Securing your .NET application software supply-chain

Niels Tanis

VERACODE

Who am I?

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

@nielstanis

Securing your .NET application software supply-chain

FUTURE TECH

@nielstanis

Agenda

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

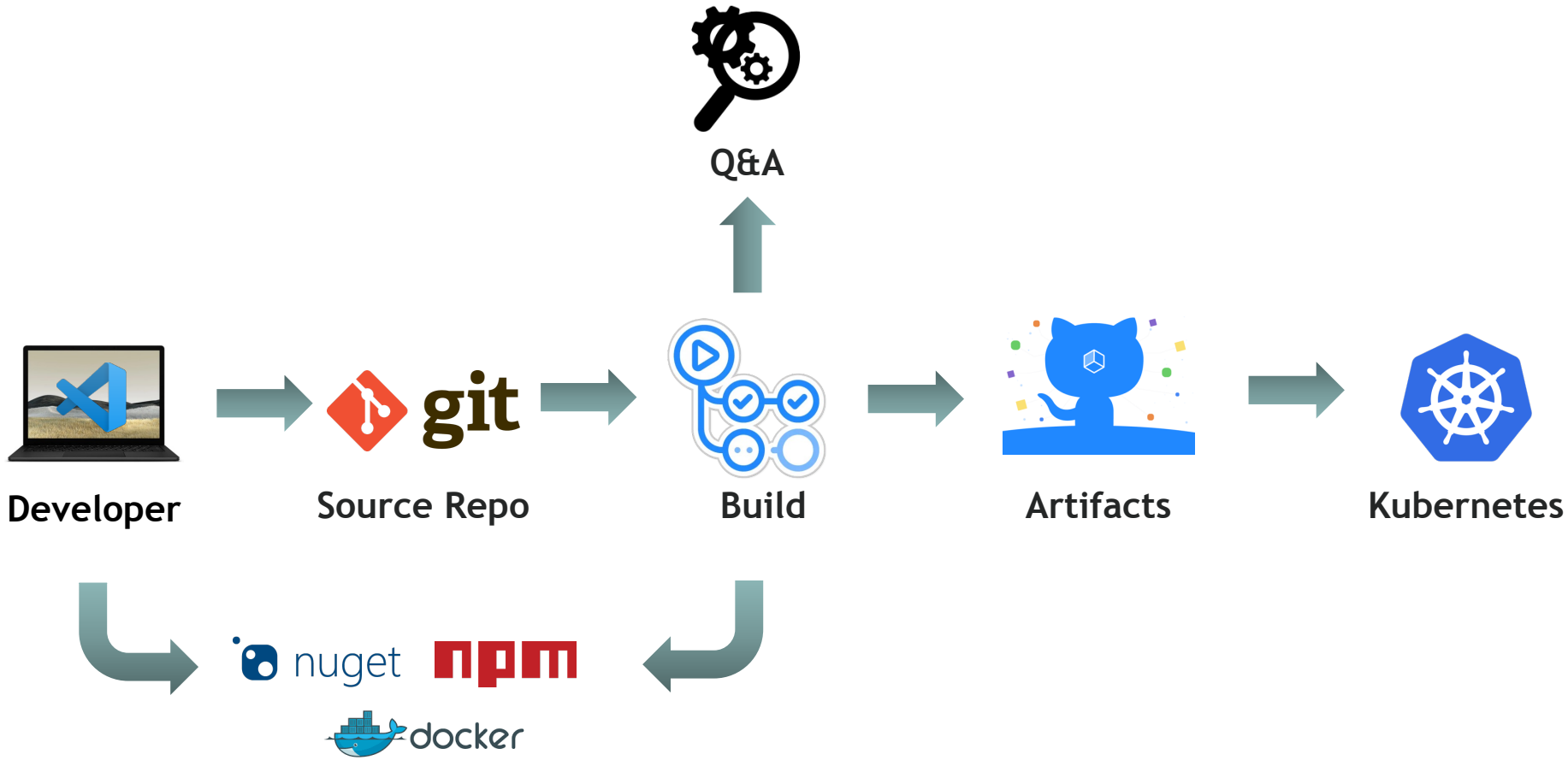
FUTURE TECH

What is a Supply Chain?

FUTURE TECH

@nielstanis

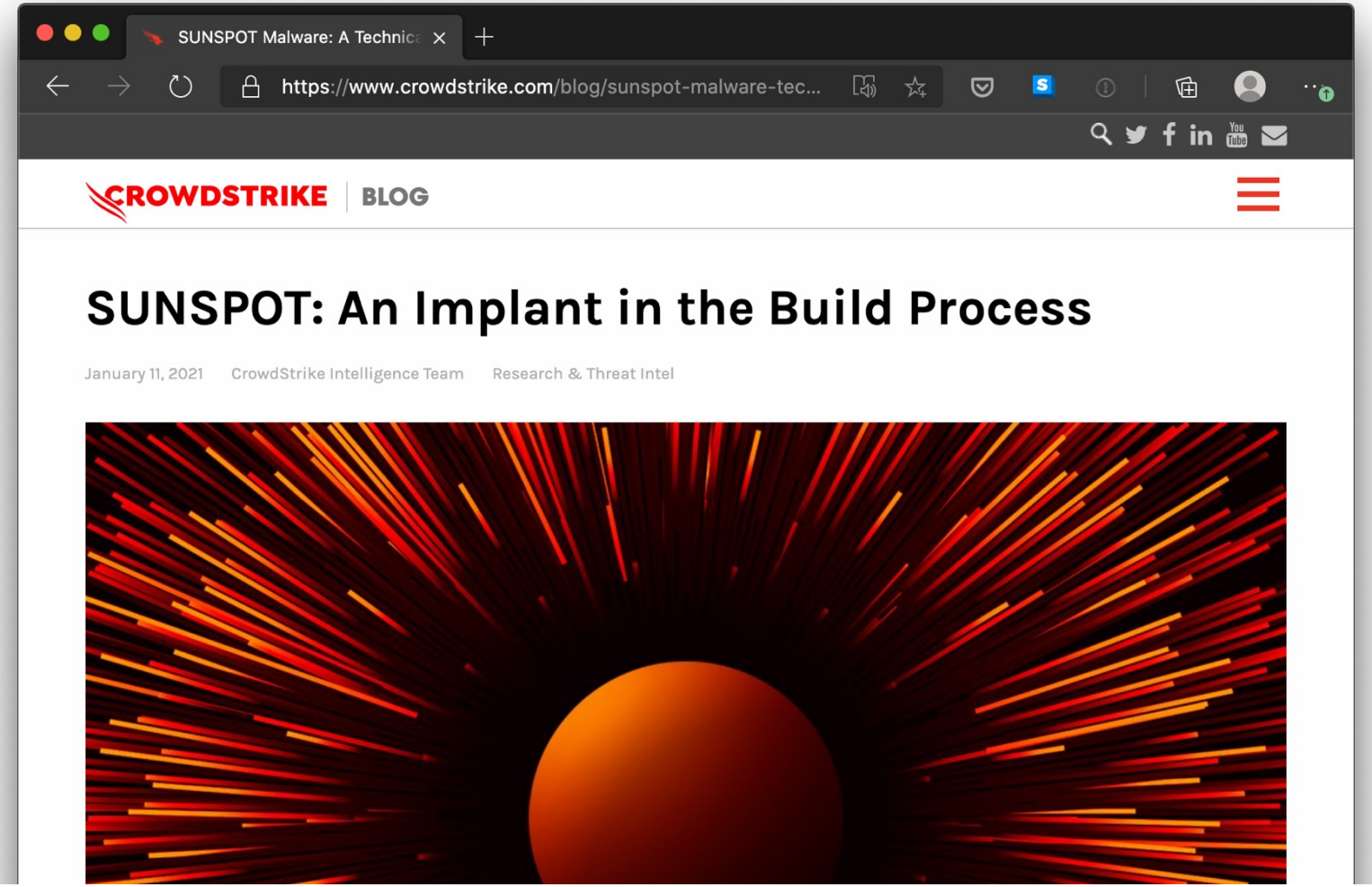
Software Supply Chain



FUTURE TECH

@nielstanis

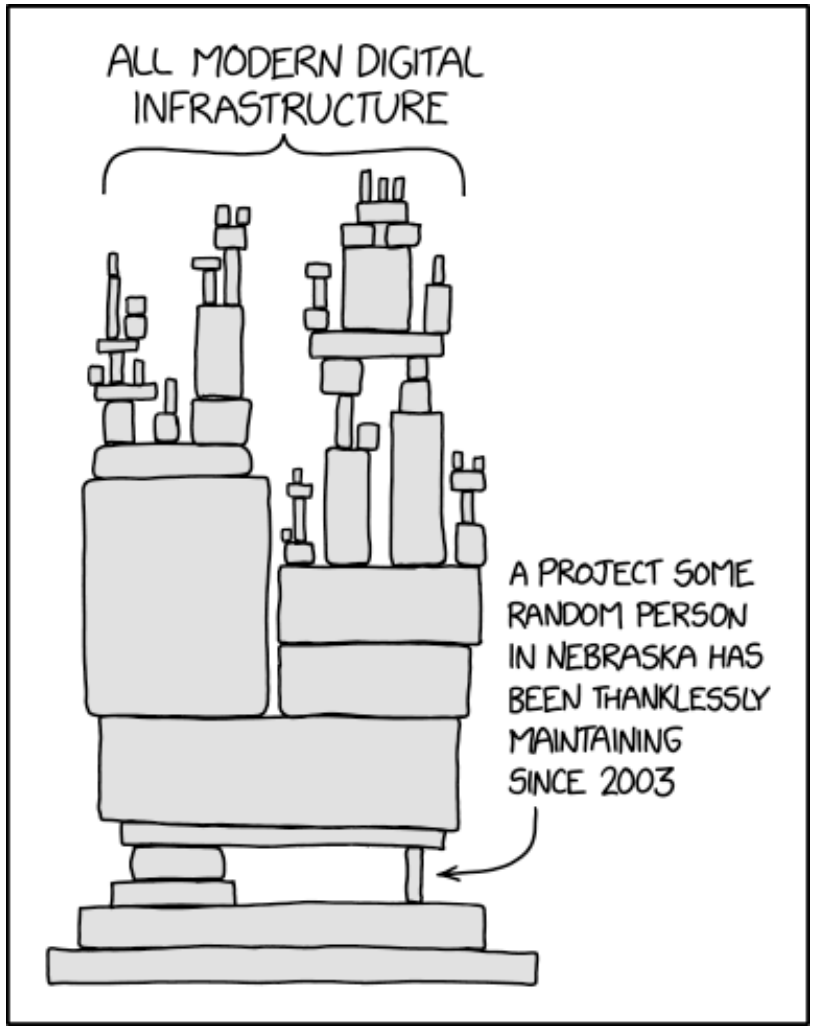
SolarWinds SunSpot



FUTURE TECH

@nielstanis

XKDC - Dependency

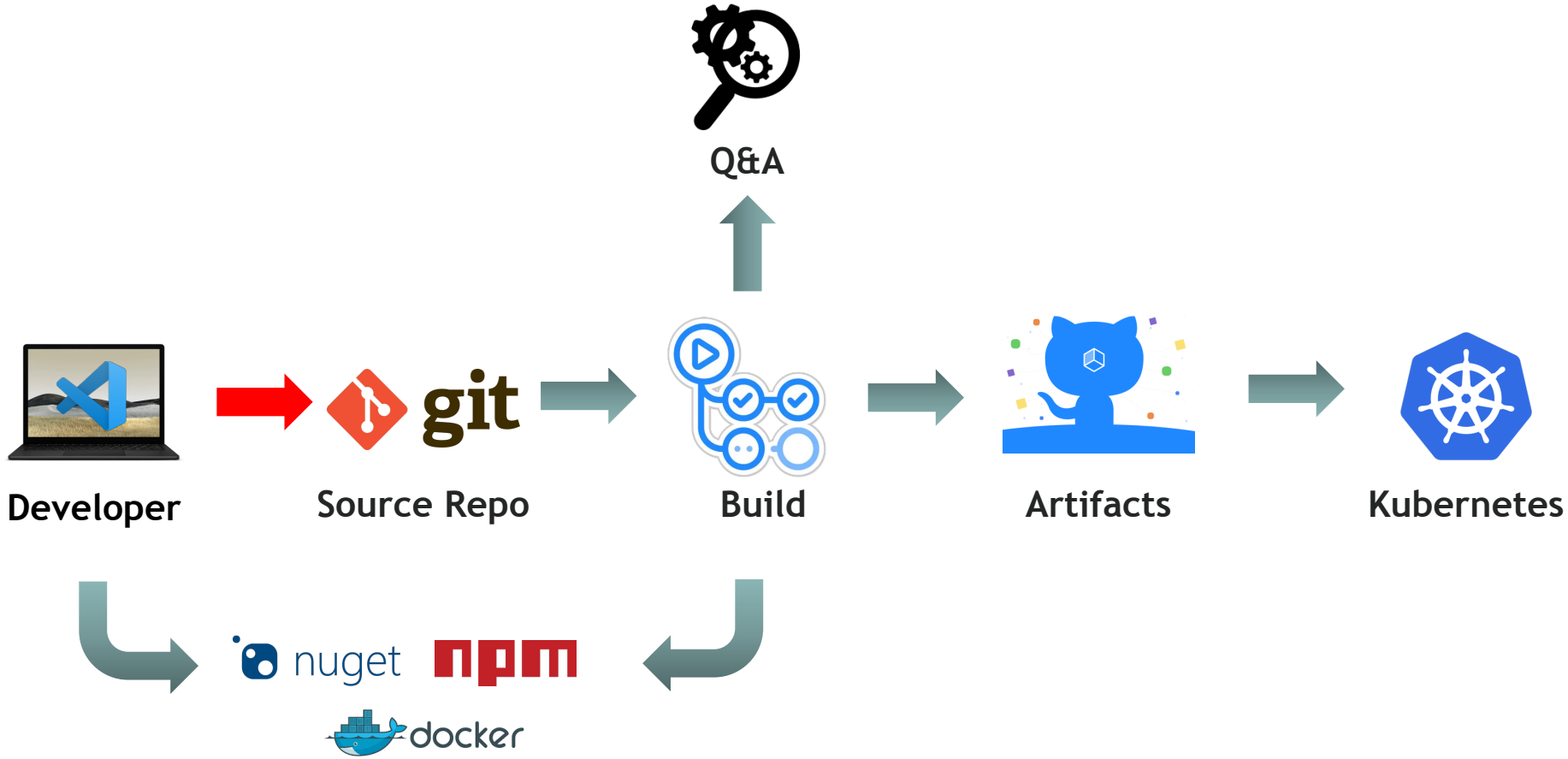


https://xkcd.com/2347/

FUTURE TECH

@nielstanis

Software Supply Chain



FUTURE TECH

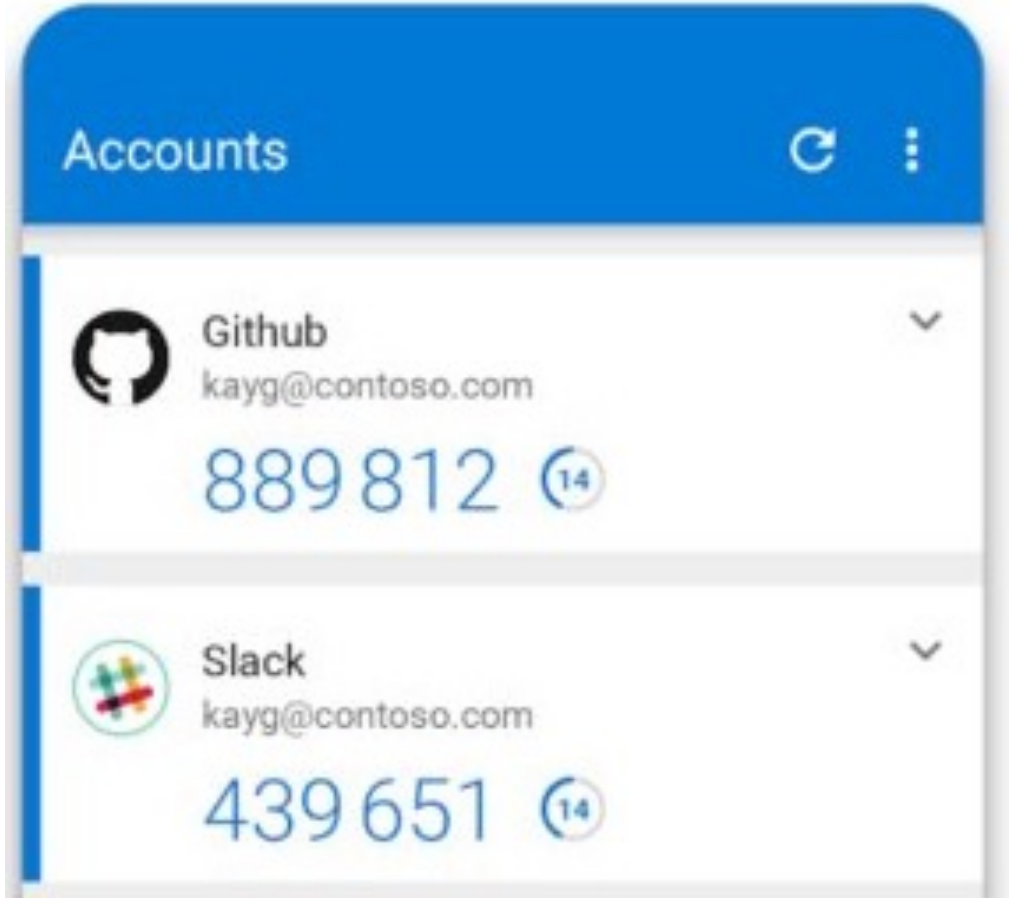
@nielstanis

GitHub account

FUTURE TECH

@nielstanis

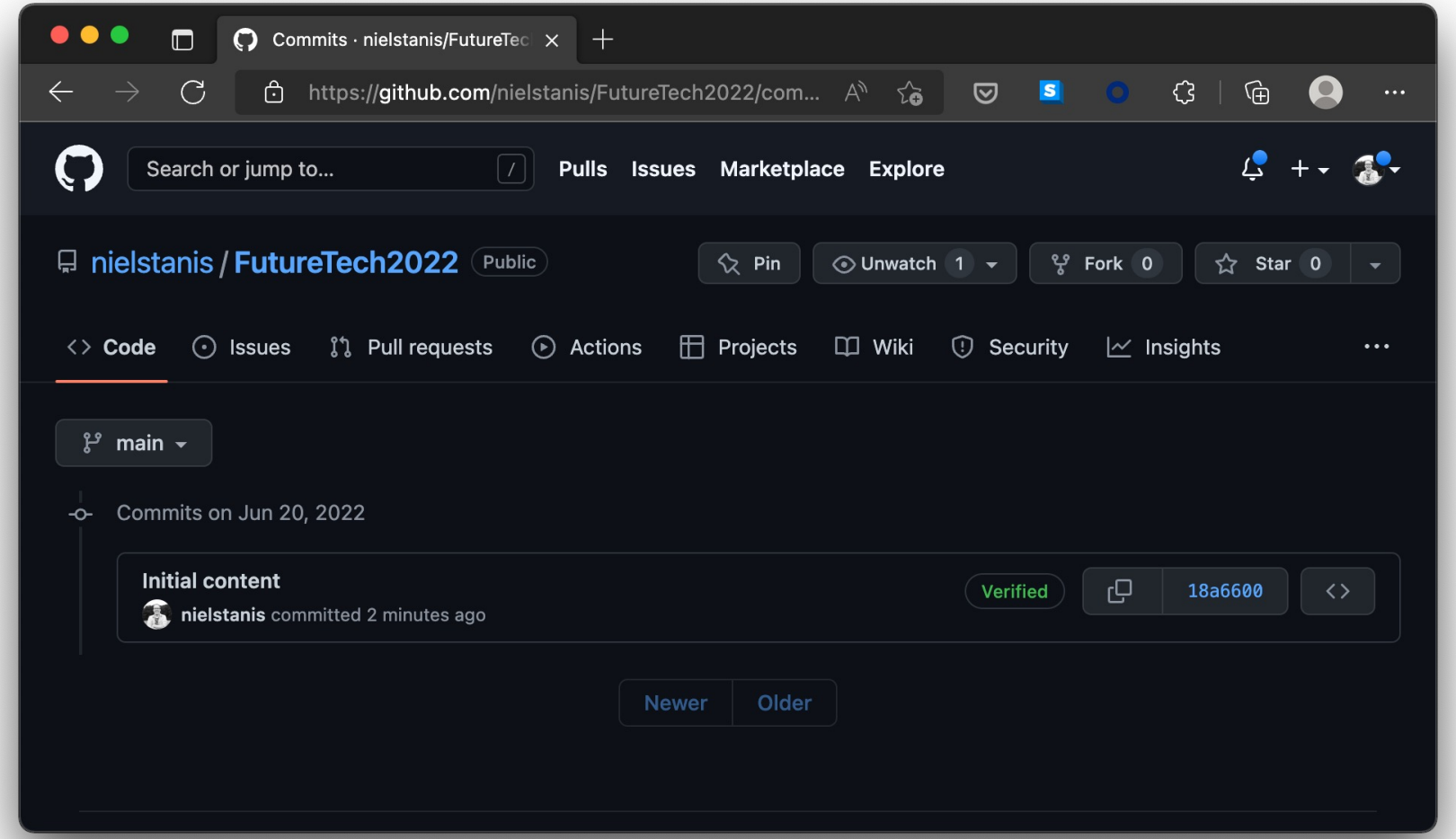
Use MFA on source-repository



FUTURE TECH

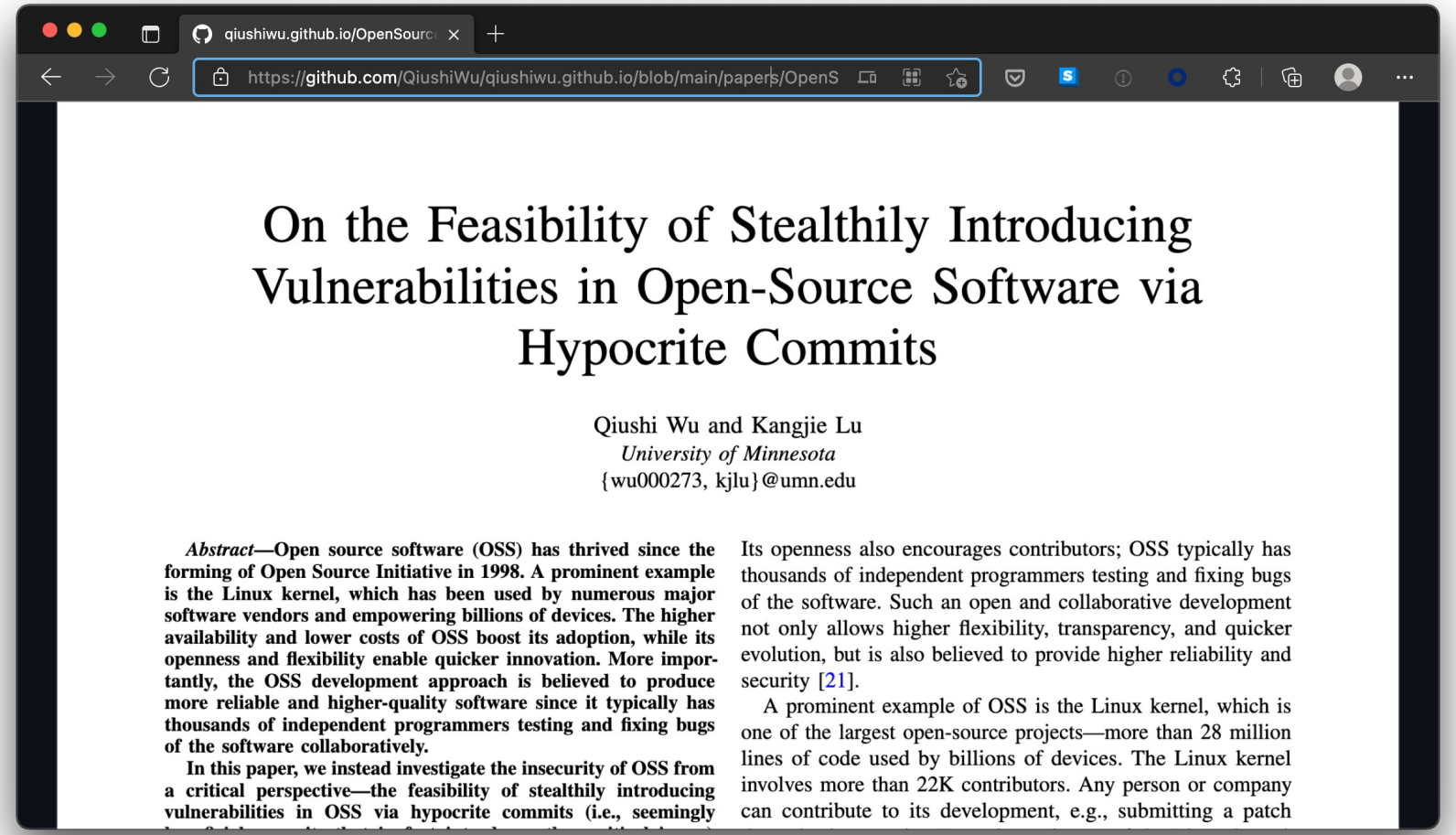
@nielstanis

GIT Commit Signing



@nielstanis

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 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

