

# Risiken in der Software Supply Chain identifizieren, mit OWASP Tools

---

Was haben wir aus Log4j gelernt?

# Stephan Kaps

Leiter Softwareentwicklung Bundesamt für Soziale Sicherung

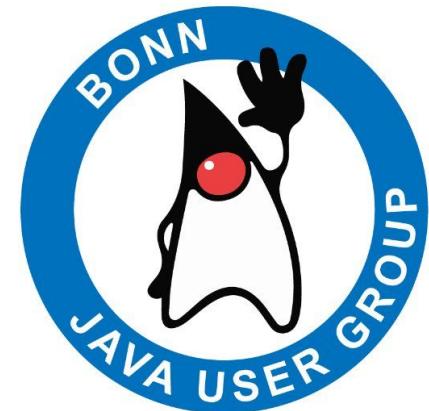
Gründer Java User Group Bonn

Java Entwickler & Architekt seit 2002

Weitere Schwerpunkte:

- Softwareentwicklungsprozesse
- BizDevSecOps
- OpenSource
- Speaker & Autor

[bizdevops.org](http://bizdevops.org)



# Wer sind wir uns was machen wir?

(Aufsichts-) Behörde mit ca. 700 Mitarbeiter

ca. 60 Fachanwendungen (Produkte)

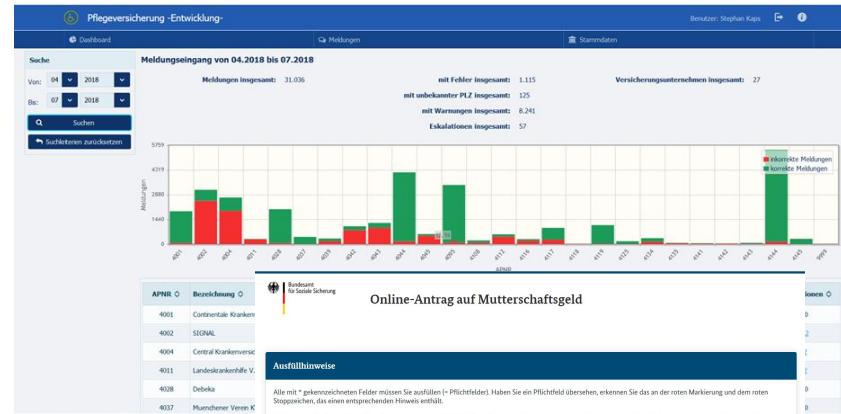
für interne Fachbereiche & Landesprüfdienste

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

## 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

with OWASP, CycloneDX improves on the older SPDX standard with a

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

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

Download

Adopt

## Why Do Developers Implementing State Management in React

Ever since Redux was released, we've seen a steady stream of tools and frameworks that manage state in different ways, each with a different set of trade-offs. We don't know exactly why, but we can only speculate: Is this the result of churn in the JavaScript ecosystem, or is it to promote?



see?

lecting what we came might have covered 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**

Apipro

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

 [Forks 51](#)  [Stars 184](#)

opensource distribute

**BOM Repository Server**

CycloneDX

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

 [Forks 8](#)  [Stars 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 lists open-source software and software licenses within embedded systems and associated cybersecurity vulnerabilities and exposures

proprietary analysis

proprietary build-integration analysis github-ac...

