



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
  - ISC<sup>2</sup> CSSLP
  - Research on static analysis for .NET apps



@nielstanis



# Securing your .NET application software supply-chain

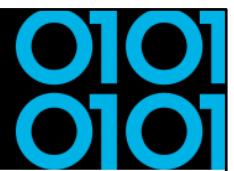
0101  
0101



@nielstanis

Picture is from Veracode report/site:

<https://www.veracode.com/sites/default/files/pdf/resources/ipapers/everything-you-need-to-know-open-source-risk/index.html>



## Agenda

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



@nielstanis

0101  
0101

## What is a Supply Chain?



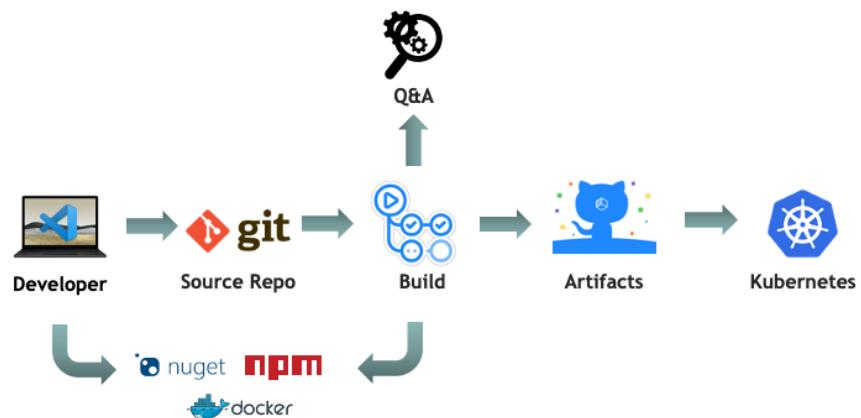
@nielstanis

Image source:

[https://www.wardsauto.com/sites/wardsauto.com/files/styles/article\\_featured\\_retina/public/Renault%20Kadjar%20assembly%20line%20-%20Palencia%20Spain-5\\_8.jpg?](https://www.wardsauto.com/sites/wardsauto.com/files/styles/article_featured_retina/public/Renault%20Kadjar%20assembly%20line%20-%20Palencia%20Spain-5_8.jpg?)

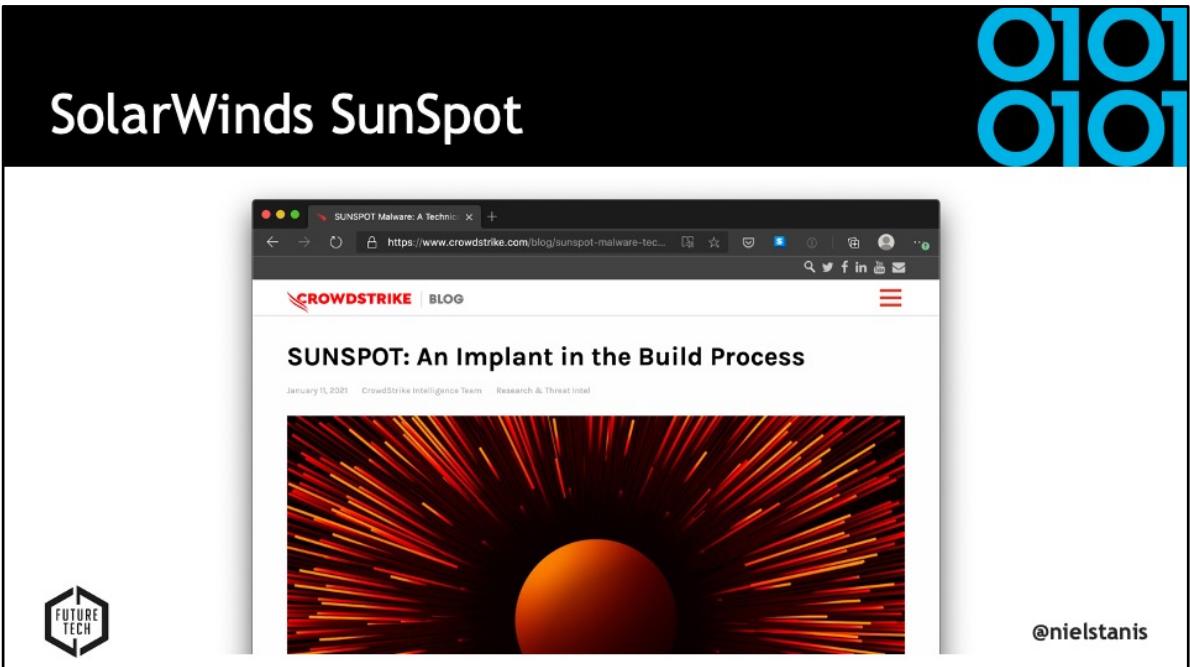
0101  
0101

## Software Supply Chain



@nielstanis



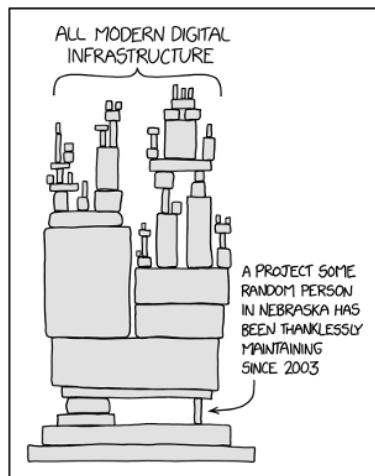


<https://www.crowdstrike.com/blog/sunspot-malware-technical-analysis/>

<https://www.microsoft.com/security/blog/2020/12/18/analyzing-solorigate-the-compromised-dll-file-that-started-a-sophisticated-cyberattack-and-how-microsoft-defender-helps-protect/>