FUTURE TECH

@nielstanis

Octopus Scanner - NetBeans

May 28, 2020

The Octopus Scanner Malware: Attacking the open source supply chain

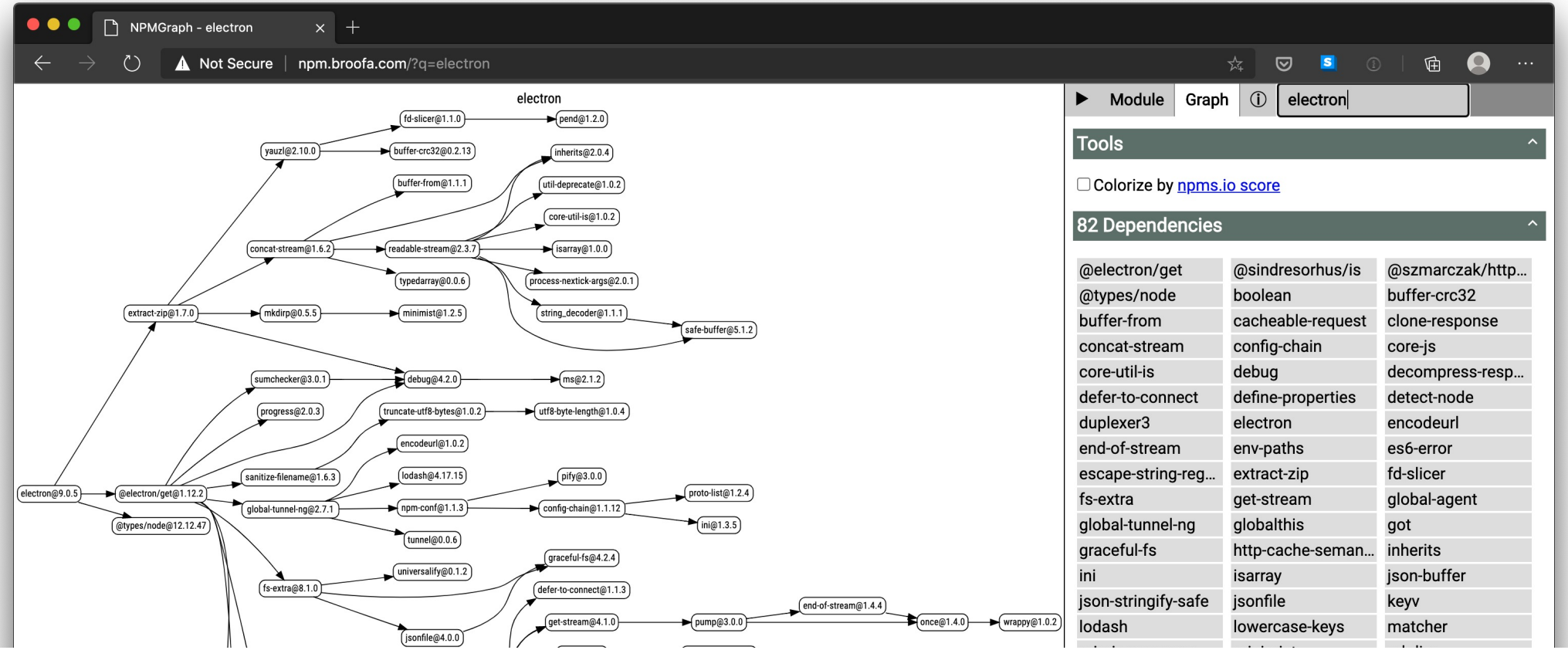
Alvaro Muñoz

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

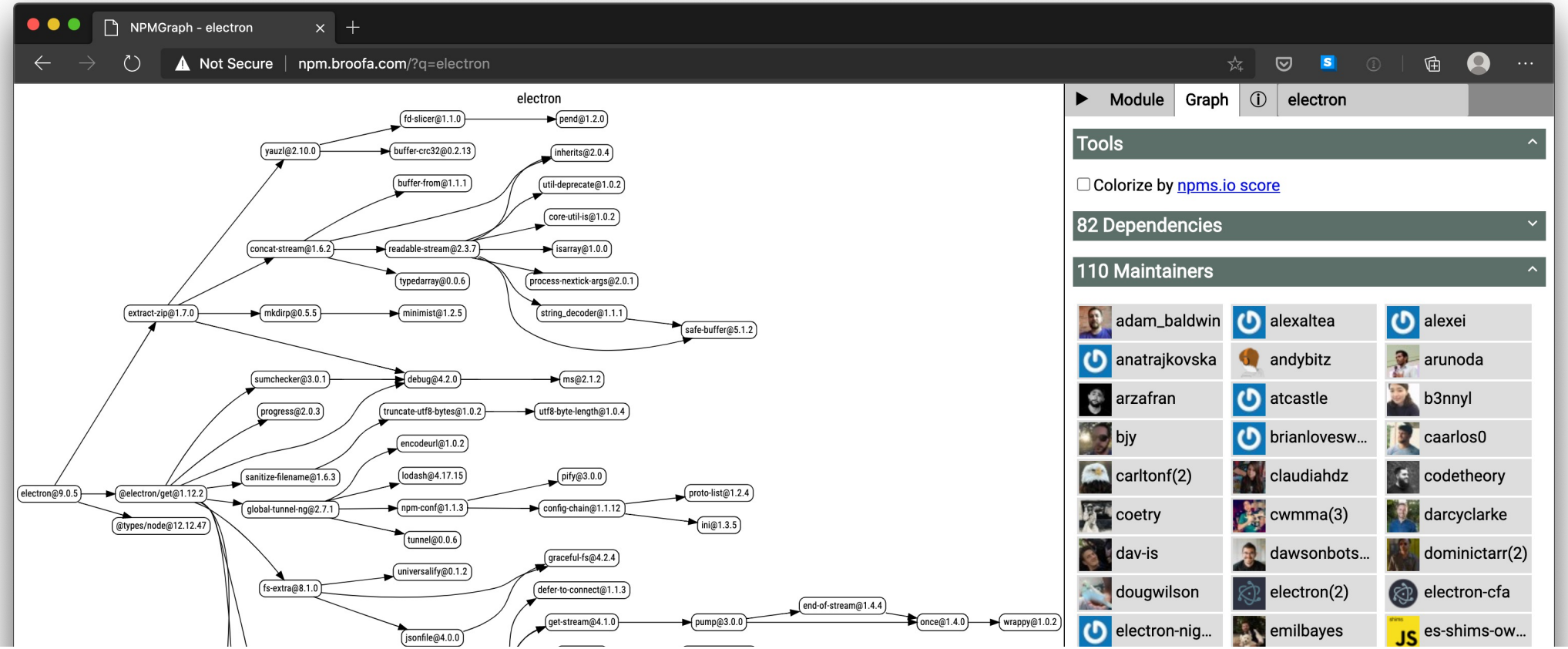
FUTURE TECH

@nielstanis

Visual Studio Code



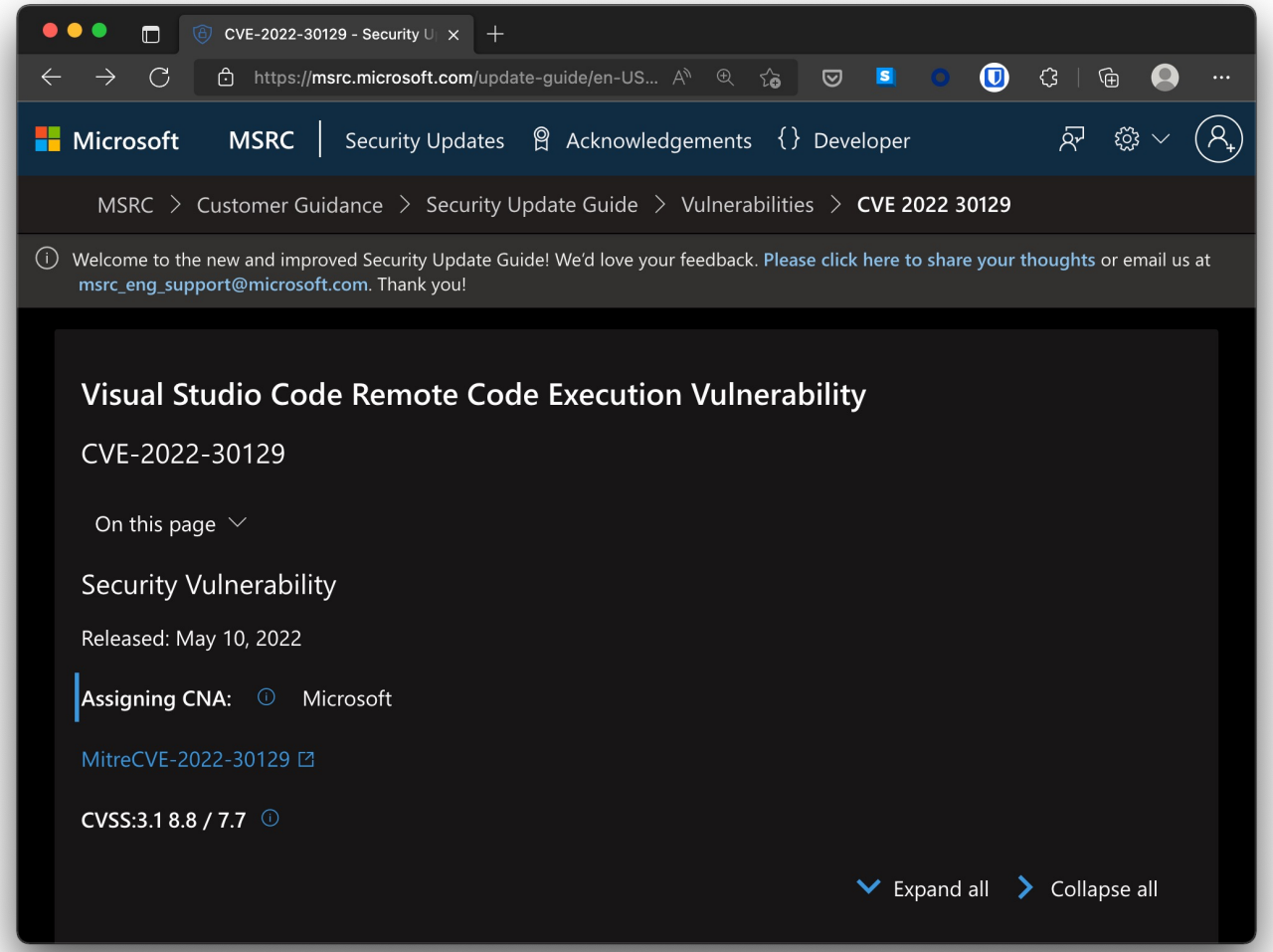
Visual Studio Code



FUTURE TECH

@nielstanis

Visual Studio Code



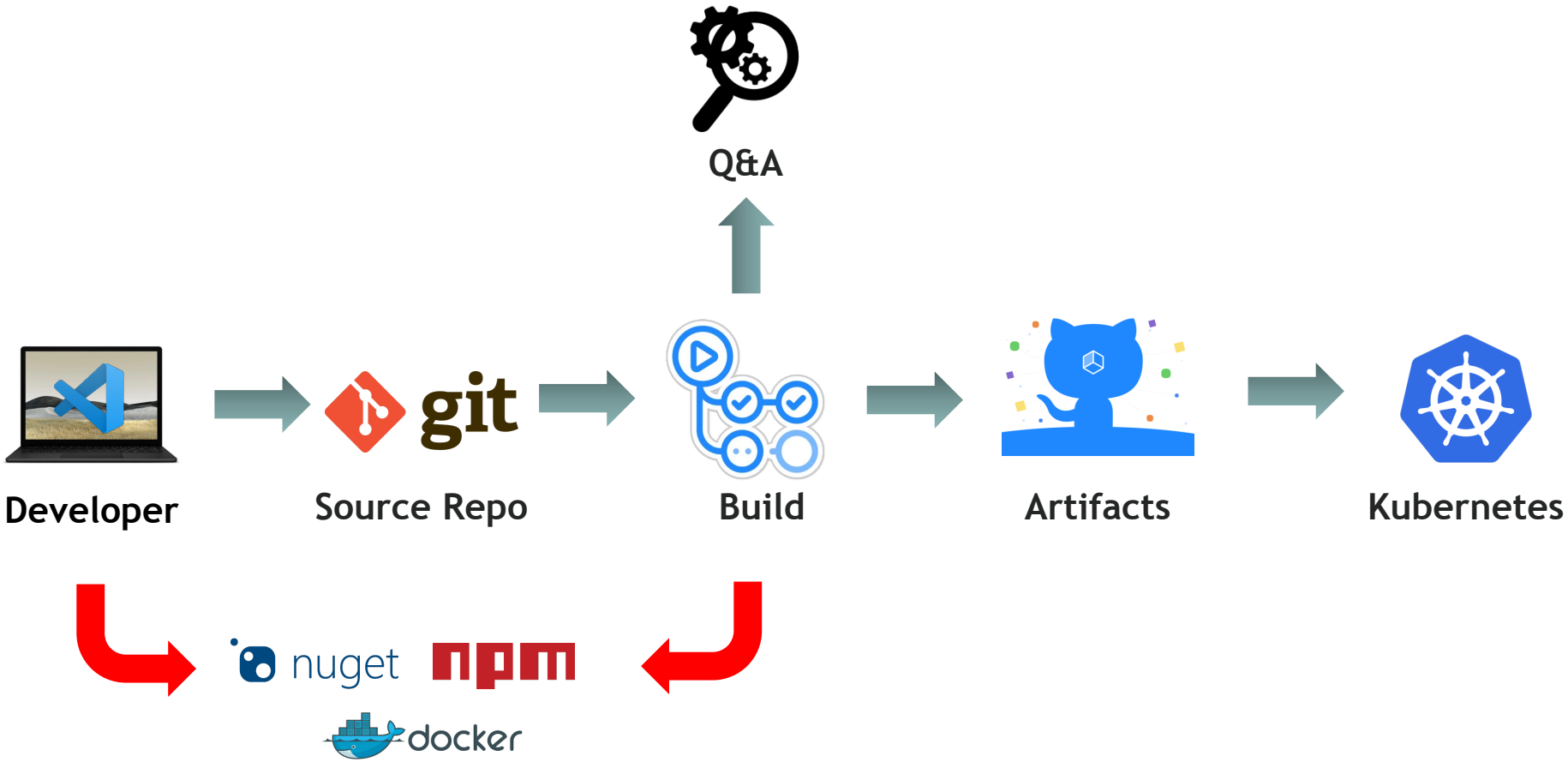
@nielstanis

Visual Studio Code

FUTURE TECH

@nielstanis

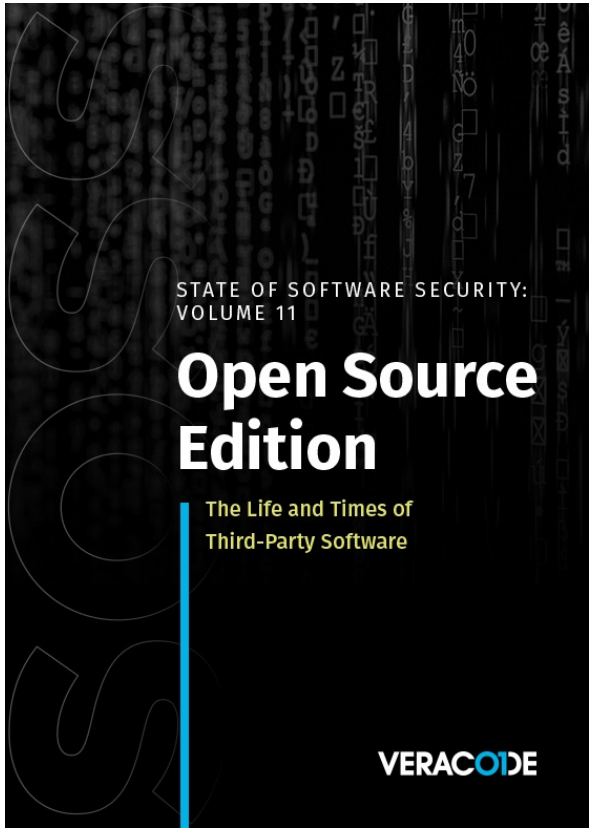
3rd Party Libraries



FUTURE TECH

State Of Software Security v11 2021

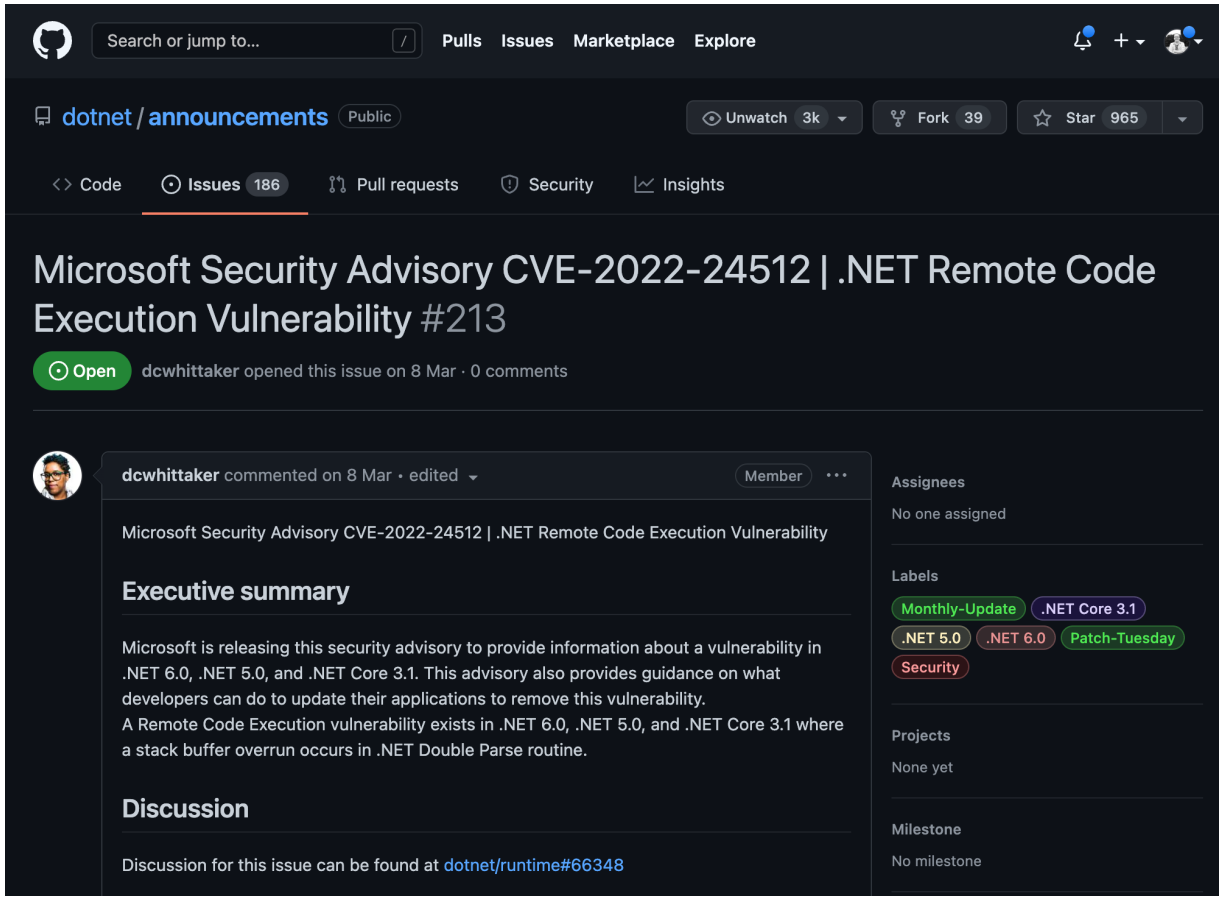
”Despite this dynamic landscape, 79 percent of the time, developers never update third-party libraries after including them in a codebase.”



FUTURE TECH

@nielstanis

Vulnerabilities in libraries



FUTURE TECH

@nielstanis

Vulnerabilities in libraries

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.

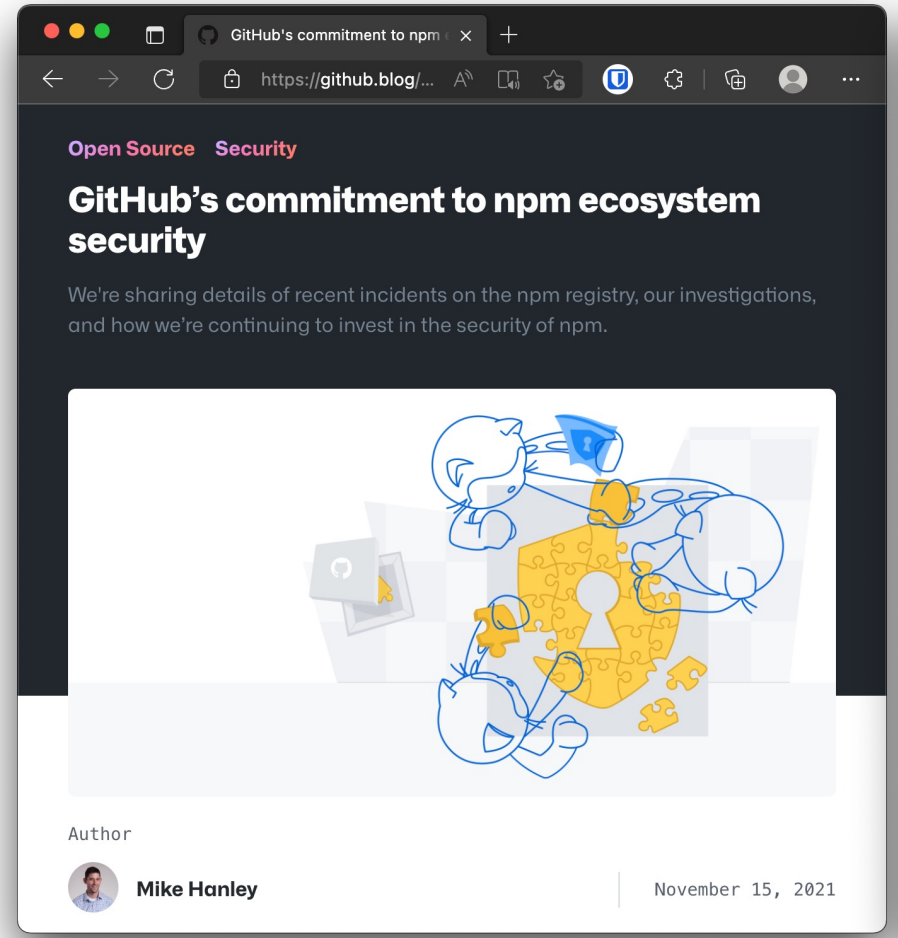
CISA urges users and administers using compromised ua-parser-js versions 0.7.29, 0.8.0, and 1.0.0 to update to the respective patched versions: 0.7.30, 0.8.1, 1.0.1

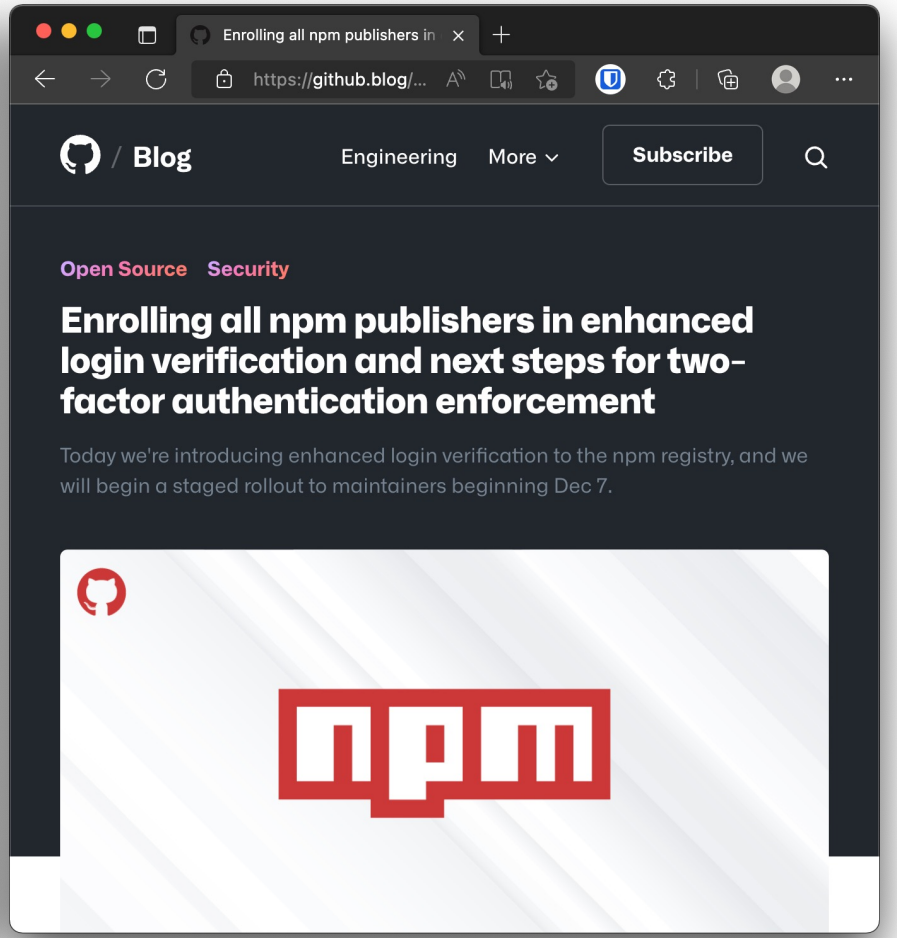
For more information, see Embedded malware in ua-parser-js.

FUTURE TECH

@nielstanis

Vulnerabilities in libraries

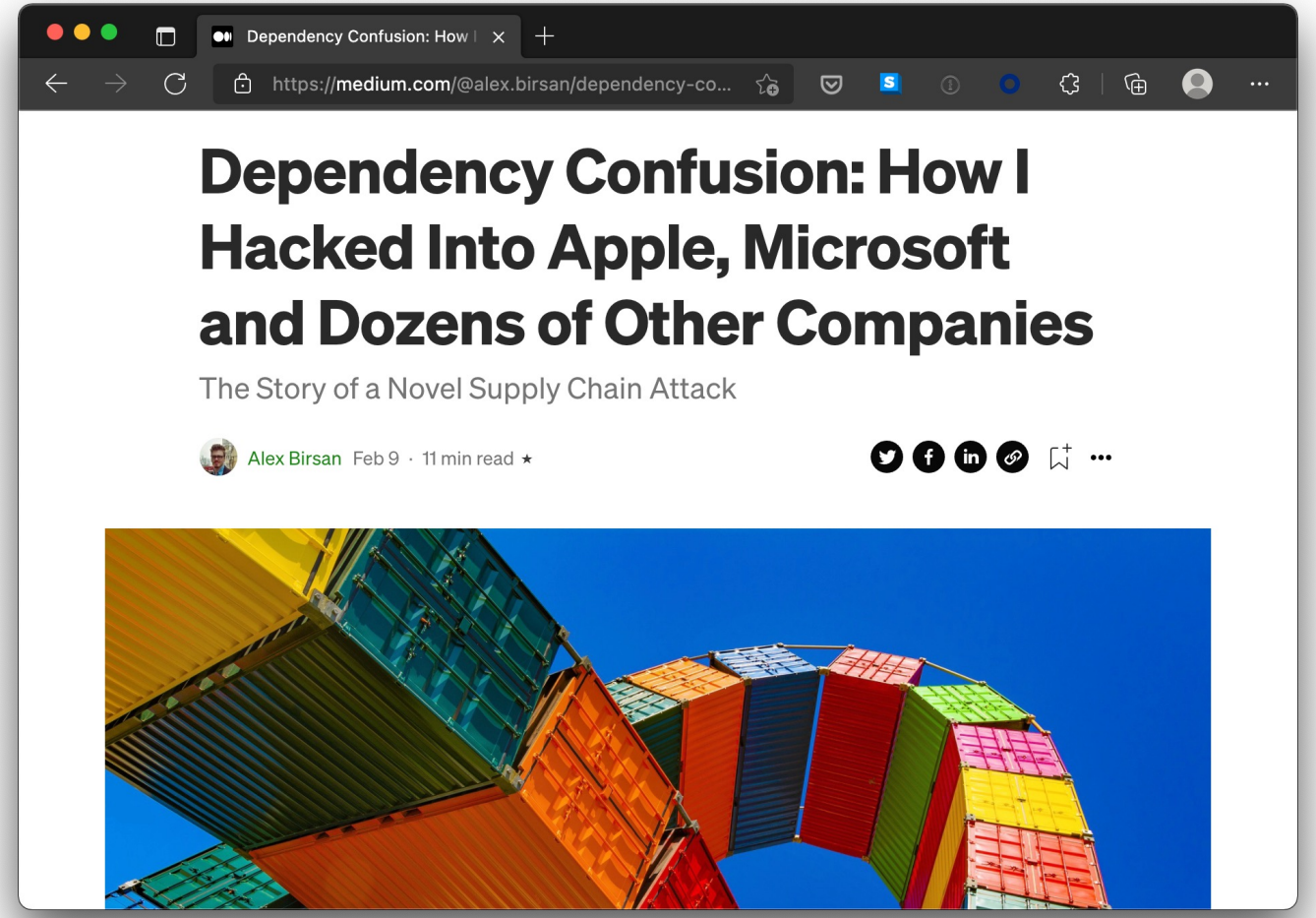




FUTURE TECH

@nielstanis

Dependency Confusion



FUTURE TECH

@nielstanis

Microsoft Whitepaper

3 ways to mitigate risk when using private package feeds

Secure Your Hybrid Software Supply Chain

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

@nielstanis

Dependency Confusion - NuGet

- Use single private repository
- Azure Artifacts can help manage and control upstream
- If your company has public packages consider registering prefix
- Lock packages with config

