

Sicherheit in der Software-Supply-Chain

Transparenz durch Attestations

> whoami - Tim Bastin

- Software Sicherheitsspezialist @ L3montree Cybersecurity
- Informatik an der Hochschule Bonn-Rhein-Sieg studiert
- Leidenschaftlicher Softwareentwickler
- Großes Interesse an Software Qualität
- OSS-Projekte:
 - Maintainer **OWASP**-Inkubationsprojekt DevGuard
 - Maintainer Security Badge Programm @ Zentrum für digitale Souveränität (ZenDiS)
 - Ex-Maintainer IT-Security-Scanner @ BMI



ANGRIFF VIA OPENSSH

Backdoor in XZ Utils gefährdet das Linux-Ökosystem

Zum Glück wurde die [Backdoor](#) entdeckt, bevor sie die breite Masse erreichen konnte. Angreifer hätten damit weltweit Millionen von [Linux](#)-Systemen infiltrieren können.

[golem.de](#)

«Das ist der verrückteste Angriff»: Ein Programmierer entdeckt per Zufall eine gefährliche Hintertüre im Code – wohl von einem Geheimdienst

[nzz.ch](#)

Aktenzeichen XZ aufgelöst

In den vergangenen Tagen sind wir mit sehr viel Glück dem wohl größten Fiasko in der Geschichte des Internets gerade so entgangen. Wie konnte das passieren?

[heise.de](#)

ZDNet / Alerts

XZ Utils-Vorfall: Open Source als Software Supply Chain-Falle

Ende März wurde in der XZ Utils Bibliothek, ein Kernbestandteil vieler Linux Distributionen, Schadcode entdeckt.

[zdnet.de](#)

Es ging um eine halbe Sekunde

Dieser Deutsche (38) hat das Internet gerettet

Andres Freund entdeckt Hacker-Angriff, jetzt feiert ihn die ganze Welt

[bild.de](#)

Was ist passiert?

Angreifer: Jia Tan
Dauer: Oktober 2021 bis März 2024
Methode: Social engineering.
Ziel: Implementierung einer Backdoor in die XZ-Uutils library

XZ-Uutils ist eine Bibliothek zum Komprimieren von Dateien

Sie wird in weit verbreiteten Linux-Distributionen verwendet

Der Angriff war mit großem Aufwand verbunden und zeugt von einem hohen Maß an Fachwissen.

Das Ziel war die Software-Lieferkette vieler Unternehmen und Organisationen

Es gibt ein Pattern!

RESEARCH

SECURITY NEWS

Tick Tock, Your Credentials Are Gone: The Maven Package With a Monthly Theft Schedule

A malicious Maven package typosquatting a popular library is secretly stealing OAuth credentials on the 15th of each month, putting Java developers at risk.

The Octopus Scanner Malware: Attacking the open source supply chain

This post details how an open source supply chain malware spread through build artifacts. 26 open source projects were backdoored by this malware and were actively serving backdoored code.

Gradle Wrapper Attack Report

January 25, 2023  Louis Jacomet  Security

Resources > Blog > Dependency hijacking software supply chain attack hits more ...

Dependency hijacking software supply chain attack hits more than 35 organizations

January 09, 2021

9 minute read time

Resources > Blog > Sonatype stops software supply chain attack aimed at the ...

Sonatype stops software supply chain attack aimed at the Java developer community

January 13, 2021 By Sonatype Security Research Team

9 minute read time

Aktuelle Schadsoftware in NPM (22.07.2025)

[GitHub Advisory Database](#) / [GitHub Reviewed](#) / CVE-2025-54313

eslint-config-prettier, eslint-plugin-prettier, synckit, @pkgr/core, napi-postinstall have embedded malicious code

High severity

[GitHub Reviewed](#)

Published last month to the GitHub Advisory Database • Updated last month

Vulnerability details

Dependabot alerts 0

Package


 @pkgr/core (npm)

Affected versions

= 0.2.8

Patched versions

0.2.9

 eslint-config-prettier (npm)

= 8.10.1


8.10.2

= 9.1.1

9.1.2

>= 10.1.6, <= 10.1.7

10.1.8

 eslint-plugin-prettier (npm)

>= 4.2.2, <= 4.2.3

4.2.4

 got-fetch (npm)

>= 5.1.11, <= 5.1.12

6.0.0

 napi-postinstall (npm)

= 0.3.1

0.3.2

 synckit (npm)

= 0.11.9

0.11.10

Severity

High 7.5 / 10


CVSS v3 base metrics


Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Changed
Confidentiality	Low
Integrity	High
Availability	None

[Learn more about base metrics](#)

Description

Noch aktuellere Schadsoftware in NPM (08.09.2025)


**Josh Junon**
@bad-at-computer.bsky.social






Yep, I've been pwned. 2FA reset email, looked very legitimate.

Only NPM affected. I've sent an email off to [@npmjs.bsky.social](#) to see if I can get access again.

Sorry everyone, I should have paid more attention. Not like me; have had a stressful week. Will work to get this cleaned up.

**charlieeriksen.bsky.social** @charlieeriksen.bsky.social
@bad-at-computer.bsky.social Hey. Your npm account seems to have been compromised. 1 hour ago it started posting packages with backdoors to all your popular packages.

Sep 8, 2025 at 17:15

 183  61  Reply

[Read 15 replies on Bluesky](#)

Noch aktuellere Schadsoftware in NPM (08.09.2025)



Noch noch aktuellere Schadsoftware in NPM (15.09.2025)

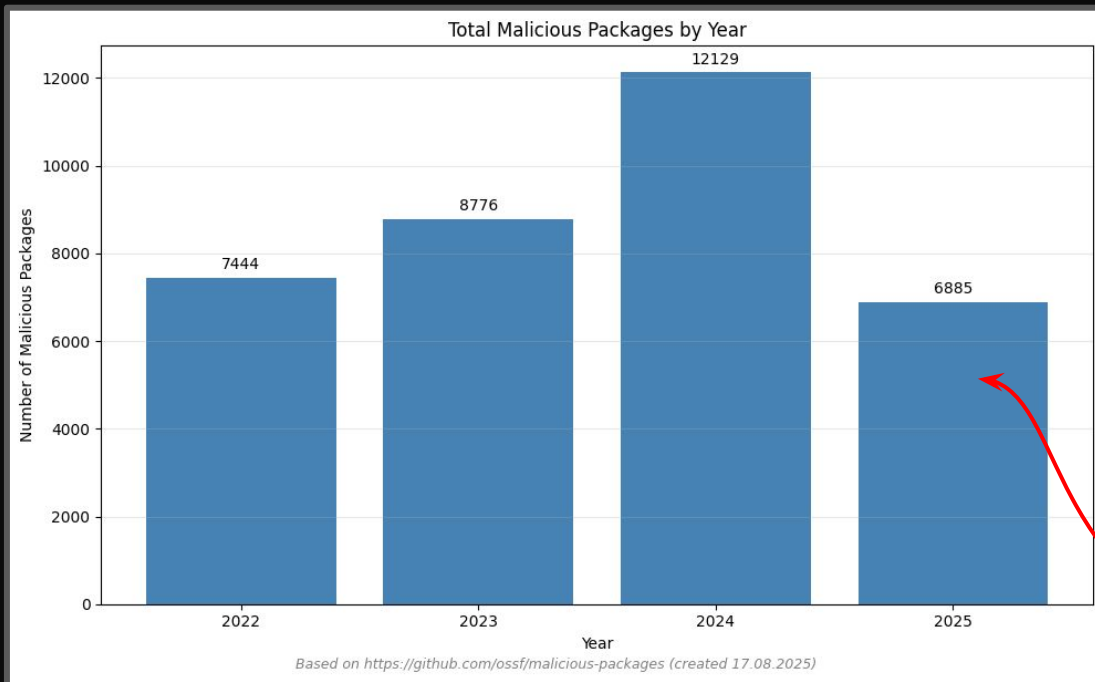
< BACK TO BLOG

Shai-Hulud: The novel self-replicating worm infecting hundreds of NPM packages

ALBERTO PELLITTERI AND MICHAEL CLARK



Die Bedeutung der Anwendungssicherheit steigt



Anwendungssicherheit bezieht sich auf den Prozess der Identifizierung und Reparatur von Schwachstellen in der Anwendungssoftware – **von der Entwicklung bis zur Bereitstellung** – [...]

Bis 17.08.2025

Was ist die Software-Supply-Chain?



Eine Software-Lieferkette besteht aus den Komponenten, Bibliotheken, Tools und Prozessen, die zur Entwicklung, Erstellung und Veröffentlichung eines Software-Artefakts verwendet werden.

