

kots

github.com/replicatedhq/kots

Scanned 24 September 2026 at 06:51 · Report generated 29 October 2026 · Go · 950 stars

OVERALL ASSESSMENT

Critical

Combined score 1471 · 692 findings after filtering

Summary

This repository shows serious security issues that could be exploited by attackers, but most of the problems are caused by standard insecure coding practices rather than intentional malicious behavior. The code contains dangerous patterns like executing commands from user-controlled inputs and decoding base64 data to run it, which creates a high risk for command injection and remote code execution vulnerabilities. While some findings suggest the code might be trying to hide its true purpose or bypass security checks, the majority of alerts point to common mistakes developers make when handling file paths, secrets, and external commands. The scan also found several workflow files that leak sensitive information by passing all secrets directly to other automated jobs, which is a significant privacy risk for anyone using this code in production environments.

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 (1424)

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

MALICIOUS PATTERN

High (47)

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	41	598	21	660

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
yara	Signatures of malicious behaviour in source files	20	0	3	23
guarddog	Malicious package and install-time behaviour	4	1	4	9

583 of 692 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>`localPath`</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	/e2e/playwright/regression/shared/cli.ts:575
HIGH	semgrep	Detected calls to <code>child_process</code> from a function argument <code>`page`</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	/e2e/playwright/shared-core/support-bundle.ts:210
HIGH	semgrep	Detected calls to <code>child_process</code> from a function argument <code>`remoteUrl`</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	/e2e/playwright/regression/shared/cli.ts:575
HIGH	semgrep	Untrusted input could be used to tamper with a web page rendering, which can lead to a Cross-site scripting (XSS) vulnerability. XSS vulnerabilities occur when untrusted input executes malicious JavaScript code, leading to issues such as account compromise and sensitive information leakage. To preve	1	/pkg/handlers/update.go:91

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	semgrep	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the path which the application uses to access files. If the application does not validate user input and sanitize file paths, sensitive files such as confi	3	/pkg/handlers/airgap.go:205
HIGH	guarddog	Detects download-and-execute patterns: fetching a remote file then executing it [threat-process-download-exec]	3	/web/src/components/config_renderer/ConfigGroups.jsx:25
HIGH	guarddog	Detects base64 decoding followed by code execution [threat-runtime-obfuscation-base64exec]	1	/pkg/operator/client/deploy.go:184
HIGH	semgrep	This workflow uses `secrets: inherit` to pass all of the calling workflow's secrets to a reusable workflow. This violates the principle of least privilege because the called workflow receives access to every secret in the repository, not just the ones it needs. If the called workflow is compromised	1	/.github/workflows/release.yaml:70
HIGH	semgrep	Detected calls to child_process from a function argument `command`. This could lead to a command injection if the input is user controllable. Try to avoid calls to child_process, and if it is needed ensure user input is correctly sanitized or sandboxed.	3	/e2e/playwright/shared-core/support-bundle.ts:237
HIGH	semgrep	Using variable interpolation `\${{...}}` with `github` context data in a `run:` step could allow an attacker to inject their own code into the runner. This would allow them to steal secrets and code. `github` context data can have arbitrary user input and should be treated as untrusted. Instead, use	17	/.github/actions/build-dep-image-with-apko/action.yml:33

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	semgrep	bcrypt hash detected	1	/dev/ manifests/ kotsadm/ deployment. yaml:84
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.	10	/pkg/airgap/ update.go:46
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'.	2	/.github/ actions/ copy-assets/ Dockerfile:1 1
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	3	/.github/ actions/ version-tag/ dist/ index.js
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	12	/Makefile
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	3	/.github/ actions/ version-tag/ dist/ index.js
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	2	/cmd/ kotsadm/cli/ completion. go

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `sequence` 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	/e2e/playwright/regression/shared/version-history.ts:221
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	/.github/actions/cmx-versions/test.js:52
MEDIUM	semgrep	RegExp() called with a `expectedSequence` 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 use	1	/e2e/playwright/regression/shared/version-history.ts:180
MEDIUM	semgrep	Variable `rows` is likely modified and later used on error. In some cases this could result in panics due to a nil dereference	79	/pkg/store/kotsstore/downstream_store.go:568
MEDIUM	semgrep	Variable `wr` is likely modified and later used on error. In some cases this could result in panics due to a nil dereference	65	/pkg/store/kotsstore/supportbundle_store.go:187
MEDIUM	semgrep	Variable `tmpFile` is likely modified and later used on error. In some cases this could result in panics due to a nil dereference	1	/pkg/base/helm.go:462
MEDIUM	semgrep	Variable `license` is likely modified and later used on error. In some cases this could result in panics due to a nil dereference	1	/pkg/pull/pull.go:167

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	<code>`command.replace`</code> method will only replace the first occurrence when used with a string argument (<code>"\n"</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	/e2e/playwright/shared-core/support-bundle.ts:235
MEDIUM	semgrep	Variable <code>`kotsadmMetadata`</code> is likely modified and later used on error. In some cases this could result in panics due to a nil dereference	1	/pkg/handlers/metadata.go:73
MEDIUM	semgrep	Iteration over a possibly empty map <code>`forwardedAdditionalPorts`</code> . This is likely a bug or redundant code	1	/pkg/k8sutil/portforward.go:238
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:	2	/.github/dependabot.yml:18
MEDIUM	semgrep	Variable <code>`backup`</code> is likely modified and later used on error. In some cases this could result in panics due to a nil dereference	1	/pkg/snapshot/backup.go:147
MEDIUM	semgrep	Variable <code>`restore`</code> is likely modified and later used on error. In some cases this could result in panics due to a nil dereference	1	/pkg/snapshot/restore.go:161
MEDIUM	semgrep	A secret is exposed in the workflow-level <code>`env:`</code> block, making it available to every job and step in this workflow — including any untrusted code run in pull-request workflows. Scope secrets as narrowly as possible: prefer step-level <code>`env:`</code> so the secret is only available where it is actually needed	13	/.github/workflows/regression.yml:73

