Cert-IX

MCP-SERVER · ABHÄNGIGKEITSSICHERHEIT

DepCheck MCP

Benutzerhandbuch

Alles, was du brauchst, um DepCheck mit deinem KI-Coding-Agenten zu verbinden und Abhängigkeiten auf bekannte Schwachstellen zu prüfen, bevor sie in deinem Code landen.

Kostenloses Entwickler-Tool · Model Context Protocol

ENDPUNKT:
`https://mcp.cert-ix.com/depcheck`

Inhalt

1 Überblick

Alles, was du brauchst, um DepCheck mit deinem KI-Coding-Agenten zu verbinden und Abhängigkeiten auf bekannte Schwachstellen zu prüfen, bevor sie in deinem Code landen.

- **Endpoint:** `https://mcp.cert-ix.com/depcheck`
- **Transport:** streamable HTTP (MCP)
- **OSV · CISA KEV · EPSS**

2 Verbinde deinen Agenten

Fordere zunächst einen kostenlosen Schlüssel unter cert-ix.com/contact an — wir senden dir einen Schlüssel der Form cixmcp_… (kein Konto nötig). Richte dann deinen Agenten mit diesem Schlüssel auf den Endpunkt aus.

Claude Code

```
claude mcp add --transport http depcheck https://mcp.cert-ix.com/depcheck \
  --header "Authorization: Bearer <your-key>"
```

Beliebiger MCP-Client (Konfigurationsdatei)

```
{
  "mcpServers": {
    "depcheck": {
      "type": "http",
      "url": "https://mcp.cert-ix.com/depcheck",
      "headers": { "Authorization": "Bearer <your-key>" }
    }
  }
}
```

> Funktioniert mit Claude Code und jedem Client, der MCP über HTTP spricht. Sobald dein Schlüssel aktiv ist, kann dein Agent die Werkzeuge sofort aufrufen.

3 Die fünf Werkzeuge

`check_package`

Argumente: `ecosystem` `name` `version`

Prüfe eine einzelne Paketversion auf bekannte Schwachstellen. Gibt die Advisories zurück, die genau diese Version betreffen, sowie die nächstgelegene sichere Version, auf die aktualisiert werden kann.

`scan_dependencies`

Argumente: `manifest_content` `manifest_name`

Scanne ein ganzes Manifest auf einmal — go.mod, package.json, package-lock.json, requirements.txt, pyproject.toml, Cargo.toml, Cargo.lock, pom.xml, composer.json. Der gehostete Server nimmt den Dateiinhalt entgegen, niemals einen Pfad. Bis zu 2000 Abhängigkeiten pro Scan.

`suggest_safe_version`

Argumente: `ecosystem` `name`

Erhalte die neueste Version eines Pakets, für die keine bekannten Advisories vorliegen.

`get_advisory`

Argumente: `id`

Rufe alle Details eines beliebigen Advisorys ab (GHSA-…, CVE-…, RUSTSEC-…, GO-…): Beschreibung, CVSS, betroffene Bereiche und Referenzen.

`get_cve_intel`

Argumente: `id`

Sieh dir die Ausnutzungsinformationen zu einem CVE an — CISA-KEV-Status und EPSS-Wert — um zu entscheiden, was zuerst behoben werden sollte. Eine in KEV gelistete oder EPSS-hohe Schwachstelle wird aktiv angegriffen und sollte in der Warteschlange vorgezogen werden.

4 Ein verdict lesen

Feld	Bedeutung
`verdict`	VULNERABLE oder OK für die genau geprüfte Version.
`advisories`	Jeweils mit einer id, einem Schweregrad und einem CVSS-Wert.
`exploitation`	Markiert, wenn die Schwachstelle in CISA KEV steht oder einen hohen EPSS-Wert hat — behebe diese zuerst.
`recommended_min_version`	Die nächstgelegene Version über der aktuellen, die die Advisories beseitigt.

5 Wie deine Daten behandelt werden

- Nichts wird gespeichert. Dein Manifest wird im Arbeitsspeicher geparst, um Paketnamen und Versionen zu extrahieren, und danach verworfen — wir behalten weder deine Dateien noch deine Ergebnisse.
- Nur Koordinaten werden geprüft — Paketname und Version, niemals der Inhalt deiner Dateien. Nachschlagen wird zur Beschleunigung kurz im Arbeitsspeicher zwischengespeichert.
- Souveräner Spiegel zuerst. Prüfungen laufen gegen unseren eigenen Advisory-Spiegel, der in EU-Infrastruktur gehostet wird, sodass die meisten Abfragen Cert-IX nie verlassen.

> Fallback-Ausgang: Wenn der Spiegel ein Paket nicht mit Sicherheit beantworten kann, werden Name und Version dieses Pakets über die öffentliche OSV.dev-API (betrieben von OpenSSF) nachgeschlagen. Auf diesem Pfad können also Paket-Koordinaten — nicht Dateiinhalte — unsere Infrastruktur verlassen. Behalte das im Hinterkopf, wenn du private oder interne Paketnamen prüfst.

6 Faire Nutzung & Grenzen

- Anfragen sind pro Schlüssel ratenbegrenzt (etwa 20 pro Sekunde). Ein vollständiger Manifest-Scan ist eine einzelne Anfrage.
- Halte deinen Schlüssel privat — ein Schlüssel pro Team oder Benutzer. Zum Rotieren fordere einen neuen an, und wir ziehen den alten zurück.

7 Fehlerbehebung

Antwort	Was es bedeutet
`401 unauthorized`	Fehlender oder ungültiger Schlüssel — prüfe deinen Authorization: Bearer-Header.
`429 too many requests`	Du hast das Ratenlimit erreicht — warte kurz und versuche es erneut.
`scan timed out`	Ein sehr großes oder stark advisory-behaftetes Manifest — teile es in kleinere Stücke auf.

8 Brauchst du einen Schlüssel oder Hilfe?

Fordere einen Schlüssel an, stelle eine Frage oder melde ein Problem — wir helfen gerne.

Kontaktiere uns: cert-ix.com/contact · https://mcp.cert-ix.com/depcheck