@kitencol

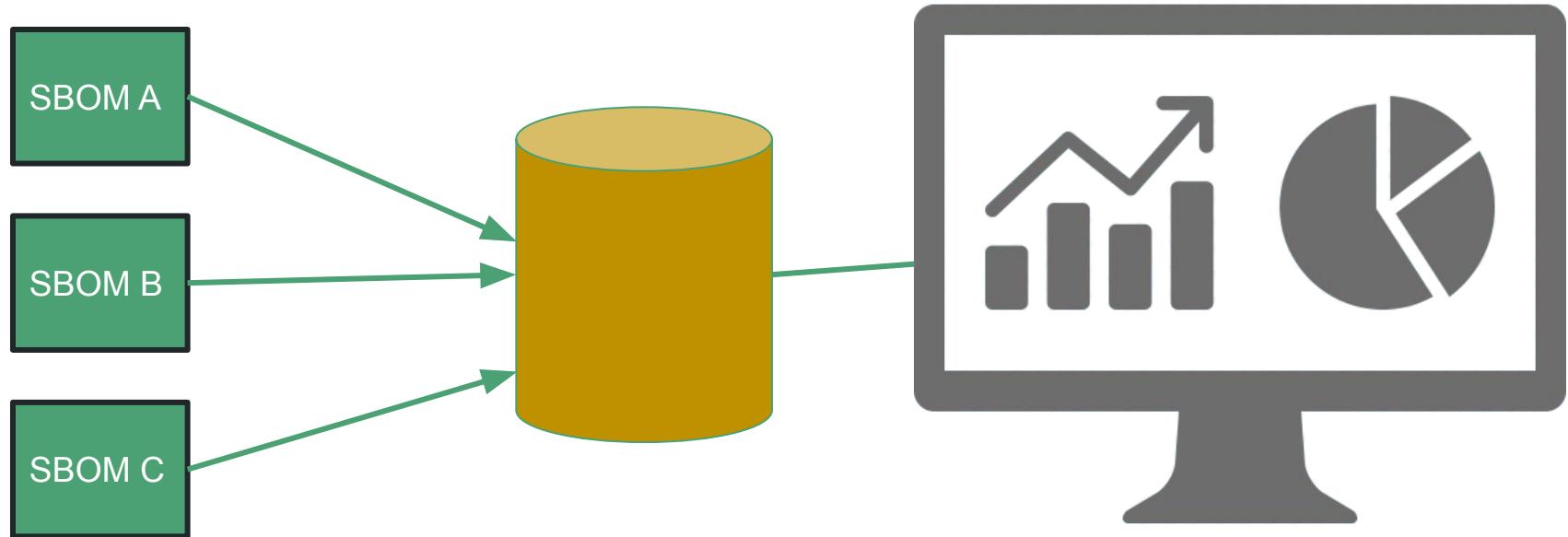
opensource analysis



```
<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">a0c018d8999356000b1077b343aac1a2e9050f84e3097e28b77869be9fc23475d4ac59bf375a2c96b6d824f06ba2d1873ee8d4495fb9bb412f848379ac7d11fb</hash>
    <hash alg="SHA3-256">84ba8a8001b44c04048d5ab1208640f0f5d54183231962cb180de7d7cfa7a386</hash>
    <hash alg="SHA3-384">2101589e294edd8d55f29be193e74c84afb8db754566b08b369c6c235e969af25ee8962e70f7c3dc262f7cbdebe3d57d</hash>
    <hash alg="SHA3-512">cc8b929d7f54aa98c606977bab56fcfe198e0bb21403b7b2a1bff6bf2940901aee9d47bde493a98af376b4b1e75d110284023bd174b7774d454fac6e14b0606c</hash>
  </hashes>
  <licenses>
    <license>
      <id>Apache-2.0</id>
    </license>
  </licenses>
  <purl>pkg:maven/org.apache.poi/poi@3.17?type=jar</purl>
  <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

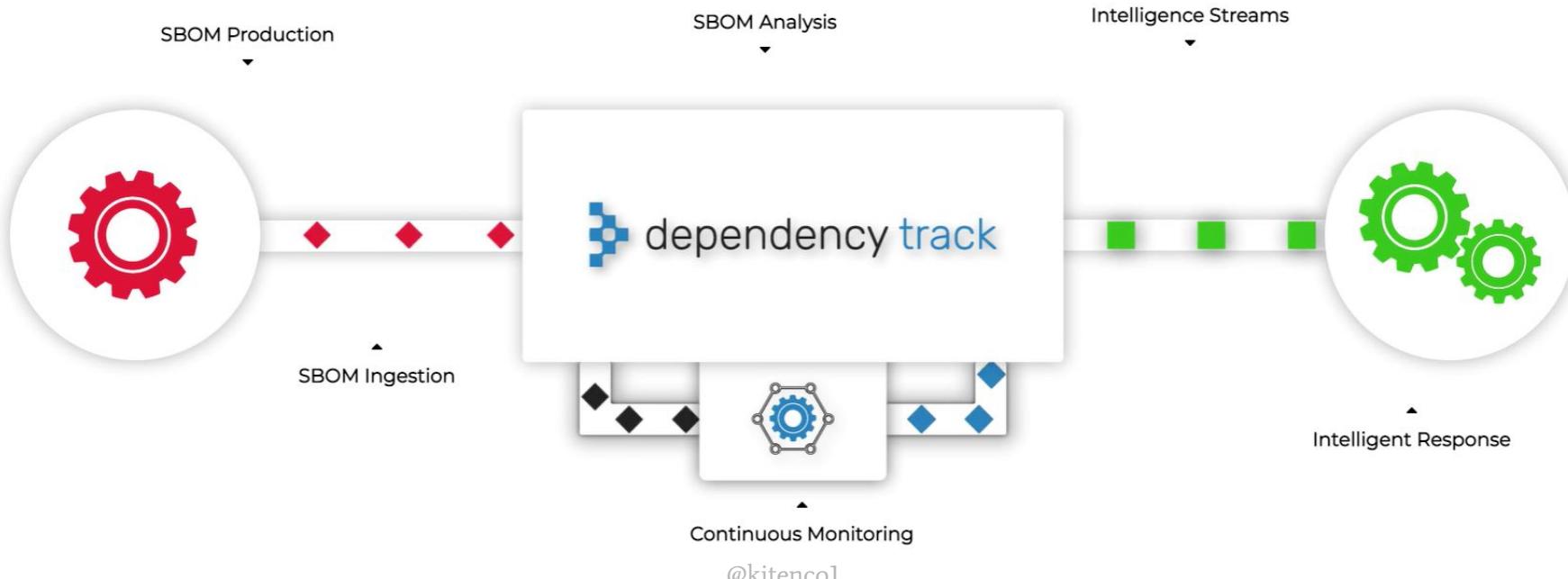


# OWASP Dependency Track



# Continuous Security Monitoring

## Operationalize Software Bill of Materials





Dashboard

PORTFOLIO

Projects

Components

Vulnerabilities

Licenses

ADMINISTRATION

Policy Management

Administration

Home / Projects

2012  
Portfolio Vulnerabilities74  
Projects at Risk425  
Vulnerable Components10679  
Inherited Risk Score[+ 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><div style="width: 33%;">3</div><div style="width: 33%;">3</div></div>
wms	0.1	18 Mar 2022 at 03:44:06	CycloneDX 1.3	13	<input checked="" type="checkbox"/>	<div><div style="width: 20%;">2</div><div style="width: 10%;">1</div></div>
rbus	0.6.0	18 Jan 2022 at 03:15:27	CycloneDX 1.3	69	<input checked="" type="checkbox"/>	<div><div style="width: 10%;">1</div><div style="width: 70%;">7</div><div style="width: 20%;">8</div></div>
rbus	0.7.0	18 Mar 2022 at 03:16:00	CycloneDX 1.3	24	<input checked="" type="checkbox"/>	<div><div style="width: 30%;">3</div><div style="width: 40%;">3</div></div>
zus-portal	1.0.0	18 Mar 2022 at 03:05:15	CycloneDX 1.3	3	<input checked="" type="checkbox"/>	<div><div style="width: 100%;">1</div></div>
ema-portal	1.0.0	18 Mar 2022 at 03:02:17	CycloneDX 1.3	0	<input checked="" type="checkbox"/>	<div><div style="width: 100%;">0</div></div>
para83a-online	1.0.0	18 Mar 2022 at 03:12:04	CycloneDX 1.3	50	<input checked="" type="checkbox"/>	<div><div style="width: 10%;">1</div><div style="width: 50%;">5</div><div style="width: 40%;">5</div></div>
mgs-agb-online	1.0.0	18 Mar 2022 at 03:24:55	CycloneDX 1.3	0	<input checked="" type="checkbox"/>	<div><div style="width: 100%;">0</div></div>
para80-online	1.0.0	18 Mar 2022 at 01:07:06	CycloneDX 1.3	50	<input checked="" type="checkbox"/>	<div><div style="width: 10%;">1</div><div style="width: 50%;">5</div><div style="width: 40%;">5</div></div>
imorsaftplogimporter	1.0.0	18 Mar 2022 at 02:19:28	CycloneDX 1.3	3	<input checked="" type="checkbox"/>	<div><div style="width: 100%;">1</div></div>

Showing 1 to 10 of 88 rows 10 ▲ rows per page

&lt; 1 2 3 4 5 ... 9 &gt;

# Suche nach Komponenten

Components

Vulnerabilities

Licenses

ADMINISTRATION

Policy Management

Administration

Portfolio Vulnerabilities

Projects at Risk

Vulnerable Components

Inherited Risk Score

Coordinates ✓

Group log4j

Version

Search

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			bizservice 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			bizservice 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



CVE-2021-44228



Name	Published	CWE	Projects	Severity
NVD CVE-2021-44228	10 Dec 2021	CWE-502 Deserialization of Untrusted Data	0	<span>Critical</span>

Showing 1 to 1 of 1 rows

CVE-2019-12415



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

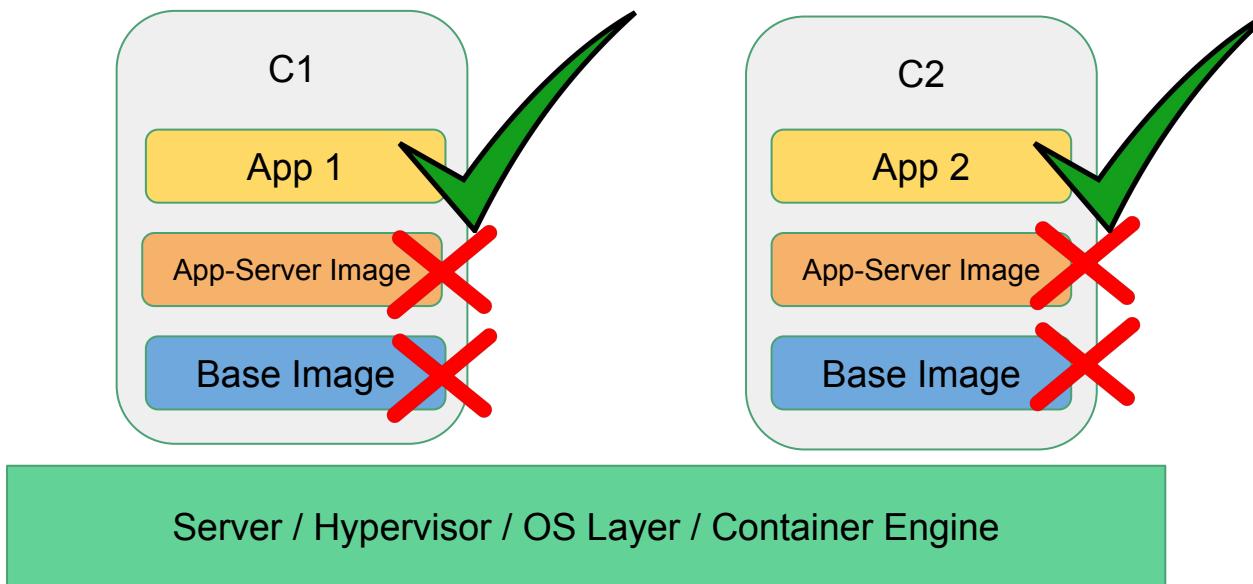
Showing 1 to 1 of 1 rows

# 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



# SBOMs für Container Images und Filesystem



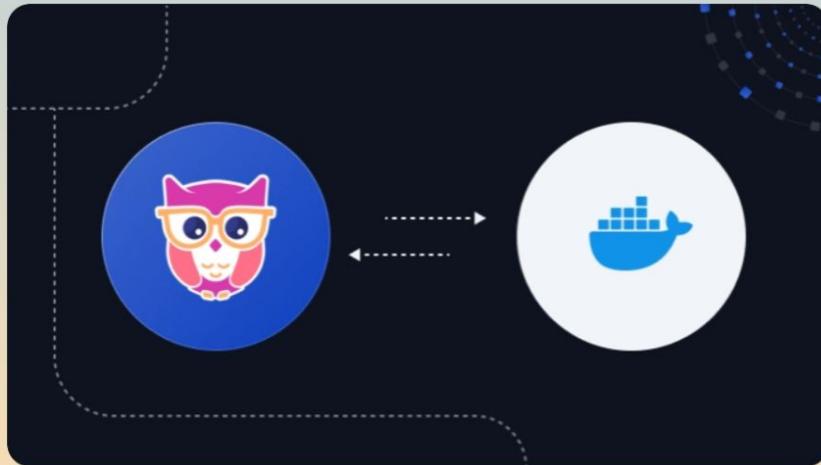
<https://github.com/anchore/syft>



<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



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.

# CLI Aufruf und direkte Auswertung

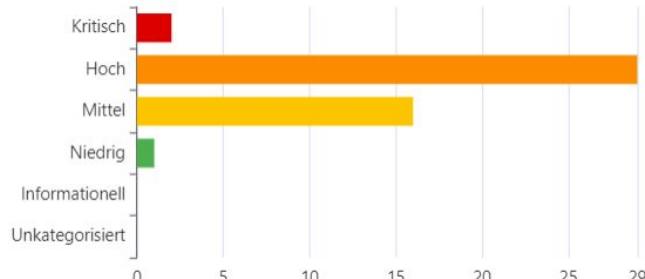
```
stage('Analyze Dependencies of Docker Image'){
    steps{
        script {
            dir ('out') {
                sh "syft -o cyclonedx-xml=sbom.xml ${env.IMAGE}:${env.TAG}"
                withCredentials([string(credentialsId: 'deptrack-apikey-text', variable: 'API_KEY')]) {
                    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 ...

