

Software-Supply-Chain Security in Practice

Sichere Software - vom Commit bis zum User

Stephan Kaps

Leiter Softwareentwicklung Bundesamt für Soziale Sicherung

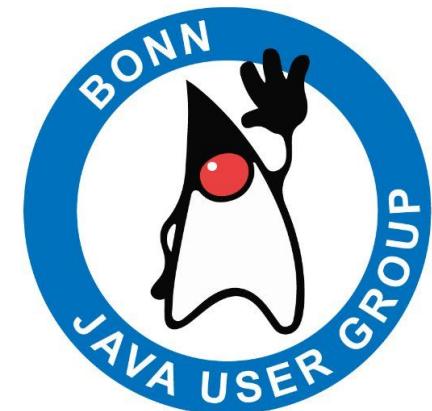
Gründer Java User Group Bonn

Java Entwickler & Architekt seit 2002

Weitere Schwerpunkte:

- Softwareentwicklungsprozesse
- BizDevSecOps
- OpenSource
- Speaker & Autor

bizdevops.org



Wer sind wir uns was machen wir?

(Aufsichts-) Behörde mit ca. 700 Mitarbeiter

ca. 60 Fachanwendungen (Produkte)

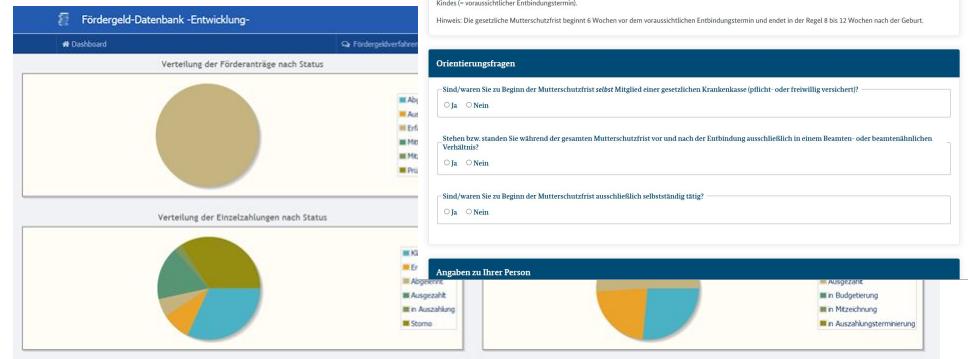
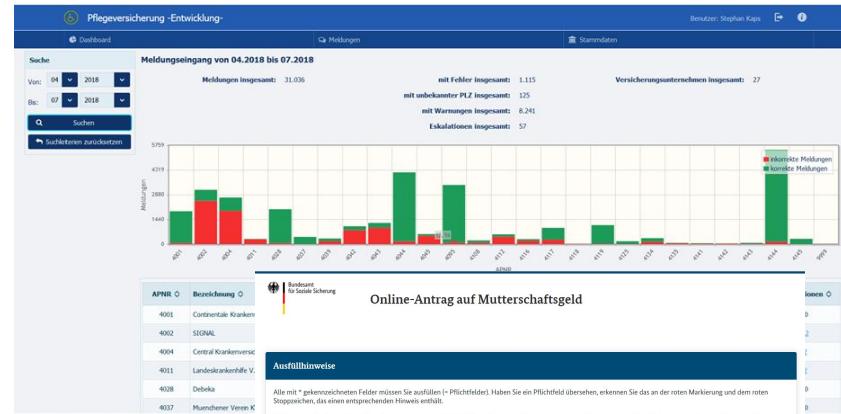
für interne Fachbereiche & Landesprüfdienste

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

Themes for this volume

Download

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 what's going on, but we can only speculate: Is this the result of churn in the JavaScript ecosystem, or is it to promote?



