

NDC { London }

Securing your .NET application
software supply-chain

Niels Tanis



0101
0101

Who am I?

- Niels Tanis
- Principal Security Researcher @ Veracode
 - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
 - ISC² CSSLP
 - Research on static analysis for .NET apps

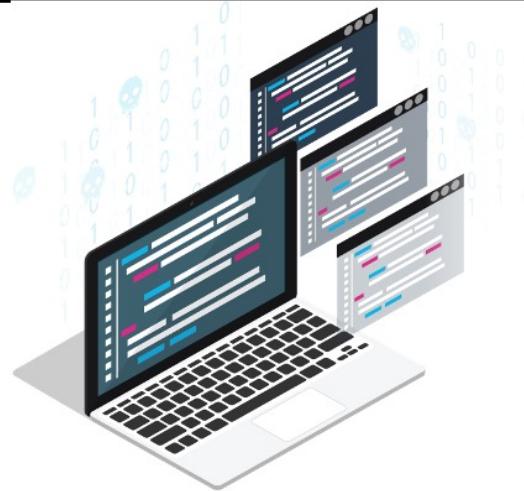


NDC { London }

@nielstanis

Securing your .NET application software supply-chain

O1O1
O1O1

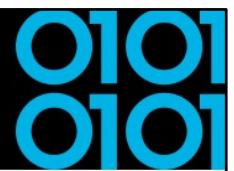


NDC { London }

@nielstanis

Picture is from Veracode report/site:

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

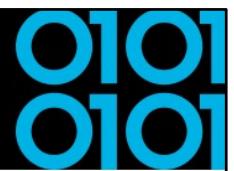


Agenda

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

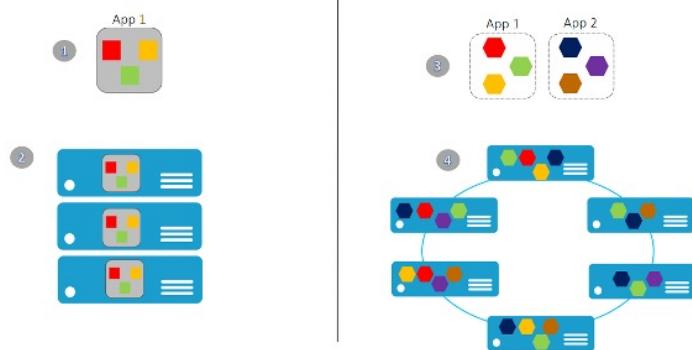
NDC { London }

@nielstanis



Evolution in Software Architecture

- Monolith
- Microservices
- Serverless
- Cloud-Native



NDC { London }

@nielstanis

O1O1
O1O1

What is a Supply Chain?

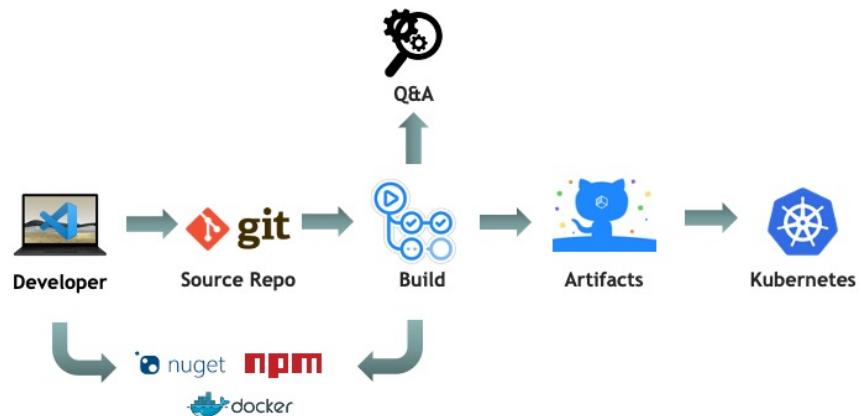


Image source:

https://www.wardsauto.com/sites/wardsauto.com/files/styles/article_featured_retina/public/Renault%20Kadjar%20assembly%20line%20-%20Palencia%20Spain-5_8.jpg?

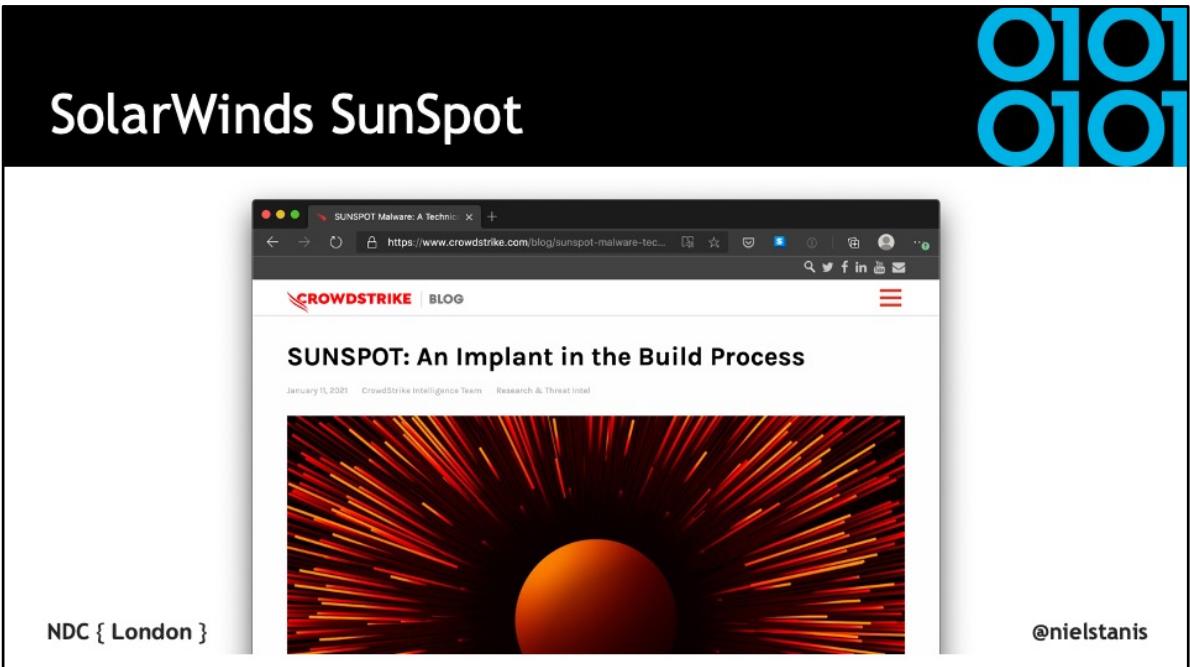
0101
0101

Software Supply Chain



NDC { London }

@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/>

Kaseya

0101
0101

Kaseya Ransomware: a Software Supply Chain Attack or Not?

