

hugo

github.com/gohugoio/hugo

Scanned 24 September 2026 at 08:00 · Report generated 29 October 2026 · Go · 89,917 stars

OVERALL ASSESSMENT

High

Combined score 235 · 106 findings after filtering

Summary

This repository has a high vulnerability risk score of 231.0, which means there are many insecure coding patterns that an attacker could exploit to break your application. The most serious issues involve missing mutex locks that can cause the program to crash unexpectedly, using functions meant for finding paths instead of sanitizing them from user input, and running commands directly without verifying what is passed to them. While some findings suggest potential malicious intent, the majority appear to be standard insecure practices rather than signs of a hidden attack. The code also contains several other medium-risk issues related to regular expression usage and memory safety that could lead to crashes or denial-of-service attacks if not 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

High (231)

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	8	89	0	97

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
yara	Signatures of malicious behaviour in source files	6	0	1	7
guarddog	Malicious package and install-time behaviour	0	1	1	2

68 of 106 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	Missing mutex unlock (`f.mu` variable) before returning from a function. This could result in panics resulting from double lock operations	1	/hugolib/ filesystems/ basefs.go:10 0
HIGH	semgrep	`Clean` is not intended to sanitize against path traversal attacks. This function is for finding the shortest path name equivalent to the given input. Using `Clean` to sanitize file reads may expose this application to path traversal attacks, where an attacker could access arbitrary files on the ser	1	/commands/ server.go:32 1
HIGH	semgrep	Detected non-static command inside Command. Audit the input to 'exec.Command'. If unverified user data can reach this call site, this is a code injection vulnerability. A malicious actor can inject a malicious script to execute arbitrary code.	6	/common/ hexec/ exec.go:542
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	2	/.github/ workflows/ test.yml
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	/internal/ warpc/js/ renderkatex. bundle.js

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `regexp` 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	/livereload/ livereload. js:1587
MEDIUM	semgrep	RegExp() called with a `separator` 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-contr	1	/livereload/ livereload. js:1779
MEDIUM	semgrep	Variable `si` is likely modified and later used on error. In some cases this could result in panics due to a nil dereference	1	/common/ hugio/ copy.go:41
MEDIUM	semgrep	Variable `file` is likely modified and later used on error. In some cases this could result in panics due to a nil dereference	1	/commands/ convert.go:1 53
MEDIUM	semgrep	Found a formatted template string passed to 'template.HTML()'. 'template.HTML()' does not escape contents. Be absolutely sure there is no user-controlled data in this template. If user data can reach this template, you may have a XSS vulnerability.	2	/tpl/ strings/ truncate.go: 154
MEDIUM	semgrep	Externally controlled data influences the size of an allocation. This can usually lead to overflow or underflow and later trigger an out of bounds conditions.	2	/internal/ warpc/ genavif/ avif.c:324
MEDIUM	semgrep	Iteration over a possibly empty map `m`. This is likely a bug or redundant code	1	/htesting/ hqt/ checkers.go: 169

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	<code>`t.envName.replace`</code> method will only replace the first occurrence when used with a string argument (<code>"*"</code>). 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	/internal/warpc/js/renderkatex.bundle.js:253
MEDIUM	semgrep	This Dependabot configuration does not set a cooldown period. Newly published packages can be malicious or unstable. Add a <code>`cooldown`</code> block with <code>`default-days: 7`</code> to each <code>`package-ecosystem`</code> entry under <code>`updates`</code> to wait 7 days before proposing updates to newly published package versions. Reference:	1	/.github/dependabot.yml:4
MEDIUM	guarddog	Detects access to sensitive files (credentials, configs, keys) [threat-filesystem-read]	1	/releaser/releaser.go:119
MEDIUM	semgrep	Semgrep could not determine that the argument to <code>'template.HTML()'</code> is a constant. <code>'template.HTML()'</code> and similar does not escape contents. Be absolutely sure there is no user-controlled data in this template. If user data can reach this template, you may have a XSS vulnerability. Instead, do not use	39	/hugolib/page_content.go:581
MEDIUM	semgrep	The Origin header in the HTTP WebSocket handshake is used to guarantee that the connection accepted by the WebSocket is from a trusted origin domain. Failure to enforce can lead to Cross Site Request Forgery (CSRF). As per "gorilla/websocket" documentation: "A <code>CheckOrigin</code> function should carefully v	1	/livereload/livereload.go:96
MEDIUM	semgrep	Detected a network listener listening on <code>0.0.0.0</code> or an empty string. This could unexpectedly expose the server publicly as it binds to all available interfaces. Instead, specify another IP address that is not <code>0.0.0.0</code> nor the empty string.	1	/helpers/general.go:39

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	An HTTP redirect was found to be crafted from user-input `r`. This can lead to open redirect vulnerabilities, potentially allowing attackers to redirect users to malicious web sites. It is recommend where possible to not allow user-input to craft the redirect URL. When user-input is necessary to cra	2	/commands/ server.go:364
MEDIUM	semgrep	Detected 'Fprintf' or similar writing to 'http.ResponseWriter'. This bypasses HTML escaping that prevents cross-site scripting vulnerabilities. Instead, use the 'html/template' package to render data to users.	1	/commands/ server.go:292
MEDIUM	semgrep	Detected MD5 hash algorithm which is considered insecure. MD5 is not collision resistant and is therefore not suitable as a cryptographic signature. Use SHA256 or SHA3 instead.	7	/tpl/crypto/ crypto.go:45
MEDIUM	semgrep	Detected SHA1 hash algorithm which is considered insecure. SHA1 is not collision resistant and is therefore not suitable as a cryptographic signature. Use SHA256 or SHA3 instead.	2	/tpl/crypto/ crypto.go:115
MEDIUM	semgrep	Found an HTTP server without TLS. Use 'http.ListenAndServeTLS' instead. See https://golang.org/pkg/net/http/#ListenAndServeTLS for more information.	1	/commands/ server.go:489
MEDIUM	semgrep	Detected directly writing or similar in 'http.ResponseWriter.write()'. This bypasses HTML escaping that prevents cross-site scripting vulnerabilities. Instead, use the 'html/template' package and render data using 'template.Execute()'.	1	/livereload/ livereload. go:144
MEDIUM	semgrep	If an attacker can supply values that the application then uses to determine which method or field to invoke, the potential exists for the attacker to create control flow paths through the application that were not intended by the application developers. This attack vector may allow the attacker to	16	/common/ hreflect/ helpers.go:204

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Do not use <code>`math/rand`</code> . Use <code>`crypto/rand`</code> instead.	6	/tpl/math/math.go:21
MEDIUM	semgrep	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted	1	/tpl/tplimpl/embedded/templates/_shortcodes/instagram.html:243
LOW	guarddog	Detects heavy obfuscation techniques commonly used by malware [threat-runtime-obfuscation]	1	/tpl/internal/go_templates/htmltemplate/js.go:21
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	/hugolib/testdata/redis.cn.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.