

avalanchego

github.com/ava-labs/avalanchego

Scanned 24 September 2026 at 08:50 · Report generated 29 October 2026 · Go · 2,359 stars

OVERALL ASSESSMENT

Critical

Combined score 383 · 146 findings after filtering

Summary

This repository has two distinct security issues that require immediate attention before it can be used safely. The most critical problem is an insecure gRPC connection found in the code, which means sensitive data could potentially be tampered with by attackers without encryption. Additionally, there are several high-risk patterns in the GitHub Actions workflows, including the use of `pull_request_target` to access external repositories and hardcoded private keys that should never be stored directly in source files. While some findings appear to be malicious intent or suspicious links, the majority represent standard insecure coding practices like using unencrypted web sockets and running commands without proper input validation. The scan also identified a Dockerfile configuration that allows containers to run as root users, which is a significant risk if an attacker controls the container process. Overall, the repository presents a critical vulnerability score of 380.0 due to these severe security flaws, so it should not be deployed until these issues are fixed by the development team.

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 (380)

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	33	107	0	140
yara	Signatures of malicious behaviour in source files	4	0	1	5
guarddog	Malicious package and install-time behaviour	0	1	0	1

116 of 146 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	Found an insecure gRPC connection using 'insecure.NewCredentials()'. This creates a connection without encryption to a gRPC server. A malicious attacker could tamper with the gRPC message, which could compromise the machine. Instead, establish a secure connection with an SSL certificate using the 'g	2	/vms/ rpcchainvm/ grpcutils/ client.go:54
HIGH	semgrep	This GitHub Actions workflow file uses `pull_request_target` and checks out code from the incoming pull request. When using `pull_request_target`, the Action runs in the context of the target repository, which includes access to all repository secrets. Normally, this is safe because the Action only	1	/.github/ workflows/ claude- review.yml:3 3
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	6	/.github/ workflows/ bazel-ci- scheduled.y ml:50

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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	15	/.github/workflows/self-hosted-load-tests.yaml:43
HIGH	semgrep	Insecure WebSocket Detected. WebSocket Secure (wss) should be used for all WebSocket connections.	1	/graft/coreth/plugin/evm/service.md:44
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.	2	/vms/rpcchainvm/runtime/subprocess/non_linux_topper.go:18
HIGH	semgrep	Private Key detected. This is a sensitive credential and should not be hardcoded here. Instead, store this in a separate, private file.	5	/staking/local/staker5.key:1
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'.	1	/Dockerfile:76
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	3	/.github/packaging/Dockerfile.rpm
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	1	/.envrc

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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:	5	<code>/.github/dependabot.yml:28</code>
MEDIUM	guarddog	Detects URLs to URL shorteners, file sharing, and suspicious services [threat-network-outbound-shady-links]	1	<code>/utils/dynamicip/resolver.go:16</code>
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	4	<code>/.github/workflows/bazel-ci.yml:49</code>
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. <code>`uses: ac</code>	75	<code>/.github/workflows/build-linux-binaries.yml:138</code>
MEDIUM	semgrep	The profiling 'pprof' endpoint is automatically exposed on <code>/debug/pprof</code> . This could leak information about the server. Instead, use <code>`import "net/http/pprof"`</code> . See https://www.farsightsecurity.com/blog/txt-record/go-remote-profiling-20161028/ for more information and mitigation.	2	<code>/graft/coreth/internal/debug/flags.go:317</code>
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 <code>`text/template`</code> package does not automatically escape HTML content, which can leave your application vuln	2	<code>/graft/coreth/accounts/abi/bind/bind.go:40</code>

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.	2	/graft/coreth/plugin/evm/config/config.go:218
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.	1	/network/peer/tls_config.go:31
MEDIUM	semgrep	Found an HTTP server without TLS. Use 'http.ListenAndServeTLS' instead. See https://golang.org/pkg/net/http/#ListenAndServeTLS for more information.	2	/graft/subnet-evm/internal/debug/flags.go:317
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()'.	2	/graft/evm/rpc/http.go:307
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	1	/vms/saevm/cmputils/cmputils.go:44
MEDIUM	semgrep	Do not use <code>`math/rand`</code> . Use <code>`crypto/rand`</code> instead.	11	/network/tracked_ip.go: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	1	/README.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.