[4] übersetzt

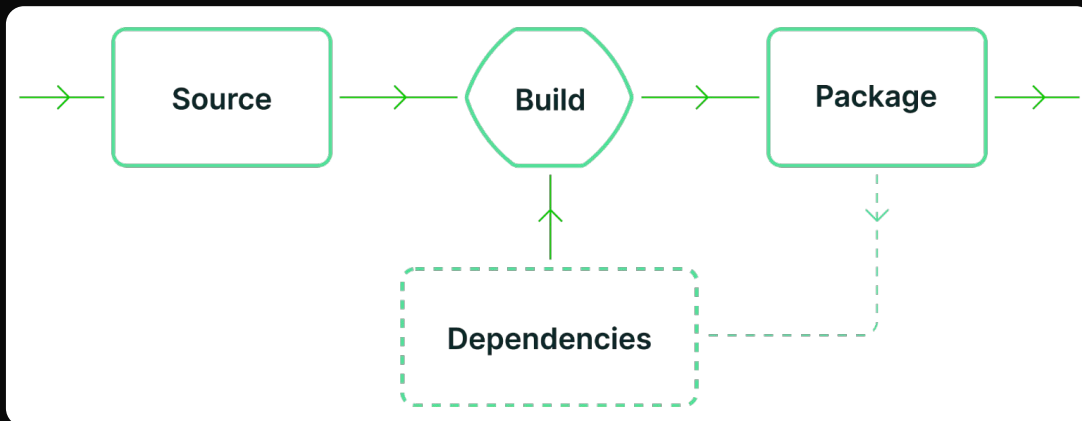
Die Softwarelieferkette nach SLSA

Lieferant

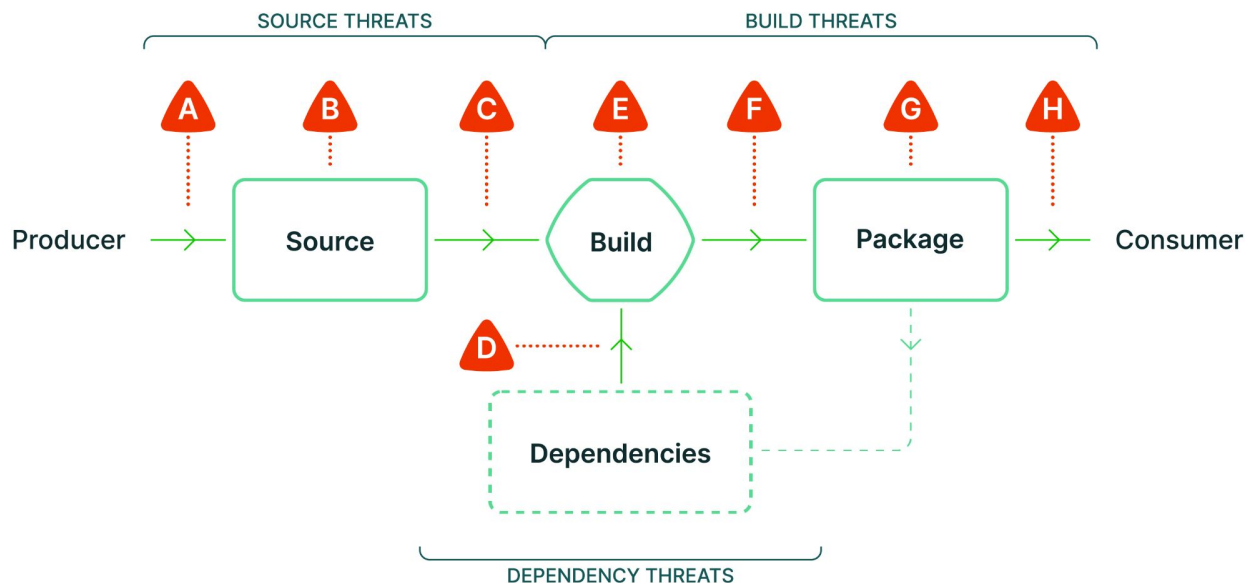
→ Eine Instanz, die Softwarekomponenten oder Softwarebibliotheken an eine folgende Instanz liefert.

Konsument

→ Die Instanz, die die vom Lieferanten produzierte Software nutzt. Konsumenten können auch Software an weitere Konsumenten oder wieder an sich selbst ausliefern.



Threat-Modell der Softwarelieferkette (SLSA)



SOURCE THREATS

- A** Submit unauthorized change
- B** Compromise source repo
- C** Build from modified source

DEPENDENCY THREATS

- D** Use compromised dependency

BUILD THREATS

- E** Compromise build process
- F** Upload modified package
- G** Compromise package registry
- H** Use compromised package

Software Supply Chain Attack Definition

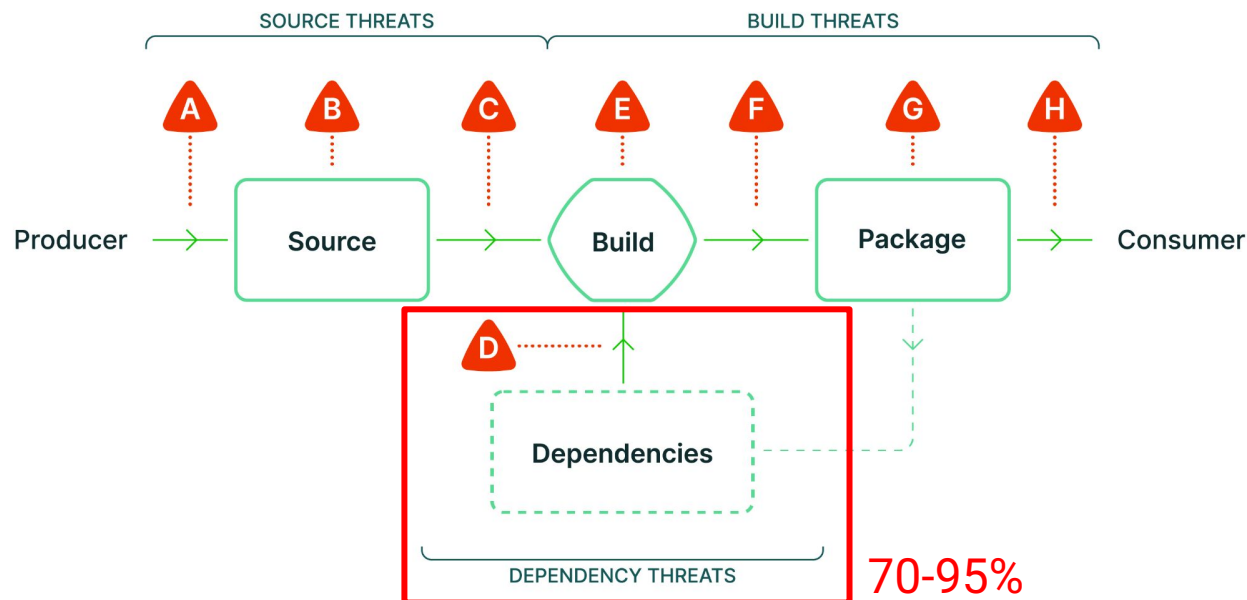
„Ein Angriff auf die Lieferkette ist eine Art von Cyberangriff, bei dem nicht die Organisationen direkt angegriffen werden, sondern die **dazwischenliegenden Parteien**, wie z. B. Anbieter und deren Softwarecode.“

[1] übersetzt

„[...] Angreifer manipulieren das Endprodukt eines bestimmten Anbieters so, [...], dass es **von den Endnutzern über vertrauenswürdige Vertriebskanäle**, z. B. Download- oder Update-Sites, bezogen werden kann.“

[2] übersetzt

Threat-Modell der Softwarelieferkette (SLSA)



SOURCE THREATS

- A** Submit unauthorized change
- B** Compromise source repo
- C** Build from modified source

