Grype Vulnerability scanner

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

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).

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

SCAN SOURCES (cont.)

`grype podman:<image>`
Use the Podman daemon

`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

`--exclude <glob>`
Skip matching paths (repeatable)

FILTERING

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

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

`--vex <file>`
Apply a VEX document (repeatable)

`--by-cve`
Report by CVE ID, not GHSA ID

FILTERING (cont.)

`--sort-by <key>`
Order: risk, severity, epss, kev, package

`--ignore-states wont-fix`
Skip issues the vendor will not fix

`grype explain --id <CVE>`
Explain why a CVE matched

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 cyclonedx-json`
CycloneDX report with vulns

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

`--file <path>`
Write the report to a file

`-q`
Quiet: no log output

`-v / -vv`
Info / debug logging

OUTPUT AND CI (cont.)

`grype --version`
Print the version

IGNORE RULES AND CONFIG

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

`-c <file>`
Use a specific config file

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

`ignore: fix-state: not-fixed`
Suppress by fix state

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

`ignore: reason: <text>`
Record why it is ignored

`fail-on-severity: high`
Config form of --fail-on

`GRYPE_<KEY>=<value>`
Set any option via environment

DATABASE

`grype db update`
Download the latest database

DATABASE (cont.)

`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 search --vuln <ID>`
Look up a vulnerability in the DB

`grype db search --pkg <name>`
Find vulns for a package, CPE or PURL

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

RECIPES

Gate CI on fixable high-severity issues

```
grype <image> --only-fixed
--fail-on high
```

Scan once, report as SARIF and JSON

```
grype <image> -o
sarif=grype.sarif -o
json=grype.json
```