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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	/web/.npmrc:1
MEDIUM	guarddog	Detects access to sensitive files (credentials, configs, keys) [threat-filesystem-read]	1	/cmd/ imagedeps/ main.go:25
MEDIUM	semgrep	Semgrep could not determine that the argument to 'template.HTML()' is a constant. 'template.HTML()' 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	3	/kurl_proxy/ cmd/ main.go:277
MEDIUM	semgrep	A session cookie was detected without setting the 'HttpOnly' flag. The 'HttpOnly' flag for cookies instructs the browser to forbid client-side scripts from reading the cookie which mitigates XSS attacks. Set the 'HttpOnly' flag by setting 'HttpOnly' to 'true' in the Cookie.	2	/pkg/ handlers/ login.go:398
MEDIUM	semgrep	Disabled host key verification detected. This allows man-in-the-middle attacks. Use the 'golang.org/x/crypto/ssh/knownhosts' package to do host key verification. See https://skarlso.github.io/2019/02/17/go-ssh-with-host-key-verification/ to learn more about the problem and how to fix it.	1	/pkg/gitops/ gitops.go:71 2
MEDIUM	semgrep	Data is being piped into `bash` from a `curl` command. An attacker with control of the server in the `curl` command could inject malicious code into the pipe, resulting in a system compromise. Avoid piping untrusted data into `bash` or any other shell if you can. If you must do this, consider checki	2	/e2e/ scripts/ deps.sh:59

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	A session cookie was detected without setting the 'Secure' flag. The 'secure' flag for cookies prevents the client from transmitting the cookie over insecure channels such as HTTP. Set the 'Secure' flag by setting 'Secure' to 'true' in the Options struct.	3	/pkg/handlers/login.go:398
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.	6	/pkg/snapshot/store.go:1755
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.	1	/kurl_proxy/cmd/main.go:246
MEDIUM	semgrep	`MinVersion` is missing from this TLS configuration. By default, as of Go 1.22, TLS 1.2 is currently used as the minimum. General purpose web applications should default to TLS 1.3 with all other protocols disabled. Only where it is known that a web server must support legacy clients with unsupport	7	/pkg/kurl/kurl_nodes.go:153
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. `uses: ac	337	/.github/workflows/build-test.yaml:3942
MEDIUM	semgrep	When working with web applications that involve rendering user-generated content, it's important to properly escape any HTML content to prevent Cross-Site Scripting (XSS) attacks. In Go, the `text/template` package does not automatically escape HTML content, which can leave your application vuln	11	/pkg/template/builder.go:7

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Deserializing into <code>`interface{}`</code> allows arbitrary data structures and types, which can lead to security vulnerabilities (CWE-502). Use a concrete struct type instead.	3	/pkg/base/ base.go:169
MEDIUM	semgrep	Detected the use of <code>eval()</code> . <code>eval()</code> 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.	4	/.github/ actions/cmx- versions/ test.js:47
MEDIUM	semgrep	Checks for disabling of TLS/SSL certificate verification. This should only be used for debugging purposes because it leads to vulnerability to MTM attacks.	5	/pkg/kurl/ kurl_nodes. go:153
MEDIUM	semgrep	Detected a possible denial-of-service via a zip bomb attack. By limiting the max bytes read, you can mitigate this attack. <code>`io.CopyN()`</code> can specify a size.	13	/pkg/util/ tar.go:123
MEDIUM	semgrep	Detection of <code>dangerouslySetInnerHTML</code> from non-constant definition. This can inadvertently expose users to cross-site scripting (XSS) attacks if this comes from user-provided input. If you have to use <code>dangerouslySetInnerHTML</code> , consider using a sanitization library such as <code>DOMPurify</code> to sanitize your HT	1	/web/src/ components/ shared/ Markdown.tsx :42
MEDIUM	semgrep	Detected directly writing or similar in <code>'http.ResponseWriter.write()'</code> . This bypasses HTML escaping that prevents cross-site scripting vulnerabilities. Instead, use the <code>'html/template'</code> package and render data using <code>'template.Execute()'</code> .	12	/pkg/ upgradeserv ice/ handlers/ spa.go:64
MEDIUM	semgrep	Do not use <code>`math/rand`</code> . Use <code>`crypto/rand`</code> instead.	3	/cmd/ kotsadm/ main.go:4

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	14	/deploy/kurl/kotsadm/template/base/kurl-proxy/tmpl-deployment.yaml:21
LOW	guarddog	Detects heavy obfuscation techniques commonly used by malware [threat-runtime-obfuscation]	1	/pkg/identity/assets.go:3
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	3	/.github/actions/copy-assets/main.go:46
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.	4	/e2e/playwright/regression/shared/cli.ts:582
LOW	semgrep	When running containers in Kubernetes, it's important to ensure that they are properly secured to prevent privilege escalation attacks. One potential vulnerability is when a container is allowed to run applications as the root user, which could allow an attacker to gain access to sensitive resou	17	/deploy/kurl/kotsadm/template/base/deployment/kotsadm-airgap.yaml:7
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	3	/.github/workflows/build-test.yaml

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.