

lan-proxy-gateway

github.com/Tght1211/lan-proxy-gateway

Scanned 24 September 2026 at 10:06 · Report generated 29 October 2026 · Go · 532 stars

OVERALL ASSESSMENT

Medium

Combined score 40 · 15 findings after filtering

Summary

This repository presents a medium-level security risk of 34, which means an attacker could potentially exploit some insecure coding patterns to compromise the system. While the code does not appear to be intentionally malicious based on the low malicious-pattern score, there are several significant vulnerabilities that need immediate attention. The most critical issues involve unsafe command execution where user input is not properly validated before running commands or scripts, which allows attackers to inject harmful code. Additionally, the application lacks proper security settings like HTTPS and public key pinning, leaving it vulnerable to man-in-the-middle attacks. There are also signs of potential supply chain risks, such as mutable tags in GitHub Actions steps that could be hijacked by others. Finally, some files contain dangerous patterns that download and execute remote resources, which is a severe threat if the code is used without strict controls. These findings represent ordinary insecure coding practices rather than malicious intent, but they require careful review and remediation before the repository can be considered safe for production use.

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 (34)

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

MALICIOUS PATTERN

Low (6)

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	6	0	10
yara	Signatures of malicious behaviour in source files	3	0	0	3
guarddog	Malicious package and install-time behaviour	0	2	0	2

7 of 15 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 non-static command inside Exec. Audit the input to 'syscall.Exec'. 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/ elevate_unix.go:22
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.	3	/internal/ platform/ platform.go:66
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	2	/ README_EN.md
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	1	/install.sh
MEDIUM	semgrep	The application's App Transport Security (ATS) configuration does not leverage the in-built public key pinning mechanisms. The application should consider leverage ATS public key pinning to ensure that the application only communicates to serves with an allow-listed certificate (and public key). By	1	/macos/ Info.plist:3

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	guarddog	Detects URLs to suspicious domains often used for exfiltration or C2 [threat-network-exfiltration]	2	/cmd/ update.go:38
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. <code>`uses: ac</code>	3	/.github/ workflows/ release.yml:43
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> .	1	/internal/ httpproxy/ pac.go:49
MEDIUM	semgrep	Do not use <code>`math/rand`</code> . Use <code>`crypto/rand`</code> instead.	1	/internal/ gateway/ natdiag.go:7

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.