0101  
0101

## XKDC - Dependency



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

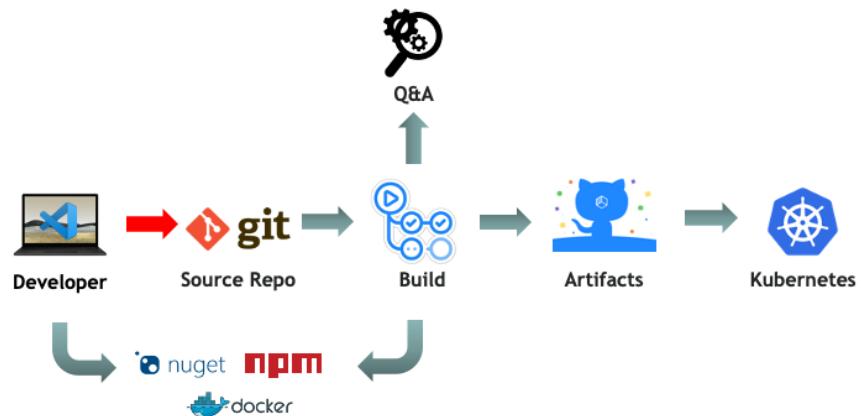
@nielstanis

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



0101  
0101

## Software Supply Chain



@nielstanis

# GitHub account

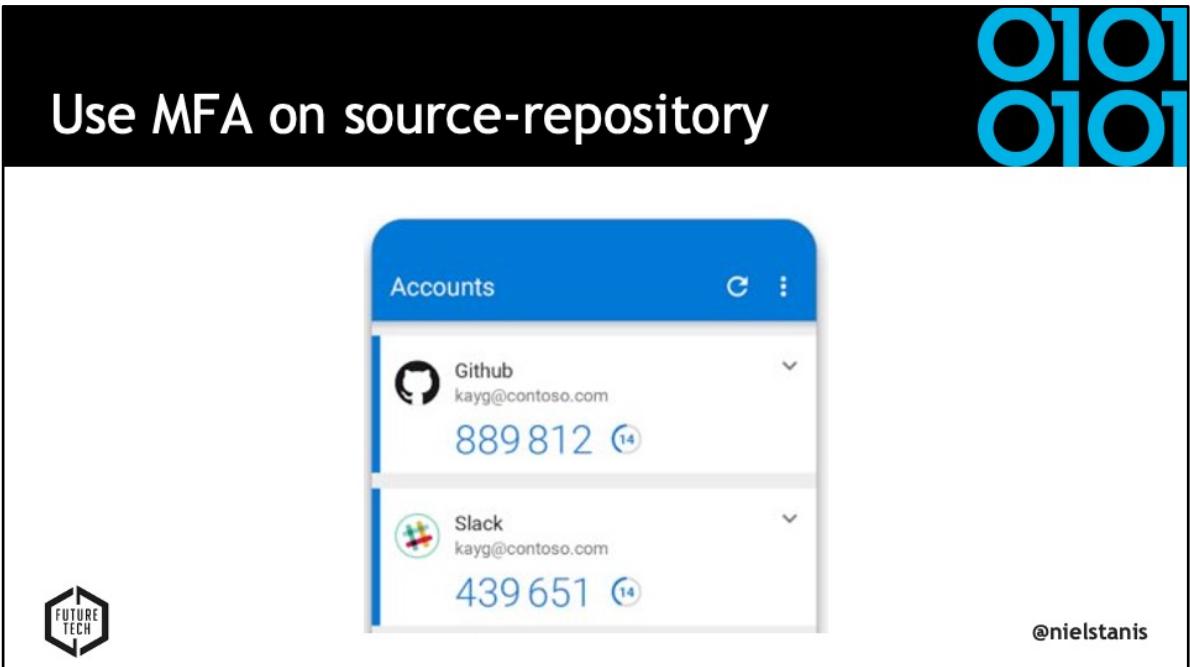
0101  
0101

The screenshot shows a news article from ZDNet. The title is "Canonical GitHub account hacked, Ubuntu source code safe". The subtext reads "Ubuntu source code appears to be safe; however Canonical is investigating." Below the title is a small image of a GitHub interface showing a pull request. The author is Catalin Cimpanu, and the date is July 7, 2019. There are social sharing icons for LinkedIn, Facebook, Twitter, and others. A sidebar on the right has a link to "Why everyone should have this cheap security tool". The ZDNet logo is in the top left corner, and the Future Tech logo is in the bottom left corner. The ZDNet URL is https://www.zdnet.com/article/canonical-github-account-hacked-ubuntu-source-code-safe/.

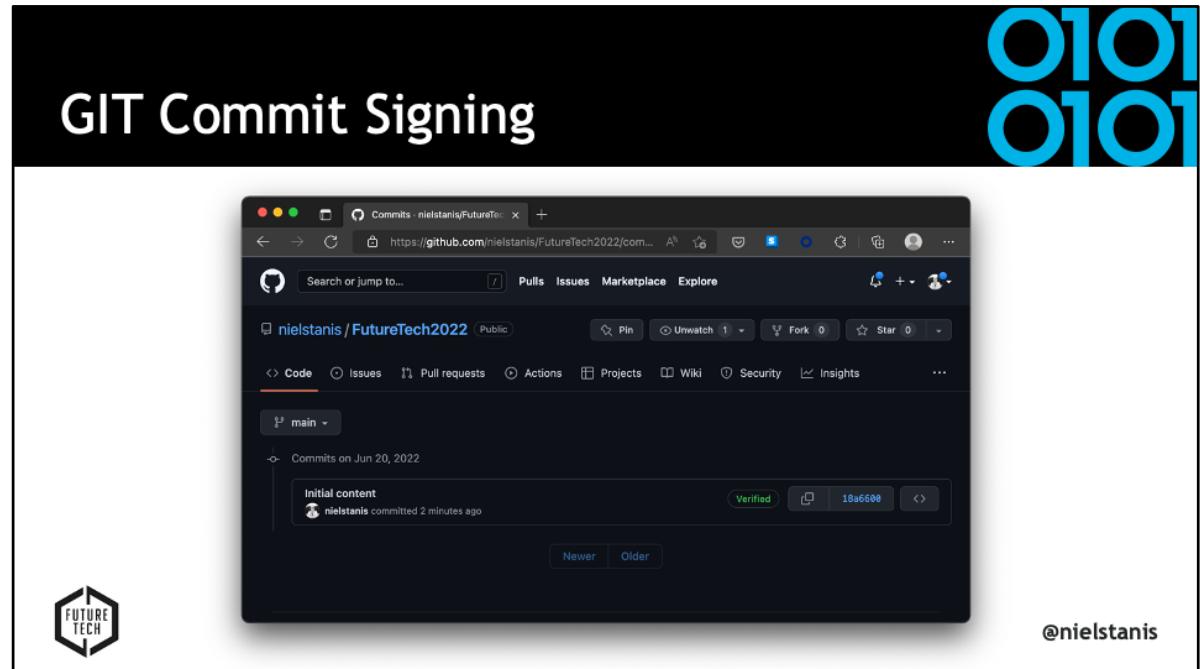
<https://www.zdnet.com/article/canonical-github-account-hacked-ubuntu-source-code-safe/>

0101  
0101

## Use MFA on source-repository



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



<https://www.hanselman.com/blog/HowToSetupSignedGitCommitsWithAYubiKeyNEOA ndGPGAndKeybaseOnWindows.aspx>

# Hypocrite Commits

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

Qiushi Wu and Kangjie Lu  
University of Minnesota  
{wu000273, kjlu}@umn.edu

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

<https://github.com/QiushiWu/qiushiwu.github.io/blob/main/papers/OpenSourceInsecurity.pdf>

# Octopus Scanner - NetBeans

0101  
0101

May 28, 2020

## The Octopus Scanner Malware: Attacking the open source supply chain

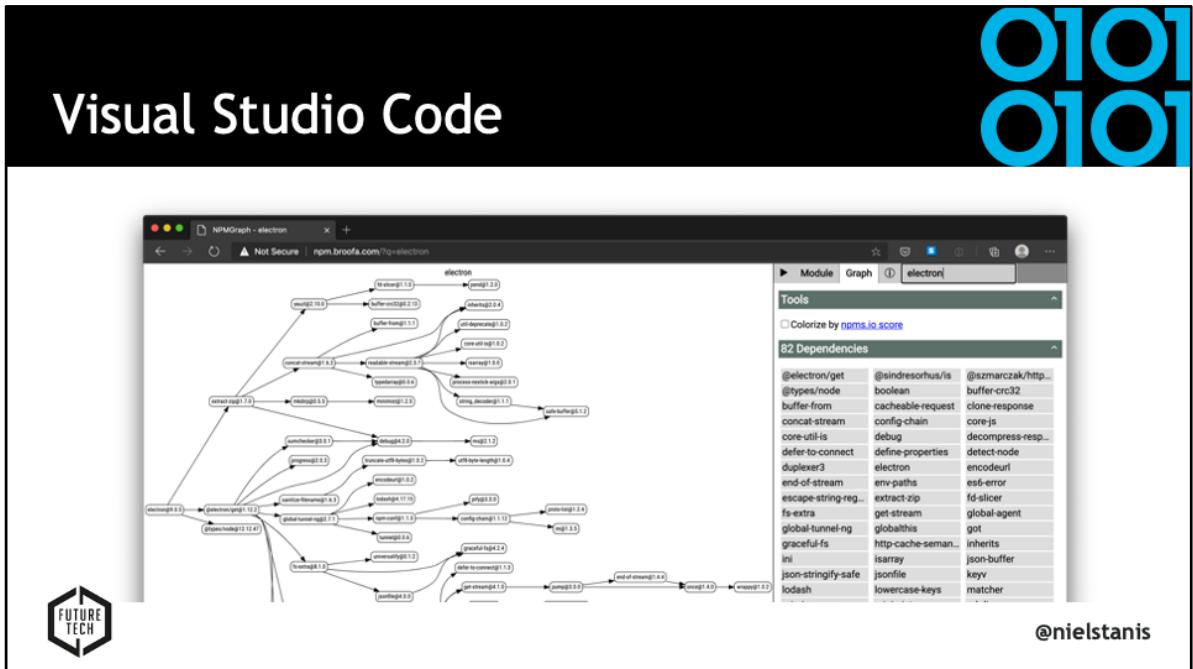
 Alvaro Muñoz

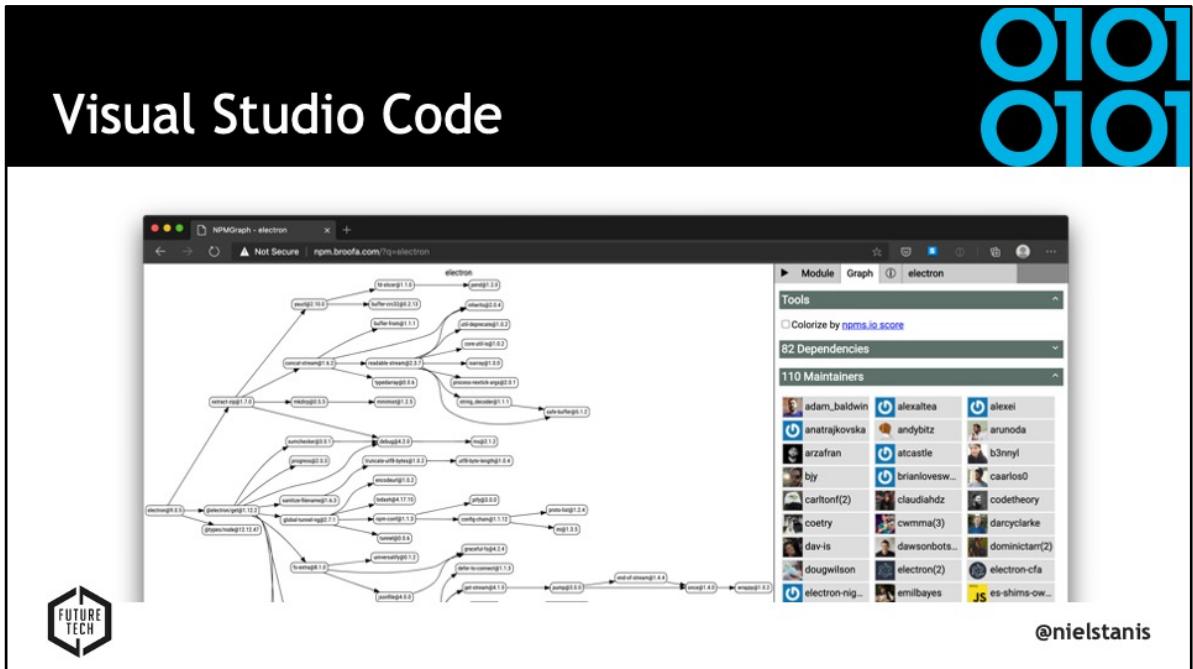
Securing the open source supply chain is an enormous task. It goes far beyond a security assessment or just patching for the latest CVEs. Supply chain security is about the integrity of the entire software development and delivery ecosystem. From the code commits themselves, to how they flow through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

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







0101  
0101

# Visual Studio Code

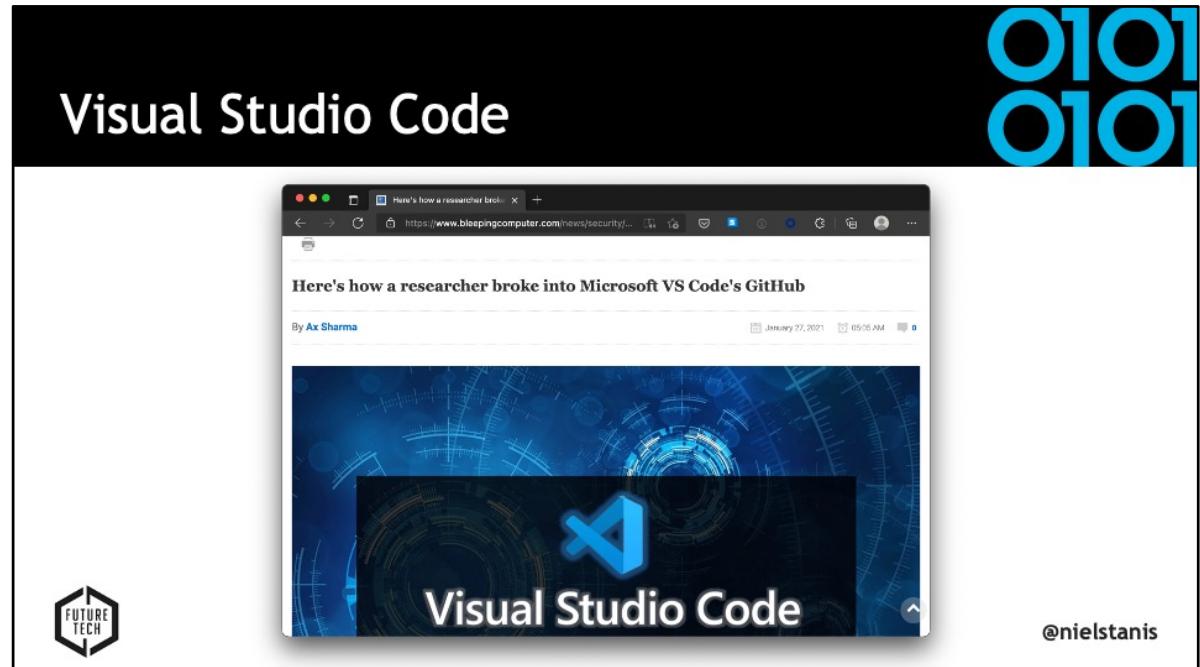
The screenshot shows a web browser displaying the Microsoft MSRC Security Update Guide for CVE-2022-30129. The page title is "CVE-2022-30129 - Security Update". The main content area is dark-themed and displays the following information:

- Visual Studio Code Remote Code Execution Vulnerability**
- CVE-2022-30129**
- On this page** dropdown menu
- Security Vulnerability**
- Released: May 10, 2022**
- Assigning CNA:** Microsoft
- MitreCVE-2022-30129**
- CVSS 3.1 8.8 / 7.7**

At the bottom right of the screenshot, there is a watermark-like text: **@nielstanis**.

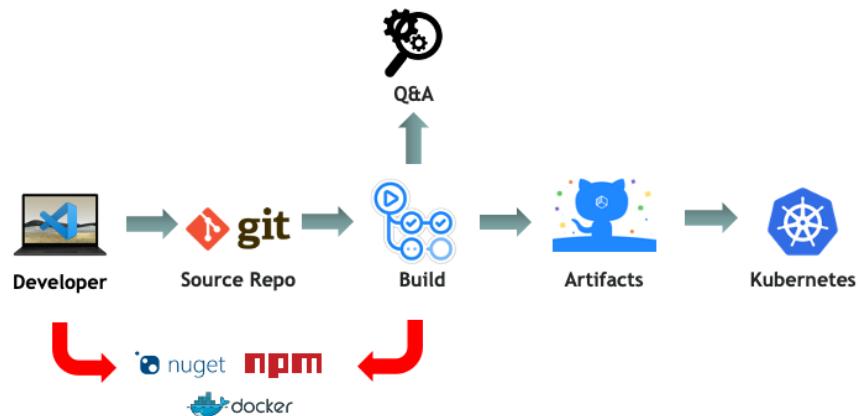
<https://msrc.microsoft.com/update-guide/vulnerability/CVE-2022-30129>





0101  
0101

## 3rd Party Libraries



@nielstanis



# 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

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

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. The "Executive summary" section states: "Microsoft is releasing this security advisory to provide information about a vulnerability in .NET 6.0, .NET 5.0, and .NET Core 3.1. This advisory also provides guidance on what developers can do to update their applications to remove this vulnerability. A Remote Code Execution vulnerability exists in .NET 6.0, .NET 5.0, and .NET Core 3.1 where a stack buffer overrun occurs in .NET Double Parse routine." Labels include `Monthly-Update`, `.NET Core 3.1`, `.NET 5.0`, `.NET 6.0`, `Patch-Tuesday`, and `Security`. The "Discussion" section links to issue #66348.