FUTURE TECH

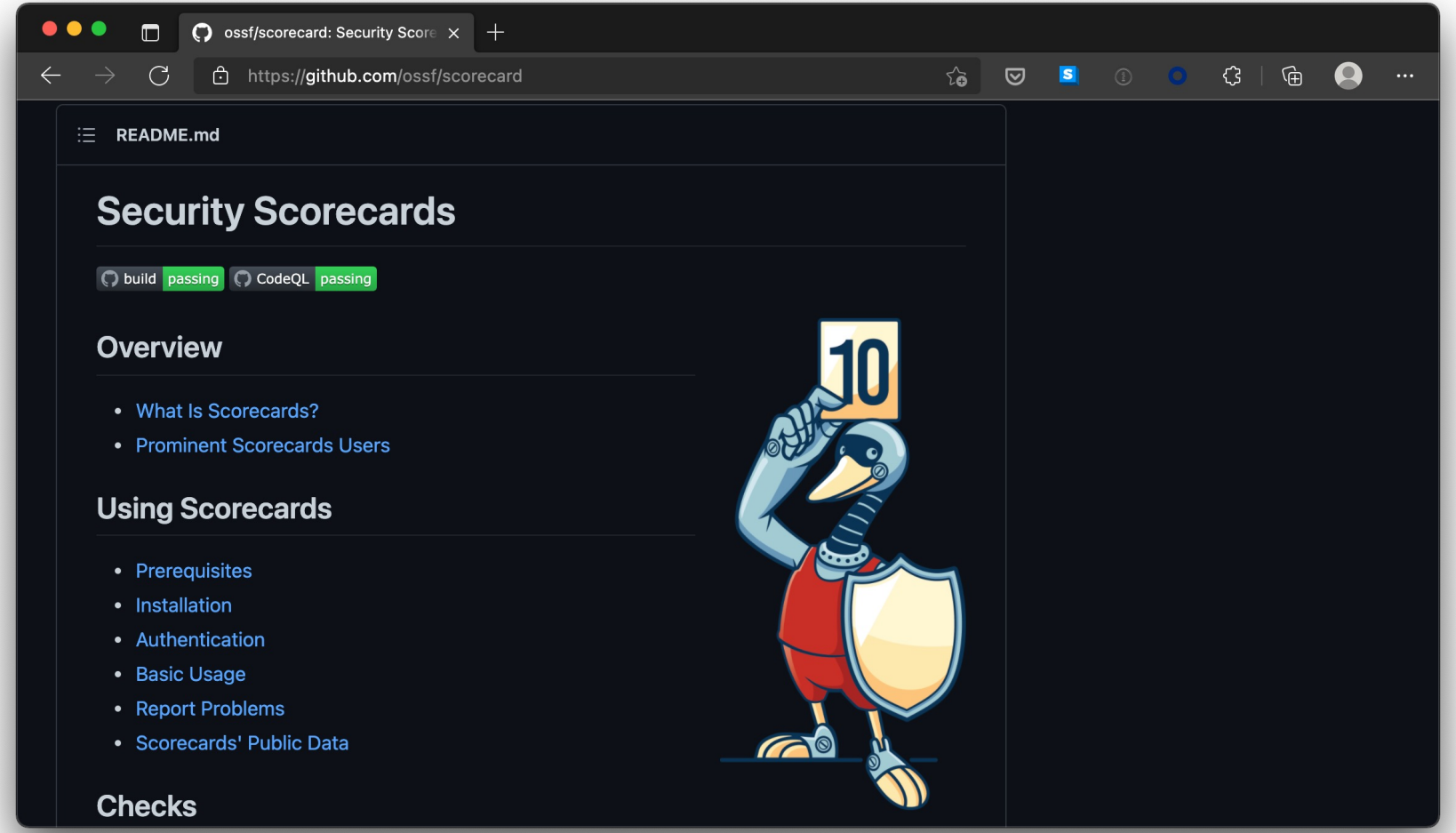
3rd Party Libraries

- 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

FUTURE TECH

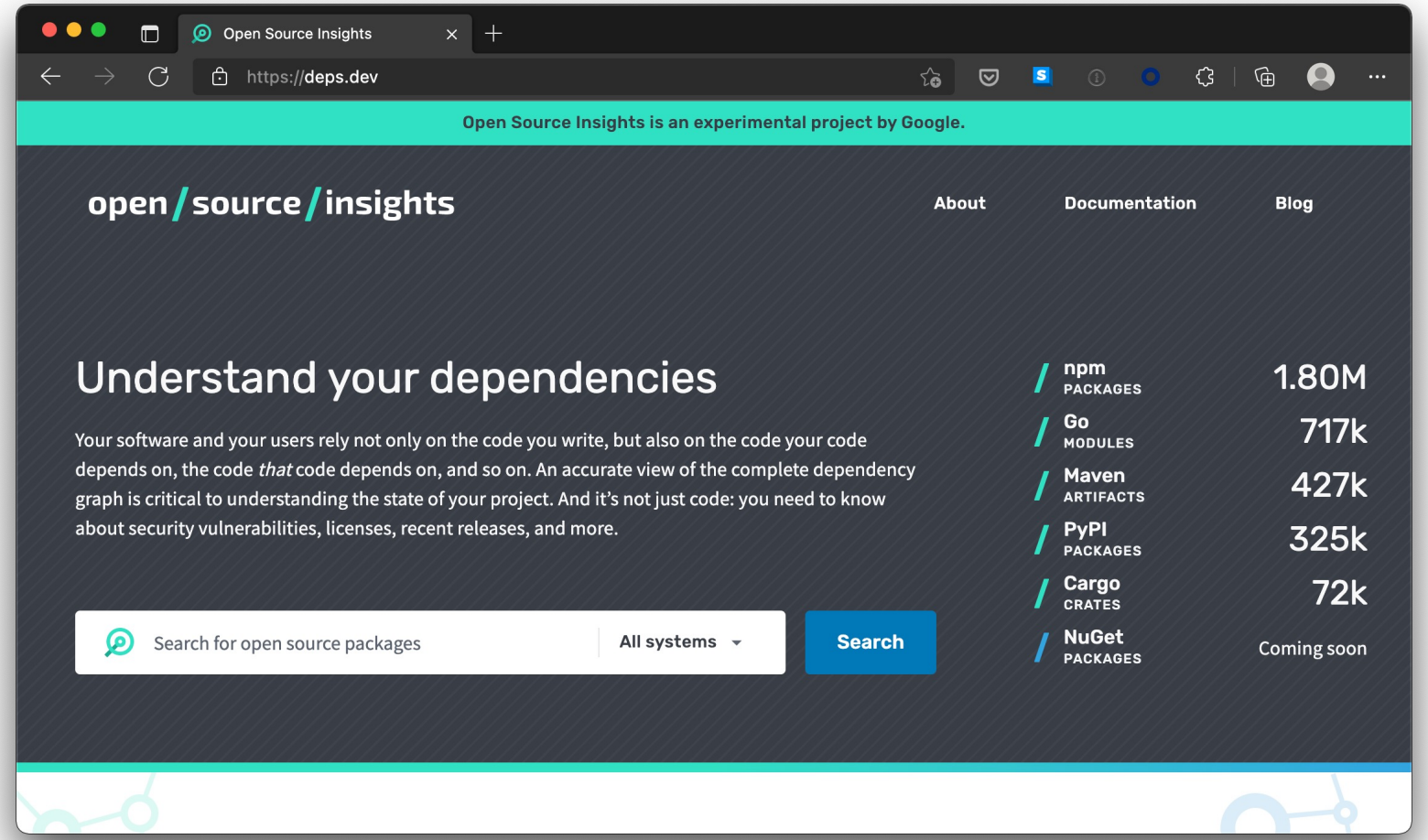
Security Scorecards - OpenSSF



FUTURE TECH

@nielstanis

Deps.Dev by Google



FUTURE TECH

@nielstanis

Deps.Dev by Google

electron/electron
GitHub

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

14k forks ★ 102k stars

OpenSSF scorecard

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

View information about checks and how to fix failures.

SCORE
5.8/10

Scorecard as of April 25, 2022.

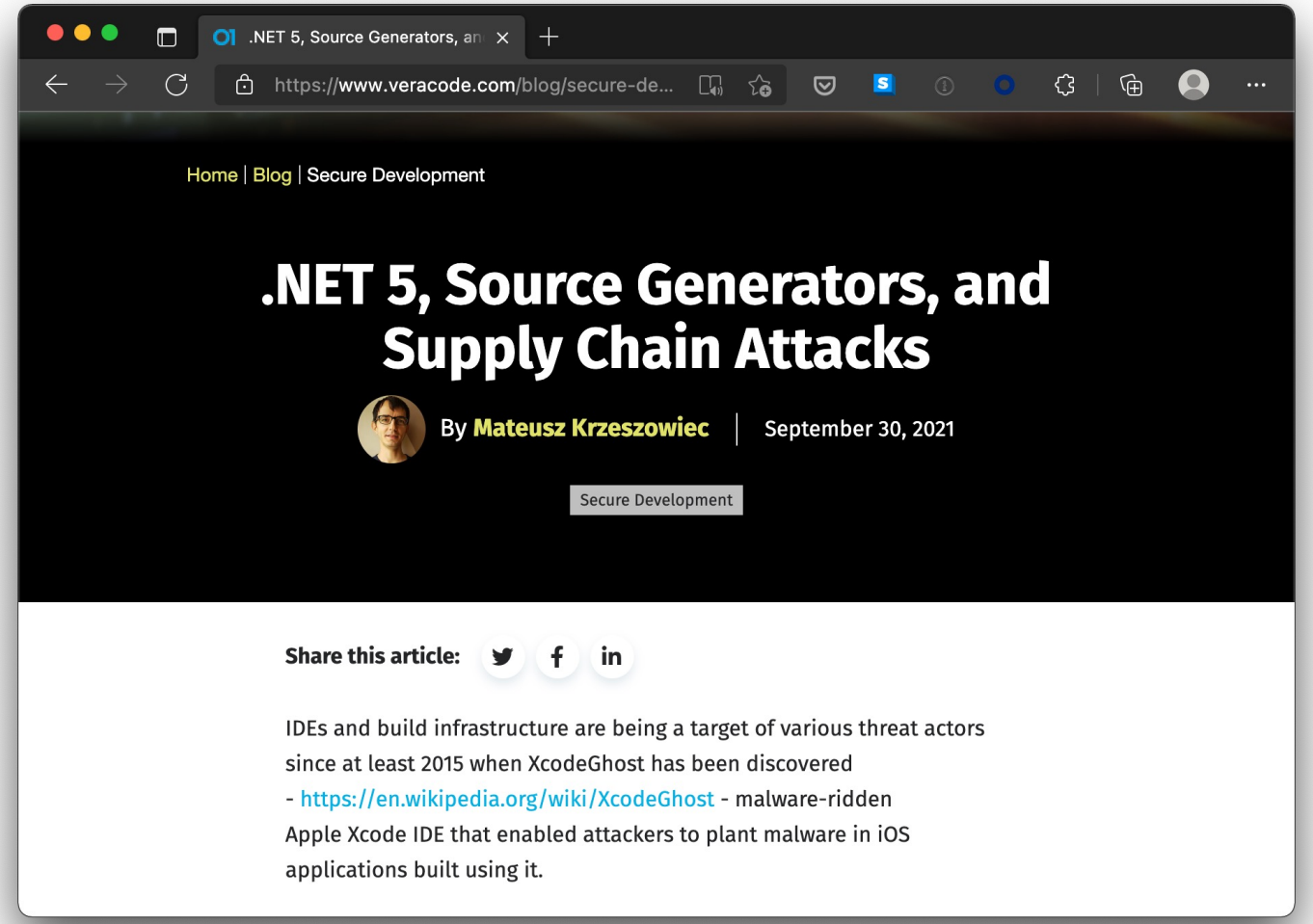
Code-Review	5/10
Maintained	10/10
CII-Best-Practices	0/10
Vulnerabilities	10/10
Dependency-Update-Tool	0/10
Security-Policy	10/10
Dangerous-Workflow	10/10
Token-Permissions	0/10
License	10/10
Pinned-Dependencies	8/10
Binary-Artifacts	10/10
Fuzzing	0/10
Signed-Releases	0/10

Project metadata as of May 7, 2022.