DEPENDENCY THREATS

- D** Use compromised dependency

BUILD THREATS

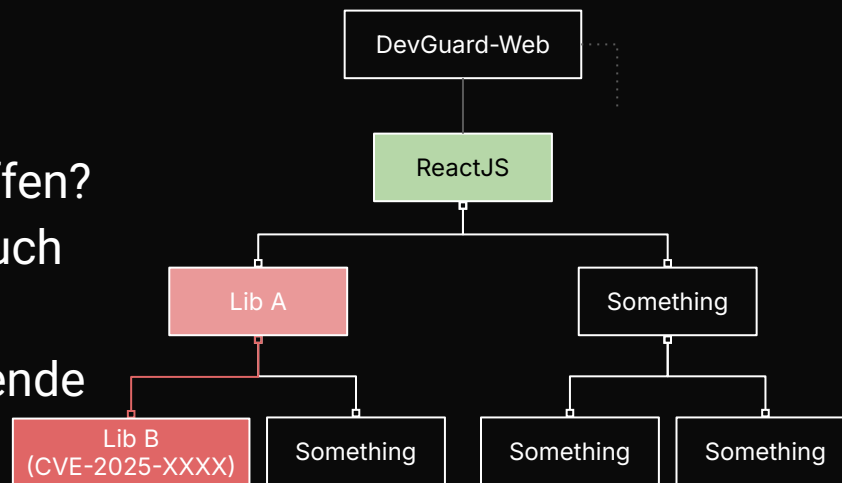
- E** Compromise build process
- F** Upload modified package
- G** Compromise package registry
- H** Use compromised package



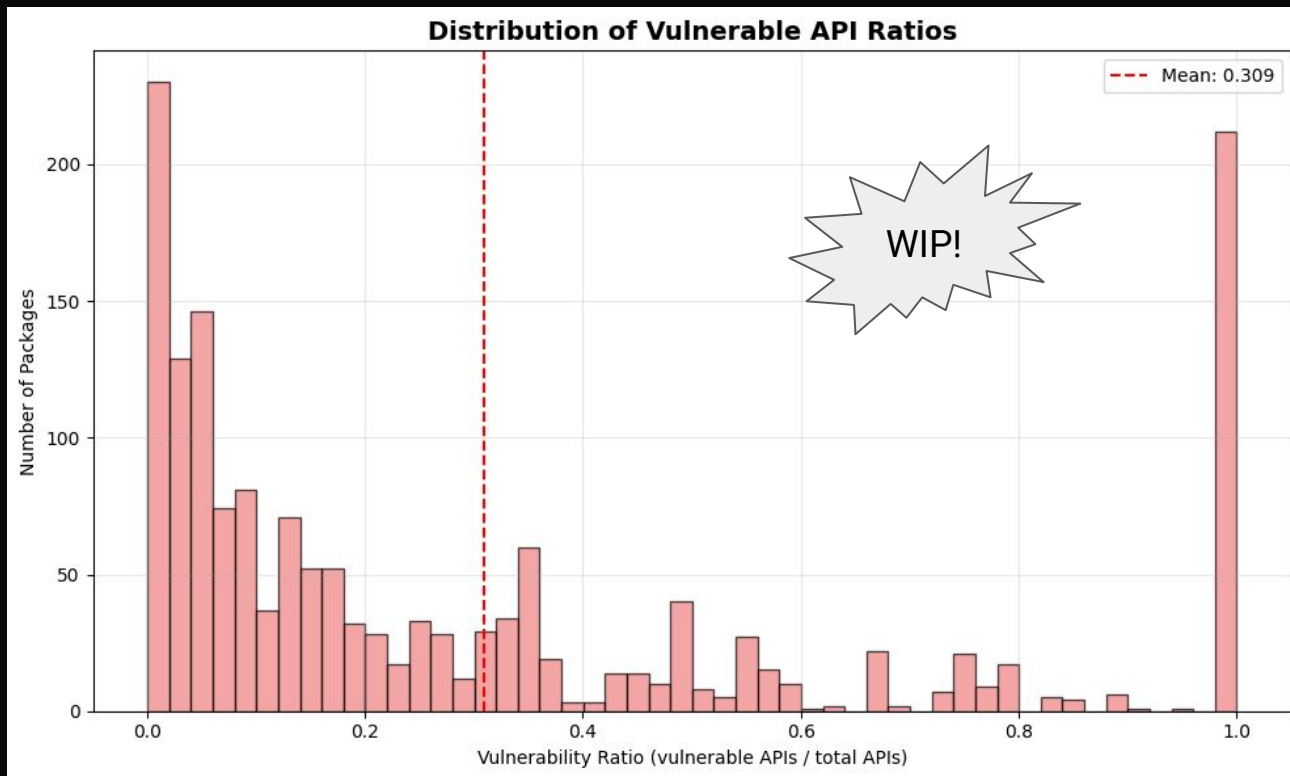
**Benutzen vulnerable oder
kompromittierte Komponenten**

Wie das Schwachstellenmanagement funktioniert

1. Ich habe eine kritische CVE!
2. Ist Lib A von CVE-2025-XXXX betroffen?
3. Wenn Lib A betroffen ist, ist dann auch ReactJS betroffen?
4. Wenn ReactJS betroffen ist – verwende ich dann die anfälligen Funktionen?



“Wie viel” einer Bibliothek ist wirklich betroffen?



Analyse der npm-Pakete
mit >100.000 Downloads
(48974)