@nielstanis

<https://github.com/dotnet/announcements/issues/213>





**CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY**

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**

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

0101  
0101

## Vulnerabilities in libraries

The image shows two side-by-side screenshots of GitHub blog posts. The left screenshot is titled "GitHub's commitment to npm ecosystem security" by Mike Hanley on November 15, 2021. It features a blue and yellow illustration of a person working on a shield with a keyhole. The right screenshot is titled "Enrolling all npm publishers in enhanced login verification and next steps for two-factor authentication enforcement" by GitHub on December 7, 2021. It features the GitHub logo and the npm logo. A watermark "@nielstanis" is visible in the bottom right corner of the image.

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

<https://github.blog/2021-12-07-enrolling-npm-publishers-enhanced-login-verification-two-factor-authentication-enforcement/>

# Dependency Confusion

The screenshot shows a web browser window displaying a Medium article. The title of the article is "Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies". Below the title, it says "The Story of a Novel Supply Chain Attack". The author is listed as "Alex Birsan" with a small profile picture. The publication date is "Feb 9 · 11 min read". To the right of the article, there are social sharing icons for LinkedIn, Facebook, Twitter, and Email. Below the article, there is a large image of a stack of colorful shipping containers against a clear blue sky. In the bottom right corner of the slide, there is a small watermark or handle that reads "@nielstanis".

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

# 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>



@nielstanis

<https://azure.microsoft.com/nl-nl/resources/3-ways-to-mitigate-risk-using-private-package-feeds/>

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



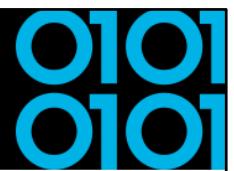
## Dependency Confusion - NuGet

- 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



@nielstanis

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



## 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



@nielstanis

0101  
0101

# Security Scorecards - OpenSSF



<https://github.com/ossf/scorecard>

The screenshot shows the Deps.Dev by Google website. At the top, there's a large blue '0101' logo. Below it, the text 'Deps.Dev by Google' is displayed. The main content area has a dark background with teal accents. It features a search bar with placeholder text 'Search for open source packages', a dropdown menu set to 'All systems', and a blue 'Search' button. To the right of the search bar, there's a list of dependency statistics:

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

At the bottom left is a hexagonal logo with 'FUTURE TECH' text. On the right side, there's a handle '@nielstanis'.

<https://deps.dev/>




**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**

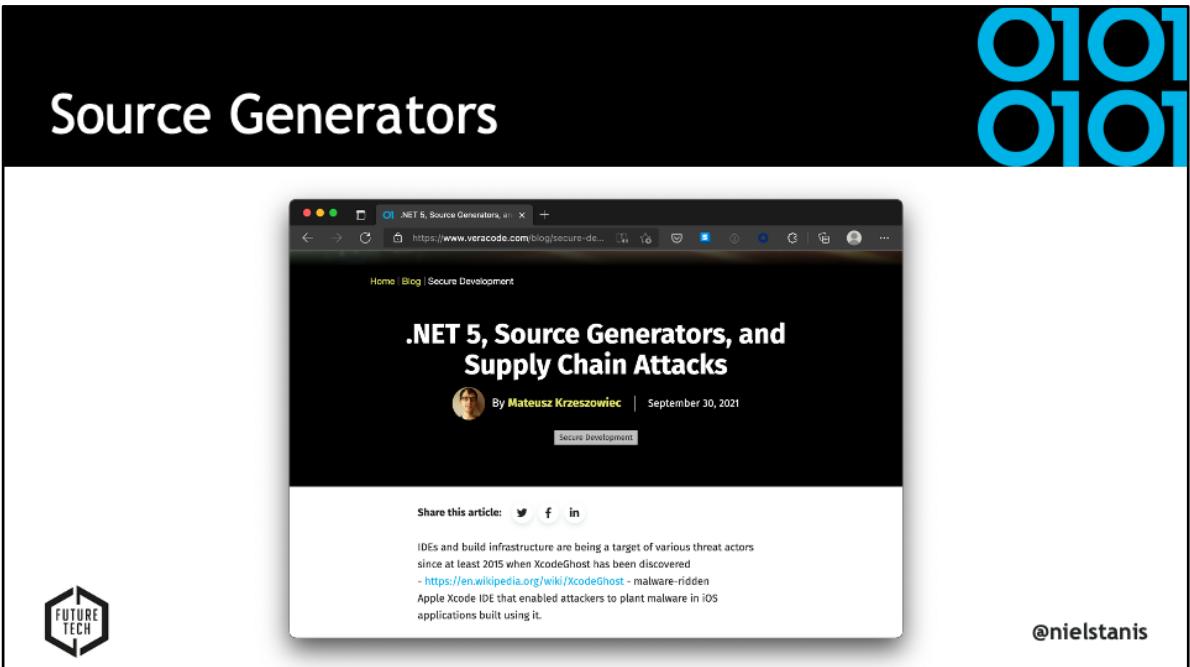
Scorecard as of April 25, 2022.

➤ Code-Review	5/10
➤ Maintained	10/10
➤ CI-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

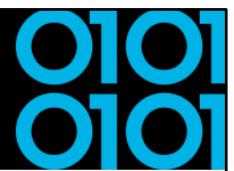
Project metadata as of May 7, 2022.

**@nielstanis**

<https://deps.dev/npm/electron>



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



## Source Generators

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

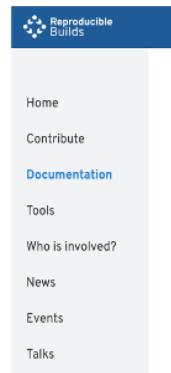
```
<Target Name="DisableAnalyzers"
        BeforeTargets="CoreCompile">
    <ItemGroup>
        <Analyzer Remove="@({Analyzer})" />
    </ItemGroup>
</Target>
```



@nielstanis

# Reproducible/Deterministic Builds

0101  
0101



## 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

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



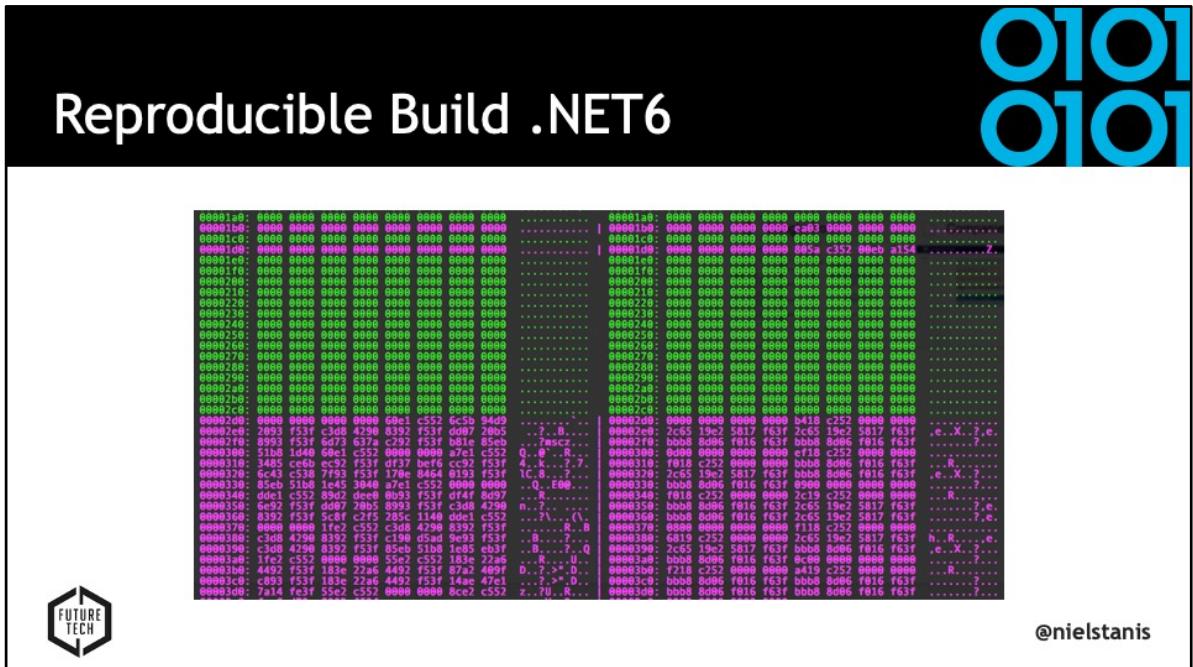
## Reproducible/Deterministic Builds

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



@nielstanis

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



<https://www.tabsoverspaces.com/233662-changing-paths-in-pdb-files-for-source-files-and-pdb-file-path-in-dll-as-well>  
<DebugOutput> in CSPROJ demo



