

- 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

More than one property should not be declared per statement

Code Smell

The "var" keyword should not be used

Code Smell

"<?php" and "<?=" tags should be used

Code Smell

File names should comply with a naming convention

Code Smell

Comments should not be located at the end of lines of code

Code Smell

Local variable and function parameter names should comply with a naming convention

Code Smell

Field names should comply with a naming convention

Code Smell

Lines should not end with trailing whitespaces

Code Smell

Files should contain an empty newline at the end

Code Smell

Modifiers should be declared in the correct order

Code Smell

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

Code Smell

Using Sockets is security-sensitive

Analyze your code

Security Hotspot Critical

Using sockets is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2011-178
- CVE-2017-5645
- CVE-2018-6597

Sockets are vulnerable in multiple ways:

- They enable a software to interact with the outside world. As this world is full of attackers it is necessary to check that they cannot receive sensitive information or inject dangerous input.
- The number of sockets is limited and can be exhausted. Which makes the application unresponsive to users who need additional sockets.

This rules flags code that creates sockets. It matches only the direct use of sockets, not use through frameworks or high-level APIs such as the use of http connections.

Ask Yourself Whether

- sockets are created without any limit every time a user performs an action.
- input received from sockets is used without being sanitized.
- sensitive data is sent via sockets without being encrypted.

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

Recommended Secure Coding Practices

- In many cases there is no need to open a socket yourself. Use instead libraries and existing protocols.
- Encrypt all data sent if it is sensitive. Usually it is better to encrypt it even if the data is not sensitive as it might change later.
- Sanitize any input read from the socket.
- Limit the number of sockets a given user can create. Close the sockets as soon as possible.

Sensitive Code Example


```
function handle_sockets($domain, $type, $protocol, $port, $b
    socket_create($domain, $type, $protocol); // Sensitive
    socket_create_listen($port, $backlog); // Sensitive
    socket_addrinfo_bind($addr); // Sensitive
    socket_addrinfo_connect($addr); // Sensitive
    socket_create_pair($domain, $type, $protocol, $fd);

    fsockopen($hostname); // Sensitive
    pfsockopen($hostname); // Sensitive
    stream_socket_server($local_socket); // Sensitive
    stream_socket_client($remote_socket); // Sensitive
    stream_socket_pair($domain, $type, $protocol); // Sensit
}
```

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

See

- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [MITRE, CWE-20](#) - Improper Input Validation
- [MITRE, CWE-400](#) - Uncontrolled Resource Consumption ('Resource Exhaustion')
- [MITRE, CWE-200](#) - Exposure of Sensitive Information to an Unauthorized Actor
- [SANS Top 25](#) - Risky Resource Management
- [SANS Top 25](#) - Porous Defenses

Deprecated

This rule is deprecated, and will eventually be removed.

Available In:

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](#)