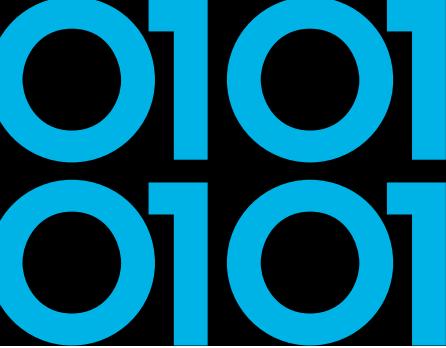
- Intent of library, know what's inside!
- Keep in mind that's a transitive list of dependencies
- My talk 'Sandboxing .NET Assemblies'
  
- Open Source Security Foundation - OpenSSF
- Security Scorecards - Security health metrics for Open Source



# Security Scorecards - OpenSSF

The screenshot shows the GitHub README page for the `ossf/scorecard` repository. The page features a large title "Security Scorecards" and a prominent "10" score indicator. Below the title, there's a list of badges including "openssf scorecard 8.6", "openssf best practices in progress 54%", "build passing", "CodeQL passing", "reference", "go report A+", "codecov 45%", "slack chat", and "SLSA level 3". The "Overview" section contains a bulleted list: "What Is Scorecards?", "Prominent Scorecards Users", and "Scorecards' Public Data". The "Using Scorecards" section lists: "Scorecards GitHub Action", "Scorecards REST API", "Scorecards Badges", "Scorecards Command Line Interface" (with sub-points for Prerequisites, Installation, Authentication, and Basic Usage). To the right of the text, there's a cartoon illustration of a robot holding up a sign with the number "10".

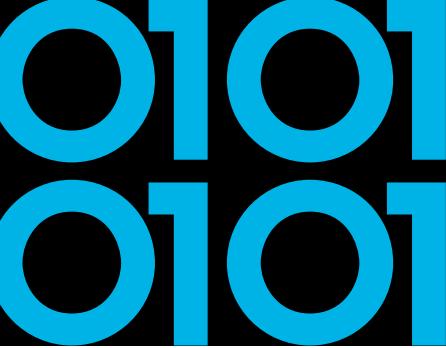
# Deps.Dev by Google



The screenshot shows the Deps.Dev website interface. At the top, there's a navigation bar with links for 'About', 'Documentation', and 'Blog'. Below the header, a main section titled 'Understand your dependencies' contains a paragraph about the importance of dependency management. To the right, a table lists various dependency statistics:

Dependency Type	Count
npm PACKAGES	3.21M
Go MODULES	1.00M
Maven ARTIFACTS	544k
PyPI PACKAGES	434k
NuGet PACKAGES	360k
Cargo CRATES	115k

At the bottom of the screenshot, there's a search bar with the placeholder 'Search for open source packages, advisories and prs' and a dropdown menu set to 'All systems'.



# 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 and its malware capabilities.

.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.



# 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 website with a dark blue header featuring the Reproducible Builds logo. The main content area has a light gray background. On the left, there's a sidebar with a dark blue header containing the Reproducible Builds logo and the word "Reproducible". Below this, the sidebar lists several menu items: Home, Contribute, Documentation (which is highlighted in blue), Tools, Who is involved?, News, Events, and Talks. The main content area starts with a large heading "Definitions" in bold black font. Below it is another heading "When is a build reproducible?" in bold black font. To the right of this heading is a block of text explaining the definition of a reproducible build. At the bottom of the main content area is another block of text providing more context about the relevant attributes of the build environment.

## 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’

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	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	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	.	.	.	.	.	00002d0:	0000	0000	0000	b418	c252	0000	0000	.	.	.	.	
00002e0:	2093	f53f	c3d8	4290	8392	f53f	dd07	20b5	.	.	.	.	00002e0:	2c65	19e2	5817	f63f	2c65	19e2	5817	f63f	.	.	.	.
00002f0:	8993	f53f	6d73	637a	c292	f53f	b81e	85eb	.	.	.	.	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	0000	.	.	.	.
0000310:	3485	ce6b	ec92	f53f	df37	bef6	cc92	f53f	4..k..?.	.	.	.	0000310:	f018	c252	0000	bbb8	8d06	f016	f63f	.	.	.	.	
0000320:	6c43	c538	7f93	f53f	170e	8464	0193	f53f	1C..8..?	.	.	.	0000320:	2c65	19e2	5817	f63f	bbb8	8d06	f016	f63f	.	.	.	.
0000330:	85eb	51b8	1e45	3040	a7e1	c552	0000	0000	.	.	.	.	0000330:	bbb8	8d06	f016	f63f	0900	0000	0000	0000	.	.	.	.
0000340:	dde1	c552	89d2	dee0	0b93	f53f	df4f	8d97	.	.	.	.	0000340:	f018	c252	0000	0000	2c19	c252	0000	0000	.	.	.	.
0000350:	6e92	f53f	dd07	20b5	8993	f53f	c3d8	4290	n..?..	.	.	.	0000350:	bbb8	8d06	f016	f63f	2c65	19e2	5817	f63f	.	.	.	.
0000360:	8392	f53f	5c8f	c2f5	285c	1140	dde1	c552	..?\\..(\\..	.	.	.	0000360:	bbb8	8d06	f016	f63f	2c65	19e2	5817	f63f	.	.	.	.
0000370:	0000	0000	1fe2	c552	c3d8	4290	8392	f53f	....R..B	.	.	.	0000370:	0800	0000	0000	f118	c252	0000	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 dark header with a light orange background below it. The header includes the Sigstore logo, a navigation bar with links to Overview, Community, How sigstore works, Trust and security, Blog, Docs, and GitHub. Below the header, the text "sign. verify. protect." is prominently displayed in large, bold, dark letters. Underneath, a subtitle reads "Making sure your software is what it claims to be." At the bottom, there's a section titled "In collaboration with" featuring logos for OpenSSF, Chainguard, Cisco, Google, Hewlett Packard Enterprise, PURDUE UNIVERSITY, Red Hat, and VMware.

0101  
0101

# Signing artifacts

The screenshot shows a web browser window with the URL <https://www.sigstore.dev>. The page title is "How sigstore works". The content includes:

- A standardized approach**: Describes how open source software uploaded for distribution has a stricter, more standardized way of checking who's been involved, ensuring no risk of key compromise.
- Building for future integrations**: Mentions a working partnership with Google, the Linux Foundation, Red Hat, and Purdue University to improve the technology.

To the right of the text is a flow diagram illustrating the sigstore workflow:

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

The diagram shows "DEVELOPERS, MAINTAINERS, MONITORS" at the top, connected by dotted lines to three circular nodes: "SIGN AND PUBLISH ARTIFACTS", "PUBLISH SIGNING CERTIFICATES", and "MONITOR LOGS". Each of these three nodes is connected by a dotted line to a rectangular node below it: "FULCIO CERTIFICATE AUTHORITY", "SIGNATURE TRANSPARENCY LOG", and "KEY TRANSPARENCY LOG". Finally, all three rectangular nodes are connected by dotted lines to a bottom rectangular node 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

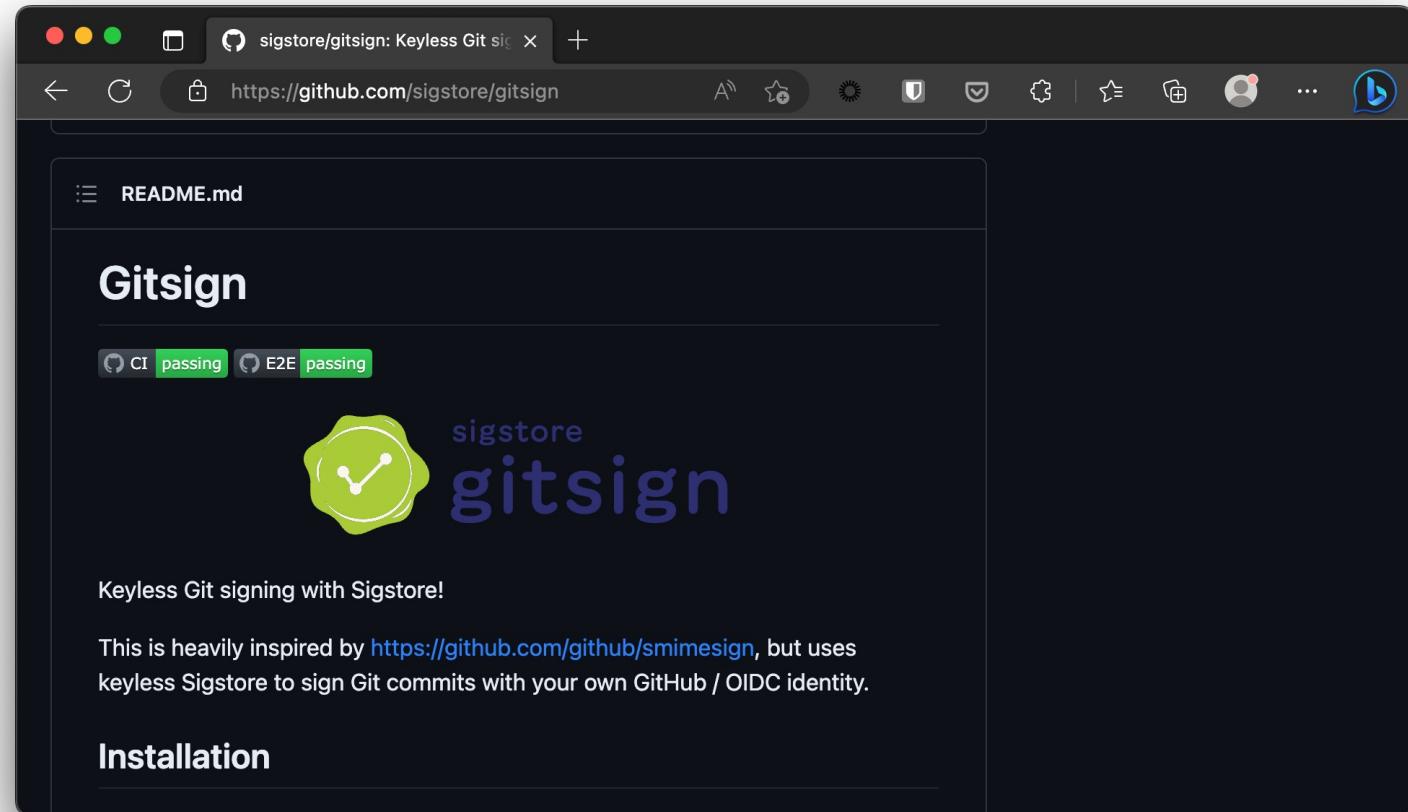
0101  
0101

# Cosign Keyless Signing



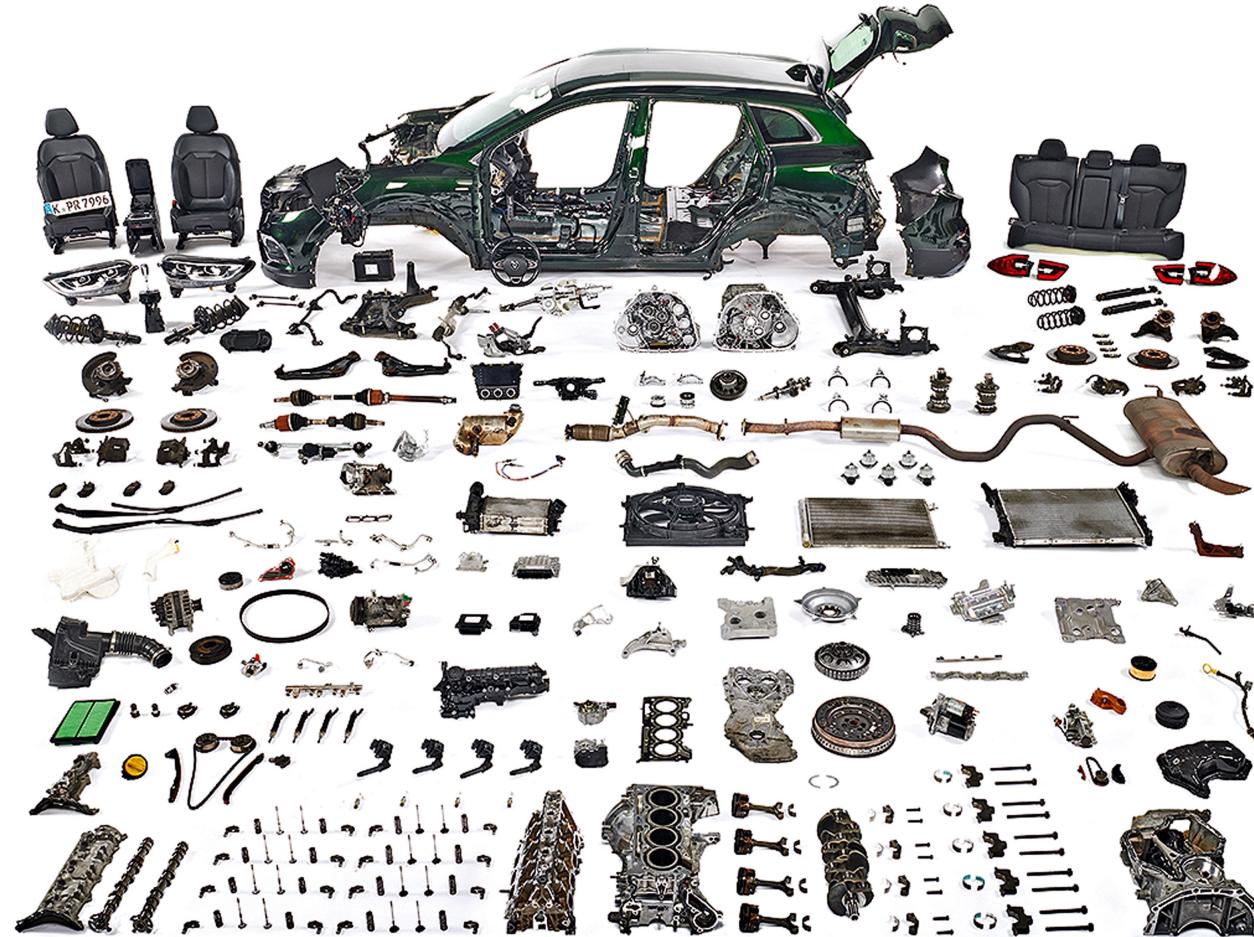
0101  
0101

# Git Commit Signing Sigstore GitSign



0101  
0101

# Automotive Industry



# Car Supply Chain

0101  
0101



## 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

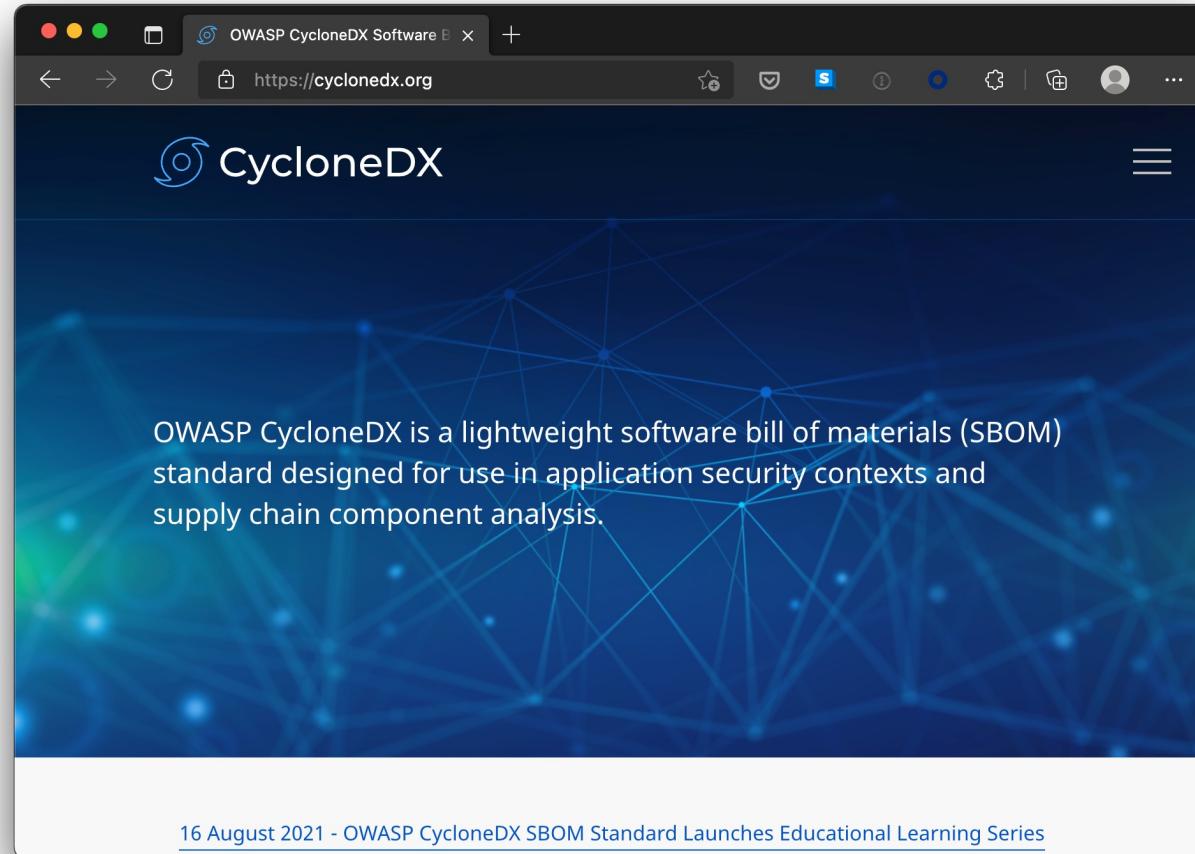


# Software Bill of Materials (SBOM)

- 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?

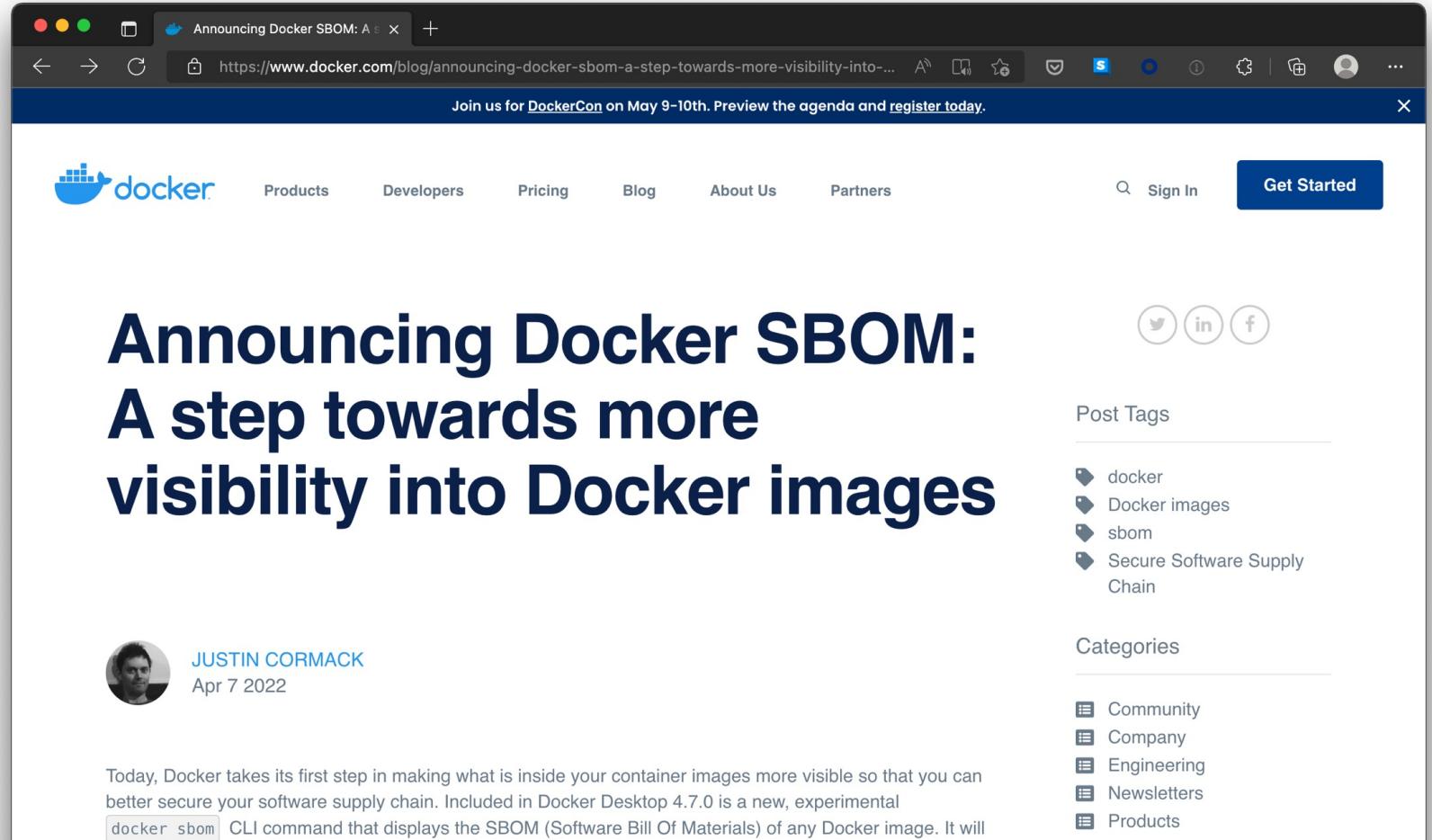
0101  
0101

# Software Bill of Materials (SBOM)



0101  
0101

# Docker SBOM



A screenshot of 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 post is authored by Justin Cormack and published on April 7, 2022. The Docker logo is visible in the top left corner of the page. The right side of the screen shows a sidebar with social media sharing icons (Twitter, LinkedIn, Facebook) and sections for "Post Tags" and "Categories". The "Post Tags" section includes tags like docker, Docker images, sbom, and Secure Software Supply Chain. The "Categories" section includes Community, Company, Engineering, Newsletters, and Products.

