

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 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

"Stream.toList()" method should be used instead of "collectors" when unmodifiable list needed

Operator "instanceof" should be used instead of "A.class.isInstance()"

String multiline concatenation should be replaced with Text Blocks

Single-character alternations in regular expressions should be replaced with character classes

Reluctant quantifiers in regular expressions should be followed by an expression that can't match the empty string

Constructors of an "abstract" class should not be declared "public"

Similar tests should be grouped in a single Parameterized test

Tests should be stable

Test methods should not contain too many assertions

AssertJ "assertThatThrownBy" should not be used alone

Character classes in regular expressions should not contain the

XML parsers should not be vulnerable to Denial of Service attacks

Analyze your code

Vulnerability

Major

An XML bomb / **billion laughs** attack is a malicious XML document containing the same large entity repeated over and over again. If no restrictions is in place, such a limit on the number of entity expansions, the XML processor can consume a lot memory and time during the parsing of such documents leading to Denial of Service.

Noncompliant Code Example

For **DocumentBuilder**, **SAXParser** and **Schema** and **Transformer** JAPX factories:

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
factory.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);

SAXParserFactory factory = SAXParserFactory.newInstance();
factory.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);

SchemaFactory factory = SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
factory.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);

TransformerFactory factory = javax.xml.transform.TransformerFactory.newInstance();
factory.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
```

For **Dom4j** library:

```
SAXReader xmlReader = new SAXReader();
xmlReader.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
```

For **Jdom2** library:

```
SAXBuilder builder = new SAXBuilder();
builder.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
```

Compliant Solution

For **DocumentBuilder**, **SAXParser** and **Schema** and **Transformer** JAPX factories:

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
factory.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, true);





SAXParserFactory factory = SAXParserFactory.newInstance();
factory.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, true);

SchemaFactory factory = SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
factory.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, true);

TransformerFactory factory = javax.xml.transform.TransformerFactory.newInstance();
factory.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, true);
```

https://rules.sonarsource.com/java/RSPEC-6376

1/2

same character twice
 Code Smell
Names of regular expressions named groups should be used
 Code Smell
Regexes containing characters subject to normalization should use the CANON_EQ flag
 Code Smell
Regular expressions should not be too complicated
 Code Smell

For [Dom4j](#) library:

```
SAXReader xmlReader = new SAXReader();
xmlReader.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING,
```

For [Jdom2](#) library:

```
SAXBuilder builder = new SAXBuilder();
builder.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, t
```

See

- [Oracle Java Documentation](#) - XML External Entity Injection Attack
- [OWASP Top 10 2017 Category A4](#) - XML External Entities (XXE)
- [OWASP XXE Prevention Cheat Sheet](#)
- [MITRE, CWE-776](#) - Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')

Available In:

