# Scan Docker Hardened Images


- Scan with OpenVEX-compliant scanners
- Docker Scout
  - Scan a DHI using Docker Scout
  - Build child images with provenance attestations
  - VEX attestations in child images
  - Automate DHI scanning in CI/CD with Docker Scout
    - Example GitHub Actions workflow
- Grype
  - Scan a DHI using Grype
- Trivy
  - Scan a DHI using Trivy
    - Using VEX Hub (recommended)
    - Using local VEX files
- Export VEX attestations
- Explore the VEX file


Docker Hardened Images (DHIs) are designed to be secure by default, but like any container image, it's important to scan them regularly as part of your vulnerability management process.

## Scan with OpenVEX-compliant scanners

To get accurate vulnerability assessments, use scanners that support VEX attestations. The following scanners can read and apply the VEX statements included with Docker Hardened Images:

- Docker Scout: Automatically applies VEX statements with zero configuration
- Trivy: Supports VEX through VEX Hub or local VEX files
- Grype: Supports VEX via the `--vex` flag

For a full list of supported scanners, see Scanner integrations.

# Docker Scout

Docker Scout is integrated into Docker Desktop and the Docker CLI. It provides vulnerability insights, CVE summaries, and direct links to remediation guidance.

## Scan a DHI using Docker Scout

To scan a Docker Hardened Image using Docker Scout, run the following command:

```
$ docker login dhi.io
$ docker scout cves dhi.io/<image>:<tag> --platform <platform>
```

Example output:

```
    v SBOM obtained from attestation, 101 packages found
    v Provenance obtained from attestation
    v VEX statements obtained from attestation
    v No vulnerable package detected
    ...
```

For more detailed filtering and JSON output, see Docker Scout CLI reference.

## Build child images with provenance attestations

When you build a custom image that uses a Docker Hardened Image as its base, you must build with `--provenance=mode=max` and `--sbom=true` so that Docker Scout can trace the base image lineage and correctly apply VEX statements.

Without these flags, Docker Scout cannot identify the DHI base image in the provenance chain. As a result, it reports CVEs that are already suppressed by VEX statements in the base image,

producing false CVE positives in your scan results.

> **Why provenance attestation is required**
>
> Docker Scout uses max-mode provenance attestations to identify the DHI base image and track its lineage. A cryptographically signed provenance attestation ensures that base image lineage is verified and tamper-resistant, giving Docker Scout the trust anchor it needs to correctly apply VEX statements from the base image.

To build with maximum provenance and SBOM attestations:

```
$ docker build \
    --provenance=mode=max \
    --sbom=true \
    --push \
    -t docker.io/<namespace>/<image>:<tag> .
```

After building with these flags, Docker Scout reads the full provenance chain, matches the DHI base image, and applies its VEX statements. Scans of your child image then reflect the correct suppressed CVEs, giving you an accurate vulnerability assessment.

# VEX attestations in child images

If you introduce new layers in your child image and want to suppress CVEs in those layers, you can attach your own VEX attestation to the child image independently, you do not need to duplicate or aggregate the VEX statements from the DHI base image.

When `docker scout cves` runs against your child image, Scout reads VEX attestations from the full provenance chain and applies them cumulatively:

- **Base image VEX** - attached to the DHI, applied to CVEs in base image layers
- **Child image VEX** - attached to your image, applied to CVEs in layers you introduced

For example, if you add a `requests` layer to a DHI Python base image and attach a VEX statement suppressing `CVE-2024-47081`, Scout applies both VEX attestations independently and attributes each to its respective author:

```
✓ VEX statements obtained from attestation
CVE-2024-47081  VEX: not affected [vulnerable code not present] : <your-namespace>
```

Scout suppresses CVEs from the DHI base VEX and CVEs from your child VEX in the same scan - no aggregate VEX document is required.