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

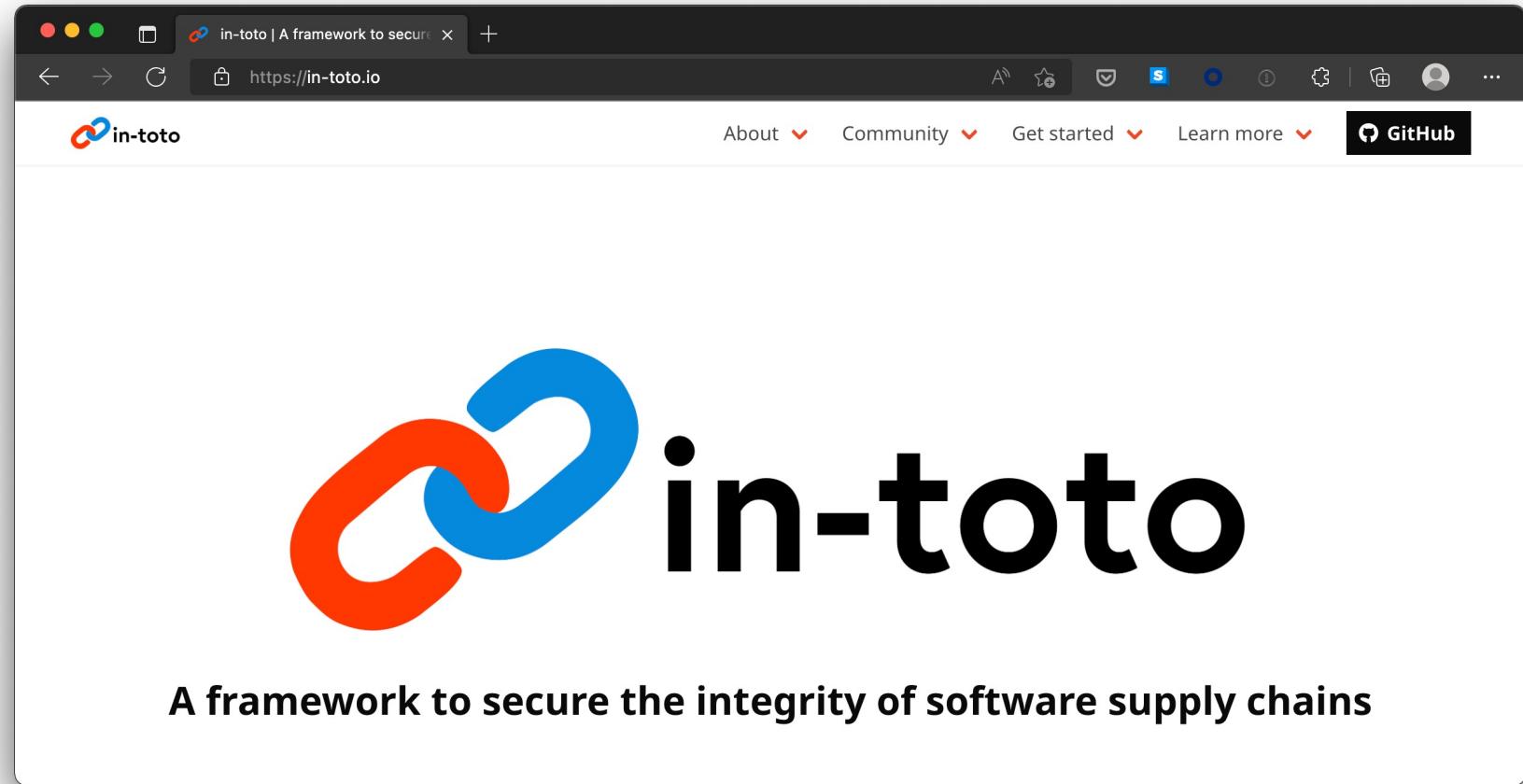
# Microsoft SBOM Tool

0101  
0101

The screenshot shows a dark-themed GitHub README page for the 'microsoft/sbom-tool' repository. The title 'SBOM Tool' is prominently displayed. Below it, there are build status badges for 'Build: passing', 'downloads 13k', and 'release v0.2.2'. A 'Languages' section indicates C# at 99.5% and PowerShell at 0.5%. The 'Introduction' section states: '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' includes links for 'Download and Installation', 'Run the tool', 'Integrating SBOM tool to your CI/CD pipelines', 'Telemetry', 'Contributing', 'Security', and 'Trademarks'. The 'Download and Installation' section points to 'Executables for Windows, Linux, macOS'.

# In-toto

0101  
0101



0101  
0101

# 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

# Google SLSA

0101  
0101

The screenshot shows a web browser window displaying the SLSA (Supply-chain Levels for Software Artifacts) website at <https://slsa.dev>. The page has a dark teal background with a large white text area in the center containing the slogan: "Safeguarding artifact integrity across any software supply chain". On the left side, there is a vertical navigation menu with the following items:

- SLSA
- Overview
- Understanding SLSA
  - What's new in v1.0
  - About SLSA
  - Supply chain threats
  - Use cases
  - Guiding principles
  - FAQ
  - Future directions
- Core specification
  - Terminology
  - Security levels



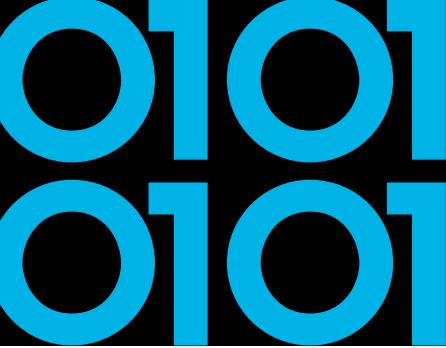
# Google SLSA Levels

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 following details:

- Title Bar:** SLSA • General availability of SLSA3 Generic Generator for GitHub Actions
- Address Bar:** https://slsa.dev/blog/2022/08/slsa-github-workflows-generic-ga
- Header:** SLSA (with logo), Overview, Specifications, Use cases, Get started, Community, Blog, GitHub icon.
- Main Content:**

# General availability of SLSA3 Generic Generator for GitHub Actions

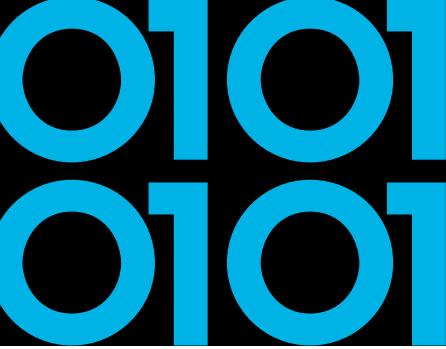
by Ian Lewis, Laurent Simon, Asra Ali  
29 Aug 2022

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.