July 06, 2021 By Matt Howard



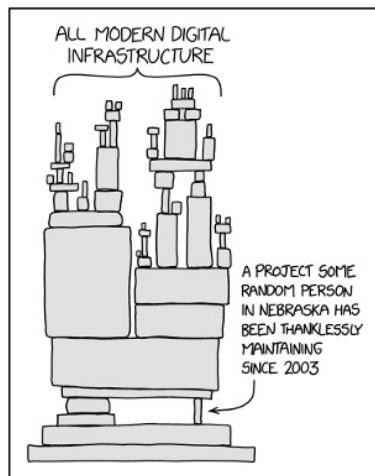
NDC { London }

@nielstanis

<https://blog.sonatype.com/kaseya-ransomware-supply-chain>

0101
0101

XKDC - Dependency



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

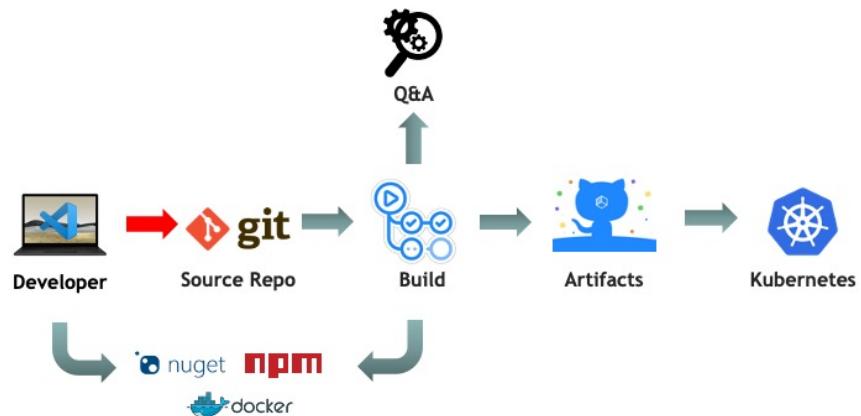
NDC { London }

@nielstanis

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

0101
0101

Software Supply Chain



NDC { London }

@nielstanis

0101
0101

GitHub account

The screenshot shows a news article from ZDNet. The title is "Canonical GitHub account hacked, Ubuntu source code safe". The subtitle reads "Ubuntu source code appears to be safe; however Canonical is investigating." Below the title is a small image of a GitHub profile page for Canonical. The URL of the article is visible at the bottom of the screenshot.

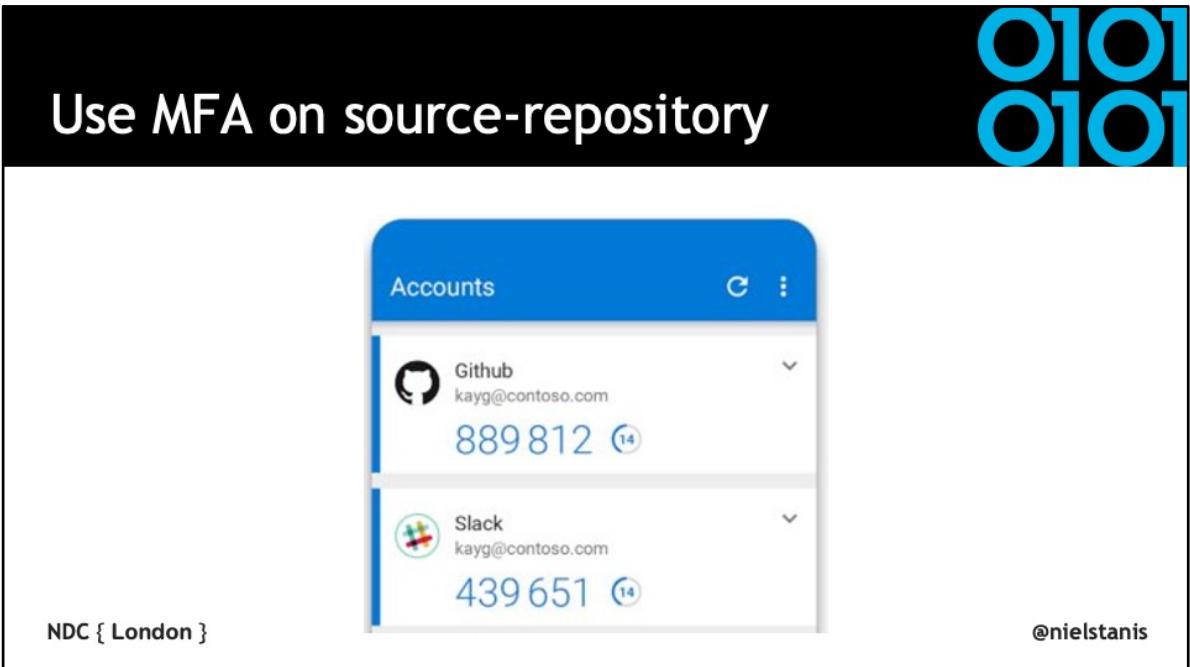
NDC { London }

@nielstanis

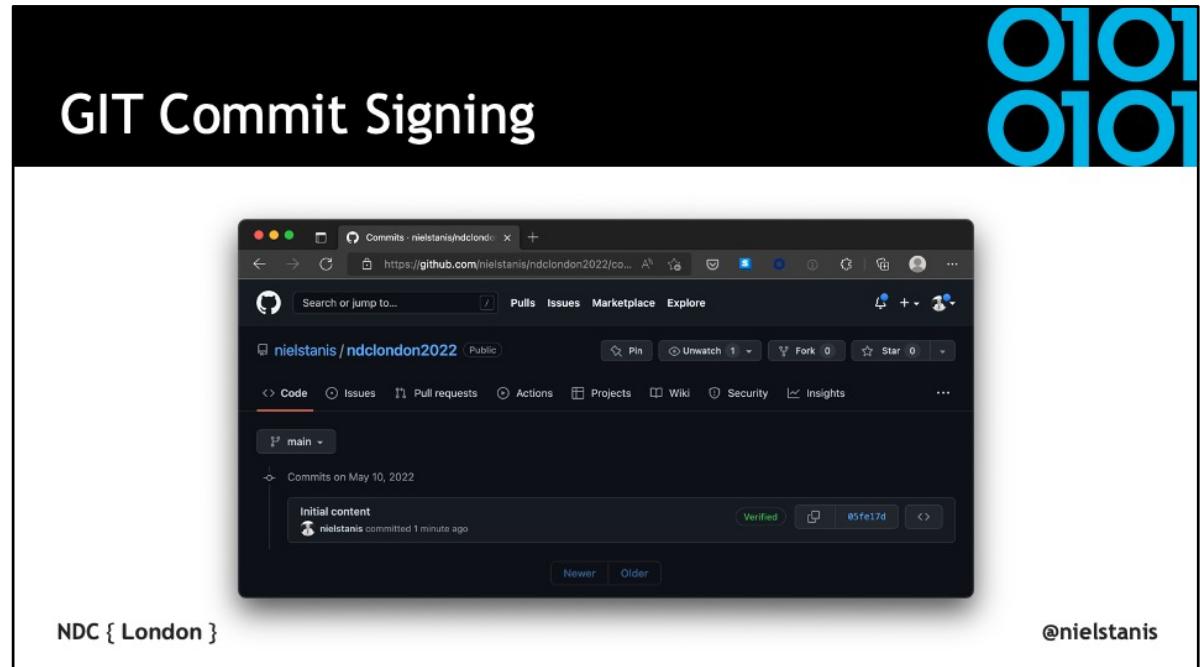
<https://www.zdnet.com/article/canonical-github-account-hacked-ubuntu-source-code-safe/>

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

Qushu 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, 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 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

