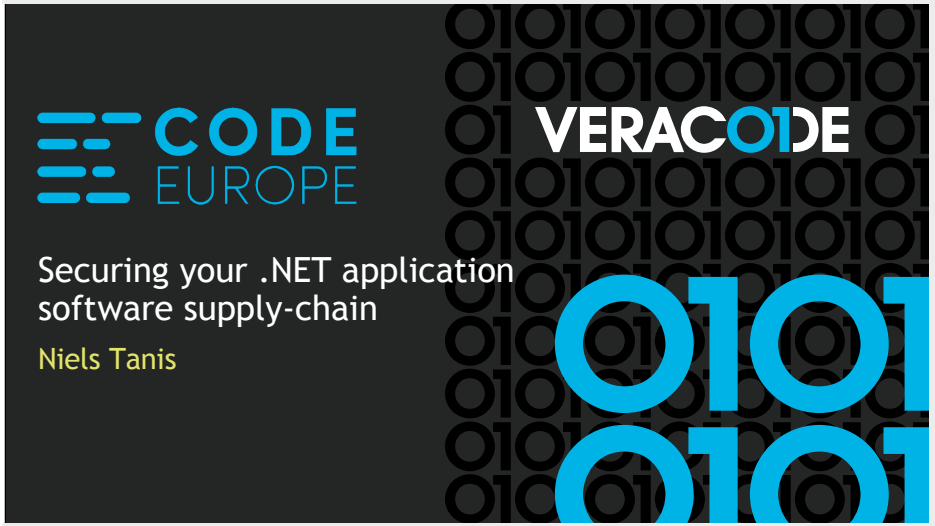

CODE
EUROPE
VERACODE
Securing your .NET application
software supply-chain
Niels Tanis
0101
0101

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

CODE EUROPE

@nielstanis@infosec.exchange

Securing your .NET application software supply-chain

CODE EUROPE

@nielstanis@infosec.exchange

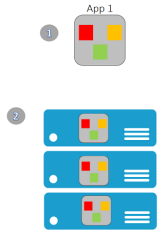
Picture is from Veracode report/site:
https://www.veracode.com/sites/default/files/pdf/resources/ipapers/everything-you-need-to-know-open-source-risk/index.html

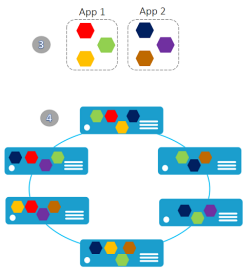
Agenda

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

Evolution in Software Architecture

- Monolith
- Microservices
- Serverless
  Cloud-Native





CODE EUROPE

@nielstanis@infosec.exchange

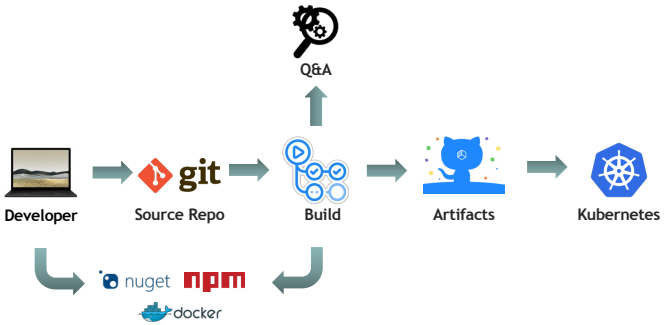
What is a Supply Chain?

CODE EUROPE

@nielstanis@infosec.exchange

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?

