

Software-Supply-Chain Security in Practice

Sichere Software - vom Commit bis zum User

Stephan Kaps

Leiter Softwareentwicklung Bundesamt für Soziale Sicherung

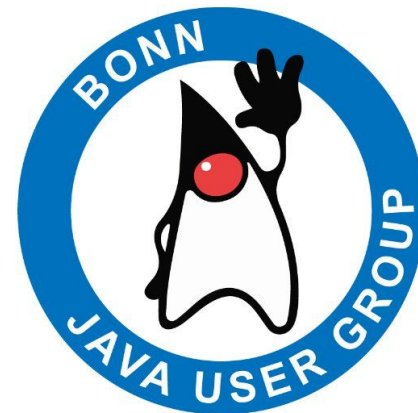
Gründer Java User Group Bonn

Java Entwickler & Architekt seit 2002

Weitere Schwerpunkte:

- Softwareentwicklungsprozesse
- BizDevSecOps
- OpenSource
- Speaker & Autor

loizodevops.org



Wer sind wir uns was machen wir?

(Aufsichts-) Behörde mit ca. 700 Mitarbeiter

ca. 60 Fachanwendungen (Produkte)

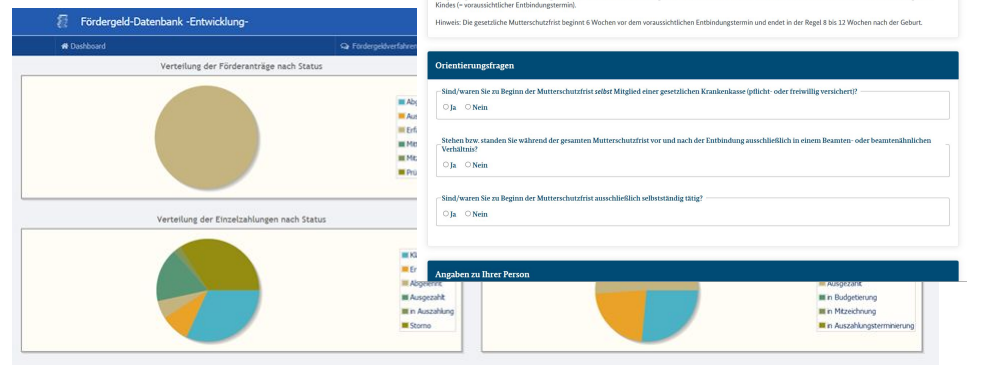
für interne Fachbereiche & Landesprüfdienste

seit 2016 agile Produktentwicklung (Scrum)

10 Entwickler und 2 Product Owner

Private-Cloud mit Containertechnologien

Sehr hohe Mitarbeiterzufriedenheit



Log4j-Sicherheitslücke

Wie löscht man ein brennendes Internet?

IT-Fachleute sind wegen der Log4j-Sicherheitslücke alarmiert, erste Angriffe laufen bereits. Welche Folgen hat das für Nutzer? Wie ließe sich so etwas verhindern? Antworten auf die wichtigsten Fragen.

Von Patrick Beuth

13.12.2021, 17.20 Uhr

Was ist passiert?

CVE-2021-44228: Risiko kritisch, CVSSv3 10/10

Cyber-Sicherheitswarnung des BSI Stufe Rot

Pressekonferenz in der Tagesschau am 13.12.2021

<https://www.tagesschau.de/multimedia/video/video-960365.html>

Maßnahmen

Identifizieren & Patchen

Härtung -> Flag setzen (`log4j2.formatMsgNoLookups=true`)

In Web Application Firewall (WAF) erkennen und blockieren

Ausschalten?

Patchen!

Patchen!!1!!

Maßnahmen

Identifizieren

- Day 0 = Bekanntwerden der Lücke
- Day 1 = Fixes und Patches ausrollen
- Day 2 = Was haben wir gelernt?

2 Herausforderungen

Wie können wir sicherstellen, dass wir Fragen zu dem, was wir in unserer Software verwenden, schnell beantworten können, wenn es eine weitere Schwachstelle (dieser Größenordnung) gibt?

Wie können wir sicherstellen, dass wir nicht anfällige Versionen von log4j aufnehmen und bereitstellen?

Welche Produkte
sind eigentlich
betroffen?

Kaufsoftware

= Black Box

Ist die Anwendung überhaupt in Java geschrieben?

Steht vielleicht was auf der Webseite?

Können wir den Hersteller kontaktieren?

Echt jetzt?



Eigenentwickelte Software

= Analyse der
Third-Party-Dependencies

Welche Anwendung verwendet
überhaupt Log4j?

Und falls ja, in welcher Version?

Haben wir einen Überblick über die
gesamte Anwendungslandschaft?



SBOM

Software Bill of Materials (SBOM)

- Beipackzettel zu einer Software
- Auflistung aller verwendeter Bibliotheken bzw. Komponenten
- Maschinenlesbare Inventarliste
- Beinhaltet direkte und transitive Abhängigkeiten und deren hierarchische Beziehung
- Für Software, die in Amerikanischer Regierung verwendet wird, inzwischen quasi verpflichtend
<https://www.cisa.gov/sbom>
- Spezifikation Standard SPDX oder CycloneDX
<https://cyclonedx.org/specification/overview/>
<https://cyclonedx.org/tool-center/>



Was ist CycloneDX?

“OWASP CycloneDX ist ein leichtgewichtiger Software Bill of Materials (SBOM)-Standard, der für den Einsatz in Anwendungssicherheitskontexten und der Analyse von Lieferkettenkomponenten (SCA) entwickelt wurde.”

SCA = supply chain component analysis

Technology

Technology Radar

Search Techniques

Techniques

Adopt ?

1. Four key metrics ▼
2. Single team remote wall ▼

Trial ?

3. Data mesh ▼
4. Definition of production readiness ▼
5. Documentation quadrants ▼
6. Rethinking remote standups ▼
7. Server-driven UI ▼

8. Software Bill of Materials ▼

Themes for this volume

Software Supply Chain Innovations

We've seen very public and severe problems that are caused by poor software supply chain governance. Now teams are realizing that responsible engineering practices include validating and governing project dependencies.

what you are looking for on a [previous Radar](#) already. We sometimes cull things just because there are too many to talk

@kitencol

Why Do Developers Implementing State Management in React

Ever since Redux was released, we've seen a steady stream of tool frameworks that manage state in different ways, each with a different set of trade-offs. We don't know how to churn the JavaScript ecosystem to promote?

our experience, it is not based on a comprehensive market analysis.

Download

Adopt



No change

to see?

lecting what we came might have covered ar already. We re too many to talk e the Radar reflects

Tool Center

[All tools](#) **149**
[Open Source](#) **112**
[Proprietary](#) **39**
[Build Integration](#) **61**
[Analysis](#) **45**
[Author](#) **2**
[GitHub Action](#) **13**
[Transform](#) **8**
[Library](#) **10**
[Signing / Notary](#) **5**
[Distribute](#) **4**
[proprietary](#) [analysis](#) [build-integration](#)

Apiiro

Apiiro

Apiiro enables security & development teams to proactively remediate critical risks in their cloud-native applications such as design flaws, secrets, IaC misconfigurations, API & OSS vulnerabilities across the software supply chain.

[proprietary](#) [analysis](#)

AppSonar

CyberTest

AppSonar Static Code Analyzer helps improve the security and quality of application code and can generate CycloneDX BOMs during analysis.

[opensource](#) [build-integration](#)

Auditjs

Sonatype

Audits an NPM package.json file to identify known vulnerabilities

 **51**  **184**

[opensource](#) [distribute](#)

BOM Repository Server

CycloneDX

A lightweight repository server used to publish, manage, and distribute CycloneDX SBOMs

 **8**  **38**

[proprietary](#) [analysis](#)

Black Duck

Synopsys

Black Duck software composition analysis (SCA) helps teams manage the security, quality, and license compliance risks that come from the use of open source and third-party code in applications and containers.

[proprietary](#) [analysis](#)

BlackBerry Jarvis

BlackBerry

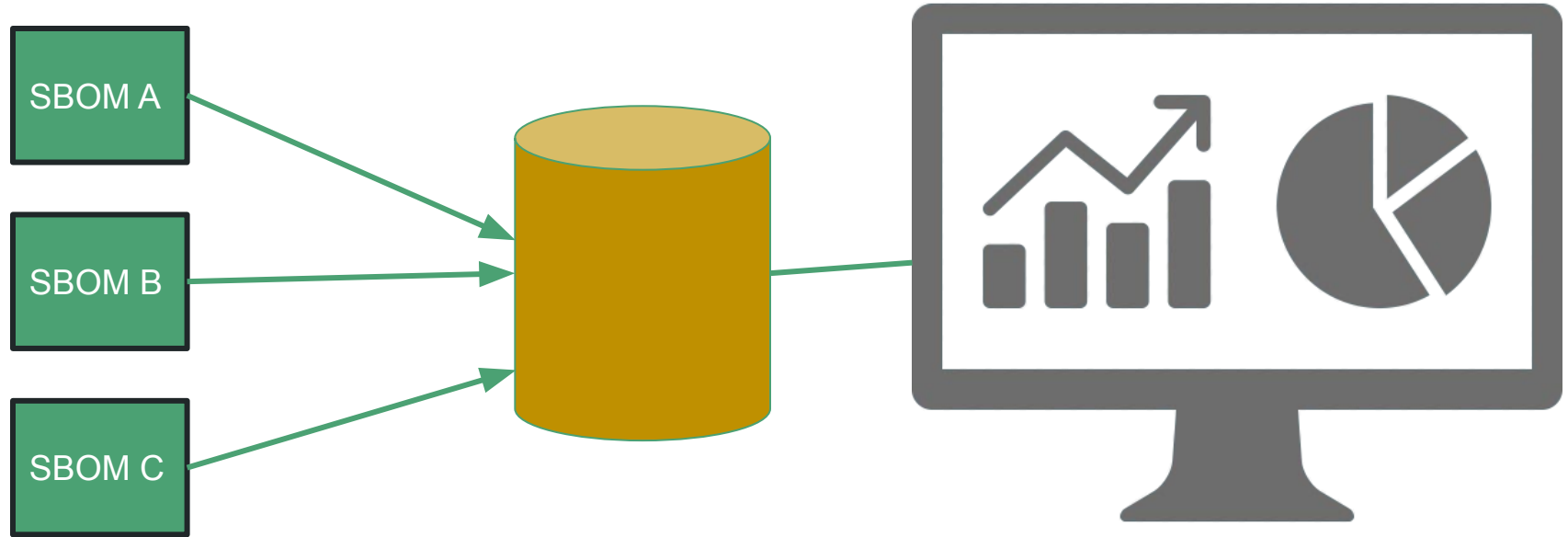
Software composition analysis (SCA) and security testing solution that detects and list open-source software and software licenses within embedded systems and associated cybersecurity vulnerabilities and exposures

[proprietary](#) [analysis](#)
[proprietary](#) [build-integration](#) [analysis](#) [github-ac...](#)
[opensource](#) [analysis](#)

