

cat-catch

github.com/xifangczy/cat-catch

Scanned 24 September 2026 at 08:13 · Report generated 29 October 2026 · JavaScript · 21,992 stars

OVERALL ASSESSMENT

Medium

Combined score 29 · 25 findings after filtering

Summary

This repository has a medium vulnerability score of 26.0, which indicates that there are several insecure coding patterns in the code that an attacker could potentially exploit to cause problems like denial-of-service or cross-site scripting attacks. While the overall risk is moderate, the scan also identified some signs of malicious intent with a low malicious-pattern score of 3.0, suggesting that certain parts of the code may have been intentionally crafted to be harmful rather than just accidentally buggy. The most critical issues involve dangerous regular expressions that could crash your application and unsafe file path handling that might let attackers access sensitive files on your server. Additionally, there are concerns about how messages are sent across different origins and how strings are being processed, which could lead to security breaches if not addressed carefully. Overall, while the codebase contains some serious security flaws, it does not appear to be entirely malicious, so you should review and fix the identified issues before using this repository in a production environment.

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 (26)

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	0	12	3	15
yara	Signatures of malicious behaviour in source files	9	0	0	9
guarddog	Malicious package and install-time behaviour	0	0	1	1

11 of 25 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	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	5	/lib/ mqtt.min. js
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	4	/lib/ mqtt.min. js
MEDIUM	semgrep	RegExp() called with a `arg` 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	/js/ templates. js:8
MEDIUM	semgrep	RegExp() called with a `urlPattern` 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-cont	1	/js/ init.js:45 2

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `item` 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	/js/ init.js:27 4
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).	1	/js/ content- script.js: 234
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.	5	/js/ m3u8.js:23 21
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	1	/js/ preview.js :127
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	/tools/ sync- locales.js :50
LOW	guarddog	Detects memory scraping and credential dumping from process memory [threat-process-memory]	1	/lib/ hls.min.js :1
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.	3	/js/ m3u8.js:89 0

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.