

- 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...

should not have exactly the same implementation	Bug
The output of functions that don't return anything should not be used	Bug
Unary prefix operators should not be repeated	Bug
"=+" should not be used instead of "+="	Bug
A "for" loop update clause should move the counter in the right direction	Bug
Return values from functions without side effects should not be ignored	Bug
Values should not be uselessly incremented	Bug
Related "if/else if" statements and "cases" in a "switch" should not have the same condition	Bug
Objects should not be created to be dropped immediately without being used	Bug
Identical expressions should not be used on both sides of a binary operator	Bug
All code should be reachable	Bug

Signalling processes is security-sensitive Analyze your code

Security Hotspot Critical cwe

Signalling processes is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2009-0390
- CVE-2002-0839
- CVE-2008-1671

Sending signals without checking properly which process will receive it can cause a denial of service.

Ask Yourself Whether

- the PID of the process to which the signal will be sent is coming from an untrusted source. It could for example come from a world-writable file.
- users who are asking for the signal to be sent might not have the permission to send those signals.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- If the signal is sent because of a user's request. Check that the user is allowed to send this signal. You can for example forbid it if the user doesn't own the process.
- Secure the source from which the process PID is read.
- Run the process sending the signals with minimal permissions.

Sensitive Code Example





```
posix_kill(42, 42); // Sensitive
```

See

- MITRE, CWE-283 - Unverified Ownership

Available In:

sonarcloud | sonarqube

<p>Loops with at most one iteration should be refactored</p> <p> Bug</p>
<p>Short-circuit logic should be used to prevent null pointer dereferences in conditionals</p> <p> Bug</p>
<p>Variables should not be self-assigned</p> <p> Bug</p>
<p>Useless "if(true) {...}" and "if(false){...}" blocks should be removed</p> <p> Bug</p>
<p>All "catch" blocks should be able to</p>