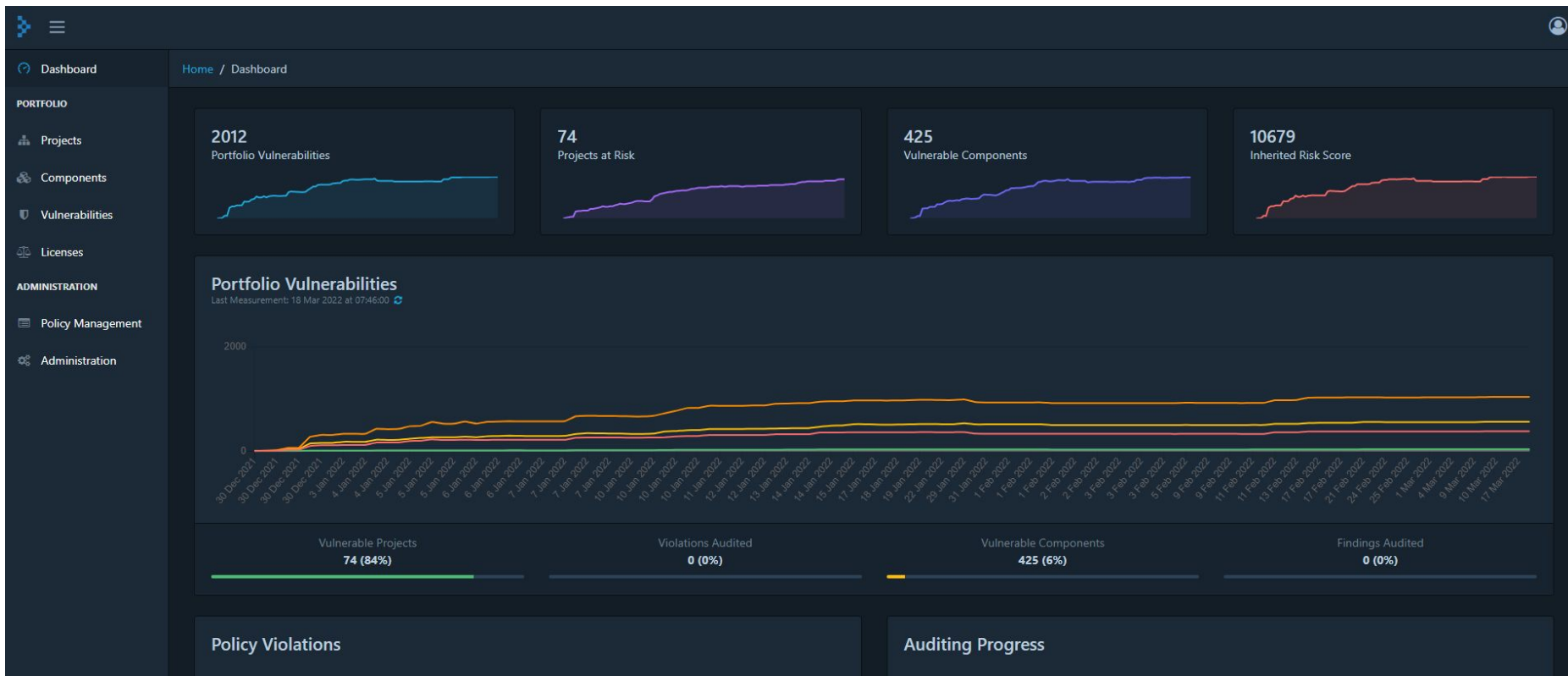
```
<component type="library" bom-ref="pkg:maven/org.apache.poi/poi@3.17?type=jar">
  <publisher>Apache Software Foundation</publisher>
  <group>org.apache.poi</group>
  <name>poi</name>
  <version>3.17</version>
  <description>Apache POI - Java API To Access Microsoft Format Files</description>
  <scope>optional</scope>
  <hashes>
    <hash alg="MD5">243bc3d431e4fadb79738719504c64f7</hash>
    <hash alg="SHA-1">0ae92292a2043888b40d418da97dc0b669fde326</hash>
    <hash alg="SHA-256">30181821dd2e849727b638b9e329aef4a64f3445c4142b13cf7a18bb3552edd</hash>
    <hash alg="SHA-384">f9a4db2b945cdba835b3c076c3f2bfa72767a5c02f81a57ffb4a5782651d5da66453c03e5c1be0505d1edad8a35c6f82</hash>
    <hash alg="SHA-512">a0c018d899935600b1077b343aac1a2e9050f84e3097e28b77869be9fc23475d4ac59bf375a2c96b6d824f06ba2d1873ee8d4495fb9bb412f848379ac7d11fb</hash>
    <hash alg="SHA3-256">84ba8a8001b44c04048d5ab1208640f0f5d54183231962cb180de7d7cfa7a386</hash>
    <hash alg="SHA3-384">2101589e294edd8d55f29be193e74c84afb8db754566b08b369c6c235e969af25ee8962e70f7c3dc262f7cbdebe3d57d</hash>
    <hash alg="SHA3-512">cc8b929d7f54aa98c606977bab56fcfe198e0bb21403b7b2a1bfff6bf2940901aee9d47bde493a98af376b4b1e75d110284023bd174b7774d454fac6e14b0606c</hash>
  </hashes>
  <licenses>
    <license>
      <id>Apache-2.0</id>
    </license>
  </licenses>
  <url>pkg:maven/org.apache.poi/poi@3.17?type=jar</url>
  <externalReferences>
    <reference type="website"><url>http://www.apache.org/</url></reference>
    <reference type="mailing-list"><url>http://mail-archives.apache.org/mod_mbox/poi-user/</url></reference>
  </externalReferences>
</component>
```



Zusammenführen



OWASP Dependency Track



Continuous Security Monitoring

Operationalize Software Bill of Materials





Dashboard

Home / Projects

PORTFOLIO

Projects

Components

Vulnerabilities

Licenses

ADMINISTRATION

Policy Management

Administration



[+ Create Project](#) Show inactive projects

Search

| Project Name | Version | Last BOM Import | BOM Format | Risk Score | Active | Vulnerabilities |
|----------------------|---------|-------------------------|---------------|------------|-------------------------------------|--|
| para80 | 0.1 | 18 Mar 2022 at 02:01:00 | CycloneDX 1.3 | 24 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 33%;"></div></div> 3 / 3 |
| wms | 0.1 | 18 Mar 2022 at 03:44:06 | CycloneDX 1.3 | 13 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 66%;"></div></div> 2 / 1 |
| rbus | 0.6.0 | 18 Jan 2022 at 03:15:27 | CycloneDX 1.3 | 69 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 100%;"></div></div> 1 / 7 / 6 |
| rbus | 0.7.0 | 18 Mar 2022 at 03:16:00 | CycloneDX 1.3 | 24 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 33%;"></div></div> 3 / 3 |
| zus-portal | 1.0.0 | 18 Mar 2022 at 03:05:15 | CycloneDX 1.3 | 3 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 100%;"></div></div> 1 |
| ema-portal | 1.0.0 | 18 Mar 2022 at 03:02:17 | CycloneDX 1.3 | 0 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 0%;"></div></div> 0 |
| para83a-online | 1.0.0 | 18 Mar 2022 at 03:12:04 | CycloneDX 1.3 | 50 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 100%;"></div></div> 1 / 5 / 5 |
| mgs-agb-online | 1.0.0 | 18 Mar 2022 at 03:24:55 | CycloneDX 1.3 | 0 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 0%;"></div></div> 0 |
| para80-online | 1.0.0 | 18 Mar 2022 at 01:07:06 | CycloneDX 1.3 | 50 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 100%;"></div></div> 1 / 5 / 5 |
| imorsaftplogimporter | 1.0.0 | 18 Mar 2022 at 02:19:28 | CycloneDX 1.3 | 3 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 100%;"></div></div> 1 |

Showing 1 to 10 of 88 rows rows per page

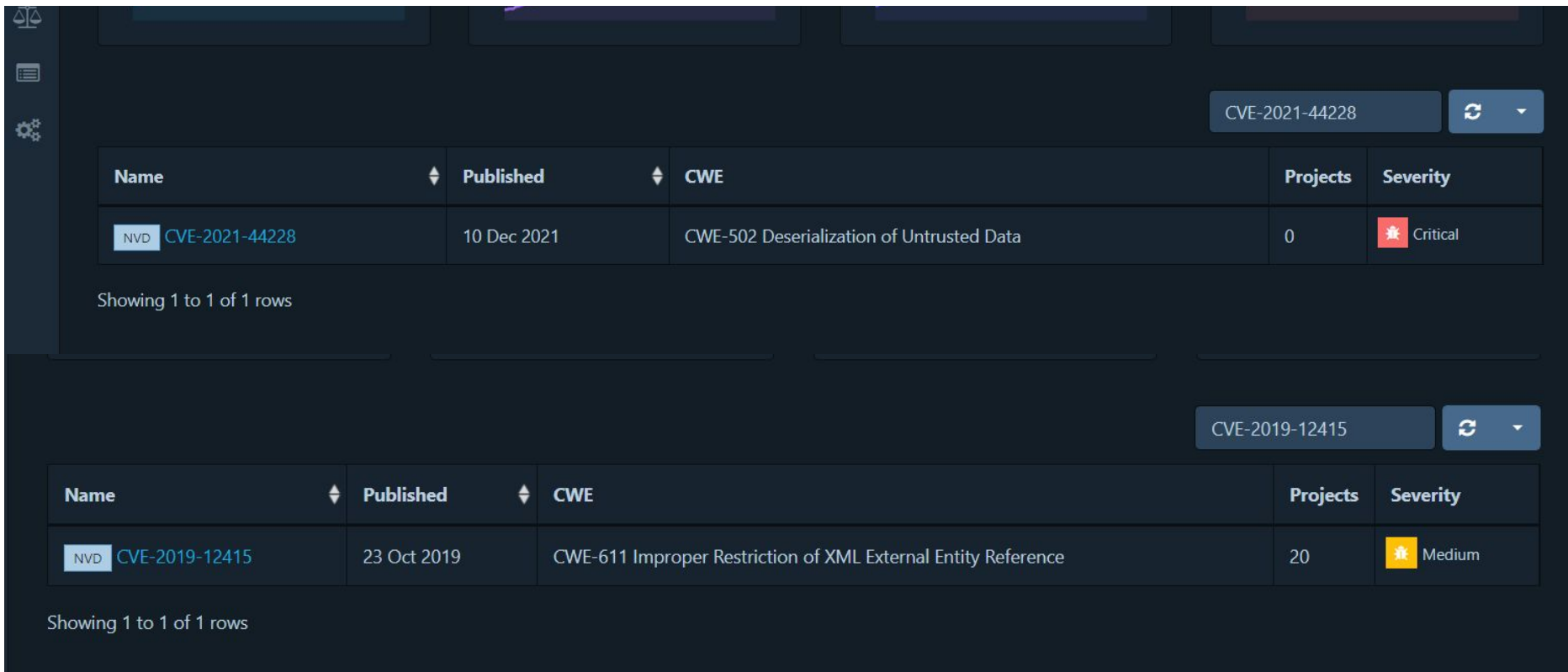
...

Suche nach Komponenten

The screenshot displays a search interface for software components. At the top, there are four line charts: 'Portfolio Vulnerabilities', 'Projects at Risk', 'Vulnerable Components', and 'Inherited Risk Score'. Below these is a search bar with the following filters: 'Coordinates' (checked), 'Group' (log4j), and 'Version'. A 'Search' button is on the right. The main area contains a table with the following columns: Component, Version, Group, Package URL (PURL), CPE, SWID Tag ID, and Project Name.

| Component | Version | Group | Package URL (PURL) | CPE | SWID Tag ID | Project Name |
|----------------|---------|--------------------------|---|-----|-------------|---------------------------------------|
| slf4j-log4j12 | 1.7.25 | org.slf4j | pkg:maven/org.slf4j/slf4j-log4j12@1.7.25?type=jar | | | zus 1.9.0 |
| slf4j-log4j12 | 1.7.25 | org.slf4j | pkg:maven/org.slf4j/slf4j-log4j12@1.7.25?type=jar | | | zus 1.9.1 |
| log4j-to-slf4j | 2.11.2 | org.apache.logging.log4j | pkg:maven/org.apache.logging.log4j/log4j-to-slf4j@2.11.2?type=jar | | | blzservice 1.1.0 |
| log4j-api | 2.11.2 | org.apache.logging.log4j | pkg:maven/org.apache.logging.log4j/log4j-api@2.11.2?type=jar | | | blzservice 1.1.0 |
| slf4j-log4j12 | 1.6.4 | org.slf4j | pkg:maven/org.slf4j/slf4j-log4j12@1.6.4?type=jar | | | dmpkeycloak 16.1.0 |
| log4j-to-slf4j | 2.13.3 | org.apache.logging.log4j | pkg:maven/org.apache.logging.log4j/log4j-to-slf4j@2.13.3?type=jar | | | mgs-statusexport-service 1.1-SNAPSHOT |
| log4j-api | 2.13.3 | org.apache.logging.log4j | pkg:maven/org.apache.logging.log4j/log4j-api@2.13.3?type=jar | | | mgs-statusexport-service 1.1-SNAPSHOT |
| log4j-to-slf4j | 2.11.2 | org.apache.logging.log4j | pkg:maven/org.apache.logging.log4j/log4j-to-slf4j@2.11.2?type=jar | | | mgs-mach-service 1.2.1 |
| log4j-api | 2.11.2 | org.apache.logging.log4j | pkg:maven/org.apache.logging.log4j/log4j-api@2.11.2?type=jar | | | mgs-mach-service 1.2.1 |
| log4j-to-slf4j | 2.11.2 | org.apache.logging.log4j | pkg:maven/org.apache.logging.log4j/log4j-to-slf4j@2.11.2?type=jar | | | templateservice 1.1-SNAPSHOT |

Suche nach Vulnerabilities



The screenshot displays a search interface for vulnerabilities. It features a search bar at the top right with the text 'CVE-2021-44228' and a refresh button. Below the search bar is a table with the following columns: Name, Published, CWE, Projects, and Severity. The first row shows the vulnerability 'CVE-2021-44228' published on '10 Dec 2021', with a CWE of 'CWE-502 Deserialization of Untrusted Data', 0 projects, and a 'Critical' severity level. Below the table, it says 'Showing 1 to 1 of 1 rows'. A second search bar at the bottom right contains 'CVE-2019-12415' and a refresh button. Below it is another table with the same columns. The second row shows 'CVE-2019-12415' published on '23 Oct 2019', with a CWE of 'CWE-611 Improper Restriction of XML External Entity Reference', 20 projects, and a 'Medium' severity level. Below this table, it also says 'Showing 1 to 1 of 1 rows'.

| Name | Published | CWE | Projects | Severity |
|------------------------------------|-------------|---|----------|----------|
| NVD CVE-2021-44228 | 10 Dec 2021 | CWE-502 Deserialization of Untrusted Data | 0 | Critical |

Showing 1 to 1 of 1 rows

| Name | Published | CWE | Projects | Severity |
|------------------------------------|-------------|---|----------|----------|
| NVD CVE-2019-12415 | 23 Oct 2019 | CWE-611 Improper Restriction of XML External Entity Reference | 20 | Medium |

Showing 1 to 1 of 1 rows

Projekt Komponenten

The screenshot shows a dashboard for managing project components. At the top right, there are five circular indicators with numbers: 1 (red), 6 (orange), 4 (yellow), 0 (green), and 1 (grey). Below the dashboard header, there are navigation tabs: Overview, Components (113), Services (0), Dependency Graph (1), Audit Vulnerabilities (12), Exploit Predictions (12), and Policy Violations (0). Below the tabs are buttons for '+ Add Component', '- Remove Component', 'Upload BOM', and 'Download BOM'. A search bar and a refresh icon are also present.

| Component | Version | Group | Internal | License | Risk Score | Vulnerabilities |
|-------------------------------|--------------|----------------------|----------|-----------------------|------------|-----------------|
| sshd-common | 2.7.0 | org.apache.sshd | | Apache-2.0 | 16 | 2 |
| jose4j | 0.7.11 | org.bitbucket.b_c | | Apache-2.0 | 10 | 1 |
| bcprov-jdk15on | 1.70 | org.bouncycastle | | Bouncy Castle Licence | 6 | 2 |
| wildfly-elytron-x500 | 1.19.1.Final | org.wildfly.security | | Apache-2.0 | 5 | 1 |
| wildfly-elytron | 1.19.1.Final | org.wildfly.security | | Apache-2.0 | 5 | 1 |
| wildfly-elytron-password-impl | 1.19.1.Final | org.wildfly.security | | Apache-2.0 | 5 | 1 |
| wildfly-elytron-credential | 1.19.1.Final | org.wildfly.security | | Apache-2.0 | 5 | 1 |
| wildfly-elytron-realm-ldap | 1.19.1.Final | org.wildfly.security | | Apache-2.0 | 5 | 1 |
| wildfly-elytron-realm-token | 1.19.1.Final | org.wildfly.security | | Apache-2.0 | 0 | 0 |

Dependency Graph

The screenshot shows a dependency graph tool interface for version 0.1.0. The top navigation bar includes tabs for Overview, Components (30), Services (0), Dependency Graph (1), Audit Vulnerabilities (0), Exploit Predictions (0), and Policy Violations (0). A search bar contains the text "Highlight outdated components".

The main area displays a dependency tree for the root artifact `para251 0.1.0`. The tree structure is as follows:

- `para251 0.1.0`
 - `pkg:maven/org.flywaydb/flyway-sqlserver@10.8.1` (depends on `pkg:maven/org.flywaydb/flyway-core@10.8.1`)
 - `pkg:maven/org.primefaces/primefaces@12.0.7`
 - `pkg:maven/de.bas.theme/bas-theme-jakarta@12.2.1` (+)
 - `pkg:maven/org.apache.commons/commons-lang3@3.14.0`
 - `pkg:maven/org.apache.commons/commons-collections4@4.4`
 - `pkg:maven/org.apache.commons/commons-csv@1.10.0`
 - `pkg:maven/org.thymeleaf/thymeleaf@3.1.2.RELEASE` (depends on `pkg:maven/ognl/ognl@3.3.4`, `pkg:maven/org.javassist/javassist@3.29.0-GA`, `pkg:maven/org.attoparser/attoparser@2.0.7.RELEASE`, `pkg:maven/org.unbescape/unbescape@1.1.6.RELEASE`, `pkg:maven/org.slf4j/slf4j-api@2.0.11`)
 - `pkg:maven/com.openhtmltopdf/openhtmltopdf-pdfbox@1.0.10` (+)
 - `pkg:maven/com.openhtmltopdf/openhtmltopdf-slf4j@1.0.10` (+)

Additional dependencies shown in the tree include:

- `pkg:maven/com.fasterxmljackson.dataformat/jackson-dataformat-toml@2.15.2` (depends on `pkg:maven/com.fasterxmljackson.core/jackson-databind@2.15.3`, `pkg:maven/com.fasterxmljackson.core/jackson-core@2.15.3`)
- `pkg:maven/com.fasterxmljackson.core/jackson-annotations@2.15.3`
- `pkg:maven/com.fasterxmljackson.core/jackson-core@2.15.3`
- `pkg:maven/com.google.code.gson/gson@2.8.9`

The bottom right corner of the interface features the handle `@kitencol`.

Audit Vulnerabilities

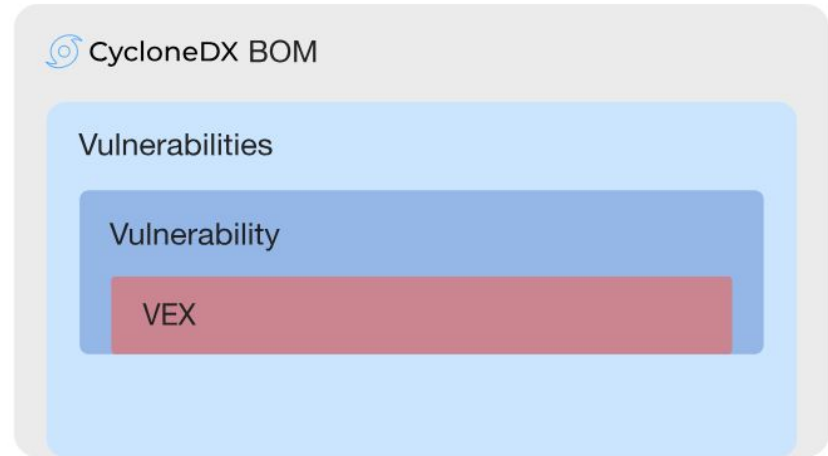
The dashboard displays a list of vulnerabilities for the version 1.3.0. At the top right, there are five circular indicators with counts: 1 (red), 6 (orange), 4 (yellow), 0 (green), and 1 (grey). Below the dashboard header, there are navigation tabs: Overview, Components (113), Services (0), Dependency Graph (1), Audit Vulnerabilities (12), Exploit Predictions (12), and Policy Violations (0). Below the tabs are action buttons: Apply VEX, Export VEX, Export VDR, Reanalyze, and a checkbox for 'Show suppressed findings'. A search bar and a refresh icon are also present.

| | Component | Version | Group | Vulnerability | Severity | Analyzer | Attributed On | Analysis | Suppressed |
|---|-------------------------------|--------------|----------------------|--------------------|----------|-----------|---------------|----------|------------|
| > | wildfly-elytron | 1.19.1.Final | org.wildfly.security | NVD CVE-2022-3143 | High | OSS Index | 25 Nov 2023 | - | |
| > | sshd-common | 2.7.0 | org.apache.sshd | NVD CVE-2022-45047 | Critical | OSS Index | 25 Nov 2023 | - | |
| > | sshd-common | 2.7.0 | org.apache.sshd | NVD CVE-2023-35887 | Medium | OSS Index | 25 Nov 2023 | - | |
| > | wildfly-elytron-credential | 1.19.1.Final | org.wildfly.security | NVD CVE-2022-3143 | High | OSS Index | 25 Nov 2023 | - | |
| > | jose4j | 0.7.11 | org.bitbucket.b_c | NVD CVE-2023-31582 | High | OSS Index | 25 Nov 2023 | - | |
| > | wildfly-elytron-password-impl | 1.19.1.Final | org.wildfly.security | NVD CVE-2022-3143 | High | OSS Index | 25 Nov 2023 | - | |
| > | wildfly-elytron-realm-ldap | 1.19.1.Final | org.wildfly.security | NVD CVE-2022-3143 | High | OSS Index | 25 Nov 2023 | - | |

VEX (Vulnerability Exploitability eXchange)

Möglichkeit zu kennzeichnen, dass ein Produkt NICHT von einer bestimmten Schwachstelle betroffen ist

<https://cyclonedx.org/capabilities/vex/>



Use-Case Beispiele:

<https://github.com/CycloneDX/bom-examples/tree/master/VEX/Use-Cases>

Apply VEX Export VEX Export VDR Reanalyze

Show suppressed findings

Search [Refresh] [Menu]

| Component | Version | Group | Vulnerability | Severity | Analyzer | Attributed On | Analysis | Suppressed |
|---------------|---------|--------------|--------------------|----------|-----------|---------------|--------------|------------|
| keycloak-core | 18.0.2 | org.keycloak | NVD CVE-2020-14359 | High | OSS Index | 27 Oct 2022 | Not Affected | |

Description

A vulnerability was found in all versions of Keycloak Gatekeeper, where on using lower case HTTP headers (via cURL) an attacker can bypass our Gatekeeper. Lower case headers are also accepted by some web servers (e.g. Jetty). This means there is no protection when we put a Gatekeeper in front of a Jetty server and use lowercase headers.

Audit Trail

- kaps - 12 Mar 2024 at 08:11:55
Analysis: NOT_SET ? NOT_AFFECTED
- kaps - 12 Mar 2024 at 08:12:17
Justification: NOT_SET ? PROTECTED_AT_RUNTIME
- kaps - 12 Mar 2024 at 08:12:31

Comment

[Empty comment box]

Add Comment

Analysis

Not Affected | Suppress

Justification Protected at runtime

Vendor Response (project) Will not fix

Details

mal sehen wo die Details auftauchen



schätzt die Wahrscheinlichkeit der Ausnutzung einer Schwachstelle in den nächsten 30 Tagen (Wert zwischen 0 und 1).

Zusätzlich das Perzentil des Anteils Schwachstellen mit Bewertung \leq

Die Auswirkungen spielen dabei keine Rolle.

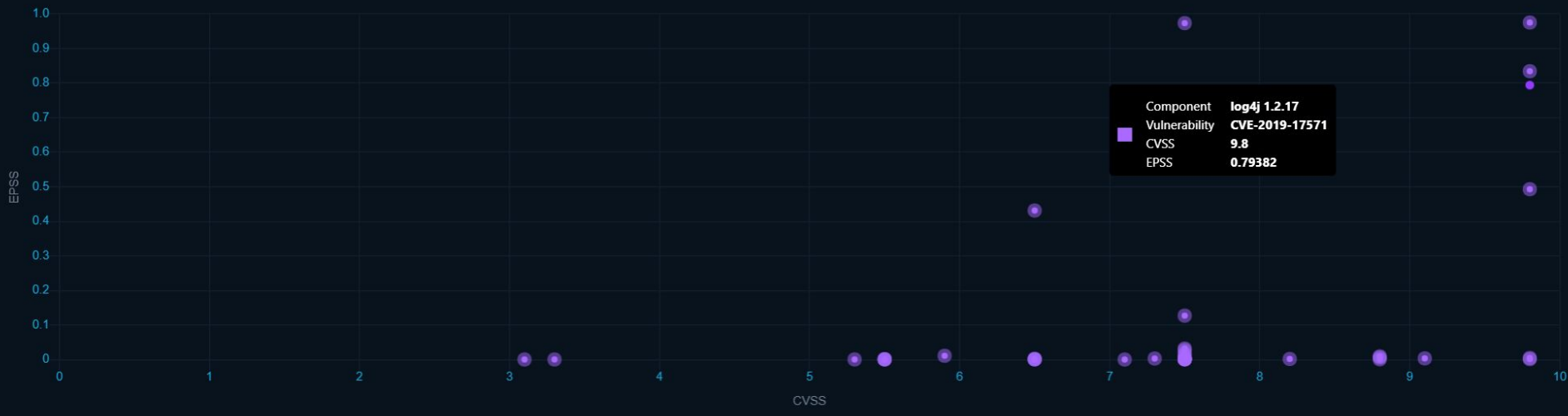
Ist kein vollständiges Risikobild und sollte nicht als solches betrachtet werden

<https://www.first.org/epss/>

Forum of Incident Response and Security Teams

View Details

- Overview
- Components **91**
- Services **0**
- Dependency Graph **1**
- Audit Vulnerabilities **41**
- Exploit Predictions **41**
- Policy Violations **0**



Show suppressed findings

Search [Refresh] [Menu]

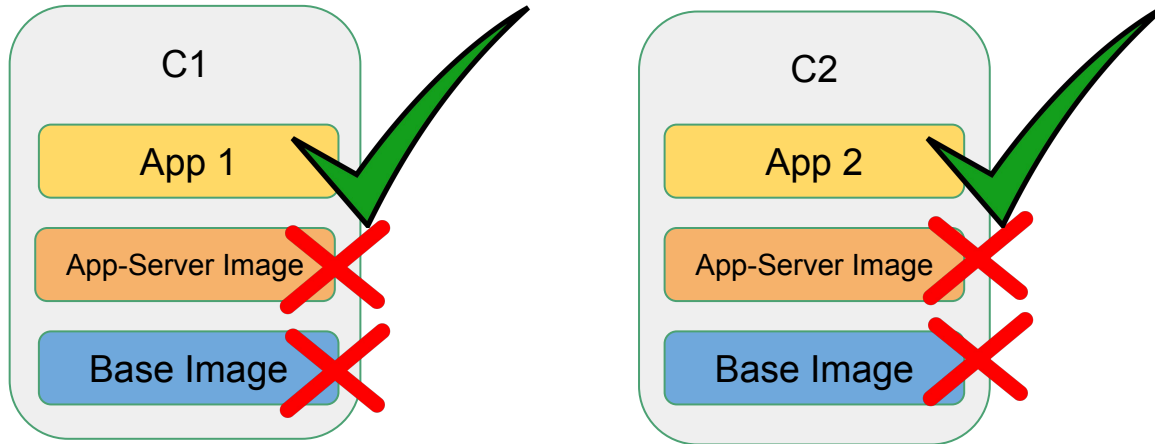
| Component | Version | Group | Vulnerability | CVSS | EPSS | EPSS Percentile | Suppressed |
|-------------------|---------------|---------------------|---------------------------|------|---------|-----------------|------------|
| spring-beans | 5.2.6.RELEASE | org.springframework | NVD CVE-2022-22965 | 9.8 | 0.97493 | 0.99971 | |
| commons-beanutils | 1.9.3 | commons-beanutils | NVD CVE-2014-0114 | - | 0.97314 | 0.99864 | |

Integration in Build-Prozess (Nightly)

```
stage('Stat. Analysen') {
  steps {
    withMaven(maven: 'Maven 3.8.6') {
      sh "mvn -U spotbugs:spotbugs org.cyclonedx:cyclonedx-maven-plugin:makeAggregateBom"
    }
    withCredentials([string(credentialsId: 'deptrack-apikey-text', variable: 'API_KEY')]) {
      dependencyTrackPublisher artifact: 'target/bom.xml', projectName: "${env.PROJECT}",
      projectVersion: "${env.OWNVERSION}", synchronous: true, dependencyTrackApiKey: API_KEY,
      failedNewCritical: 3, failedNewHigh: 5, failedNewLow: 15, failedNewMedium: 10,
      failedTotalCritical: 5, failedTotalHigh: 10, failedTotalLow: 50, failedTotalMedium: 30,
      unstableNewCritical: 1, unstableNewHigh: 2, unstableNewLow: 10, unstableNewMedium: 5,
      unstableTotalCritical: 1, unstableTotalHigh: 5, unstableTotalLow: 30, unstableTotalMedium: 15
    }
  }
}
```

Was ist mit der
Ausführungsschicht?

