

cloudcmd

github.com/coderaiser/cloudcmd

Scanned 24 September 2026 at 09:59 · Report generated 29 October 2026 · JavaScript · 2,028 stars

OVERALL ASSESSMENT

Medium

Combined score 68 · 173 findings after filtering

Summary

The repository has a medium vulnerability score of 65, which means there are several insecure coding patterns that an attacker could exploit to cause problems like slow performance or denial of service. These issues stem from common mistakes in how the code handles regular expressions and file permissions rather than signs of intentional malicious behavior. However, the scan also found some concerning evidence of potential malware, including a YARA rule that detects dangerous downloads followed by execution and another that flags obfuscated string manipulation used to hide malicious code. While these specific patterns suggest the code might be trying to bypass security controls or execute harmful commands, they are not enough on their own to make the entire project unsafe for use. The developer should review the specific files mentioned in the report to understand exactly how these risks could affect their application and decide if they need to apply fixes before deploying it to production.

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 (65)

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

MALICIOUS PATTERN

Low (3)

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
dep_checker	Vulnerable, unpinned or typosquatted dependencies	0	0	130	130
semgrep	Insecure coding patterns across many languages	1	33	0	34
yara	Signatures of malicious behaviour in source files	8	0	1	9

33 of 173 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	/docker/ Dockerfile: 29
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	5	/ChangeLog
HIGH	yara	YARA rule matched: detect_obfuscated_string_execution — Reconstructs a string character-by-character or from hex escapes, then immediately executes it - fromCharCode/chr() alone are common in ordinary text-processing code, so this requires both the reconstruction and an execution call together	1	/client/ listeners/ index.js
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	2	/docker/ Dockerfile. io

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `prefix` 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-controll	1	/client/dom/current-file.js:73
MEDIUM	semgrep	RegExp() called with a `ext` 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	2	/client/modules/view/types.js:7
MEDIUM	semgrep	RegExp() called with a `wildcard` 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-contro	1	/common/util.js:24
MEDIUM	semgrep	RegExp() called with a `char` 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	/client/key/set-current-by-char.js:12
MEDIUM	semgrep	Service 'web' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-privileges:true' in 'security_opt' to prevent this.	1	/docker-compose.yml:3
MEDIUM	semgrep	Service 'web' 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 service to	1	/docker-compose.yml:3

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Checks for setting the environment variable <code>NODE_TLS_REJECT_UNAUTHORIZED</code> to 0, which disables TLS verification. This should only be used for debugging purposes. Setting the option <code>rejectUnauthorized</code> to false bypasses verification against the list of trusted CAs, which also leads to insecure transpor	1	/server/distribute/import.js:94
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>	25	/.github/workflows/nodejs.yml:24
LOW	dep_checker	Loose version constraint: '@cloudcmd/read-files-sync^2.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'http-auth^4.2.1' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'url-loader^4.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'package-json^10.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'cssnano-preset-default^9.0.1' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: '@cloudcmd/create-element^2.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: '@cloudcmd/move-files^9.0.1' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'just-snake-case^3.2.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'just-pascal-case^3.2.0' — attacker could push malicious update	1	package.js on

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	dep_checker	Loose version constraint: 'just-capitalize^3.2.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'html-webpack-plugin^5.6.3' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'request^2.76.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'files-io^4.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'for-each-key^2.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'mime-types^3.0.1' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'load.js^3.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'minor^1.2.2' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'montag^2.0.1' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'morgan^1.6.1' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'multi-rename^3.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'nano-memoize^3.0.16' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'nodemon^3.0.1' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'nomine^4.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'once^1.4.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'onezip^7.0.0' — attacker could push malicious update	1	package.js on

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	dep_checker	Loose version constraint: 'open^11.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'path-browserify^1.0.1' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'philip^3.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'pipe-io^4.0.1' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'place^1.1.4' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'ponse^8.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'postcss^8.5.3' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'process^0.11.10' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'pullout^5.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'putout^42.0.5' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'unionfs^4.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'style-loader^4.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'superc8^12.6.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'supermenu^5.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'supertape^13.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'table^6.0.1' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'tar-stream^3.0.0' — attacker could push malicious update	1	package.js on

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	dep_checker	Loose version constraint: 'tryrequire^3.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'win32^8.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'yargs-parser^22.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'writejson^3.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'wraptile^3.0.0' — attacker could push malicious update	1	package.js on
LOW	dep_checker	Loose version constraint: 'squad^3.0.0' — attacker could push malicious update	1	package.js on

Showing the 60 most significant of 143 distinct findings. The counts in the previous section cover all of them.

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.