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...

HTTP request redirections should not be open to forging attacks
Vulnerability

Deserialization should not be vulnerable to injection attacks
Vulnerability

Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks
Vulnerability

Database queries should not be vulnerable to injection attacks
Vulnerability

XML parsers should not be vulnerable to XXE attacks
Vulnerability

A secure password should be used when connecting to a database
Vulnerability

XPath expressions should not be vulnerable to injection attacks
Vulnerability

I/O function calls should not be vulnerable to path injection attacks
Vulnerability

LDAP queries should not be vulnerable to injection attacks
Vulnerability

OS commands should not be vulnerable to command injection attacks
Vulnerability

Class of caught exception should be defined

HTTP request redirections should not be open to forging attacks

Analyze your code

Vulnerability | Blocker | injection cwe sans-top25 owasp

User-provided data, such as URL parameters, POST data payloads, or cookies, should always be considered untrusted and tainted. Applications performing HTTP redirects based on tainted data could enable an attacker to redirect users to a malicious site to, for example, steal login credentials.

This problem could be mitigated in any of the following ways:

- Validate the user-provided data based on a whitelist and reject input not matching.
- Redesign the application to not perform redirects based on user-provided data.

Noncompliant Code Example

Symfony

```
public function redirect(Request $request)
{
  $url = $request->query->get('url');
  return $this->redirect($url); // Noncompliant
}

public function setLocatioHeader(Request $request)
{
  $url = $request->query->get('url');
  $response = new Response('Redirecting...', 302);
  $response->headers->set('Location', $url); // Noncomp
  return $response;
}
```

Laravel

```
public function redirect(Request $request)
{
  $url = $request->input('url');
  return $this->redirect($url); // Noncompliant
}

public function setLocatioHeader(Request $request)
{
  $url = $request->input('url');
  return response("", 302)
    ->header('Location', $url) // Noncompliant
}
```

Compliant Solution

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

Symfony

```
public function redirect(Request $request)
{
  $url = $request->query->get('url');
  $allowedUrls = ["/index", "/login", "/logout"];

  if (in_array($url, $allowedUrls, true)) {
    return $this->redirect($url);
  } else {
    $this->redirect("/");
  }
}
```

Laravel

```
public function redirect(Request $request)
{
  $url = $request->input('url');
  $allowedUrls = ["/index", "/login", "/logout"];

  if (in_array($url, $allowedUrls, true)) {
    return $this->redirect($url);
  } else {
    return redirect("/");
  }
}
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2017 Category A5 - Broken Access Control
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-601 - URL Redirection to Untrusted Site ('Open Redirect')
- SANS Top 25 - Risky Resource Management

Available In:

sonarcloud | sonarqube Developer Edition