## Reproducible/Deterministic Builds

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



@nielstanis

<https://blog.paranoidcoding.com/2016/04/05/deterministic-builds-in-roslyn.html>

<https://github.com/dotnet/roslyn/blob/master/docs/compilers/Deterministic%20Inputs.md>

<https://devblogs.microsoft.com/dotnet/producing-packages-with-source-link/>

<https://github.com/dotnet/reproducible-builds>



## 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`

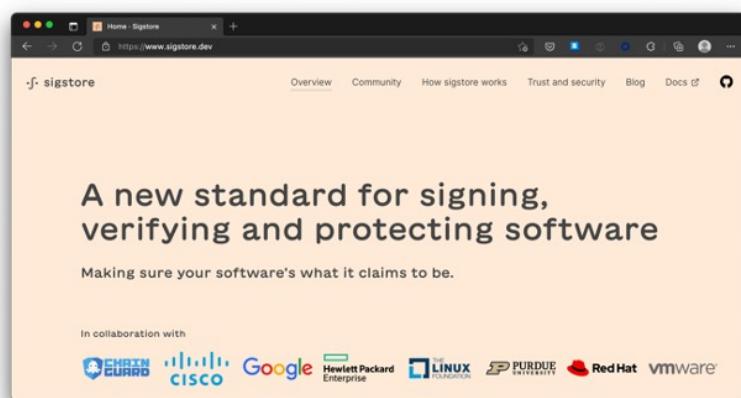


@nielstanis

<https://github.com/dotnet/designs/blob/main/accepted/2020/reproducible-builds.md>

# Signing artifacts

0101  
0101



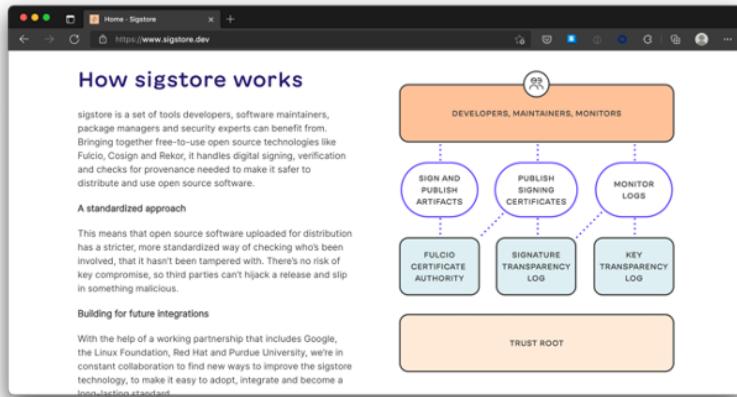
@nielstanis

<https://sigstore.dev>



0101  
0101

# Signing artifacts



@nielstanis

<https://sigstore.dev>





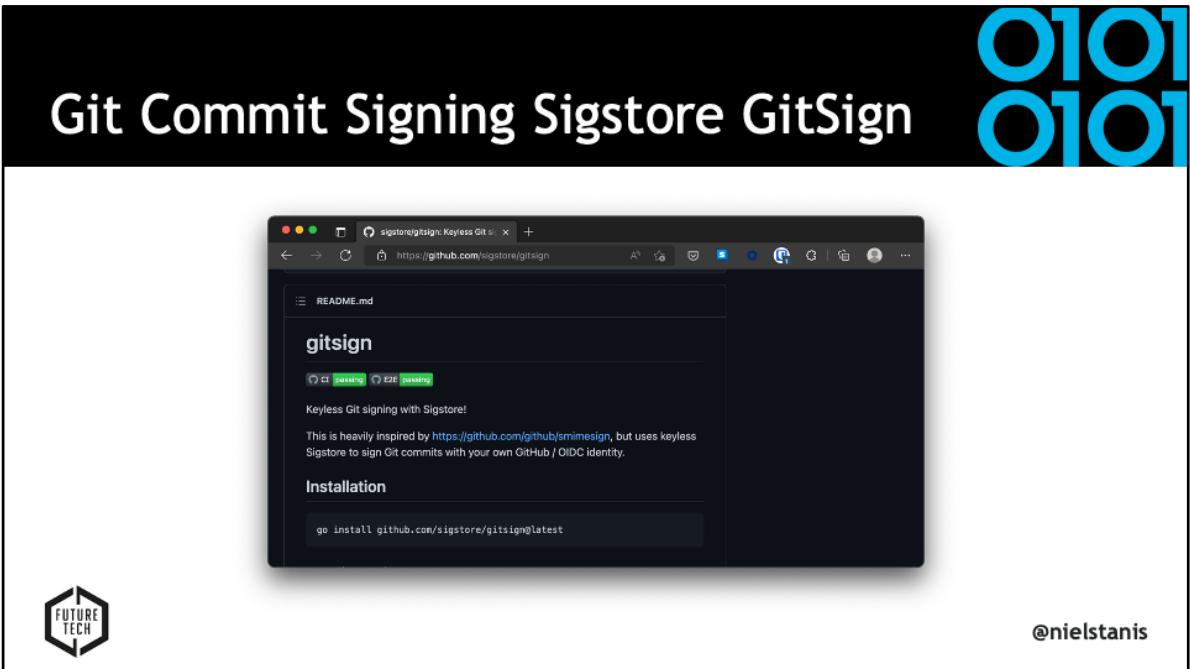
## Signing artifacts

- 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



@nielstanis

<https://sigstore.dev>



<https://sigstore.dev>

0101  
0101

## Automotive Industry



@nielstanis

0101  
0101

## Car Supply Chain



### Tata Steel Factory

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

### Bosch Factory

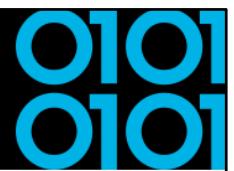
- Steel Batch #1234 Tata
- ECE-R90 Tested/Certified
  - Serie #45678
- Used by Ford, Volkswagen and Renault