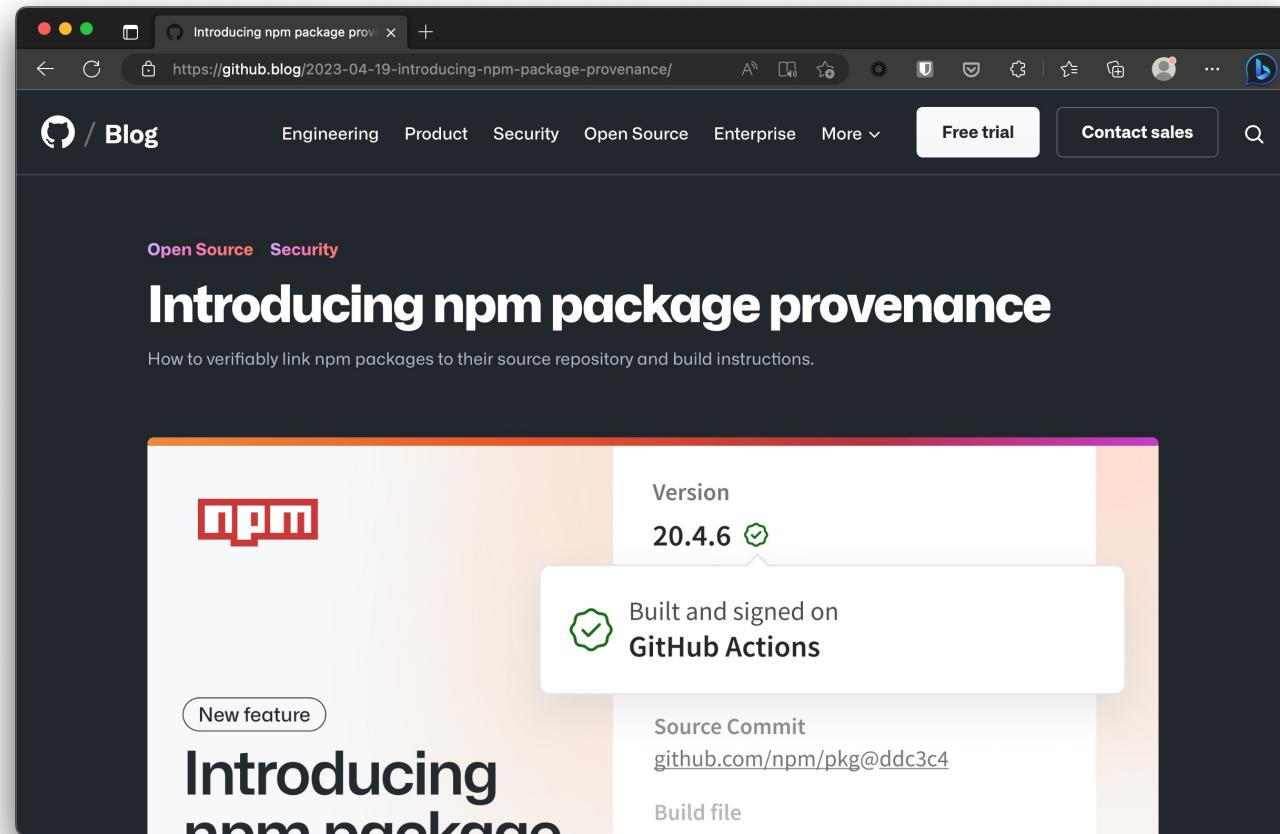


# MyAwesomePDFComponent Demo

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

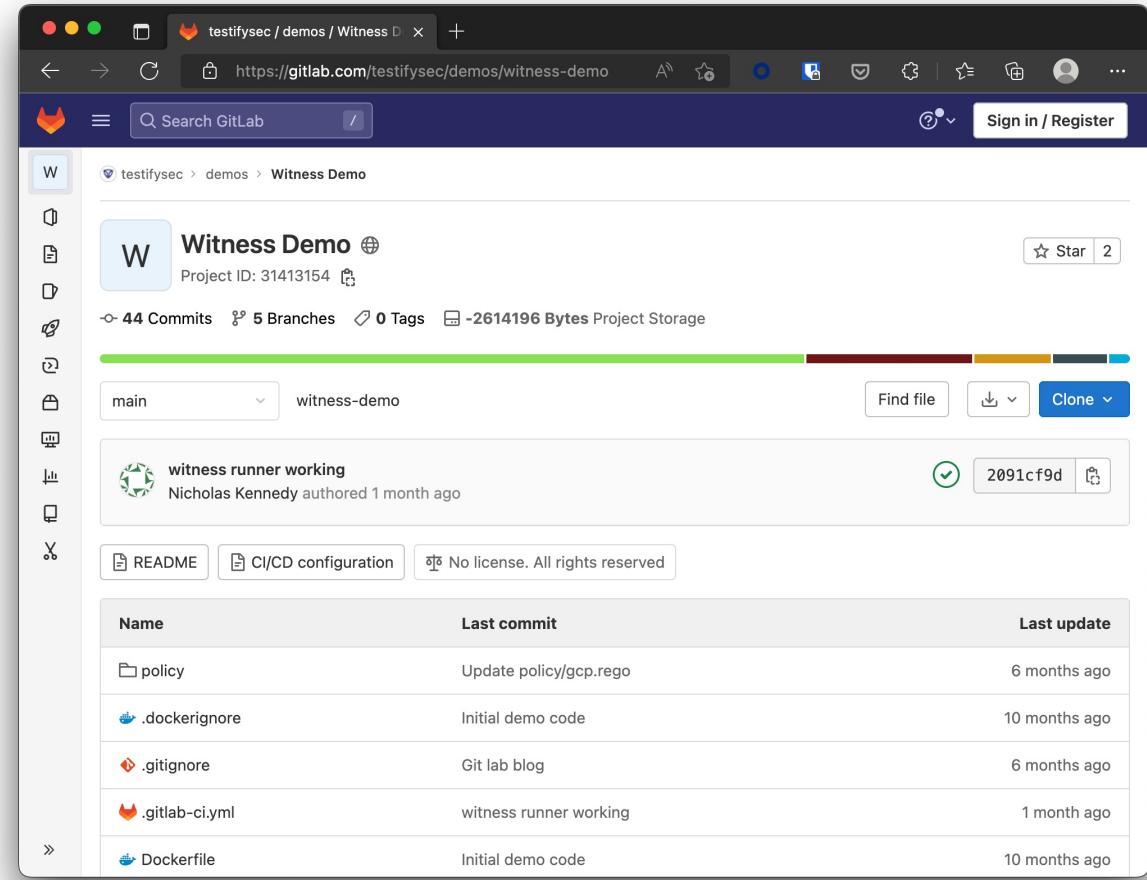


# NPM Package SLSA Provenance



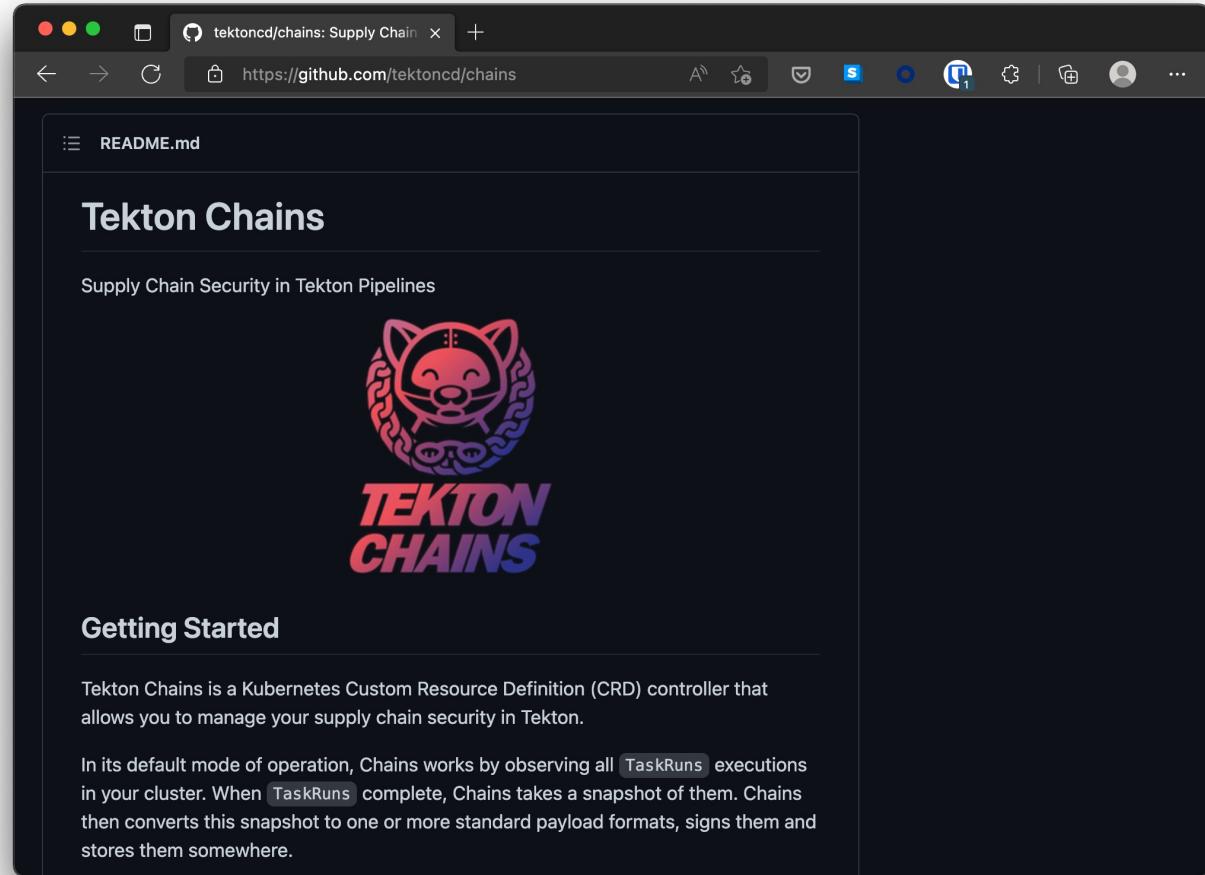
0101  
0101

# Witness & GitLab Attestator



0101  
0101

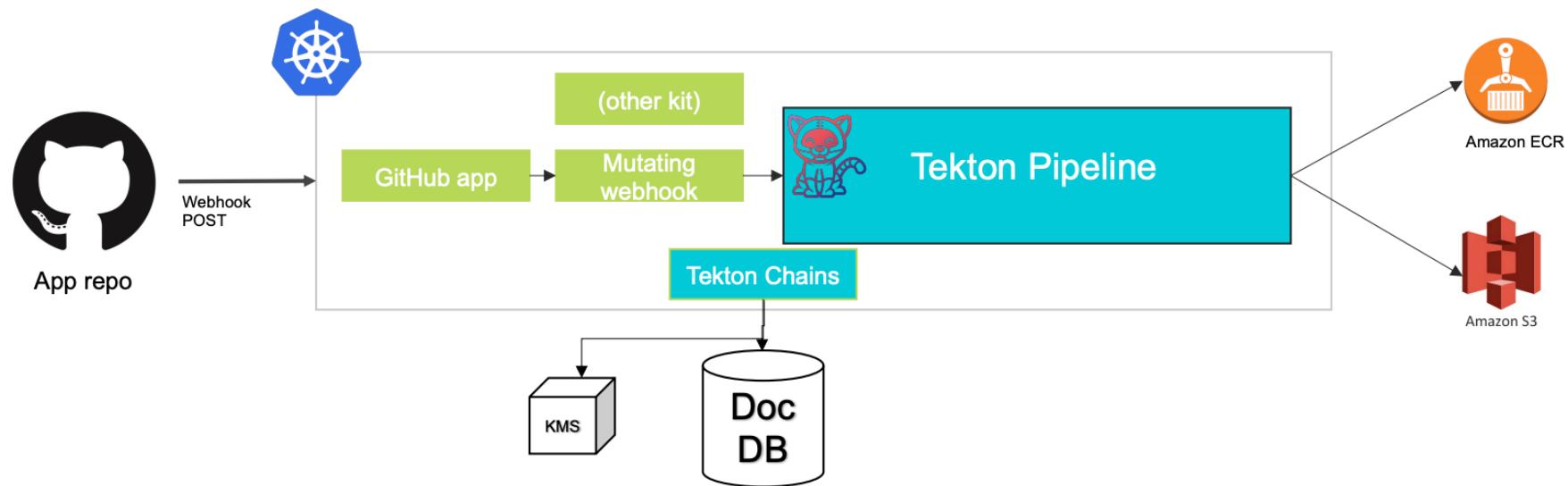
# Open Shift - Tekton - Tekton Chains



# SolarWinds Project Trebuchet



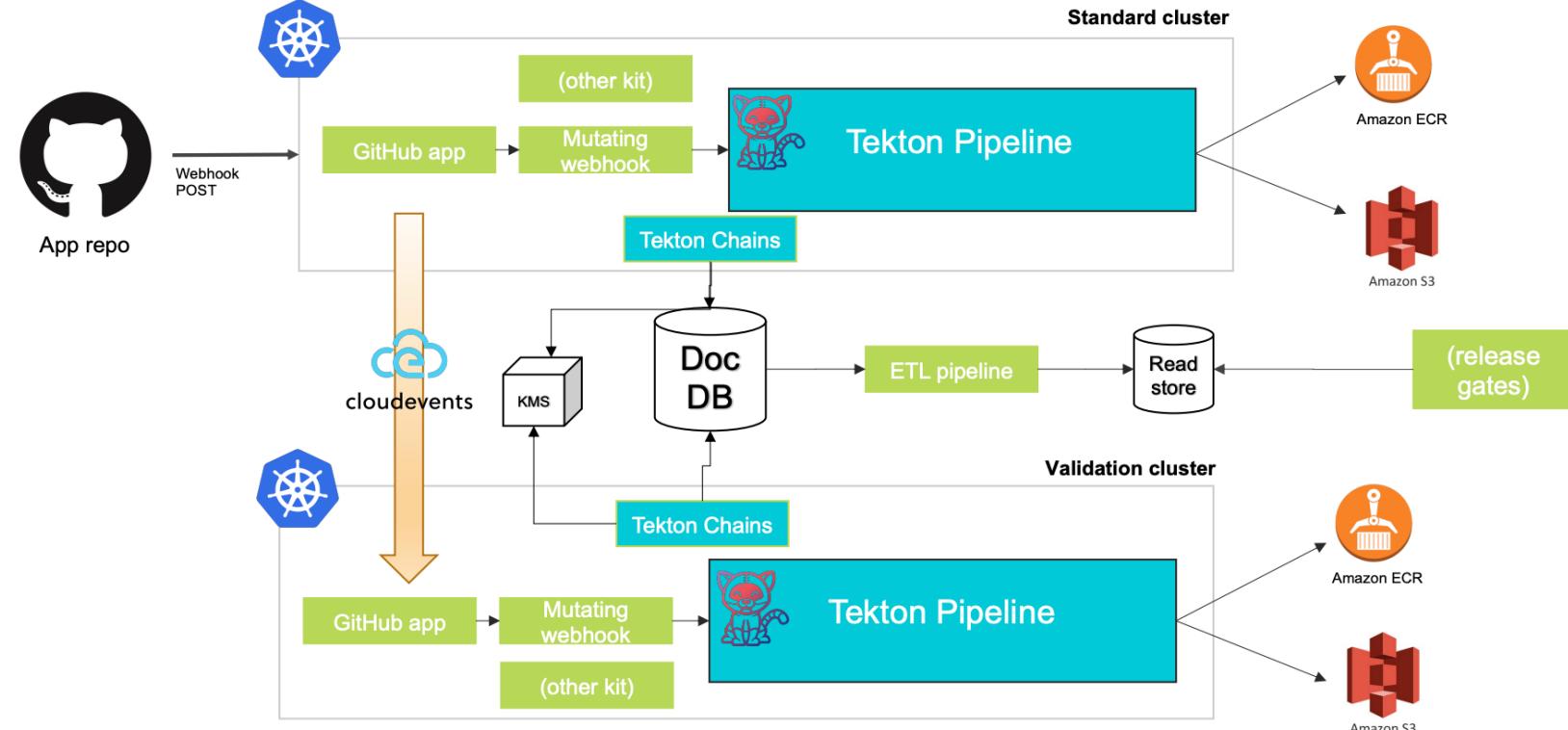
## Pipeline With Attestations



# SolarWinds Project Trebuchet

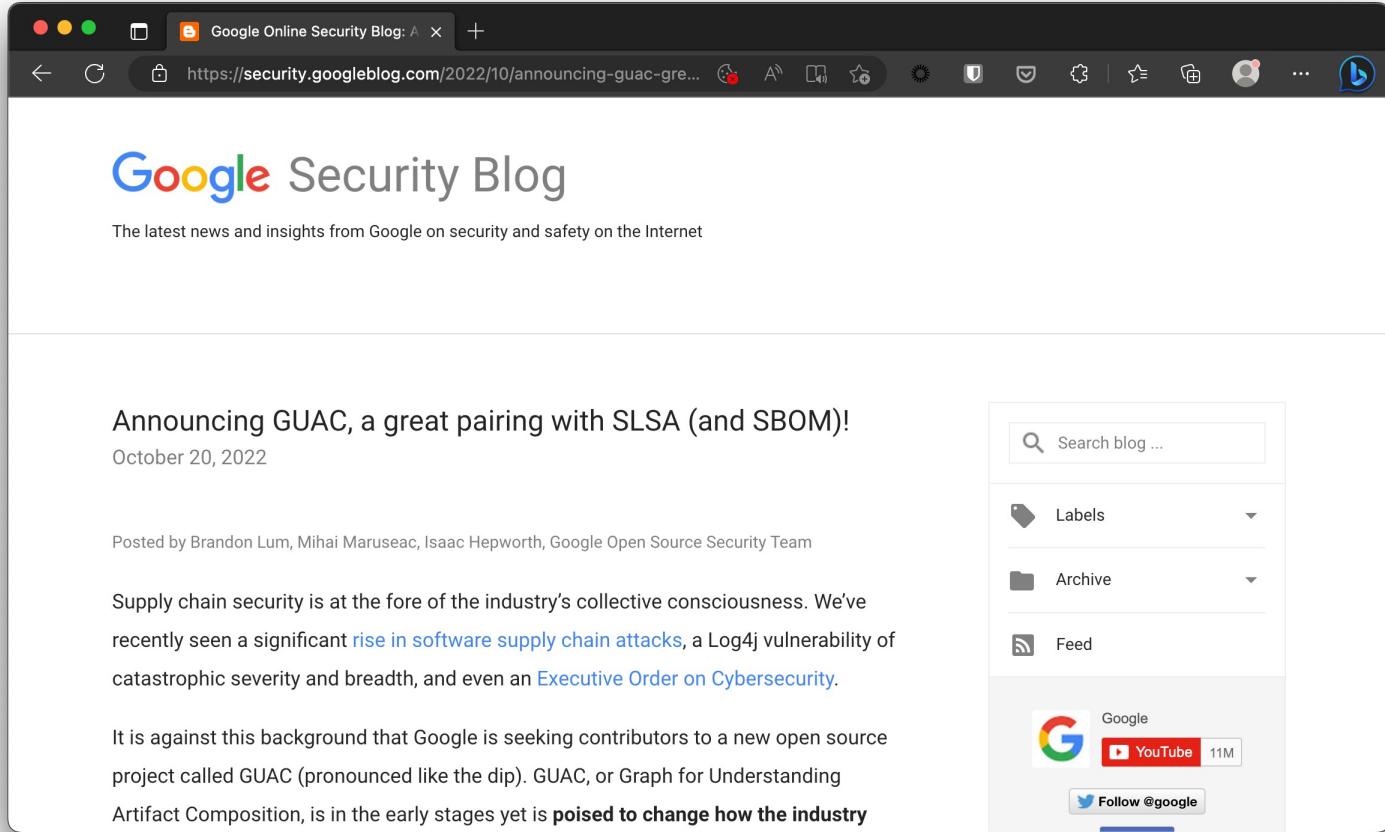


## Reading Results



# Google Graph for Understanding Artifact Composition (GUAC)

0101  
0101



The screenshot shows a web browser window displaying a blog post from the Google Online Security Blog. The title of the post is "Announcing GUAC, a great pairing with SLSA (and SBOM)!" and it was published