Trivy Security scanner

START HERE

`trivy image <image>`
Scan a container image

`--exit-code 1`
Fail the CI job on findings

`trivy fs .`
Scan a project folder

`-f json -o out.json`
Write a JSON report

`trivy config .`
Check IaC for misconfigs

`trivy repo <url>`
Scan a git repository

`--severity HIGH,CRITICAL`
Show only serious findings

`--ignore-unfixed`
Hide CVEs without a fix

Target image, fs, repo, k8s → *analyze* → **Scanners** vuln, secret, misconfig, license → *match* → **Databases** cached vuln DB and checks → *filter* → **Report** table, json, sarif, SBOM

CONCEPTS

Pick the target type (image, fs, repo, config, k8s, sbom), then choose what to look for with --scanners.

Defaults: image, fs and repo run vuln + secret; config runs misconfig only. License scanning is off unless requested.

The first run downloads the vulnerability DB into the cache; later runs reuse it, so offline CI needs a pre-warmed cache.

SCAN TARGETS

`trivy image <image>`
Scan an image (local daemon, then registry)

`trivy image --input <file.tar>`
Scan an image saved as tar
Create the tar with docker save <image> -o file.tar.

`trivy fs <dir>`
Scan a local project, incl. lock files

`trivy repo <path|url>`
Scan a local or remote git repo

`trivy repo --branch <name> <url>`
Scan one branch of a remote repo

SCAN TARGETS (cont.)

`trivy repo --tag <tag> <url>`
Scan one tag of a remote repo

`trivy repo --commit <sha> <url>`
Scan one commit of a remote repo

`trivy rootfs <dir>`
Scan an extracted root filesystem

`trivy sbom <file>`
Scan an existing SBOM for vulns

`trivy image --image-src <list>`
Pick image sources, e.g. docker,podman

`trivy image --platform <os/arch>`
Scan a non-default platform

`trivy image --docker-host <host>`
Use a specific Docker daemon socket

`trivy registry login <registry>`
Store registry credentials

SCANNERS

`--scanners vuln`
Only known vulnerabilities

`--scanners vuln,secret,misconfig`
Combine several scanners

SCANNERS (cont.)

`--scanners misconfig`
IaC and Dockerfile misconfigurations

`--scanners secret`
Hard-coded secrets only

`--scanners license`
Report package licenses

`trivy config <dir>`
Misconfig scan of Terraform, K8s, Helm, Dockerfile

`--image-config-scanners misconfig`
Check the image's own config

`--pkg-types os`
Only OS packages, skip libraries

`--pkg-types library`
Only app dependencies, skip OS

FILTERING

`--severity <list>`
Keep only these severity levels

`--ignore-unfixed`
Skip vulns that have no fix yet

`--ignore-status <list>`
Hide statuses like affected,will_not_fix

`.trivyignore`
One finding ID per line; # comments
Add exp:YYYY-MM-DD after an ID to make the exception expire.

FILTERING (cont.)

`--ignorefile <file>`
Use another ignore file (.trivyignore.yaml)

`--skip-dirs <glob>`
Do not scan matching directories

`--skip-files <glob>`
Do not scan matching files

OUTPUT AND CI

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

`-f json -o <file>`
Write JSON to a file

`-f sarif -o <file>`
SARIF for code-scanning tools

`-f template --template "@contrib/html.tpl"`
Render HTML via a template

`--exit-code <n>`
Exit with n when issues are found
Default is 0, so scans never fail a pipeline unless you set this.

`--dependency-tree`
Show which dependency pulls in a vuln

`trivy convert -f <fmt> <result.json>`
Re-render a saved JSON report

`-q`
Quiet: no progress bar or log noise

OUTPUT AND CI (cont.)

`--no-progress`
Hide the progress bar only

`-d`
Debug logging

`--timeout <dur>`
Raise the 5m default for big scans

SBOM

`-f cyclonedx -o <file>`
Generate a CycloneDX SBOM
CycloneDX output skips vulnerabilities unless you add --scanners vuln.

`-f spdx-json -o <file>`
Generate an SPDX JSON SBOM

KUBERNETES

`trivy k8s --report summary`
Cluster overview (default report)

`trivy k8s --report all`
Detailed findings for the cluster

`trivy k8s --scanners misconfig`
Only misconfigurations in the cluster

`--include-namespaces <list>`
Limit to these namespaces

`--compliance <id>`
Check a spec, e.g. k8s-cis-1.23

DATABASE AND CACHE

`--skip-db-update`
Use cached vuln DB, no download

`--download-db-only`
Fetch the DB, do not scan

`--db-repository <registry/repo>`
Pull the DB from a mirror

`--cache-dir <dir>`
Move the cache (CI caching)

`trivy clean --all`
Delete all cached data

RECIPES

Gate a CI build on serious, fixable issues

```
trivy image --severity
HIGH,CRITICAL \
  --ignore-unfixed --exit-code
1 <image>
```

Project audit with a JSON report

```
trivy fs --scanners
vuln,secret,misconfig \
  -f json -o report.json .
```

Build an image and scan it before pushing

```
docker build -t <image> .
trivy image --exit-code 1 --
severity CRITICAL <image>
docker push <image>
```

Ship an SBOM and re-scan it later

```
trivy image -f cyclonedx -o
sbom.cdx.json <image>
trivy sbom sbom.cdx.json
```