

































-  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
















JavaScript static code analysis

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

- All rules 285
-  Vulnerability 29
-  Bug 62
-  Security Hotspot 43
-  Code Smell 151
-  Quick Fix 41

Tags 

Search by name... 

Assertions should be complete	 Code Smell
Variables should be declared explicitly	 Code Smell
Tests should include assertions	 Code Smell
"future reserved words" should not be used as identifiers	 Code Smell
Octal values should not be used	 Code Smell
Switch cases should end with an unconditional "break" statement	 Code Smell
"switch" statements should not contain non-case labels	 Code Smell
A new session should be created during user authentication	 Vulnerability
JWT should be signed and verified with strong cipher algorithms	 Vulnerability
Cipher algorithms should be robust	 Vulnerability
Encryption algorithms should be used with secure mode and padding scheme	 Vulnerability
Server hostnames should be verified during SSL/TLS connections	 Vulnerability
Server certificates should be verified	

XML parsers should not be vulnerable to XXE attacks

Analyze your code

-  Vulnerability
-  Blocker
- 
-  cwe owasp

XML standard allows the use of entities, declared in the DOCTYPE of the document, which can be [internal](#) or [external](#).

When parsing the XML file, the content of the external entities is retrieved from an external storage such as the 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.

It's recommended to limit resolution of external entities by using one of these solutions:

- If DOCTYPE is not necessary, completely disable all DOCTYPE declarations.
- If external entities are not necessary, completely disable their declarations.
- If external entities are necessary then:
  - Use XML processor features, if available, to authorize only required protocols (eg: https).
  - And use an entity resolver (and optionally an XML Catalog) to resolve only trusted entities.

Noncompliant Code Example

[libxmljs](#) module:

```
const libxmljs = require("libxmljs");
var fs = require('fs');

var xml = fs.readFileSync('xxe.xml', 'utf8');

var xmlDoc = libxmljs.parseXmlString(xml, { noblanks: true,
```

Compliant Solution

[libxmljs](#) module:

```
const libxmljs = require("libxmljs");
var fs = require('fs');

var xml = fs.readFileSync('xxe.xml', 'utf8');


var xmlDoc = libxmljs.parseXmlString(xml); // Compliant: noe
```

See


- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [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:


during SSL/TLS connections

 Vulnerability


Cryptographic keys should be robust

 Vulnerability




Weak SSL/TLS protocols should not be used

 Vulnerability

Origins should be verified during cross-origin communications

 Vulnerability

Regular expressions should not be vulnerable to Denial of Service attacks

sonarlint  | sonarcloud  | sonarqube 

© 2008-2022 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.  
[Privacy Policy](#)