on October 20, 2022. The post discusses supply chain security challenges and the introduction of GUAC. On the right side of the page, there is a sidebar with search, labels, archive, and feed options, along with social media links for Google, YouTube, and Twitter.

Google Online Security Blog: A × +

https://security.googleblog.com/2022/10/announcing-guac-gre...

Google Security Blog

The latest news and insights from Google on security and safety on the Internet

Announcing GUAC, a great pairing with SLSA (and SBOM)!

October 20, 2022

Posted by Brandon Lum, Mihai Maruseac, Isaac Hepworth, Google Open Source Security Team

Supply chain security is at the fore of the industry's collective consciousness. We've recently seen a significant [rise in software supply chain attacks](#), a Log4j vulnerability of catastrophic severity and breadth, and even an [Executive Order on Cybersecurity](#).

It is against this background that Google is seeking contributors to a new open source project called GUAC (pronounced like the dip). GUAC, or Graph for Understanding Artifact Composition, is in the early stages yet is **poised to change how the industry**

Search blog ...

Labels

Archive

Feed

Google YouTube 11M

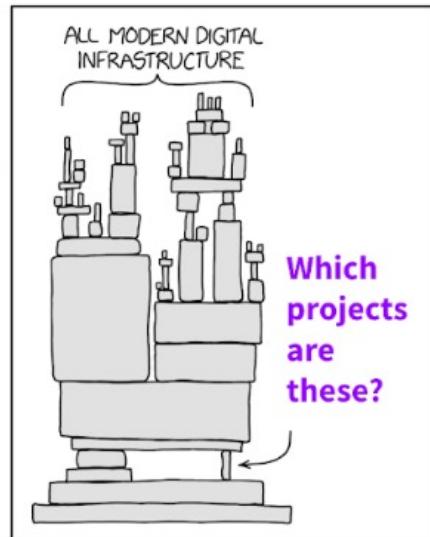
Follow @google

# Graph for Understanding Artifact Composition

0101  
0101

## Proactive

How do I prevent large scale supply chain compromises?



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

## Preventive

Have I taken the right safeguards?

When deciding to use and deploy software, are there sufficient security checks and approvals?



## Reactive

HOW AM I AFFECTED???

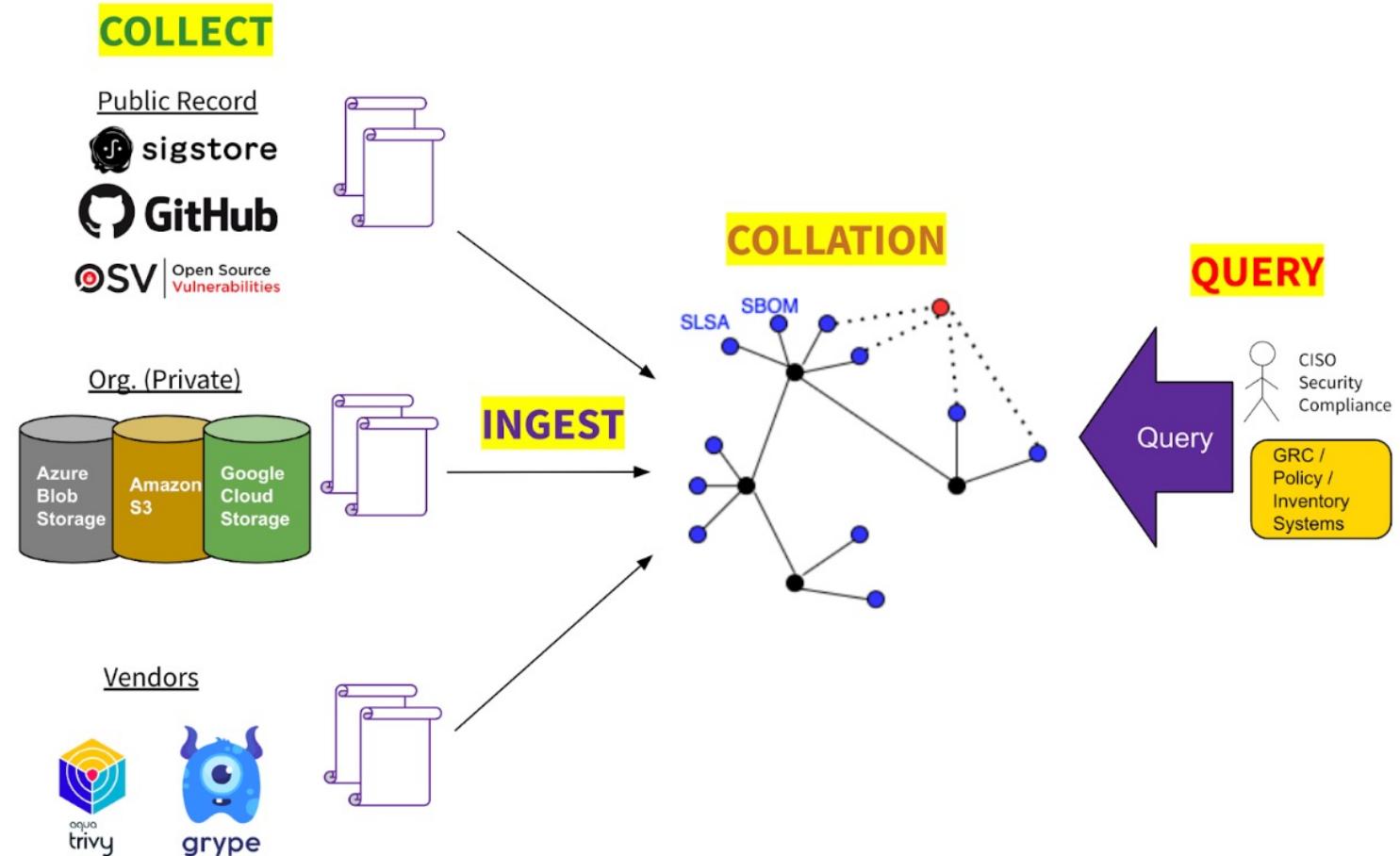
A vulnerability or supply chain compromise is discovered!