FUTURE TECH

@nielstanis

Source Generators



FUTURE TECH

@nielstanis

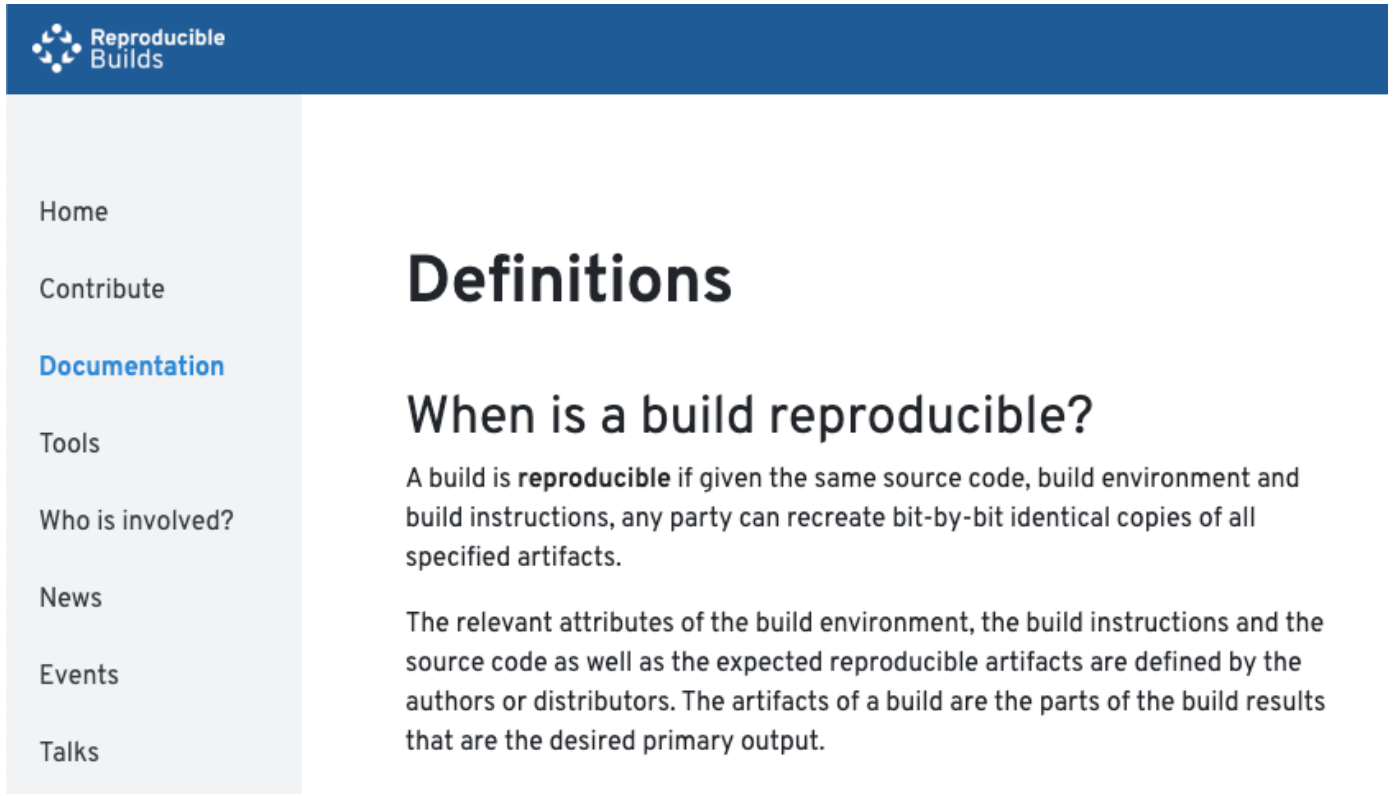
Source Generators

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

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

FUTURE TECH

Reproducible/Deterministic Builds



FUTURE TECH

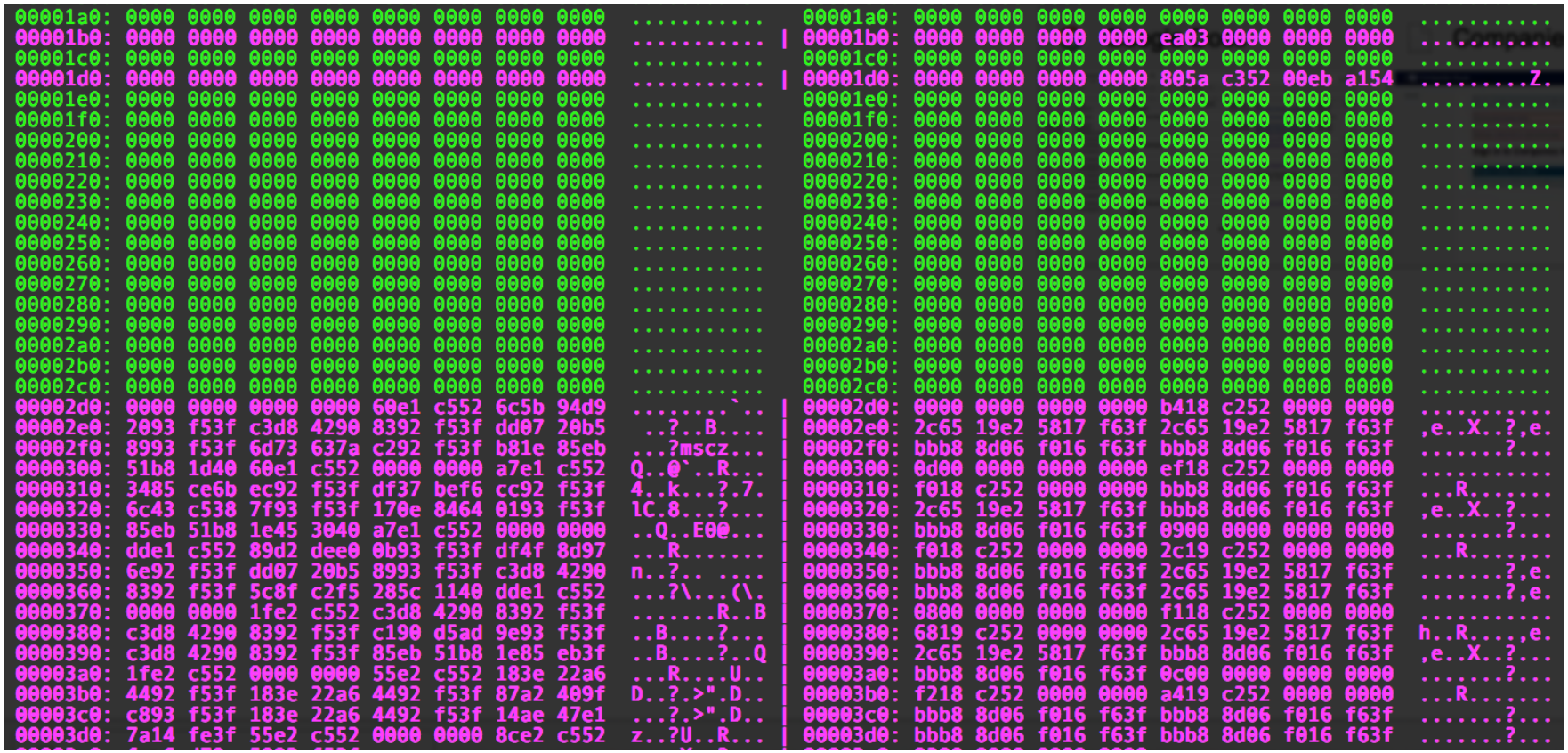
@nielstanis

Reproducible/Deterministic Builds

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

FUTURE TECH

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  .........Z.R...T
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 0000 60e1 c552 6c5b 94d9  ........`..R.....| 00002d0: 0000 0000 0000 0000 b418 c252 0000 0000  ...........R....
00002e0: 2093 f53f c3d8 4290 8392 f53f dd07 20b5  ...?..B....?.... | 00002e0: 2c65 19e2 5817 f63f 2c65 19e2 5817 f63f  ,e..X..?,e..X..?
00002f0: 8993 f53f 6d73 637a c292 f53f b81e 85eb  ...?mscz...?....| 00002f0: bbb8 8d06 f016 f63f bbb8 8d06 f016 f63f  .......?.......?
0000300: 51b8 1d40 60e1 c552 0000 0000 a7e1 c552  Q..@`..R.......R| 0000300: 0d00 0000 0000 0000 ef18 c252 0000 0000  ...........R....
0000310: 3485 ce6b ec92 f53f df37 bef6 cc92 f53f  4..k...?.7.....?| 0000310: f018 c252 0000 0000 bbb8 8d06 f016 f63f  ...R.......?....
0000320: 6c43 c538 7f93 f53f 170e 8464 0193 f53f  lC.8...?...d...?| 0000320: 2c65 19e2 5817 f63f bbb8 8d06 f016 f63f  ,e..X..?.......?
0000330: 85eb 51b8 1e45 3040 a7e1 c552 0000 0000  ..Q..E0@...R....| 0000330: bbb8 8d06 f016 f63f 0900 0000 0000 0000  .......?........