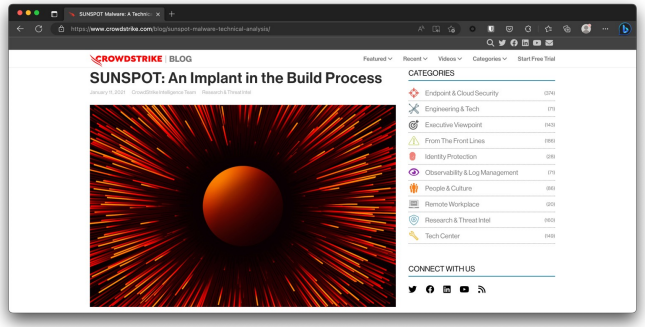
Software Supply Chain



CODE EUROPE

@nielstanis@infosec.exchange

SolarWinds SunSpot

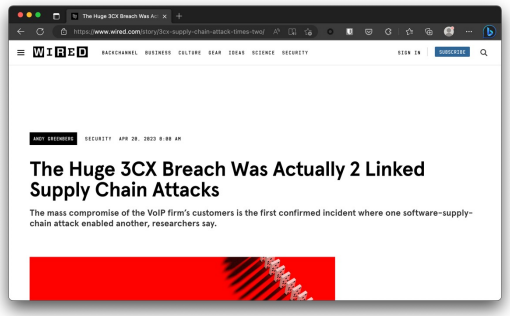


CODE EUROPE

@nielstanis@infosec.exchange

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/

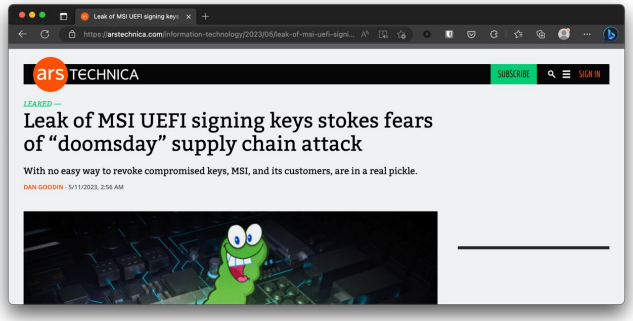
3CX Supply Chain Attack



CODE EUROPE

@nielstanis@infosec.exchange

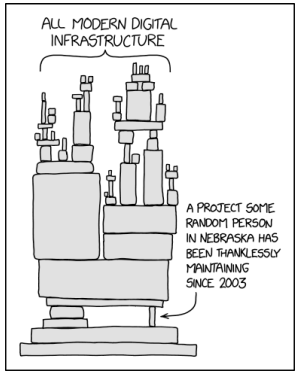
MSI UEFI Signing Keys



CODE EUROPE

@nielstanis@infosec.exchange

XKDC - Dependency



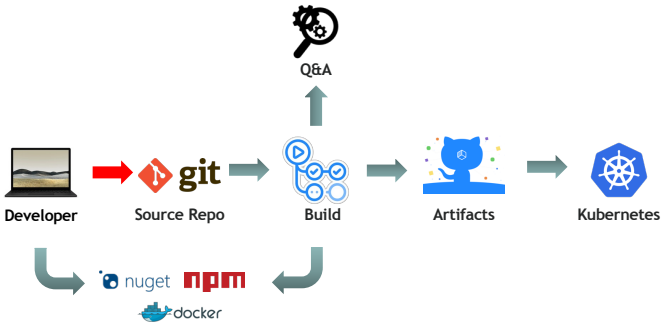
https://xkcd.com/2347/

CODE EUROPE

@nielstanis@infosec.exchange

https://xkcd.com/2347/

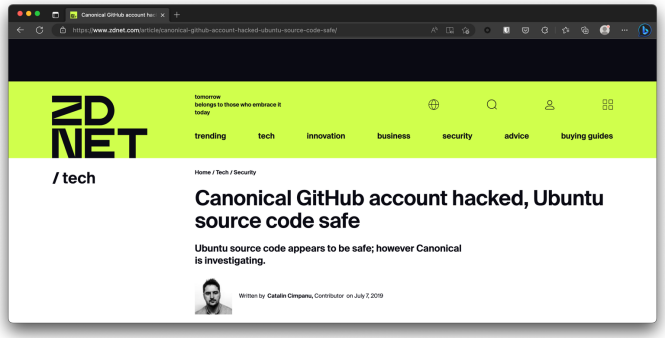
Software Supply Chain



CODE EUROPE

@nielstanis@infosec.exchange

GitHub account

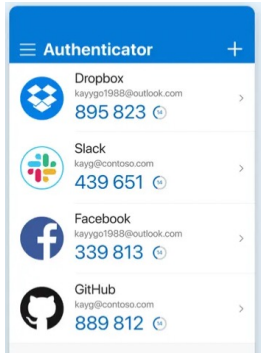


CODE EUROPE

@nielstanis@infosec.exchange

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

Use MFA on source-repository



CODE EUROPE

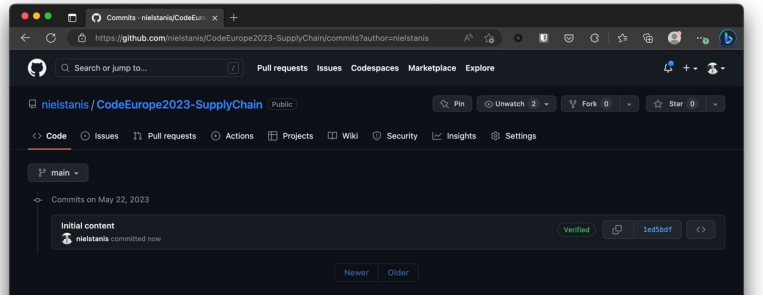
@nielstanis@infosec.exchange

Slide with larger h
when there is not a
be text heavy

The content text slid
bullets as preview, h
you do not need t
bullets. Text is pres
Trebuchet size 14
bullets may be rem
not needed.

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

GIT Commit Signing

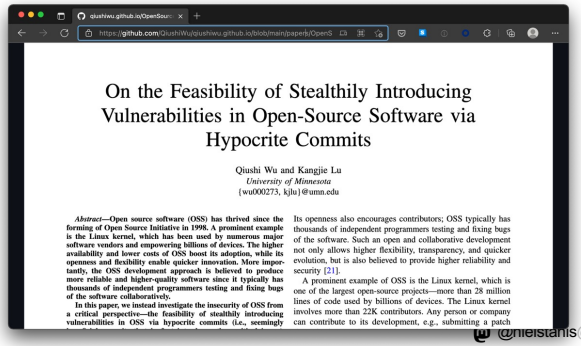


CODE EUROPE

@nielstanis@infosec.exchange

https://www.hanselman.com/blog/HowToSetupSignedGitCommitsWithAYubiKeyNEOAndGPGAndKeybaseOnWindows.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 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].

