

agent-router

github.com/theagentrouter/agent-router

Scanned 07 October 2026 at 10:10 · Report generated 29 October 2026 · Go · 2,192 stars

OVERALL ASSESSMENT

Medium

Combined score 77 · 89 findings after filtering

Summary

This repository has a medium vulnerability score of 74, which means there are several insecure coding patterns that an attacker could exploit to break into your system. The most serious issues involve gRPC connections that lack encryption and secrets being passed to workflows without proper restrictions, allowing attackers to tamper with data or steal credentials. While some findings suggest the code might be malicious, many of these are actually common mistakes in Go development rather than intentional attacks, such as using variable interpolation or writing directly to HTTP response handlers. The low score for malicious patterns indicates that the code does not appear to be intentionally harmful, but it still requires attention to fix the security gaps before deployment.

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 (74)

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
yara	Signatures of malicious behaviour in source files	9	0	54	63

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
semgrep	Insecure coding patterns across many languages	6	17	0	23
guarddog	Malicious package and install-time behaviour	0	0	3	3

11 of 89 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	Found an insecure gRPC connection using 'insecure.NewCredentials()'. This creates a connection without encryption to a gRPC server. A malicious attacker could tamper with the gRPC message, which could compromise the machine. Instead, establish a secure connection with an SSL certificate using the 'g	1	/cmd/extproc/mainlib/admin.go:34
HIGH	semgrep	Found an insecure gRPC server without 'grpc.Creds()' or options with credentials. This allows for a connection without encryption to this server. A malicious attacker could tamper with the gRPC message, which could compromise the machine. Include credentials derived from an SSL certificate in order	2	/cmd/aigw/run.go:177
HIGH	semgrep	This workflow uses `secrets: inherit` to pass all of the calling workflow's secrets to a reusable workflow. This violates the principle of least privilege because the called workflow receives access to every secret in the repository, not just the ones it needs. If the called workflow is compromised	1	/.github/workflows/release.yaml:11
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	1	/.github/workflows/docker_build_job.yaml:78

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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	/cmd/aigw/stdio2http.go:51
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	9	/site/versioned_docs/version-0.6/cli/installation.md
MEDIUM	semgrep	RegExp() called with a `k` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled in	1	/site/src/plugins/llmsTxtPlugin.ts:75
MEDIUM	semgrep	This .npmrc does not set a minimum release age or sets it too low. Newly published packages can be malicious or unstable. Add `min-release-age = 7` to wait 7 days before resolving newly published package versions. Added in: v11.10 Reference: https://github.blog/changelog/2026-02-18-npm-bulk-trusted-	1	/site/.npmrc:1
MEDIUM	semgrep	When working with web applications that involve rendering user-generated content, it's important to properly escape any HTML content to prevent Cross-Site Scripting (XSS) attacks. In Go, the `text/template` package does not automatically escape HTML content, which can leave your application vuln	1	/internal/autoconfig/config.go:16
MEDIUM	semgrep	Detection of dangerouslySetInnerHTML from non-constant definition. This can inadvertently expose users to cross-site scripting (XSS) attacks if this comes from user-provided input. If you have to use dangerouslySetInnerHTML, consider using a sanitization library such as DOMPurify to sanitize your HT	3	/site/src/components/ApiField.tsx:77

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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()'.	5	/internal/mcpproxy/modern.go:707
MEDIUM	semgrep	Detected possible user input going into a `path.join` or `path.resolve` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	6	/site/src/plugins/latestBlogsPlugin.ts:35
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	3	/cmd/aigw/main.go:129
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	54	/site/versioned_docs/version-0.7/getting-started/connect-providers/azure-openai.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 07 October 2026.