Grype Vulnerability scanner

CONCEPTS

Grype matches the packages it finds (or reads from an SBOM) against a local vulnerability DB. It reports; it does not generate SBOMs (that is Syft).

Source image, dir, SBOM, PURL
↓ `catalog`
Packages OS and language deps
↓ `match`
Vuln DB local, auto-updated
↓ `filter`
Report table, json, sarif, cyclonedx

A bare argument is auto-detected (local daemon, then registry, then path). Add a scheme prefix or --from when the guess is wrong.

SCAN SOURCES

`grype <image>`
Scan an image (name:tag)

`grype dir:<path>`
Scan a directory

`grype sbom:<file>`
Scan a Syft/SPDX/CycloneDX SBOM

`cat <sbom.json> | grype`
Read an SBOM from stdin

`grype docker:<image>`
Force the local Docker daemon

`grype registry:<image>`
Force a pull from the registry

`grype docker-archive:<file.tar>`
Scan a docker save tarball

`grype pkg:<purl>`
Check one package URL

`--platform <os/arch>`
Pick the image platform (linux/arm64)

`--scope all-layers`
Include layers hidden by later ones

FILTERING

`--only-fixed`
Show only vulns with a fix available

`--ignore-states <list>`
Drop fix states: fixed,not-fixed,unknown,wont-fix

OUTPUT AND CI

`--fail-on <severity>`
Exit code 2 at or above this severity

`-o table`
Human-readable table (default)

`-o json`
Full JSON report

`-o sarif`
SARIF for code-scanning tools

`-o json=<file>`
Write one format to a file

`-q`
Quiet: no log output

IGNORE RULES AND CONFIG

`.grype.yaml`
Config file in the project root

`ignore: - vulnerability: <CVE>`
Suppress one CVE

`ignore: package: name: <regex>`
Suppress by package name

DATABASE

`grype db update`
Download the latest database

`grype db check`
Is a newer database available

`grype db status`
Show the cached DB's age and validity

`grype db import <file|url>`
Load a DB archive (air-gapped)

`GRYPE_DB_AUTO_UPDATE=false`
Do not fetch updates during a scan