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

Arguments in long RUN instructions should be sorted

Analyze your code

Consistency - Conventional | Maintainability | Reliability | Security

Code Smell | Minor

In Dockerfiles, when commands within a `RUN` instruction have a lot of arguments, especially those that install system packages, it is important to ensure that the arguments are sorted alphabetically (if the order is not enforced by a command). This practice enhances the readability and maintainability of the code. It allows for easier tracking of modifications and can help prevent potential errors.

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

When arguments in `RUN` instructions are not sorted alphabetically, it can make the Dockerfile harder to read and maintain. However, when arguments are sorted, it is easier to track changes in version control systems and to locate specific arguments. This applies first and foremost to package managers, where a list of installed packages can grow over time.

Available In:

sonarlint | sonarcloud | sonarqube