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

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

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

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

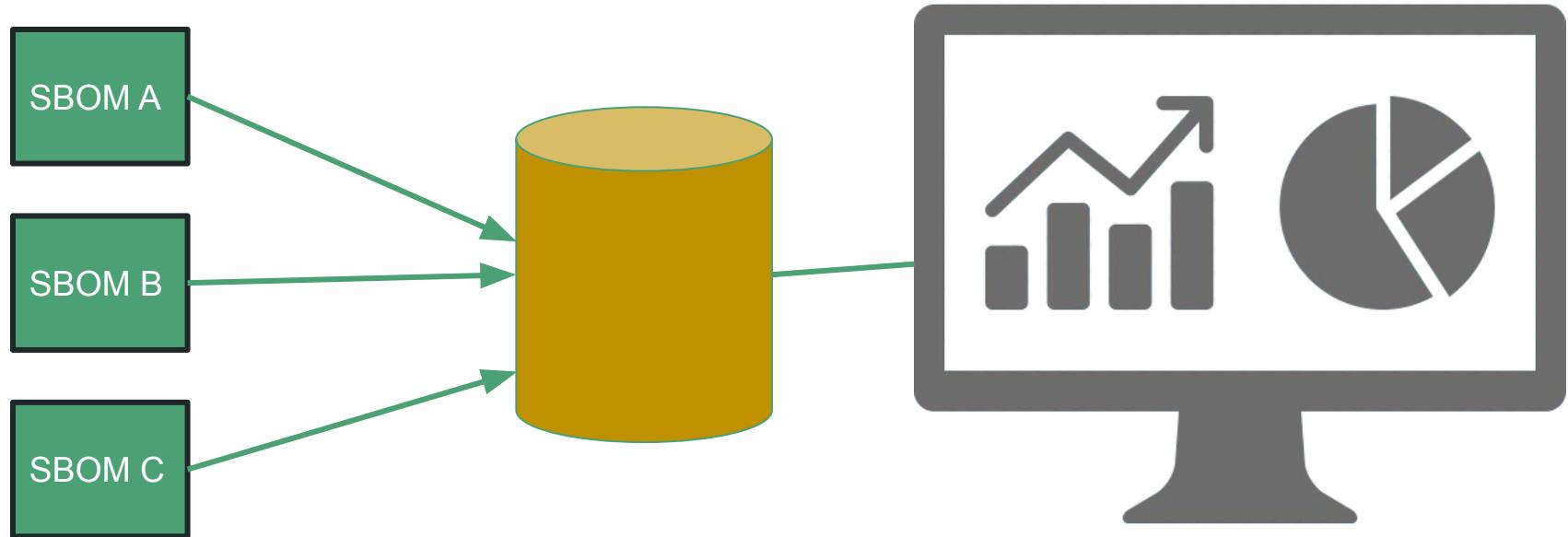
opensource analysis



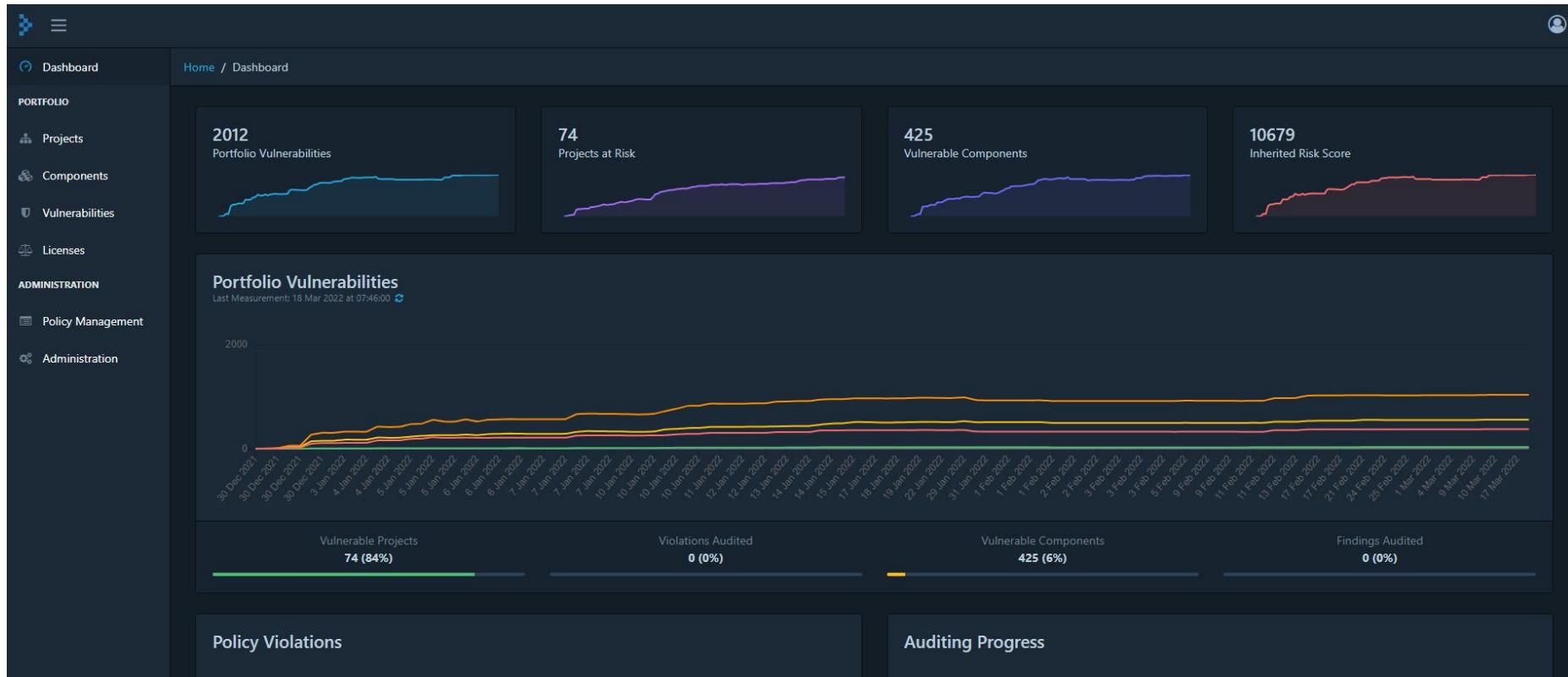
```
<component type="library" bom-ref="pkg:maven/org.apache.poi/poi@3.17?type=jar">
  <publisher>Apache Software Foundation</publisher>
  <group>org.apache.poi</group>
  <name>poi</name>
  <version>3.17</version>
  <description>Apache POI - Java API To Access Microsoft Format Files</description>
  <scope>optional</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

< 1 2 3 4 5 ... 9 >

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	Critical

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	Medium

Showing 1 to 1 of 1 rows

Projekt Komponenten

View Details ▾ 1.3.0

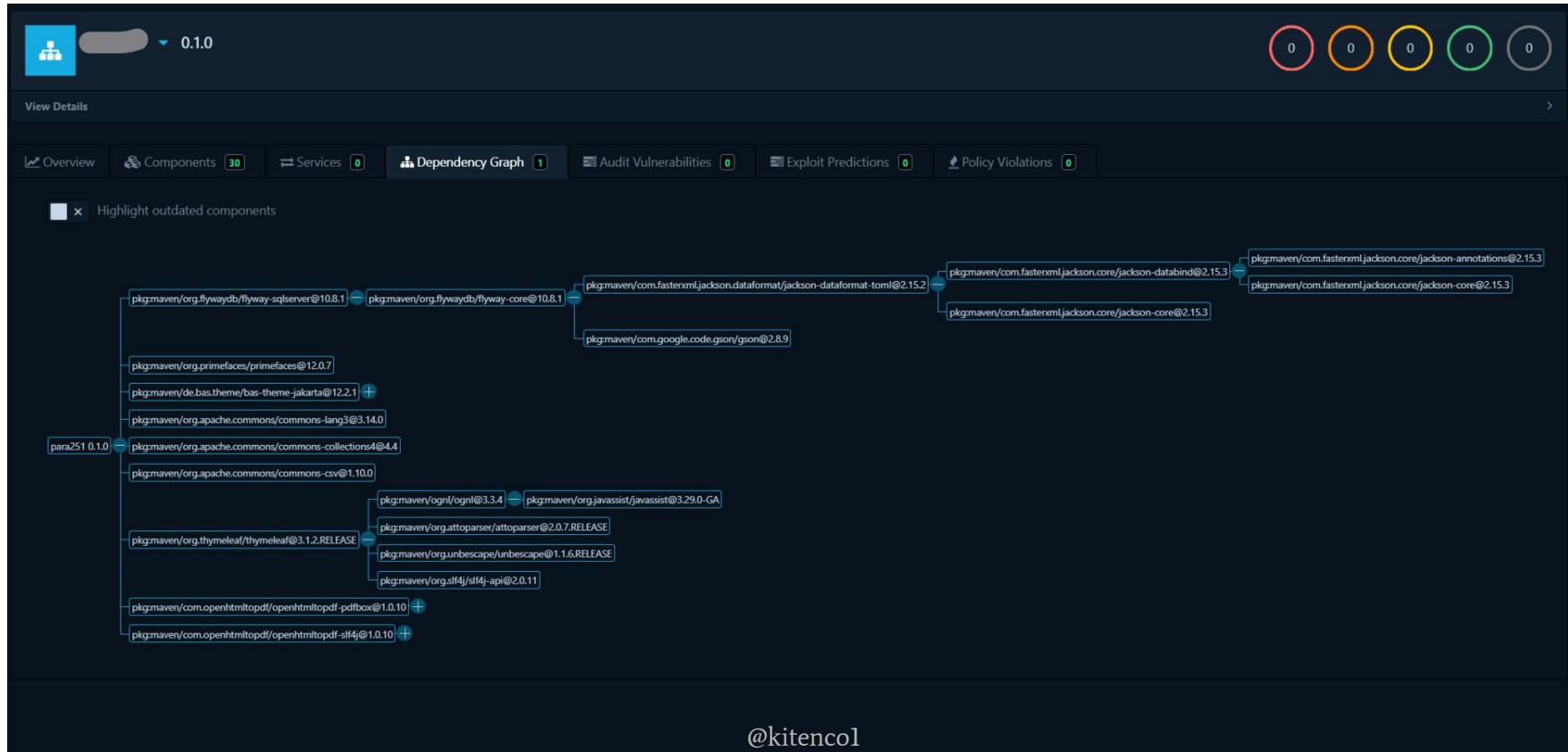
1 6 4 0 1

Overview Components 113 Services 0 Dependency Graph 1 Audit Vulnerabilities 12 Exploit Predictions 12 Policy Violations 0

+ Add Component - Remove Component Upload BOM Download BOM Search

Component	Version	Group	Internal	License	Risk Score	Vulnerabilities
sshd-common	2.7.0	org.apache.sshd		Apache-2.0	16	<div style="width: 100%;">1 2</div>
jose4j	0.7.11	org.bitbucket.b_c		Apache-2.0	10	<div style="width: 100%;">1 1</div>
bcprov-jdk15on	1.70	org.bouncycastle		Bouncy Castle Licence	6	<div style="width: 100%;">2</div>
wildfly-elytron-x500	1.19.1.Final	org.wildfly.security		Apache-2.0	5	<div style="width: 100%;">1</div>
wildfly-elytron	1.19.1.Final	org.wildfly.security		Apache-2.0	5	<div style="width: 100%;">1</div>
wildfly-elytron-password-impl	1.19.1.Final	org.wildfly.security		Apache-2.0	5	<div style="width: 100%;">1</div>
wildfly-elytron-credential	1.19.1.Final	org.wildfly.security		Apache-2.0	5	<div style="width: 100%;">1</div>
wildfly-elytron-realm-ldap	1.19.1.Final	org.wildfly.security		Apache-2.0	5	<div style="width: 100%;">1</div>
wildfly-elytron-realm-token	1.19.1.Final	org.wildfly.security		Apache-2.0	0	<div style="width: 100%;">0</div>

Dependency Graph



Audit Vulnerabilities

View Details >

1.3.0

1 6 4 0 1

Overview Components 113 Services 0 Dependency Graph 1 Audit Vulnerabilities 12 Exploit Predictions 12 Policy Violations 0

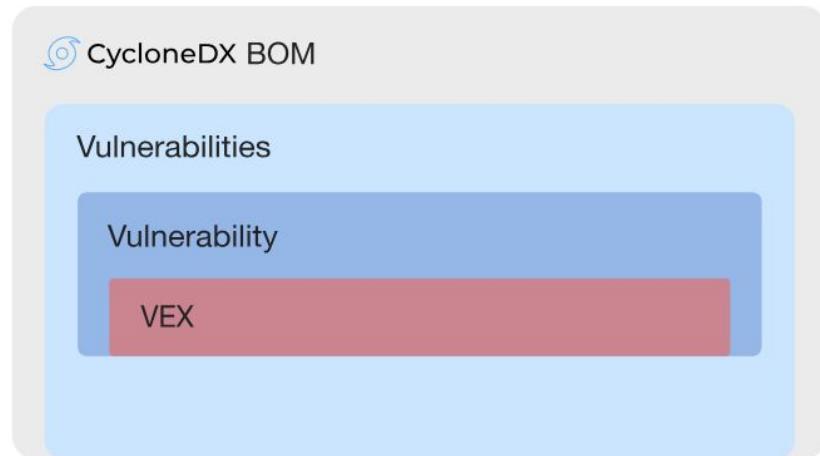
Apply VEX Export VEX Export VDR Reanalyze Show suppressed findings Search

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



	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

[Add Comment](#)

Analysis

Not Affected

Suppress

Justification

Vendor Response (project)

Protected at runtime

Will not fix

Details

mal sehen wo die Details auftauchen
@kitencol