NDC { London }

@nielstanis

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

Octopus Scanner - NetBeans



May 28, 2020

The Octopus Scanner Malware: Attacking the open source supply chain

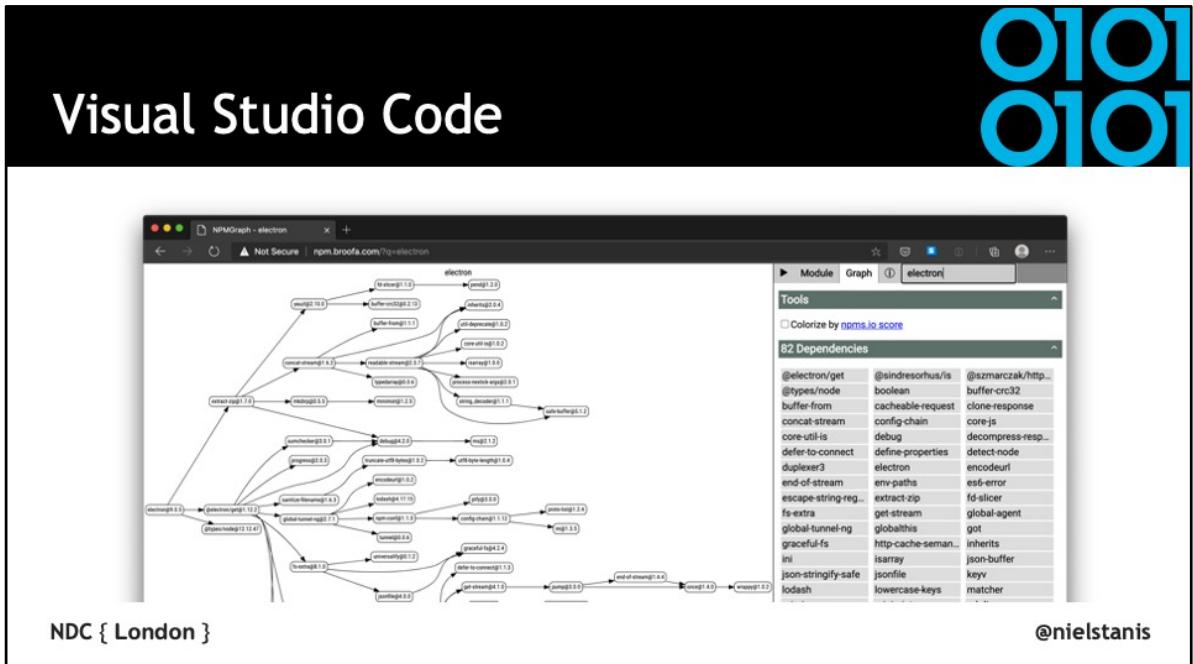
 Alvaro Muñoz

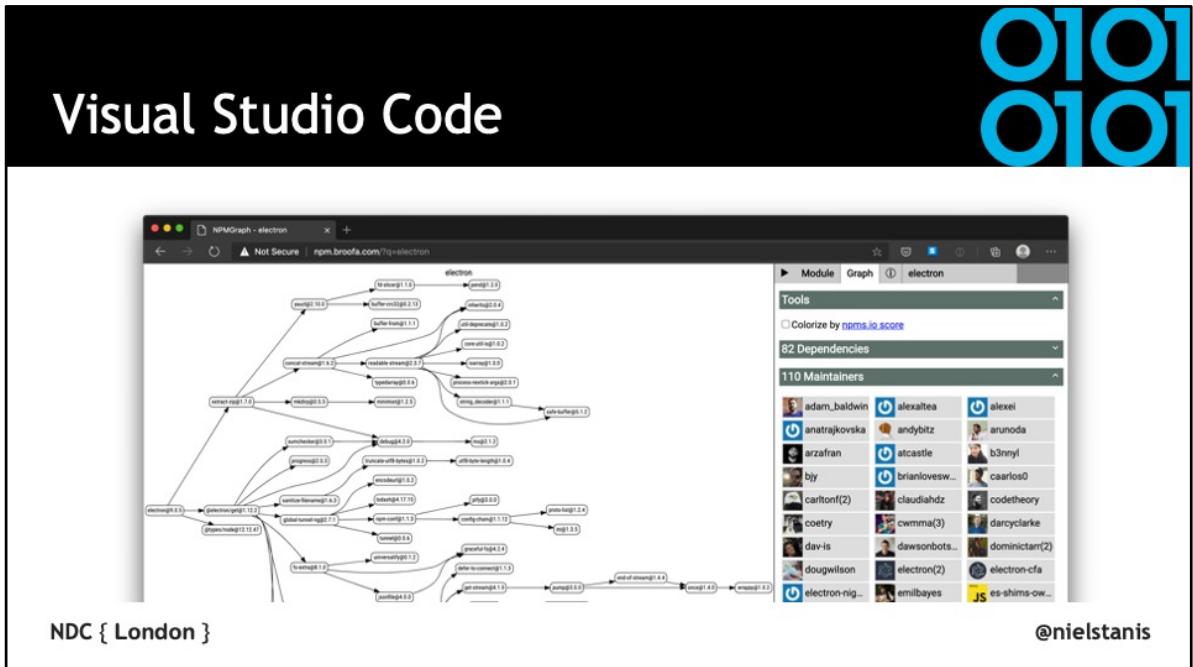
Securing the open source supply chain is an enormous task. It goes far beyond a security assessment or just patching for the latest CVEs. Supply chain security is about the integrity of the entire software development and delivery ecosystem. From the code commits themselves, to how they flow through the CI/CD pipeline, to the actual delivery of releases, there's the potential for loss of integrity and security concerns, throughout the entire lifecycle.

NDC { London }

@nielstanis

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





Visual Studio Code

0101
0101

The screenshot shows a web browser displaying a Microsoft MSRC page. The URL is <https://msrc.microsoft.com/update-guide/vulnerability/CVE-2022-21991>. The page title is "CVE-2022-21991 - Security Update". The main content is about a "Visual Studio Code Remote Development Extension Remote Code Execution Vulnerability" (CVE-2022-21991). It includes sections for "On this page", "Security Vulnerability", and "Assigning CNA". The page was released on Feb 8, 2022. The CVSS score is 7.1. The page footer contains the text "NDC { London }" and the handle "@nielstanis".

<https://msrc.microsoft.com/update-guide/vulnerability/CVE-2022-21991>

Visual Studio Code

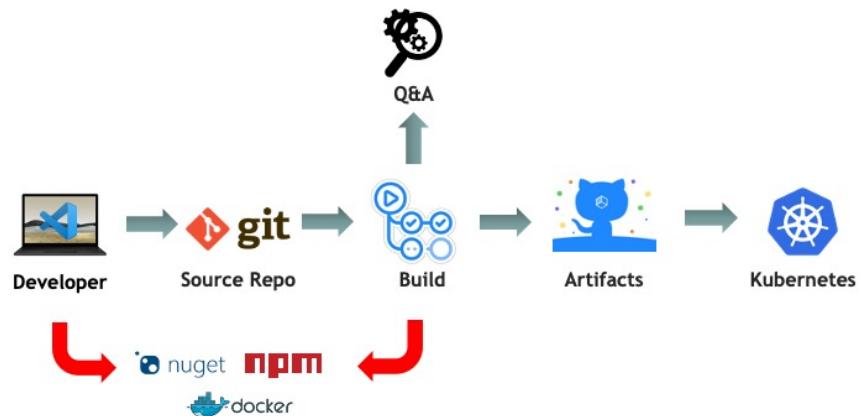
0101
0101



NDC { London }

0101
0101

3rd Party Libraries



NDC { London }

@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."*



NDC { London }

@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 a Microsoft security advisory. The title is "Microsoft Security Advisory CVE-2022-24512 | .NET Remote Code Execution Vulnerability #213". The issue was opened by `dewhittaker` 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 6.0`, `Patch-Tuesday`, and `Security`. The "Discussion" section links to `dotnet/runtime#68348`. On the right, there are sections for "Assignees" (none assigned), "Labels" (with the three mentioned labels), "Projects" (none yet), and "Milestone" (none yet). The GitHub URL is `https://github.com/dotnet/announcements/issues/213`.

NDC { London }

@nielstanis



CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

Vulnerabilities in libraries

0101
0101

Alerts and Tips Resources Industrial Control Systems

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

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

Original release date: October 22, 2021

Print Tweet Send Share

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

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

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

NDC { London } @nielstanis

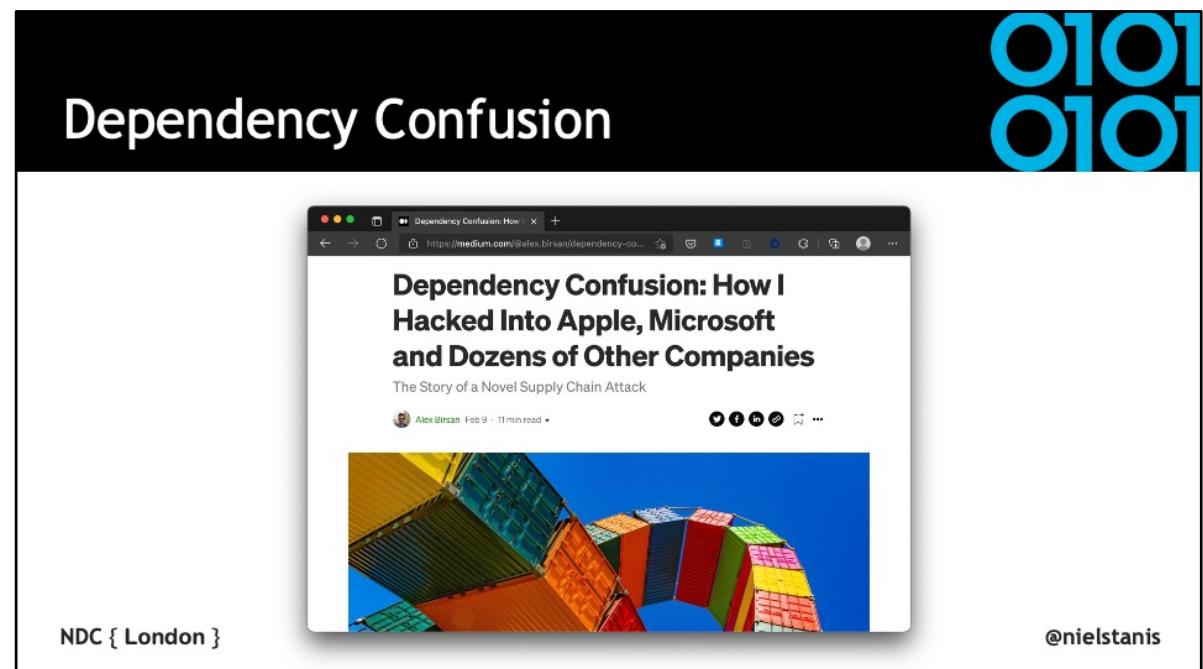
<https://us-cert.cisa.gov/ncas/current-activity/2021/10/22/malware-discovered-popular-npm-package-ua-parser-js>
<https://portswigger.net/daily-swig/popular-npm-package-ua-parser-js-poisoned-with-cryptomining-password-stealing-malware>

Vulnerabilities in libraries

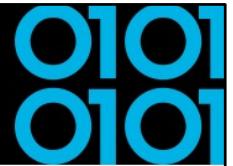
NDC { London }

@nielstanis

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



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



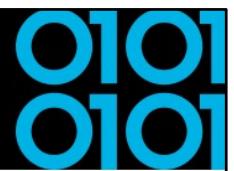
Dependency Confusion - NuGet

- 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

NDC { London }

@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
- Look out for talk on 'Sandboxing .NET Assemblies' I gave two weeks ago at NDC Porto!
- Open Source Security Foundation - OpenSSF
- Security Scorecards - Security health metrics for Open Source

NDC { London }

@nielstanis

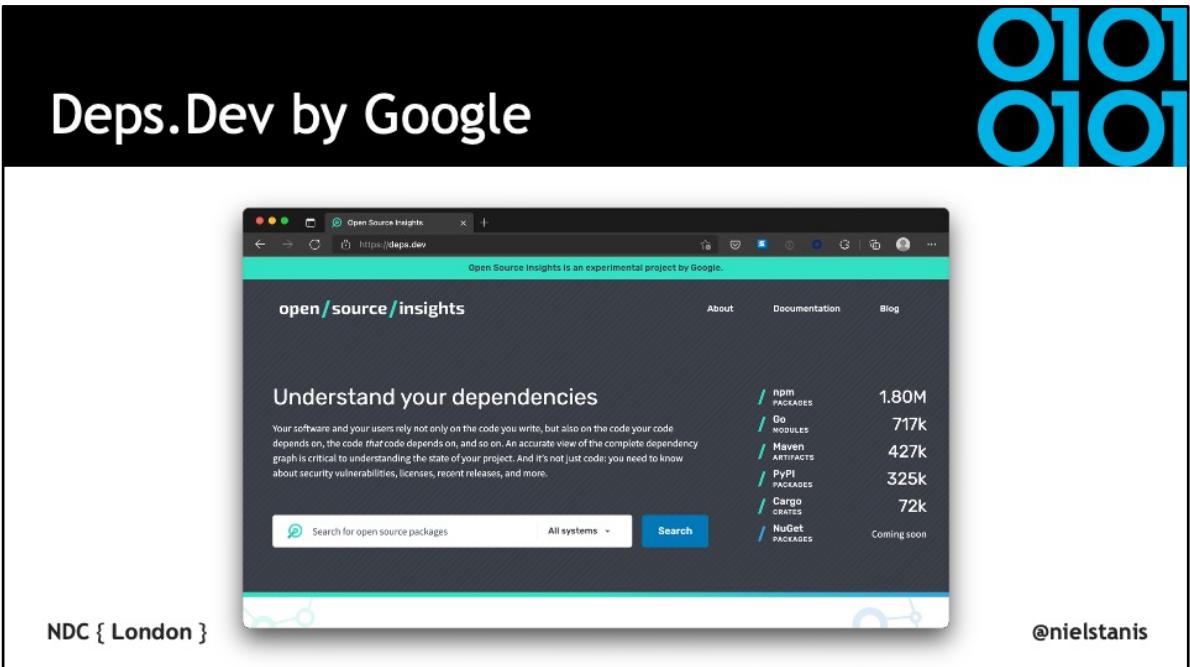
Security Scorecards - OpenSSF



NDC { London }

@nielstanis

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



<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.

