

ship-safe

github.com/asamassekou10/ship-safe

Scanned 24 September 2026 at 07:10 · Report generated 29 October 2026 · JavaScript · 848 stars

OVERALL ASSESSMENT

Critical

Combined score 721 · 443 findings after filtering

Summary

This repository has two major red flags that make it unsafe to use without significant changes. The code contains many instances of command injection vulnerabilities, specifically when functions like `child_process` are called with user-controlled arguments such as file paths or request data. These patterns could allow an attacker to run arbitrary commands on the server if they manage to trick the application into passing malicious input. Additionally, there are signs of potential malicious intent, including destructive operations that delete files and keylogging code that captures user input. While some issues like unencrypted HTTP requests and suspicious URLs are present, these appear to be common security practices rather than deliberate attempts to harm the system. The overall risk score is 647.0, which indicates a critical level of vulnerability that requires immediate attention before deployment.

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

Critical (647)

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

MALICIOUS PATTERN

High (73)

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	20	374	3	397
yara	Signatures of malicious behaviour in source files	16	0	3	19
guarddog	Malicious package and install-time behaviour	5	8	0	13
dep_checker	Vulnerable, unpinned or typosquatted dependencies	0	0	8	8
bandit	Insecure coding patterns in Python source	0	4	2	6

86 of 443 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 calls to <code>child_process</code> from a function argument <code>`req`</code> . This could lead to a command injection if the input is user controllable. Try to avoid calls to <code>child_process</code> , and if it is needed ensure user input is correctly sanitized or sandboxed.	2	<code>/ benchmarks/ corpus/ flow/cmd-hop.tainted.js:6</code>
HIGH	semgrep	Detected calls to <code>child_process</code> from a function argument <code>`targetPath`</code> . This could lead to a command injection if the input is user controllable. Try to avoid calls to <code>child_process</code> , and if it is needed ensure user input is correctly sanitized or sandboxed.	1	<code>/cli/ commands/ deps.js:118</code>
HIGH	semgrep	Detected calls to <code>child_process</code> from a function argument <code>`pm`</code> . This could lead to a command injection if the input is user controllable. Try to avoid calls to <code>child_process</code> , and if it is needed ensure user input is correctly sanitized or sandboxed.	1	<code>/cli/ commands/ deps.js:201</code>
HIGH	guarddog	Detects destructive operations (recursive deletion, wiping) [threat-filesystem-destruction]	1	<code>/cli/ commands/ guard.js:320</code>

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	semgrep	Unencrypted request over HTTP detected.	1	/ benchmarks/ corpus/ literal/ loopback.js :5
HIGH	semgrep	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input	2	/ benchmarks/ corpus/ flow/cmd-hop.guarded.js:7
HIGH	guarddog	Detects keylogging and input capture patterns [threat-runtime-keylogging]	3	/cli/ agents/ supply-chain-agent.js:34
HIGH	guarddog	Detects steganography decode followed by code execution [threat-runtime-obfuscation-steganography]	1	/cli/ utils/ patterns.js :1047
HIGH	semgrep	Avoiding SQL string concatenation: untrusted input concatenated with raw SQL query can result in SQL Injection. In order to execute raw query safely, prepared statement should be used. SQLAlchemy provides TextualSQL to easily use prepared statement with named parameters. For complex SQL composition	2	/ benchmarks/ corpus/ flow-py-guarded/ guarded_dao.py:6
HIGH	semgrep	Functions reliant on pickle can result in arbitrary code execution. Consider loading from `state_dict`, using fickle, or switching to a safer serialization method like ONNX	1	/ benchmarks/ corpus/ safe/model-load.py:2
HIGH	semgrep	Insecure WebSocket Detected. WebSocket Secure (wss) should be used for all WebSocket connections.	10	/cli/ commands/ openclaw.js :218
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	5	/cli/ utils/ patterns.js

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	9	/cli/ commands/ scan- skill.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	1	/cli/ agents/ roblox- security- agent.js
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	1	/vps/ setup.sh
MEDIUM	semgrep	RegExp() called with a `content` 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-control	2	/cli/ agents/ hermes- security- agent.js:17 76
MEDIUM	semgrep	RegExp() called with a `regex` 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	/cli/ agents/mcp- security- agent.js:94 3
MEDIUM	semgrep	RegExp() called with a `binding` 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-control	1	/cli/ agents/mcp- security- agent.js:86 6

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `value` 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	/cli/agents/hermes-security-agent.js:1686
MEDIUM	semgrep	RegExp() called with a `scope` 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	4	/cli/agents/mcp-security-agent.js:809
MEDIUM	semgrep	RegExp() called with a `code` 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	4	/cli/agents/mcp-security-agent.js:809
MEDIUM	semgrep	RegExp() called with a `credentialName` 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-	3	/cli/agents/hermes-security-agent.js:1776
MEDIUM	semgrep	RegExp() called with a `envVars` 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-control	1	/cli/commands/remediate.js:387

