

tuios

github.com/Gaurav-Gosain/tuios

Scanned 24 September 2026 at 10:04 · Report generated 29 October 2026 · Go · 3,716 stars

OVERALL ASSESSMENT

High

Combined score 227 · 80 findings after filtering

Summary

The repository has a high vulnerability score of 220.0, which means there are many obvious coding mistakes that could be exploited by attackers if they find and use them. These issues are mostly about unsafe commands like running shell scripts or executing code with user input without proper checks, rather than signs of intentional malicious behavior. While some findings involve reading sensitive files or downloading remote resources, the scan also identified legitimate patterns such as bidirectional text support and startup scheduling changes that do not indicate a security threat. The developer should focus on fixing the command injection vulnerabilities first by ensuring all inputs are validated before running external commands, then address the missing mutex locks to prevent crashes in the code.

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 (220)

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	28	34	0	62

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
yara	Signatures of malicious behaviour in source files	10	0	2	12
guarddog	Malicious package and install-time behaviour	0	1	3	4
bandit	Insecure coding patterns in Python source	0	0	2	2

39 of 80 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	/internal/tmuxcompat/pane_unix.go:79
HIGH	semgrep	Detected non-static command inside exec.Cmd. Audit the input to 'exec.Cmd'. 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	/internal/tmuxcompat/pane_unix.go:133
HIGH	semgrep	Missing mutex unlock (`f.mu` variable) before returning from a function. This could result in panics resulting from double lock operations	1	/internal/session/host_fleet.go:223
HIGH	semgrep	Missing `RUnlock` on an `RWMutex` (`p.outputMu` variable) lock before returning from a function	3	/internal/session/session.go:2519
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.	20	/internal/sound/sound.go:177

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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	/ Dockerfile:22
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	6	/install-web.sh
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	4	/internal/vt/testdata/corpus/ls_color.bin
MEDIUM	semgrep	This code contains bidirectional (bidi) characters. While this is useful for support of right-to-left languages such as Arabic or Hebrew, it can also be used to trick language parsers into executing code in a manner that is different from how it is displayed in code editing and review tools. If this	2	/internal/fuzz/vtgen/vtgen.go:1022
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:	1	/.github/dependabot.yml:6
MEDIUM	guarddog	Detects access to sensitive files (credentials, configs, keys) [threat-filesystem-read]	1	/cmd/tuios/ssh_auth.go:53
MEDIUM	semgrep	Downcasting or changing sign of an integer with `uint32` method	1	/internal/session/daemon_windows.go:44

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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	24	/.github/workflows/release.yml:84
MEDIUM	semgrep	Detected the use of eval(). eval() can be dangerous if used to evaluate dynamic content. If this content can be input from outside the program, this may be a code injection vulnerability. Ensure evaluated content is not definable by external sources.	1	/cmd/tuios-wasm/smoke.mjs:43
MEDIUM	semgrep	Do not use `math/rand`. Use `crypto/rand` instead.	5	/internal/webshell/programs.go:6
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	3	/internal/release/github.go:59
LOW	bandit	Consider possible security implications associated with the subprocess module.	1	/internal/integration/assets/hermes/__init__.py:17
LOW	bandit	subprocess call - check for execution of untrusted input.	1	/internal/integration/assets/hermes/__init__.py:44
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	2	/install-web.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 24 September 2026.