Container-Technologien



Server / Hypervisor / OS Layer / Container Engine

SBOMs für Container Images und Filesystem



syft

<https://github.com/anchore/syft>

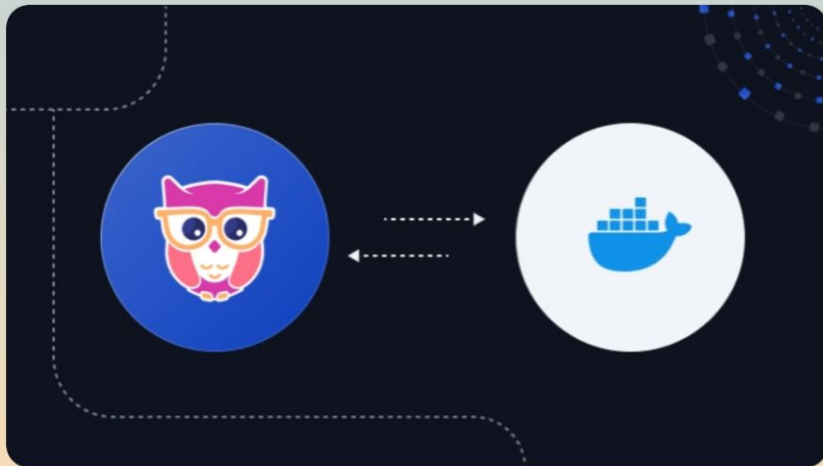


grype

<https://github.com/anchore/grype>

Anchore and Docker Release 'docker sbom' to Create Comprehensive SBOMs Based on Syft

By: Dan Nurmi APR 07, 2022 4 MIN READ



in

f

👍

🐦

Today Anchore and Docker [released](#) the first feature in what we anticipate will be an ongoing initiative to bring the value of the software bill of materials (SBOM) to all container-oriented build and publication systems. Now included in the latest Docker Desktop version is an operation called '[docker sbom](#)' that is available via the 'docker' command. This new operation, which is built on top of [Anchore's open source Syft project](#), enables Docker users to quickly generate detailed SBOM documents against container images using the native Docker CLI.

@kitencol

CLI Aufruf und direkte Auswertung

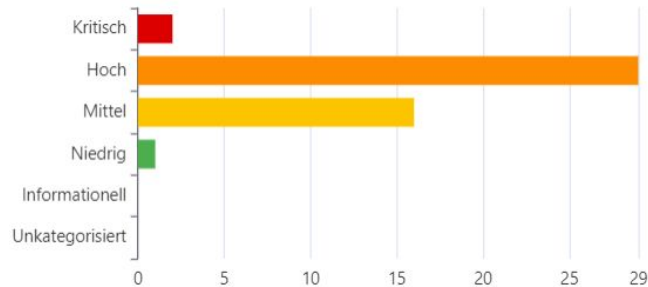
```
stage('Analyze Dependencies of Docker Image'){
  steps{
    script {
      dir ('out') {
        sh "syft -o cyclonedx-xml=sbom.xml ${env.IMAGE}:${env.TAG}"
        withCredentials([string(credentialsId: 'deptrack-apikey-text', variable: 'API_KEY')]) {
          dependencyTrackPublisher artifact: 'sbom.xml', projectName: "${env.IMAGE}",
            projectVersion: "${env.TAG}", synchronous: true, dependencyTrackApiKey: API_KEY,
            failedNewCritical: 3, failedNewHigh: 5, failedNewLow: 15, failedNewMedium: 10,
            failedTotalCritical: 5, failedTotalHigh: 10, failedTotalLow: 50, failedTotalMedium: 30,
            unstableNewCritical: 1, unstableNewHigh: 2, unstableNewLow: 10, unstableNewMedium: 5,
            unstableTotalCritical: 1, unstableTotalHigh: 5, unstableTotalLow: 30, unstableTotalMedium: 15
        }
      }
    }
  }
}
```


Syft findet ein bisschen mehr ...

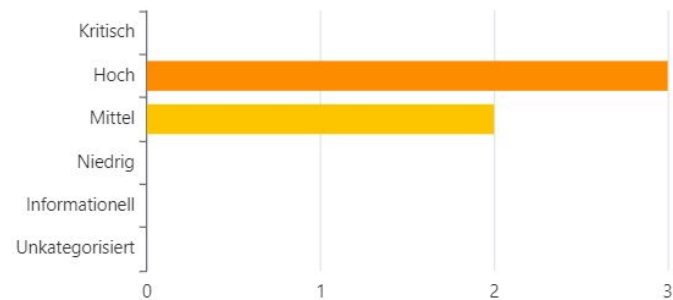
| Project Name | Version | Classifier | Last BOM Import | BOM Format | Risk Score | Active | Policy Violations | Vulnerabilities |
|--|-----------|------------|-------------------------|---------------|------------|-------------------------------------|---|--|
| mgs-keycloak | 19.0.1 | Library | 14 Sep 2022 at 01:33:29 | CycloneDX 1.4 | 21 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 100%; background-color: #f08080;">11</div></div> | <div style="width: 100%;"><div style="width: 100%; background-color: #ff9900;">3</div><div style="width: 100%; background-color: #ffcc00;">2</div></div> |
| etw-docker-03.bvaetw.de/mgs/mgs-keycloak | 19.0.1-19 | Container | 18 Aug 2022 at 17:45:40 | CycloneDX 1.4 | 246 | <input checked="" type="checkbox"/> | <div style="width: 100%;"><div style="width: 100%; background-color: #f08080;">3</div></div> | <div style="width: 100%;"><div style="width: 100%; background-color: #ff9900;">33</div><div style="width: 100%; background-color: #ffcc00;">20</div></div> |



Zusammenfassung der Abhängigkeitsanalyse



Zusammenfassung der Abhängigkeitsanalyse





View Details

Overview

Components 1208

Services 0

Dependency Graph 0

Audit Vulnerabilities 56

Policy Violations 3

+ Add Component

- Remove Component

Upload BOM

Download BOM

Search

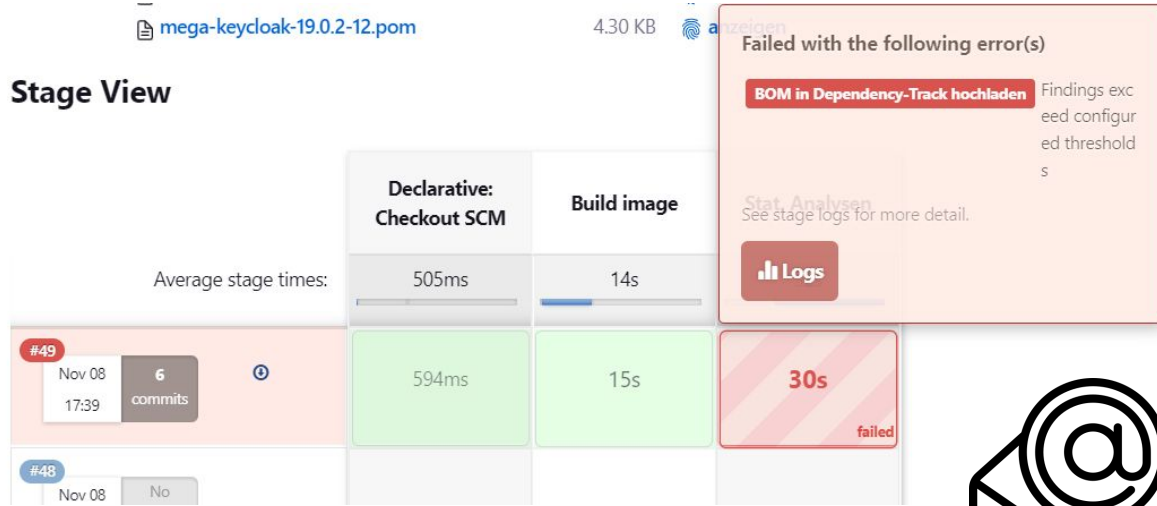


| Component | Version | Group | Internal | License | Risk Score | Vulnerabilities |
|-------------------|-----------------|----------------------|----------|---------|------------|-----------------|
| openldap | 2.4.46-18.el8 | | | | 96 | 1 16 2 |
| libarchive | 3.3.3-3.el8_5 | | | | 40 | 5 5 |
| freetype | 2.9.1-4.el8_3.1 | | | | 23 | 1 2 1 |
| keycloak-services | 19.0.1 | org.keycloak | | | 15 | 1 3 1 |
| pip | 9.0.3 | | | MIT | 13 | 2 1 |
| keycloak-core | 19.0.1 | org.keycloak | | | 13 | 2 1 |
| libjpeg-turbo | 1.5.3-12.el8 | | | | 11 | 1 2 |
| rhel | 8.6 | | | | 8 | 1 1 |
| snakeyaml | 1.30 | org.yaml | | | 8 | 1 1 |
| okhttp | 3.14.9 | com.squareup.okhttp3 | | | 5 | 1 |

Showing 1 to 10 of 1208 rows 10 rows per page

Was jetzt?

Organisatorisch



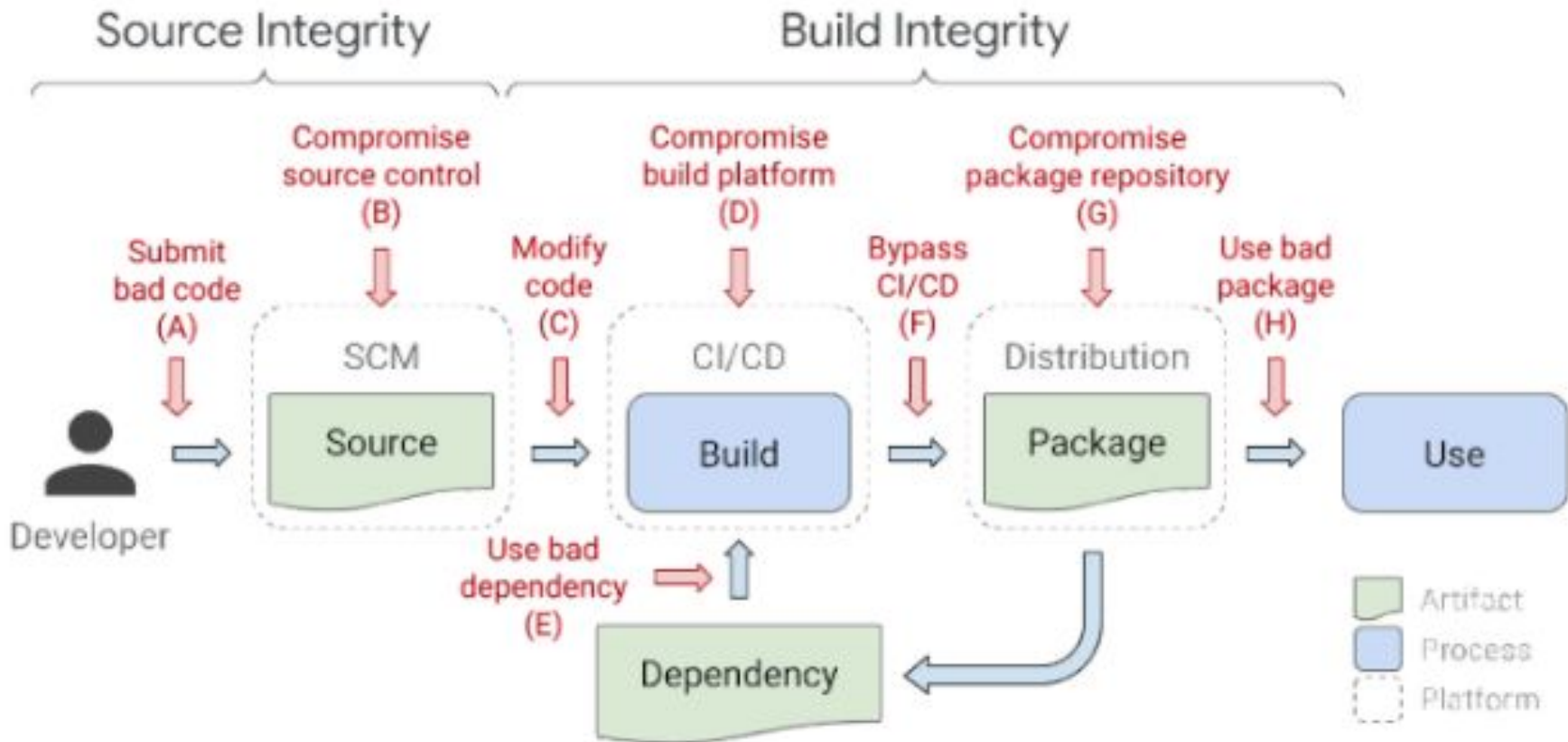
~~CVE-2022-12345~~



DEVOPS-1016 Der Nightly-Build von 'mega-keycloak' ist fehlgeschlagen

IN ARBEIT Nicht zugewiesen

Software Supply Chain ist mehr!



