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...

Weak SSL/TLS protocols should not be used
Vulnerability

Disabling builder sandboxes is security-sensitive
Security Hotspot

Exposing administration services is security-sensitive
Security Hotspot

Recursively copying context directories is security-sensitive
Security Hotspot

Using clear-text protocols is security-sensitive
Security Hotspot

Using weak hashing algorithms is security-sensitive
Security Hotspot

Malformed JSON in Exec form leads to unexpected behavior
Bug

Dockerfile should only have one ENTRYPOINT and CMD instruction
Bug

Access variable which is not available in the current scope
Bug

A space before the equal sign in key-value pair may lead to unintended behavior
Bug

Allowing downgrades to a clear-text protocol is security-sensitive
Security Hotspot

Allowing shell scripts execution during package

Credentials should not be hard-coded

Analyze your code

Responsibility - Trustworthy | Security

Vulnerability | Blocker | cwe

Secret leaks often occur when a sensitive piece of authentication data is stored with the source code of an application. Considering the source code is intended to be deployed across multiple assets, including source code repositories or application hosting servers, the secrets might get exposed to an unintended audience.

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

In most cases, trust boundaries are violated when a secret is exposed in a source code repository or an uncontrolled deployment environment. Unintended people who don't need to know the secret might get access to it. They might then be able to use it to gain unwanted access to associated services or resources.

The trust issue can be more or less severe depending on the people's role and entitlement.

In Dockerfiles, hard-coded secrets and secrets passed through as variables or created at build-time will cause security risks. The secret information can be exposed either via the container environment, the image metadata, or the build environment logs.

What is the potential impact?

The consequences vary greatly depending on the situation and the secret-exposed audience. Still, two main scenarios should be considered.

Financial loss

Financial losses can occur when a secret is used to access a paid third-party-provided service and is disclosed as part of the source code of client applications. Having the secret, each user of the application will be able to use it without limit to use the third party service to their own need, including in a way that was not expected.

This additional use of the secret will lead to added costs with the service provider.

Moreover, when rate or volume limiting is set up on the provider side, this additional use can prevent the regular operation of the affected application. This might result in a partial denial of service for all the application's users.

Application's security downgrade

A downgrade can happen when the disclosed secret is used to protect security-sensitive assets or features of the application. Depending on the affected asset or feature, the practical impact can range from a sensitive information leak to a complete takeover of the application, its hosting server or another linked component.

For example, an application that would disclose a secret used to sign user authentication tokens would be at risk of user identity impersonation. An attacker accessing the leaked secret could sign session tokens for arbitrary users and take over their privileges and entitlements.

Available In:
sonarlint | sonarcloud | sonarqube