+ Codecov, Solarwinds compromises

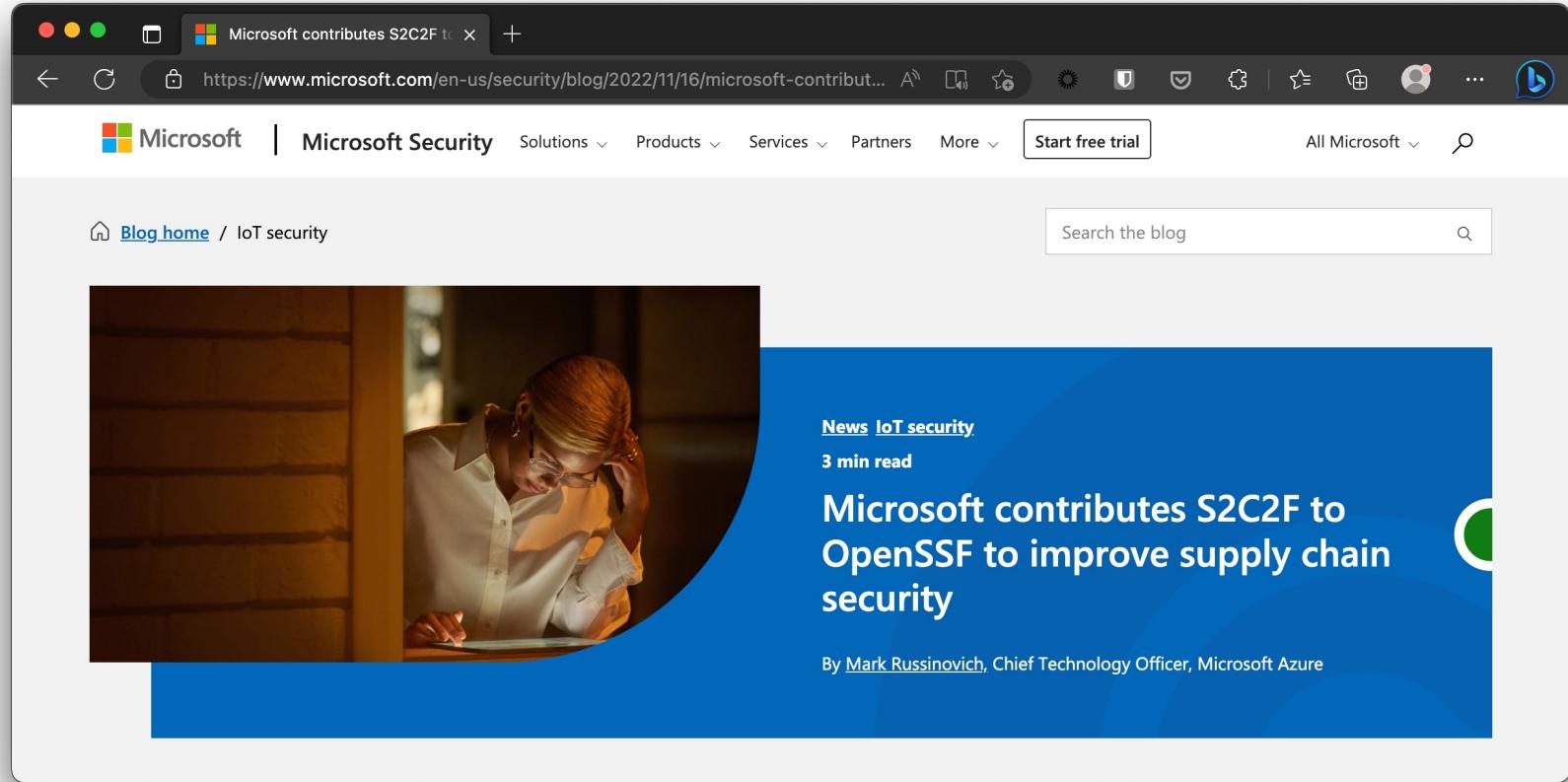
# Graph for Understanding Artifact Composition

0101  
0101



# Secure Supply Chain Consumption Framework (S2C2F)

0101  
0101



# Secure Supply Chain Consumption Framework (S2C2F)

0101  
0101

- Provide a strong OSS governance program
- Improve the Mean Time To Remediate (MTTR) for resolving known vulnerabilities in OSS
- Prevent the consumption of compromised and malicious OSS packages



# Secure Supply Chain Consumption Framework (S2C2F)



Level 1	Level 2	Level 3	Level 4
<b>Minimum OSS Governance Program</b> <ul style="list-style-type: none"><li>Use package managers</li><li>Local copy of artifact</li><li>Scan with known vulns</li><li>Scan for software licenses</li><li>Inventory OSS</li><li>Manual OSS updates</li></ul>	<b>Secure Consumption and Improved MTTR</b> <ul style="list-style-type: none"><li>Scan for end life</li><li>Have an incident response plan</li><li>Auto OSS updates</li><li>Alert on vulns at PR time</li><li>Audit that consumption is through the approved ingestion method</li><li>Validate integrity of OSS</li><li>Secure package source file configuration</li></ul>	<b>Malware Defense and Zero-Day Detection</b> <ul style="list-style-type: none"><li>Deny list capability</li><li>Clone OSS source</li><li>Scan for malware</li><li>Proactive security reviews</li><li>Enforce OSS provenance</li><li>Enforce consumption from curated feed</li></ul>	<b>Advanced Threat Defense</b> <ul style="list-style-type: none"><li>Validate the SBOMs of OSS consumed</li><li>Rebuild OSS on trusted infrastructure</li><li>Digitally sign rebuilt OSS</li><li>Generate SBOM for rebuilt OSS</li><li>Digitally sign protected SBOMs</li><li>Implement fixes</li></ul>



# Conclusion

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



# Conclusion

- Start working on creating SBOM's and see how SLSA can fit into your process
- Look how S2C2F can help you on your projects where you consume open-source
- Work smart with SBOM/provenance outputs!

0101  
0101

# Questions?

- [https://github.com/nielstanis/  
codeeurope2023-supplychain](https://github.com/nielstanis/codeeurope2023-supplychain)
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://blog.fennec.dev>
- Dziękuję! Thank you!