Docker static code analysis

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

Credentials should not be hard-coded
Vulnerability

Using ENV or ARG to handle secrets is security-sensitive
Security Hotspot

Permissions of sensitive mount points should be restrictive
Vulnerability

Server certificates should be verified during SSL/TLS connections
Vulnerability

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

Weak SSL/TLS protocols should not be used

Analyze your code

Responsibility - Trustworthy | Security

Vulnerability | Critical | cwe privacy

This vulnerability exposes encrypted data to a number of attacks whose goal is to recover the plaintext.

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

Which component or framework contains the issue?

cURL | Wget

How to fix it in cURL

Noncompliant code example

```
FROM ubuntu:22.04

# Noncompliant
RUN curl --tlsv1.0 -O https://tlsv1-0.example.com/downloads/install.sh
```

Compliant solution

```
FROM ubuntu:22.04

RUN curl --tlsv1.2 -O https://tlsv1-3.example.com/downloads/install.sh
```

How does this work?

As a rule of thumb, by default you should use the cryptographic algorithms and mechanisms that are considered strong by the cryptographic community.

The best choices at the moment are the following.

Use TLS v1.2 or TLS v1.3

Even though TLS V1.3 is available, using TLS v1.2 is still considered good and secure practice by the cryptography community.

The use of TLS v1.2 ensures compatibility with a wide range of platforms and enables seamless communication between different systems that do not yet have TLS v1.3 support.

The only drawback depends on whether the framework used is outdated: its TLS v1.2 settings may enable older and insecure cipher suites that are deprecated as insecure.

On the other hand, TLS v1.3 removes support for older and weaker cryptographic algorithms, eliminates known vulnerabilities from previous TLS versions, and improves performance.

Available In:

sonarlint | sonarcloud | sonarqube