

redis-ui

github.com/patrikx3/redis-ui

Scanned 24 September 2026 at 10:00 · Report generated 29 October 2026 · JavaScript · 808 stars

OVERALL ASSESSMENT

Medium

Combined score 57 · 50 findings after filtering

Summary

This repository has a medium vulnerability score of 53.0, which means there are several insecure coding patterns that an attacker could exploit to cause problems like denial-of-service or data theft. The most critical issues involve the use of regular expressions that might slow down your application and allowing the container to run as root, both of which require immediate attention before deployment. While some findings suggest potential malicious intent, many of these appear to be standard security checks rather than signs of a hidden attack, so they are not necessarily proof of bad faith behavior. However, because the code allows users to input paths that could lead to file traversal issues and uses loose version constraints for dependencies, it is safer to treat this as a risky project until those specific vulnerabilities are fixed.

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 (53)

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

MALICIOUS PATTERN

Low (4)

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	1	29	1	31

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
dep_checker	Vulnerable, unpinned or typosquatted dependencies	0	0	15	15
yara	Signatures of malicious behaviour in source files	1	0	1	2
guarddog	Malicious package and install-time behaviour	0	1	1	2

29 of 50 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	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: 17
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	1	/artifacts/readme/start-up-server.md
MEDIUM	semgrep	RegExp() called with a `year` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled	1	/scripts/fix-change-log.js:23
MEDIUM	semgrep	Service 'registry' is running with a writable root filesystem. This may allow malicious applications to download and run additional payloads, or modify container files. If an application inside a container has to save something temporarily consider using a tmpfs. Add 'read_only: true' to this servic	1	/docker-compose.yml :4
MEDIUM	semgrep	Service 'registry' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-privileges:true' in 'security_opt' to prevent this.	1	/docker-compose.yml :4

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	guarddog	Detects unicode homoglyphs and uncommon characters used for obfuscation [threat-runtime-obfuscation-unicode]	1	/src/strings/ky/index.mjs:116
MEDIUM	semgrep	No validation of origin is done by the addEventListener API. It may be possible to exploit this flaw to perform Cross Origin attacks such as Cross-Site Scripting(XSS).	2	/src/electron/window/main/onload.mjs:164
MEDIUM	semgrep	The target origin of the window.postMessage() API is set to "*". This could allow for information disclosure due to the possibility of any origin allowed to receive the message.	2	/src/electron/window/main/onload.mjs:93
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	20	/.github/workflows/build.yml:265
MEDIUM	semgrep	Detected possible user input going into a `path.join` or `path.resolve` 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.	1	/scripts/fix-change-log.js:22
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 `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	1	/k8s/manifests/deployment.yaml:18
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	1	/scripts/notarize.js:16

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	1	/k8s/manifests/deployment.yaml:15
LOW	dep_checker	Loose version constraint: 'p3x-redis-ui-server^2026.10.100' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'rimraf^6.1.3' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: '@fontsource/roboto^5.3.0' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'playwright^1.62.1' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'p3x-redis-ui-material^2026.10.100' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'node-fetch^3.3.2' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'electron^44.1.1' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'electron-updater^6.8.9' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'electron-store^11.0.2' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'electron-builder^26.15.3' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'corifeus-utils^2026.4.142' — attacker could push malicious update	1	package.json

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	dep_checker	Loose version constraint: 'corifeus-builder^2026.4.163' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'copyfiles^2.4.1' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'axios^1.20.0' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: '@playwright/test^1.62.1' — attacker could push malicious update	1	package.json
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	/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.