

ProjectWork_CIS

Inhaltsverzeichnis:

1	Docker-Umgebung	3
1.1	Docker Konstellation 1	3
2	Sicherheits-scans & Tools	3
2.1	SBOM-Erstellung mit Syft	3
2.2	Scannen mit Trivy	3
3	Alle Scan in JSON Format → Website hochzuladen	4
4	Sicherheitsbewertung Docker-Setup.....	4
4.1	MySQL-Container (local/mysql:8.0-fixed)	5
4.2	PHP-Webcontainer (local/php-db-app:latest).....	5
5	Sicherheit im Entwicklungszyklus (SECDevOps)	6
5.1	PLAN: Sicherheitsplanung und Bedrohungsanalyse	6
5.2	CODE: Sicherer Quellcode und Abhängigkeiten	7
5.3	BUILD: Containererstellung und Härtung.....	7
5.4	TEST: Sicherheitsüberprüfung.....	7
5.5	RELEASE: Richtlinienkontrolle	7
5.6	DEPLOY: Sichere Bereitstellung	8
5.7	OPERATE: Betrieb und Überwachung	8
6	Container-Security nie 100 % automatisierbar.....	8
6.1	CVE-Datenbanken sind generisch.....	8
6.2	Automatisierte Patches haben Grenzen	9
6.3	Relevanz hängt vom Einsatz ab	9
6.4	Security-Scans = Momentaufnahme	9
6.5	Workflow → Empfehlung für die Praxis	9
6.6	Zusammenfassung	10
7	NIS2 & BSI – Relevanz für unser Projekt	10
8	Wie gehen Firmen damit um?	11
9	Aikido	12
9.1	Container & Image Scanning Fertiglösungen.....	12

9.2	CVE-Bewertung & Priorisierung.....	12
9.3	Quellcode-Scan (SAST & SCA)	13
9.4	Aikido in der Praxis:.....	13

1 Docker-Umgebung

2x Docker Container vom öffentlichen Docker Repository mit kleiner Webapplikation.

Diese WebApplikation ist nicht mehr Teil der Endabgabe, aber haben wir am Anfang des Projekts gemacht, einen PHP Container und einen Datenbankcontainer, wo man die JSON Scans von Trivy & Syft hochladen kann und dann einen Report bekommt.

1.1 Docker Konstellation 1

- **Image 1:** local/php-db-app:latest
- **Image 2:** mysql:8.0
- **Docker Build:**
 - `docker compose up --build -d`

2 Sicherheitsscans & Tools

2.1 SBOM-Erstellung mit Syft

Syft von Anchore ist ein SBOM-Generator.

Er erkennt in Docker-Images:

- Betriebssystem-Pakete (z.B. Debian, Ubuntu, Alpine)
- Anwendungsbibliotheken (Composer, npm, pip, Go, ...)
- Versionen, Herkunft, Hashes und Lizenzinformationen

Befehle:

- `syft local/php-db-app:latest -o cyclonedx-json > sbom.cyclonedxwww.json`
- `syft mysql:8.0 -o cyclonedx-json > sbom.cyclonedxdb.json`

Ergebnis:

Eine CycloneDX- oder SPDX-kompatible JSON-Datei, die den kompletten Software-Inhalt beschreibt. (enthält aber keine Sicherheitsbewertungen)

2.2 Scannen mit Trivy

Trivy ist eine Open-Source Vulnerability Scanner, der CVEs in Images, Dateien oder SBOMs erkennt.