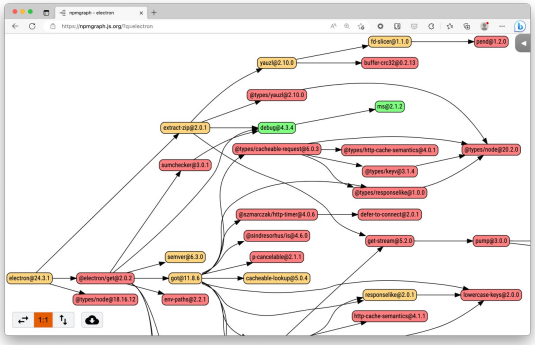
A prominent example of OSS is the Linux kernel, which is one of the largest open-source projects—more than 28 million lines of code used by billions of devices. The Linux kernel involves more than 22K contributors. Any person or company can contribute to its development, e.g., submitting a patch

CODE EUROPE

@melstanis@infosec.exchange

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

Visual Studio Code

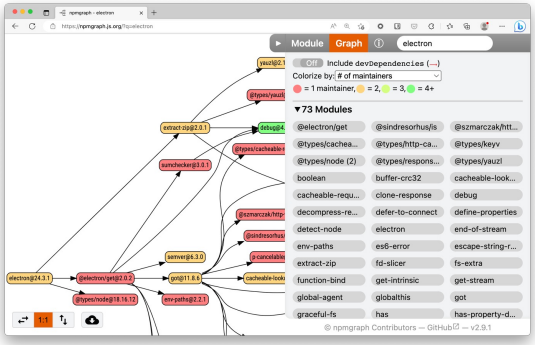


CODE EUROPE

@nielstanis@infosec.exchange

https://npmgraph.jsorg/?q=electron

Visual Studio Code

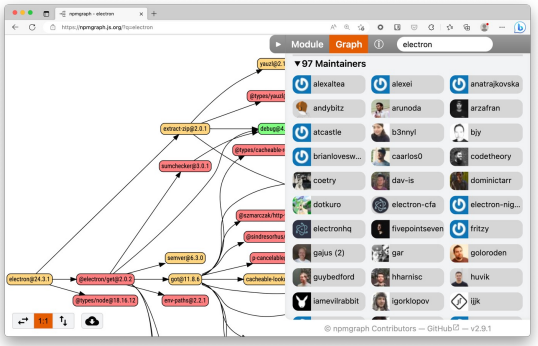


CODE EUROPE

@nielstanis@infosec.exchange

https://npmgraph.jsorg/?q=electron

Visual Studio Code

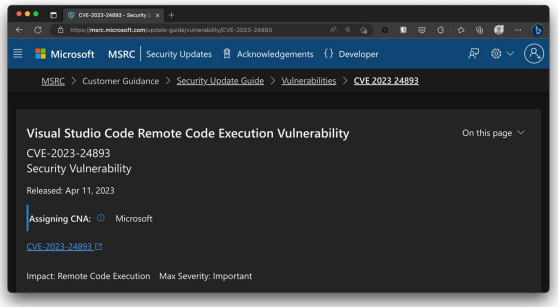


CODE EUROPE

@nielstanis@infosec.exchange

https://npmgraph.jsorg/?q=electron

Visual Studio Code



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

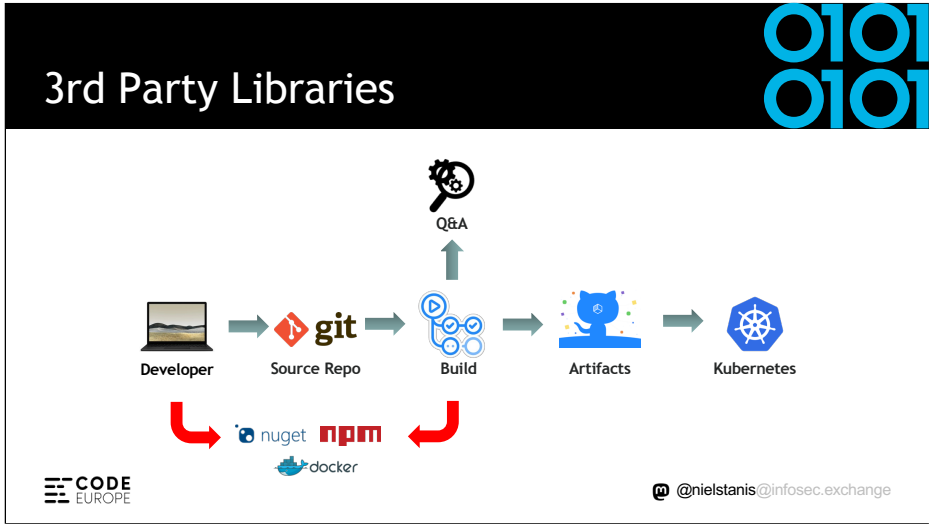
Visual Studio Code

CODE EUROPE

@nielstanis@infosec.exchange

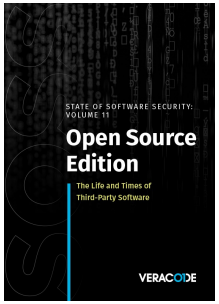
https://www.bleepingcomputer.com/news/security/heres-how-a-researcher-broke-into-microsoft-vs-codes-github/

3rd Party Libraries



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.”

