

Resin

github.com/Resinat/Resin

Scanned 24 September 2026 at 09:40 · Report generated 29 October 2026 · Go · 2,372 stars

OVERALL ASSESSMENT

High

Combined score 133 · 48 findings after filtering

Summary

The repository Resinat/Resin has a high vulnerability risk score of 120.0, which means the code contains many insecure patterns that an attacker could easily exploit to break things. While there are also some signs of potentially malicious intent with a low score of 14.0, these appear to be ordinary coding mistakes rather than intentional attacks. The most serious issues involve SQL injection risks from user-controlled data and dangerous dynamic ClientTrace functions that could allow attackers to bypass security controls. Additionally, the code uses methods like math/rand instead of the safer crypto/rand, and there are several references to downloading remote resources which could be unsafe if not handled correctly. Overall, while some issues look like standard errors, the combination of these vulnerabilities makes it important for developers to review and fix them before using this code in production.

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 (120)

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

MALICIOUS PATTERN

Low (14)

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	33	0	41
guarddog	Malicious package and install-time behaviour	0	4	2	6
yara	Signatures of malicious behaviour in source files	0	0	1	1

40 of 48 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	6	/internal/ state/ schema.go:62
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	/ Dockerfile:4 8
MEDIUM	semgrep	Detected a potentially dynamic ClientTrace. This occurred because semgrep could not find a static definition for 'upstreamTrace.clientTrace()'. Dynamic ClientTraces are dangerous because they deserialize function code to run when certain Request events occur, which could lead to code being run witho	2	/internal/ proxy/ reverse.go:3 32

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Detected a potentially dynamic ClientTrace. This occurred because semgrep could not find a static definition for 'reporter.clientTrace()'. Dynamic ClientTraces are dangerous because they deserialize function code to run when certain Request events occur, which could lead to code being run without yo	1	/internal/proxy/reverse.go:337
MEDIUM	semgrep	Downcasting or changing sign of an integer with `uint16` method	1	/internal/outbound/secure_dns.go:223
MEDIUM	guarddog	Detects URLs to URL shorteners, file sharing, and suspicious services [threat-network-outbound-shady-links]	4	/webui/src/features/platforms/PlatformAccessPanel.tsx:180
MEDIUM	semgrep	ReverseProxy can remove headers added by Director. Consider using ReverseProxy.Rewrite instead of ReverseProxy.Director.	1	/internal/proxy/reverse.go:324
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.	6	/internal/state/migrate.go:152
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	17	/.github/workflows/release.yml:217
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()'.	1	/internal/proxy/errors.go:93
MEDIUM	semgrep	Do not use `math/rand`. Use `crypto/rand` instead.	4	/internal/probe/manager.go:6

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	1	/internal/config/env.go:138
LOW	guarddog	Detects access to sensitive files (credentials, configs, keys) [threat-filesystem-read]	1	/cmd/resin/app_runtime.go:47
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	1	/docker-compose.yml.example

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.