

picoclaw

github.com/sipeed/picoclaw

Scanned 24 September 2026 at 09:16 · Report generated 29 October 2026 · Go · 30,011 stars

OVERALL ASSESSMENT

High

Combined score 257 · 101 findings after filtering

Summary

This repository has a high vulnerability score of 244.0, which means there are many dangerous patterns in the code that an attacker could easily exploit if they find and use them. While the malicious-pattern score is low, suggesting the code isn't intentionally trying to harm others, the high vulnerability score indicates serious issues like SQL injection risks, path traversal attacks, and potential command injection vulnerabilities. The scan also found several other security concerns, including a dangerous workflow that runs as root, a file download followed by execution, and improper handling of sensitive data. These findings look more like ordinary insecure coding mistakes than signs of malicious intent, but they are still critical enough to prevent the repository from being used safely without significant changes.

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

High (244)

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

MALICIOUS PATTERN

Low (12)

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	24	68	0	92

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
yara	Signatures of malicious behaviour in source files	3	0	4	7
guarddog	Malicious package and install-time behaviour	1	1	0	2

74 of 101 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 string concatenation with a non-literal variable in a "database/sql" Go SQL statement. This could lead to SQL injection if the variable is user-controlled and not properly sanitized. In order to prevent SQL injection, use parameterized queries or prepared statements instead. You can use pre	1	/pkg/seahorse/schema.go:203
HIGH	semgrep	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the path which the application uses to access files. If the application does not validate user input and sanitize file paths, sensitive files such as confi	1	/pkg/channels/pico/pico.go:918
HIGH	guarddog	Detects base64 decoding followed by code execution [threat-runtime-obfuscation-base64exec]	1	/pkg/auth/oauth.go:71
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.yml:126

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	semgrep	`Clean` is not intended to sanitize against path traversal attacks. This function is for finding the shortest path name equivalent to the given input. Using `Clean` to sanitize file reads may expose this application to path traversal attacks, where an attacker could access arbitrary files on the ser	1	/web/backend/embed.go:56
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	8	/.github/workflows/create-tag.yml:50
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.	11	/pkg/mcp/manager.go:404
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	/docker/Dockerfile:34
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	/docker/Dockerfile.heavy
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	1	/web/backend/api/startup.go

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	This Dependabot configuration does not set a cooldown period. Newly published packages can be malicious or unstable. Add a `cooldown` block with `default-days: 7` to each `package-ecosystem` entry under `updates` to wait 7 days before proposing updates to newly published package versions. Reference:	3	/.github/dependabot.yml:6
MEDIUM	guarddog	Detects access to sensitive files (credentials, configs, keys) [threat-filesystem-read]	1	/pkg/isolation/runtime.go:229
MEDIUM	semgrep	The Origin header in the HTTP WebSocket handshake is used to guarantee that the connection accepted by the WebSocket is from a trusted origin domain. Failure to enforce can lead to Cross Site Request Forgery (CSRF). As per "gorilla/websocket" documentation: "A CheckOrigin function should carefully v	1	/pkg/channels/pico/pico.go:1013
MEDIUM	semgrep	Detected 'Fprintf' or similar writing to 'http.ResponseWriter'. This bypasses HTML escaping that prevents cross-site scripting vulnerabilities. Instead, use the 'html/template' package to render data to users.	1	/web/backend/api/oauth.go:516
MEDIUM	semgrep	A session cookie was detected without setting the 'Secure' flag. The 'secure' flag for cookies prevents the client from transmitting the cookie over insecure channels such as HTTP. Set the 'Secure' flag by setting 'Secure' to 'true' in the Options struct.	2	/web/backend/middleware/launcher_dashboard_auth.go:104
MEDIUM	semgrep	Detected SHA1 hash algorithm which is considered insecure. SHA1 is not collision resistant and is therefore not suitable as a cryptographic signature. Use SHA256 or SHA3 instead.	4	/pkg/evolution/pattern_clusterer.go:722
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.	3	/pkg/channels/weixin/media.go:790

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	String-formatted SQL query detected. This could lead to SQL injection if the string is not sanitized properly. Audit this call to ensure the SQL is not manipulable by external data.	3	/pkg/seahorse/store.go:1555
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/channels/irc/irc.go:82
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	41	/.github/workflows/nightly.yml:90
MEDIUM	semgrep	Checks for disabling of TLS/SSL certificate verification. This should only be used for debugging purposes because it leads to vulnerability to MTM attacks.	1	/pkg/channels/mqtt/mqtt.go:83
MEDIUM	semgrep	Detected a possible denial-of-service via a zip bomb attack. By limiting the max bytes read, you can mitigate this attack. `io.CopyN()` can specify a size.	1	/pkg/updater/updater.go:582
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()'.	2	/web/backend/api/auth.go:330
MEDIUM	semgrep	Do not use `math/rand`. Use `crypto/rand` instead.	4	/pkg/config/config.go:7

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 24 September 2026.