```
],
  "vulnerabilities" : [
    {
      "bom-ref" : "91745472-775f-4d24-a377-24c252a6e35a",
      "id" : "CVE-2020-14359",
      "source" : {
        "name" : "NVD",
        "url" : "https://nvd.nist.gov/"
      },
      "ratings" : [
        {
          "source" : {
            "name" : "NVD",
            "url" : "https://nvd.nist.gov/"
          },
          "score" : 7.5,
          "severity" : "high",
          "method" : "CVSSv2",
          "vector" : "(AV:N/AC:L/Au:N/C:P/I:P/A:P)"
        },
        {
          "source" : {
            "name" : "NVD",
            "url" : "https://nvd.nist.gov/"
          },
          "score" : 7.3,
          "severity" : "high",
          "method" : "CVSSv3",
          "vector" : "CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/"
        }
      ],
      "cwes" : [
        305
      ],
      "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 webservers (e.g. Jetty).. This means there is no protection when we put a Gatekeeper in front of a Jetty server and use lowercase headers.",
      "published" : "2021-02-23T13:15:00Z",
      "updated" : "2022-08-10T20:28:00Z",
      "analysis" : {
        "state" : "not_affected",
        "justification" : "protected_at_runtime",
        "response" : [
          "will_not_fix"
        ],
        "detail" : "mal sehen wo die Details auftauchen"
      },
      "affects" : [
        {
          "ref" : "1ee9bd3a-f548-4744-a7fd-639725e7d8ab"
        }
      ]
    },
    {
      "bom-ref" : "f5d26c6e-859d-4259-abc3-2c2c59b0f3ed",
      "id" : "CVE-2022-42003",
      "source" : {
        "name" : "NVD",
        "@kitenco1"
        "url" : "https://nvd.nist.gov/"
      }
    }
  ]
}
```



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



Dashboard

▼ 1.4.0

7 21 11 2 0

View Details >

Overview

Components 91

Services 0

Dependency Graph 1

Audit Vulnerabilities 41

Exploit Predictions 41

Policy Violations 0

 Show suppressed findings

Search

⟳ ⚙️

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	

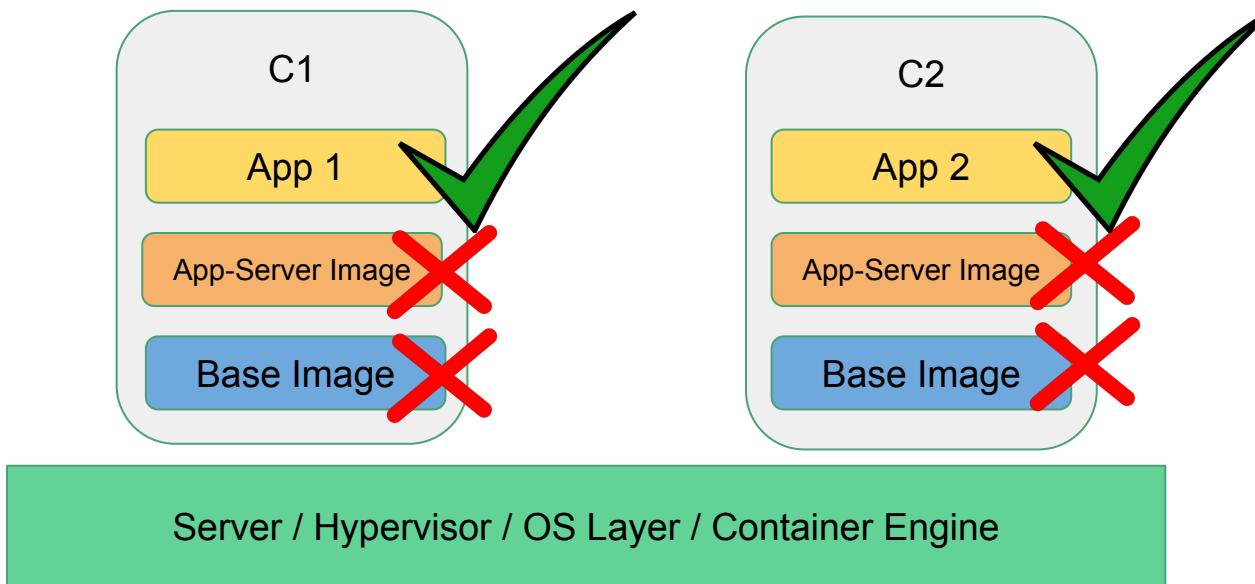
Integration in Build-Prozess (Nightly)

```
● ○ ●

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



SBOMs für Container Images und Filesystem



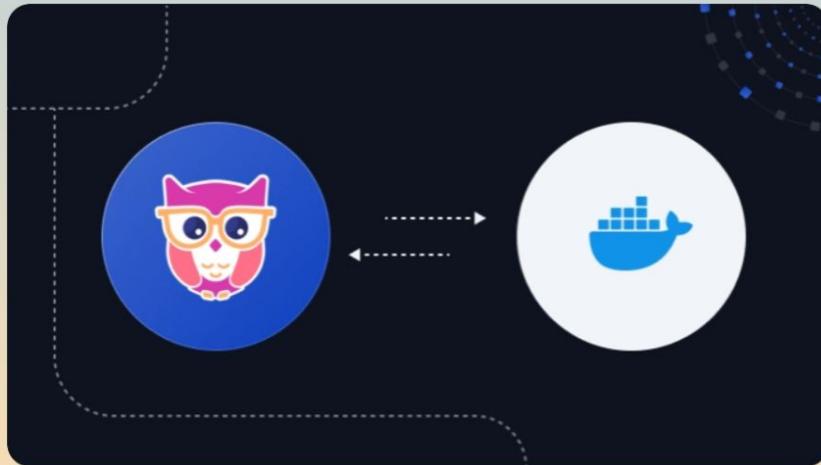
<https://github.com/anchore/syft>



<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

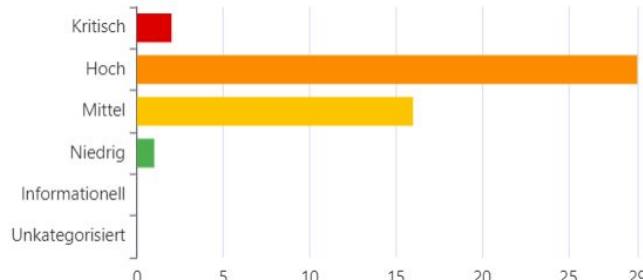
```
stage('Analyze Dependencies of Docker Image'){
    steps{
        script {
            dir ('out') {
                sh "syft -o cyclonedx-xml=sbom.xml ${env.IMAGE}:${env.TAG}"
                withCredentials([string(credentialsId: 'deptrack-apikey-text', variable: 'API_KEY')]) {
               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

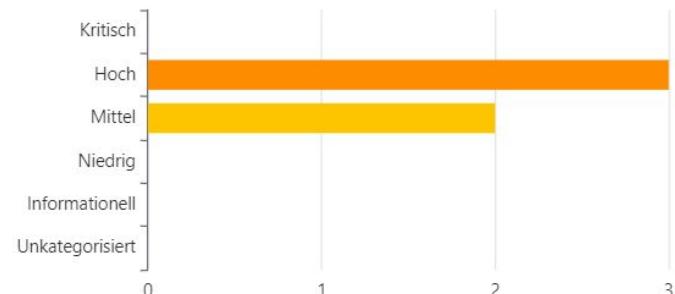
Project Name	Version	Classifier	Last BOM Import	BOM Format	Risk Score	Active	Policy Violations	Vulnerabilities
mgs-keycloak	19.0.1	Library	14 Sep 2022 at 01:33:29	CycloneDX 1.4	21	<input checked="" type="checkbox"/>	<div style="width: 11px;"></div>	<div style="width: 3px; background-color: orange;">3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



Zusammenfassung der Abhängigkeitsanalyse



Zusammenfassung der Abhängigkeitsanalyse





etw

-keycloak ▾ 19.0.1-19



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	<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

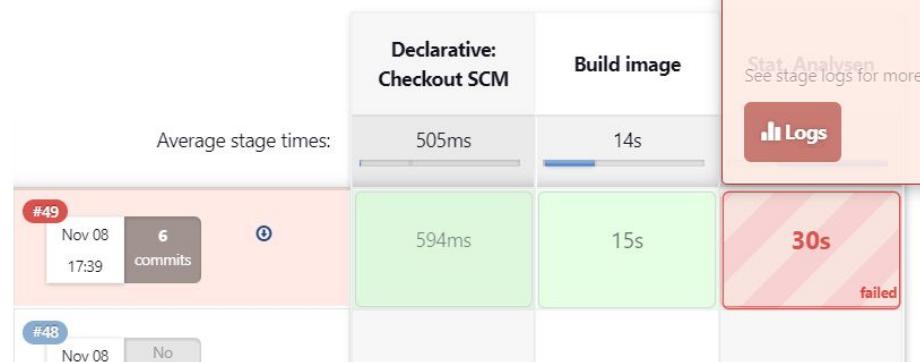
< 1 2 3 4 5 ... 121 >

@kitencol

Was jetzt?

Organisatorisch

Stage View

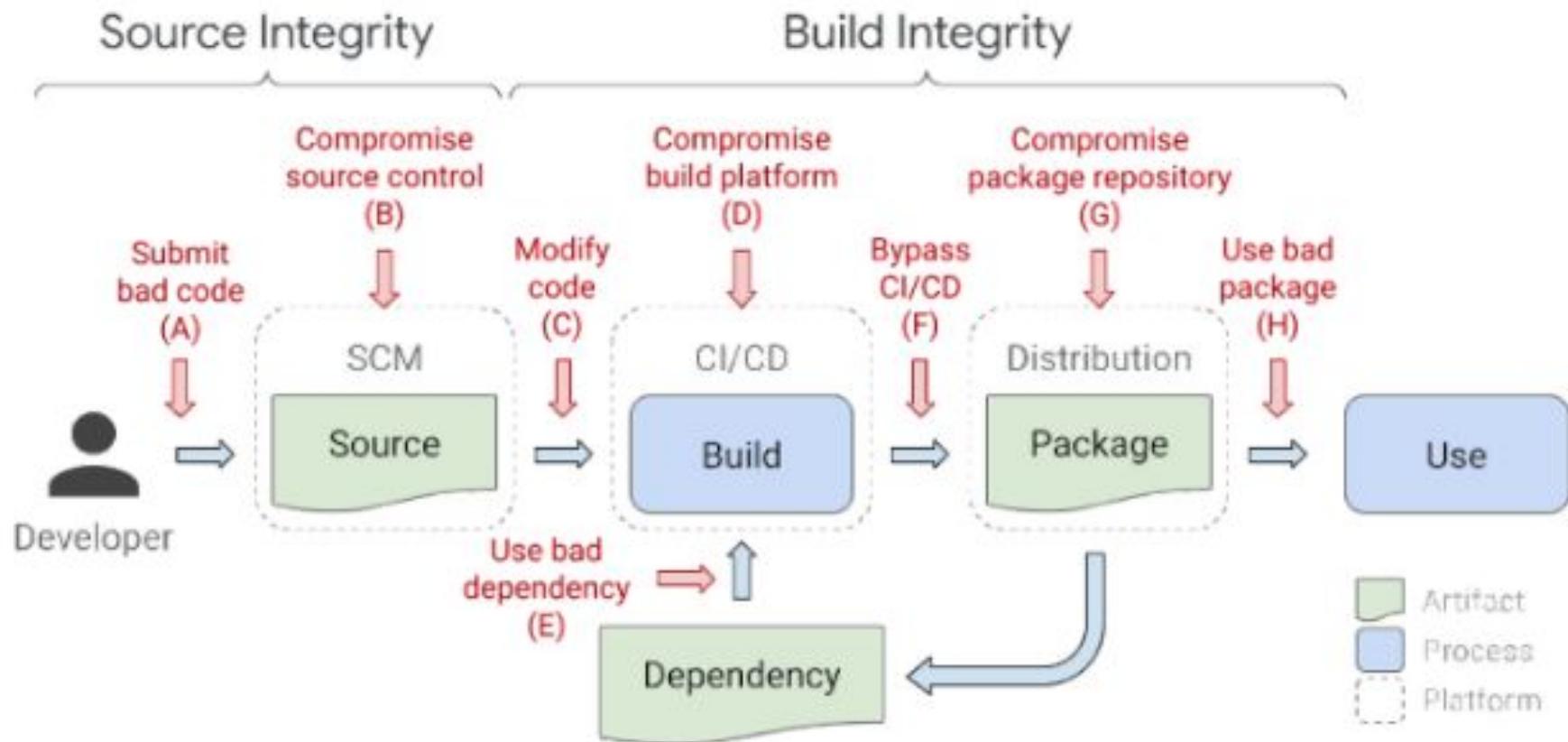


DEVOPS-1016 Der Nightly-Build von 'mega-keycloak' ist fehlgeschlagen



CVE-2022-12345