<https://opensource.googleblog.com/2021/10/protect-your-open-source-project-from-supply-chain-attacks.html>

Maßnahmen Entwicklung

Verified / Trusted Commits

Multi-Faktor Authentifizierung

Keyless Gitsign (<https://docs.sigstore.dev/signing/gitsign/>)

Github Branch Protection Rules

Require signed commits

Commits pushed to matching refs must have verified signatures.



Secure Coding

| Nr | Beschreibung | Risiko | Verwundbarkeit /Schwachstelle | Attacken | testbar durch |
|----|---|-------------------------|-------------------------------|----------------------|---|
| | | | control, XSS, etc. | Verwundbarkeiten | |
| 10 | Verhindere das unzulässige Umleiten zu anderen URLs | A10 - OWASP Top 10 2013 | CWE-601 | CAPEC-194 WASC-38 | FindSecBugs |
| 11 | Verhindere Click-Jacking Attacken | A5 - OWASP Top10 2013 | CWE-16 CWE-693 | WASC-15 | How To: Security-Tests mit Zed Attack Proxy (ZAP) |
| 12 | Verhindere Cross-Site-Scripting | A3 - OWASP Top10 2013 | CWE-79 | WASC-8 | How To: Security-Tests mit Zed Attack Proxy (ZAP) |
| 13 | Verhindere MIME Type sniffing | A3 - OWASP Top10 2013 | CWE-79 | WASC-8 | How To: Security-Tests |

https://wiki.mozilla.org/WebAppSec/Secure_Coding_Guidelines

<https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/>

Find Security Bugs



Plugin für SpotBugs

141 Bug-Pattern

Integration in Jenkins, SonarQube usw.

<https://find-sec-bugs.github.io/>

<https://owasp.org/www-project-find-security-bugs/>

Secure Coding & Testing

- Sichere Softwareentwicklung & Security Testing (SAST, DAST)



Sichere Softwareentwicklung: Einstieg in Secure-Coding und Continuous Security Testing

Sichere Softwareentwicklung: Einstieg in Secure-Coding und Continuous Security Testing

Für die eigene Web-Anwendung zuverlässige Prozesssicherheit als Sicherheitsmaßnahme vorzuziehen, ist ein Faktor von Continuous Delivery entscheidend. Die Nachhaltigkeit und Flexibilität von Funktionen stellt die IT-Sicherheit vor neue Herausforderungen und erfordert neue Lösungsansätze. Dieser Artikel stellt die Überlegungen und Erfahrungen bei der Einführung von Sicherheitslösungen für die eigene Entwicklung, von Java Web-Anwendungen und bei der Definition entsprechender Sicherheitsziele vor, die in die Software-entwicklungsprozesse integriert werden können.

Die Anforderungen an die IT-Sicherheit sind in den letzten Jahren erheblich gewachsen. Heute können Unternehmen, Behörden und Behördenverbände, die ihre Geschäftsprozesse digitalisieren und, wie 1970/1980 oder bei Computerviren, gefährdet sein. Die Anforderungen sind nicht nur die Produktion, sondern auch die Nutzung von IT-Systemen. Neben der Sicherheit ist es auch die Verfügbarkeit, die für Unternehmen von zentraler Bedeutung ist. Die IT-Sicherheit ist ein zentraler Bestandteil der Unternehmensstrategie. Die IT-Sicherheit ist ein zentraler Bestandteil der Unternehmensstrategie. Die IT-Sicherheit ist ein zentraler Bestandteil der Unternehmensstrategie.

Wie man beginnt

Die erste Maßnahme ist die Identifizierung der kritischen Assets. Die zweite Maßnahme ist die Identifizierung der Risiken. Die dritte Maßnahme ist die Identifizierung der Schwachstellen. Die vierte Maßnahme ist die Identifizierung der Bedrohungen. Die fünfte Maßnahme ist die Identifizierung der Auswirkungen. Die sechste Maßnahme ist die Identifizierung der Kontrollen. Die siebte Maßnahme ist die Identifizierung der Überwachungsmaßnahmen. Die achte Maßnahme ist die Identifizierung der Berichtsmaßnahmen. Die neunte Maßnahme ist die Identifizierung der Reaktionsmaßnahmen. Die zehnte Maßnahme ist die Identifizierung der Wiederherstellungsmaßnahmen.



FROSCON 2016
Stephan Kaps: Sichere Softwareentwicklung

<https://www.youtube.com/watch?v=7xwJMfA3mYQ&t=158s&pp=yqUMc3RlcGhhbiBrYXBz>

Architektur & Anforderungen

- Sicherheitskonzepte (IT Grundschutz)
- ggf. eigene BSI Bausteine
 - [↓ BausteinSecrets Managementmit Hashicorp Vault](#) / [↓ zur englischen Version](#) (Autor: Bundesamt für Soziale Sicherung)
 - [↓ Baustein Service-Proxy Traefik](#) (Autor: Bundesamt für Soziale Sicherung)
 - [↓ Baustein Hashicorp Consul](#) / [↓ zur englischen Version](#) (Autor: Bundesamt für Soziale Sicherung)
 - [↓ Baustein IAM Dienst Keycloak](#) (Autor: Bundesamt für Soziale Sicherung)
- https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/IT-Grundschutz/Hilfsmittel_un_d_Anwenderbeitraege/Hilfsmittel_von_Anwendern/Benutzerdefinierte-Bausteine/benutzerdefinierte-bausteine_node.html
- Richtlinien Kryptografie
- Standards zum Umgang mit eingehenden Daten
- Makroarchitektur
 - Sichere Sprachen (Rust statt C)
 - welche Tools sollten für eine Anforderung zum Einsatz kommen, welche nicht

Threat Modeling

Analyse der eigenen Software, der umliegenden Systeme und Prozesse

Strategien:

- Asset-zentriert
- Software-zentriert
- Risiko-zentriert
- Angreifer-zentriert

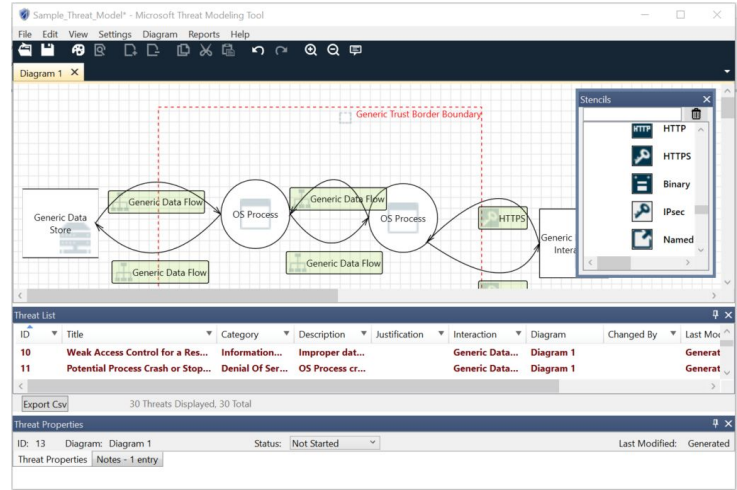
PASTA (Process for Attack Simulation and Threat Analysis)

STRIDE-Technik

<https://www.threatmodelingmanifesto.org/>



Threat Modeling Tools



<https://learn.microsoft.com/de-de/azure/security/develop/threat-modeling-tool>

<https://owasp.org/www-project-threat-dragon/>

Umgang mit Sicherheitswarnungen

Dokumentiertes Vorgehen

- Eingehende CERT Warnungen
<https://wid.cert-bund.de/>
- Warnungen durch statische Codeanalysen
- Warnungen durch Vulnerability Scans der Komponente
- Warnungen durch Vulnerability Scans auf Ausführungsebene



Aktuelle

Abonnements

CSAF

Über CERT-Bund

Fragen & Antworten

CSAF Einstellungen



Sicherheitshinweise

Abonnements

1 Datensätze

50 pro Seite



Risikostufe wählen

Alle abonnieren

Alle kündigen

Zurücksetzen

Speichern

(0) neu ausgewählt (0) abgewählt

| Abo | Produktname | Hersteller |
|-------------------------------------|-------------|------------|
| <input checked="" type="checkbox"/> | log4j | Apache |

Sensibilisierung & Training

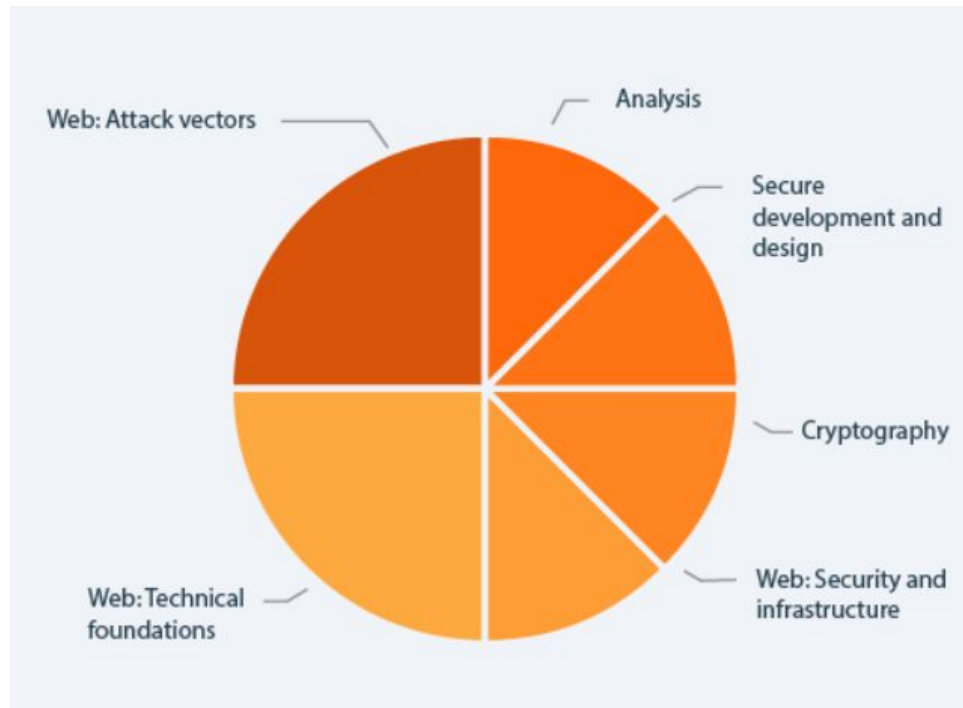
Informationsveranstaltungen

Live-Hacking

Schulungen

- ISAQB WebSec
- Certified Security Champion
- Fraunhofer IEM

TechTalks



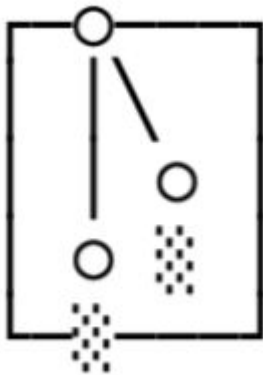
Secrets Detection



<https://docs.github.com/de/code-security/secret-scanning/about-secret-scanning>



https://docs.gitlab.com/ee/user/application_security/secret_detection/



<https://thoughtworks.github.io/talisman/>

<https://gitleaks.io/>



Verifizierung von Third-Party-Komponenten

Maßnahmen

- Erzeugung SBOM
- Zentrale Analyse der Verwundbarkeiten
- Benachrichtigung bei verwundbaren Abhängigkeiten
- Automatisierte Updates von Komponenten



<https://github.com/dependabot>



<https://github.com/renovatebot/renovate>

- Dokumentiertes Vorgehen für autom. Updates

Gehärtetes Build-System

Maßnahmen

- Authentifizierung
- Regelmäßige Patches
- Pipeline as Code
- Vertrauenswürdige Artefakt Repositories
- Sichere Kommunikations-Verbindungen
- Keine Secrets im Build-Tool



Überprüfung der Integrität der Artefakte

z.B. per Checksum, um veränderte Packages zu erkennen

Negativ-Beispiel: Github Packages

Namespace Shadowing (Dependency Confusion)

wie Paketmanager öffentliche gegenüber privaten Paketen priorisieren

Typosquatting (Schreibfehler im Namen)

cross-env statt crossenv: übermittelt alle Env-Variablen an Angreifer

<https://blog.npmjs.org/post/163723642530/crossenv-malware-on-the-npm-registry.html>

Aber Gott sei Dank benutzen wir Maven und Java

6.1.11. Downloading and Verifying Dependencies

The following command line options affect the way that Maven will interact with remote repositories and how it verifies downloaded artifacts:

-C, --strict-checksums

Fail the build if checksums don't match

-c, --lax-checksums

Warn if checksums don't match

-U, --update-snapshots

Forces a check for updated releases and snapshots on remote repositories

<https://books.sonatype.com/mvnref-book/reference/running-sect-options.html#running-sect-deps-option>

Ab Version 4.0.0 ist das der default!