0000340: dde1 c552 89d2 dee0 0b93 f53f df4f 8d97  ...R.......?.O..| 0000340: f018 c252 0000 0000 2c19 c252 0000 0000  ...R....,..R....
0000350: 6e92 f53f dd07 20b5 8993 f53f c3d8 4290  n..?.. ....?..B.| 0000350: bbb8 8d06 f016 f63f 2c65 19e2 5817 f63f  .......?,e..X..?
0000360: 8392 f53f 5c8f c2f5 285c 1140 dde1 c552  ...?\...(\.@...R| 0000360: bbb8 8d06 f016 f63f 2c65 19e2 5817 f63f  .......?,e..X..?
0000370: 0000 0000 1fe2 c552 c3d8 4290 8392 f53f  .......R..B....?| 0000370: 0800 0000 0000 0000 f118 c252 0000 0000  ...........R....
0000380: c3d8 4290 8392 f53f c190 d5ad 9e93 f53f  ..B....?.......?| 0000380: 6819 c252 0000 0000 2c65 19e2 5817 f63f  h..R....,e..X..?
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..R.>".| 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.......R....
00003c0: c893 f53f 183e 22a6 4492 f53f 14ae 47e1  ...?.>".D..?..G.| 00003c0: bbb8 8d06 f016 f63f bbb8 8d06 f016 f63f  .......?.......?
00003d0: 7a14 fe3f 55e2 c552 0000 0000 8ce2 c552  z..?U..R.......R| 00003d0: bbb8 8d06 f016 f63f bbb8 8d06 f016 f63f  .......?.......?
```

@nielstanis

Reproducible/Deterministic Builds

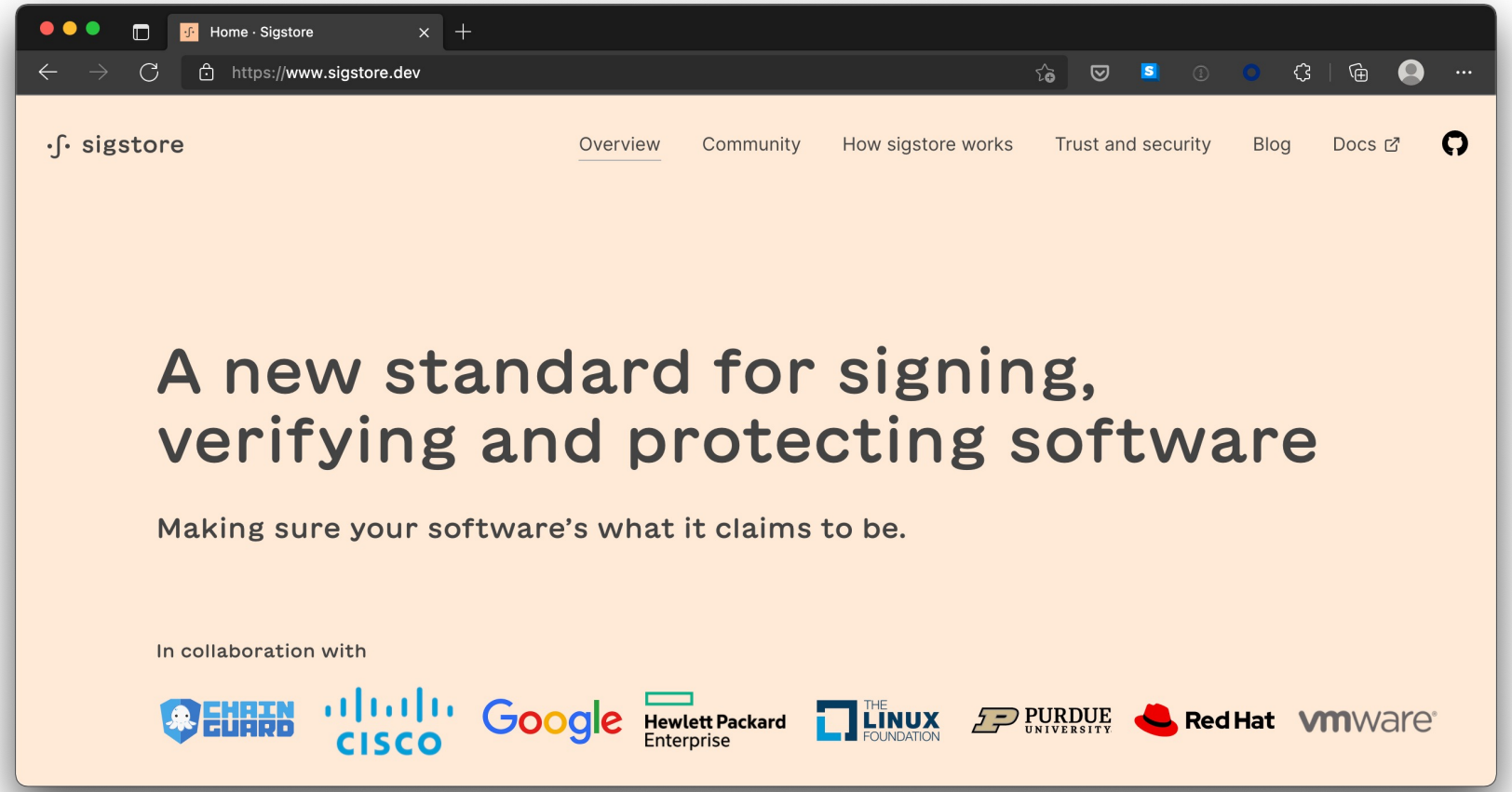
- DotNet.Reproducible NuGet Package
  - MSBuild *ContinuousIntegrationBuild*
  - SourceLink

- 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`

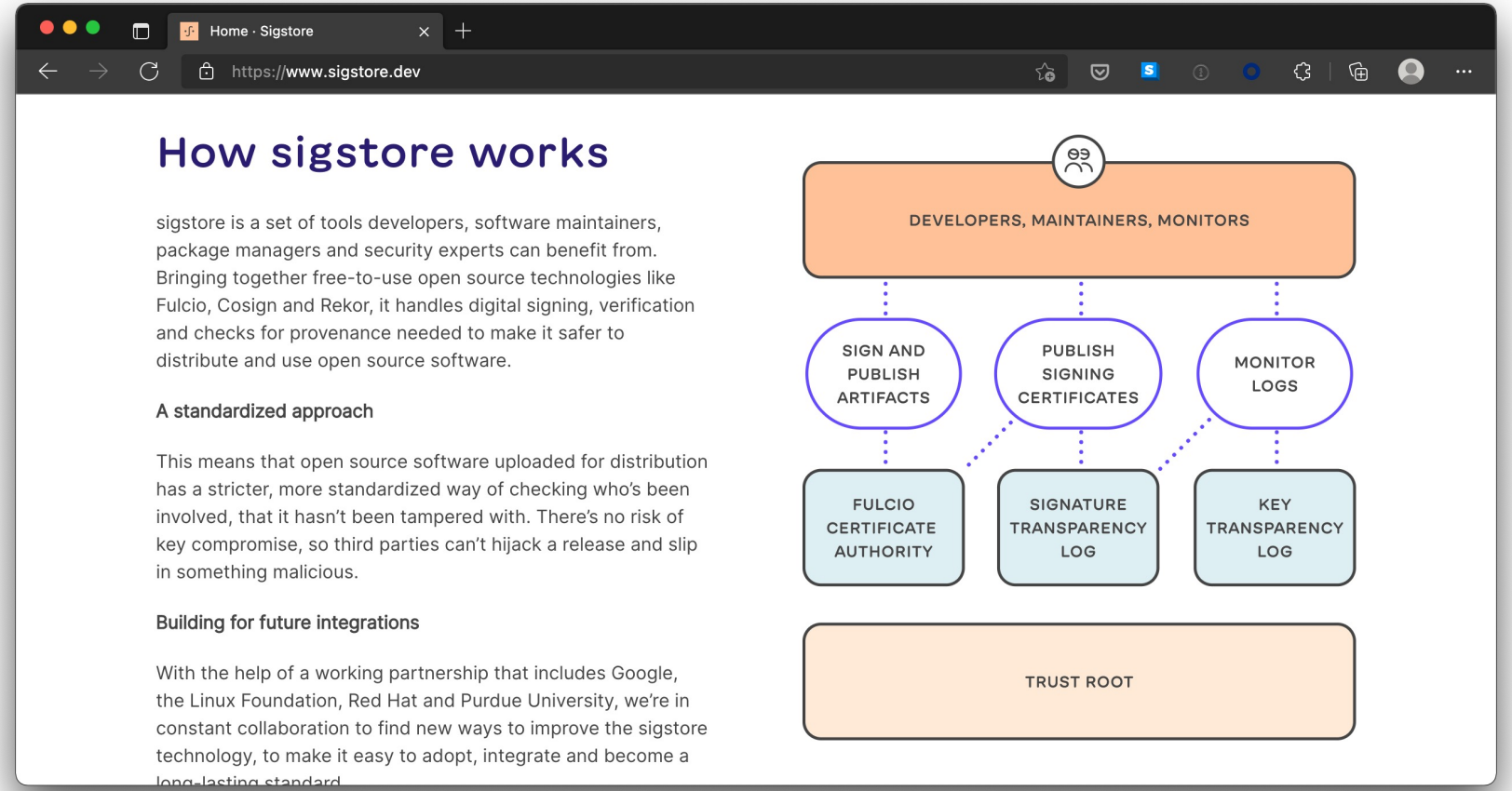
Signing artifacts



FUTURE TECH

@nielstanis

Signing artifacts



FUTURE TECH

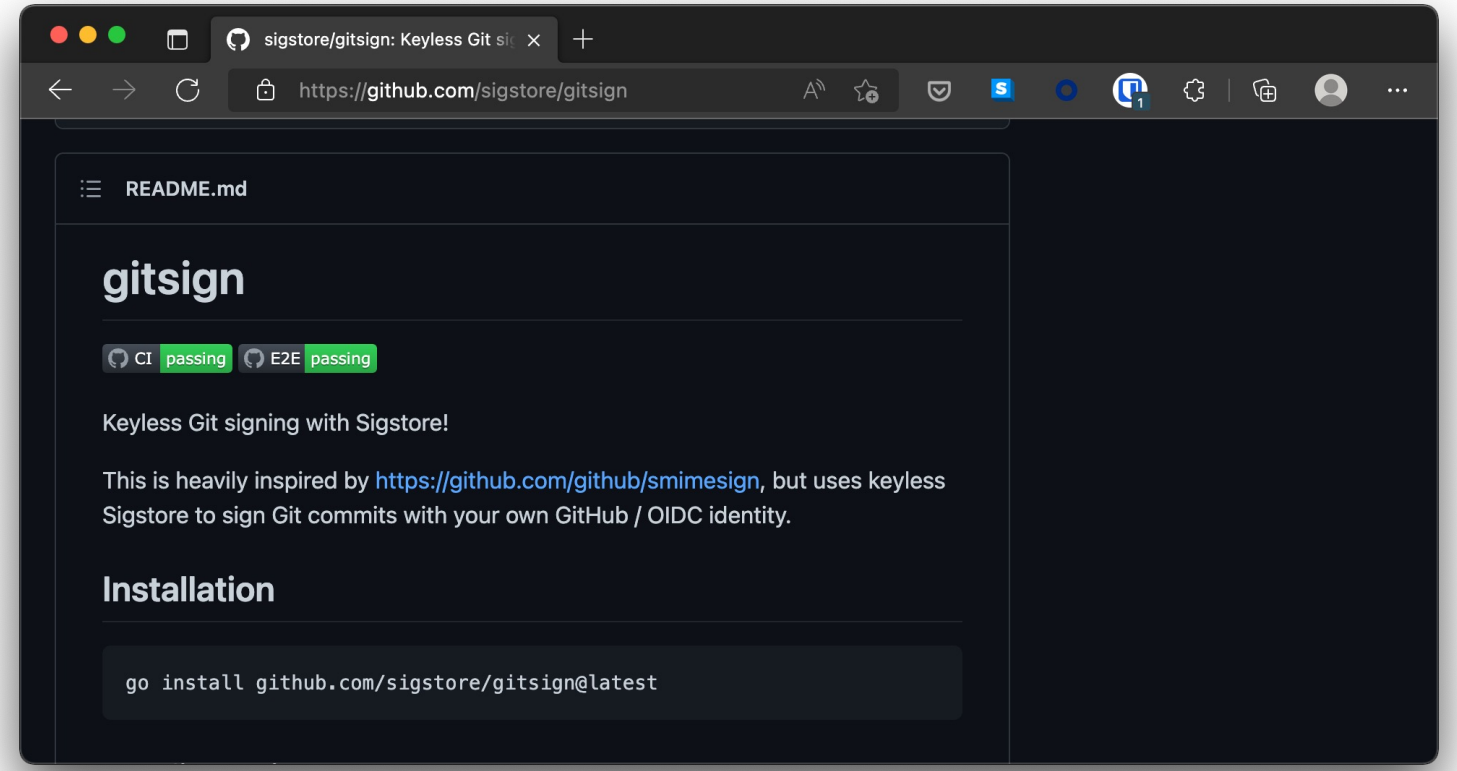
@nielstanis

Signing artifacts

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

FUTURE TECH

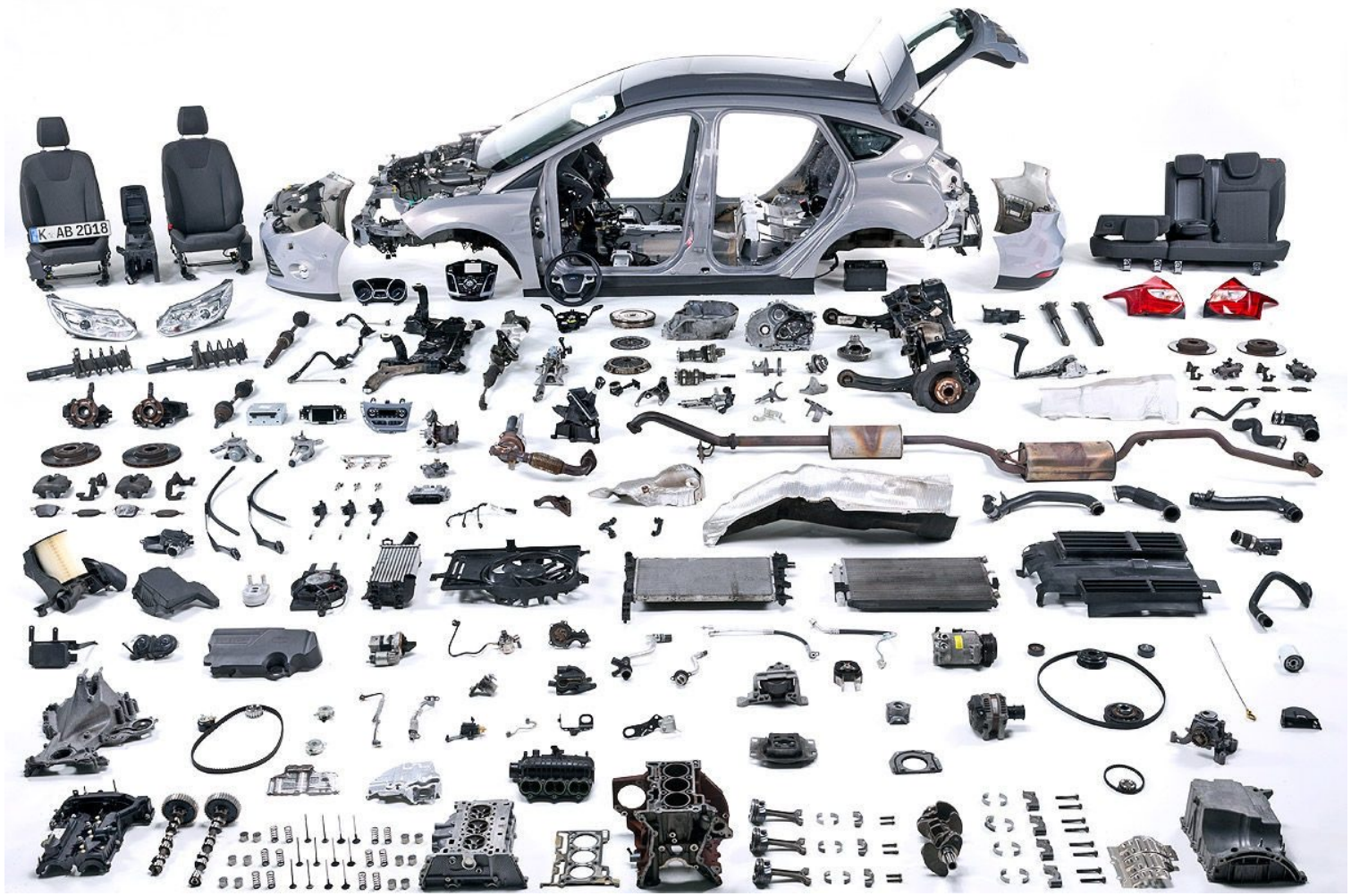
