

Secrets

ABAP

Apex

C

C++

CloudFormation

COBOL

C#

CSS

Flex

Go

HTML

Java

JavaScript

Kotlin

Objective C

PHP

PL/I

PL/SQL

Python

RPG

Ruby

Scala

Swift

Terraform

Text

TypeScript

T-SQL

VB.NET

VB6

XML

Java

Java static code analysis

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

All rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags

Search by name...

Code Smell

"toString()" should never be called on a String object

Code Smell

Annotation repetitions should not be wrapped

Code Smell

Multiple variables should not be declared on the same line

Code Smell

Strings should not be concatenated using '+' in a loop

Code Smell

Maps with keys that are enum values should be replaced with EnumMap

Code Smell

Lambdas should be replaced with method references

Code Smell

Parentheses should be removed from a single lambda input parameter when its type is inferred

Code Smell

Abstract classes without fields should be converted to interfaces

Code Smell

Lambdas containing only one statement should not nest this statement in a block

Code Smell

"Collections.EMPTY_LIST", "EMPTY_MAP", and "EMPTY_SET" should not be used

Code Smell

Allowing requests with excessive content length is security-sensitive

Analyze your code

Security Hotspot

Major

cwe cert owasp

Rejecting requests with significant content length is a good practice to control the network traffic intensity and thus resource consumption in order to prevents DoS attacks.

Ask Yourself Whether

size limits are not defined for the different resources of the web application.

the web application is not protected by rate limiting features.

the web application infrastructure has limited resources.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

For most of the features of an application, it is recommended to limit the size of requests to:

lower or equal to 8mb for file uploads.

lower or equal to 2mb for other requests.

It is recommended to customize the rule with the limit values that correspond to the web application.

Sensitive Code Example

With default limit value of 8388608 (8MB).

A 100 MB file is allowed to be uploaded:

```
@Bean(name = "multipartResolver")
public CommonsMultipartResolver multipartResolver() {
    CommonsMultipartResolver multipartResolver = new CommonsMu
    multipartResolver.setMaxUploadSize(104857600); // Sensitiv
    return multipartResolver;
}

@Bean(name = "multipartResolver")
public CommonsMultipartResolver multipartResolver() {
    CommonsMultipartResolver multipartResolver = new CommonsMu
    return multipartResolver;
}





@Bean
public MultipartConfigElement multipartConfigElement() {
    MultipartConfigFactory factory = new MultipartConfigFactor
    return factory.createMultipartConfig();
}
```

Compliant Solution

File upload size is limited to 8 MB:

https://rules.sonarsource.com/java/RSPEC-5693

1/2

| |
|--|
| Local variables should not be declared and then immediately returned or thrown |
|  Code Smell |
| Unused local variables should be removed |
|  Code Smell |
| Private fields only used as local variables in methods should become local variables |
|  Code Smell |
| "public static" fields should be constant |
|  Code Smell |

```
@Bean(name = "multipartResolver")
public CommonsMultipartResolver multipartResolver() {
    multipartResolver.setMaxUploadSize(8388608); // Compliant
    return multipartResolver;
}
```

See

- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [Owasp Cheat Sheet](#) - Owasp Denial of Service Cheat Sheet
- [OWASP Top 10 2017 Category A6](#) - Security Misconfiguration
- [MITRE, CWE-770](#) - Allocation of Resources Without Limits or Throttling
- [MITRE, CWE-400](#) - Uncontrolled Resource Consumption

Available In:

