





































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

Analyze your code

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

Responsibility - Trustworthy Security ⚠️

Security Hotspot Major cwe

The usage of HTTPS is not enforced here. As it is possible for the HTTP client to follow redirects, such redirects might lead to websites using HTTP.

As HTTP is a clear-text protocol, it is considered insecure. Due to its lack of encryption, attackers that are able to sniff traffic from the network can read, modify, or corrupt the transported content. Therefore, allowing redirects to HTTP can lead to several risks:

- Exposure of sensitive data
- Malware-infected software updates or installers
- Corruption of critical information

Even in isolated networks, such as segmented cloud or offline environments, it is important to ensure the usage of HTTPS. If not, then insider threats with access to these environments might still be able to monitor or tamper with communications.

Ask Yourself Whether

- It is possible for the requested resource to be redirected to an insecure location in the future.

There is a risk if you answered yes to the question.

Recommended Secure Coding Practices

- Ensure that the HTTP client only accepts HTTPS pages. In `curl` this can be enabled using the option `--proto "=https"`.
- If it is not necessary to follow HTTP redirects, disable this in the HTTP client. In `curl` this is done by omitting the `-L` or `--location` option. In `wget` this is done by adding the option `--max-redirect=0`.

Sensitive Code Example

In the examples below, an install script is downloaded using `curl` or `wget` and then executed.

While connections made using HTTPS are generally considered secure, `https://might-redirect.example.com/install.sh` might redirect to a location that uses HTTP. Downloads made using HTTP are not secure and can be intercepted and modified. An attacker could modify the install script to run malicious code inside the container.

`curl` will not follow redirects unless either `-L` or `--location` option is used.

```
FROM ubuntu:22.04

# Sensitive
RUN curl --tlsv1.2 -sSf -L https://might-redirect.example.com/install.sh | sh
```

`wget` will follow redirects by default.

```
FROM ubuntu:22.04

# Sensitive
RUN wget --secure-protocol=TLsv1_2 -q -O - https://might-redirect.example.com/install.sh | sh
```

Compliant Solution

If you expect the server to redirect the download to a new location, `curl` can use the option `--proto "=https"` to ensure requests are only made using HTTPS. Any attempt to redirect to a location using HTTP will result in an error.

```
FROM ubuntu:22.04

RUN curl --proto "=https" --tlsv1.2 -sSf -L https://might-redirect.example.com/install.sh | sh
```

`wget` does not support this functionality so `curl` should be used instead.

If you expect the server to return the file without redirects, `curl` should not be instructed to follow redirects. Remove any `-L` or `--location` options from the command.

```
FROM ubuntu:22.04

RUN curl --tlsv1.2 -sSf https://might-redirect.example.com/install.sh | sh
```

`wget` uses the option `--max-redirect=0` to disable redirects.

```
FROM ubuntu:22.04

RUN wget --secure-protocol=TLsv1_2 --max-redirect=0 -q -O - https://might-redirect.example.com/install.sh | sh
```

See

- CWE - CWE-757 - Selection of Less-Secure Algorithm During Negotiation (Algorithm Downgrade)
- [curl.1 the man page](#) - `--proto <protocols>`
- [wget - GNU Wget Manual](#) - `--max-redirect=`
- [SSL and TLS Deployment Best Practices](#) - Encrypt Everything

Available In:
sonarlint sonarcloud | sonarqube

