

server

github.com/buggregator/server

Scanned 24 September 2026 at 09:33 · Report generated 29 October 2026 · Go · 1,077 stars

OVERALL ASSESSMENT

High

Combined score 172 · 50 findings after filtering

Summary

This repository has a high vulnerability score of 169, which means there are many insecure coding patterns that an attacker could easily exploit to break into the system. While the malicious-pattern score is low and suggests the code isn't intentionally harmful, the high vulnerability score indicates that ordinary mistakes in how the code is written are likely causing these risks. Most of the issues found are standard security flaws like SQL injection or missing TLS settings rather than signs of a hidden malware attack. The developer should fix the identified insecure patterns immediately to ensure the application remains safe for users and compliant with best practices.

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 (169)

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

MALICIOUS PATTERN

Low (3)

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	14	33	0	47
guarddog	Malicious package and install-time behaviour	0	0	3	3

40 of 50 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	Detected string concatenation with a non-literal variable in a "database/sql" Go SQL statement. This could lead to SQL injection if the variable is user-controlled and not properly sanitized. In order to prevent SQL injection, use parameterized queries or prepared statements instead. You can use pre	11	/internal/server/http/attachments.go:38
HIGH	semgrep	Detected non-static command inside Command. Audit the input to 'exec.Command'. If unverified user data can reach this call site, this is a code injection vulnerability. A malicious actor can inject a malicious script to execute arbitrary code.	1	/modules/vardumper/php_process.go:55
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'.	2	/e2e/mcp/Dockerfile:12
MEDIUM	semgrep	An HTTP redirect was found to be crafted from user-input `r`. This can lead to open redirect vulnerabilities, potentially allowing attackers to redirect users to malicious web sites. It is recommend where possible to not allow user-input to craft the redirect URL. When user-input is necessary to cra	1	/internal/auth/handlers.go:70
MEDIUM	semgrep	String-formatted SQL query detected. This could lead to SQL injection if the string is not sanitized properly. Audit this call to ensure the SQL is not manipulable by external data.	2	/internal/storage/migrator.go:87

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	`MinVersion` is missing from this TLS configuration. By default, as of Go 1.22, TLS 1.2 is currently used as the minimum. General purpose web applications should default to TLS 1.3 with all other protocols disabled. Only where it is known that a web server must support legacy clients with unsupport	4	/modules/proxy/server.go:110
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	19	/.github/workflows/release.yml:151
MEDIUM	semgrep	Checks for disabling of TLS/SSL certificate verification. This should only be used for debugging purposes because it leads to vulnerability to MTM attacks.	2	/modules/sentry/source_ext.go:43
MEDIUM	semgrep	Found an HTTP server without TLS. Use 'http.ListenAndServeTLS' instead. See https://golang.org/pkg/net/http/#ListenAndServeTLS for more information.	1	/internal/app/app.go:125
MEDIUM	semgrep	Detected directly writing or similar in 'http.ResponseWriter.write()'. This bypasses HTML escaping that prevents cross-site scripting vulnerabilities. Instead, use the 'html/template' package and render data using 'template.Execute()'.	3	/internal/server/http/attachments.go:124
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	/php/varddumper/parser.php:52
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	3	/internal/app/config.go:198

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.