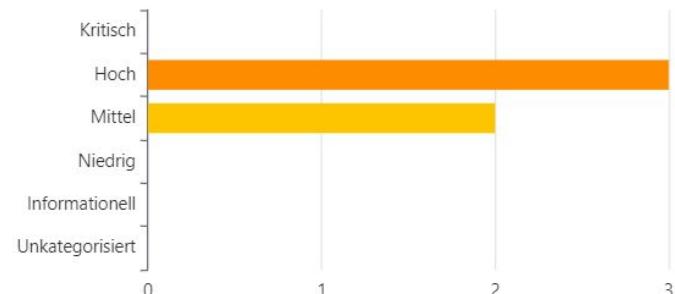
Project Name	Version	Classifier	Last BOM Import	BOM Format	Risk Score	Active	Policy Violations	Vulnerabilities
mgs-keycloak	19.0.1	Library	14 Sep 2022 at 01:33:29	CycloneDX 1.4	21	<input checked="" type="checkbox"/>	<div style="width: 11px;"></div>	<div style="width: 3px; background-color: orange;"><span>3</span></div> <div style="width: 2px; background-color: yellow;"><span>2</span></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: 3px;"></div>	<div style="width: 2px; background-color: orange;"><span>2</span></div> <div style="width: 33px; background-color: yellow;"><span>33</span></div> <div style="width: 20px; background-color: green;"><span>20</span></div>