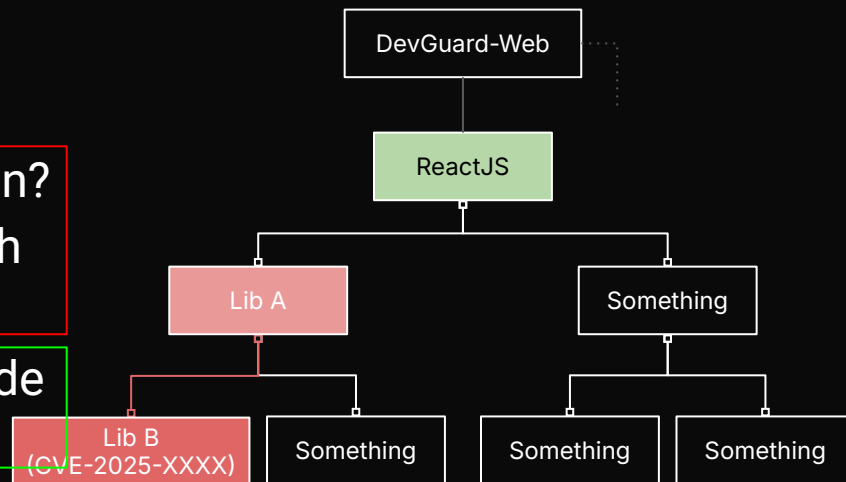
1623 Pakete mit
Schwachstellen

Wie das Schwachstellenmanagement funktioniert

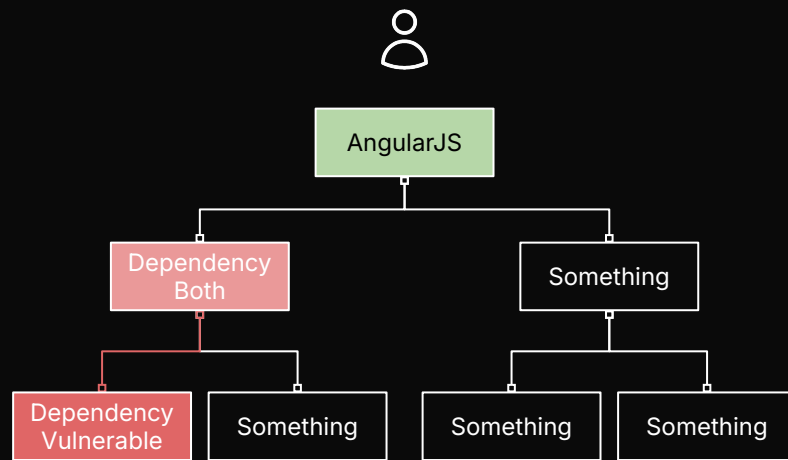
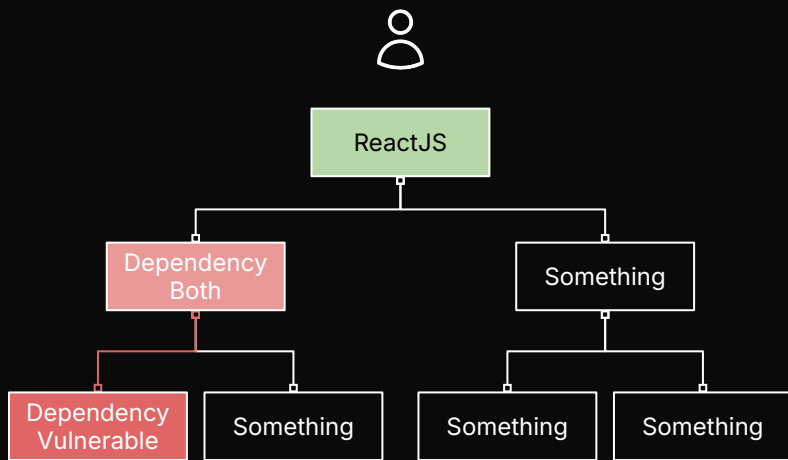
1. Ich habe eine kritische CVE!
2. Ist Lib A von CVE-2025-XXXX betroffen?
3. Wenn Lib A betroffen ist, ist dann auch ReactJS betroffen?
4. Wenn ReactJS betroffen ist – verwende ich dann die anfälligen Funktionen?

Hard part!

Easy



Crowdsourced VEXing




Lassen sich die Analyseergebnisse von Teilbäumen mit anderen teilen?
Oder Ergebnisse vom Upstream konsumieren?

Was ist der VeX

```


{
  ...
  "statements": [
    {
      ...
      "vulnerability": {
        "name": "CVE-2021-23017"
      },
      "status": "not_affected",
      "justification": "vulnerable_code_not_present",
      "impact_statement": "The vulnerable code path is not present in our build configuration."
    },
    {
      ...
      "vulnerability": {
        "name": "CVE-2022-41741"
      },
      "status": "affected",
      "action_statement": "Update to nginx version 1.23.2 or later to resolve this vulnerability."
    },
  ]
}
```

False-Positives im OCI-Image Ökosystem






golang:trixie MULTI-PLATFORM

LANGUAGES & FRAMEWORKS


INDEX DIGEST sha256:c8c8d55e02aa456e99a4fd0d2a224c0fb487d609c86a2550e8d64c3021b1df9a 

OS/ARCH





















linux/amd64 

COMPRESSED SIZE 	LAST PUSHED	TYPE	VULNERABILITIES	MANIFEST DIGEST
290.67 MB	2 days by doijanky	Image	<div><div>0</div><div>1</div><div>1</div><div>67</div><div>0</div></div>	sha256:9c0c9e53... 

Bin ich von CVE-2025-59375 in `expat` betroffen?

Vulnerabilities (69) Packages (292) [Give feedback](#) Analyzed by 

Package or CVE name ☐ Fixable ☐ Show excepted [Reset filters](#)

	CVE ID	Severity	Fixable	Present in	Affected package(s)
>	CVE-2025-59375	7.5 H			deb / debian/expat / 2.7.1-2
>	CVE-2025-45582	4.1 M			deb / debian/tar / 1.35+dfsg-3.1
>	CVE-2005-2541	N/A L			deb / debian/tar / 1.35+dfsg-3.1
>	CVE-2025-3198	N/A L			deb / debian/binutils / 2.44-3
>	CVE-2025-1152	N/A L			deb / debian/binutils / 2.44-3
>	CVE-2025-1150	N/A L			deb / debian/binutils / 2.44-3
>	CVE-2025-1181	N/A L			deb / debian/binutils / 2.44-3
>	CVE-2025-1147	N/A L			deb / debian/binutils / 2.44-3
>	CVE-2025-8225	N/A L			deb / debian/binutils / 2.44-3
>	CVE-2025-1149	N/A L			deb / debian/binutils / 2.44-3

1-10 of 69 [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#)

Wie können VeX-Informationen ausgetauscht werden?