Git Commit Signing Sigstore GitSign



FUTURE TECH

@nielstanis

Automotive Industry



FUTURE TECH

@nielstanis

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

FUTURE TECH

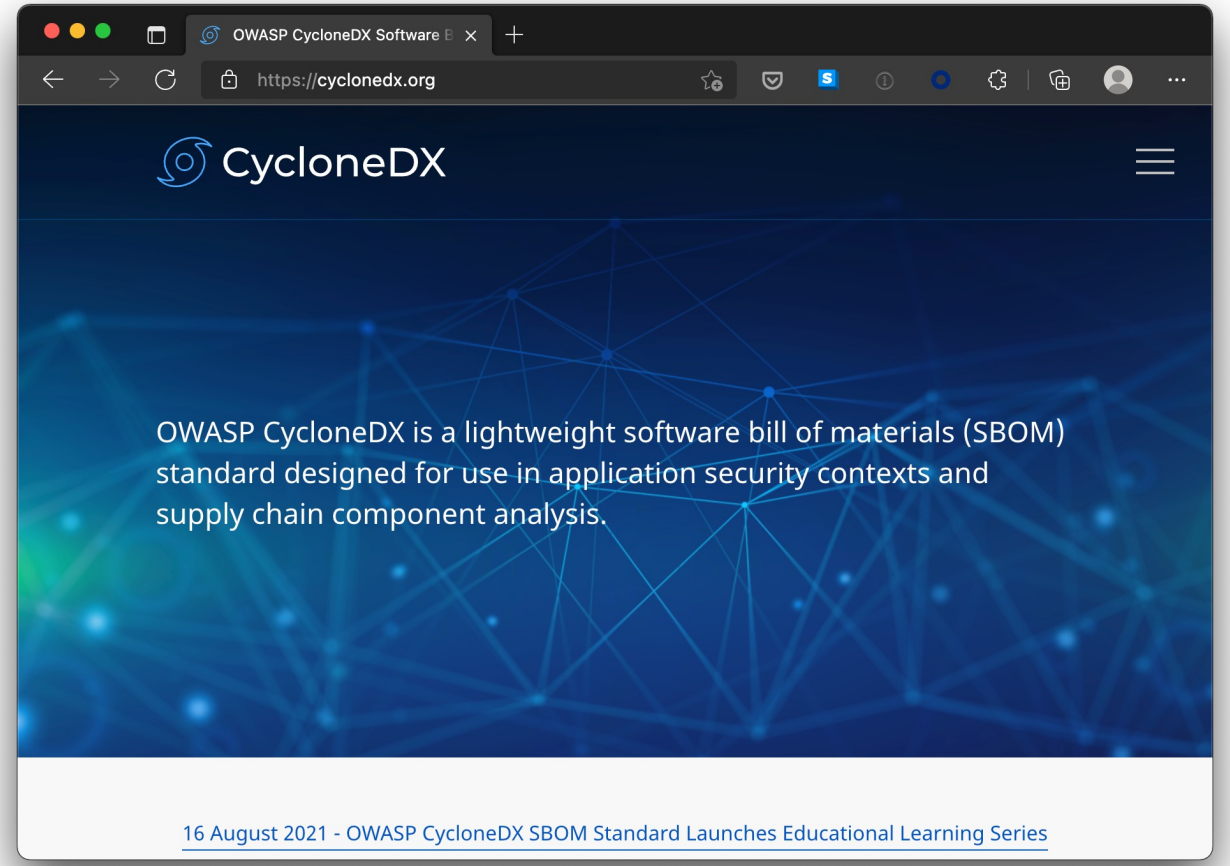
@nielstanis

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?

FUTURE TECH

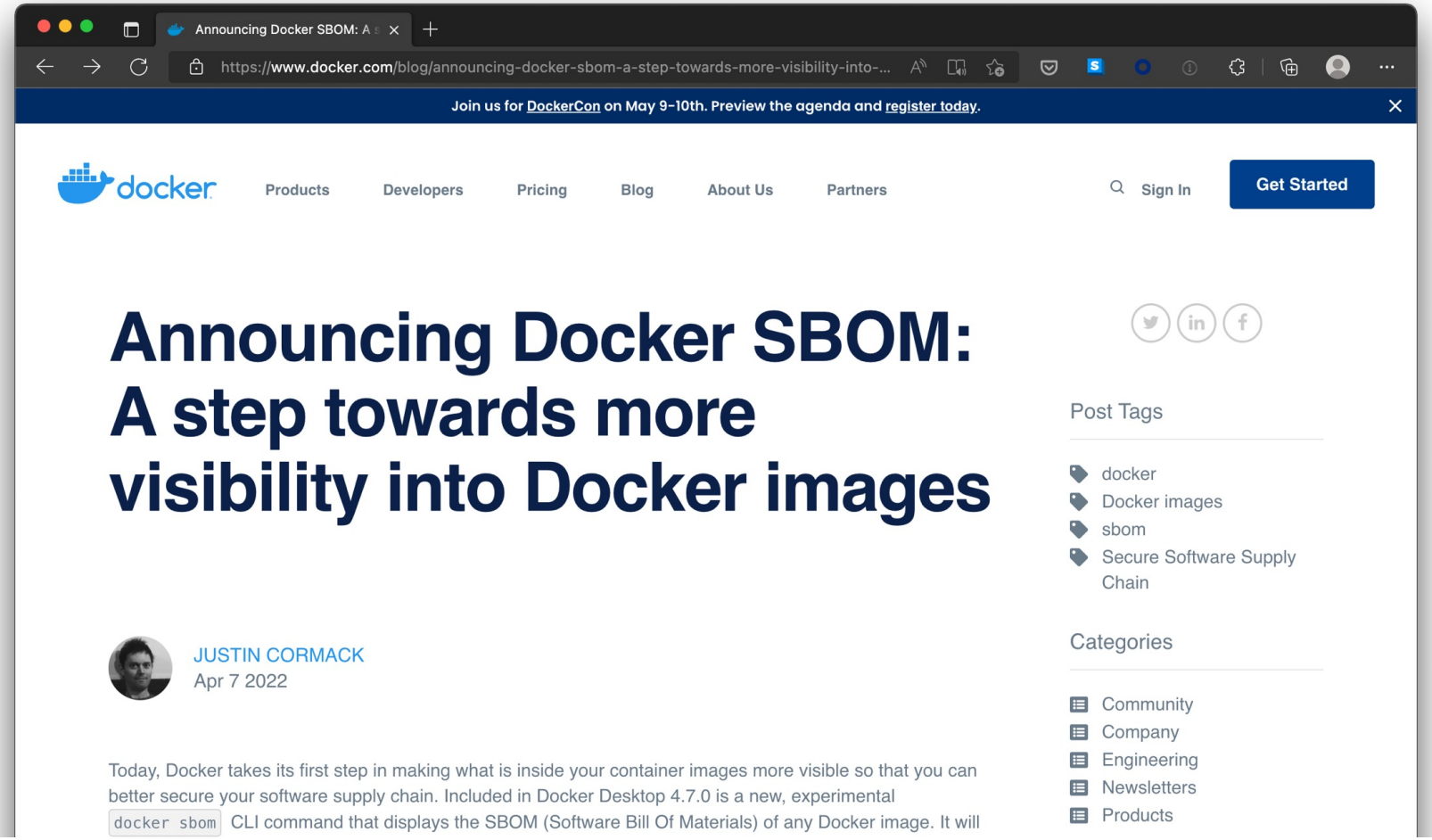
Software Bill of Materials (SBOM)



FUTURE TECH

@nielstanis

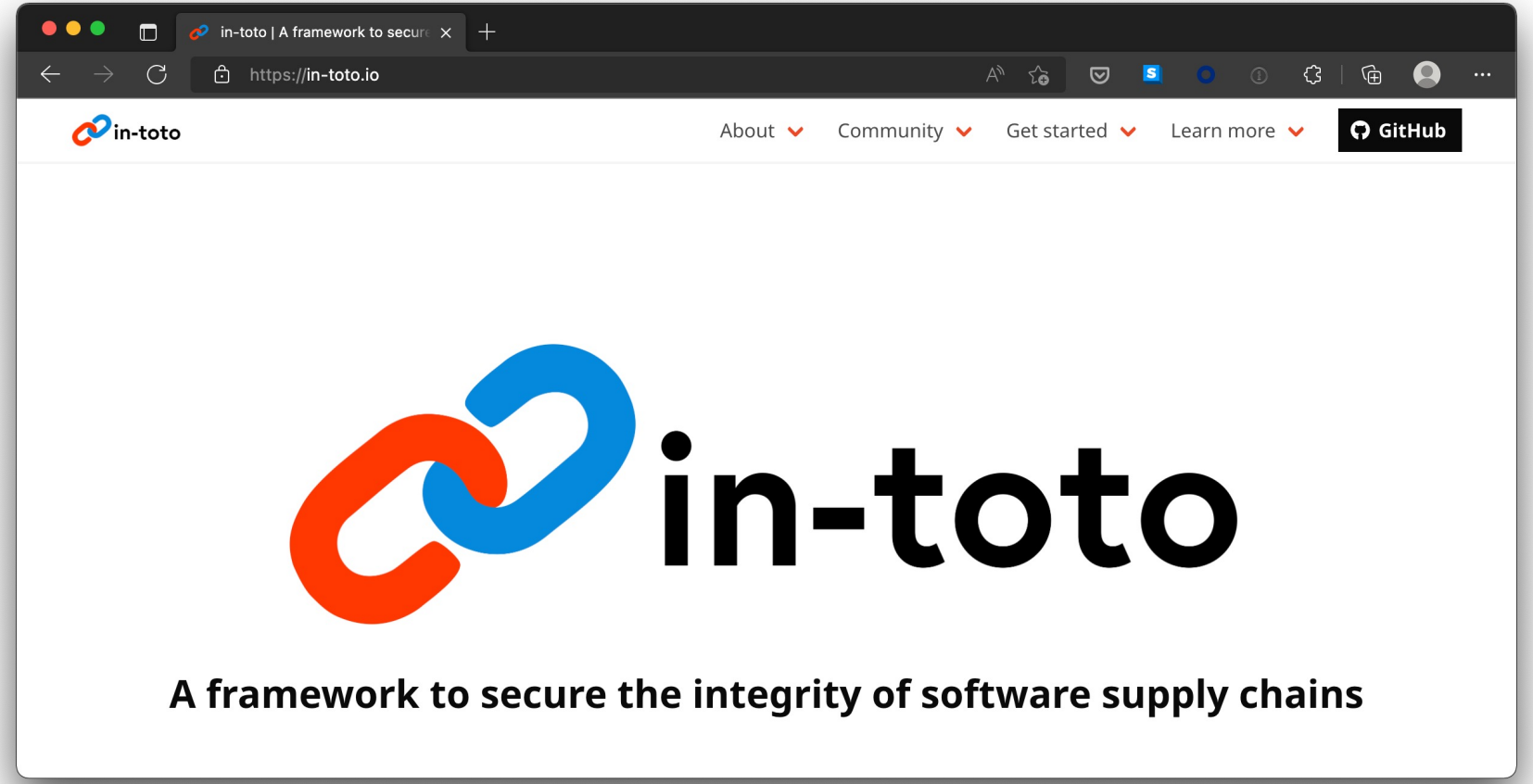
Docker SBOM



FUTURE TECH

@nielstanis

In-toto



FUTURE TECH

@nielstanis

In-Toto - Terminology

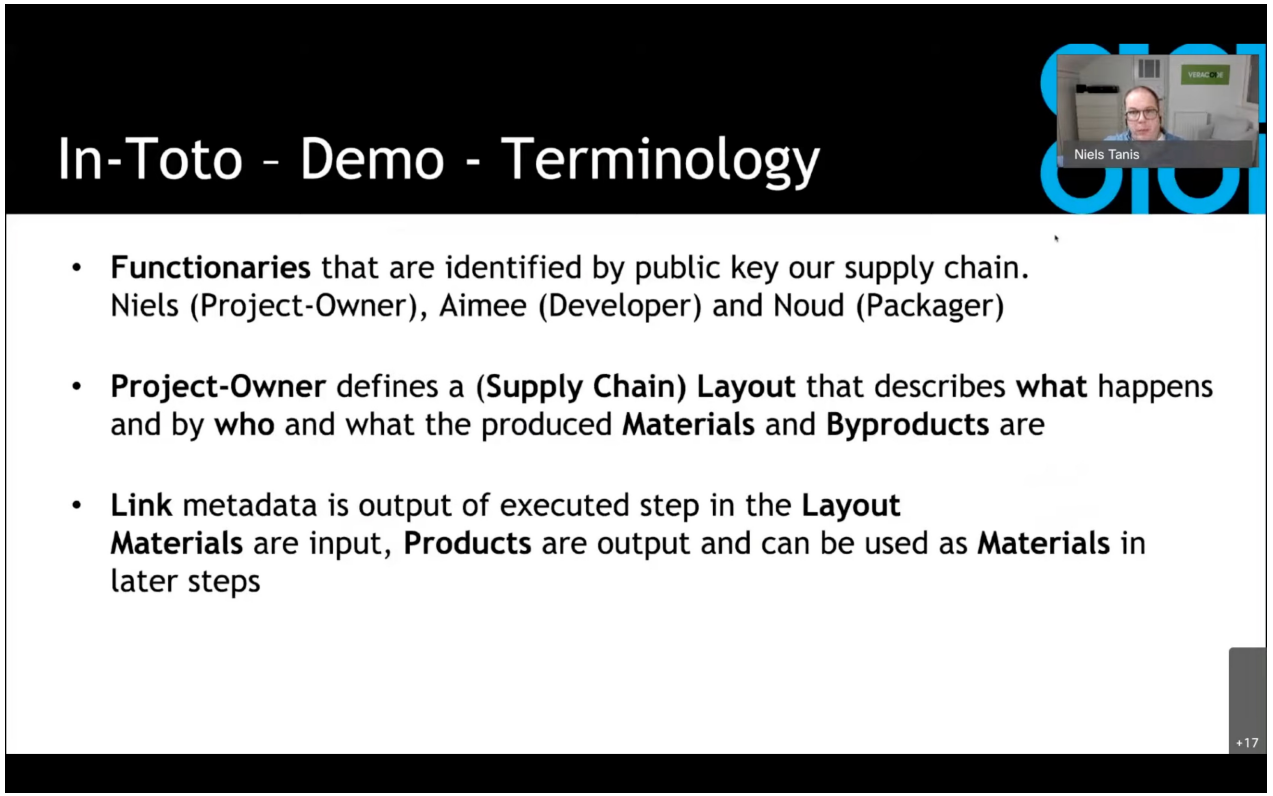
- **Functionaries** that are identified by public key our supply chain.
  For example, the Project-Owner, Developer, and Release Manager

- **Project-Owner** defines a (**Supply Chain) Layout** that describes **what** happens and by **who** and what the produced **Materials** and **Byproducts** are.

- **Link** metadata is output of executed step in the **Layout**
  Materials are input, **Products** are output and can be used as **Materials** in later steps

FUTURE TECH

In-Toto - Demo



FUTURE TECH

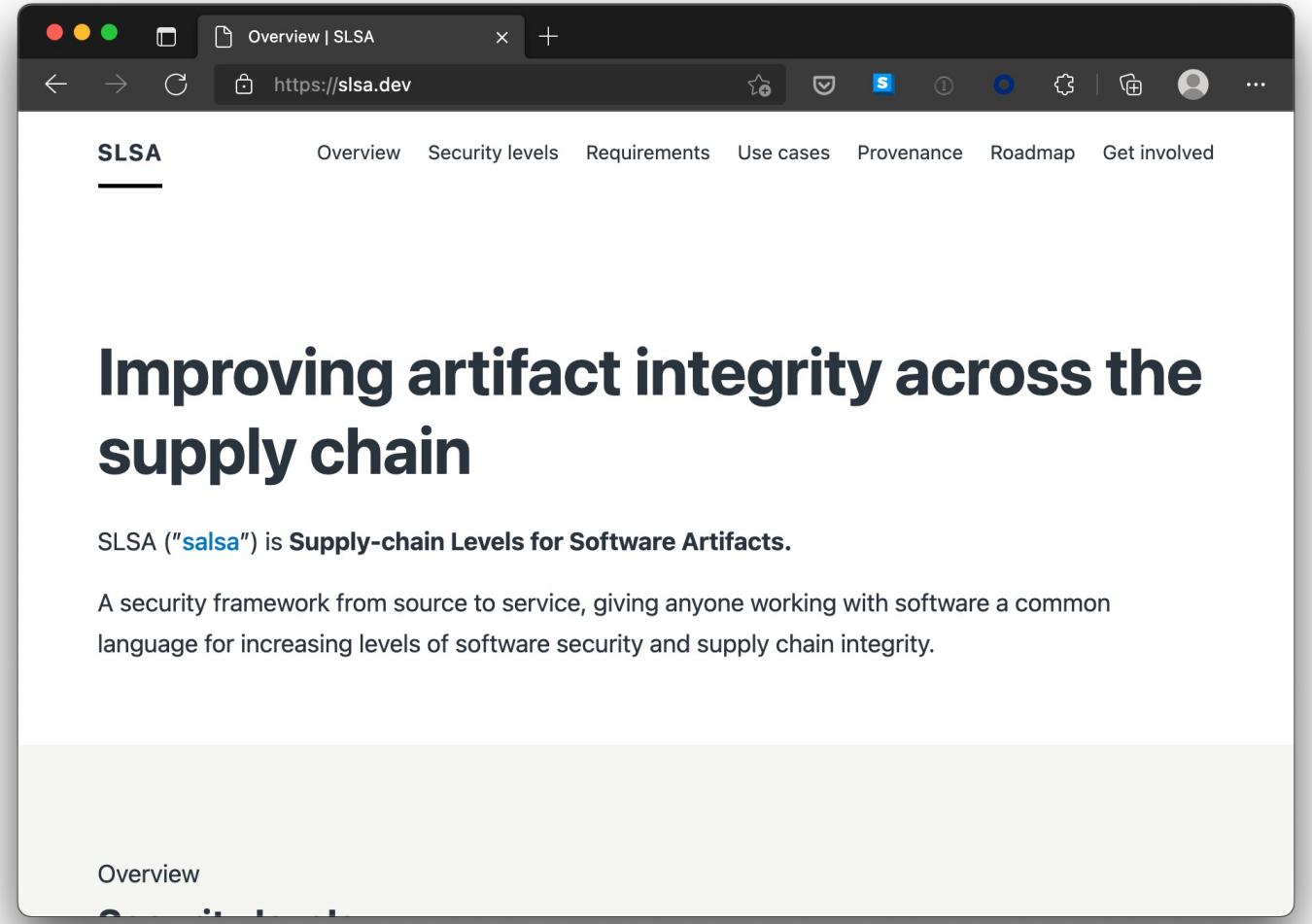
@nielstanis

MyAwesomeWebApp Demo

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

FUTURE TECH

Google SLSA



FUTURE TECH

@nielstanis

Google SLSA Levels

Level	Description	Example
1	Documentation of the build process	Unsigned provenance
2	Tamper resistance of the build service	Hosted source/build, signed provenance
3	Extra resistance to specific threats	Security controls on host, non-falsifiable provenance