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `p` 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	/cli/ commands/ scan.js:74
MEDIUM	semgrep	RegExp() called with a `lines` 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	/cli/ agents/ dataflow- investigat or.js:641
MEDIUM	semgrep	RegExp() called with a `primary` 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-control	1	/cli/ agents/ install- guard- agent.js:26 6
MEDIUM	semgrep	RegExp() called with a `lang` 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	3	/cli/ agents/ dataflow- investigat or.js:811
MEDIUM	semgrep	RegExp() called with a `extra` 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	2	/cli/ agents/ dataflow- investigat or.js:698

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `call` 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	/cli/agents/model-scan-agent.js:242
MEDIUM	semgrep	RegExp() called with a `context` 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-control	1	/cli/agents/supabase-rls-agent.js:122
MEDIUM	semgrep	RegExp() called with a `propertyName` 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-co	1	/cli/agents/hermes-security-agent.js:1660
MEDIUM	semgrep	RegExp() called with a `functionName` 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-co	1	/cli/agents/hermes-security-agent.js:1811
MEDIUM	semgrep	RegExp() called with a `moduleName` 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	/cli/agents/base-agent.js:625

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `applicationName` 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	1	/cli/agents/hermes-security-agent.js:739
MEDIUM	semgrep	RegExp() called with a `providerName` 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-co	1	/cli/providers/llm-provider.js:745
MEDIUM	semgrep	RegExp() called with a `sectionName` 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-con	1	/cli/agents/hermes-security-agent.js:1653
MEDIUM	semgrep	`l.replace` method will only replace the first occurrence when used with a string argument (*). If this method is used for escaping of dangerous data then there is a possibility for a bypass. Try to use sanitization library instead or use a Regex with a global flag.	1	/cli/hooks/pre-tool-use.js:157
MEDIUM	semgrep	Running flask app with host 0.0.0.0 could expose the server publicly.	1	/vps/agent-wrapper.py:355
MEDIUM	guarddog	Detects advanced API call obfuscation using introspection and reflection techniques [threat-runtime-obfuscation-api]	1	/cli/commands/benchmark.js:267
MEDIUM	guarddog	Detects URLs to suspicious domains often used for exfiltration or C2 [threat-network-exfiltration]	3	/cli/commands/guard.js:374

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	guarddog	Detects URLs to URL shorteners, file sharing, and suspicious services [threat-network-outbound-shady-links]	3	/cli/commands/openclaw.js:349
MEDIUM	semgrep	Skill contains a directive to activate an "unrestricted", "debug", "developer", or "god" mode that disables the agent's built-in safety controls. These phrases are a hallmark of prompt injection and jailbreak attacks documented in the ClawHavoc malicious skill campaign. The instruction attempts to c	1	/claude-code-plugin/skills/ship-safe-fix/SKILL.md:27
MEDIUM	semgrep	This .npmrc does not set a minimum release age or sets it too low. Newly published packages can be malicious or unstable. Add `min-release-age = 7` to wait 7 days before resolving newly published package versions. Added in: v11.10 Reference: https://github.blog/changelog/2026-02-18-npm-bulk-trusted-	1	/.npmrc:1
MEDIUM	guarddog	Detects LOLBAS usage in install hooks (execution and network tools) [threat-process-hooks]	1	/benchmarks/corpus/safe/package.json:4
MEDIUM	semgrep	RegExp() called with a `name` 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	10	/cli/agents/dataflow-investigator.js:682
MEDIUM	semgrep	RegExp() called with a `pattern` 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-control	3	/scripts/hermes-baseline-drift.mjs:67

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Detected possible formatted SQL query. Use parameterized queries instead.	2	/ benchmarks/ corpus/ flow-py/ tainted_dao.py:6
MEDIUM	semgrep	Checks for any usage of http servers instead of https servers. Encourages the usage of https protocol instead of http, which does not have TLS and is therefore unencrypted. Using http can lead to man-in-the-middle attacks in which the attacker is able to read sensitive information.	1	/vps/ orchestrator/ index.js:233
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	/ benchmarks/ corpus/ vulnerable/ llm-output.js:1
MEDIUM	bandit	Use of unsafe PyTorch load	1	/ benchmarks/ corpus/ vulnerable/ model-load.py:2
MEDIUM	bandit	Possible SQL injection vector through string-based query construction.	2	/ benchmarks/ corpus/ flow-py-guarded/ guarded_dao.py:5
MEDIUM	bandit	Possible binding to all interfaces.	1	/vps/agent-wrapper.py:355
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.	319	/cli/ commands/ diff.js:95

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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	/vps/orchestrator/nginx.js:86
LOW	bandit	Consider possible security implications associated with the subprocess module.	1	/vps/agent-wrapper.py:18
LOW	dep_checker	Loose version constraint: 'write-file-atomic^7.0.0' — attacker could push malicious update	1	package.json
LOW	bandit	subprocess call - check for execution of untrusted input.	1	/vps/agent-wrapper.py:205
LOW	dep_checker	Loose version constraint: 'fast-glob^3.3.3' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'globals^17.6.0' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'eslint^10.3.0' — attacker could push malicious update	1	package.json

Showing the 60 most significant of 65 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.