Mit KI!

Mit Attestations!

Was sind Attestations?

Attestierungen bieten einen Mechanismus, um **kryptografisch signierte Metadaten (Jegliche JSON-Dateien)** direkt an Images oder Libraries anzuhängen, und ermöglichen so einen Informationsaustausch zwischen Entitäten in der Software Supply Chain.

Was sind Attestations?

1. Generate a key pair using `cosign`:

```
cosign generate-key-pair
```

This creates `cosign.key` (private key) and `cosign.pub` (public key).

2. Create an example JSON file:

```
{  
  "key": "test-attestation-1"  
}
```

Save this file as `some-json-file.json`.


3. Sign and upload the attestation:


```
cosign attest --predicate some-json-file.json --key cosign.key IMAGE_NAME --tlog-upload=false
```


- The attestation file will be stored as `digest(IMAGE_NAME).att`.
- You can attach multiple attestations (stored as image layers of `digest(IMAGE_NAME).att`), allowing different security assertions (e.g., SBOMs, vulnerability reports).


Was sind Attestations?



Ausschnitt aus GitLab-Container Registry


☐ sha256-bdaa476198815c8d6853308d2f471bd97fe18a089b578207d725cc52214f670e.att  ▾
168.69 KiB



☐ sha256-bdaa476198815c8d6853308d2f471bd97fe18a089b578207d725cc52214f670e.sig  ▾
1.04 KiB

☐ 2.3.1  ^
7.89 MiB

 Published to the *open-code/badgebackend/badge-api* image repository on August 11, 2025 at 4:45:07 PM GMT+2

 Manifest digest: sha256:bdaa476198815c8d6853308d2f471bd97fe18a089b578207d725cc52214f670e 

 Manifest media type: application/vnd.docker.distribution.manifest.v2+json

 Configuration digest: sha256:233f471966a587715c82b941112374900859cd8b2049b7c4feea1c88826093ae 

Was sind Attestations?

Ausschnitt aus GitLab-Container Registry

The screenshot displays the GitLab Container Registry interface for a specific container image. It lists three attestations, each with a checkbox, a filename, a size, and a download icon. The first two attestations are highlighted with red boxes. Below the list, a detailed view of the first attestation is shown, including its publication date and time, its manifest digest (also highlighted with a red box), its media type, and its configuration digest.

Checkbox	Filename	Size	Download Icon
<input type="checkbox"/>	sha256-bdaa476198815c8d6853308d2f471bd97fe18a089b578207d725cc52214f670e.att	168.69 KiB	Download
<input type="checkbox"/>	sha256-bdaa476198815c8d6853308d2f471bd97fe18a089b578207d725cc52214f670e.sig	1.04 KiB	Download
<input type="checkbox"/>	2.3.1	7.89 MiB	Download

Published to the *open-code/badgebackend/badge-api* image repository on August 11, 2025 at 4:45:07 PM GMT+2

Manifest digest: sha256:bdaa476198815c8d6853308d2f471bd97fe18a089b578207d725cc52214f670e

Manifest media type: application/vnd.docker.distribution.manifest.v2+json

Configuration digest: sha256:233f471966a587715c82b941112374900859cd8b2049b7c4feea1c88826093ae

Was kann attestiert werden

- Schwachstelleninformationen
- Provenance (Herkunftsnachweis)
- Test Ergebnisse
- SBOMs
- ...
- Alles, was sich als JSON darstellen lässt

Einzelne Ökosysteme beschränken die Attestierungstypen. npm bspw. erlaubt Stand September 2025 nur Provenance- und Publish-Attestations.

Rekursive Absicherung der Software Supply Chain

Kernprinzip

- **Rekursives System**, in dem jede Schicht zum akkumulierten Sicherheitswissen beiträgt und davon profitiert
- **Automatisierte Weitergabe** durch die gesamte Dependency Chain
- **Verbesserung der Supply Chain Transparency** auf jeder Stufe

1. Base Image Analysis

- Extraktion der finalen `FROM` Anweisung aus Containerfiles
- Berücksichtigung von Multi-Stage Builds
- Base Image wird zur Grundlage der Schwachstellenvererbungsanalyse

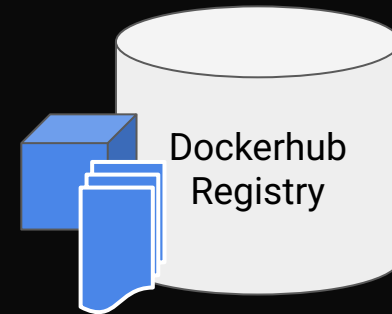
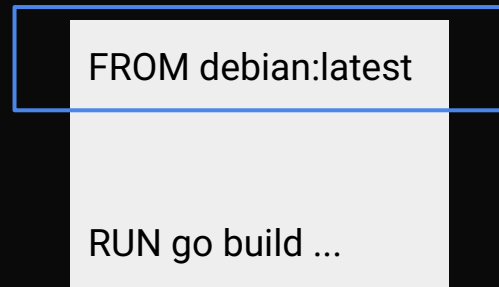


```
FROM debian:latest
```

```
RUN go build ...
```

