



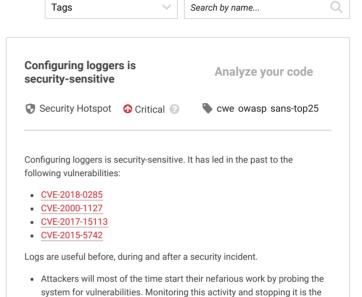


# PHP static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your PHP code







to understand what damage an attacker may have inflicted.

Logs are also a target for attackers because they might contain sensitive information. Configuring loggers has an impact on the type of information logged and how they are logged.

• In case of a successful attack, logs should contain enough information

first step to prevent an attack from ever happening.

This rule flags for review code that initiates loggers configuration. The goal is to guide security code reviews.

### Ask Yourself Whether

- unauthorized users might have access to the logs, either because they
  are stored in an insecure location or because the application gives
  access to them
- the logs contain sensitive information on a production server. This can happen when the logger is in debug mode.
- the log can grow without limit. This can happen when additional information is written into logs every time a user performs an action and the user can perform the action as many times as he/she wants.
- the logs do not contain enough information to understand the damage an attacker might have inflicted. The loggers mode (info, warn, error) might filter out important information. They might not print contextual information like the precise time of events or the server hostname.
- the logs are only stored locally instead of being backuped or replicated.

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

## Recommended Secure Coding Practices

- Check that your production deployment doesn't have its loggers in "debug" mode as it might write sensitive information in logs.
- Production logs should be stored in a secure location which is only accessible to system administrators.
- Configure the loggers to display all warnings, info and error messages.
   Write relevant information such as the precise time of events and the best name.
- Choose log format which is easy to parse and process automatically. It is
  important to process logs rapidly in case of an attack so that the impact



Variables should not be self-assigned



Useless "if(true) {...}" and "if(false){...}" blocks should be removed



All "catch" blocks should be able to catch exceptions

# Bug

Constructing arguments of system commands from user input is security-sensitive

Security Hotspot

is known and limited.

- Check that the permissions of the log files are correct. If you index the logs in some other service, make sure that the transfer and the service are secure too.
- Add limits to the size of the logs and make sure that no user can fill the disk with logs. This can happen even when the user does not control the logged information. An attacker could just repeat a logged action many times

Remember that configuring loggers properly doesn't make them bullet-proof. Here is a list of recommendations explaining on how to use your logs:

- Don't log any sensitive information. This obviously includes passwords and credit card numbers but also any personal information such as user names, locations, etc... Usually any information which is protected by law is good candidate for removal.
- Sanitize all user inputs before writing them in the logs. This includes
  checking its size, content, encoding, syntax, etc... As for any user input,
  validate using whitelists whenever possible. Enabling users to write what
  they want in your logs can have many impacts. It could for example use
  all your storage space or compromise your log indexing service.
- Log enough information to monitor suspicious activities and evaluate the impact an attacker might have on your systems. Register events such as failed logins, successful logins, server side input validation failures, access denials and any important transaction.
- Monitor the logs for any suspicious activity.

#### **Sensitive Code Example**

Basic PHP configuration:

```
function configure logging() {
 error reporting(E RECOVERABLE ERROR); // Sensitive
 error_reporting(32); // Sensitive
 ini_set('docref_root', '1'); // Sensitive
 ini_set('display_errors', '1'); // Sensitive
 ini_set('display_startup_errors', '1'); // Sensitive
 ini_set('error_log', "path/to/logfile"); // Sensitive
 ini_set('error_reporting', E_PARSE ); // Sensitive
 ini_set('error_reporting', 64); // Sensitive
 ini_set('log_errors', '0'); // Sensitive
 ini_set('log_errors_max_length', '512'); // Sensitive
 ini_set('ignore_repeated_errors', '1'); // Sensitive
 ini set('ignore repeated source', '1'); // Sensitive
 ini_set('track_errors', '0'); // Sensitive
 ini alter('docref root', '1'); // Sensitive
 ini_alter('display_errors', '1'); // Sensitive
 ini alter('display startup errors', '1'); // Sensitiv
 ini_alter('error_log', "path/to/logfile"); // Sensiti
 ini_alter('error_reporting', E_PARSE ); // Sensitive
 ini_alter('error_reporting', 64); // Sensitive
 ini_alter('log_errors', '0'); // Sensitive
 ini_alter('log_errors_max_length', '512'); // Sensiti
 ini_alter('ignore_repeated_errors', '1'); // Sensitiv
 ini_alter('ignore_repeated_source', '1'); // Sensitiv
 ini_alter('track_errors', '0'); // Sensitive
```

Definition of custom loggers with psr/log

## Exceptions

No issue will be raised for logger configuration when it follows recommended settings for production servers. The following examples are all valid:

```
ini_set('docref_root', '0');
ini_set('display_errors', '0');
ini_set('display_startup_errors', '0');

error_reporting(0);
ini_set('error_reporting', 0);

ini_set('log_errors', '1');
ini_set('log_errors_max_length', '0');
ini_set('ignore_repeated_errors', '0');
ini_set('ignore_repeated_source', '0');
ini_set('track_errors', '1');
```

#### See

- OWASP Top 10 2021 Category A9 Security Logging and Monitoring Failures
- OWASP Top 10 2017 Category A3 Sensitive Data Exposure
- OWASP Top 10 2017 Category A10 Insufficient Logging & Monitoring
- MITRE, CWE-117 Improper Output Neutralization for Logs
- MITRE, CWE-532 Information Exposure Through Log Files
- SANS Top 25 Porous Defenses

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