





































-  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 shell scripts execution during package installation is security-sensitive	Security Hotspot
Using host operating system namespaces is security-sensitive	Security Hotspot
Setting loose POSIX file permissions is security-sensitive	Security Hotspot
Reduce the amount of consecutive RUN instructions	Code Smell
Prefer COPY over ADD for copying local resources	Code Smell
WORKDIR instruction should only be used with absolute path	Code Smell
Too long RUN instruction should be split into multiple lines	Code Smell
Prefer Exec form for ENTRYPOINT and CMD instructions	Code Smell
"WORKDIR" instruction should be used instead of "cd" commands	Code Smell
Specific version tag for image should be used	Code Smell
Package update should not be executed without installing it	Code Smell
Cache should be cleaned after package installation	

## Allowing shell scripts execution during package installation is security-sensitive

Analyze your code

Responsibility - Trustworthy

Security ⬆

 Security Hotspot  Major  cwe

When installing dependencies, package managers like `npm` will automatically execute shell scripts distributed along with the source code. Post-install scripts, for example, are a common way to execute malicious code at install time whenever a package is compromised.

### Ask Yourself Whether

- The execution of dependency installation scripts is required for the application to function correctly.

There is a risk if you answered no to the question.

### Recommended Secure Coding Practices

Execution of third-party scripts should be disabled if not strictly necessary for dependencies to work correctly. Doing this will reduce the attack surface and block a well-known supply chain attack vector.

Commands that are subject to this issue are: `npm install`, `yarn install` and `yarn` (`yarn` without an explicit command will execute `install`).

### Sensitive Code Example

```
FROM node:latest

# Sensitive
RUN npm install
```

```
FROM node:latest

# Sensitive
RUN yarn install
```

### Compliant Solution

```
FROM node:latest

RUN npm install --ignore-scripts
```

```
FROM node:latest

RUN yarn install --ignore-scripts
```

### See

- CWE - [CWE-506 - Embedded Malicious Code](#)
- CWE - [CWE-829 - Inclusion of Functionality from Untrusted Control Sphere](#)
- [ESLint blog](#) - Postmortem for Malicious Packages Published on July 12th, 2018

Available In:

**sonarlint**  | **sonarcloud**  | **sonarqube** 

