

cliamp

github.com/bjarneo/cliamp

Scanned 29 October 2026 at 10:21 · Report generated 29 October 2026 · Go · 4,540 stars

OVERALL ASSESSMENT

High

Combined score 86 · 51 findings after filtering

Summary

32 findings likely need attention, 10 look like noise. Most notable: Missing mutex unlock (`p.mu`` variable) before returning from a function. This could re... (99% confidence).

Generated from finding counts. A plain-English summary was not available for this scan.

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 (80)

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	5	33	0	38
yara	Signatures of malicious behaviour in source files	9	0	1	10
guarddog	Malicious package and install-time behaviour	0	1	2	3

32 of 51 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	Missing mutex unlock (`p.mu` variable) before returning from a function. This could result in panics resulting from double lock operations	1	/external/local/provider.go:105
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/aur.yml:20
HIGH	semgrep	Generic Secret detected	1	/external/tidal/creds.go:16
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.	2	/cmd/protocol.go:116
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	9	/site/og-image.svg
MEDIUM	semgrep	This GitHub Actions workflow file uses `workflow_run` and checks out code from the incoming pull request. When using `workflow_run`, the Action runs in the context of the target repository, which includes access to all repository secrets. Normally, this is safe because the Action only runs code from	1	/.github/workflows/aur.yml:13
MEDIUM	guarddog	Detects URLs to suspicious domains often used for exfiltration or C2 [threat-network-exfiltration]	1	/external/radio/channels.go:19

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	An HTTP redirect was found to be crafted from user-input <code>`r`</code> . This can lead to open redirect vulnerabilities, potentially allowing attackers to redirect users to malicious web sites. It is recommend where possible to not allow user-input to craft the redirect URL. When user-input is necessary to cra	1	/external/spotify/session.go:430
MEDIUM	semgrep	Detected MD5 hash algorithm which is considered insecure. MD5 is not collision resistant and is therefore not suitable as a cryptographic signature. Use SHA256 or SHA3 instead.	4	/luaplugin/api_crypto.go:18
MEDIUM	semgrep	<code>`MinVersion`</code> 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	1	/internal/httpclient/client.go:236
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>	19	/.github/workflows/release.yml:34
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	/external/ytmusic/session.go:235
MEDIUM	semgrep	Do not use <code>`math/rand`</code> . Use <code>`crypto/rand`</code> instead.	3	/playlist/playlist.go:6

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted	3	/site/index.html:618
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	2	/external/lyrion/client.go:121
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	/site/install.sh

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 October 2026.