

Rewrite

github.com/Yu9191/Rewrite

Scanned 29 July 2026 at 15:18 · Report generated 29 October 2026 · JavaScript · 259 stars

This repository is no longer available on GitHub. It was deleted or made private after this scan, so these findings cannot be re-verified. Read them as a record of what the repository contained on 29 July 2026.

OVERALL ASSESSMENT

High

Combined score 183 · 152 findings after filtering

Summary

This repository has been flagged as having a high vulnerability risk score of 180, which means that an attacker could easily exploit insecure coding patterns to break into your system or steal data. While the code does not appear to be intentionally malicious based on other checks, it contains several dangerous regular expressions that could crash your application if used incorrectly. Additionally, there are multiple instances where code attempts to execute base64-encoded payloads or obfuscated strings directly from the file system, which poses a significant threat of remote code execution. You should treat these findings as serious security concerns and avoid using this repository in any production environment until you can review and fix the dangerous code patterns.

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 (180)

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
semgrep	Insecure coding patterns across many languages	9	96	1	106
yara	Signatures of malicious behaviour in source files	46	0	0	46

23 of 152 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	Generic Secret detected	1	/ KuGougainian ban.js:18
HIGH	semgrep	JWT token detected	5	/insck.js:2
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	2	/ 123_extracted /123/static/ js/chunk-vendors.53e0e8c6.js
HIGH	semgrep	User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities	3	/RSA.js:4016
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	44	/ 123_extracted /123/static/ js/pagesContentDetails-contentDetails.07c2eec1.js

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `s` 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 in	1	/lsp/zhibojuhe.js:2
MEDIUM	semgrep	RegExp() called with a `F` 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 in	1	/kugouv5.js:1
MEDIUM	semgrep	RegExp() called with a `k` 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 in	1	/TgRedirect.js:26
MEDIUM	semgrep	RegExp() called with a `width` 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-controlle	1	/RSA.js:4468
MEDIUM	semgrep	RegExp() called with a `y` 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 in	1	/AppRavenTask.js:5

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Cannot determine what 'tabsHTML' is and it is used with a '<script>' tag. This could be susceptible to cross-site scripting (XSS). Ensure 'tabsHTML' is not externally controlled, or sanitize this data.	2	/KDF/video-parser.js:376
MEDIUM	semgrep	Cannot determine what 'curMedia' is and it is used with a '<script>' tag. This could be susceptible to cross-site scripting (XSS). Ensure 'curMedia' is not externally controlled, or sanitize this data.	2	/KDF/video-parser.js:416
MEDIUM	semgrep	Cannot determine what 'streamsHTML' is and it is used with a '<script>' tag. This could be susceptible to cross-site scripting (XSS). Ensure 'streamsHTML' is not externally controlled, or sanitize this data.	1	/KDF/video-parser.js:378
MEDIUM	semgrep	Cannot determine what 'qBtns' is and it is used with a '<script>' tag. This could be susceptible to cross-site scripting (XSS). Ensure 'qBtns' is not externally controlled, or sanitize this data.	1	/KDF/video-parser.js:417
MEDIUM	semgrep	Cannot determine what 'newData' is and it is used with a '<script>' tag. This could be susceptible to cross-site scripting (XSS). Ensure 'newData' is not externally controlled, or sanitize this data.	1	/kugou.js:203
MEDIUM	semgrep	Cannot determine what 'epsHTML' is and it is used with a '<script>' tag. This could be susceptible to cross-site scripting (XSS). Ensure 'epsHTML' is not externally controlled, or sanitize this data.	1	/KDF/video-parser.js:419
MEDIUM	semgrep	Depending on the context, user control data in `Object.assign` can cause web response to include data that it should not have or can lead to a mass assignment vulnerability.	10	/kugou.js:447

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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.	27	/RSA.js:4817
MEDIUM	semgrep	RegExp() called with a `t` 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 in	3	/insav.tv.js:58
MEDIUM	semgrep	RegExp() called with a `e` 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 in	43	/lsp/wm4k.js:29
LOW	semgrep	Detected string concatenation with a non-literal variable in a util.format / console.log function. If an attacker injects a format specifier in the string, it will forge the log message. Try to use constant values for the format string.	1	/LectMate.js:23

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 July 2026.