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	A2 - 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)



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

Architektur & Anforderungen

- Sicherheitskonzepte (IT Grundschutz)
- ggf. eigene BSI Bausteine
 - [↓ BausteinSecrets Management mit 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_und_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

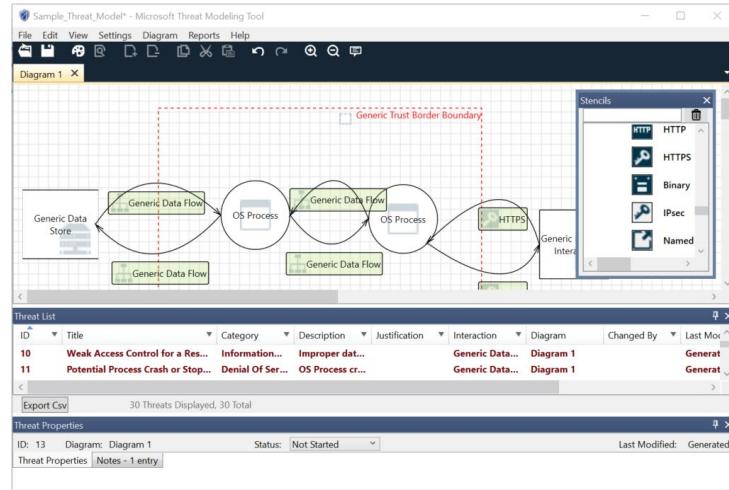
<https://www.threatmodelingmanifesto.org/>



Threat Modeling Tools



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



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

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
	log4j	
<input checked="" type="checkbox"/>	log4j	Apache

Sensibilisierung & Training

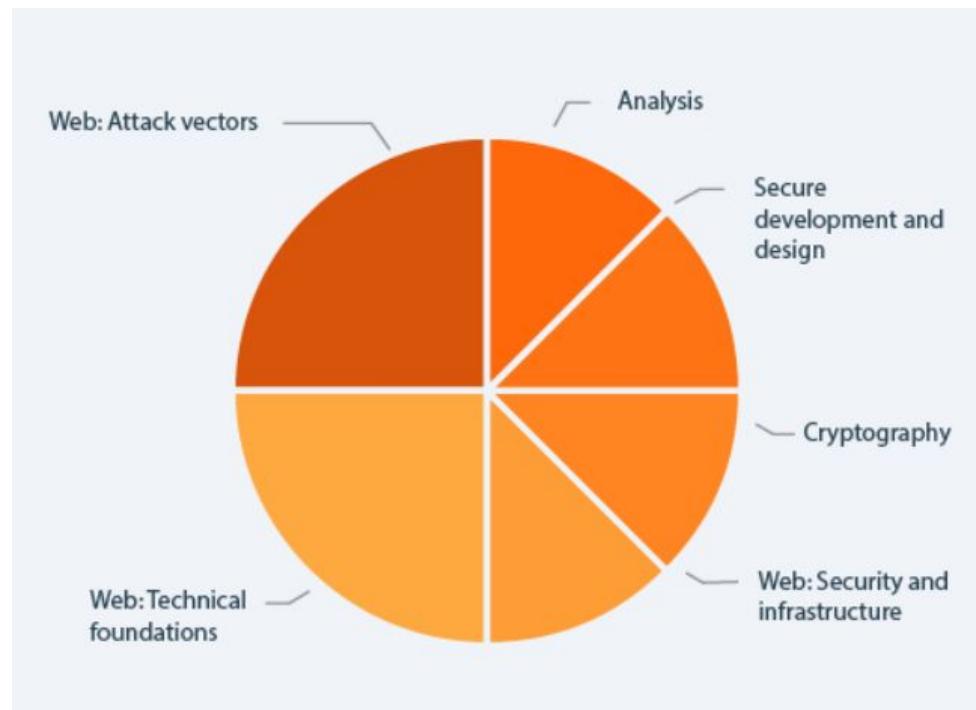
Informationsveranstaltungen

Live-Hacking

Schulungen

- ISAQB WebSec
- Certified Security Champion
Fraunhofer IEM

TechTalks



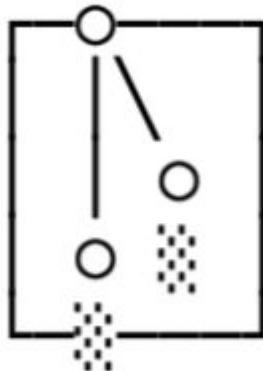
Secrets Detection



<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>

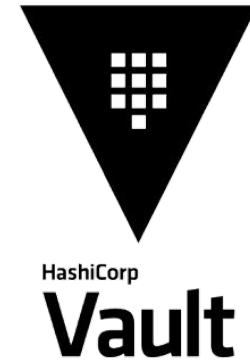
<https://github.com/renovatebot/renovate>

- 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

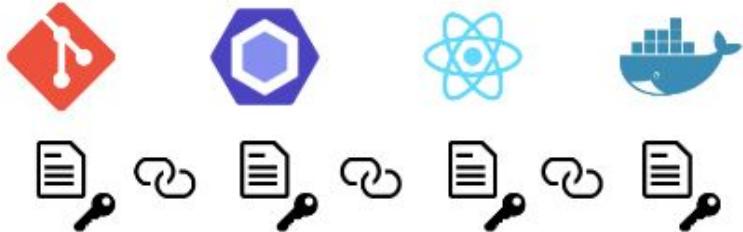
Eine Bescheinigung (Attest) ist eine kryptografisch signierte „Aussage“, die behauptet, dass etwas (ein „Prädikat“) über eine andere Sache (ein „Subjekt“) wahr ist.

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}"
                }
            }
        }
    }
}
```

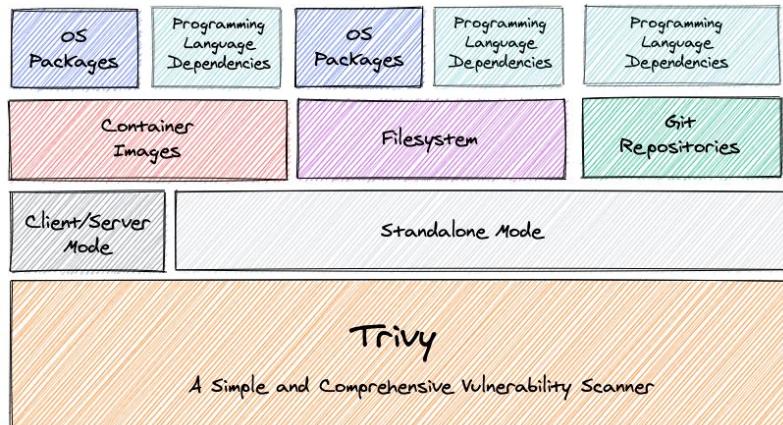
Secure Container Registry

- Vulnerability Scans
- Image Signing
- Pull Policies



HARBOR

<https://goharbor.io/>



Targets

Artifacts

Modes



<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



Info Artefakte

AKTIONEN ▾

Q | C

	Artefakte	Pull Befehl	Tags	Signiert mit Cosign	Größe	Schwachstellen	Annotationen	Label	Push Zeit	Pull Zeit
<input type="checkbox"/>	👉 sha256:add50289	<input type="button" value=""/>	latest	✗	86.14MiB	Keine Schwachstelle			04.11.22, 02:31	07.11.22, 01:00
<input type="checkbox"/> >	👉 sha256:e7d34034	<input type="button" value=""/>		✓	86.14MiB	Keine Schwachstelle			27.10.22, 14:39	07.11.22, 01:01
<input type="checkbox"/> >	👉 sha256:Oedb0aea	<input type="button" value=""/>		✓	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 with the 'Konfiguration' tab selected in the top navigation bar. On the left, there is a sidebar with various configuration options 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 displays several configuration sections:

- Projekt Registry**: A checkbox labeled 'Öffentlich' (Public) with the note: "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' with the note: "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) with the note: "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 stating: "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'). Individual CVE IDs can be added via 'VON SYSTEM KOPIEREN'.
- System-Allowliste** and **Projekt-Allowliste**: Radio buttons for selecting the allowlist type.
- HINZUFÜGEN** (Add): A button to add individual CVE IDs.
- VON SYSTEM KOPIEREN** (Copy from System): A button to copy the system's allowlist.
- Läuft ab am** (Expires at): A field for specifying an expiration date.
- Läuft niemals ab** (Never expires): A checked checkbox indicating the policy never expires.

At the bottom center, there is a box containing the text '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 user interface for a build history or dashboard. At the top, there's a header with a sun icon labeled "Build-Verlauf" and a "Trend" dropdown. Below the header is a search bar with a magnifying glass icon and the placeholder "suchen". A slash character is also present in the search bar.

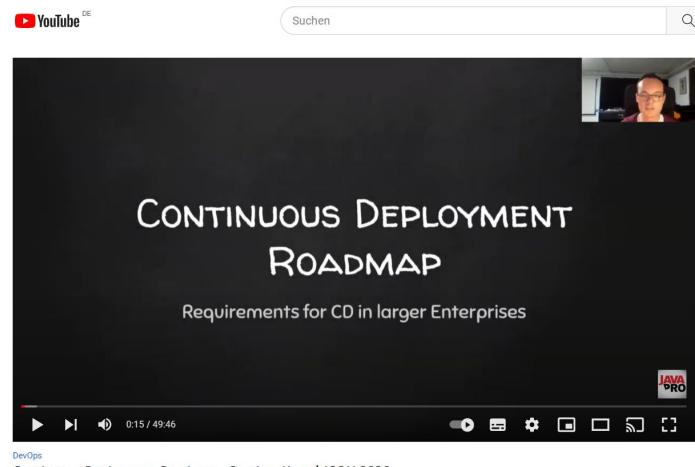
The main area displays a list of three build entries:

- #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

At the bottom of the interface, there are two links with RSS feed icons: "Atom feed aller Builds" and "Atom feed der Fehlschläge".

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 härteln



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



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

OWASP CSVS

Sicherheitsvorgaben

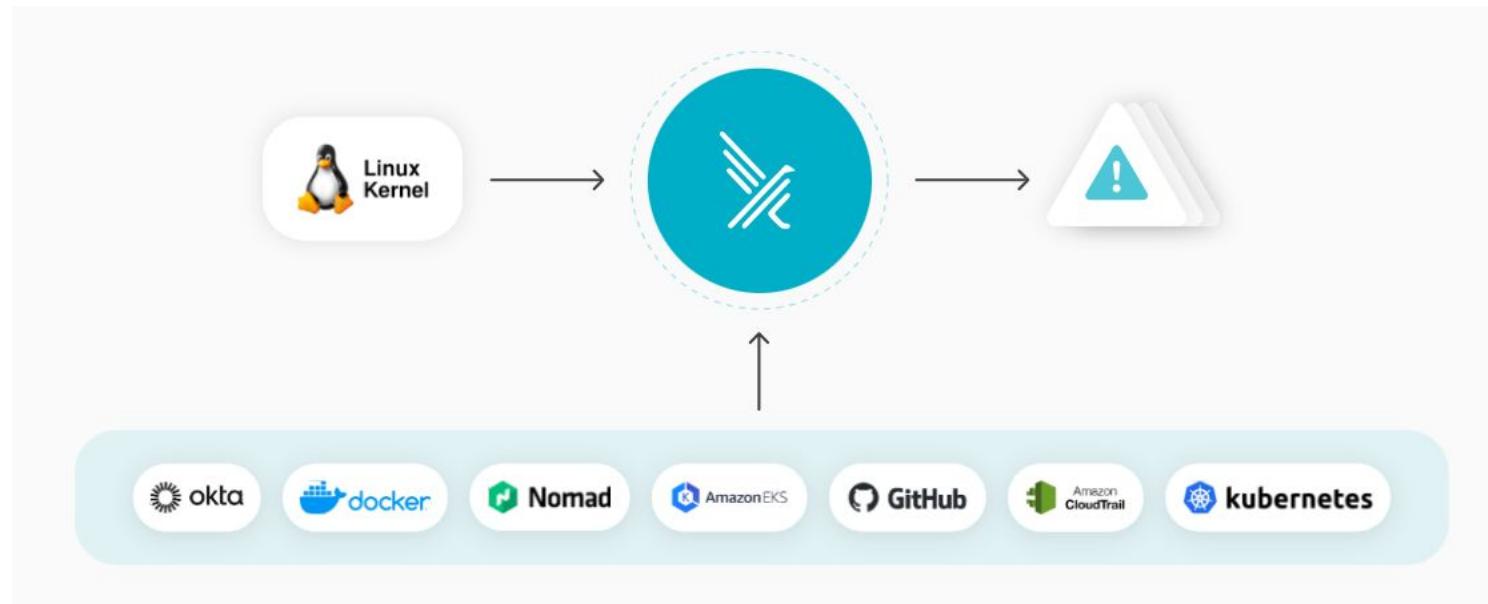
Bitte beachten Sie, dass die Anforderungen in diesem Abschnitt nicht vollständig sind, da sich viele organisatorische Sicherheitsvorgaben nicht nur auf Containerinfrastrukturen beschränken allerdings einen indirekten Einfluss auf diese haben können.

#	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/>

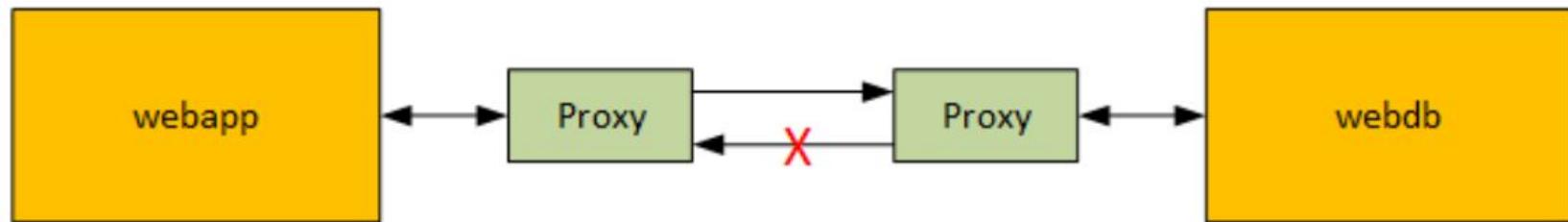
@kitencol

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
- Spawning processes using `execve`
- Executing shell binaries such as `sh`, `bash`, `csh`, `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:

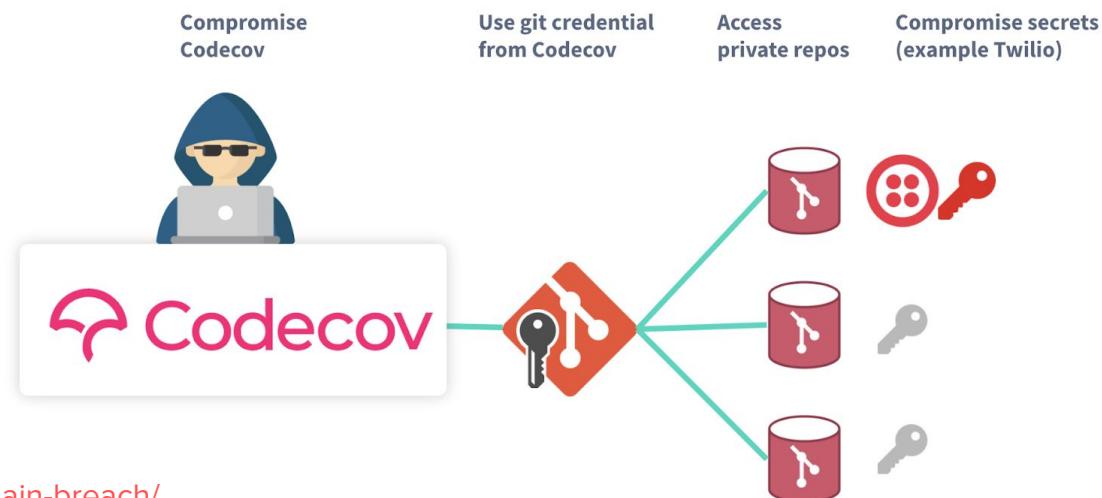
- kurzlebige Tokens
- häufig rotierte Zertifikate
- Mikrosegmentierung
- Mutual TLS
- Infrastructure as Code
- Workload Identities
- Kontinuierliche Überwachung



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

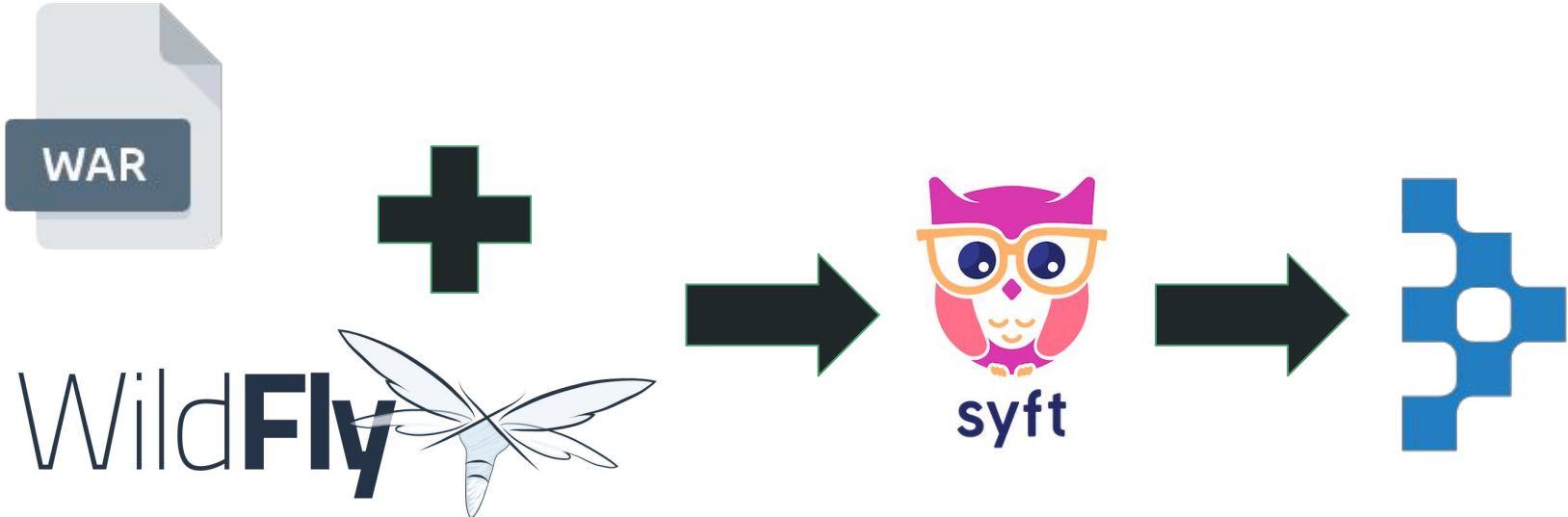
Beispiel: CodeCov (2021)

- Angreifer konnten Secrets aus Docker Images ziehen
- Schadhaftes 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 Schadhaftes unterschiebt?

Jemand eine Idee?

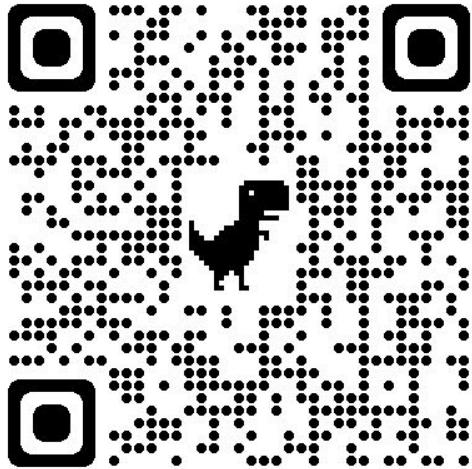
- Überprüfen der Integrität der Artefakte
- Vulnerability Scans, Image Signing & Pull Policy
- Vertrauenswürdige Artefakt-Repositories
- Container-Security Maßnahmen & Härtung der Hosts
- Attestierung und deren Prüfung
- Sensibilisierung & Training

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



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 Überlastungen in IT-Anwendungen genutzt hat. Doch im Dezember 2021 war die CVE-2021-41413 (zu Log4j) bzw. Log4Shell sogar die erste, die einen globalen Ausfall des Internets herbeigeführt.

Doch was haben wir daraus gelernt?

Haben wir Erkenntnisse gewonnen oder Methoden ergriffen, um bei nächsten Zero-Day-Vulnerability-Attacken schneller und sicherer reagieren zu können? Oder ist es weiterhin schwierig, eine solche Schwachstelle in einer Software zu entdecken?

Hätten wir aktuelle Beipackzettel zu jeder Software, eingerichtete 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 kommen viele weitere Informationen enthalten sind wie z.B. Lizenzinformationen.

Es unterscheidet zwei SBOMs (Standard Software Data Exchange), die nach SSBOM 1.0 und SSBOM 2.0 genannt werden. Ein Source-Code wie Codelock von ORNL oder Salt von der University of Illinois kann gleiches produzieren.

Log4Shell oder SolarWinds

Was haben wir daraus gelernt?



PRODUKT-INHALT IST UNBEKANNT

Aktuell befinden wir Software, von der wir nicht wissen, wie sie funktioniert. Es kann sich dabei um komplexe Systeme mit Millionen von Bauteilen handeln. Daher ist auch unklar, ob diese Produkte tatsächlich sicher sind, bevor man sie tatsächlich betrachten kann.



RECHERCHEN SIND UNFÄWIG

Werden schwierige Sicherheitslücken wie Log4Shell entdeckt, kann es schwierig sein, diese zu reparieren. Wenn ein Angreifer eine Firma per E-Mail oder gar Friedensdruck dazu bringt, dass sie die Sicherheitslücke ausnutzt, kann es schwierig werden.



WIR SIND EVENTUELLE ANGREIFBAR

Unterschreibt nicht nur schwierig, sondern auch teuer, wenn es darum geht, eine Software zu aktualisieren. Zudem kann es schwierig sein, internationale Anbieter der Software zu kontaktieren, um sie über Sicherheitslücken aufzuklären.

Was können wir selbst tun?



Regelmäßige Analyse auf Sicherheitslücken

Für spezielle Sicherheitslücken kann es nötig sein, eine SBOM zur Verfügung zu stellen, um die Anzahl der technischen Details zu begrenzen. So kann es einfacher sein, die Anzahl der technischen Details zu begrenzen. Durch gezielte Tests wird es leichter möglich, Zentrale Fehler in der Software zu erkennen, die aufgrund der gesammelten Bewertungsprofile aufwändiger untersucht werden.



Aufnahme der Verpflichtung in Ausschreibungen

In Ausschreibungen kann festgelegt werden, dass Hersteller von Software verpflichtet sind, jede übergeordnete Version auch eine SBOM bereitzustellen. Zusätzlich sollten sie aufgefordert werden, die weiteren Zulieferer-Behörden von Sicherheitslücken fortzusetzen.

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

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 abbrechen 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 angebundenen 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

 @kitencol

 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>

@kitencol



MC Flavour
Der Nächste
Request



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