<https://issues.apache.org/jira/browse/MNG-5728>

Attestierung

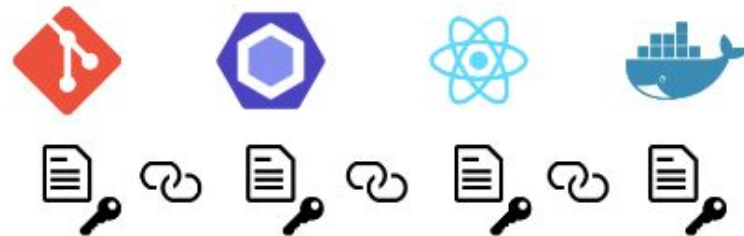
Eine Bescheinigung (Attest) ist eine kryptografisch signierte „Aussage“, die behauptet, dass etwas (ein „Prädikat“) über eine andere Sache (ein „Subjekt“) wahr ist.

Beispiel: Syft & Cosign

<https://anchore.com/sbom/creating-sbom-attestations-using-syft-and-sigstore/>

- Integriert in Syft Tool
- Neuer CLI Befehl “attest”
- Attestiert bspw. SBOM, kann z.B. in Container Registry abgelegt werden (durch cosign attached, als OCI annotation)
- Attest stellt in-toto aus
- Consumer kann mit Public Key verifizieren

Wo geht es hin?



bisher existierende Predicates:

<https://github.com/in-toto/attestation/tree/main/spec/predicates>

seit 11.03.24 verlangt die US-Regierung eine Bestätigung der Hersteller

<https://www.cisa.gov/secure-software-attestation-form>

mit öffentlichem Repo: <https://softwaresecurity.cisa.gov/>

→ SLSA

SLSA

SLSA ist eine Spezifikation zur Beschreibung und schrittweisen Verbesserung der Lieferkettensicherheit, die im Branchenkonsens festgelegt wurde. Es ist in eine Reihe von Ebenen gegliedert, die zunehmende Sicherheitsgarantien beschreiben.

Dabei handelt es sich um Version 1.0 der SLSA-Spezifikation, die die SLSA-Stufen und empfohlenen Bescheinigungsformate, einschließlich der Herkunft, definiert.

<https://www.slsa.dev>

Auslieferungsprozess

Image Signing mit Sigstore cosign



```
stage('Sign Image'){
  steps{
    script {
      dir ('out') {
        withCredentials([string(credentialsId: 'cosignkey', variable: 'COSIGN_KEY')]) {
          sh "cosign sign --key $COSIGN_KEY ${env.IMAGE}:${env.TAG}"
        }
      }
    }
  }
}
```

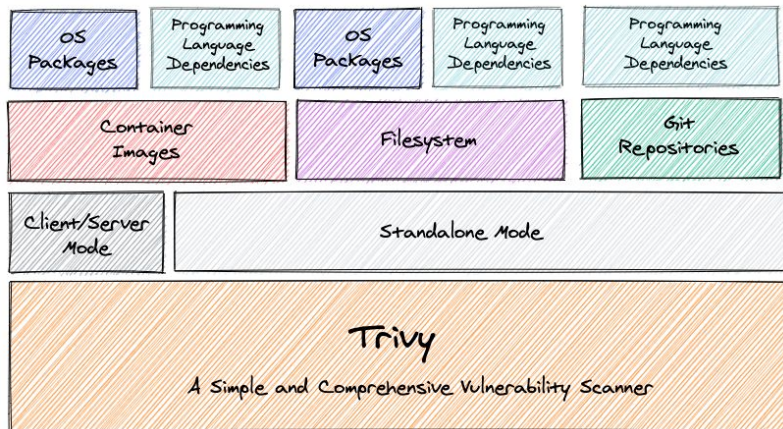
Secure Container Registry

- Vulnerability Scans
- Image Signing
- Pull Policies



HARBOR

<https://goharbor.io/>



Targets

Artifacts

Modes



aqua
trivy

<https://github.com/aquasecurity/trivy>

@kitencol

Vulnerability Scanning

| | | | | | | |
|---|----------------|--------|-----|---|-------------------------|---|
| > | CVE-2020-16156 | Hoch | | perl-base | 5.28.1-6+deb10u1 | |
| > | CVE-2020-25649 | Hoch | | com.fasterxml:jackson.core: jackson-databind | 2.10.4 |  2.10.5.1, 2.9.10.7, 2.6.7.4 |
| > | CVE-2021-40690 | Hoch | | org.apache.santuario:xmlsec | 2.1.4 |  2.1.7, 2.2.3 |
| > | CVE-2020-13949 | Hoch | | org.apache.thrift:libthrift | 0.13.0 |  0.14.0 |
| > | CVE-2020-13936 | Hoch | | org.apache.velocity:velocity- engine-core | 2.1 |  2.3 |
| > | CVE-2020-28052 | Hoch | | org.bouncycastle:bcprov- jdk15on | 1.65 |  1.67 |
| > | CVE-2020-25638 | Hoch | | org.hibernate:hibernate- core | 5.3.17.Final |  5.3.20.Final, 5.4.24.Final |
| > | CVE-2021-37714 | Hoch | | org.jsoup:jsoup | 1.8.3 |  1.14.2 |
| > | CVE-2014-3530 | Hoch | | org.picketlink:picketlink- common | 2.5.5.SP12-redhat-00009 |  2.6.1.Final |
| > | CVE-2019-25013 | Mittel | 4.8 | libc-bin | 2.28-10 | |

Image Signing mit Sigstore cosign



Info Artefakte

SCAN SCAN STOPPEN AKTIONEN Q ↻

| <input type="checkbox"/> | Artefakte | Pull Befehl | Tags | Signiert mit Cosign | Größe | Schwachstellen | Annotationen | Label | Push Zeit | Pull Zeit |
|--------------------------|-----------------|-------------|--------|---------------------|----------|--|--------------|-------|-----------------|-----------------|
| <input type="checkbox"/> | sha256:add50289 | | latest | | 86.14MiB | Keine Schwachstelle | | | 04.11.22, 02:31 | 07.11.22, 01:00 |
| <input type="checkbox"/> | sha256:e7d34034 | | | | 86.14MiB | Keine Schwachstelle | | | 27.10.22, 14:39 | 07.11.22, 01:01 |
| <input type="checkbox"/> | sha256:0edb0aea | | | | 88.11MiB | H 1 Gesamt - 1 Fixable | | | 04.10.22, 22:42 | 07.11.22, 01:01 |

Einträge pro Seite 15 1 - 3 von 3 Einträge

Pull Policy

The screenshot shows the Harbor web interface. The top navigation bar includes the Harbor logo and a search field. The left sidebar contains a menu with items like 'Projekte', 'Logs', 'Administration', 'Nutzer', 'Robot-Zugänge', 'Gruppen', 'Registries', 'Replikationen', 'P2P-Verteilung', 'Label', 'Projekt-Begrenzungen', 'Schwachstellen Scan', 'Clean Up', and 'Konfiguration'. The main content area is titled 'Konfiguration' and contains several sections: 'Projekt Registry' with an 'Öffentlich' checkbox; 'Deployment Sicherheit' with 'Cosign' and 'Verhindere den Download von Images mit Schwachstellen' checked; 'Scannen auf Schwachstellen' with 'Images automatisch beim Hochladen scannen' checked; and 'CVE Allowliste' with 'System-Allowliste' selected. At the bottom, there is a 'HINZUFÜGEN' button, a 'VON SYSTEM KOPIEREN' button, and a 'Läuft ab am' field with a dropdown menu set to 'Läuft niemals ab' and a checked 'Läuft niemals ab' checkbox. A text input field contains 'CVE-2021-43816'.

Harbor

Suche Harbor...

Zusammenfassung Repositories Mitglieder Labels Scanner P2P-Verteilung Policy Robot-Zugänge Webhooks Logs Konfiguration

Projekte

Logs

Administration

Nutzer

Robot-Zugänge

Gruppen

Registries

Replikationen

P2P-Verteilung

Label

Projekt-Begrenzungen

Schwachstellen Scan

Clean Up

Konfiguration

Projekt Registry Öffentlich

Ein Projekt öffentlich einzustellen, macht die Repositories für alle zugreifbar.

Deployment Sicherheit Cosign

Erlaube ausschließlich verifizierte Images.

Verhindere den Download von Images mit Schwachstellen.

Verhindere den Download von Images mit Schwachstellen des Schweregrads **Hoch** und darüber.

Scannen auf Schwachstellen Images automatisch beim Hochladen scannen

Scanne Images automatisch, wenn sie in das Projekt hochgeladen werden.

CVE Allowliste

Die Projekt-Allowliste erlaubt es, Schwachstellen in der Liste beim pushen und pullen von Images zu ignorieren.

Es kann entweder die Standard Allowliste des Systems verwendet werden. Alternativ kann über 'Projekt-Allowliste' eine neue Allowliste erstellt werden.

Individuelle CVE IDs hinzufügen, bevor über 'VON SYSTEM KOPIEREN' die System-Allowliste ebenfalls hinzugefügt wird.

System-Allowliste Projekt-Allowliste

HINZUFÜGEN

VON SYSTEM KOPIEREN

Läuft ab am

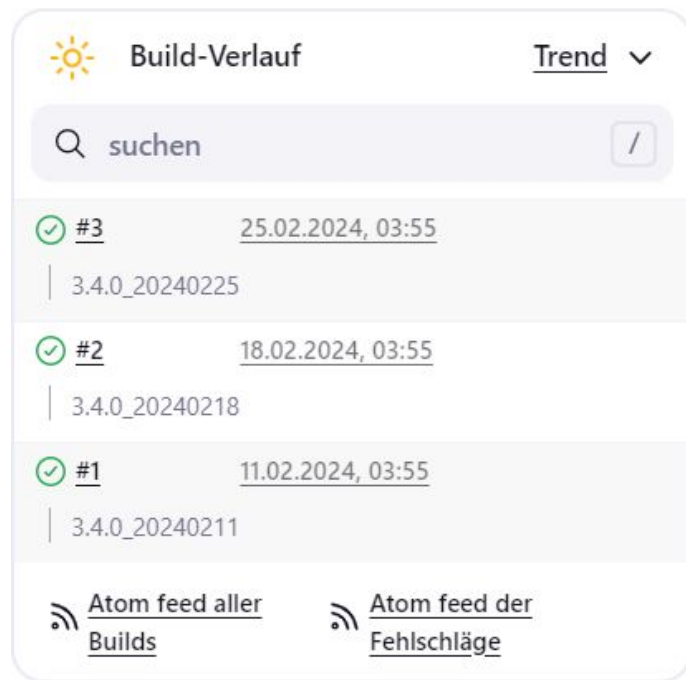
Läuft niemals ab

Läuft niemals ab

CVE-2021-43816

Automatisierte Basis-Images Updates

- Weekly Jobs im Build Tool
- Wenn letzter Commit mit “Produktion” getaggt ist
- Muss mit autom. Updates durch Bots synchronisiert werden

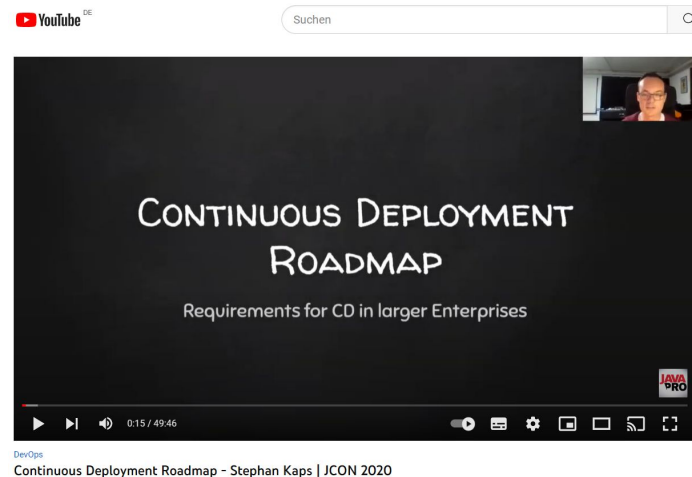


The screenshot shows a 'Build-Verlauf' (Build History) interface. At the top, there is a sun icon, the title 'Build-Verlauf', and a 'Trend' dropdown menu. Below the title is a search bar with the text 'suchen' and a search icon. The main content area displays a list of three builds, each with a green checkmark icon, a build number, a timestamp, and a version string. At the bottom, there are two RSS feed icons with labels: 'Atom feed aller Builds' and 'Atom feed der Fehlschläge'.

