

galgebra

github.com/pygae/galgebra

Scanned 24 September 2026 at 10:15 · Report generated 29 October 2026 · Python · 285 stars

OVERALL ASSESSMENT

Medium

Combined score 57 · 49 findings after filtering

Summary

This repository has a medium vulnerability risk score of 55.0, which means there are several insecure coding patterns that an attacker could potentially exploit if they gain access to the code. While the malicious-pattern risk is very low at 2.0, suggesting the code does not appear to be intentionally harmful, the most concerning issues involve unsafe subprocess calls and shell usage that could lead to command injection attacks. Additionally, there are supply chain risks because some dependencies like 'sympy' lack version locks, making them vulnerable to tampering by others who control their installation. The scan also flagged a specific YARA rule indicating that downloading files from the internet and executing them together is a dangerous combination, which requires careful handling. Overall, while the code may not be malicious, it contains significant security holes that should be addressed before using this library 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

Medium (55)

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

MALICIOUS PATTERN

Low (2)

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
bandit	Insecure coding patterns in Python source	7	5	20	32
semgrep	Insecure coding patterns across many languages	0	15	0	15
yara	Signatures of malicious behaviour in source files	1	0	0	1
dep_checker	Vulnerable, unpinned or typosquatted dependencies	0	1	0	1

11 of 49 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	bandit	Starting a process with a shell, possible injection detected, security issue.	7	/galgebra/ printer.py: 1034
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	1	/scripts/ validate_n b_refresh. py
MEDIUM	dep_checker	Unpinned dependency: 'sympy' has no version lock — supply chain risk	1	pyproject. toml
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	10	/.github/ workflows/ ci.yml:31
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.	5	/galgebra/ printer.py: 1157

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	bandit	Use of possibly insecure function - consider using safer <code>ast.literal_eval</code> .	5	/galgebra/ga.py:636
LOW	bandit	Consider possible security implications associated with the subprocess module.	2	/galgebra/gprinter.py:2
LOW	bandit	Starting a process with a partial executable path	5	/galgebra/gprinter.py:262
LOW	bandit	subprocess call - check for execution of untrusted input.	10	/galgebra/gprinter.py:264
LOW	bandit	Use of assert detected. The enclosed code will be removed when compiling to optimised byte code.	3	/galgebra/dop.py:358

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.