Docker static code analysis

Unique rules to find Vulnerabilities, Security Hotspots, and Code Smells in your DOCKER code

All rules 44 | Vulnerability 4 | Bug 4 | Security Hotspot 15 | Code Smell 21

Tags | Impact | Clean code attribute | Search by name...

Using ENV or ARG to handle secrets is security-sensitive

Analyze your code

Responsibility - Trustworthy | Security

Security Hotspot | Blocker | dockerfile cwe

Using ENV and ARG to handle secrets can lead to sensitive information being disclosed to an inappropriate sphere.

The ARG and ENV instructions in a Dockerfile are used to configure the image build and the container environment respectively. Both can be used at image build time, during the execution of commands in the container, and ENV can also be used at runtime.

In most cases, build-time and environment variables are used to propagate configuration items from the host to the image or container. A typical example for an environmental variable is the PATH variable, used to configure where system executables are searched for.

Using ARG and ENV to propagate configuration entries that contain secrets causes a security risk. Indeed, in most cases, artifacts of those values are kept in the final image. The secret information leak can happen either in the container environment itself, the image metadata or the build environment logs.

The concrete impact of such an issue highly depends on the secret's purpose and the exposure sphere:

- Financial impact if a paid service API key is disclosed and used.
- Application compromise if an application's secret, like a session signing key, is disclosed.
- Infrastructure component takeover, if a system secret, like a remote access key, is leaked.

Ask Yourself Whether

- The variable contains a value that should be kept confidential.
- The container image or Dockerfile will be distributed to users who do not need to know the secret value.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Use Buildkit's secret mount options when secrets have to be used at build time.
- For run time secret variables, best practices would recommend only setting them at runtime, for example with the --env option of the docker run command.

Note that, in both cases, the files exposing the secrets should be securely stored and not exposed to a large sphere. In most cases, using a secret vault or another similar component should be preferred. For example, **Docker Swarm** provides a **secrets** service that can be used to handle most confidential data.

Sensitive Code Example

```
FROM example
# Sensitive
ARG ACCESS_TOKEN
# Sensitive
ENV ACCESS_TOKEN=${ACCESS_TOKEN}
CMD /run.sh
```

Compliant Solution

For build time secrets, use Buildkit's secret mount type instead:

```
FROM example
RUN --mount=type=secret,id=build_secret ./installer.sh
```

For runtime secrets, leave the environment variables empty until runtime:

```
FROM example
ENV ACCESS_TOKEN=""
CMD /run.sh
```

Store the runtime secrets in an environment file (such as .env) and then start the container with the --env-file argument:

```
docker run --env-file .env myImage
```

See

- Dockerfile reference - ENV command
- Dockerfile reference - ARG command
- Dockerfile reference - RUN command secrets mount points
- Docker documentation - Manage sensitive data with Docker secrets
- CWE - CWE-522 - Insufficiently Protected Credentials

Available In:

sonarlint | sonarcloud | sonarqube