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...

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

Pulling an image based on its digest is security-sensitive
Security Hotspot

Use ADD instruction to retrieve remote resources

Analyze your code

Consistency - Conventional | Maintainability

Code Smell | Minor

In Dockerfiles, a common use case is downloading remote resources to use during the build. This is often done using third-party tools inside the image, like `wget` or `curl`. However, this practice can lead to inefficient use of Docker's build cache and unnecessary complexity. The `ADD` instruction is a built-in feature of Docker that is specifically designed for this purpose, making it a more efficient and safer choice.

Why is this an issue? | How can I fix it? | More Info

Using `wget` or `curl` commands to retrieve remote content in Dockerfiles instead of the `ADD` instruction can lead to several issues, particularly related to the inefficient use of Docker's build cache.

Docker's build cache is a powerful feature that can significantly speed up the build process by reusing intermediate layers from previous builds if no changes were detected. When you use `wget`, `curl`, or similar commands, these commands are run during the build process, and Docker has no way of knowing if the remote content has changed without executing the commands. This makes it impossible to cache the results of these commands efficiently.

Moreover, installing third-party tools inside the image can introduce unnecessary complexity, dependency on external tools and increase the size of the final image.

Exceptions

In some cases, the `ADD` instruction is not able to replace the `wget` or `curl` command, especially if specific HTTP parameters are required: method, headers, body, etc.

```
FROM ubuntu:20.04
RUN wget --header="Authorization: Bearer your_token" --method=POST https://example.com/resource
```

Available In:

sonarlint | sonarcloud | sonarqube