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...

Cipher algorithms should be robust
Vulnerability

Encryption algorithms should be used with secure mode and padding scheme
Vulnerability

Server hostnames should be verified during SSL/TLS connections
Vulnerability

Server certificates should be verified during SSL/TLS connections
Vulnerability

LDAP connections should be authenticated
Vulnerability

Cryptographic keys should be robust
Vulnerability

Weak SSL/TLS protocols should not be used
Vulnerability

Regular expressions should not be vulnerable to Denial of Service attacks
Vulnerability

Hashes should include an unpredictable salt
Vulnerability

Regular expressions should have valid delimiters
Bug

Regex lookahead assertions should not be contradictory
Bug

Back references in regular expressions should only refer to capturing groups that are matched before the reference

OS commands should not be vulnerable to command injection attacks

Analyze your code

Vulnerability | Blocker | injection cwe owasp sans-top25

Applications that allow execution of operating system commands from user-controlled data should control the command to execute, otherwise an attacker can inject arbitrary commands that will compromise the underlying operating system.

The mitigation strategy can be based on a list of authorized and safe commands to execute and when a shell is spawned to sanitize shell meta-characters. Keep in mind that when a single argument to the command is user-controlled and shell-metachars are sanitized, it can still lead to vulnerabilities if the attacker can inject a dangerous option supported by the command, such as `-exec` available with find, in that case, mark end of option processing on the command line using `--` (double-dash) or restrict options to only trusted values.

Noncompliant Code Example

An arbitrary command can be executed:

```
$cmd = $_GET["cmd"];

// If the command "/sbin/shutdown" is executed and the web s
// then the machine running the web server will be shut down

exec($cmd); // Noncompliant
```

Shell meta-characters can be used to execute additional commands in PHP functions that execute commands:

```
$arg = $_GET["arg"];

// the attacker can perform: "argvalue | /sbin/shutdown"

exec("/usr/bin/cat ".$arg); // Noncompliant
```

Compliant Solution

Restrict the command to be executed with an allowlist:

```
$cmd = $_GET["cmd"];

if ($cmd === "/usr/bin/stat" || $cmd === "/usr/bin/cat") {
   exec($cmd." /tmp/test.txt", $output);
}
```

Use `escapeshellarg` to escape shell meta-characters from a specific argument:

```
$arg = $_GET["arg"];

exec("/usr/bin/cat ".escapeshellarg($arg));
```

Bug

Regex boundaries should not be used in a way that can never be matched

Bug

Regex patterns following a possessive quantifier should not always fail

Bug

Assertion failure exceptions should not be ignored

Bug

References used in "foreach" loops should be "unset"

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP OS Command Injection Defense Cheat Sheet
- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-78 - Improper Neutralization of Special Elements used in an OS Command
- SANS Top 25 - Insecure Interaction Between Components

Available In:

sonarcloud | sonarqube Developer Edition