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...

Code Smell

The names of methods with boolean return values should start with "is" or "has"
Code Smell

"php_sapi_name()" should not be used
Code Smell

Classes should not have too many lines of code
Code Smell

Deprecated features should not be used
Code Smell

Files should not contain inline HTML
Code Smell

Files should contain only one top-level class or interface each
Code Smell

Classes should not have too many fields
Code Smell

Track uses of "NOSONAR" comments
Code Smell

Statements should be on separate lines
Code Smell

Classes should not be coupled to too many other classes (Single Responsibility Principle)
Code Smell

"switch case" clauses should not have too many lines of code

Ternary operators should not be nested

Analyze your code

Code Smell | Major | confusing

Just because you *can* do something, doesn't mean you should, and that's the case with nested ternary operations. Nesting ternary operators results in the kind of code that may seem clear as day when you write it, but six months later will leave maintainers (or worse - future you) scratching their heads and cursing.

Instead, err on the side of clarity, and use another line to express the nested operation as a separate statement.

Noncompliant Code Example

```
function get_readable_status($is_running, $has_errors)
  return $is_running ? "Running" : ($has_errors ? "Fail
}
```

Compliant Solution

```
function get_readable_status($is_running, $has_errors)
  if ($is_running) {
    return "Running";
  }
  return $has_errors ? "Failure. " : "Succeeded ";
}
```

Exceptions

Exclusively chained shorthand ternary operators `?:` are excluded from this rule.

```
$result = $option1 ?: $option2 ?: 'default';
```

Available In:

sonarlint | sonarcloud | sonarqube

Code Smell

Assignments should not be made from within sub-expressions

Code Smell

Files should not have too many lines of code

Code Smell

Lines should not be too long

Code Smell

HTTP response headers should not be vulnerable to injection attacks

Vulnerability

"sleep" should not be called