sonar RULES

Products

Secrets
ABAP
Apex
AzureResourceManager
C
C#
C++
CloudFormation
COBOL
CSS
Dart
Docker
Flex
Go
HTML
Java
JavaScript
JCL
Kotlin
Kubernetes
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

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...

Code Smell

Instructions should be upper case
Code Smell

Allowing non-root users to modify resources copied to an image is security-sensitive
Security Hotspot

Automatically installing recommended packages is security-sensitive
Security Hotspot

Running containers as a privileged user is security-sensitive
Security Hotspot

Delivering code in production with debug features activated is security-sensitive
Security Hotspot

Use ADD instruction to retrieve remote resources
Code Smell

Arguments in long RUN instructions should be sorted
Code Smell

Track uses of "TODO" tags
Code Smell

Descriptive labels are mandatory
Code Smell

Use digest to pin versions of base images
Code Smell

Dockerfile parsing failure
Code Smell

Allowing non-root users to modify resources copied to an image is security-sensitive

Analyze your code

Responsibility - Trustworthy | Security

Security Hotspot | Minor | dockerfile cwe

Ownership or write permissions for a file or directory copied to the Docker image have been assigned to a user other than root.

Write permissions enable malicious actors, who have a foothold on the container, to tamper with the resource and thus potentially manipulate the container's expected behavior.
Manipulating files could disrupt services or aid in escalating privileges inside the container.

This also breaches the container immutability principle as it facilitates container changes during its life. Immutability, a container best practice, allows for a more reliable and reproducible behavior of Docker containers.

If a user is given ownership on a file but no write permissions, the user can still modify it by using his ownership to change the file permissions first. This is why both ownership and write permissions should be avoided.

Ask Yourself Whether

- A non-root user owns the resource.
- A non-root user has been granted write permissions for the resource.

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

Recommended Secure Coding Practices

- Use `--chmod` to change the permissions so that only root users can write to files.
- Use `--chown` to change the file/directory owner to a root user.
- Be mindful of the container immutability principle.

Sensitive Code Example

```
FROM example

RUN useradd exampleuser
# Sensitive
COPY --chown=exampleuser:exampleuser src.py dst.py
```

Compliant Solution

```
FROM example

COPY --chown=root:root --chmod=755 src.py dst.py
```

See

- Dockerfile reference - ADD instruction
- Dockerfile reference - COPY instruction
- CWE - CWE-732 - Incorrect Permission Assignment for Critical Resource
- Google Cloud, Immutability - Best practices for operating containers

Available In:
sonarlint | sonarcloud | sonarqube