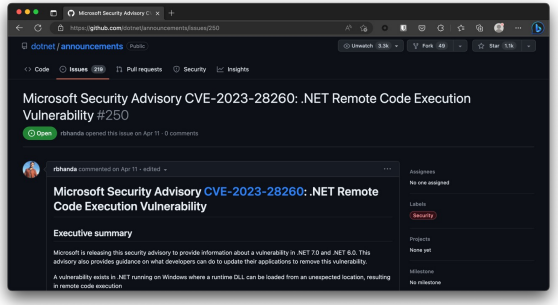


CODE EUROPE

@nielstanis@infosec.exchange

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

Vulnerabilities in libraries



CODE EUROPE

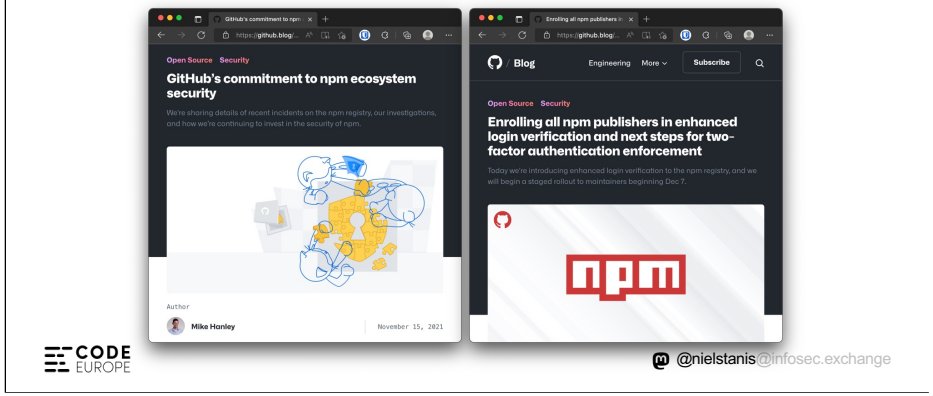
@nielstanis@infosec.exchange

https://github.com/dotnet/announcements/issues/250

Vulnerabilities in libraries

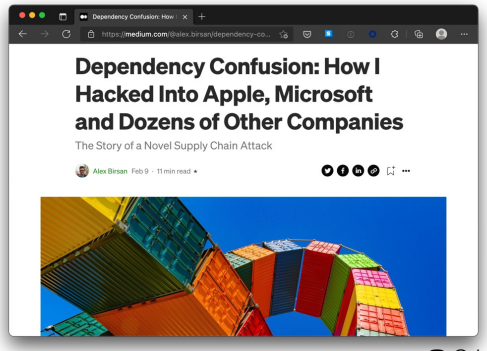
https://us-cert.cisa.gov/ncas/current-activity/2021/10/22/malware-discovered-popular-npm-package-ua-parser-js
https://portswigger.net/daily-swig/popular-npm-package-ua-parser-js-poisoned-with-cryptomining-password-stealing-malware

Vulnerabilities in libraries



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



CODE EUROPE

@nielstanis@infosec.exchange

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

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

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
- My talk 'Sandboxing .NET Assemblies'

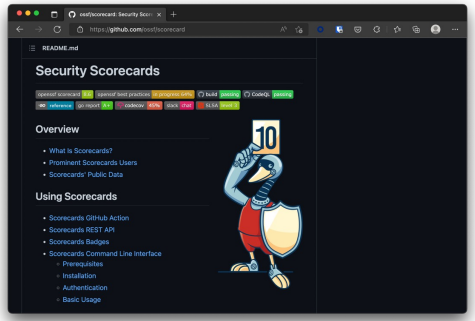
- Open Source Security Foundation - OpenSSF
- Security Scorecards - Security health metrics for Open Source

CODE EUROPE

@nielstanis@infosec.exchange

[Sandboxing .NET assemblies for fun, profit and of course security! - Niels Tanis - NDC Porto 2022 - YouTube]

