

merlin

github.com/Ne0nd0g/merlin

Scanned 18 September 2026 at 19:38 · Report generated 29 October 2026 · Go · 5,605 stars

OVERALL ASSESSMENT

Medium

Combined score 60 · 36 findings after filtering

Summary

This repository has a medium vulnerability score of 56, which means there are several insecure coding patterns that an attacker could exploit to break into the system. The scan also found some signs of potentially malicious intent, but these findings look more like ordinary mistakes or weak defaults rather than intentional attacks. Most of the issues involve common security practices such as using untrusted user input, missing encryption settings, and relying on default Docker configurations instead of explicitly restricting permissions. While there are a few specific code injection risks and potential supply chain vulnerabilities, they do not appear to be part of a coordinated attack plan targeting this project. Overall, the code is likely safe to use if developers apply basic security updates to fix the identified issues.

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 (56)

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

MALICIOUS PATTERN

Low (4)

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	3	24	0	27

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
yara	Signatures of malicious behaviour in source files	5	0	4	9

22 of 36 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	Untrusted input could be used to tamper with a web page rendering, which can lead to a Cross-site scripting (XSS) vulnerability. XSS vulnerabilities occur when untrusted input executes malicious JavaScript code, leading to issues such as account compromise and sensitive information leakage. To preve	1	/pkg/servers/http/handler.go:136
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	/pkg/modules/sharpgen/sharpgen.go:25
HIGH	yara	YARA rule matched: detect_process_injection — Windows process-injection API sequence	1	/data/modules/windows/x64/powershell/detection/Get-InjectedThread.json
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:17
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	/data/modules/darwin/x64/python/enumeration/venator.json

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	3	/data/modules/linux/x64/bash/persistence/CrontabPersistence.json
MEDIUM	semgrep	Detected RC4 cipher algorithm which is insecure. The algorithm has many known vulnerabilities. Use AES instead.	1	/pkg/transformer/encrypters/rc4/rc4.go:55
MEDIUM	semgrep	Variable `msg` is likely modified and later used on error. In some cases this could result in panics due to a nil dereference	1	/pkg/services/message/message.go:216
MEDIUM	semgrep	Downcasting or changing sign of an integer with `uint32` method	1	/pkg/services/job/job.go:366
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 unsuppor	2	/pkg/servers/http/http.go:568
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	16	/.github/workflows/go_main.yml:45
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	/pkg/servers/http/handler.go:136
MEDIUM	semgrep	Do not use `math/rand`. Use `crypto/rand` instead.	2	/pkg/core/core.go:24

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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	4	/README.MD

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 18 September 2026.