4	Highest levels of confidence and trust	Two-party review + hermetic builds

FUTURE TECH

SLSA GitHub Action

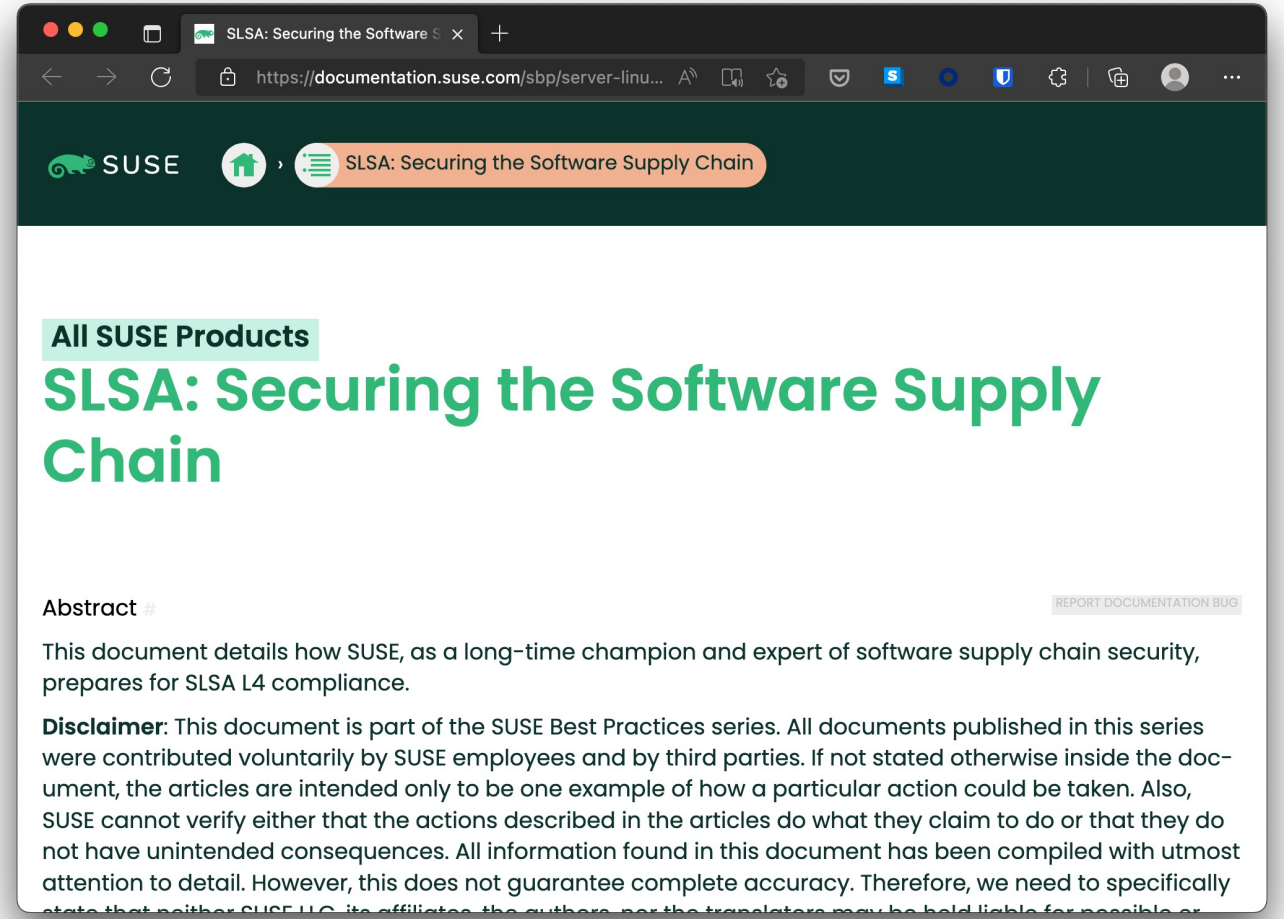
- Released April 2022

- 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

FUTURE TECH

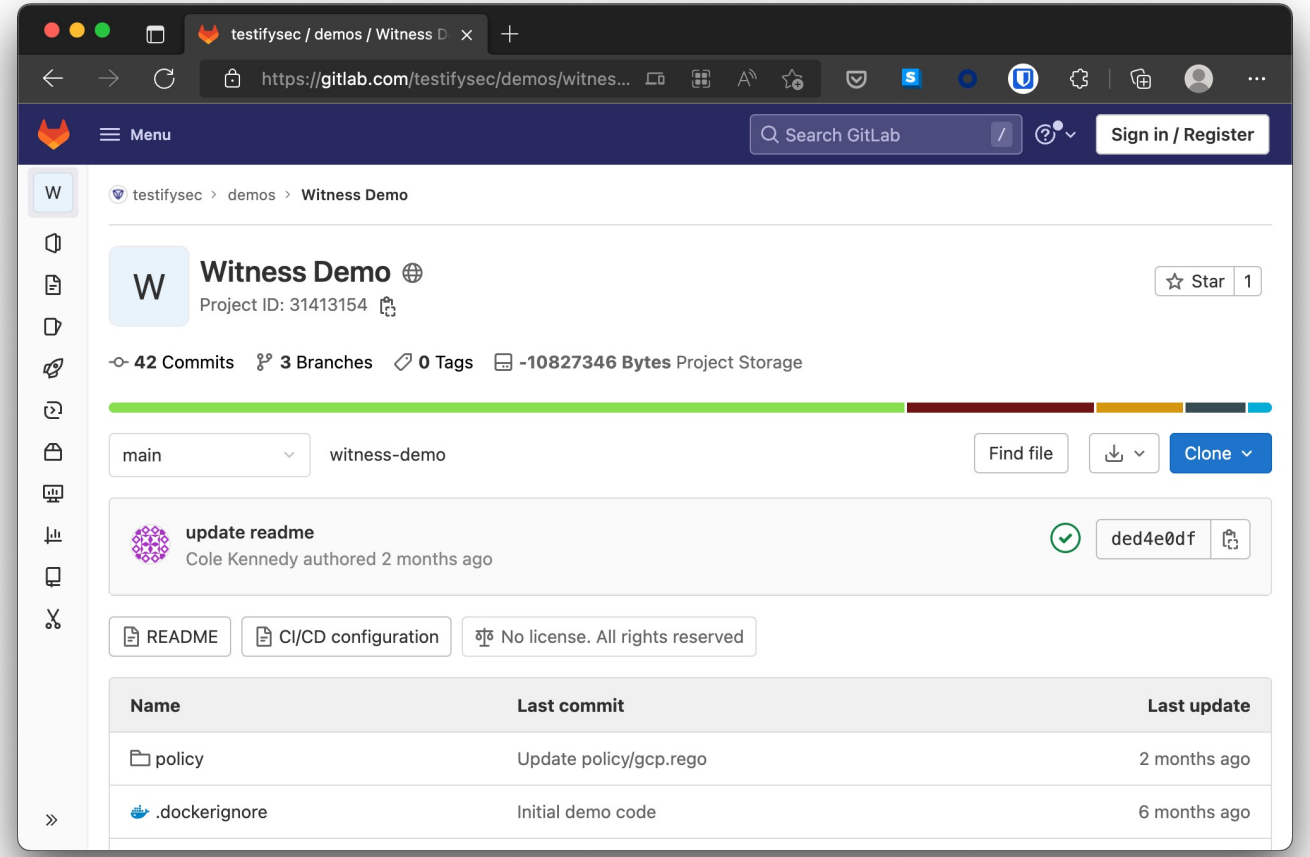
SUSE SLSA Level 4



FUTURE TECH

@nielstanis

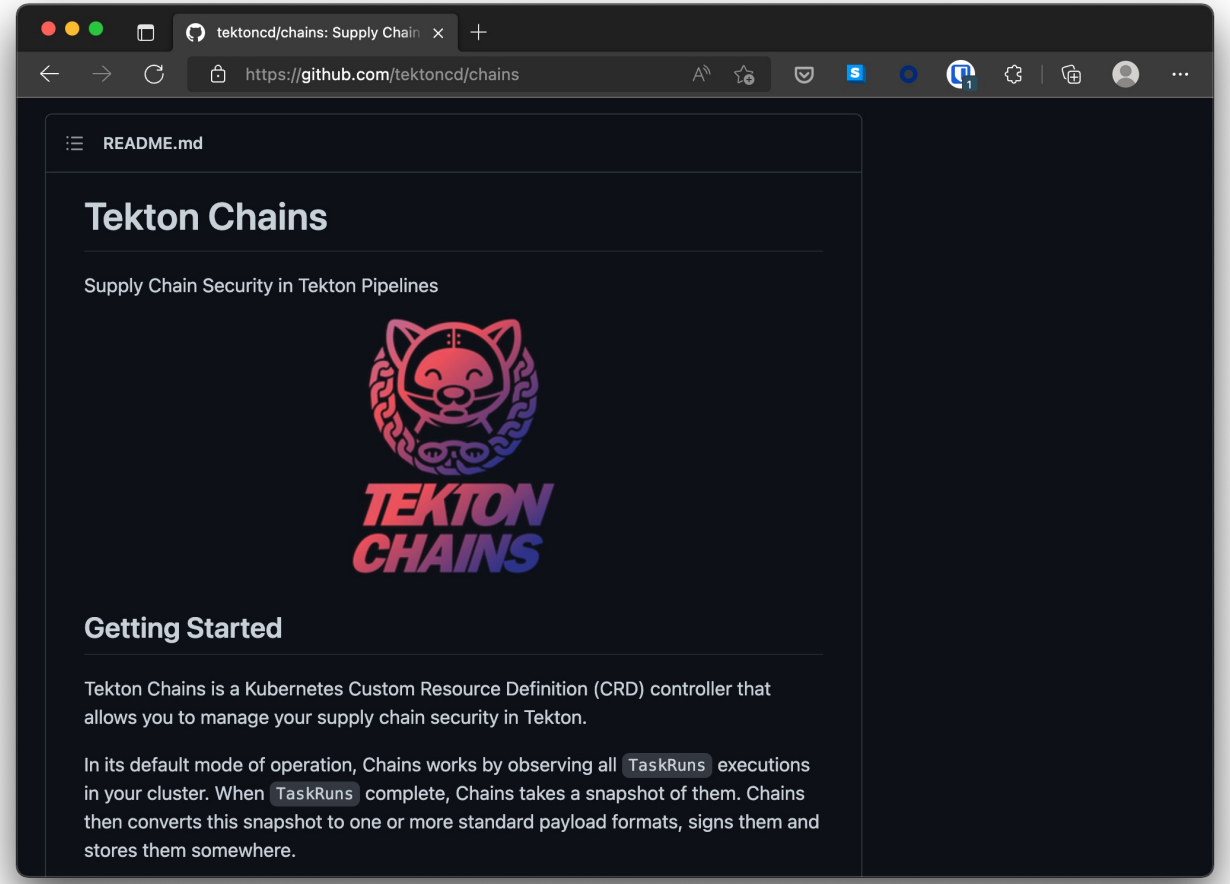
Witness & GitLab Attestator



FUTURE TECH

@nielstanis

Open Shift - Tekton - Tekton Chains

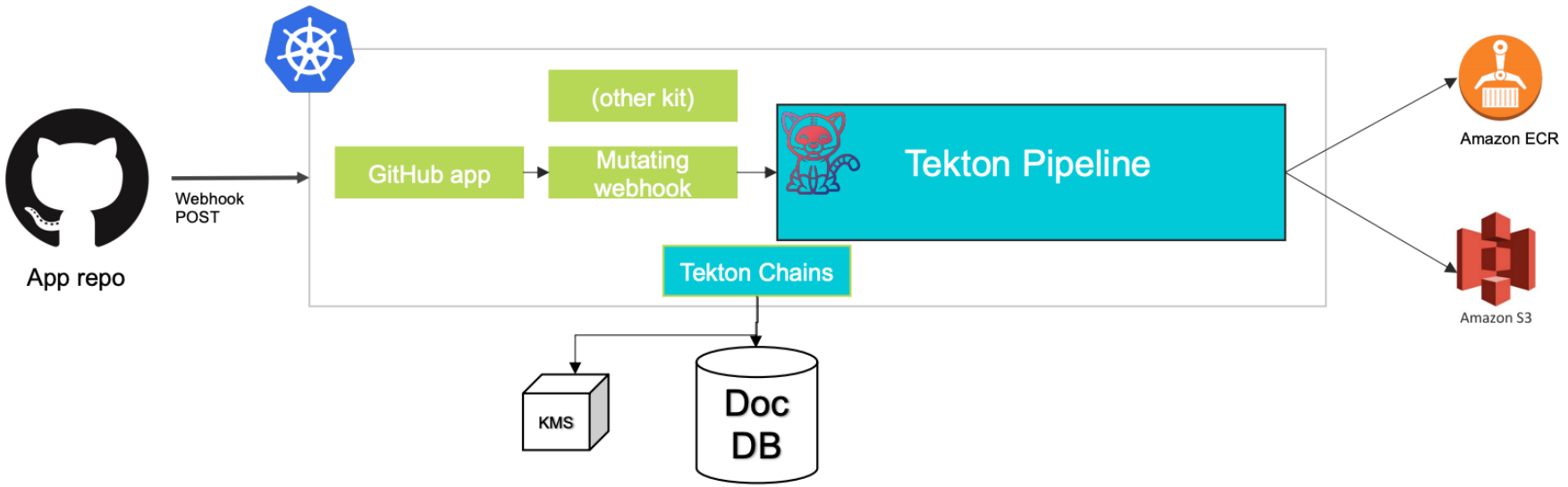


FUTURE TECH

@nielstanis

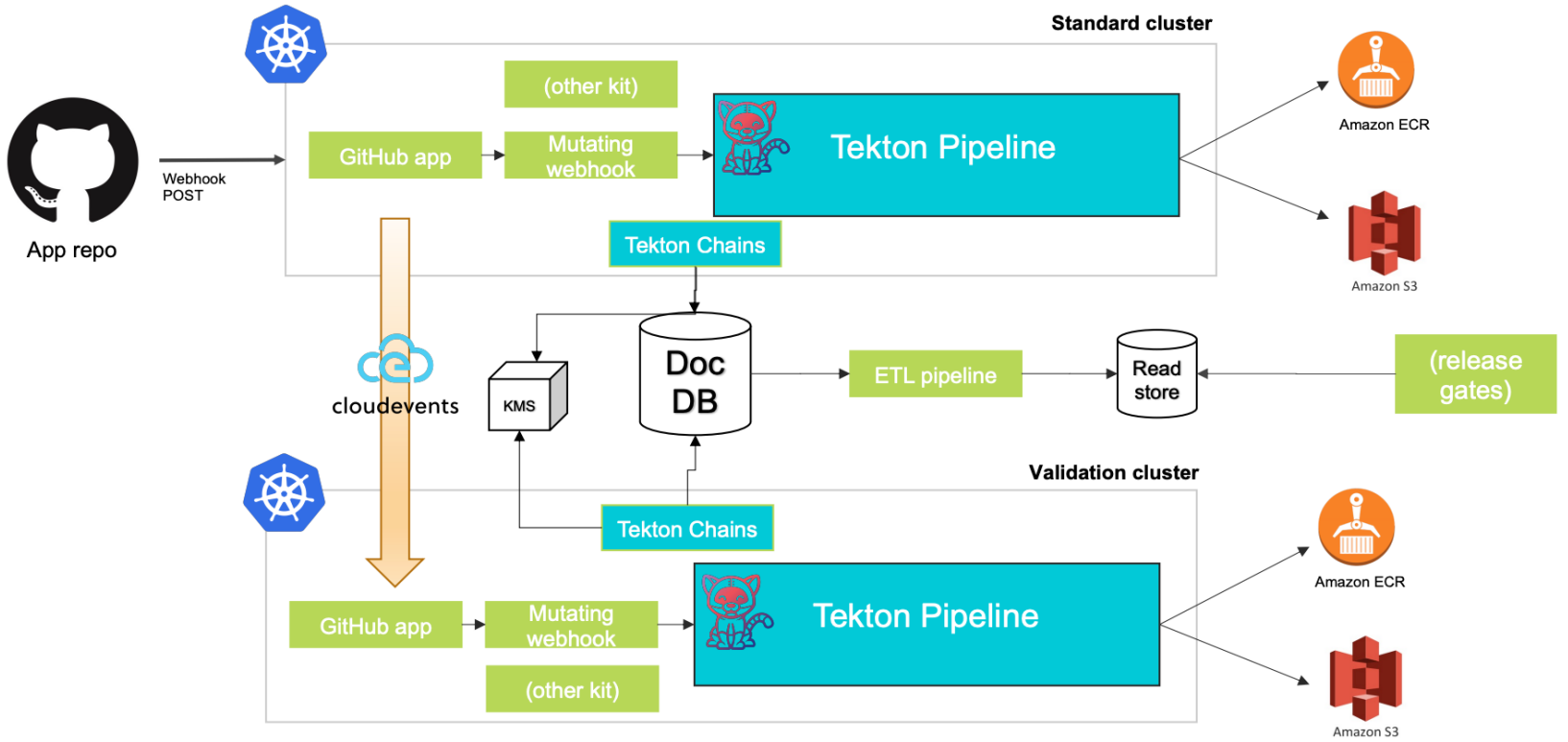
SolarWinds Project Trebuchet

Pipeline With Attestations



SolarWinds Project Trebuchet

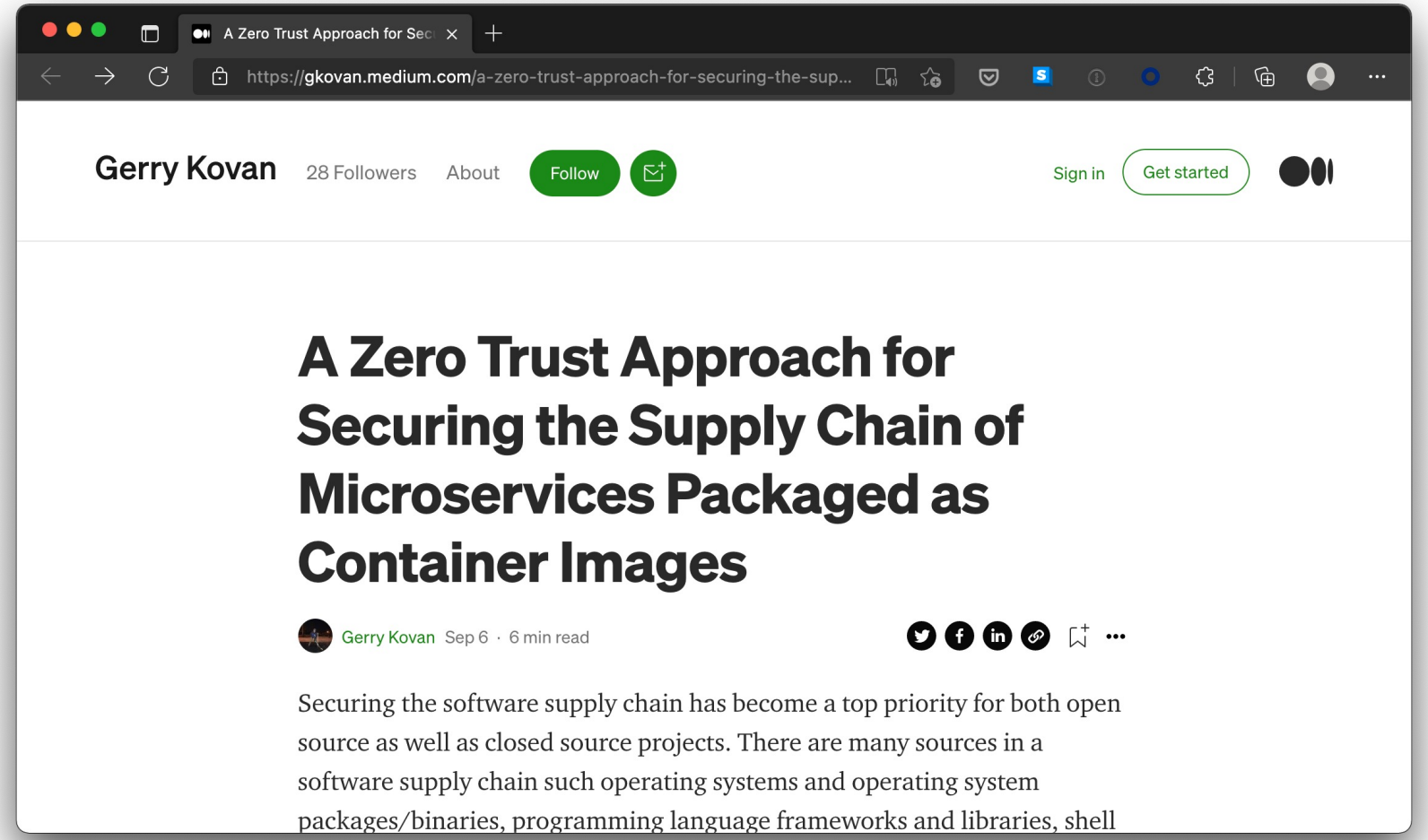
Reading Results



FUTURE TECH

@nielstanis

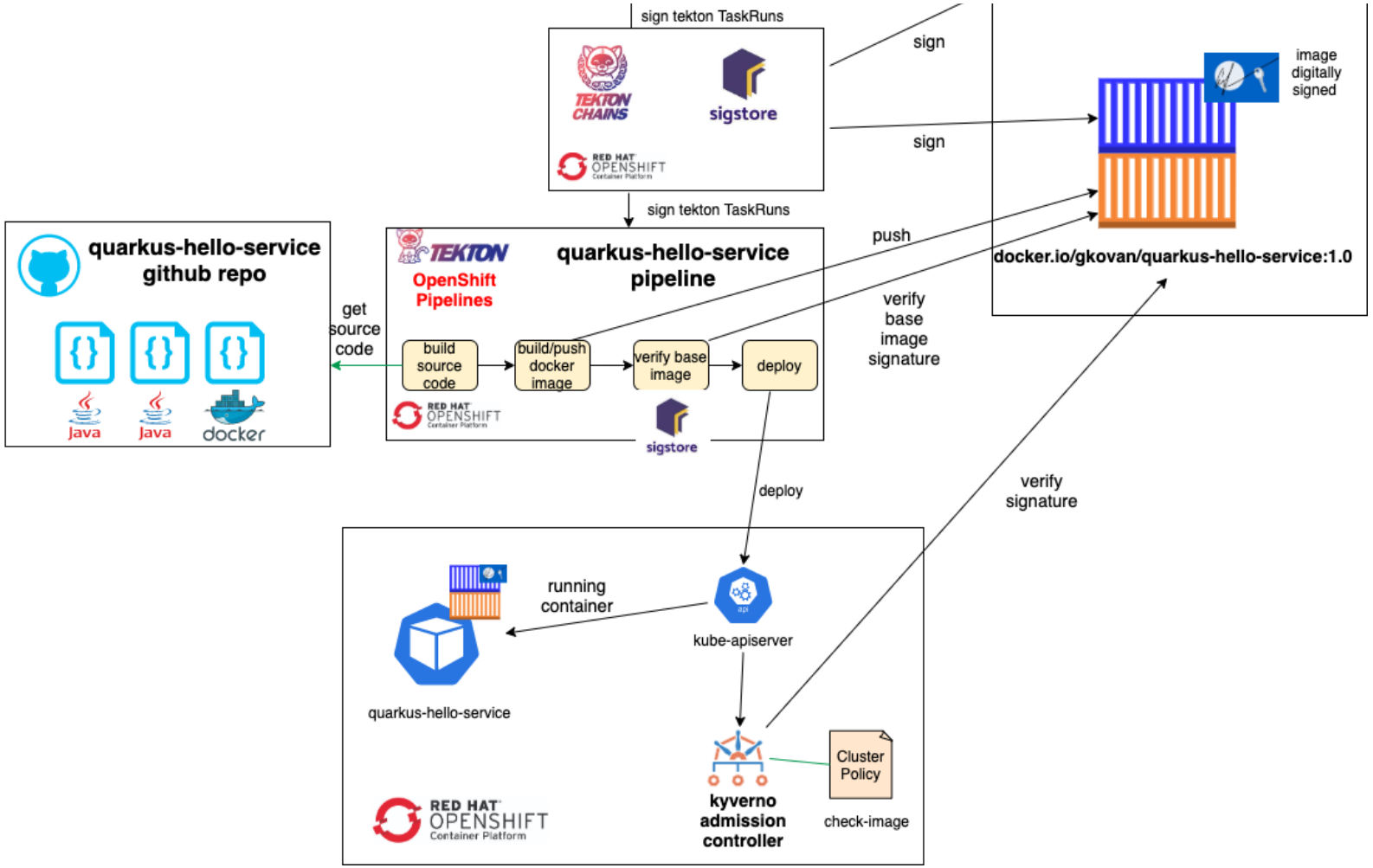
IBM OpenShift



FUTURE TECH

@nielstanis

IBM OpenShift

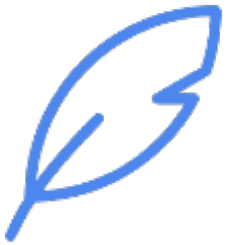


FUTURE TECH

@nielstanis

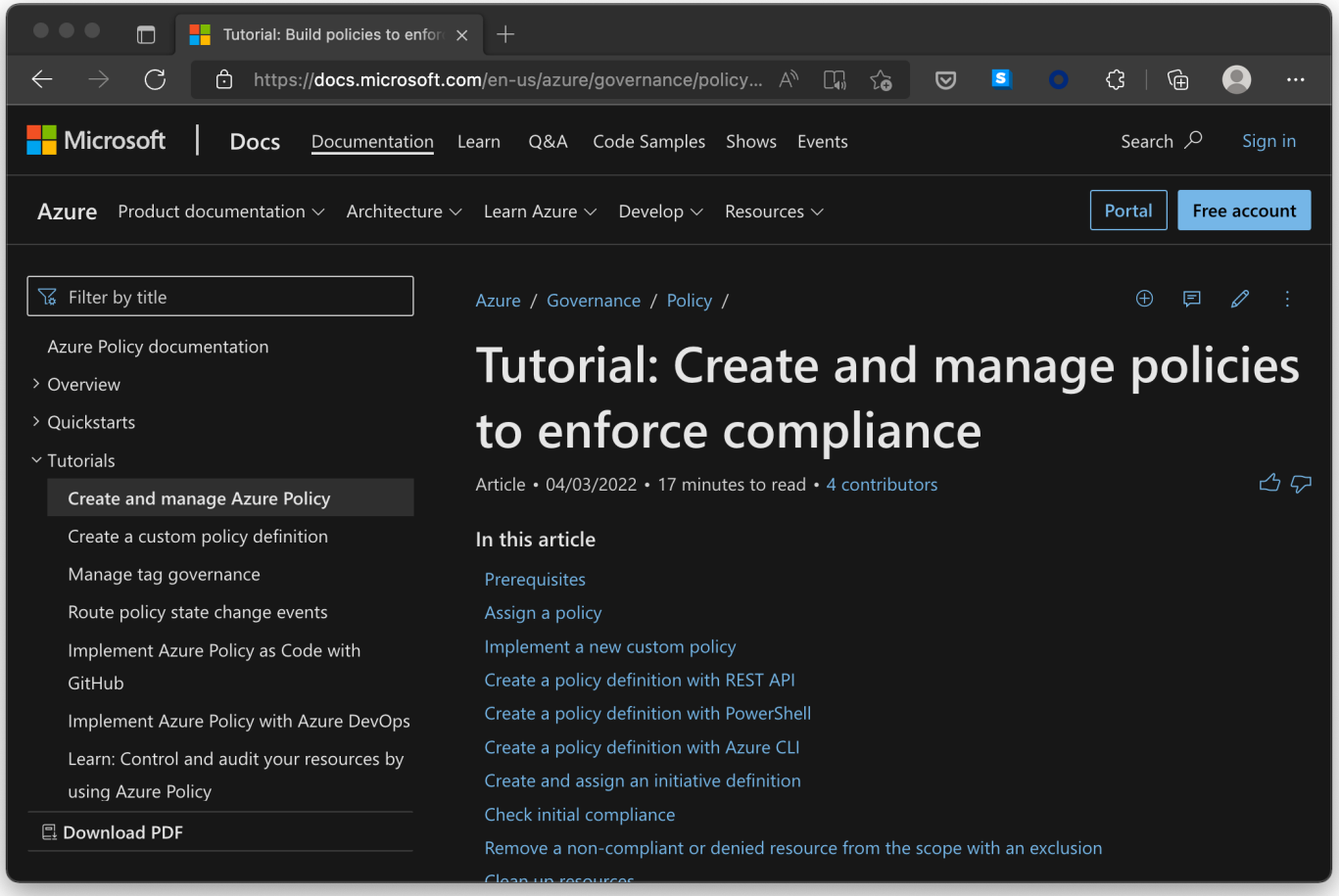
Grafeas and Kritis by Google

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



FUTURE TECH

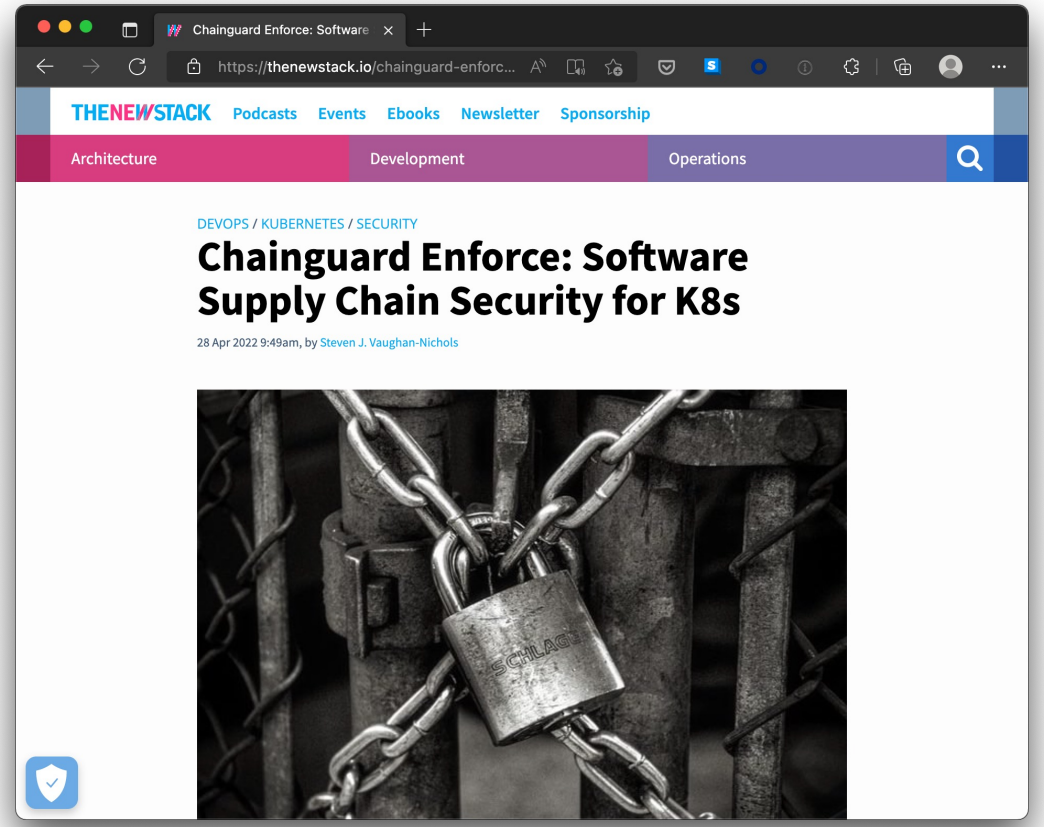
Azure Policy

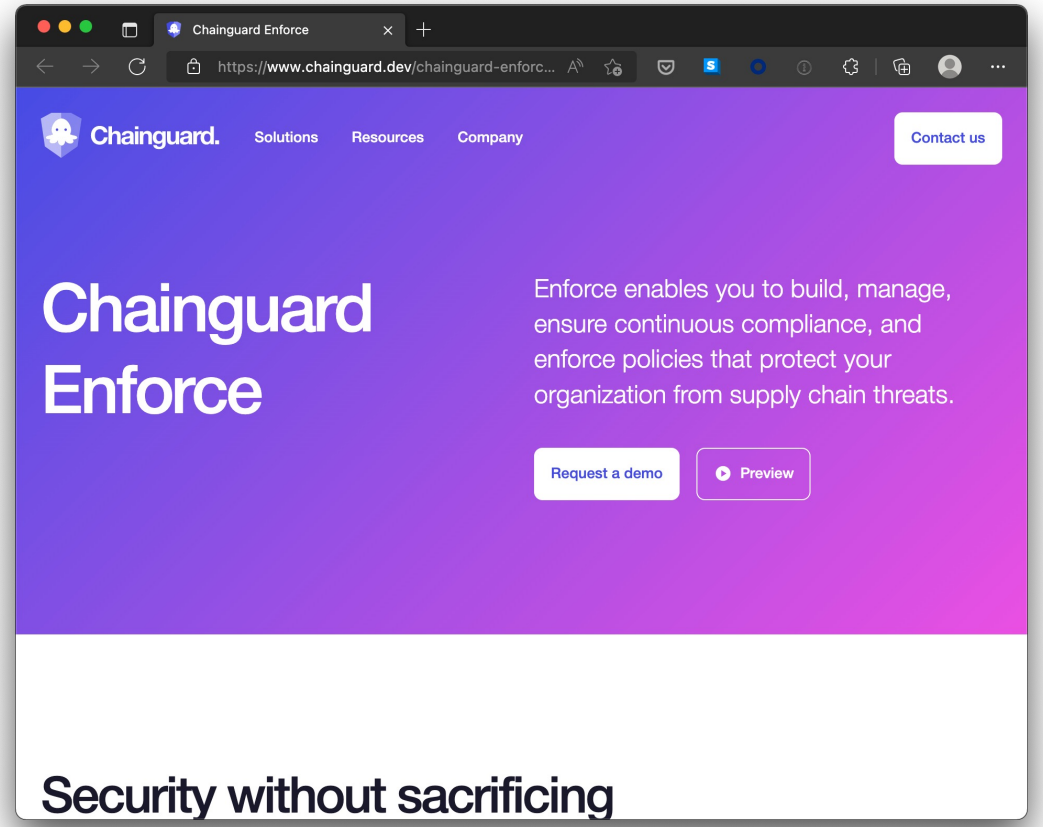


FUTURE TECH

@nielstanis

Chainguard Enforce





FUTURE TECH

@nielstanis

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.

FUTURE TECH

Conclusion

- Integrate security into your software lifecycle.
- Start working on creating SBOM’s and see how SLSA can fit into your process.
- Try to work with SBOM output and use it!

VERACODE

Thanks! Questions?

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