

shopware

github.com/shopware/shopware

Scanned 24 September 2026 at 07:17 · Report generated 29 October 2026 · PHP · 3,430 stars

OVERALL ASSESSMENT

Medium

Combined score 35 · 9 findings after filtering

Summary

This repository has been flagged as having a very low vulnerability risk score of 0, meaning there are no obvious coding errors that could be exploited by attackers right now. However, the code does show signs of malicious intent with a medium pattern score of 35, which suggests the developer may have intentionally included dangerous tools rather than accidentally leaving them in. The most concerning issues involve suspicious startup scripts and npm preinstall commands that are commonly used to deliver malware, along with evidence of base64-encoded payloads being executed directly. While some low-severity findings like reading environment variables or running processes silently could be accidental mistakes, the presence of these specific patterns strongly indicates that the code is not safe for general use without further review.

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

Safe (0)

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

MALICIOUS PATTERN

Medium (35)

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
guarddog	Malicious package and install-time behaviour	3	1	3	7

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
yara	Signatures of malicious behaviour in source files	2	0	0	2

7 of 9 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	guarddog	Detects suspicious autostart persistence mechanisms [threat-filesystem-autostart]	2	/src/ Administration/ Resources/app/ administration/src/ module/sw-import- export/component/ sw-import-export- activity/ index.js:361
HIGH	guarddog	Detects npm preinstall scripts, which are almost exclusively used for malware delivery [threat-npm-preinstall-script]	1	/src/ Administration/ Resources/app/ administration/ package.json:27
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	1	/.github/bin/js/ zugferd- compliance.ts
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	1	/.github/workflows/ lint-actions.yml
MEDIUM	guarddog	Detects LOLBAS usage in install hooks (execution and network tools) [threat-process-hooks]	1	/src/ Administration/ Resources/app/ administration/ package.json:27
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	1	/.github/bin/js/ zugferd- compliance.ts:118
LOW	guarddog	Detects fully silent process execution (suppressing all output channels) [threat-process-spawn-silent]	2	/src/Storefront/ Resources/app/ storefront/build/ start-hot- reload.js:266

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.