

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- Code Smell
- 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...

at the beginning of a line
Code Smell
URIs should not be hardcoded
Code Smell
Class names should comply with a naming convention
Code Smell
Track uses of "TODO" tags
Code Smell
"file_uploads" should be disabled
Vulnerability
"enable_dl" should be disabled
Vulnerability
"session.use_trans_sid" should not be enabled
Vulnerability
"allow_url_fopen" and "allow_url_include" should be disabled
Vulnerability
"open_basedir" should limit file access
Vulnerability
Neither DES (Data Encryption Standard) nor DESede (3DES) should be used
Vulnerability
"exit(...)" and "die(...)" statements should not be used
Bug
Functions and variables should not be defined outside of classes
Code Smell

## Allowing requests with excessive content length is security-sensitive

Analyze your code

Security Hotspot Major cwe owasp

Rejecting requests with significant content length is a good practice to control the network traffic intensity and thus resource consumption in order to prevents DoS attacks.

### Ask Yourself Whether

- size limits are not defined for the different resources of the web application.
- the web application is not protected by **rate limiting** features.
- the web application infrastructure has limited resources.

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

### Recommended Secure Coding Practices

- For most of the features of an application, it is recommended to limit the size of requests to:
  - lower or equal to 8mb for file uploads.
  - lower or equal to 2mb for other requests.

It is recommended to customize the rule with the limit values that correspond to the web application.

### Sensitive Code Example

For **Symfony Constraints**:





```
use Symfony\Component\Validator\Constraints as Assert;
use Symfony\Component\Validator\Mapping\ClassMetadata;

class TestEntity
{
    public static function loadValidatorMetadata(ClassMetadata $metadata)
    {
        $metadata->addPropertyConstraint('upload', new Assert\MaxSize('100M', // Sensitive));
    }
}
```

For **Laravel Validator**:

```
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class TestController extends Controller
{
    public function test(Request $request)
```

Track lack of copyright and license headers
 Code Smell
Octal values should not be used
 Code Smell
Switch cases should end with an unconditional "break" statement
 Code Smell
Session-management cookies should not be persistent
 Vulnerability
Cryptographic RSA algorithms should always incorporate OAEP (Optimal Asymmetric Encryption Padding)

```
{
    $validatedData = $request->validate([
        'upload' => 'required|file', // Sensitive
    ]);
}
```

#### Compliant Solution

For [Symfony Constraints](#):

```
use Symfony\Component\Validator\Constraints as Assert;
use Symfony\Component\Validator\Mapping\ClassMetadata;

class TestEntity
{
    public static function loadValidatorMetadata(ClassM
    {
        $metadata->addPropertyConstraint('upload', new
            'maxSize' => '8M', // Compliant
        ));
    }
}
```

For [Laravel Validator](#):

```
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class TestController extends Controller
{
    public function test(Request $request)
    {
        $validatedData = $request->validate([
            'upload' => 'required|file|max:8000', // Co
        ]);
    }
}
```

#### See

- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [Owasp Cheat Sheet](#) - Owasp Denial of Service Cheat Sheet
- [OWASP Top 10 2017 Category A6](#) - Security Misconfiguration
- [MITRE, CWE-770](#) - Allocation of Resources Without Limits or Throttling
- [MITRE, CWE-400](#) - Uncontrolled Resource Consumption

Available In:

**sonardcloud**  **sonarqube** 