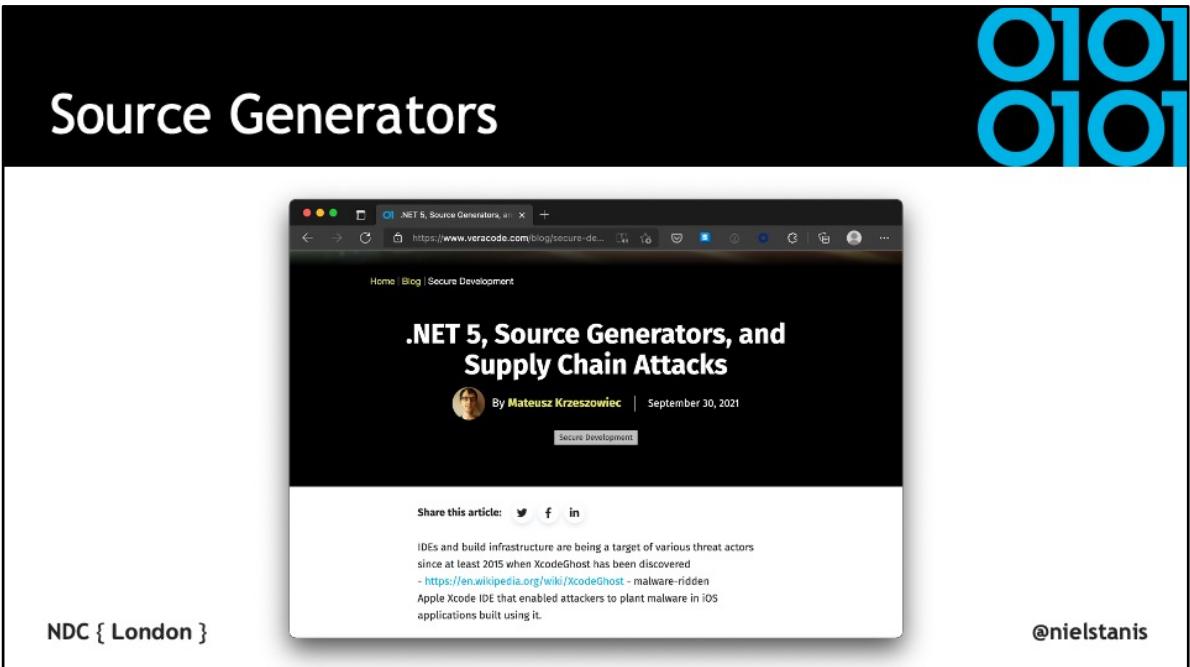
Category	Score
Code-Review	5/10
Maintained	10/10
CI-Best-Practices	0/10
Vulnerabilities	10/10
Dependency-Update-Tool	0/10
Security-Policy	10/10
Dangerous-Workflow	10/10
Token-Permissions	0/10
License	10/10
Pinned-Dependencies	8/10
Binary-Artifacts	10/10
Fuzzing	0/10
Signed-Releases	0/10

Project metadata as of May 7, 2022.

NDC { London }

@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>
```

NDC { London }

@nielstanis

Reproducible/Deterministic Builds

0101
0101



Home

Contribute

[Documentation](#)

Tools

Who is involved?

News

Events

Talks

Definitions

When is a build reproducible?

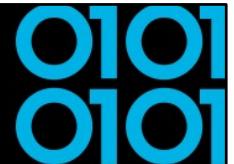
A build is **reproducible** if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts.

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.

NDC { London }

@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’

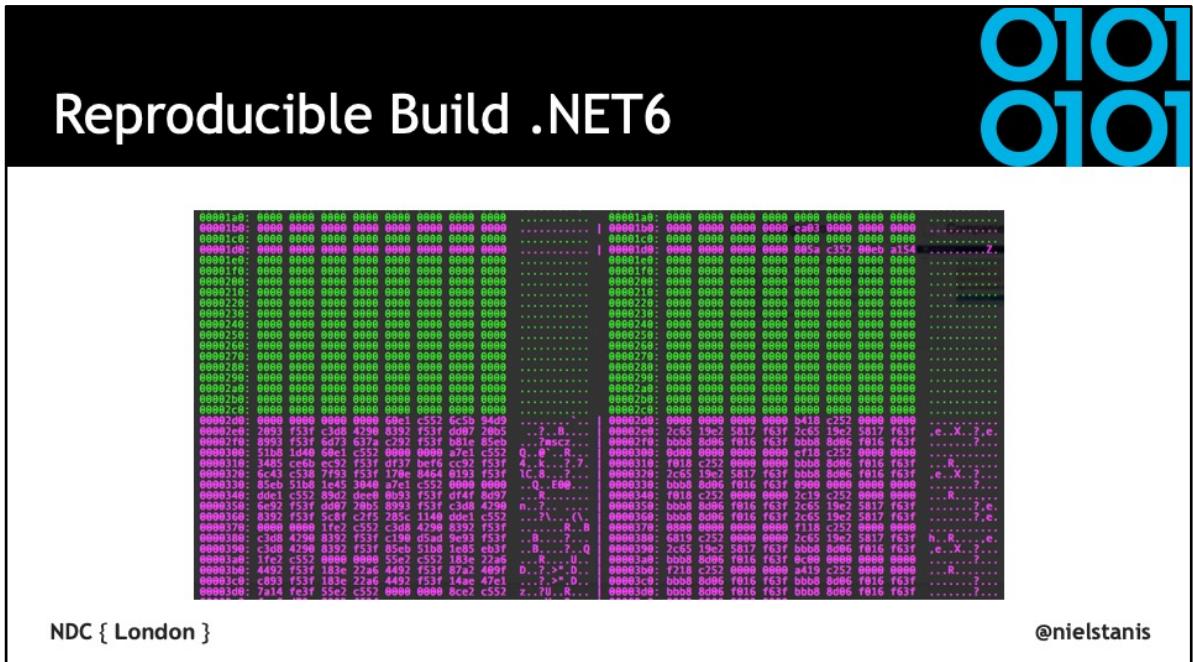
NDC { London }

@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

NDC { London }

@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`

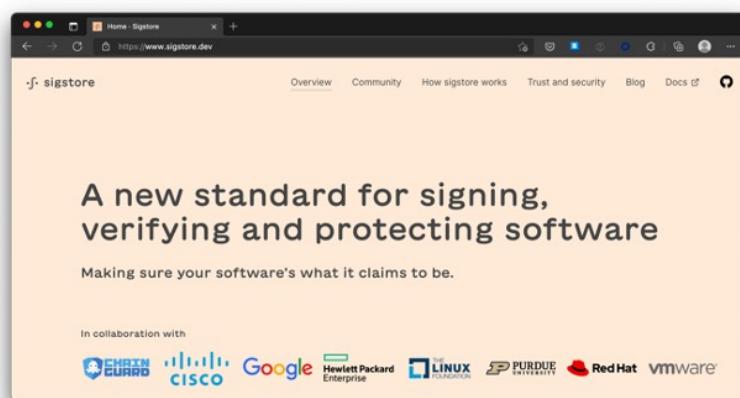
NDC { London }

@nielstanis

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

Signing artifacts

0101
0101



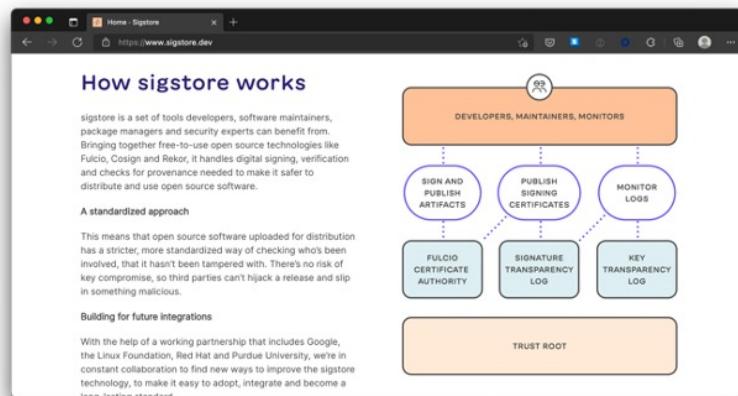
NDC { London }

@nielstanis

<https://sigstore.dev>

Signing artifacts

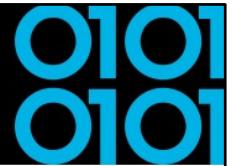
0101
0101



NDC { London }

@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

NDC { London }

@nielstanis

<https://sigstore.dev>

O1O1
O1O1

Automotive Industry



NDC { London }

@nielstanis

O1O1
O1O1

Car Supply Chain



Tata Steel Factory

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

Bosch Factory

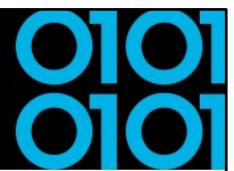
- Steel Batch #1234 Tata
- ECE-R90 Tested/Certified
 - Serie #45678
- Used by Ford, Volkswagen and Renault

Renault Manufacturing

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

NDC { London }

@nielstanis



Software Bill of Materials (SBOM)

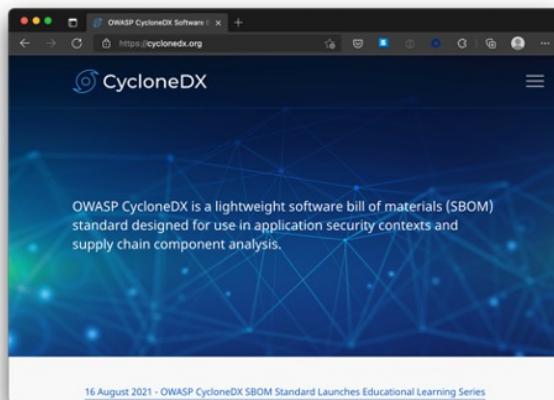
- Industry standard of describing the software
 - Producer Identity - Who Created it?
