sonar RULES

Products

- 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 rules 632 | Vulnerability 53 | Bug 154 | Security Hotspot 36 | Code Smell 389 | Quick Fix 42

Tags | Search by name...

vulnerable to session fixation
Vulnerability

Extracting archives should not lead to zip slip vulnerabilities
Vulnerability

Dynamic code execution should not be vulnerable to injection attacks
Vulnerability

NoSQL operations should not be vulnerable to injection attacks
Vulnerability

HTTP request redirections should not be open to forging attacks
Vulnerability

Deserialization should not be vulnerable to injection attacks
Vulnerability

Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks
Vulnerability

Database queries should not be vulnerable to injection attacks
Vulnerability

XML parsers should not be vulnerable to XXE attacks
Vulnerability

A secure password should be used when connecting to a database
Vulnerability

XPath expressions should not be vulnerable to injection attacks
Vulnerability

I/O function calls should not be vulnerable to path injection attacks

XML parsers should not allow inclusion of arbitrary files

Analyze your code

Vulnerability | Blocker

XML standard allows the inclusion of xml files with the xinclude element.

XML processors will replace an xinclude element with the content of the file located at the URI defined in the href attribute, potentially from an external storage such as file system or network, which may lead, if no restrictions are put in place, to arbitrary file disclosures or server-side request forgery (SSRF) vulnerabilities.

Noncompliant Code Example

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

```
factory.setXIncludeAware(true); // Noncompliant
// or
factory.setFeature("http://apache.org/xml/features/xinclude"
```

For Dom4j library:

```
SAXReader xmlReader = new SAXReader();
xmlReader.setFeature("http://apache.org/xml/features/xinclud
```

For Jdom2 library:

```
SAXBuilder builder = new SAXBuilder();
builder.setFeature("http://apache.org/xml/features/xinclude"
```

Compliant Solution

Xinclude is disabled by default and can be explicitely disabled like below.

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

```
factory.setXIncludeAware(false);
// or
factory.setFeature("http://apache.org/xml/features/xinclude"
```

For Dom4j library:

```
SAXReader xmlReader = new SAXReader();
xmlReader.setFeature("http://apache.org/xml/features/xinclud
```

For Jdom2 library:

```
SAXBuilder builder = new SAXBuilder();
builder.setFeature("http://apache.org/xml/features/xinclude"
```

Vulnerability

LDAP queries should not be vulnerable to injection attacks

Vulnerability

OS commands should not be vulnerable to command injection attacks

Vulnerability

"@SpringBootApplication" and "@ComponentScan" should not be used in the default package

Bug

"@Controller" classes that use

Exceptions

This rule does not raise issues when Xinclude is enabled with a custom `EntityResolver`:

For DocumentBuilderFactory:

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newI
factory.setXIncludeAware(true);
// ...
DocumentBuilder builder = factory.newDocumentBuilder();
builder.setEntityResolver((publicId, systemId) -> new MySafe
```

For SAXBuilder:

```
SAXBuilder builder = new SAXBuilder();
builder.setFeature("http://apache.org/xml/features/xinclude"
builder.setEntityResolver((publicId, systemId) -> new MySafe
```

For SAXReader:

```
SAXReader xmlReader = new SAXReader();
xmlReader.setFeature("http://apache.org/xml/features/xinclud
xmlReader.setEntityResolver((publicId, systemId) -> new MySa
```

For XMLInputFactory:

```
XMLInputFactory factory = XMLInputFactory.newInstance();
factory.setProperty("http://apache.org/xml/features/xinclude
factory.setXMLResolver(new MySafeEntityResolver());
```

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-611 - Information Exposure Through XML External Entity Reference
- MITRE, CWE-827 - Improper Control of Document Type Definition

Available In:

sonarlint | sonarcloud | sonarqube