Scan einer SBOM-Datei:

- `trivy sbom sbom.cyclonedxwww.json > trivy-sbomwww-report.json`

(oder ohne TXT Part für Konsolenansicht)

- `trivy sbom sbom.cyclonedx.json > trivy-sbomdb-report.txt`

Direkter Image-Scan:

- `trivy image local/php-db-app:latest`
- `trivy image mysql:8.0`

Syft → erstellt den genauen Inhaltskatalog (SBOM)

Trivy → bewertet diesen Katalog auf Schwachstellen (CVE-Analyse)

Trivy

- Trivy schaut in dein Image und gleicht alle gefundenen Pakete gegen seine CVE-Datenbank ab.
 - `trivy image local/php-db-app:latest`

Eine SBOM scannen (z. B. die von Syft erzeugte)

- Trivy prüft die darin gelisteten Komponenten auf bekannte Sicherheitslücken. Beispiel:
 - `trivy sbom sbom.cyclonedx.json`

3 Alle Scan in JSON Format → Website hochzuladen

Zur automatisierten Weiterverarbeitung werden alle Sicherheitsscans in JSON-Format exportiert:

- `trivy image --format json --output reports/trivy-image.json local/php-db-app:latest`
- `syft local/php-db-app:latest -o cyclonedx-json > reports/sbom.cyclonedx.json`
- `trivy sbom --format json --output reports/trivy-sbom.json reports/sbom.cyclonedx.json`
- `trivy image --format json --output reports/trivy-db.json mysql:8.0`

4 Sicherheitsbewertung Docker-Setup

Ziel: Identifikation und Bewertung von Schwachstellen (CVEs), die Aktualisierung relevanter Pakete sowie die Beurteilung des Sicherheitsniveaus. In diesem Fall haben

wir versucht die Schwachstellen händisch zu bewerten und zu eruieren, ob ein Update / weitere Konfiguration notwendig ist.

4.1 MySQL-Container (local/mysql:8.0-fixed)

Basis: mysql:8.0

Identifizierte Schwachstellen:

- **CVE-2025-47273** (*setuptools*, Path Traversal Vulnerability)
- **CVE-2025-8869** (*pip*, Symbolic Link Handling in tar archives)

Maßnahmen:

- *setuptools* wurde auf Version 78.1.1 aktualisiert → CVE-2025-47273 geschlossen.
- *pip* konnte aufgrund der schreibgeschützten eingebetteten Python-Runtime nicht aktualisiert werden.
 - Diese Komponente wird im MySQL-Container nicht verwendet, da kein Python-Paketmanagement erfolgt.
 - CVE-2025-8869 wurde als nicht ausnutzbar eingestuft.

Bewertung:

MySQL-Container gilt als sicher.

Die einzige relevante Schwachstelle (*setuptools*) wurde behoben, weitere CVEs sind auf nicht genutzte Komponenten oder Tools beschränkt.

4.2 PHP-Webcontainer (local/php-db-app:latest)

Basis: php:8.1-apache (Debian Bookworm Slim)

Identifizierte Schwachstellen:

Trivy meldet mehrere *High*-Severity CVEs in Systembibliotheken, z. B.:

Paket	CVEs	Bewertung
libssl3 / openssl	CVE-2023-3817	Bezieht sich auf TLS-Serverfehler; Container nutzt TLS nicht direkt, sondern nur über Apache – Patch durch apt-get upgrade erfolgt
libgnutls30	CVE-2023-0361	Betrifft DTLS-Kommunikation, im Apache-Webserver-Setup nicht verwendet
libzstd1 / liblz4-1	CVE-2022-4899	Kompressionsbibliotheken, nur für interne Archive – kein direkter Angriffsvektor
linux-libc-dev	CVE-2019-19814	Kernel-Header-Paket, wird im Containerlauf nicht ausgeführt
binutils / cpp / make	CVE-2021-42574	Build-Tools, nicht installiert oder zur Laufzeit nicht nutzbar
curl	CVE-2025-10148	Harmlos (predictable WebSocket mask); Fix folgt im nächsten Debian-Release

Maßnahmen:

- Regelmäßige Systemupdates werden im Build-Prozess ausgeführt:
- `RUN apt-get update && apt-get upgrade -y && apt-get clean`

→ alle sicherheitsrelevanten Laufzeitbibliotheken (z. B. openssl, curl, libc6) werden auf die neuesten Debian-Sicherheitsstände gebracht.

- Nicht benötigte Pakete (Compiler, Build-Tools) können optional entfernt werden, um CVE-Noise zu reduzieren.

Bewertung:

- PHP-Webcontainer gilt als betriebs- und sicherheitsbereit.
- Alle zur Laufzeit verwendeten Bibliotheken sind gepatcht.
- Die verbliebenen High-Einträge beziehen sich auf inaktive Build-Pakete, die im laufenden Container nicht ausführbar sind und daher kein Risiko darstellen.
- Beide Container wurden im Rahmen der Sicherheitsanalyse mit *Trivy* geprüft.
- Für den MySQL-Container war ein konkreter Fix (CVE-2025-47273 in setup tools) erforderlich und erfolgreich umgesetzt.
- Der PHP-Webcontainer wurde durch ein Upgrade aller Systembibliotheken (`apt-get upgrade`) auf den neuesten Debian-Sicherheitsstand gebracht.
- Die verbleibenden „High“-Severity-Meldungen betreffen Build- oder Kompressionsbibliotheken, die im Containerlauf nicht aktiv verwendet werden und keinen Angriffsvektor bieten.
- Damit gilt das Gesamtsystem als sicher gemäß aktuellem Patch-Stand (November 2025).

5 Sicherheit im Entwicklungszyklus (SECDevOps)

Um die Sicherheit unserer Docker-Umgebungen über den gesamten Entwicklungszyklus hinweg zu gewährleisten, werden in jeder Phase gezielte Maßnahmen umgesetzt. Dies ermöglicht eine frühzeitige Erkennung von Schwachstellen und reduziert Risiken.

5.1 PLAN: Sicherheitsplanung und Bedrohungsanalyse

In der Planungsphase steht die Identifikation von Risiken im Vordergrund.

Hierfür soll ein **Threat Modelling** durchgeführt werden, um potenzielle Bedrohungen für die Webapplikation und Containerumgebung zu erkennen.

Es sollte **Security Requirements** definiert werden:

- Verwendung ausschließlich vertrauenswürdiger Images aus offiziellen Quellen → Vermeidung von unsicheren Basisimages

- Nutzung aktueller Versionen von Betriebssystemen und Software → bekannte Sicherheitslücken zu schließen
- Minimierung der installierten Software → um die Angriffsfläche der Container zu reduzieren

5.2 CODE: Sicherer Quellcode und Abhängigkeiten

Während der Entwicklungsphase liegt der Fokus auf sicherer Programmierung und der Überprüfung aller Abhängigkeiten:

- **Statische Codeanalyse:** Automatische Prüfungen erkennen fehlerhafte oder unsichere Programmiermuster
- **Dependency Check:** Überprüfung aller eingesetzten PHP-Pakete, um Schwachstellen frühzeitig zu identifizieren

5.3 BUILD: Containererstellung und Härtung

Beim Erstellen der Images werden die enthaltenen Komponenten dokumentiert und Container gehärtet:

- **SBOM-Erzeugung** (Syft): Alle Pakete und Versionen werden dokumentiert, sodass die Bestandteile des Images nachvollziehbar sind.
- **Container-Hardening:**
 - Nur notwendige Pakete installieren
 - Entfernung temporärer Dateien und unnötiger Tools, um die Angriffsfläche zu verringern.

5.4 TEST: Sicherheitsüberprüfung

In der Testphase sollen die erstellten Images aktiv auf Schwachstellen geprüft werden:

- **CVE-Scan** (Trivy): Erkennt bekannte Sicherheitslücken in Betriebssystempaketen
- **Security Unit Tests:**
 - Überprüfung unsicherer Ports und fehlerhafter Dateiberechtigungen
 - Tests für sichere Kommunikation zwischen Web- und Datenbank-Container

5.5 RELEASE: Richtlinienkontrolle

Vor der Veröffentlichung soll sichergestellt werden, dass nur geprüfte und vertrauenswürdige Versionen freigegeben werden:

- **Signierung von Images:** Bestätigung der Authentizität und Integrität

- **Policy Enforcement:**
 - Zulassung nur signierter oder geprüfter Images
 - Automatische Blockierung von Builds, die kritischen Schwachstellen enthalten

5.6 DEPLOY: Sichere Bereitstellung

Bei dieser Phase liegt der Fokus auf Kommunikationsschutz und Zugriffskontrolle:

- Nutzung einer sicheren **Container Registry** (Authentifizierung und HTTPS)
- **TLS-Verschlüsselung**
- **Secrets Management:**
 - Keine Passwörter im Code
 - Verwendung von Docker Secrets/verschlüsselte Umgebungsvariable
- Einschränkung **Netzwerkzugriffe** auf das notwendige Minimum

5.7 OPERATE: Betrieb und Überwachung

Umgebung wird kontinuierlich überwacht, aktualisiert und gepflegt:

- **Runtime Scans:** Überprüfung der laufenden Container auf neue Sicherheitslücken
- **Log-Analyse:** Überwachung von Zugriffen, Fehlern und Auffälligkeiten
- **Patch-Management:** Regelmäßige Updates der Images und Pakete, um bekannte Sicherheitslücken zu schließen.
- **Incident Response:** Vorgehensweise bei Sicherheitsvorfällen, Analyse und Wiederherstellung

6 Container-Security nie 100 % automatisierbar

6.1 CVE-Datenbanken sind generisch

- Trivy, Anchore, Grype, etc. ziehen ihre Daten aus öffentlichen CVE-Feeds (z. B. NVD, Debian, Red Hat).
- Diese Listen *jede* bekannte Schwachstelle für ein Paket — unabhängig davon, ob sie in deinem Container überhaupt relevant oder ausnutzbar ist.
- Beispiel:
Ein CVE in systemd betrifft Dienstverwaltung — aber dein Container startet gar

kein systemd.
→ Scanner meldet „HIGH“, real aber irrelevant.

6.2 Automatisierte Patches haben Grenzen

- apt-get upgrade aktualisiert nur Debian-Pakete, die offiziell über Repos verteilt werden.
- Eingebettete Komponenten (wie mysqlsh-Python oder statisch kompilierte Tools) → können nur manuell oder vendorseitig aktualisiert werden.
- Selbst wenn man wöchentlich neu baut, bleiben solche Komponenten auf dem alten Stand, bis der Hersteller (z. B. Oracle) ein neues Base-Image veröffentlicht.

6.3 Relevanz hängt vom Einsatz ab

- Nicht jede Schwachstelle ist in jedem Kontext ausnutzbar.
Man muss bewerten:
 - Wird die betroffene Komponente überhaupt genutzt?
 - Ist sie von außen erreichbar?
 - Läuft sie mit Privilegien?
- Beispiel aus dem Projekt:
pip 24.2 → CVE vorhanden, aber nicht exploitable, weil pip in deinem MySQL-Container nie aufgerufen wird.

6.4 Security-Scans = Momentaufnahme

- Trivy prüft nur, was im aktuellen Image installiert ist und was die Datenbank zu diesem Zeitpunkt weiß. Es kann nicht den laufenden Container scannen, sondern aus dem Container müsste ein Image erstellt werden und dieses kann Trivy wieder scannen.
- Wenn morgen eine neue CVE veröffentlicht wird, kann dein heutiger „clean“-Scan morgen schon wieder „HIGH“ anzeigen.
- **Maßnahmen:**
Regelmäßige, geplante Scans, teilautomatisierte Updates → Github Workflow

6.5 Workflow → Empfehlung für die Praxis

Schritt	Automatisierbar	Beschreibung
Base-Image aktualisieren	JA	docker pull + Rebuild
Security-Scan mit Trivy	JA	GitHub Action oder Cronjob
Patchen via apt-get upgrade	JA	im Dockerfile integriert

Patchen eingebetteter Komponenten (pip, mysqlsh, etc.)	TEILMANUELL	wenn CVE relevant und Patch vorhanden
Bewertung der CVE-Relevanz	NEIN	muss manuell oder durch Security-Gate erfolgen
Dokumentation & Ignorieren von False Positives	TEILMANUELL	.trivyignore oder Security-Report

6.6 Zusammenfassung

Vollständige Automatisierung von Container-Sicherheitsupdates ist technisch nicht möglich, da CVE-Datenbanken generische Schwachstellen unabhängig vom Anwendungskontext melden.

