






































-  Secrets
-  ABAP
-  Ansible
-  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



Java static code analysis

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

All rules

703



Vulnerability

57



Bug

173



Security Hotspot

38



Code Smell

435



Quick Fix

63

Tags



Impact



Clean code attribute



Search by name...



XML parsers should not be vulnerable to XXE attacks

Analyze your code

Intentionality - Complete

Security



Vulnerability



Blocker



cwe symbolic-execution

This vulnerability allows the usage of external entities in XML.

Why is this an issue?

How can I fix it?

More Info

External Entity Processing allows for XML parsing with the involvement of external entities. However, when this functionality is enabled without proper precautions, it can lead to a vulnerability known as XML External Entity (XXE) attack.

What is the potential impact?

Exposing sensitive data

One significant danger of XXE vulnerabilities is the potential for sensitive data exposure. By crafting malicious XML payloads, attackers can reference external entities that contain sensitive information, such as system files, database credentials, or configuration files. When these entities are processed during XML parsing, the attacker can extract the contents and gain unauthorized access to sensitive data. This poses a severe threat to the confidentiality of critical information.

Exhausting system resources

Another consequence of XXE vulnerabilities is the potential for denial-of-service attacks. By exploiting the ability to include external entities, attackers can construct XML payloads that cause resource exhaustion. This can overwhelm the system’s memory, CPU, or other critical resources, leading to system unresponsiveness or crashes. A successful DoS attack can disrupt the availability of services and negatively impact the user experience.

Forging requests

XXE vulnerabilities can also enable Server-Side Request Forgery (SSRF). If an application allows users to provide URLs for external entities, and it does not properly validate these URLs, attackers can include external entities, an attacker can make arbitrary requests to other internal or external IP addresses for unintended actions, such as retrieving sensitive data from internal networks, or attacking other systems. The consequences can be severe, including data leakage, system compromise, or even further exploitation.











By clicking “Accept All Cookies”, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.

Accept All Cookies

Reject All

No, let me choose

Available In:

<div>Interface names should comply with a naming convention</div> <div> Code Smell</div>
<div>Exceptions in "throws" clauses should not be superfluous</div> <div> Code Smell</div>
<div>Unnecessary imports should be removed</div> <div> Code Smell</div>
<div>Return of boolean expressions should not be wrapped into an "if-then-else" statement</div> <div> Code Smell</div>
<div>Boolean literals should not be redundant</div> <div> Code Smell</div>
<div>Modifiers should be declared in the correct order</div> <div> Code Smell</div>
<div>Empty statements should be removed</div> <div> Code Smell</div>
<div>Class variable fields should not have public accessibility</div> <div> Code Smell</div>
<div>URIs should not be hardcoded</div> <div> Code Smell</div>
<div>Class names should comply with a naming convention</div> <div> Code Smell</div>
<div>Method names should comply with a naming convention</div>

sonarlint



|

sonarcloud



|

sonarqube



© 2008-2024 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE, and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Sonar helps developers write Clean Code.
[Privacy Policy](#) | [Cookie Policy](#)

By clicking “Accept All Cookies”, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.