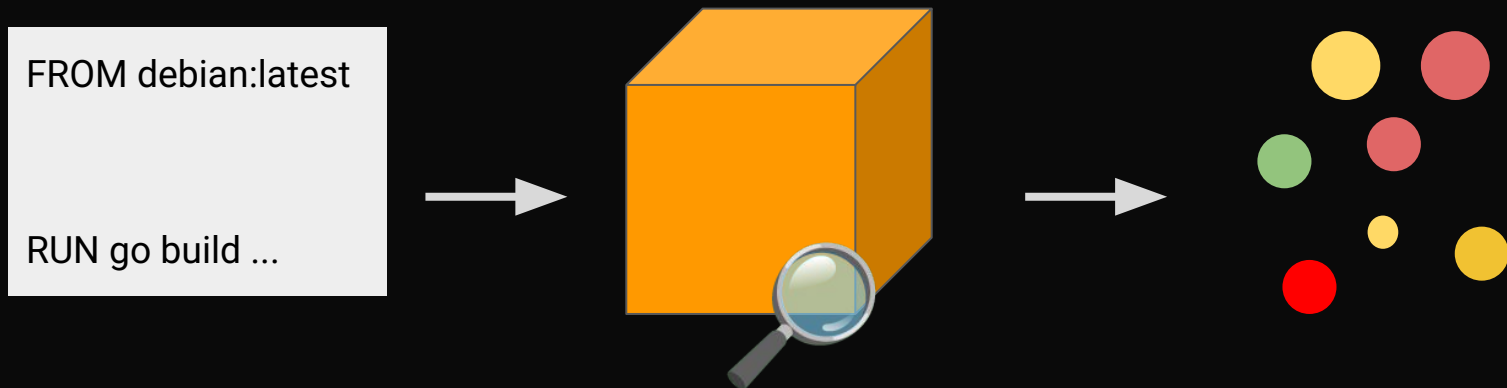
2. Attestation Discovery

- Abfrage der OCI Registry nach existierenden Attestations
- Abruf von VEX Dokumenten des Base-Images



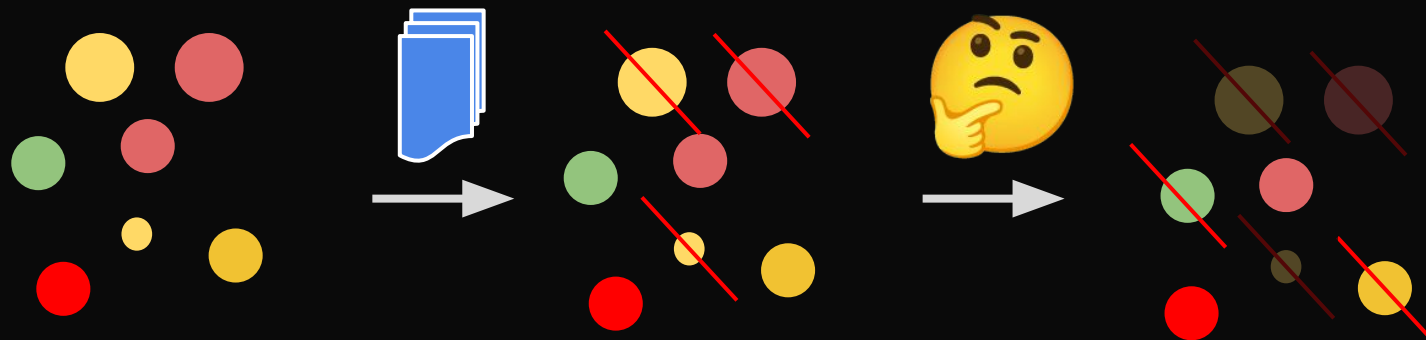
3. Sicherheitsüberprüfung des gebauten Images

- Umfassender CVE Scan des neu gebauten Images
- Generierung eines rohen Vulnerability Reports
- Initial alle identifizierten Schwachstellen, unabhängig von ihrer Exploitbarkeit



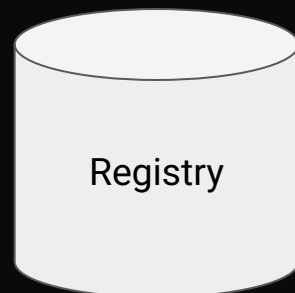
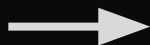
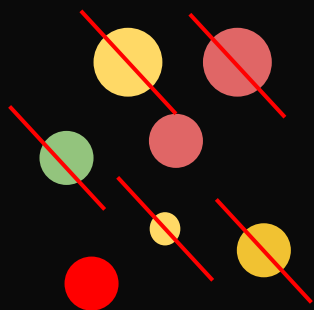
4. Filtern der gefundenen Schwachstellen

- Anwendung von VeX-Attestations zur Filterung der lokalen Scan-Ergebnisse
- Markierung bereits behandelter oder als nicht-exploitable eingestufter Schwachstellen → **Reduzierung** von False Positive Noise
- Eigene Analyse



5. Veröffentlichen der konsolidierten Information

- Zusammenführung der Base Image Vulnerability Intelligence mit neu entdeckten Issues
- Erstellung einer konsolidierten Sicht auf die Image Security Posture
- Vollständige Transparenz für Downstream Consumer



**Wieso nur VeX-Informationen? Was ist
mit Angriffen?**

Provenance-Attestation (SLSA)

- Wo wurde die Bibliothek / das Image gebaut?
- Wie wurde die Bibliothek / das Image gebaut?
- Wann wurde die Bibliothek / das Image gebaut?
- Was wurde verwendet, um die Bibliothek zu bauen?