Zusammenfassung der Abhängigkeitsanalyse



Zusammenfassung der Abhängigkeitsanalyse





etw

-keycloak ▾ 19.0.1-19



View Details

&gt;

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	<div><div style="width: 16%;">1</div><div style="width: 84%;">16</div></div> 2
libarchive	3.3.3-3.el8_5				40	<div><div style="width: 5%;">5</div><div style="width: 95%;">5</div></div>
freetype	2.9.1-4.el8_3.1				23	<div><div style="width: 1%;">1</div><div style="width: 99%;">2</div></div> 1
keycloak-services	19.0.1	⚠ org.keycloak			15	<div><div style="width: 1%;">1</div><div style="width: 9%;">3</div><div style="width: 88%;">1</div></div>
pip	9.0.3	⚠	MIT		13	<div><div style="width: 2%;">2</div><div style="width: 98%;">1</div></div>
keycloak-core	19.0.1	⚠ org.keycloak			13	<div><div style="width: 2%;">2</div><div style="width: 98%;">1</div></div>
libjpeg-turbo	1.5.3-12.el8				11	<div><div style="width: 1%;">1</div><div style="width: 99%;">2</div></div>
rhel	8.6				8	<div><div style="width: 1%;">1</div><div style="width: 99%;">1</div></div>
snakeyaml	1.30	⚠ org.yaml			8	<div><div style="width: 1%;">1</div><div style="width: 99%;">1</div></div>
okhttp	3.14.9	⚠ com.squareup.okhttp3			5	<div><div style="width: 1%;">1</div><div style="width: 99%;">1</div></div>

Showing 1 to 10 of 1208 rows

10 ▾ rows per page

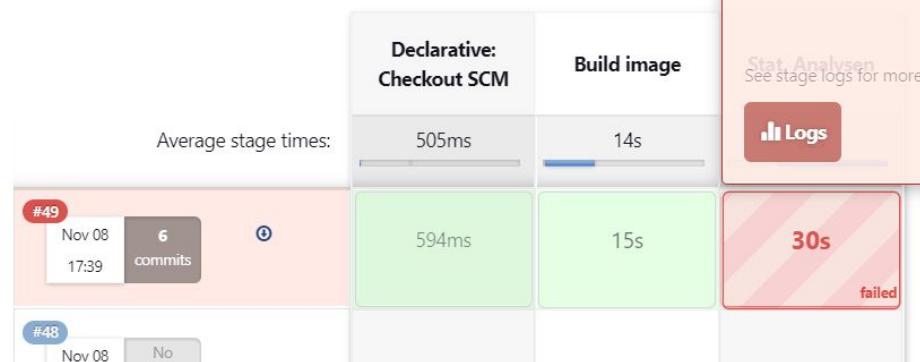
&lt; 1 2 3 4 5 ... 121 &gt;

@kitencol

# Was jetzt?

# Organisatorisch

## Stage View



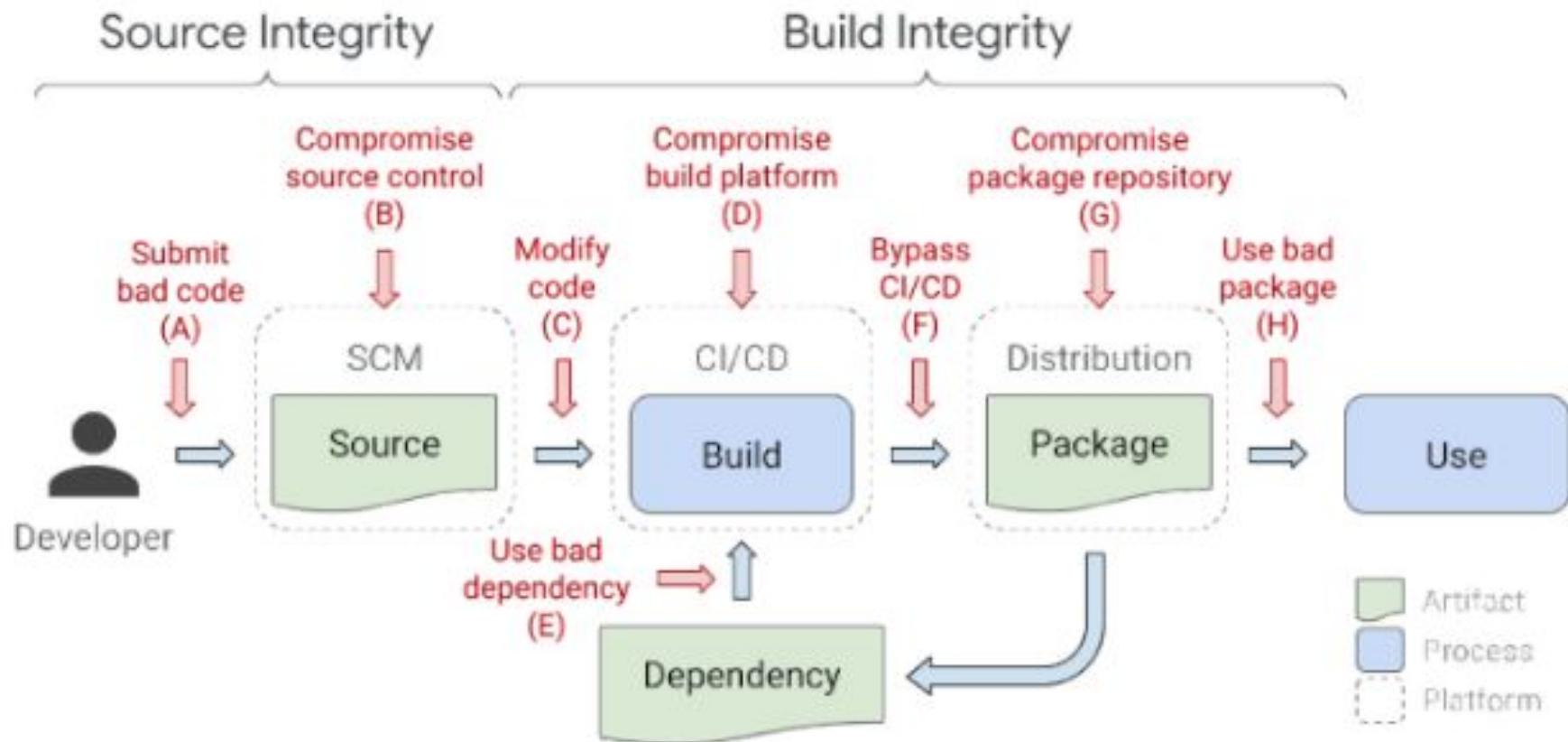
CVE-2022-12345

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>

# Trusted Commits

Multi-Faktor Authentifizierung

IP Allowlist

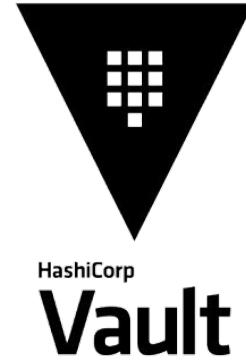
Github Branch Protection Rules



# Secrets

Keine Secrets im Repo (z.B.  
Thoughtworks Talisman)

Keine Secrets im Build Tool  
(z.B. Hashicorp Vault)



# Automatische Updates für Dependencies

- Dependabot, Renovate und Ähnliches



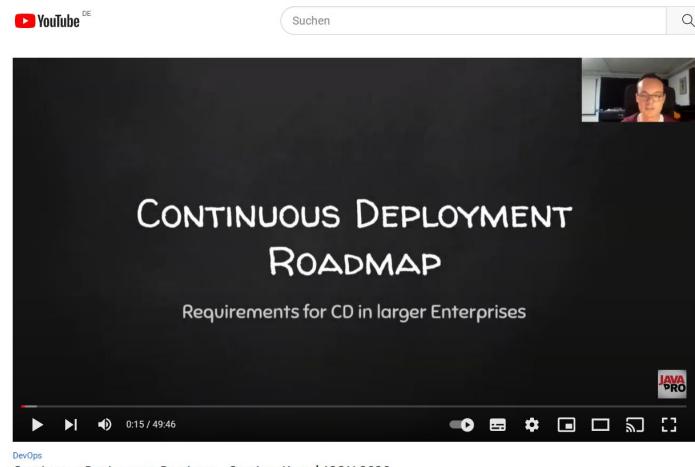
<https://github.com/dependabot>



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

# Fähigkeit, schnell auszuliefern

- Continuous Delivery & Deployment

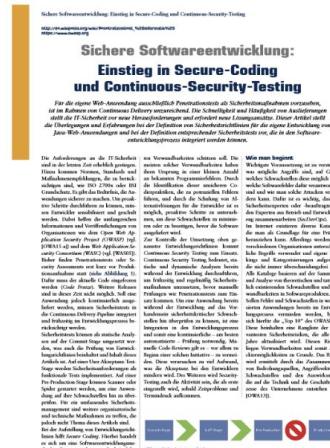


<https://entwickler.de/devops/roadmap-einer-spannenden-reise>

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

# Secure Coding & Testing

- Sichere Softwareentwicklung & Security Testing (SAST, DAST)



Frocon 2016  
Stephan Kaps: Sichere Softwareentwicklung

# Container Security

- Angriffsfläche verkleinern
- Privilegien minimieren
- Docker Engine härten

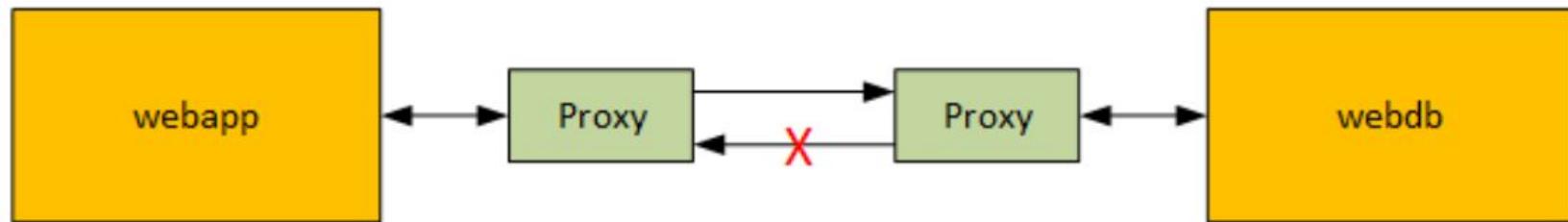


In den sicheren Hafen: Einstieg in Container Security - Stephan Kaps | JCON 2020

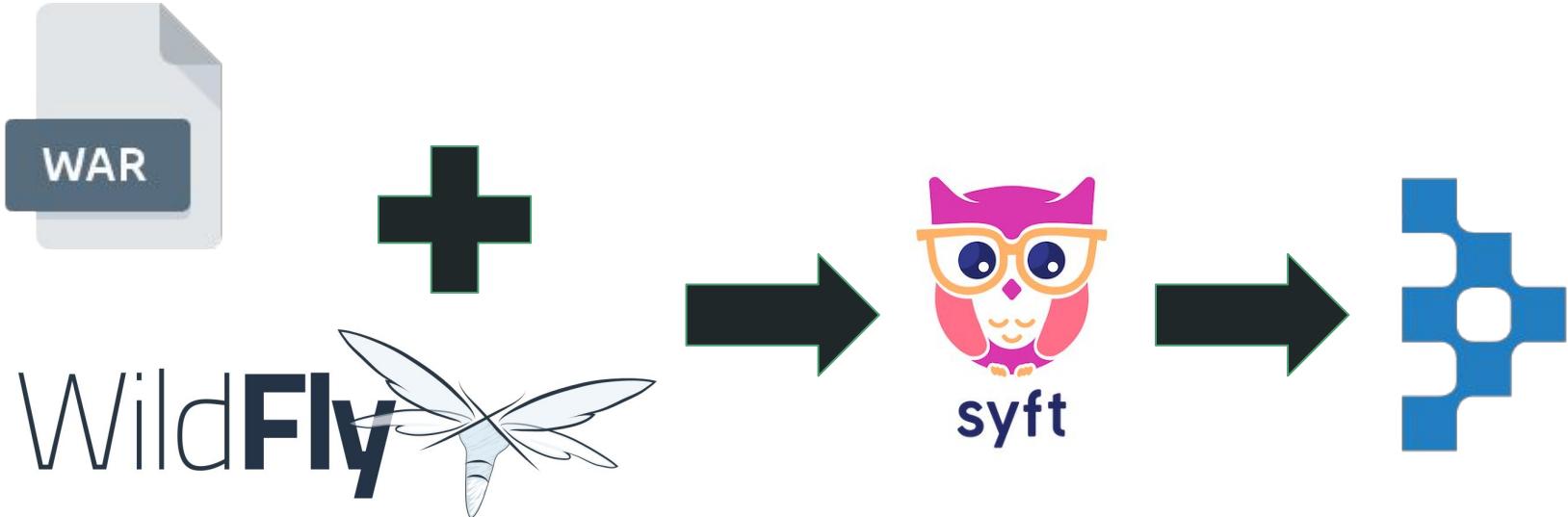


# Isolierung

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



# Was war jetzt mit Kaufsoftware?



Aufnahme von Anforderung in Ausschreibungen

***“Bereitstellung einer SBOM bei jeder ausgelieferten Version”***

***“Zeitnahe Bereitstellung von Patches bei neuen Sicherheitslücken”***

## Weitere Maßnahmen:

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

... oder dass jemand uns etwas schadhaftes unterschiebt?