Security Scorecards - OpenSSF

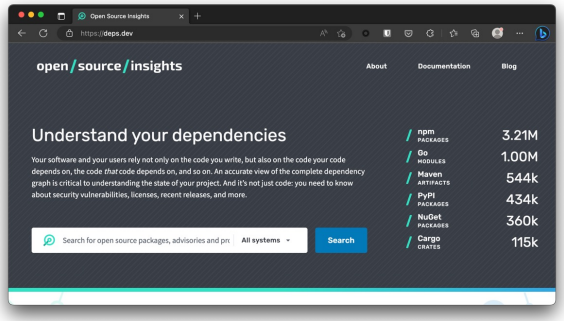


CODE EUROPE

@nielstanis@infosec.exchange

https://github.com/ossf/scorecard

Deps.Dev by Google

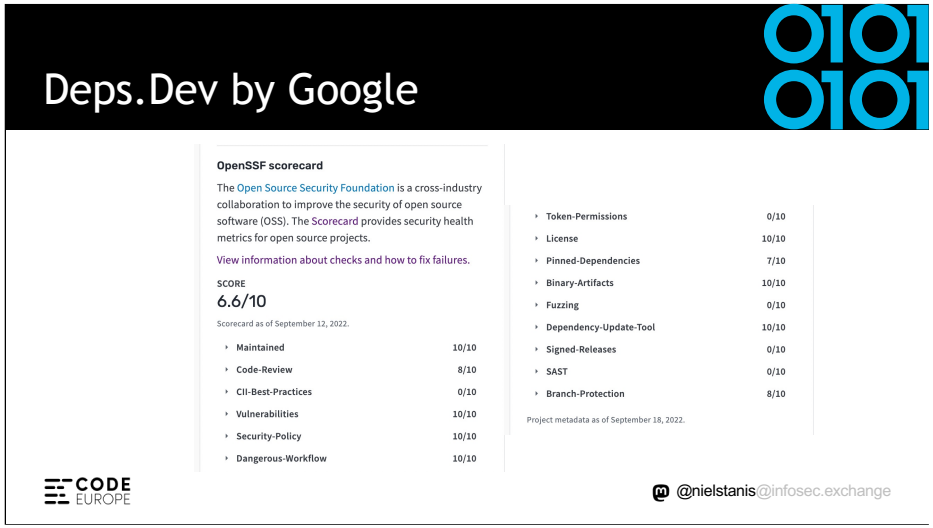


CODE EUROPE

@nielstanis@infosec.exchange

https://deps.dev/

Deps.Dev by Google

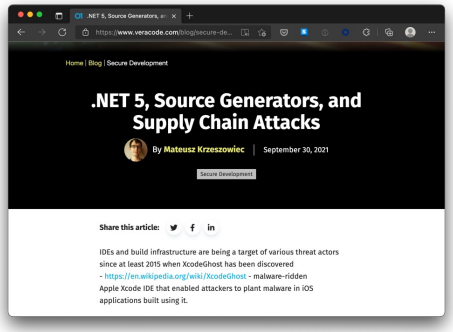


CODE EUROPE

@nielstanis@infosec.exchange

https://deps.dev/npm/electron

Source Generators



CODE EUROPE

@nielstanis@infosec.exchange

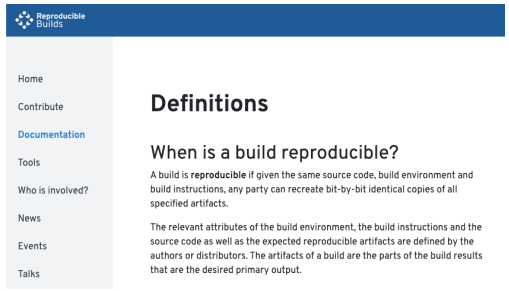
https://www.veracode.com/blog/secure-development/net-5-source-generators-and-supply-chain-attacks

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>
```

Reproducible/Deterministic Builds



CODE EUROPE

@nielstanis@infosec.exchange

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

Reproducible/Deterministic Builds

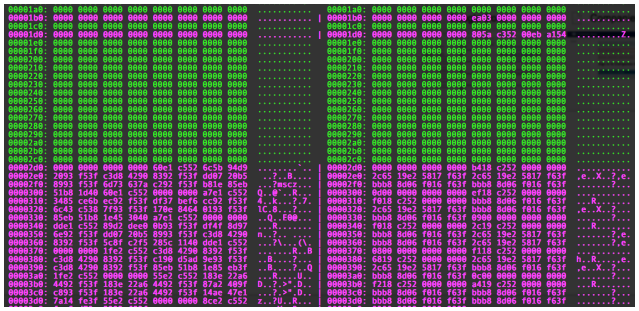
- Roslyn v1.1 started supporting some kind of determinism on how items are emitted
- Given same inputs, the compiled output will always be deterministic
- Inputs can be found in Roslyn compiler docs ‘Deterministic Inputs’

CODE EUROPE

@nielstanis@infosec.exchange

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

Reproducible Build .NET6



CODE EUROPE

@nielstanis@infosec.exchange

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

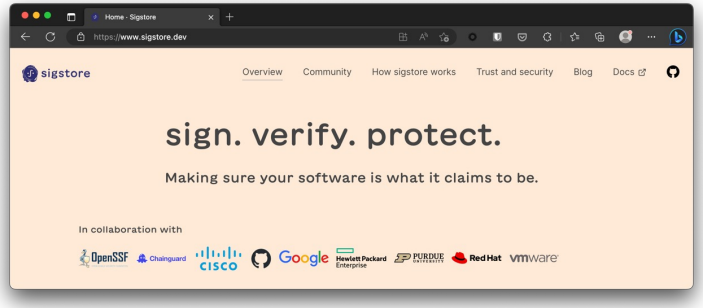
https://blog.paranoidcoding.com/2016/04/05/deterministic-builds-in-roslyn.html
https://github.com/dotnet/roslyn/blob/master/docs/compilers/Deterministic%20Inputs.md
https://devblogs.microsoft.com/dotnet/producing-packages-with-source-link/
https://github.com/dotnet/reproducible-builds

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`

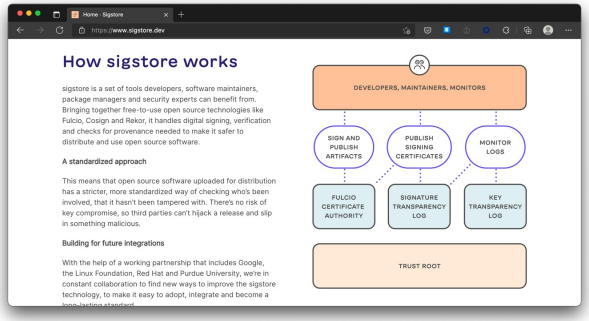
https://github.com/dotnet/designs/blob/main/accepted/2020/reproducible-builds.md

Signing artifacts



https://sigstore.dev

Signing artifacts



CODE EUROPE

@nielstanis@infosec.exchange

https://sigstore.dev

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

https://sigstore.dev

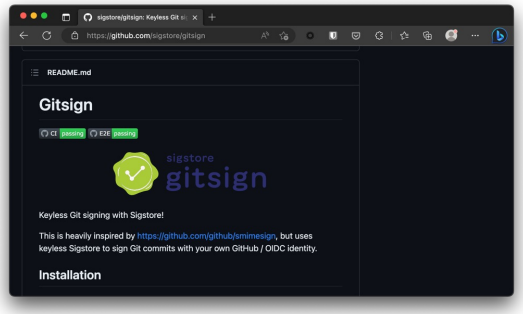
Cosign Keyless Signing

CODE EUROPE

anis@infosec.exchange

https://sigstore.dev
https://blog.sigstore.dev/introducing-gitsign-9fd3f1b682aa

Git Commit Signing Sigstore GitSign



CODE EUROPE

@nielstanis@infosec.exchange

https://sigstore.dev
https://blog.sigstore.dev/introducing-gitsign-9fd3f1b682aa

Automotive Industry

CODE EUROPE

@nielsdanis@infosec.exchange

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

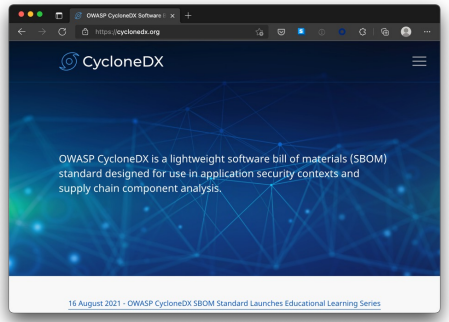
CODE EUROPE

@nielstanis@infosec.exchange

Software Bill of Materials (SBOM)

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

Software Bill of Materials (SBOM)

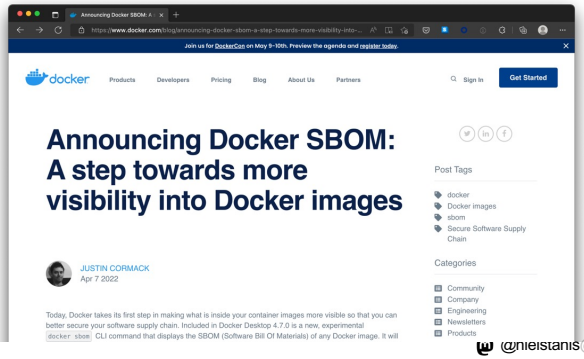


CODE EUROPE

@nielstanis@infosec.exchange

https://cyclonedx.org

Docker SBOM

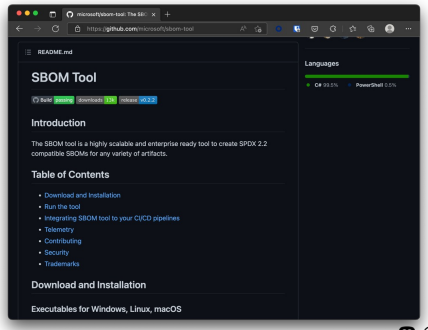


CODE EUROPE

@mfeistanis@infosec.exchange

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

Microsoft SBOM Tool

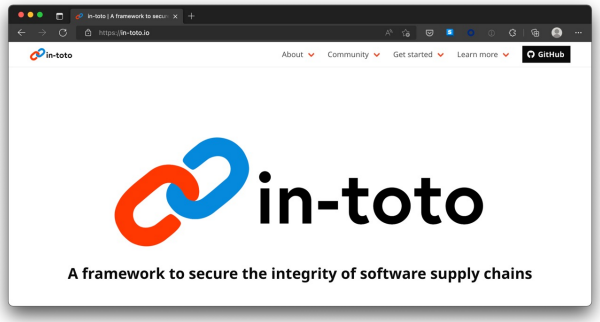


CODE EUROPE

@nielstanis@infosec.exchange

https://github.com/microsoft/sbom-tool

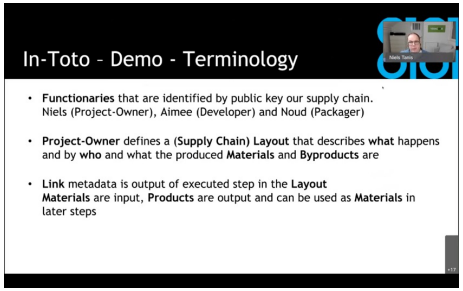
In-toto



CODE EUROPE

@nielstanis@infosec.exchange

In-Toto – Demo

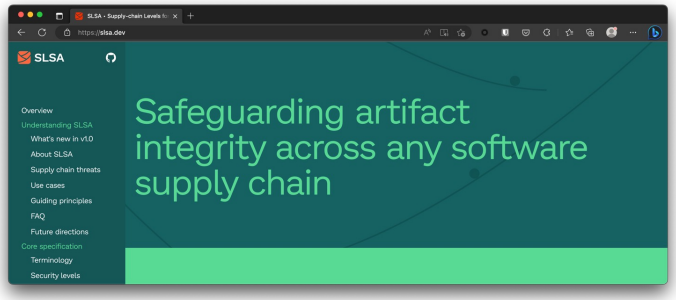


CODE EUROPE

@nielstanis@infosec.exchange

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

Google SLSA



CODE EUROPE

@nielstanis@infosec.exchange

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

https://slsa.dev

SLSA GitHub Action

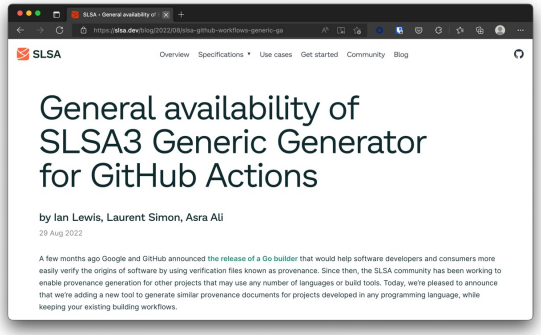
- Released April 2022

- 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

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

SLSA3 Generator GitHub Actions



CODE EUROPE

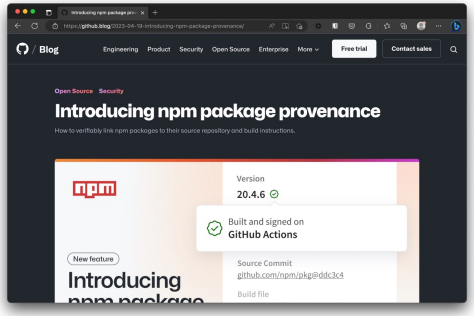
@nielstanis@infosec.exchange

https://slsa.dev/blog/2022/08/slsa-github-workflows-generic-ga

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

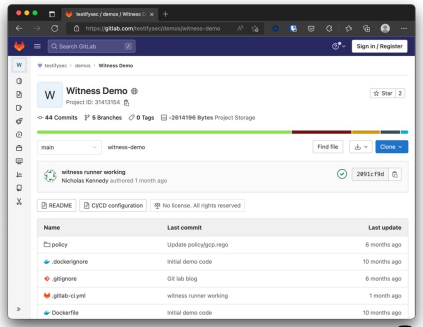


CODE EUROPE

@nielstanis@infosec.exchange

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

Witness & GitLab Attestator

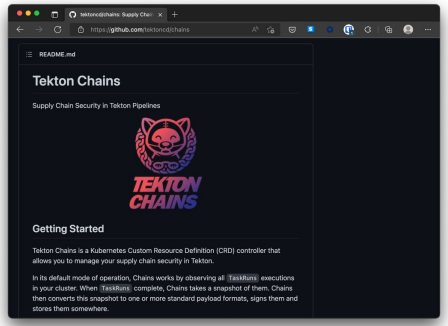


CODE EUROPE

@nielstanis@infosec.exchange

https://gitlab.com/testifysec/demos/witness-demo
https://github.com/testifysec/witness

Open Shift – Tekton – Tekton Chains



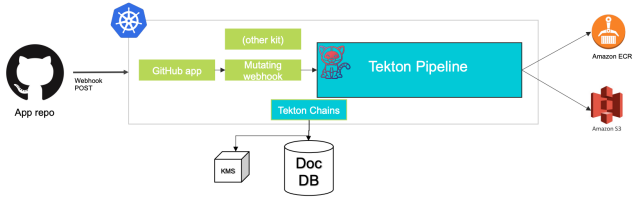
CODE EUROPE

@nielstanis@infosec.exchange

https://github.com/tektoncd/chains

SolarWinds Project Trebuchet

Pipeline With Attestations



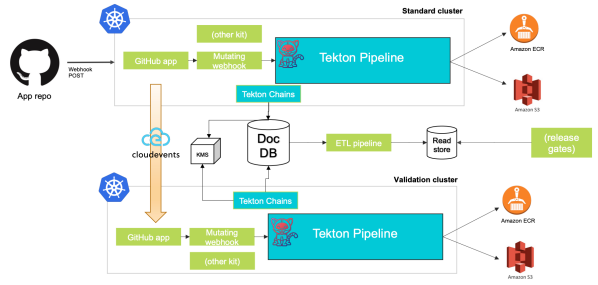
CODE EUROPE

@nielstanis@infosec.exchange

https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf
https://www.youtube.com/watch?v=1-tMRxqMwTQ

SolarWinds Project Trebuchet

Reading Results

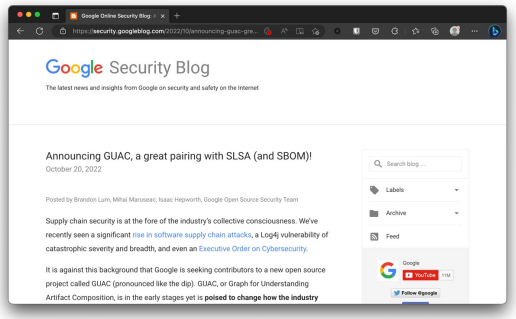


CODE EUROPE

@nielstanis@infosec.exchange

https://static.sched.com/hosted_files/supplychainsecurityconna21/df/SupplyChainCon-TrevorRosen-Keynote.pdf
https://www.youtube.com/watch?v=1-tMRxqMwTQ

Google Graph for Understanding Artifact Composition (GUAC)



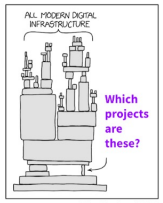
CODE EUROPE

@nielstanis@infosec.exchange

Graph for Understanding Artifact Composition

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?

SLSA

trivy

grype

Reactive

HOW AM I AFFECTED???

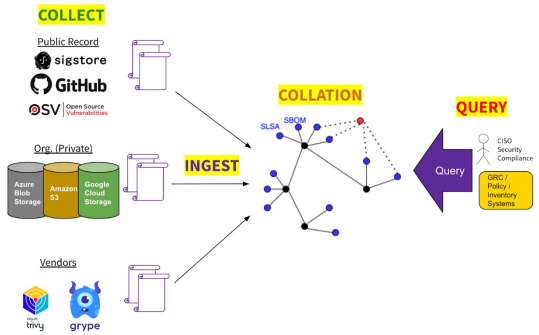
A vulnerability or supply chain compromise is discovered!

+ Codecov, Solarwinds compromises

CODE EUROPE

@nielstanis@infosec.exchange

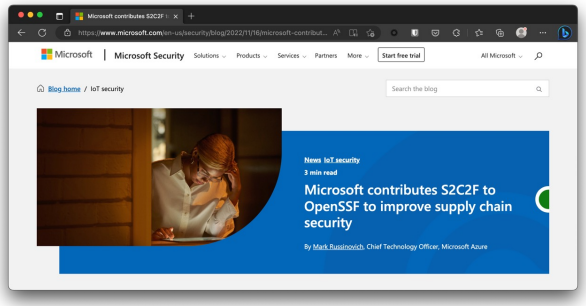
Graph for Understanding Artifact Composition



CODE EUROPE

@nielstanis@infosec.exchange

Secure Supply Chain Consumption Framework (S2C2F)



CODE EUROPE

@nielstanis@infosec.exchange

https://www.microsoft.com/en-us/security/blog/2022/11/16/microsoft-contributes-s2c2f-to-openssf-to-improve-supply-chain-security/

Secure Supply Chain Consumption Framework (S2C2F)

- 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

CODE EUROPE

@nielstanis@infosec.exchange

https://github.com/ossf/s2c2f/blob/main/specification/framework.md

Secure Supply Chain Consumption Framework (S2C2F)

Level 1	Level 2	Level 3	Level 4
Minimum OSS Governance Program • Use package managers • Local copy of artifact • Scan with known vulns • Scan for software licenses • Inventory OSS • Manual OSS updates	**Secure Consumption and Improved MTTR** • Scan for end life • Have an incident response plan • Auto OSS updates • Alert on vulns at PR time • Audit that consumption is through the approved ingestion method • Validate integrity of OSS • Secure package source file configuration	**Malware Defense and Zero-Day Detection** • Deny list capability • Clone OSS source • Scan for malware • Proactive security reviews • Enforce OSS provenance • Enforce consumption from curated feed	**Advanced Threat Defense** • Validate the SBOMs of OSS consumed • Rebuild OSS on trusted infrastructure • Digitally sign rebuilt OSS • Generate SBOM for rebuilt OSS • Digitally sign protected SBOMs • Implement fixes

CODE EUROPE

@nielstanis@infosec.exchange

https://github.com/ossf/s2c2f/blob/main/specification/framework.md

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!

Questions?

- https://github.com/nielstanis/codeeurope2023-supplychain
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- https://blog.fennec.dev
- Dziękuję! Thank you!