Scan and tag container images with Wiz CLI

When using Wiz CLI, this is the recommended integration into the CI/CD flow of Docker images creation:

1. During the CI phase, changes to the code trigger a build that creates the Docker image.
2. Use the `wizcli docker scan` command to analyze the binaries and packages of Docker images, detect vulnerabilities, sensitive data, and secrets, and collect metadata. A scan either passes or fails (as indicated by the exit code) based on the selected CI/CD policies. This automates image scanning as part of your CI/CD pipeline and lets you block vulnerable images from ever reaching the registry without slowing down their development process.
3. The Docker image is pushed to the registry.
4. Use the `wizcli docker tag` command to fetch the digest of the Docker image and pull the metadata to the Wiz backend (for later Graph enrichment).
5. The Docker image is deployed.
6. Wiz scans the deployed Docker image in runtime and enriches its graph object with the CI metadata.

Running a container image scan

Scans can be performed using either tarball, Podman, Docker, or Docker daemon.

The prerequisites are:

- Wiz CLI downloaded and authenticated, including a service account with the appropriate permissions (for authentication via service account only).
- (For tarball) The image name or the path to the image file (.tar, .tar.gz, or .tgz).
- (For Docker) The Docker images must exist locally. You can use `docker image list` to verify.

> ⚠ On Windows hosts, Wiz CLI
>
> - Supports only Windows container image scanning.

- Does not support Linux container image scanning. To scan these, you can run Wiz CLI as a container.

Usage

To view the help page, enter `wizcli docker scan --help`.

Help

```
Scans a local Docker image

Usage:
  wizcli docker scan [flags]

Flags:
      --dockerfile string            Dockerfile file used to build the image
  -d, --driver string                Driver used to scan image. Supported optior
  -f, --format string                Scan's output format. Available options are
      --group-by string              Scan's output grouping field. Available opt
  -h, --help                         help for scan
  -i, --image string                 Image name, including the tag/digest, or pa
      --no-dotnet-binary-scanning    Disable scanning of .NET binaries (.dll and
                                     Binary file scanning spawns an additional .
      --no-publish                   Disable publishing scan results to portal
  -o, --output file-outputs          Output to file, can be passed multiple time
                                     Must be specified in the following format:
                                     Options for file-format: [csv-zip, human, j
  -p, --policy strings               Scan policy to use. Can be passed multiple
      --policy-hits-only             Only display results that failed the applie
      --project string               Scan's scoped project UUID. If project is c
      --secrets                      Scan secrets (default true)
      --sensitive-data               Find sensitive data like PII, PCI and PHI,
      --show-secret-snippets         Enable snippets in secrets
      --show-vulnerability-details   Show vulnerability descriptions and CVSS me
  -t, --tag tags                     Tags to mark the scan with, can be KEY or K
      --timeout string               Operation timeout (default "1h0m0s")

Global Flags:
      --log string     File path at which to write debug logs (defaults to no lo
  -C, --no-color       Disable color output
  -S, --no-style       Disable stylized output
  -T, --no-telemetry   Disable telemetry
```

- All command arguments are case sensitive, so `-p your-policy` is different from `-p Your-Policy`.

- If your policy name contains spaces, ensure it is wrapped in quotes. For example:
  `wizcli iac scan --path path --policy "my policy name"`

- When using the `--policy-hits-only` flag, results that failed audit policies are not shown.
- When using the `--output` flag, do not use square brackets. For example:
  `--output /pathToOutput/my_results.zip,csv-zip,true,layer`

Scan drivers

Wiz CLI allows you to use several scanning drivers, based on your environment, by adding the `-d/ --driver` flag.

	extract	mount	mountWithLayers
Description	The default driver option. Suitable for most hosts.	Faster than the extract option as there is no need to unpack the image.	Similar to the mount method. In this option, scanning is done layer by layer.
How it works	Wiz CLI saves the image on the file system, unpacks it, and then scans it.	Wiz CLI mounts the image on the file system and then scans it. It leverages the runtime storage engine to mount a read-only view of the layers/images, in the same manner as when the runtime starts a new container.	Same as mount.
Limitations	Scan time might be long, as Wiz CLI saves and unpacks the image on the file system	• On operating systems that don't run Docker natively, running on a remote docker daemon is required • Requires high permissions (sudo)	Same as mount.

Working with mount/mountWithLayers drivers

- Running with a local Docker daemon–The local Unix/domain socket (usually /var/run/docker.sock under Linux) or named piped (\.\pipe\docker_engine under Windows) must be accessible.
- Running with a remote Docker daemon–Wiz CLI should be run as a container on the container runtime in which the scanned image is at.

Example commands

Basic example command

If you are running Wiz CLI as a Docker image, be sure to use the relevant command:

Pick your system Linux, macOS or Windows Docker image

Example command with per-layer vulnerability assessment

Wiz CLI can perform a per-layer analysis of container images, allowing you to map and correlate each detected CVE on a container image with the specific image layer where the vulnerability was introduced. The scan results output displays the layer build command that introduced the vulnerability and flags base-image vulnerabilities.

> Per-layer vulnerability assessment is currently supported on Linux and Windows only. Per-layer vulnerability assessment support on macOS is available when running Wiz CLI as a container.
>
> Windows containers should be scanned on Windows hosts, whereas Linux containers should be scanned on Linux hosts.

To perform per-layer analysis, add the `--driver mountWithLayers` flag:

Example command Linux Docker image Example command Windows

```
sudo wizcli docker scan --image mongo:1.2.3-ab1 --driver mountWithLayers
```

```
➜  cli git:(develop) ✗ sudo ./wizcli docker scan --image postgres:15.3 --driver mountWithLayers
 __      _(_)_____   ___| (_)
 \ \ /\ / / |_  /  / __| | |
  \ V  V /| |/ /  | (__| | |
   \_/\_/ |_/___|  \___|_|_|
 SUCCESS  Ready to scan Docker image postgres:15.3
 SUCCESS  Scanned Docker image
 SUCCESS  Docker image scan analysis ready
Layer build command: 'ADD file:88252a7f118b4d6f55dd5baf49dbcaa053c9d6172c652963c1151fa76f625e44 in /' (introduced by base image)
    OS Package vulnerabilities:
        Name: libext2fs2, Version: 1.46.2-2
            CVE-2022-1304, Severity: MEDIUM, Source: https://security-tracker.debian.org/tracker/CVE-2022-1304
        Name: libtinfo6, Version: 6.2+20201114-2+deb11u1
            CVE-2023-29491, Severity: MEDIUM, Source: https://security-tracker.debian.org/tracker/CVE-2023-29491
        Name: libmount1, Version: 2.36.1-8+deb11u1
            CVE-2022-0563, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2022-0563
Layer build command: 'set -ex; apt-get update; apt-get install -y --no-install-recommends gnupg ; rm -rf /var/lib/apt/lists/*'
    OS Package vulnerabilities:
        Name: libldap-2.4-2, Version: 2.4.57+dfsg-3+deb11u1
            CVE-2023-2953, Severity: MEDIUM, Source: https://security-tracker.debian.org/tracker/CVE-2023-2953
            CVE-2015-3276, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2015-3276
            CVE-2017-14159, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2017-14159
            CVE-2017-17740, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2017-17740
            CVE-2020-15719, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2020-15719
        Name: gnupg-l10n, Version: 2.2.27-2+deb11u2
            CVE-2022-3219, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2022-3219
        Name: gnupg-utils, Version: 2.2.27-2+deb11u2
            CVE-2022-3219, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2022-3219

Evaluated policy: Default vulnerabilities policy
Vulnerabilities: CRITICAL: 0, HIGH: 1, MEDIUM: 40, LOW: 93, INFORMATIONAL: 2
    Total: 136, out of which 4 are fixable
Scan results: PASSED. Container image meets policy requirements
Scan report: https://app.wiz.io/reports/cicd-scans#~(cicd_scan~'25910d72-db1c-4638-84a0-e7aba3e56461)
```

Example command with Podman

If you are running Wiz CLI to scan a container image on a machine that has a Podman socket, perform the following steps:

1. Open a Podman Unix socket for the duration of the scan. This can be done ad hoc:

Linux or Windows | macOS

```
podman system service --time 120 &
```

2. Depending on your OS system, follow these instructions:

   i. On MacOS, use the output of the previous step to set your `DOCKER_HOST` environment variable to your Podman socket path and make sure to add the `unix://` prefix to the path. Then, run Wiz CLI normally.

   ii. On Linux or Windows, execute the regular Wiz CLI container image scan command and have it work with Podman by setting the Docker host path.

Linux or Windows | macOS

```
PODMAN_SOCKET_PATH="$(podman system info -f json | jq -r
.host.remoteSocket.path)"
```

```
DOCKER_HOST="unix://${PODMAN_SOCKET_PATH}" ./wizcli docker scan --image
mongo:1.2.3-ab1
```

Example command with a remote docker daemon

If you are running Wiz CLI to scan a container image on a remote machine, then perform the following steps:

1. Pull Wiz CLI as a Docker image.
2. Execute Wiz CLI via the remote daemon:

Example command

```
docker run -v ~/.wiz:/cli wizcli:latest --no-style auth --id ${WIZCLI_ID}
--secret ${WIZCLI_SECRET}
docker run --security-opt apparmor:unconfined --cap-add SYS_ADMIN -v
/var/lib/docker:/var/lib/docker -v
/var/run/docker.sock:/var/run/docker.sock -v ~/.wiz:/cli wizcli:latest --
no-style docker scan --image  PARAMETERS.IMAGE  --policy "
PARAMETERS.POLICY "
```

Enriching the Security Graph with CI metadata

The prerequisites are:

- Wiz CLI downloaded and authenticated, including a service account with the appropriate permissions (for authentication via service account only).
- The image name.
- The Docker images must exist locally. You can use `docker image list` to verify.
- The Docker images must have a digest (assigned after a successful push to the registry). You can use `docker images --digests` to verify.
- The Docker images must be scanned locally before attempting to tag them.

