

secure-repo

github.com/step-security/secure-repo

Scanned 24 September 2026 at 09:34 · Report generated 29 October 2026 · Go · 332 stars

OVERALL ASSESSMENT

Critical

Combined score 1311 · 776 findings after filtering

Summary

This repository has been flagged as having critical vulnerabilities because it contains insecure coding patterns that attackers could exploit to run malicious code or steal secrets. While the scan also identified some signs of potentially malicious intent, these findings do not necessarily mean the code is intentionally harmful; they mostly point to common mistakes like using unsafe variable interpolation and exposing secrets in environment variables. The most dangerous issues involve running code as root, which allows attackers to gain full control over the container, and executing base64-encoded payloads that could be used to hide malicious instructions. Although there are some medium-severity findings about supply chain attacks and dependency management, these represent standard practices rather than intentional attempts to harm the system. Overall, while the repository is not completely safe due to the critical vulnerabilities, the presence of many low-confidence findings suggests the code may have been written by a developer who made errors rather than an attacker trying to compromise the system.

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

Critical (1300)

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

MALICIOUS PATTERN

Low (11)

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	22	747	0	769
yara	Signatures of malicious behaviour in source files	5	0	0	5
guarddog	Malicious package and install-time behaviour	1	0	1	2

770 of 776 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	Using variable interpolation `\${{...}}` with `github` context data in a `actions/github-script`'s `script:` 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	1	/.github/workflows/kbanalysis.yml:43
HIGH	guarddog	Detects base64 decoding followed by code execution [threat-runtime-obfuscation-base64exec]	1	/Automate-PR/dist/src/index.js:12180
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	20	/testfiles/pinactions/input/action.yml:148
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:19
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	2	/Automate-PR/dist/src/index.js

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	2	/testfiles/ dockerfiles/ output/ Dockerfile- multiple- images
HIGH	yara	YARA rule matched: detect_obfuscated_string_execution — Reconstructs a string character-by-character or from hex escapes, then immediately executes it - fromCharCode/chr() alone are common in ordinary text-processing code, so this requires both the reconstruction and an execution call together	1	/Automate- PR/dist/src/ index.js
MEDIUM	semgrep	This Dependabot configuration does not set a cooldown period. Newly published packages can be malicious or unstable. Add a `cooldown` block with `default-days: 7` to each `package-ecosystem` entry under `updates` to wait 7 days before proposing updates to newly published package versions. Reference:	3	/.github/ dependabot. yaml:9
MEDIUM	semgrep	A secret is exposed in the workflow-level `env:` block, making it available to every job and step in this workflow — including any untrusted code run in pull-request workflows. Scope secrets as narrowly as possible: prefer step-level `env:` so the secret is only available where it is actually needed	2	/testfiles/ joblevelper mskb/input/ workflow- level- env.yaml:5
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	742	/testfiles/ pinactions/ output/ pinusingmap. yaml:25
LOW	guarddog	Detects heavy obfuscation techniques commonly used by malware [threat-runtime-obfuscation]	1	/Automate- PR/dist/src/ index.js:145 15

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.