| Build # | Timestamp | Version |
|---------|-------------------|----------------|
| #3 | 25.02.2024, 03:55 | 3.4.0_20240225 |
| #2 | 18.02.2024, 03:55 | 3.4.0_20240218 |
| #1 | 11.02.2024, 03:55 | 3.4.0_20240211 |

Fähigkeit, schnell auszuliefern

- Continuous Delivery & Deployment



<https://entwickler.de/devops/roadmap-einer-spannenden-reise>

<https://devm.io/continuous-delivery/prerequisites-continuous-deployment-enterprises-001>

Runtime Security

Container Security

- Angriffsfläche verkleinern
- Privilegien minimieren
- Docker Engine härten



DevOps
In den sicheren Hafen: Einstieg in Container Security - Stephan Kaps | JCON 2020



<https://www.youtube.com/watch?v=GVavsR9jpC8&pp=ygUMc3RlcGhhbiBrYXBz>

OWASP CSVS

Sicherheitsvorgaben

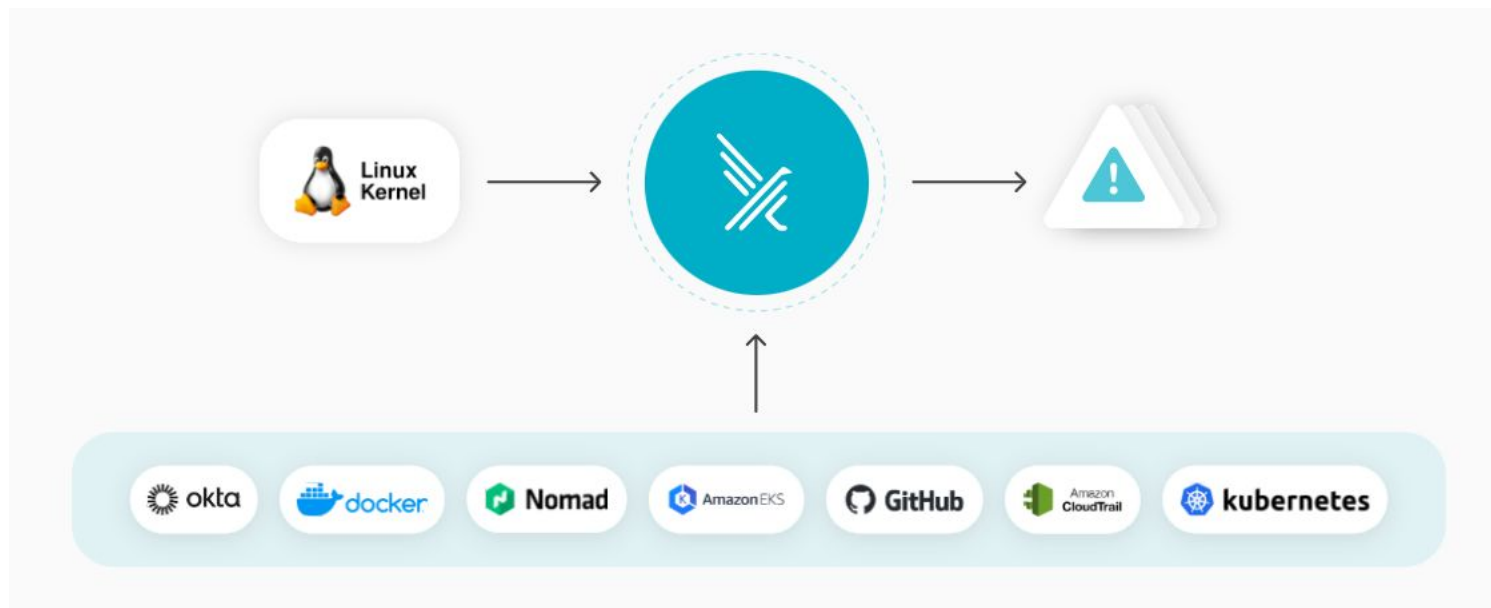
Bitte beachten Sie, dass die Anforderungen in diesem Abschnitt nicht vollständig sind, da sich viele organisatorische Sicherheitsvorgaben nicht nur auf Containerinfrastrukturen beschränken allerdings einen indirekten Einfluss auf diese haben können.

| # | Beschreibung | L1 | L2 | L3 | Seit |
|-----|---|----|----|----|------|
| 1.1 | Stellen Sie sicher, dass die technischen Mitarbeiter (insbesondere die mit DevOps-Aktivitäten beauftragten Mitarbeiter und Architekten) regelmäßig zu Sicherheitsaspekten der von ihnen verwendeten Technologien geschult werden. | ✓ | ✓ | ✓ | 1.0 |
| 1.2 | Stellen Sie sicher, dass die Manager regelmäßig zu Sicherheitsaspekten der in ihren Projekten verwendeten Technologien geschult werden. | | | ✓ | 1.0 |
| 1.3 | Stellen Sie sicher, dass alle verarbeiteten Daten gemäß vorhandenen internen Datenklassifizierungsvorgaben klassifiziert sind. | ✓ | ✓ | ✓ | 1.0 |

<https://github.com/OWASP/Container-Security-Verification-Standard>

Detektion von Anomalien

Falco ist ein Kernel-Überwachungs- und Erkennungsagent, der Ereignisse wie Systemaufrufe auf der Grundlage benutzerdefinierter Regeln beobachtet



<https://falco.org/>

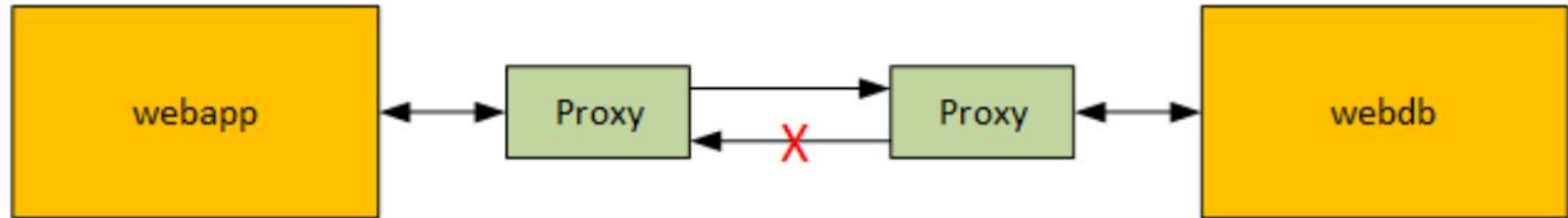
@kitencol

Standard Regel-Set

- Privilege escalation using privileged containers
- Namespace changes using tools like `setns`
- Read/Writes to well-known directories such as `/etc` , `/usr/bin` , `/usr/sbin` , etc
- Creating symlinks
- Ownership and Mode changes
- Unexpected network connections or socket mutations
- Spawned processes using `execve`
- Executing shell binaries such as `sh` , `bash` , `csch` , `zsh` , etc
- Executing SSH binaries such as `ssh` , `scp` , `sftp` , etc
- Mutating Linux `coreutils` executables
- Mutating login binaries
- Mutating `shadowutil` or `passwd` executables such as `shadowconfig` , `pwck` , `chpasswd` , `getpasswd` , `change` , `useradd` , etc , and others.

Isolierung

- Service-Mesh
- Mikro-Segmentierung
- Kommunikations-Regeln (Policies / Intentions)

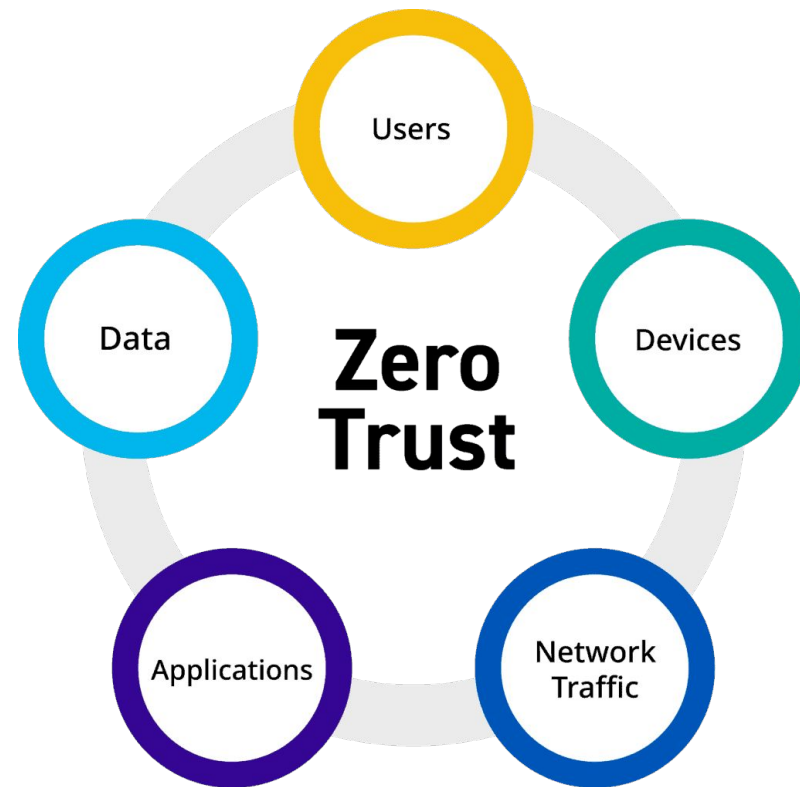


Zero Trust

Traue nichts und niemandem!

Maßnahmen:

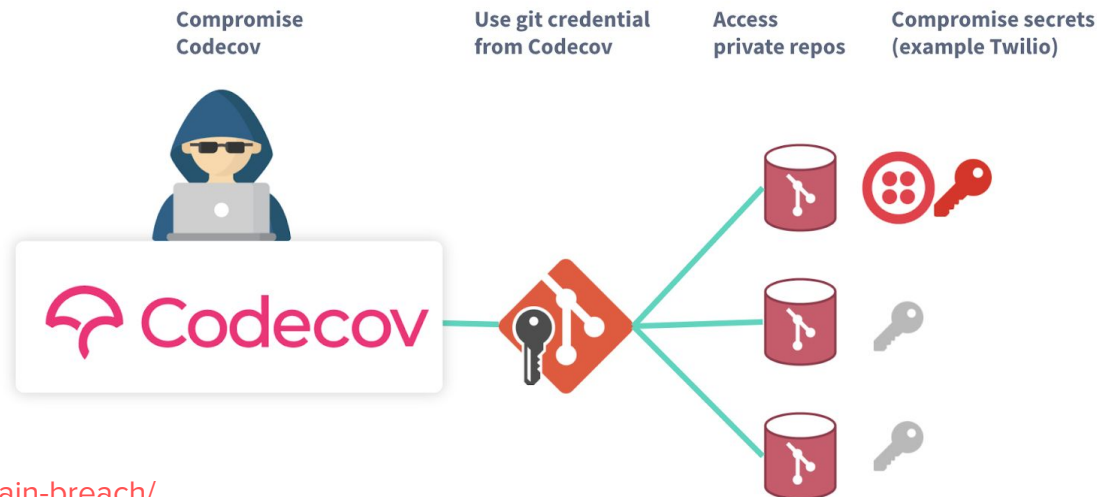
- kurzlebige Tokens
- häufig rotierte Zertifikate
- Mikrosegmentierung
- Mutual TLS
- Infrastructure as Code
- Workload Identities
- Kontinuierliche Überwachung



Quelle: <https://logrhythm.com/solutions/security/zero-trust-security-model/>

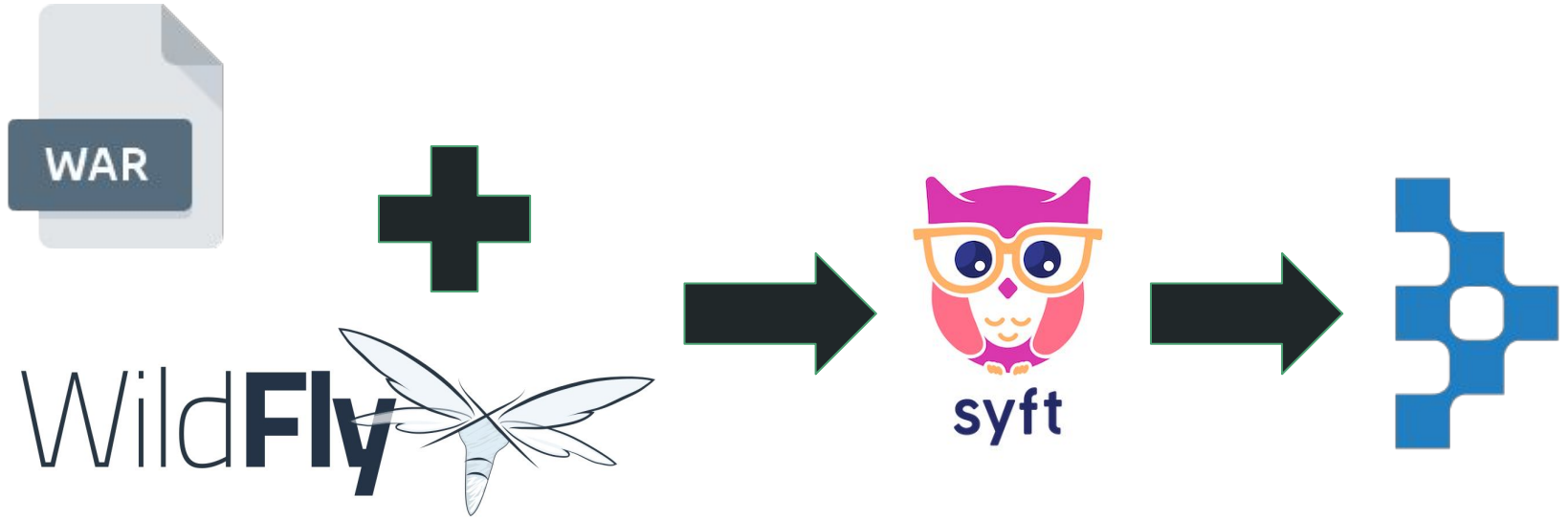
Beispiel: CodeCov (2021)

- Angreifer konnten Secrets aus Docker Images ziehen
- Schadhafte Skript eingeschleust (eine Zeile verändert)
- Umgebungsvariablen aus CI gezogen und übermittelt
- Damit auf Git Repos zugegriffen und Secrets kompromittiert
- Und das von mehreren Kunden, an die die Software ausgeliefert wurde



<https://blog.gitguardian.com/codecov-supply-chain-breach/>

Was war jetzt mit
Kaufsoftware?