Usage

To view the help page, enter `wizcli docker tag --help`.

Help

```
Fetches the Image ID of a local Docker image and publishes its CI metadata to
the Security Graph

Usage:
  wizcli docker tag [flags]
```

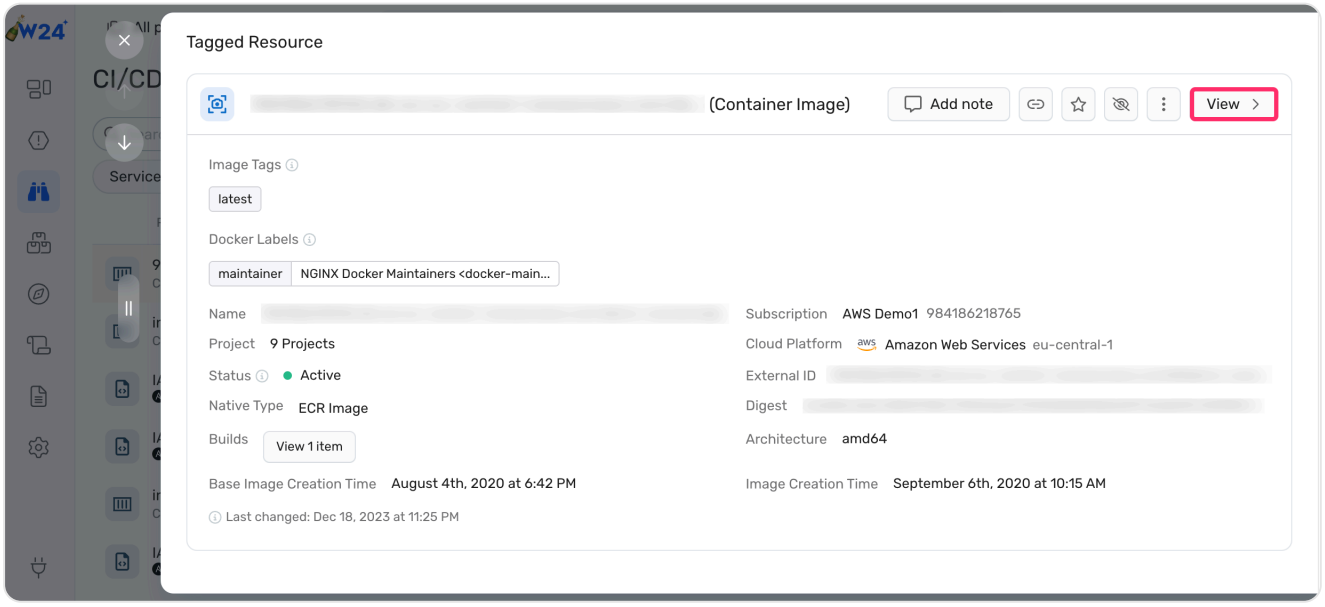
```
Flags:
  -d, --driver string   Driver used to scan image. Supported options are
[extract, mount, mountWithLayers] (default "extract")
  -h, --help            help for tag
  -i, --image string    Image name, including the tag/digest (the same value
as used in the scan command) (required)

Global Flags:
      --log string     File path at which to write debug logs (defaults to no
logs)
  -C, --no-color       Disable color output
  -S, --no-style       Disable stylized output
  -T, --no-telemetry   Disable telemetry
```

> ⚠ The `wizcli docker tag` command is not applicable to exported images (such as .tar, .tar.gz, and .tgz files) since they don't contain the image digest.

Metadata enrichment flow

1. When using the `wizcli docker tag` command, Wiz CLI fetches the digest of the scanned Docker image and links it to the relevant scan.
2. CI metadata (including the Docker image contents) is added to the Security Graph in the next scan of your environment, given that Wiz finds a container image with the same repository and digest.
3. Information about the tagged Docker image appears in the CI/CD scan details drawer:
   i. First, a section called Tagged Resources is added and includes the resource ID of the Docker image's future Graph object.
   ii. Once the Graph object is created, this section is updated with a Graph link and the CI metadata.



> The CI metadata helps you correlate the Docker image to the CI phase, gain further insights, and potentially improve your CI/CD pipeline conduct.

Multi-platform container images

Multi-platform container images may have more than one target platform associated with them. Since scanning such container images only covers the (platform) container image that matches your host platform, you need to scan each container image separately. On the other hand, when tagging such container images, the image tag should refer to the index manifest (i.e., the multi-platform container image manifest) and not to any platform-specific manifest.

The examples below assume the following setup:

- You have built a multi-platform container image for `linux/amd64` and `linux/arm64` and named it "mycompany/nginx".
- The container image was pushed into your Docker Hub container registry, to an account called "mycompany" and image repository called "nginx".

Scanning multi-platform container images

You can scan all (platform) container images that are part of a multi-platform container image using various third party tools such as Skopeo, Crane, and Docker.

Using Skopeo

Using Skopeo

```
skopeo \
  --override-os linux \
  --override-arch amd64 \
  copy docker://mycompany/myimage docker-daemon:mycompany/myimage:latest-amd64
wizcli docker scan --image mycompany/nginx:latest-amd64

skopeo \
  --override-os linux
  --override-arch arm64 \
  copy docker://mycompany/myimage docker-daemon:mycompany/myimage:latest-arm64
wizcli docker scan --image mycompany/nginx:latest-arm64
```

Using Crane

Using Crane

```
crane pull --platform linux/arm64 mycompany/nginx nginx-arm64.tar
wizcli docker scan nginx-arm64.tar

crane pull --platform linux/amd64 mycompany/nginx nginx-amd64.tar
wizcli docker scan nginx-amd64.tar
```

Using Docker

Using Docker

```
docker pull --platform linux/arm64 mycompany/nginx
wizcli docker scan mycompany/nginx
docker rmi mycompany/nginx

docker pull --platform linux/amd64 mycompany/nginx
wizcli docker scan mycompany/nginx
docker rmi mycompany/nginx
```

Tagging multi-platform container images

Below is an example for tagging a multi-platform container image:

Example tag command

```
wizcli docker tag --image mycompany/nginx
```

When tagging exported tarball images (e.g. `.tar` / `.tar.gz` / `.tgz`), you should specify the digest of the multi-arch manifest. For example:

Example tag command of a tarball image

```
crane pull --platform linux/amd64 mycompany/nginx nginx-amd64.tar

wizcli docker scan nginx-amd64.tar

wizcli docker tag \
  --digest
sha256:b05e69a3d5049e71d58b260d212b2496f6ddd97a97870ee38f3d0e74778563f0 \
  --image nginx-amd64.tar
```

Scan results

Scan results are output directly to the console:

```
[ec2-user@[redacted] ~]$ ./wizcli docker scan --image docker.io/library/debian:latest
          _           _ _
 __      _(_)___   ___| (_)
 \ \ /\ / / |_  /  / __| | |
  \ V  V /| |/ /  | (__| | |
   \_/\_/ |_/___|  \___|_|_|
 SUCCESS  Ready to scan Docker image docker.io/library/debian:latest
 SUCCESS  Scanned Docker image
 SUCCESS  Docker image scan analysis ready
OS Package vulnerabilities:
    Name: libpcre3, Version: 2:8.39-13
        CVE-2017-16231, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2017-16231
        CVE-2017-7245, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2017-7245
        CVE-2017-7246, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2017-7246
        CVE-2019-20838, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2019-20838
        CVE-2017-11164, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2017-11164
    Name: libc-bin, Version: 2.31-13
        CVE-2021-33574, Severity: CRITICAL, Source: https://security-tracker.debian.org/tracker/CVE-2021-33574
        CVE-2019-1010023, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2019-1010023
        CVE-2019-1010024, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2019-1010024
        CVE-2019-1010025, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2019-1010025
                        ⋮
    Name: libgcrypt20, Version: 1.8.7-6
        CVE-2021-40528, Severity: MEDIUM, Source: https://security-tracker.debian.org/tracker/CVE-2021-40528
        CVE-2018-6829, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2018-6829
    Name: libgnutls30, Version: 3.7.1-5
        CVE-2011-3389, Severity: LOW, Source: https://security-tracker.debian.org/tracker/CVE-2011-3389

Evaluated policy: Default vulnerabilities policy
Vulnerabilities: CRITICAL: 2, HIGH: 0, MEDIUM: 5, LOW: 55, INFORMATIONAL: 0
Scan results: PASSED. Image meets policy requirements
```

Wiz CLI scan results are also listed on the [Findings > CI/CD Scans]() page, where they can be filtered and inspected in JSON format.

Updated 3 days ago

← Scan IaC Templates with Wiz CLI

Scan Directories with Wiz CLI →