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...

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 installation is security-sensitive
Security Hotspot

Disabling builder sandboxes is security-sensitive

Analyze your code

Responsibility - Trustworthy | Security

Security Hotspot | Critical | dockerfile cwe

Disabling builder sandboxes can lead to unauthorized access of the host system by malicious programs.

By default, programs executed by a `RUN` statement use only a subset of capabilities which are considered safe: this is called `sandbox` mode.

If you disable the sandbox with the `--security=insecure` option, the executed command can use the full set of Linux capabilities.
This can lead to a container escape. For example, an attacker with the `SYS_ADMIN` capability is able to mount devices from the host system.

This vulnerability allows an attacker who controls the behavior of the ran command to access the host system, break out of the container and penetrate the infrastructure.

After a successful intrusion, the underlying systems are exposed to:

- theft of intellectual property and/or personal data
- extortion
- denial of service

Ask Yourself Whether

- The program is controlled by an external entity.
- The program is part of a supply chain that could be a victim of a supply chain attack.

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

Recommended Secure Coding Practices

- Whenever possible, the sandbox should stay enabled to reduce unnecessary risk.
- If elevated capabilities are absolutely necessary, make sure to verify the integrity of the program before executing it.

Sensitive Code Example

```
# syntax=docker/dockerfile:1-labs
FROM ubuntu:22.04
# Sensitive
RUN --security=insecure ./example.sh
```

Compliant Solution

```
# syntax=docker/dockerfile:1-labs
FROM ubuntu:22.04
RUN ./example.sh
RUN --security=sandbox ./example.sh
```

See

- CWE - CWE-250 - Execution with Unnecessary Privileges
- CWE - CWE-284 - Improper Access Control
- Dockerfile reference - RUN

Available In:

sonarlint | sonarcloud | sonarqube