To create and attach a VEX attestation to your child image:

```
cat > child-vex.json << 'EOF'
{
  "@context": "https://openvex.dev/ns/v0.2.0",
  "@id": "https://<your-namespace>/vex/<image-name>/1",
  "author": "<your-namespace>",
  "timestamp": "<timestamp>",
  "version": 1,
  "statements": [
    {
      "vulnerability": {
        "name": "<CVE-ID>"
      },
      "products": [
        {
          "@id": "pkg:pypi/<package>@<version>"
        }
      ],
      "status": "not_affected",
      "justification": "vulnerable_code_not_present"
    }
  ]
}
EOF

docker scout attestation add \
  --file child-vex.json \
  --predicate-type https://openvex.dev/ns/v0.2.0 \
  docker.io/<your-namespace>/<image>:<tag>
```

> This is only possible because you built with `--provenance=mode=max`. Without the full provenance chain, Scout cannot traverse back to the base image to retrieve its VEX attestations.

# Automate DHI scanning in CI/CD with Docker Scout

Integrating Docker Scout into your CI/CD pipeline enables you to automatically verify that images built from Docker Hardened Images remain free from known vulnerabilities during the build process. This proactive approach ensures the continued security integrity of your images throughout the development lifecycle.

## Example GitHub Actions workflow

The following is a sample GitHub Actions workflow that builds an image, scans it and pushes to the registry only if the scan passes:

```
name: DHI Vulnerability Scan

on:
  push:
    branches:
      - main
  pull_request:

env:
  REGISTRY: docker.io
  IMAGE_NAME: ${{ github.repository }}
  SHA: ${{ github.event.pull_request.head.sha || github.event.after }}

jobs:
  scan:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write
      pull-requests: write
    steps:
      - name: Checkout repository
        uses: actions/checkout@{{% param "checkout_action_version" %}}

      - name: Set up Docker with containerd image store
        uses: docker/setup-docker-action@{{% param "setup_docker_action_version" %}}
        with:
          daemon-config: |
            {
              "features": {
                 "containerd-snapshotter": true
```

```
          }
        }

    - name: Log in to Docker Hub
      uses: docker/login-action@{{% param "login_action_version" %}}
      with:
        registry: ${{ env.REGISTRY }}
        username: ${{ secrets.DOCKER_USERNAME }}
        password: ${{ secrets.DOCKER_PASSWORD }}

    - name: Build
      uses: docker/build-push-action@{{% param "build_push_action_version" %}}
      with:
        context: .
        sbom: true
        tags: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}:${{ env.SHA }}

    - name: Run Docker Scout CVE scan
      uses: docker/scout-action@v1
      with:
        command: cves
        image: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}:${{ env.SHA }}
        only-severities: critical,high
        exit-code: true

    - name: Push image
      if: success()
      run: |
        docker push ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}:${{ env.SHA }}
```

The `exit-code: true` parameter ensures that the workflow fails if any critical or high-severity vulnerabilities are detected, preventing the deployment of insecure images.

> The `--provenance=mode=max` and `--sbom=true` flags are required so that Docker Scout can trace the DHI base image lineage and correctly apply its VEX statements. Enabling the containerd image store via `docker/setup-docker-action` allows BuildKit to store attestations locally without pushing to a registry first. Without the containerd image store, Docker Engine rejects the build with: `Attestation is not supported for the docker driver. Switch to a different driver, or turn on the containerd image store, and try again.` The `Push image` step runs only if the scan passes, using `if: success()` to ensure images are only pushed to the

registry when they are free of critical or high-severity vulnerabilities.

For more details on using Docker Scout in CI, see Integrating Docker Scout with other systems.

# Grype

Grype is an open-source scanner that checks container images against vulnerability databases like the NVD and distro advisories.

## Scan a DHI using Grype

To scan a Docker Hardened Image using Grype with VEX filtering, first export the VEX attestation and then scan with the `--vex` flag:

```
$ docker login dhi.io
$ docker pull dhi.io/<image>:<tag>
$ docker scout vex get dhi.io/<image>:<tag> --output vex.json
$ grype dhi.io/<image>:<tag> --vex vex.json
```

The `--vex` flag applies VEX statements during the scan, filtering out known non-exploitable CVEs for accurate results.

For more information on exporting VEX attestations, see Export VEX attestations.

# Trivy

Trivy is an open-source vulnerability scanner for containers and other artifacts. It detects vulnerabilities in OS packages and application dependencies.

## Scan a DHI using Trivy

After installing Trivy, you can scan a Docker Hardened Image by pulling the image and running the scan command:

```
$ docker login dhi.io
$ docker pull dhi.io/<image>:<tag>
```

```
$ trivy image --scanners vuln dhi.io/<image>:<tag>
```

To filter vulnerabilities using VEX statements, Trivy supports multiple approaches. Docker recommends using VEX Hub, which provides a seamless workflow for automatically downloading and applying VEX statements from configured repositories.

## Using VEX Hub (recommended)

Configure Trivy to download the Docker Hardened Images advisories repository from VEX Hub. Run the following commands to set up the VEX repository:

```
$ trivy vex repo init
$ cat << REPO > ~/.trivy/vex/repository.yaml
repositories:
  - name: default
    url: https://github.com/aquasecurity/vexhub
    enabled: true
    username: ""
    password: ""
    token: ""
  - name: dhi-vex
    url: https://github.com/docker-hardened-images/advisories
    enabled: true
REPO
$ trivy vex repo list
$ trivy vex repo download
```

After setting up VEX Hub, you can scan a Docker Hardened Image with VEX filtering:

```
$ docker login dhi.io
$ docker pull dhi.io/<image>:<tag>
$ trivy image --scanners vuln --vex repo dhi.io/<image>:<tag>
```

For example, scanning the `dhi.io/python:3.13` image:

```
$ trivy image --scanners vuln --vex repo dhi.io/python:3.13
```

Example output:

```
Report Summary

┌
```

```
──────────────┐
│                                   Target                                    │    Type    │
Vulnerabilities │
├─────────────────────────────────────────────────────────────────────────────┼────────────┼─
──────────────┤
│ dhi.io/python:3.13 (debian 13.2)                                            │   debian
│       0       │
├─────────────────────────────────────────────────────────────────────────────┼────────────┼─
──────────────┤
│ opt/python-3.13.11/lib/python3.13/site-packages/pip-25.3.dist-info/METADATA │ python-pkg
│       0       │
└─────────────────────────────────────────────────────────────────────────────┴────────────┴──
──────────────┘

Legend:
- '-': Not scanned
- '0': Clean (no security findings detected)
```

The `--vex repo` flag applies VEX statements from the configured repository during the scan, which filters out known non-exploitable CVEs.

## Using local VEX files

In addition to VEX Hub, Trivy also supports the use of local VEX files for vulnerability filtering. You can download the VEX attestation that Docker Hardened Images provide and use it directly with Trivy.

First, download the VEX attestation for your image:

```
$ docker scout vex get dhi.io/<image>:<tag> --output vex.json
```

Then scan the image with the local VEX file:

```
$ trivy image --scanners vuln --vex vex.json dhi.io/<image>:<tag>
```

# Export VEX attestations

For scanners that need local VEX files (like Grype or Trivy with local files), you can export the VEX attestations from Docker Hardened Images.

> By default, VEX attestations are fetched from `registry.scout.docker.com`. Ensure that you can access this registry if your network has outbound restrictions. You can also mirror the attestations to an alternate registry. For more details, see Mirror to a third-party registry.

Export VEX attestations to a JSON file:

```
$ docker scout vex get dhi.io/<image>:<tag> --output vex.json
```

> The `docker scout vex get` command requires Docker Scout CLI version 1.18.3 or later.
>
> If the image exists locally on your device, you must prefix the image name with `registry://`. For example, use `registry://docs/dhi-python:3.13` instead of `docs/dhi-python:3.13`.

# Explore the VEX file

After exporting the VEX attestation, you can inspect its contents to understand what each statement means and filter by status. For a guided walkthrough of the VEX document structure, status values, and `jq` filtering commands, see Explore VEX statements in Docker Hardened Images .

---

□□□ #2
□ □□□ □□□ 2026-10-31 19:17:34 UTC
□ □□□ □□□ 2026-10-31 19:17:36 UTC