

php-async

github.com/true-async/php-async

Scanned 24 September 2026 at 10:22 · Report generated 29 October 2026 · PHP · 385 stars

OVERALL ASSESSMENT

High

Combined score 101 · 60 findings after filtering

Summary

This repository has a high vulnerability score of 101.0, which means there are several serious coding errors that could allow an attacker to crash your system or steal secrets. While the code does not appear to be intentionally malicious, it contains many common mistakes like using unsafe memory functions and executing commands with user input. These issues look more like ordinary insecure coding than signs of a hidden attack, but they still need to be fixed before the project can be used safely. The developer should address these specific problems immediately to prevent potential security breaches in production environments.

Written by a language model running locally on the scanning machine, from the findings in this report. The risk levels above and below were decided by the scoring model, not by this summary.

Risk, by type

MalDet scores two different questions separately, because they call for different responses: code can be badly written without being hostile, and hostile code is not always badly written.

VULNERABILITY

High (101)

Insecure coding an attacker could exploit. Usually mistakes, and usually fixable in place.

MALICIOUS PATTERN

Safe (0)

Signs the code may be trying to do harm: hidden payloads, credential access, persistence, exfiltration.

What each detector found

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
semgrep	Insecure coding patterns across many languages	8	40	0	48
yara	Signatures of malicious behaviour in source files	5	0	7	12

35 of 60 findings were rated more likely genuine than noise by the classifier.

Evidence

Identical findings are grouped, so a single rule firing many times takes one row. Ordered worst first. “Seen” is how many times that finding occurred.

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	semgrep	This expression points to memory that has been freed. This can lead to a segmentation fault or memory corruption.	2	/internal/allocator.c:52
HIGH	semgrep	Using variable interpolation `\${{...}}` with `github` context data in a `run:` step could allow an attacker to inject their own code into the runner. This would allow them to steal secrets and code. `github` context data can have arbitrary user input and should be treated as untrusted. Instead, use	1	/.github/workflows/debug-bugfix.yml:60
HIGH	semgrep	Executing non-constant commands. This can lead to command injection.	4	/fuzzy-tests/_harness/Steps.php:2720
HIGH	semgrep	By not specifying a USER, a program in the container may run as 'root'. This is a security hazard. If an attacker can control a process running as root, they may have control over the container. Ensure that the last USER in a Dockerfile is a USER other than 'root'.	1	/Dockerfile:105
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	5	/.github/workflows/debug-exec.yml
MEDIUM	semgrep	File name based on user input risks server-side request forgery.	5	/fuzzy-tests/_harness/freeze.php:77

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	GitHub Actions step uses a mutable tag or branch reference. Tags and branch names can be silently repointed by the action owner, enabling supply-chain attacks — as seen in the trivy-action and kics-github-action compromises. Pin the reference to a full 40-character commit SHA instead, e.g. `uses: ac	27	/.github/workflows/build-freebsd.yml:20
MEDIUM	semgrep	Calling `unserialize()` with user input in the pattern can lead to arbitrary code execution. Consider using JSON or structured data approaches (e.g. Google Protocol Buffers).	1	/fuzzy-tests/_peers/forked_peer.php:23
MEDIUM	semgrep	Using user input when deleting files with `unlink()` is potentially dangerous. A malicious actor could use this to modify or access files they have no right to.	7	/fuzzy-tests/_harness/ChaosNet.php:348
LOW	yara	YARA rule matched: detect_downloader — Downloads a remote resource - common and low-signal alone; see detect_download_and_execute for the composite version	7	/Dockerfile

How to read this report

- **This is a static scan.** The code was read, never executed. Behaviour that only appears at runtime is outside what any of these detectors can see.
- **Findings are not all equally real.** A classifier rates each finding on how likely it is to be genuine rather than noise, and that rating weights the scores. Measured against a hand-reviewed sample it reached precision 0.52 and recall 0.62 — useful for ranking, not a verdict on any single finding.
- **A high score is not proof of malice.** Security tools, penetration-testing frameworks and malware research repositories contain the same patterns as real malware by design. The two risk types above exist to keep that distinction visible.
- **Absence of findings is not proof of safety.** A detector that is not installed, or a language none of them parse, produces silence rather than a warning.
- **This scan is a snapshot.** The repository may have changed since 24 September 2026.