Inhalt einer Provenance-Attestation

- **Builder:** Welches System wurde genutzt um das Artefakt zu bauen
- **Source Code Information:** Repository URL und Commit Hash, Commit Message, Committer...
- **Build-Prozess:** Kommandos, wie das Paket gebaut wurde und Umgebungsvariablen
- **Timestamps**

```

{
  "_type": "https://in-toto.io/Statement/v0.1",
  "predicate": {
    "buildDefinition": {
      "buildType": "",
      "externalParameters": {
        "author": "Sebastian Kawelke",
        "authoremail": "66557440+seb-kw@users.noreply.github.com",
        "branch": "main",
        "commitdate": "2025-09-17 18:11:57 +0200 +0200",
        "commitdigest": {
          "sha1": "1b78cec8ccc54d500ba845b1b118c76b98e4be46"
        },
        "commithash": "1b78cec8ccc54d500ba845b1b118c76b98e4be46",
        "commitmessage": "Merge pull request #1183 from l3montree-dev/chore/improve-testing-composeAdds
notice, comments port mapping - compose try it",
        "...",
        "signature": "...",
        "treehash": "84b063bb328b150c526c986de8c000361c588246",
        "username": "root",
        "variables": {
          "...",
          "ACTIONS_RUNTIME_URL":
"https://pipelinesghubeus7.actions.githubusercontent.com/jieWLMgLxLjWpb2rvhIrgFuKgEDIAfS0SsY10YACMplu9m6s0t/",
          "BUILD_ARGS": "--context=. --dockerfile=Dockerfile.scanner --no-push --tarPath
/github/workspace/tmp-image.tar",
          "CI": "true",
          "GITHUB_ACTION": "__ghcr_io_l3montree-dev_devguard_scanner_main-latest",
          "GITHUB_ACTIONS": "true",
        },
        "resolvedDependencies": [
          {
            "digest": {
              "sha256": "68216e8df3bc592127560639de1c5d4a897f9c8a56720e8b6f298fcd271c4c91"
            },
            "uri": "file://cmd/devguard/api/api_test.go"
          },
          ...
        ]
      },
      "runDetails": {
        "builder": {
          "id": "devguard.org"
        },
        "metadata": {}
      }
    },
    "predicateType": "https://slsa.dev/provenance/v1",
  }
}

```

Ecosystem-Support



Provenance



Built and signed on

GitHub Actions

[View build summary](#)

Source Commit github.com/sigstore/sigstore-js@21dd66d

Build File [.github/workflows/release.yml](https://github.com/workflows/release.yml)

Public Ledger [Transparency log entry](#)

=== Provenance Check Summary ===

Total packages checked: 48974

With provenance: 3997 (8.16%)

Without provenance: 44977



Using artifact attestations to establish provenance for builds

Artifact attestations enable you to increase the supply chain security of your builds by establishing where and how your software was built.

Producing attestations

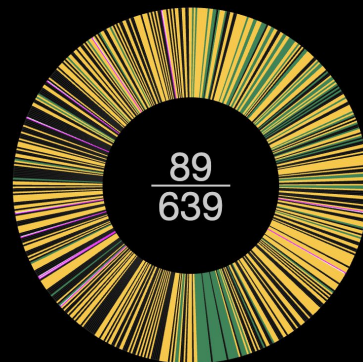


Info

Index attestations are currently under active development, and are not yet considered stable.

PyPI allows attestations to be attached to individual *release files* (source and binary distributions within a release) at upload time.

Are we attested yet? 📝



OWASP-DevGuard soll bei diesem Prozess unterstützen

- Automatische Attestierungen in der Pipeline
- VeXing und automatisierter Austausch durch Attestations
- Compliance as Code
- Prüfung und Erreichung der SLSA Levels

Quellen

- [1] GeeksforGeeks. (2023). *Evolution of Software Development | History, Phases and Future Trends*. [online] Available at: <https://www.geeksforgeeks.org/evolution-of-software-development-history-phases-and-future-trends/>.
- [2] Intelligence, G.T. (2020). Cybersecurity: Timeline. [online] Verdict. Available at: <https://www.verdict.co.uk/cybersecurity-timeline>
- [3] Bundesamt für Sicherheit in der Informationstechnik. (n.d.). Die Lage der IT-Sicherheit in Deutschland. [online] Available at: https://www.bsi.bund.de/DE/Service-Navi/Publikationen/Lagebericht/lagebericht_node.html.
- [4] "For Good Measure Counting Broken Links: A Quant's View of Software Supply Chain Security" (PDF). USENIX ;login. Retrieved 2022-07-04.
- [5] Wintergerst, R. and Berlin, B.-P. (2023). Wirtschaftsschutz 2023. [online] Available at: <https://www.bitkom.org/sites/main/files/2023-09/Bitkom-Charts-Wirtschaftsschutz-Cybercrime.pdf>.
- [6] owasp.org. (n.d.). OWASP DevSecOps Guideline | OWASP Foundation. [online] Available at: <https://owasp.org/www-project-devsecops-guideline/>.
- [7] Supply-chain levels for software artifacts. (n.d.-b). SLSA. <https://slsa.dev/>

Quellen

- [8] Supply chain compromise, Technique T1195 - Enterprise | MITRE ATT&CK®. (n.d.). <https://attack.mitre.org/techniques/T1195/>
- [9] Bedrohungen der Softwarelieferkette. (n.d.). Google Cloud. <https://cloud.google.com/software-supply-chain-security/docs/attack-vectors>
- [10] Sonatype Inc. (n.d.). 2020 Software Supply Chain Report | Download. <https://www.sonatype.com/resources/white-paper-state-of-the-software-supply-chain-2020>
- [11] www.threatmodelingmanifesto.org. (n.d.). Threat Modeling Manifesto. [online] Available at: <https://www.threatmodelingmanifesto.org/>.
- [12] ReversingLabs: The state of Software Supply Chain Security 2024
- [13] Perry, N. et al. (2023) 'Do Users Write More Insecure Code with AI Assistants?', in Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: Association for Computing Machinery (CCS '23), pp. 2785–2799. doi: 10.1145/3576915.3623157.
- [14] Li Zhong, Z. W. (2024) Can LLM Replace Stack Overflow? A Study on Robustness and Reliability of Large Language Model Code Generation. Available at: <https://arxiv.org/html/2308.10335v5>.
- [15] No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT (2023). Available at: <https://arxiv.org/abs/2308.04838>.