
































-  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... 

be used to end lines	
More than one property should not be declared per statement	
The "var" keyword should not be used	
"<?php" and "<?=" tags should be used	
File names should comply with a naming convention	
Comments should not be located at the end of lines of code	
Local variable and function parameter names should comply with a naming convention	
Field names should comply with a naming convention	
Lines should not end with trailing whitespaces	
Files should contain an empty newline at the end	
Modifiers should be declared in the correct order	
An open curly brace should be located at the beginning of a line	

SHA-1 and Message-Digest hash algorithms should not be used in secure contexts

Analyze your code

 Vulnerability  Critical 

The MD5 algorithm and its successor, SHA-1, are no longer considered secure, because it is too easy to create hash collisions with them. That is, it takes too little computational effort to come up with a different input that produces the same MD5 or SHA-1 hash, and using the new, same-hash value gives an attacker the same access as if he had the originally-hashed value. This applies as well to the other Message-Digest algorithms: MD2, MD4, MD6, HAVAL-128, HMAC-MD5, DSA (which uses SHA-1), RIPEMD, RIPEMD-128, RIPEMD-160, HMACRIPEMD160.

Consider using safer alternatives, such as SHA-256, SHA-512 or SHA-3.

Noncompliant Code Example

```
$password = ...

if (md5($password) === '1f3870be274f6c49b3e31a0c6728957f') {
    [...]
}

if (sha1($password) === 'd0be2dc421be4fcd0172e5afceea3970e2f
    [...]
}
```

See

- [OWASP Top 10 2017 Category A6](#) - Security Misconfiguration
- [MITRE, CWE-328](#) - Reversible One-Way Hash
- [MITRE, CWE-327](#) - Use of a Broken or Risky Cryptographic Algorithm
- [SANS Top 25](#) - Porous Defenses
- [SHAttered](#) - The first concrete collision attack against SHA-1.

Deprecated

This rule is deprecated; use {rule:php:S4790} instead.

Available In:


An open curly brace should be located at the end of a line

 Code Smell


Tabulation characters should not be used

 Code Smell

Method and function names should comply with a naming convention

 Code Smell

Creating cookies with broadly defined "domain" flags is security-sensitive

 Security Hotspot