### Renault Manufacturing

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



@nielstanis

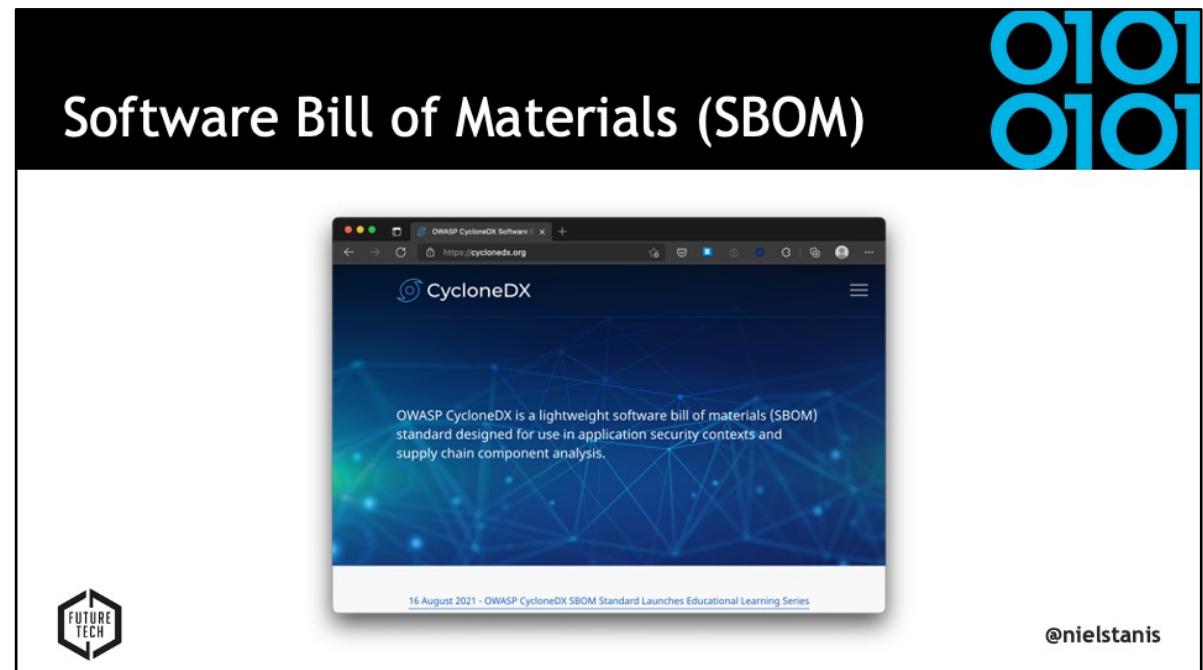


## Software Bill of Materials (SBOM)

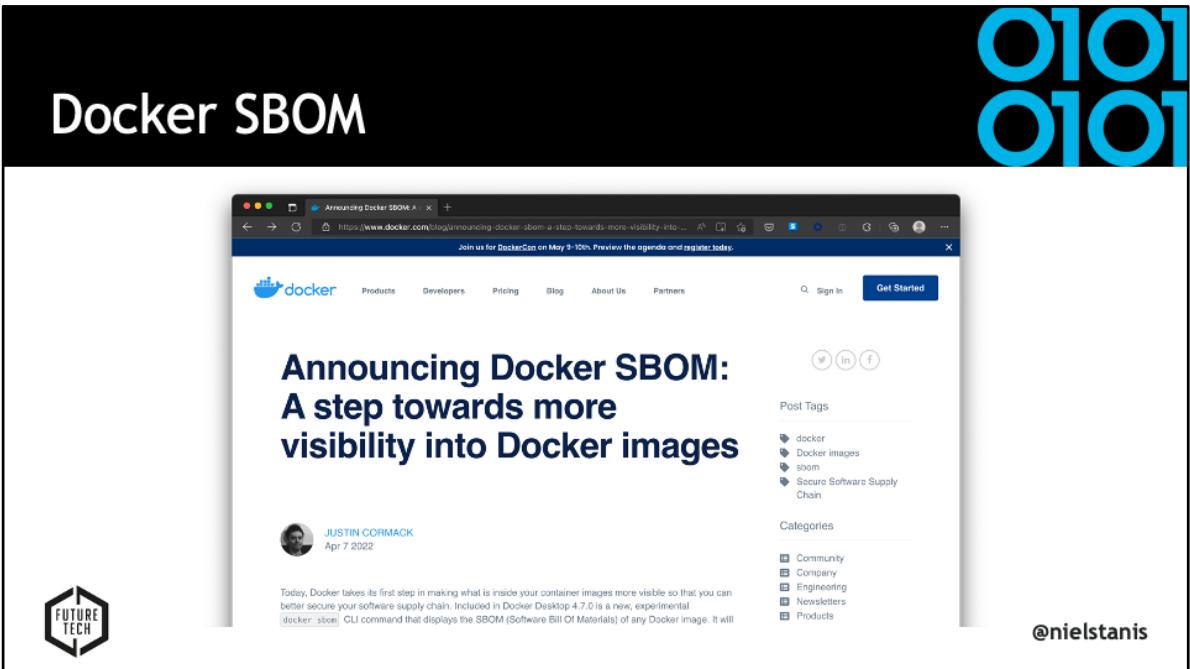
- Industry standard of describing the software
  - Producer Identity - Who Created it?
  - Product Identity - What's the product?
  - Integrity - Is the project unaltered?
  - Licensing - How can the project be used?
  - Creation - How was the product created? Process meets requirements?
  - Materials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



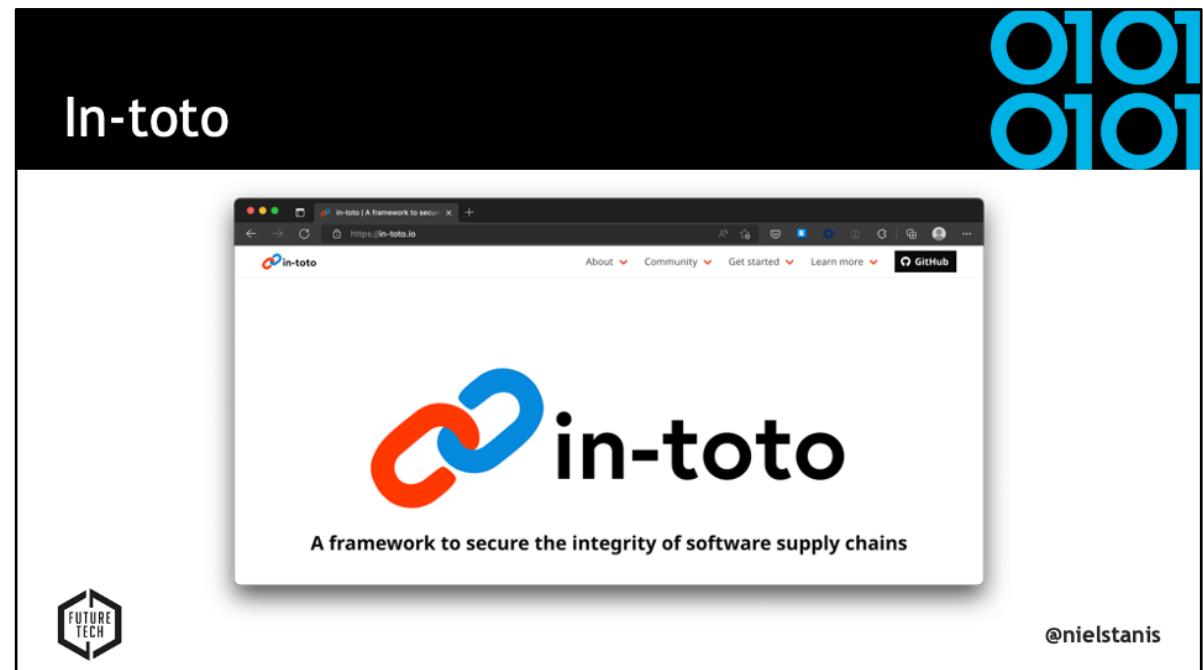
@nielstanis



<https://cyclonedx.org>



<https://www.docker.com/blog/announcing-docker-sbom-a-step-towards-more-visibility-into-docker-images/>





## 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



@nielstanis

<https://in-toto.io/>

# In-Toto - Demo

0101  
0101

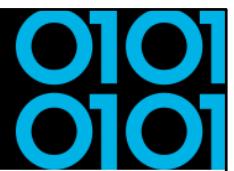
## In-Toto - Demo - Terminology

- **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

@nielstanis

<https://youtu.be/fYCfB7MZPh4?t=2777>





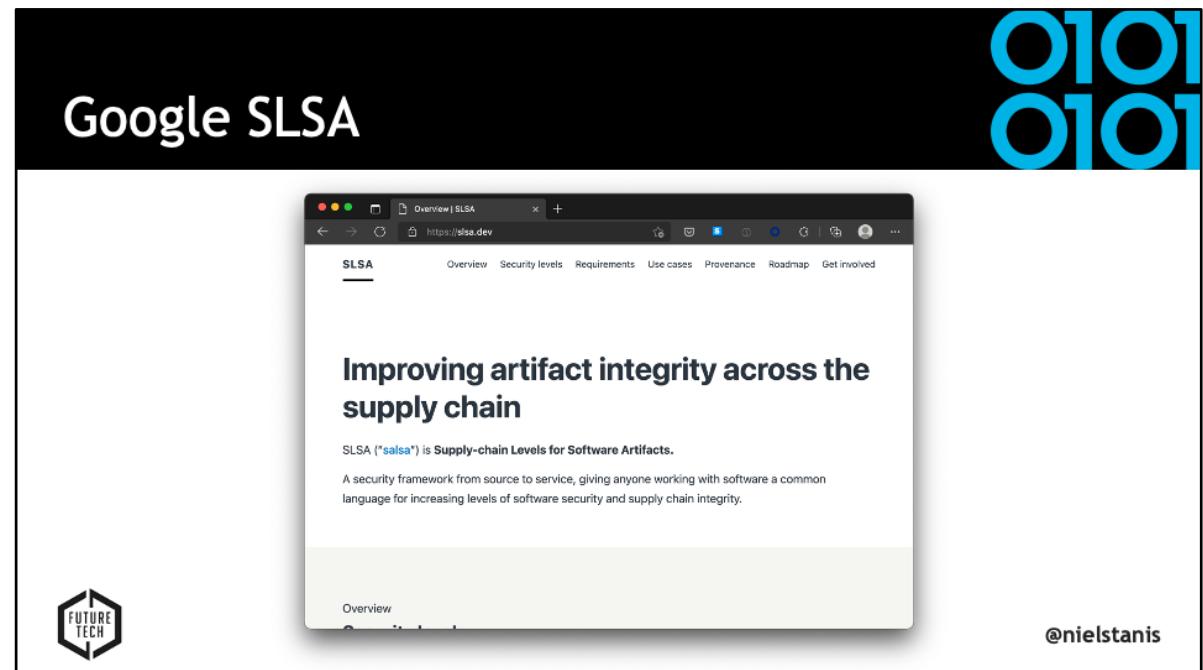
## MyAwesomeWebApp Demo

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



@nielstanis

<https://github.com/nielstanis/myawesome-webapp>



<https://slsa.dev>



## 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



@nielstanis

<https://slsa.dev>



## 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

<https://security.googleblog.com/2022/04/improving-software-supply-chain.html>

# SUSE SLSA Level 4

0101  
0101

The screenshot shows a web browser window with the URL <https://documentation.suse.com/sbp/server-linux/html/SBP-SLSA4/index.html>. The page title is "SLSA: Securing the Software Supply Chain". The content includes an "Abstract" section and a "Disclaimer" section. The "Disclaimer" section states: "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 nor the authors can be held liable for any damages caused by the use of this document." A small "FUTURE TECH" logo is visible in the bottom left corner of the page.

@nielstanis

<https://documentation.suse.com/sbp/server-linux/html/SBP-SLSA4/index.html>

0101  
0101

## Witness & GitLab Attestator

The screenshot shows a GitLab project page for 'Witness Demo'. The page includes a file tree, commit history, and a table of files with their last commits and updates.

**File Tree:**

- main
- witness-demo

**Commit History:**

Name	Last commit	Last update
policy	Update policy/gcp.rego	2 months ago
.dockerignore	Initial demo code	6 months ago

**Attestation:**

ded4ebdf

**Project Summary:**

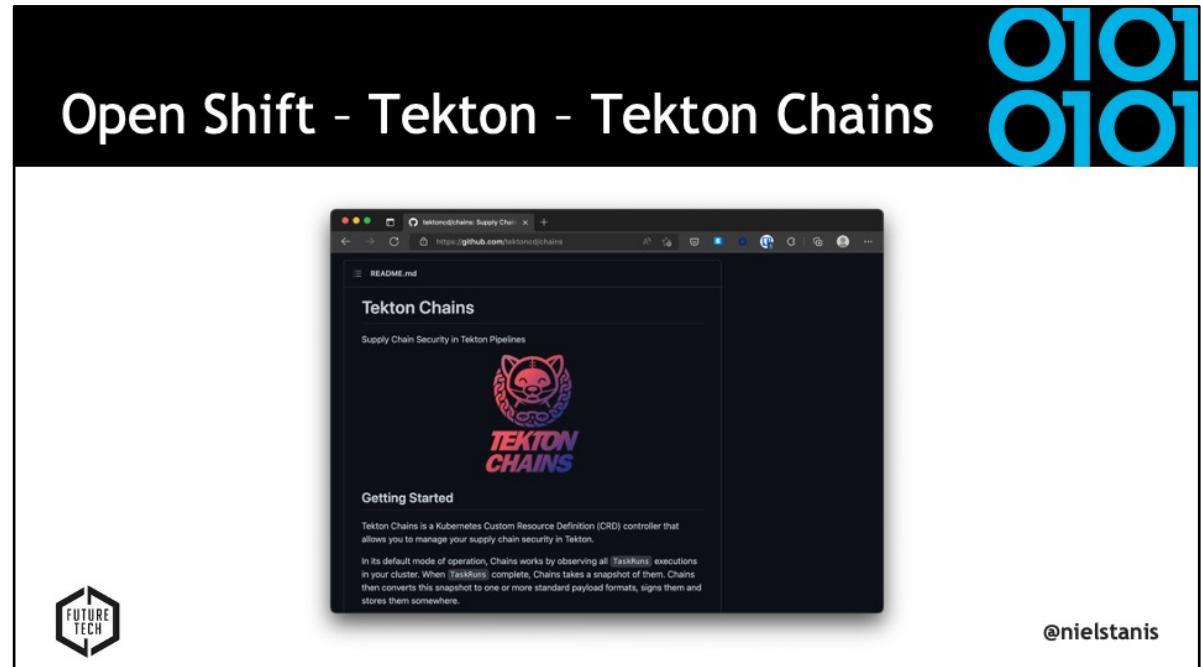
- 42 Commits
- 3 Branches
- 0 Tags
- 10827346 Bytes Project Storage

**Footer:**

@nielstanis

<https://gitlab.com/testifysec/demos/witness-demo>

<https://github.com/testifysec/witness>

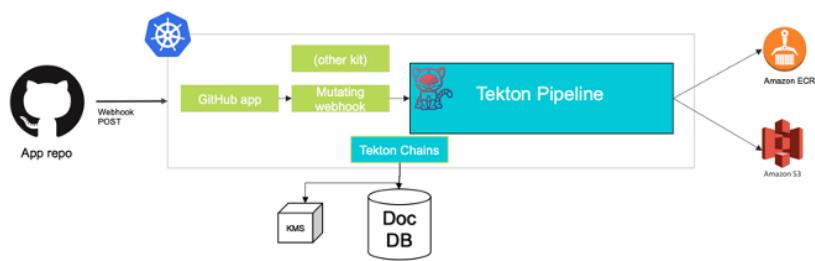


<https://github.com/tektoncd/chains>



## SolarWinds Project Trebuchet

### Pipeline With Attestations



@nielstanis

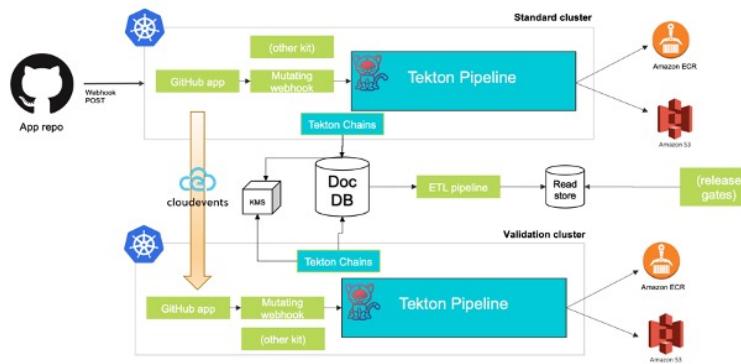
[https://static.sched.com/hosted\\_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf](https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf)