Automatisierte Builds (Pull, Update, Scan) liefern nur eine Momentaufnahme. Die tatsächliche Relevanz eines CVEs muss projektbezogen bewertet werden, insbesondere bei eingebetteten oder statisch kompilierten Komponenten.

Eine nachhaltige Lösung besteht darin, die Builds regelmäßig zu automatisieren, Scans zu planen (z. B. via CI/CD) und sicherheitsrelevante CVEs manuell zu triggern.

7 NIS2 & BSI – Relevanz für unser Projekt

Empfehlung	Beschreibung / Ziel laut NIS2 & BSI	Umsetzung im Projekt	Relevanz
Asset Management	Dokumentation aller eingesetzten Systeme, Container und Softwarekomponenten.	Umsetzung durch SBOM-Erstellung mit Syft für Web- und DB-Container.	Erfüllt
Risikomanagement / Schwachstellenscans	Regelmäßige Prüfung der Container auf bekannte CVEs.	Trivy-Scans regelmäßig durchgeführt (Image + SBOM)	Erfüllt
Patch-Management	Zeitnahe Updates sicherheitsrelevanter Pakete.	Im Dockerfile integriert: apt-get update && upgrade -y.	Erfüllt
Minimale Images / Hardening	Nur notwendige Pakete installieren, um Angriffsfläche zu minimieren.	Verwendung von Debian Slim-Images mit entfernten Build-Tools.	Erfüllt
Keine Root-Rechte	Container sollen ohne Root laufen (Least Privilege Principle).		Nicht erforderlich
Sichere Registries / Lieferkettensicherheit	Nutzung vertrauenswürdiger, signierter Images aus offiziellen Quellen.	Nur offizielle Docker Hub Images werden verwendet.	Erfüllt
Secrets Management	Keine Passwörter oder API-Keys im Quellcode oder Dockerfile speichern.	Keine sensiblen Daten im Code; ggf. Nutzung von Docker Secrets möglich.	Erfüllt

Netzwerkisolation	Trennung der Container (z. B. Web ↔ DB) zur Minimierung von Angriffspfaden.	In Docker Compose möglich, für Lernzweck nicht zwingend aktiviert.	Optional
Logging & Monitoring / Incident Response	Überwachung und Reaktion auf Sicherheitsvorfälle.	Nicht notwendig, da kein Dauer- oder Produktivbetrieb.	Nicht erforderlich
Security Governance / Verantwortlichkeiten	Zuweisung einer Sicherheitsrolle oder eines Verantwortlichen.	Nicht relevant, da Lern- und Testprojekt ohne Betriebsteam.	Nicht erforderlich

8 Wie gehen Firmen damit um?

Viele Unternehmen implementieren DevSecOps-Strategien, um Sicherheit in jede Phase der Softwareentwicklung zu integrieren.

Typische Praxis in Unternehmen:

- **CI/CD-Integration:** Automatisierte Scans mit Tools wie Trivy, Anchore oder Snyk bei jedem Build.
- **Security Gates:** Builds mit kritischen CVEs werden automatisch blockiert, bis ein Fix erfolgt.
- **Monitoring:** Nutzung von Security-Dashboards (z. B. GitHub Security, DefectDojo).
- **SBOM-Compliance:** SBOMs werden für Audits und NIS2-Nachweise gespeichert.
- **Container Registry Policies:** Nur signierte und geprüfte Images dürfen deployed werden.
- **Schulung & Awareness:** Entwickler werden in sicherer Container-Konfiguration und DevSecOps geschult.

Best Practice Workflow:

1. Commit im GitHub-Repo
2. CI-Job baut Container → führt Trivy-Scan aus
3. SBOM und Scan-Report werden als Artefakt gespeichert
4. Falls „HIGH“ oder „CRITICAL“ CVEs → Build stoppt
5. Manuelle Bewertung & Freigabe nach Triage

9 Aikido

9.1 Container & Image Scanning Fertiglösungen

Im Zuge des Projekts haben wir eine eigene Plattform (Github Workflow) entwickelt, welche Images scannt und eine Sicherheitsbewertung mit Tabelle erstellt und automatische Security Patches mit Renovate ausführt. Das Ganze ist auch in den CI/CD integriert sobald ein Container erstellt wird aus der GHCR alle Scans angestoßen und bei hohen CVE Werten wird ein Nachricht erstellt.