# 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>

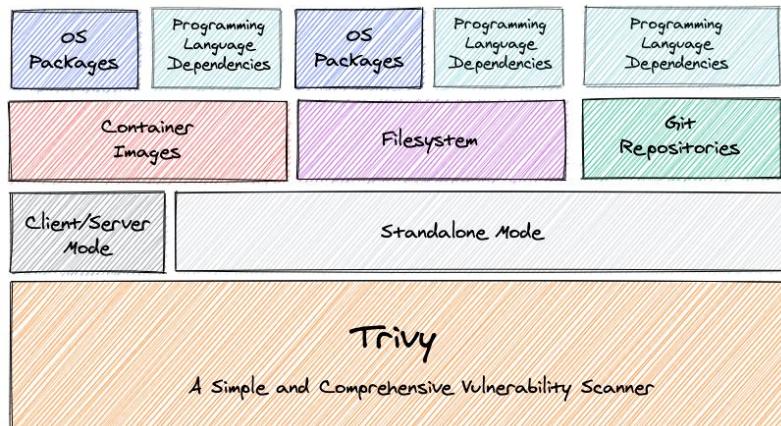
# Secure Container Registry

- Vulnerability Scans
- Image Signing
- Pull Policies



# HARBOR

<https://goharbor.io/>



Targets

Artifacts

Modes



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

# 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	libc-bin	2.28-10	

# 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}"
                }
            }
        }
    }
}
```

# Image Signing mit Sigstore cosign

Artefakte										
<input type="button" value="SCAN"/>		<input type="button" value="SCAN STOPPEN"/>		AKTIONEN		ACTIONS				
	Artefakte	Pull Befehl	Tags	Signiert mit Cosign	Größe	Schwachstellen	Annotationen	Label	Push Zeit	Pull Zeit
<input type="checkbox"/>	sha256:add50289	<input type="button" value="Pull"/>	latest	<span>✗</span>	86.14MiB	Keine Schwachstelle			04.11.22, 02:31	07.11.22, 01:00
<input type="checkbox"/>	sha256:e7d34034	<input type="button" value="Pull"/>		<span>✓</span>	86.14MiB	Keine Schwachstelle			27.10.22, 14:39	07.11.22, 01:01
<input type="checkbox"/>	sha256:Oedb0aea	<input type="button" value="Pull"/>		<span>✓</span>	88.11MiB	<span>H</span> 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 with the 'Konfiguration' tab selected in the top navigation bar. The left sidebar contains links for 'Projekte', 'Logs', 'Administration' (with sub-links for 'Nutzer', 'Robot-Zugänge', 'Gruppen', 'Registries', 'Replikationen', 'P2P-Verteilung', 'Label', 'Projekt-Begrenzungen', 'Schwachstellen Scan', 'Clean Up', and 'Konfiguration'). The main content area displays various configuration settings:

- Projekt Registry**: A checkbox labeled 'Öffentlich' (Public) is unchecked. A note below states: "Ein Projekt öffentlich einzustellen, macht die Repositories für alle zugreifbar." (Setting a project public makes its repositories accessible to all).
- Deployment Sicherheit**: A checkbox labeled 'Cosign' is checked. A note below states: "Erlaube ausschließlich verifizierte Images." (Allow only verified images).
- Scannen auf Schwachstellen**: A checkbox labeled 'Verhindere den Download von Images mit Schwachstellen' (Prevent download of images with vulnerabilities) is checked. A note below states: "Verhindere den Download von Images mit Schwachstellen des Schweregrads **Hoch** und darüber." (Prevent download of images with vulnerabilities of severity **Hoch** and above).
- CVE Allowliste**: A note states: "Die Projekt-Allowliste erlaubt es, Schwachstellen in der Liste beim pushen und pullen von Images zu ignorieren." (The project allowlist allows ignoring vulnerabilities in the list during pushing and pulling of images). It also notes: "Es kann entweder die Standard Allowliste des Systems verwendet werden. Alternativ kann über 'Projekt-Allowliste' eine neue Allowliste erstellt werden." (Either the system's standard allowlist can be used. Alternatively, a new allowlist can be created via 'Project-Allowlist'). A note below states: "Individuelle CVE IDs hinzufügen, bevor über 'VON SYSTEM KOPIEREN' die System-Allowliste ebenfalls hinzugefügt wird." (Add individual CVE IDs before copying from the system to add them to the system allowlist). Below this, there are two radio buttons: 'System-Allowliste' (selected) and 'Projekt-Allowliste'. A 'HINZUFÜGEN' (ADD) button is located below the radio buttons.
- VON SYSTEM KOPIEREN**: A note above this section says 'Läuft ab am' (Expires at). A note below says 'Läuft niemals ab' (Never expires). A checkbox labeled 'Läuft niemals ab' is checked.
- CVF-2021-43816**: This is listed under 'VON SYSTEM KOPIEREN'.

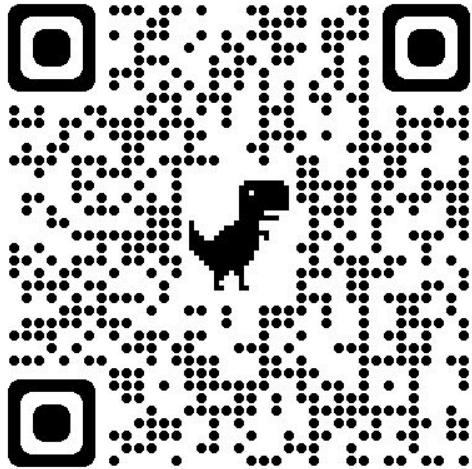
# Fazit

---

# Was ist zu tun?

- 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



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

## WARUM?

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

## WAS BEDEUTET DAS?

Es war erstmals das erste mal, dass eine Schwachstelle in einer Open-Source Bibliothek für Überlizenzen in IT-Anwendungen genutzt hat. Dies im Dezember 2021 war die CVE-2021-44228 zu Log4j bzw. Log4Shell sogar in der Titelseite des Nachrichtenmagazins Der Spiegel.

Doch was haben wir daraus gelernt?

Haben wir Erkenntnisse gewonnen oder Methoden ergriffen, um bei nächsten Zero-Day-Vulnerability öffentlich bekannter Software schneller und sicherer heranzuführen, als es zuvor möglich war?

Hätten wir aktuelle Beipackzettel zu jeder Software, eingesetzte SBOMs, wären wir dazu in der Lage,

### Was ist eine SBOM?

SBOM steht für Software Bill of Materials und ist eine Inventur an allen einzelnen Bauteilen, die sich in einer Software befinden. Sie enthält eine Auflistung aller verwendeten Bibliotheken bzw. Komponenten, sowie deren direkt und indirekte Abhängigkeiten und Hersteller-Daten. Darauf hinzu können viele weitere Informationen enthalten sein wie z.B. Lizenzinformationen.

Es unterscheidet sich vom DFIRM (Standard Software Data Exchange), da es ab SSBOM 1.500 (2021) gewisse im Open-Source-Code wie Credenciales von OpenSSL oder Soft von Apache Maven (Maven) auch priorisiert.

### Log4Shell oder SolarWinds Was haben wir daraus gelernt?



#### PRODUKT-INHALT IST UNBEKANNT

Aktuell befinden wir Software, von der wir nicht wissen, wie sie funktioniert und welche Sicherheitsmaßnahmen sie einsetzt. Daher ist auch unklar, ob diese Produkte unsicher sind und ob sie tatsächlich betrügt werden.



#### RECHERCHEN SIND UNFÄWIG

Werden schwierige Sicherheitslücken wie Log4Shell entdeckt, kann es schwer sein, die entsprechenden Softwarehersteller zu kontaktieren, um nach vorliegenden Anweisungen der Hersteller zu schließen, ob sie tatsächlich ausgenutzt werden.



#### WIR SIND EVENTUELLE ANGREIFBAR

Untersucht wird, ob schwierige, ob wir tatsächlich ungewollt und unbewusst in unsere Systeme integriert haben. Wenn ja, müssen wir die Anweisungen der Hersteller befolgen, um unser System abgeschützen.

### Was können wir selbst tun?



#### Regelmäßige Analyse auf Sicherheitsthemen

Für spezielle Sicherheitsthemen, wie z.B. wenn eine SBOM zur Verfügung steht, ist die Analyse der technischen Details, in welchen Umständen die Software verwendet wird, und welche Bibliotheken benutzt werden, leicht möglich. Durch gezielte Tools ist auch ein Überblick über das gesamte Bewertungsportfolio aufwandsarm umsetzbar.

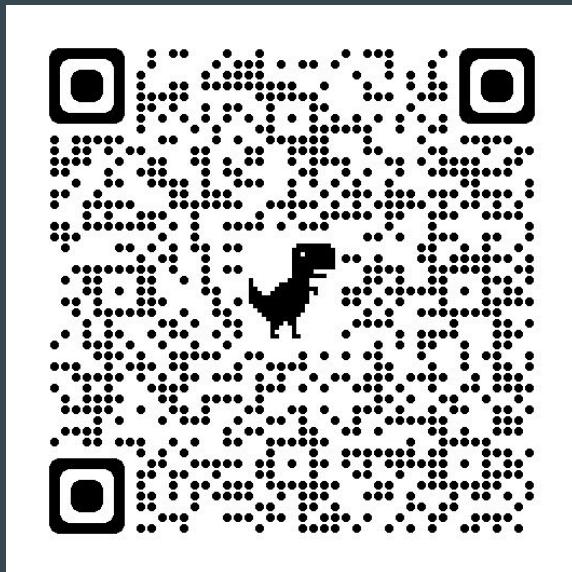


#### Aufnahme der Verpflichtung in Auslieferungsbedingungen

In regelmäßigen Basis-Checkups sollte aufgeklärt werden, dass Hersteller von Software verpflichtet sind, jede übergeordnete Version auch eine SBOM bereitzustellen. Zusätzlich sollten sie in Aufgabenstellungen werden, die eine zielgerichtete Behandlung von Sicherheitslücken fordern.

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

# Cloud- / DevOps Engineer (m/w/d)



bis 75.000 €  
Standort: Bonn  
bis zu 100% remote

# New Work



@kitencol

# Vielen Dank!

Kontakt:

Stephan Kaps  
[info@kitenco.de](mailto:info@kitenco.de)

[www.kitenco.de](http://www.kitenco.de)



@kitencol



[https://www.xing.com/profile/Stephan\\_Kaps2](https://www.xing.com/profile/Stephan_Kaps2)



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



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

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

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

@kitencol

