
































-  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



PHP static code analysis

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

All rules 268


 Vulnerability 40













 Bug 51

 Security Hotspot 33

 Code Smell 144

Tags ▾

Search by name... 

OS commands should not be vulnerable to command injection attacks		Vulnerability
Class of caught exception should be defined		Bug
Caught Exceptions must derive from Throwable		Bug
Raised Exceptions must derive from Throwable		Bug
"\$this" should not be used in a static context		Bug
Hard-coded credentials are security-sensitive		Security Hotspot
Test class names should end with "Test"		Code Smell
Tests should include assertions		Code Smell
TestCases should contain tests		Code Smell
Variable variables should not be used		Code Smell
A new session should be created during user authentication		Vulnerability
Cipher algorithms should be robust		Vulnerability
Encryption algorithms should be used		

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

[SimpleXML](#) object:

```
$xml = file_get_contents("xe.xml");  
$doc = simplexml_load_string($xml, "SimpleXMLElement", LIBXML
```

[DOMDocument](#) object:

```
$doc = new DOMDocument();  
$doc->load("xe.xml", LIBXML_NOENT); // Noncompliant (LIBXML
```

[XMLReader](#) object:

```
$reader = new XMLReader();  
$reader->open("xe.xml");  
$reader->setParserProperty(XMLReader::SUBST_ENTITIES, true);
```

Compliant Solution


[SimpleXML](#) object:

```
$xml = file_get_contents("xe.xml");  
$doc = simplexml_load_string($xml, "SimpleXMLElement"); // C
```

[DOMDocument](#) object:

```
$doc = new DOMDocument();  
$doc->load("xe.xml"); // Compliant (external entities subst
```


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 during SSL/TLS connections

 Vulnerability

LDAP connections should be authenticated

 Vulnerability

[XMLReader](#) object:

```
$reader = new XMLReader();  
$reader->open("xxe.xml");  
$reader->setParserProperty(XMLReader::SUBST_ENTITIES, false)
```

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:

sonarlint  | **sonarcloud**  | **sonarqube** 