Aufnahme von Anforderung in Ausschreibungen

“Bereitstellung einer SBOM bei jeder ausgelieferten Version”

“Zeitnahe Bereitstellung von Patches bei neuen Sicherheitslücken”

2. Herausforderung

Wie können wir sicherzustellen, dass wir nicht versehentlich anfällige Versionen von log4j aufnehmen und bereitstellen?

... oder dass jemand uns etwas Schadhafes unterschiebt?

Jemand eine Idee?

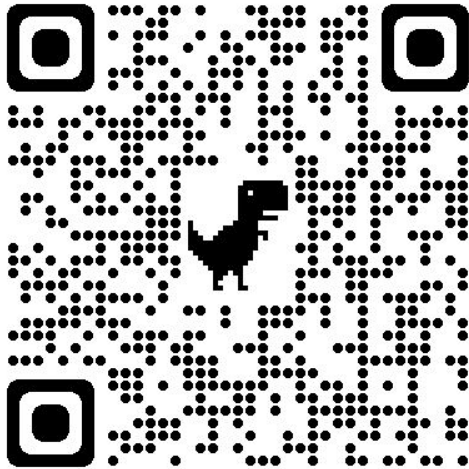
- Überprüfen der Integrität der Artefakte
- Vulnerability Scans, Image Signing & Pull Policy
- Vertrauenswürdige Artefakt-Repositories
- Container-Security Maßnahmen & Härtung der Hosts
- Attestierung und deren Prüfung
- Sensibilisierung & Training

Fazit

Wie kann man starten?

- Generierung von SBOMs für eigenentwickelte Software
(automatisiert im Rahmen von CI/CD)
- Verpflichtung der Hersteller von Software zur Bereitstellung einer SBOM zu jeder ausgelieferten Version
- Einführung einer Software zur kontinuierlichen Analyse von SBOMs
- Zusätzlich Ausführungsschicht überwachen
- Plan zum Umgang mit Ergebnissen überlegen
- Kontinuierlich fixen und deployen

Infographic zur freien Verfügung



<https://kitenco.de/download/SBOM-Infographic.png>

Verpflichtung der Hersteller zur Bereitstellung eines Beispackzettels (SBOM) für Ihre Software

WARUM?

Um Risiken in der Software-Supply-Chain frühzeitig zu identifizieren

WAS BEDEUTET DAS?

Es war NEM, das erste mal, das eine Schwachstelle in einer Open Source Bibliothek für Überstunden in 17 Millionen genutzt hat. Doch im Dezember 2022, war die CVE-2022-44228 zu Longi bzw. Logshell sogar in der Speicher- und der Spring-Code des Internet-Browsers.

Doch was haben wir daraus gelernt?

Haben wir Schwachstellen gemeldet oder Indikatoren ergreift, um bei der nächsten Open-Source-Materialität offenere agieren zu können? Können wir z.B. schneller herausfinden, in welchen Produkten eine Bibliothek in einer bestimmten Version verwendet wird?

Haben wir aktuelle Beispackzettel zu jeder Software, sogenannte SBOM, haben wir dazu in der Lage.

Was ist eine SBOM?

SBOM steht für Software Bill of Materials und ist eine Tabelle in Form einer maschinenlesbaren Struktur, die enthält eine Auflistung aller verwendeten Bibliotheken bzw. Komponenten, sowie deren direkt und indirekte Abhängigkeiten und deren zugehörige Identifizierung. Solche Daten können viele weitere Informationen enthalten wie wie z.B. Lizenzinformationen.

Es existieren Standards wie SPDX (Standard Software Package Data Exchange), die sich als SBOM-ISO/IEC 23247 genannt in Open Source Tools wie CycloneDX und OWASP oder auch von Anbietern können SBOM generieren.

Log4Shell oder SolarWinds
Was haben wir daraus gelernt?

| | | |
|--|--|---|
| <p>PRODUKT-INHALT IST UNBEKANNT</p> <p>Alles ist bekannt ist Software, was für wir nicht wissen, was für Hersteller, Kaufbehörden für, managen und Best-Prac. Dinge ist auch bekannt, ist dieses Produkt von einer anderen Softwareversion betroffen ist.</p> | <p>RECHERCHEN SIND AUFWENDIG</p> <p>Welches überlegener Software-Produkt eine spezifisch relevante, risikoreich Komponenten auf Best-Prac. Versionen oder Patch, per C-Mod oder per Netzwerk bei jedem einzelnen Produkt Hersteller angefragt werden.</p> | <p>WIR SIND EVENTUELL ANGREIFBAR</p> <p>Indirekt wird ein Angriffspunkt, aber Hersteller ergreift auf und können kritisch helfen, muss viele Anzeigen der Hersteller zu erhalten. Angreifende können Systeme angreifen werden.</p> |
|--|--|---|

Was können wir selbst tun?

| | |
|--|--|
| <p>Regelmäßige Analyse auf Sicherheitslücken</p> <p>Für eigenentwickelte Software bzw. von einer SBOM zur Verfügung steht, ist die Analyse der Schwachstellen in verschiedenen Identifikatoren rechtlich möglich. Durch geeignete Tools ist auch ein Überblick über das gesamte Anwendungsspektrum aufwandarm erstellbar.</p> | <p>Aufnahme der Verpflichtung in Ausschreibungen</p> <p>In künftigen Ausschreibungen sollte aufgenommen werden, dass Hersteller von Software verpflichtet sind, zu jeder angefragten Version auch eine SBOM bereitzustellen. Zusätzlich sollten SLAs aufgenommen werden, die eine zeitliche Definition von Sicherheitslücken festlegen.</p> |
|--|--|

© Kitenco AG (E.ON Energy Research Center) | 100% Softwareentwicklung im Rechenzentrum für Deutsche Telekom

OWASP Software Component Verification Standard (SCVS)

V1: Inventaranforderungen

V2: Anforderungen an Software Bill of Materials

V3: Anforderungen an die Build-Umgebung

V4: Anforderungen an die Paketverwaltung

V5: Anforderungen an die Komponentenanalyse

V6: Stammbaum- und Herkunftsanforderungen

<https://owasp.org/www-project-software-component-verification-standard/> bzw. <https://scvs.owasp.org/>

Maßnahmen: Entwicklung

| | |
|---------------------------------|---|
| Verified Commits | Bei dem Pushen in die Versionsverwaltung sollte ein zweiter Faktor zur Authentifizierung verwendet werden |
| Secure Coding | Für die Entwicklung von Software sollte ein Secure Coding Guide gepflegt und publiziert werden |
| Architektur & Anforderungen | Es müssen dokumentierte Anforderungen zu Sicherheitsaspekten in der Architektur existieren. |
| Threat Models | Es sollte zu jedem Produkt bzw. einer Produktgruppe ein entsprechendes Threat Model erstellt werden. |
| Umgang mit Sicherheitswarnungen | Es sollte ein Vorgehen definiert sein, wie mit neuen Sicherheitswarnungen umgegangen wird. |
| Sensibilisierung & Training | Es sollten regelmäßige Sensibilisierungsmaßnahmen bzw. Trainings für die Entwickler /innen stattfinden |
| Security Testing | Es sollten statische (SAST) und dynamische (DAST) Tests existieren, die das Produkt auf Sicherheitslücken hin überprüfen. |
| Secrets Detection | Es sollte vor dem Commit in die Versionsverwaltung eine Überprüfung stattfinden, ob sensible Daten eingecheckt werden. Zusätzlich sollte ein Analyse Werkzeug im Nightly Build die gesamte Git Historie eines Produktes nach sensiblen Daten scannen und alarmieren. |

Maßnahmen: Verifizierung von Third-Party Komponenten

| | |
|--|---|
| Erzeugung einer SBOM | Jede Komponente muss bei jedem Nightly-Build eine Software Bill of Materials erzeugen, in der alle verwendeten Third-Party-Libraries enthalten sind. |
| Zentrale Analyse der Verwundbarkeiten | Jede Komponente muss ihre erzeugte SBOM an eine zentrale Stelle zur Analyse senden (Software Composition Analysis SCA) |
| Benachrichtigung bei verwundbaren Abhängigkeiten | Jeder Nightly-Build muss bei einer Überschreitung des definierten Quality Gates abrechnen und die Entwickler darüber informieren |
| Automatisierte Updates von Komponenten | Jedes Projekt in der Versionsverwaltung sollte mit Hilfe eines Bots automatisierte Merge-Requests für neue Versionen von Third-Party Komponenten erstellen. |
| Dokumentiertes Vorgehen für Updates | Es sollte das Vorgehen dokumentiert sein, wie die Merge-Requests der automatisierten Updates in das Produkt überführt und produktiv genommen werden. |

Maßnahmen: Gehärtetes Build-System

| | |
|---|---|
| Authentifizierung | Das Ausführung von Jobs im Build System darf nur von wenigen autorisierten Personen durchführbar sein, da ein Deployment bis in die Produktion möglich ist. |
| Regelmäßige Patches | Das Build System und alle darin verwendeten Werkzeuge müssen regelmäßig aktualisiert werden, damit auch deren Sicherheitslücken zeitnah geschlossen werden |
| Pipeline as Code | Pipelines für das Bauen oder Deployen von Komponenten müssen geskriptet sein, um deren autom. Durchführung und Reproduzierbarkeit sicherzustellen |
| Vertrauenswürdige Artefakt-Repositories | Zusätzliche Bibliotheken für den Build-Prozess sollten nur aus dem internen Artefakte Repository oder dem ans Internet angebotenen Mirror erfolgen |
| Sichere Kommunikations-Verbindungen | Systeme, die am Build- und Deployment Prozess beteiligt sind, sollten keine direkte Verbindung ins Internet haben. Des Weiteren müssen alle Verbindungen transportverschlüsselt sein.. |
| Integrität der Artefakte | Die im Build-Prozess verwendeten Third-Party-Libraries sollten auf ihre Integrität hin überprüft werden. |
| Umgang mit Secrets | Es dürfen keine Zugangsdaten zu Systemen im Klartext im Build-System konfiguriert sein. |
| Attestierung | Um die Qualität des Builds zu bescheinigen, werden Umgebung, Prozess, Material und Artefakte auf ihre Integrität hin überprüft, um sicherzustellen, dass jede Aufgabe in der Chain wie geplant und nur von autorisiertem Personal ausgeführt und das Produkt während des Transports nicht manipuliert wird. |

Maßnahmen: Auslieferungsprozess

| | |
|------------------------------------|--|
| Signieren der Images | Jedes durch das Build System gebaute Image sollte signiert werden.. |
| Pull-Policy | Auf den Hosts dürfen nur signierte Images gepullt werden. |
| Automatisierte Basis-Image Updates | Jedes Produkt sollte regelmäßig Updates des Basis-Images durchführen und sehr zeitnah auch produktiv nehmen. |

Maßnahmen: Runtime Security

| | |
|-------------------------|--|
| Container Security | Angriffsfläche verkleinern, Härtung, Einsatz nur von geprüften oder offiziellen Third-Party-Images |
| Detektion von Anomalien | Die auf den Hosts betriebenen Container sollten kontinuierlich auf Anomalien hin überprüft werden, z.B. durch Intrusion Detection |
| Mikrosegmentierung | Zur besseren Isolation sollten Container in separaten logischen Netzen betrieben und mit Zugriffsregeln versehen werden. |
| Zero Trust | Es sollte keiner Komponente im Verbund vertraut werden, sondern stattdessen immer eine Authentifizierung oder andere Arten der Absicherung verwendet werden. |

Ende

- unterstützt bei der NIS2 Umsetzung
- Security-by-design
- Erprobe den Ernstfall
- Das muss alles nicht viel kosten!



Vielen Dank!

Kontakt:

Stephan Kaps
info@kitenco.de

www.kitenco.de



@kitenco1



https://www.xing.com/profile/Stephan_Kaps2



<https://www.linkedin.com/in/stephan-kaps-b246b0ab/>



<http://de.slideshare.net/kitenco>

Quelle aller Hintergrundgrafiken ist <https://www.pixabay.com>

@kitenco1

MC Flavour

Der Nächste Request



<https://open.spotify.com/album/2tHppzsY0ZPb57Xa7PRkEX>