

ai-gateway

github.com/envoyproxy/ai-gateway

Scanned 29 July 2026 at 15:18 · Report generated 29 October 2026 · Go · 1,477 stars

This repository has since been renamed or transferred. GitHub now serves it under a different name, so this report describes the repository as it stood at the scan date under its former name.

OVERALL ASSESSMENT

Medium

Combined score 60 · 41 findings after filtering

Summary

This repository has a medium vulnerability score of 60, which means there are several insecure coding patterns that an attacker could exploit to break into your system. The scan did not find any signs of malicious intent, so the issues appear to be ordinary mistakes rather than deliberate attacks. However, because the code contains many unencrypted connections and potential security flaws, it is not safe to use without fixing these problems first. You should treat this as a high-risk project until you have addressed all the flagged issues in your own codebase.

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 (60)

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

MALICIOUS PATTERN

Safe (0)

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	4	24	12	40
bandit	Insecure coding patterns in Python source	0	1	0	1

4 of 41 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 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	/internal/testing/testotel/collector.go:98
HIGH	semgrep	JWT token detected	1	/examples/mcp/mcp_oauth_example.yaml:6
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
MEDIUM	semgrep	Checks for requests sent via http.NewRequest to http:// URLs. This is dangerous because the server is attempting to connect to a website that does not encrypt traffic with TLS. Instead, send requests only to https:// URLs.	1	/internal/testing/ollama.go:62

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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 <code>`text/template`</code> package does not automatically escape HTML content, which can leave your application vuln	1	/internal/ autoconfig/ config.go:16
MEDIUM	semgrep	Detection of <code>dangerouslySetInnerHTML</code> 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 <code>dangerouslySetInnerHTML</code> , consider using a sanitization library such as <code>DOMPurify</code> to sanitize your HT	3	/site/src/ components/ ApiField.tsx :47
MEDIUM	semgrep	Detected directly writing or similar in <code>'http.ResponseWriter.write()'</code> . This bypasses HTML escaping that prevents cross-site scripting vulnerabilities. Instead, use the <code>'html/template'</code> package and render data using <code>'template.Execute()'</code> .	4	/internal/ mcpproxy/ handlers.go: 176
MEDIUM	bandit	Possible binding to all interfaces.	1	/tests/ internal/ testopeninf erence/ openai_proxy .py:147
MEDIUM	semgrep	Detected possible user input going into a <code>`path.join`</code> or <code>`path.resolve`</code> 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.	3	/site/src/ plugins/ latestBlogs Plugin.ts:27
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain <code>`setuid`</code> or <code>`setgid`</code> binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	12	/examples/ inference- pool/with- annotations. yaml:128

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	semgrep	When running containers in Kubernetes, it's important to ensure that they are properly secured to prevent privilege escalation attacks. One potential vulnerability is when a container is allowed to run applications as the root user, which could allow an attacker to gain access to sensitive resou	12	/examples/monitoring/monitoring.yaml:115

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 29 July 2026.