 - Product Identity - What's the product?
 - Integrity - Is the project unaltered?
 - Licensing - How can the project be used?
 - Creation - How was the product created? Process meets requirements?
 - Materials - How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

NDC { London }

@nielstanis

Software Bill of Materials (SBOM)

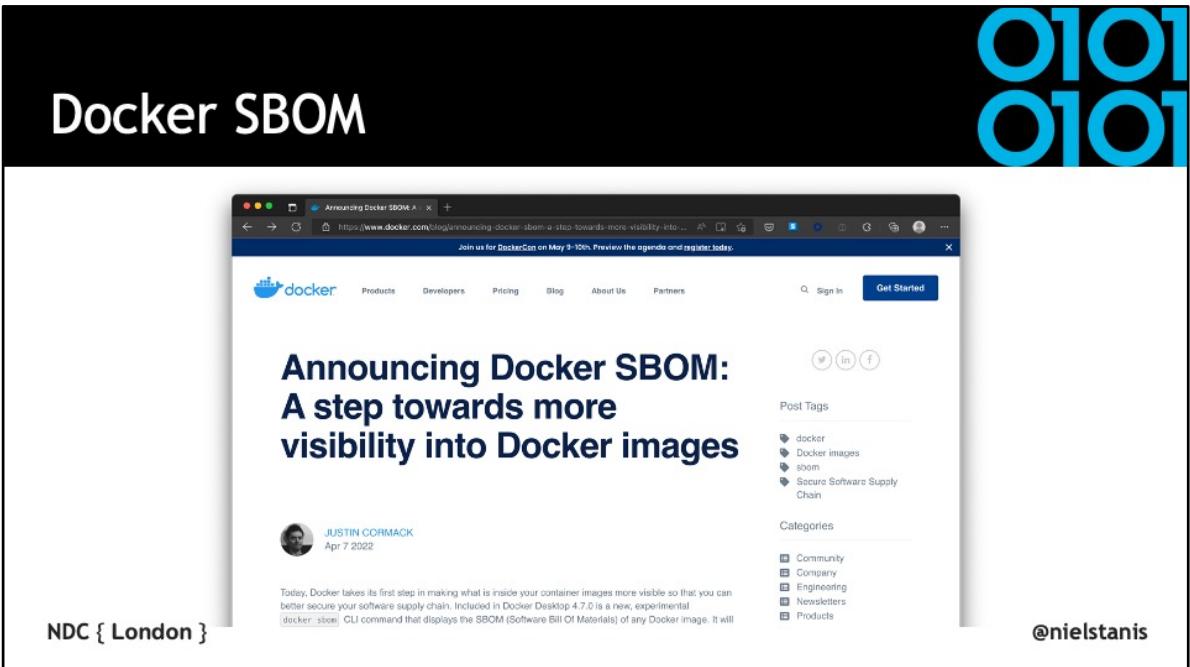
0101
0101



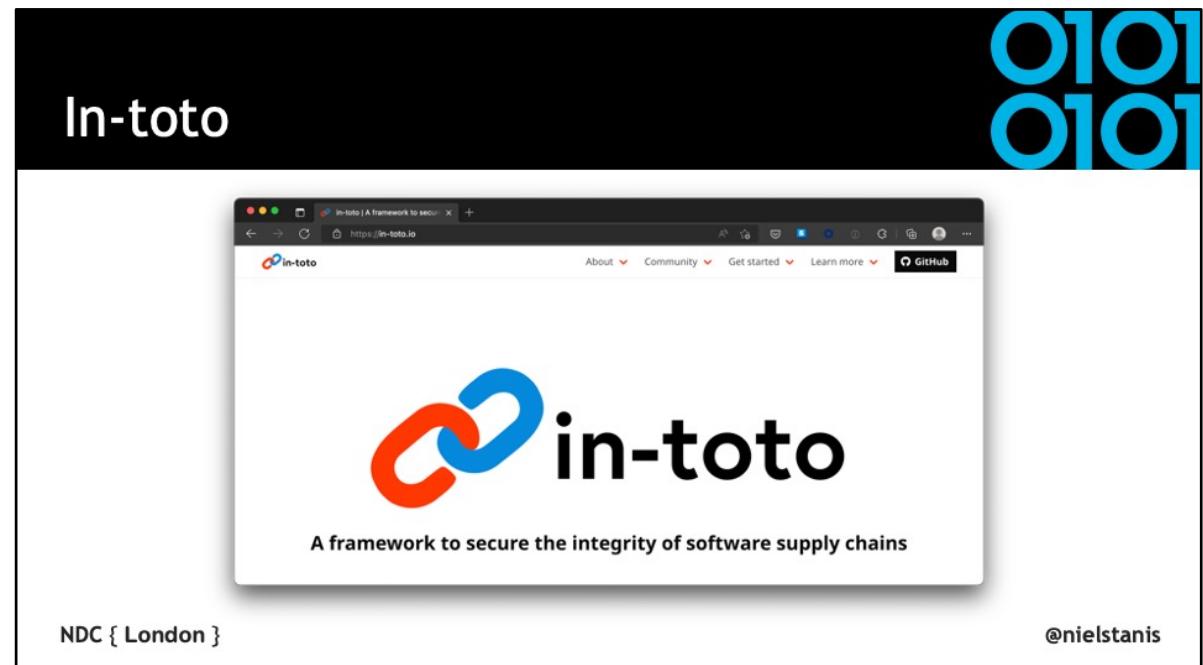
NDC { London }

@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

NDC { London }

@nielstanis

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

In-Toto - Demo

O1O1
O1O1

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

NDC { London }

@nielstanis

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

MyAwesomeWebApp Demo

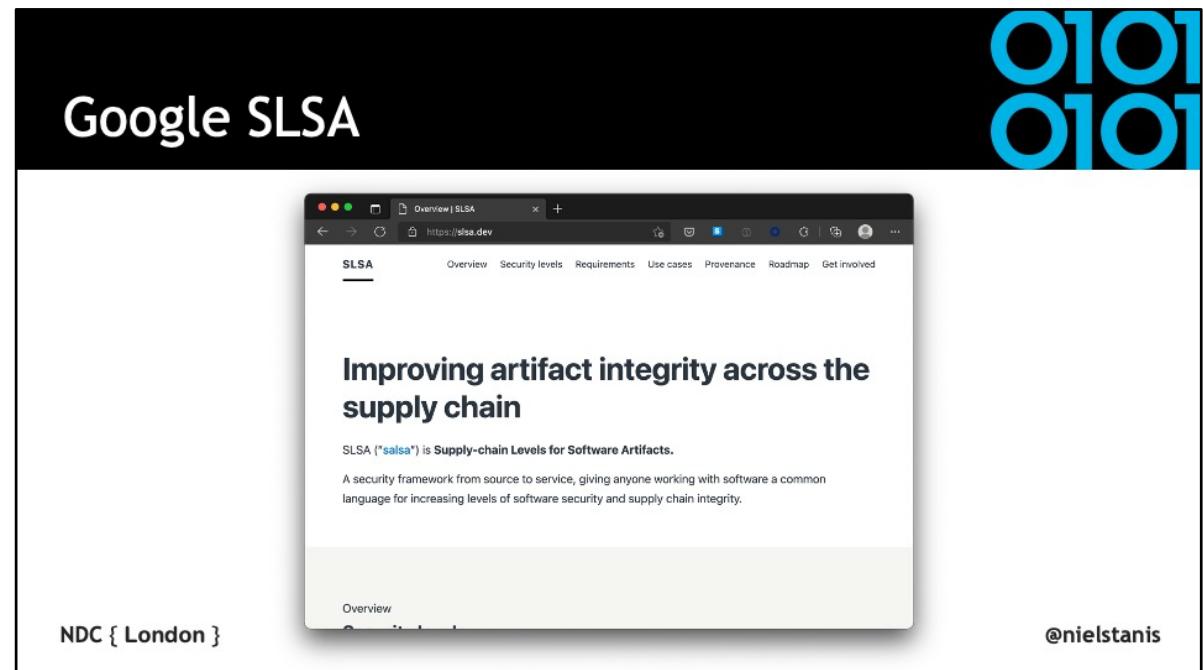
0101
0101

- CycloneDX
- In-Toto
- Sigstore
- Docker SBOM

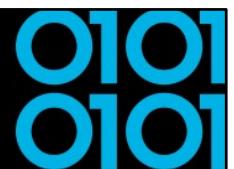


NDC { London }

@nielstanis



<https://slsa.dev>



Google SLSA Levels

1. The build process must be fully scripted/automated and generate provenance.
2. Requires using version control and a hosted build service that generates authenticated provenance.
3. The source and build platforms meet specific standards to guarantee the auditability of the source and the integrity of the provenance respectively.
4. Requires two-person review of all changes and a hermetic, reproducible build process.

NDC { London }

@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

NDC { London }

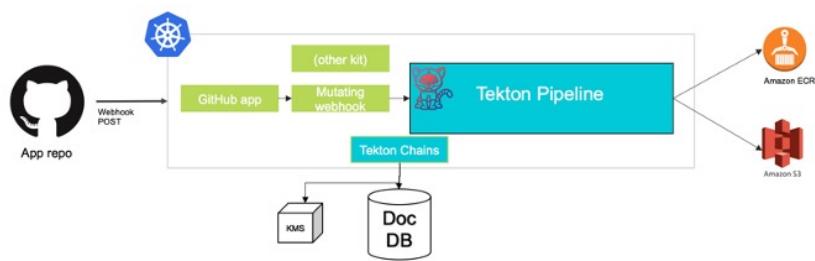
@nielstanis

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



SolarWinds Project Trebuchet

Pipeline With Attestations



NDC { London }

@nielstanis

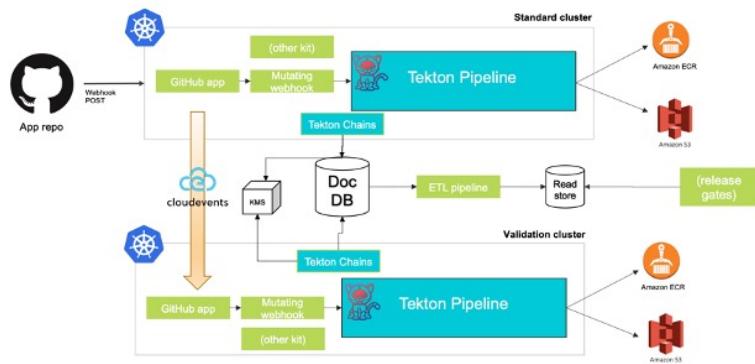
https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf

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

0101
0101

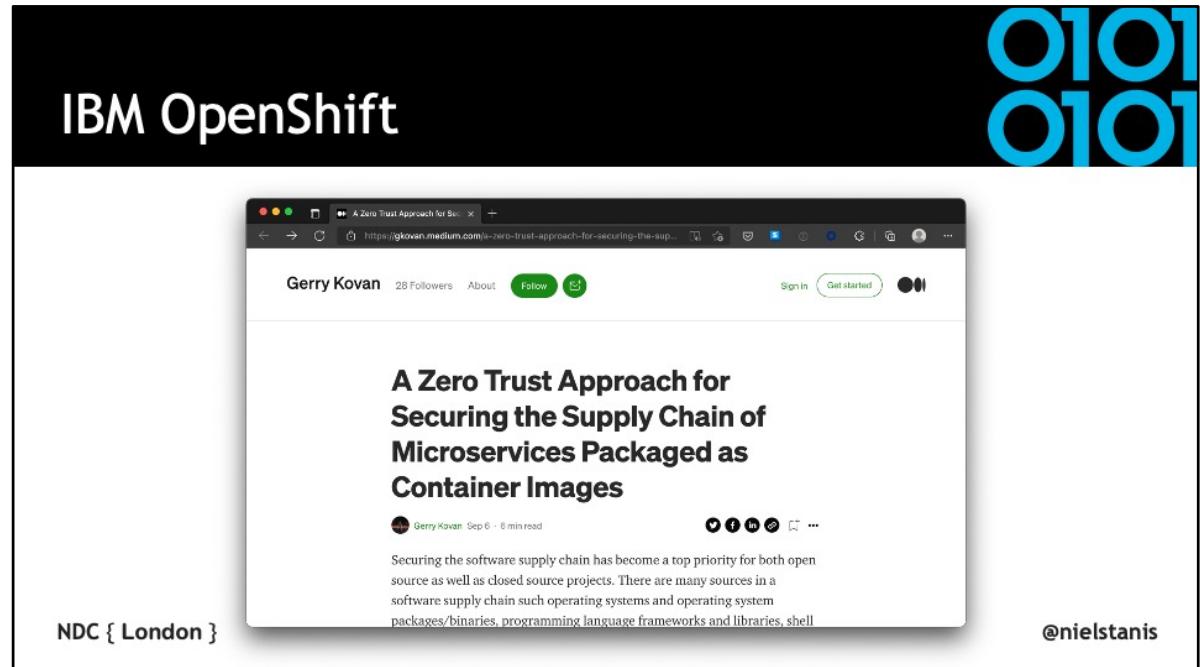
SolarWinds Project Trebuchet

Reading Results



https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf

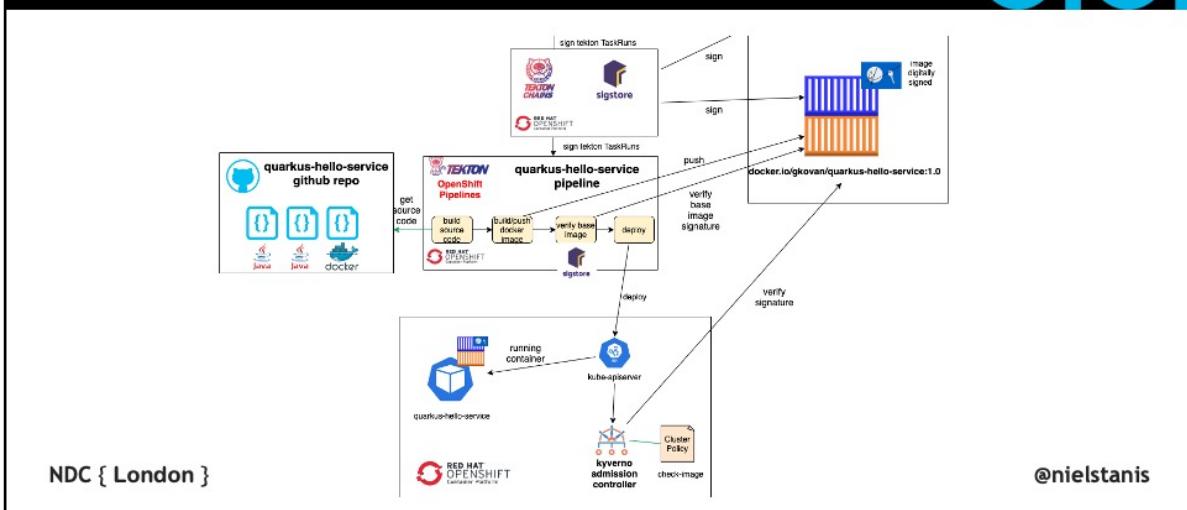
<https://www.youtube.com/watch?v=1-tMRxqMwTQ>



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

0101
0101

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



NDC { London }

@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>

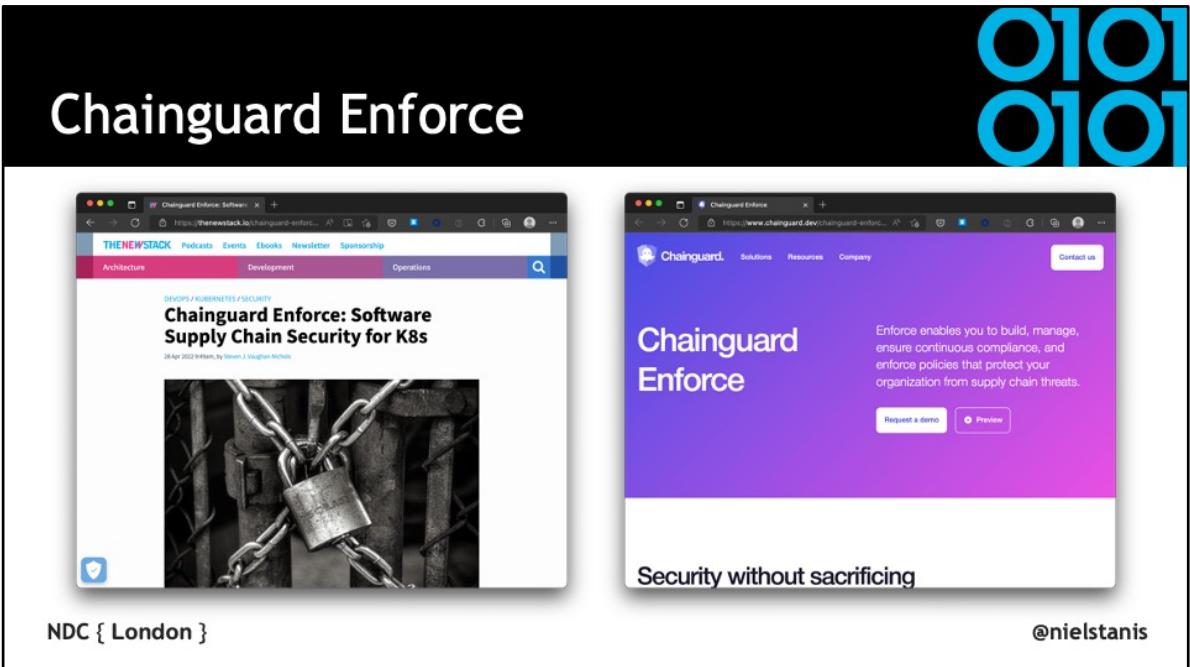
A screenshot of a Microsoft Azure DevOps documentation page titled "Artifact policy checks". The page is part of the "Azure Pipelines" section under "Reference". The main content area displays the article's title, last update date (11/05/2019), and a brief description of artifact policies. On the left, there is a sidebar with navigation links for "Version" (Azure DevOps Services), "Filter by title", and several sections like "Configure security & settings", "Migrate", "Pipeline tasks", "Troubleshooting", and "Reference". The top right features a "Try for free" button. The top of the slide has a decorative graphic with the binary code "0101 0101" in blue.

NDC { London }

@nielstanis

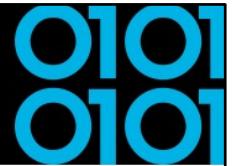
<https://devblogs.microsoft.com/devops/secure-software-supply-chain-with-azure-pipelines-artifact-policies/>

<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/>

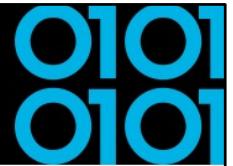


Conclusion

- 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.

NDC { London }

@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!

NDC { London }

@nielstanis

VERACODE

Thanks! Questions?

<https://github.com/nielstanis/ndclondon2022>
ntanis at veracode.com
@nielstanis on Twitter