Parallel dazu haben wir eine Plattform gesucht, welche all diese Features bereits integriert hat und sind dabei auf Aikido.dev gestoßen. Herr Professor Drack hat uns hier eine Premium Lizenz zur Verfügung gestellt, damit wir die Software testen können.

Lizenzen starten bei 300€ / Monat für die Basis Lizenz und bis 900€ / Monat für die Advanced Lizenz. (Lizenz für 10 Benutzer)

Es gebe noch Alternativen wie Snyk, GitHub Advanced Security, Jit welche alle ähnlich im Funktionsumfang sind und sich in einer ähnlichen Preistrange bewegen.

Aikido scannt Container-Images sowohl in der Build-Pipeline (CI/CD) als auch in der Registry (z.B. Docker Hub, AWS ECR, Azure CR).

- **OS-Pakete & Bibliotheken:** Es erkennt veraltete oder unsichere Pakete innerhalb des Images (z.B. eine verwundbare OpenSSL-Version).
- **Infrastructure as Code (IaC):** Aikido prüft die Dockerfiles auf Fehlkonfigurationen, wie das Ausführen als root-User oder das Hinterlassen von Passwörtern in Umgebungsvariablen.
- **Layer-Analyse:** Es zeigt dir genau, in welcher Schicht des Images die Sicherheitslücke eingeführt wurde.

9.2 CVE-Bewertung & Priorisierung

Anstatt den User mit Tausenden kritischen Meldungen allein zu lassen, nutzt Aikido einen **Erreichbarkeits-Check (Reachability Analysis)**.

- **Vulnerability Ranking:** Aikido bewertet CVEs nicht nur nach dem CVSS-Score, sondern prüft, ob die verwundbare Funktion in deinem Code tatsächlich aufgerufen wird.
- **Rauschunterdrückung:** Wenn eine Bibliothek eine schwere Sicherheitslücke hat, dein Code diesen Teil der Bibliothek aber gar nicht nutzt, wird die Priorität herabgestuft. Das spart massiv Zeit.

- **Zentrales Dashboard:** Du erhältst eine klare Liste mit "Must-Fix"-Items statt einer endlosen Liste an theoretischen Risiken.

9.3 Quellcode-Scan (SAST & SCA)

Aikido kombiniert zwei wichtige Scan-Verfahren für den Quellcode:

- **SCA (Software Composition Analysis):** Scannt deine Abhängigkeiten (Open-Source-Libraries) in Dateien wie package.json, pom.xml oder requirements.txt auf bekannte Schwachstellen.
- **SAST (Static Application Security Testing):** Durchleuchtet deinen selbst geschriebenen Code auf typische Fehler wie SQL-Injection, Cross-Site Scripting (XSS) oder hartcodierte Secrets (API-Keys).
- **Secrets Detection:** Sucht aktiv nach vergessenen Zugangsdaten im Repository, bevor diese in die Produktion gelangen.

9.4 Aikido in der Praxis:

Man fügt die Registry URL von seinen Containern hinzu. Die GHCR muss dabei public sein oder man hinterlegt einen Zugriffstoken für das Repository bei Aikido.

Containers 4 Containers Start Scan

Containers Checks

Q Search All Types All Distros Connect Registry Add Public image Actions

Type	Name	Registry	Domain	Severity	Open	Last scan	
	drackthor2/sandbox:2.0	drackthor2	Configure	Critical	27	Upgrade	
	drackthor2/node-as-code-demo:1.0.11	drackthor2	Configure	Critical	34	Upgrade	
	ghcr.io/kjeel/php_web	Public image	Configure	High	10	Upgrade	
	ghcr.io/kjeel/php_db:db	Public image	Configure	No Risk	0	No scan yet	

In einer Liste findet man alle bekannten CVE Einträge zu diesem Image.

ghcr.io/kjeel/php_web 10 issues Configure Github Public Debian 13 latest Start scan

Issues Checks Settings Last scan 2 weeks ago

Q Search All types Actions

Type	Name	Severity	Location	Fix time	Status
	symfony/http-foundation Attacker can abuse improper authorization	High	ghcr.io/kjeel/php_web	30 min	To Do
	openssl-provider-legacy Out-of-bounds read can expose sensitive info	High	ghcr.io/kjeel/php_web	15 min	To Do
	openssl Out-of-bounds read can expose sensitive info	High	ghcr.io/kjeel/php_web	15 min	To Do
	libssl3t64 Out-of-bounds read can expose sensitive info	High	ghcr.io/kjeel/php_web	15 min	To Do
	libcurl4t64	High	ghcr.io/kjeel/php_web	15 min	To Do
	libssl3t64	High	ghcr.io/kjeel/php_web	15 min	To Do
	libcurl4t64	High	ghcr.io/kjeel/php_web	15 min	To Do
	libssl3t64	High	ghcr.io/kjeel/php_web	15 min	To Do
	libcurl4t64	Medium	ghcr.io/kjeel/php_web	30 min	To Do

Hier steht bei jedem Eintrag bereits die geschätzte Zeit, die es dauert um das Problem zu beheben. Mit einem Klick auf den Eintrag kommt man auf die erweiterte Ansicht, die wir leider nicht mehr herzeigen können, da die Lizenz bereits abgelaufen ist.

Hier stehen Tipps wie man den Fehler behebt, z.B. mit einem Update oder sollte irgendeine Library verwendet werden und es gibt noch kein Update schlägt er Aikido auch alternative Implementierungsmöglichkeiten vor.

Actions

81
High Risk

symfony/http-foundation
Affected by 2 CVEs last detected 2 weeks ago

No fix Dependency

Overview Activity Tasks

TLDR
symfony/http-foundation is affected by 2 CVEs. To learn more about each one, consult the table below. The worst case impact is an attacker can abuse improper authorization.

How do I fix it?
In order to fix all of these CVEs, you need to upgrade one of a time but.

Subissues

Subissue	Fix
ghcr.io/kjeel/php_web	
Aikido-2025-10807 CVE-2025-64500 /var/www/html/vendor/composer/installed.json	v6.4.26 → 6.4.29
CVE-2025-64500 /var/www/html/vendor/composer/installed.json	v6.4.26 → 6.4.29




⚠️ Upgraded: Exploit available on GitHub

← → View Detail

Zusätzlich kann Code gescannt werden, um Verbesserungsvorschläge anzuzeigen, Dependencies, als auch IaC Code überprüft werden. Sollten z.B. Secrets in einem IaC Code sein würde eine Warnung kommen, oder eine veraltete Library usw.

All diese Funktionen lassen sich in den CI/CD Prozess einbauen, um sicherzustellen, dass sowohl der Code sicher ist als auch die verwendeten Images.

All Teams > AutoFix

Q Docs   

AutoFix

Fix issues with Aikido's AI agent. Generate pull requests to fix SAST, IaC, and dependency issues.

0 / 427 hours saved with AutoFix

Dependencies 47

SAST 9

IaC 255

Containers

Settings [Enable Now](#)

Q Search

Extended Lifecycle Support

Infrastructure-as-code-goof | helm/hack/requirements.txt [Create PR](#)

<input type="checkbox"/>	Name	Severity	CVE	Version Upgrade	Status
<input type="checkbox"/>	urlib3 This package has 7 vulnerabilities	High	CVE-2023-43804	1.25.11 → 2.6.0 major 1.25.11 → 1.26.4 (ELS)	To Do
<input type="checkbox"/>	requests This package has 3 vulnerabilities	Medium	CVE-2023-32681	2.22.0 → 2.32.4 minor	To Do
<input type="checkbox"/>	idna Attacker can trigger DOS-attack	Low	CVE-2024-3651	2.8 → 3.7 major 2.8 → 2.8 (ELS)	To Do