<https://www.youtube.com/watch?v=1-tMRxqMwTQ>

0101  
0101

## SolarWinds Project Trebuchet

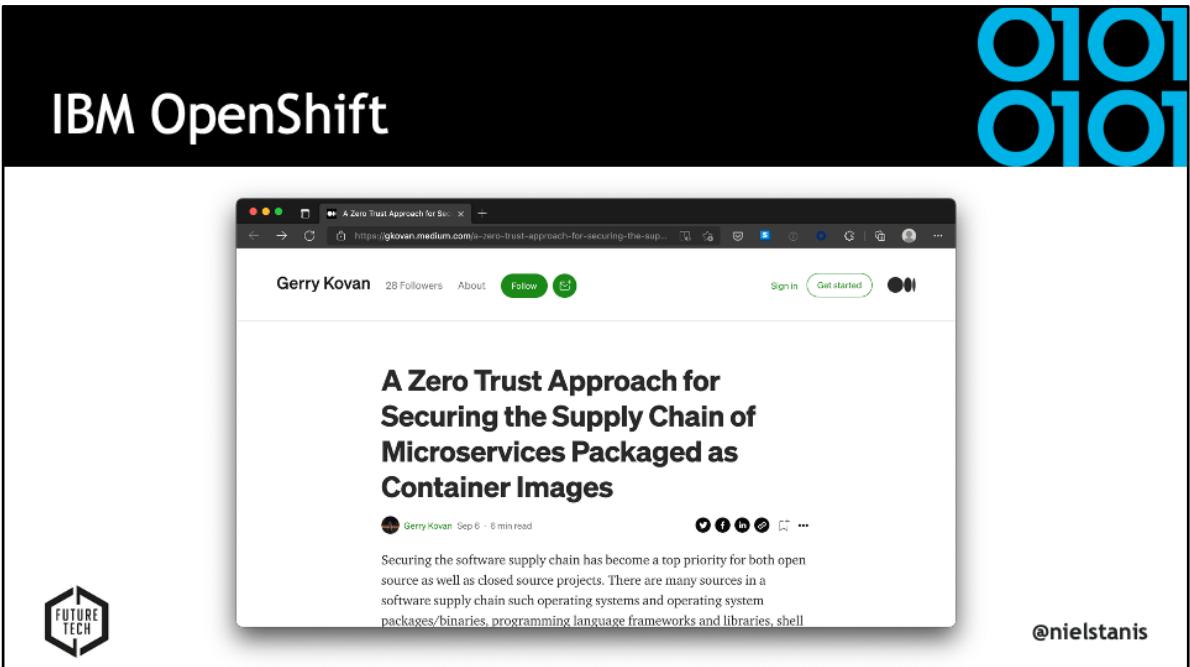
### Reading Results



@nielstanis

[https://static.sched.com/hosted\\_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf](https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf)

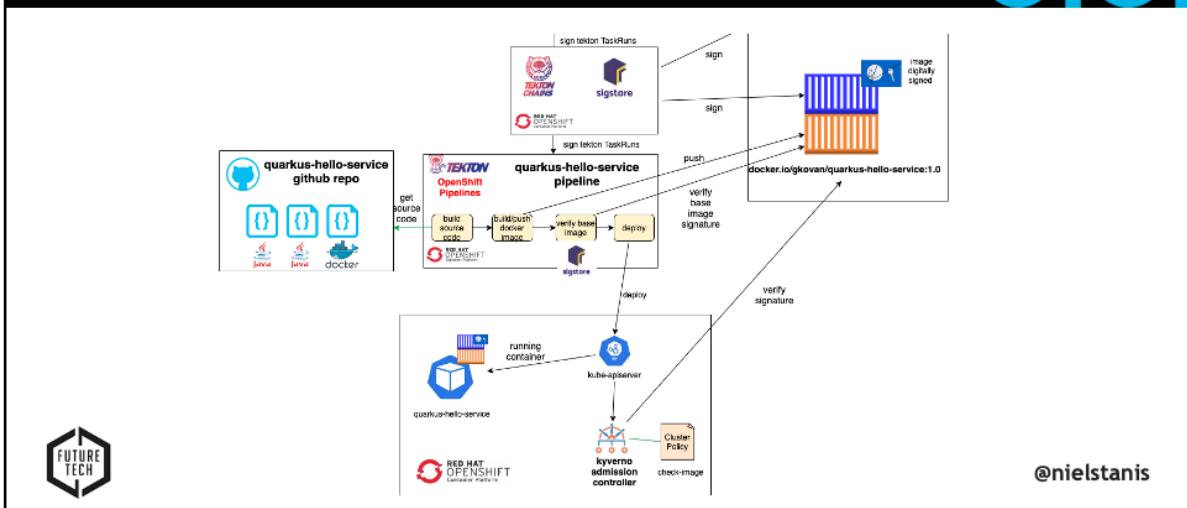
<https://www.youtube.com/watch?v=1-tMRxqMwTQ>



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

O1O1  
O1O1

# IBM OpenShift



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



## Grafeas and Kritis by Google

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



@nielstanis

<https://grafeas.io/>

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

<https://www.infoq.com/presentations/supply-grafeas-kritis/>

<https://www.youtube.com/watch?v=hOzH3mOApjs>

<https://www.youtube.com/watch?v=05zN-YQxEAM>

**Azure Policy**

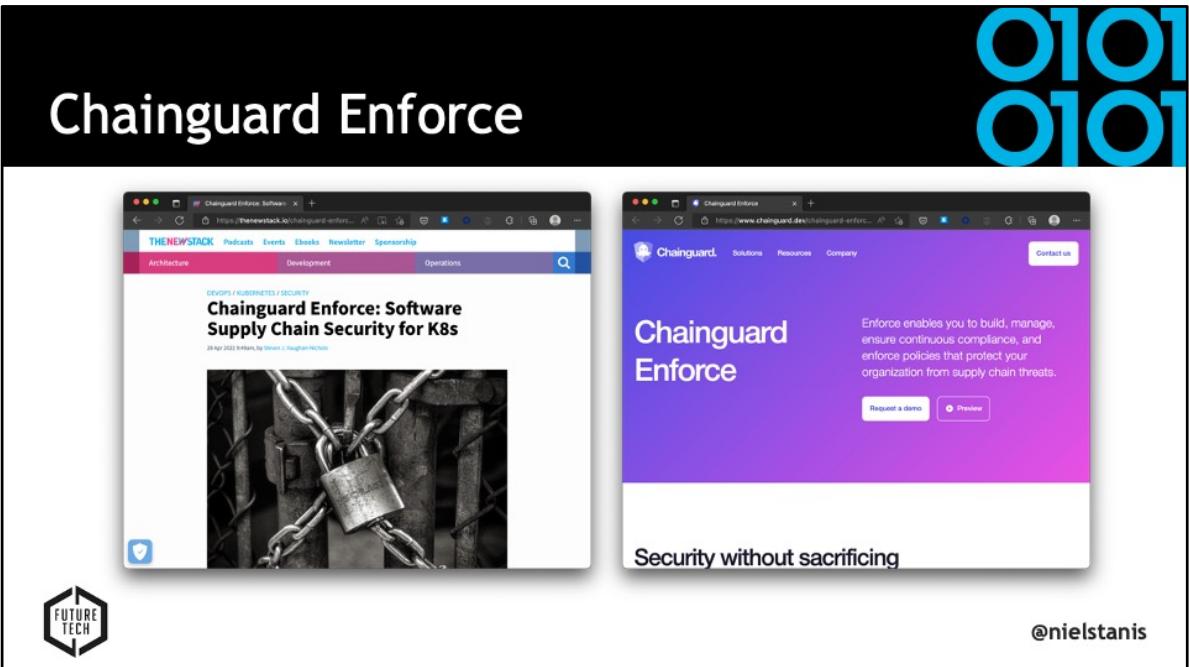
**Tutorial: Create and manage policies to enforce compliance**

<https://docs.microsoft.com/en-us/azure/governance/policy/tutorials/create-manage-policies-enforce-compliance>

**@nielstanis**

<https://devblogs.microsoft.com/devops/secure-software-supply-chain-with-azure-pipelines-artifact-policies/>

<https://docs.microsoft.com/en-us/azure/devops/pipelines/process/artifact-policy?view=azure-devops>



<https://www.chainguard.dev/chainguard-enforce>

<https://thenewstack.io/chainguard-enforce-software-supply-chain-security-for-k8s/>

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!



@nielstanis

**VERACODE**

Thanks! Questions?

<https://github.com/nielstanis/futuretech2022>  
ntanis at veracode.com  
@nielstanis on Twitter

