

goro

github.com/KarpelesLab/goro

Scanned 24 September 2026 at 09:39 · Report generated 29 October 2026 · PHP · 695 stars

OVERALL ASSESSMENT

Medium

Combined score 75 · 32 findings after filtering

Summary

This repository has a medium vulnerability score of 72, which means there are several insecure coding patterns that an attacker could potentially exploit if they find and use the code. The most serious issues involve using MD5 and SHA1 hash algorithms, which are now considered too weak for protecting data, and missing security settings like TLS 1.3 in some network configurations. While these problems represent ordinary mistakes rather than signs of malicious intent, they still require immediate attention because they could allow attackers to steal sensitive information or break critical parts of the system. The low score from other tools suggests that the code does not appear to be trying to do anything harmful, but the high number of medium-risk findings means the code is not ready for production use without significant fixes.

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

Medium (72)

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	6	23	0	29

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
guarddog	Malicious package and install-time behaviour	0	0	3	3

24 of 32 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	`Clean` is not intended to sanitize against path traversal attacks. This function is for finding the shortest path name equivalent to the given input. Using `Clean` to sanitize file reads may expose this application to path traversal attacks, where an attacker could access arbitrary files on the ser	1	/core/ phpctx/ http.go:20
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.	5	/ext/ standard/ fs.go:1835
MEDIUM	semgrep	Downcasting or changing sign of an integer with `\$CAST_METHOD` method	3	/core/ zscanf.go:208
MEDIUM	semgrep	The package `net/http/cgi` is on the import blacklist. The package is vulnerable to httpoxy attacks (CVE-2015-5386). It is recommended to use `net/http` or a web framework to build a web application instead.	1	/sapi/php-cgi/ main.go:35
MEDIUM	semgrep	Detected a network listener listening on 0.0.0.0 or an empty string. This could unexpectedly expose the server publicly as it binds to all available interfaces. Instead, specify another IP address that is not 0.0.0.0 nor the empty string.	1	/sapi/php-httpd/ main.go:38
MEDIUM	semgrep	Detected MD5 hash algorithm which is considered insecure. MD5 is not collision resistant and is therefore not suitable as a cryptographic signature. Use SHA256 or SHA3 instead.	5	/ext/ standard/ crypt_md5.go:35

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Detected SHA1 hash algorithm which is considered insecure. SHA1 is not collision resistant and is therefore not suitable as a cryptographic signature. Use SHA256 or SHA3 instead.	2	/ext/ standard/ strings.go: 771
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	1	/ext/curl/ curl.go:347
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	/ext/ mysqli/ mysqli.go:3 82
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	5	/.github/ workflows/ ci.yml:19
MEDIUM	semgrep	Do not use `math/rand`. Use `crypto/rand` instead.	3	/ext/gmp/ misc.go:9
LOW	guarddog	Detects active collection of system information (hostname, platform, architecture, user) [threat-runtime-system-info]	3	/ext/ standard/ uname.go:27

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.