

artcraft

github.com/storytold/artcraft

Scanned 13 October 2026 at 06:15 · Report generated 29 October 2026 · Rust · 22,936 stars

OVERALL ASSESSMENT

Low

Combined score 9 · 20 findings after filtering

Summary

The repository has a low overall vulnerability score of 3.0, which means there are some insecure coding patterns that could be exploited if an attacker finds them, but these issues do not indicate malicious intent. The scan also identified several signs that the code might be intentionally harmful, such as downloading and executing remote files or attempting to harvest credentials from environment variables. While one specific finding suggests a process is being run silently without any output, this behavior alone does not confirm that the code is dangerous. Most of the flagged issues relate to standard security practices like handling timeouts for API calls or checking subprocess inputs, which are common mistakes rather than deliberate attacks. Therefore, while the code contains some risky patterns that should be addressed, it does not appear to be a malicious repository based on these specific indicators.

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

Low (3)

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

MALICIOUS PATTERN

Low (6)

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	0	1	11	12
yara	Signatures of malicious behaviour in source files	3	0	2	5
guarddog	Malicious package and install-time behaviour	0	0	3	3

3 of 20 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	yara	YARA rule matched: detect_credential_harvester — References credential/secret material by name AND a deliberate outbound-send API in the same file - narrower than 'reads any env var', which large legitimate codebases do constantly regardless of intent. Still not proof the two are connected (YARA c	1	/_docs/ artcraft_omni_ api.md
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	1	/crates/ api_clients/ grok_consumer_ client/src/ credentials/ _metadata/ crypto_signatu re/ xStatsigIDGene rator.py
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	/crates/ api_clients/ grok_consumer_ client/src/ credentials/ _metadata/ crypto_signatu re/ xStatsigIDGene rator.py

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	bandit	Call to requests without timeout	1	/crates/ api_clients/ openai_sora_client/src/ requests/ auth_sentinel_2/ metadata/ python_impl/ sentinel_token.py:20
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	1	/crates/ api_clients/ fal_client/src/ requests/ core_api/ fal_request.rs:52
LOW	guarddog	Detects fully silent process execution (suppressing all output channels) [threat-process-spawn-silent]	2	/frontend/ scripts/unix-dev.mjs:87
LOW	bandit	Try, Except, Continue detected.	5	/crates/ api_clients/ grok_consumer_client/src/ credentials/ _metadata/ crypto_signature/ xStatsigIDGenerator.py:208
LOW	bandit	Consider possible security implications associated with the subprocess module.	3	/crates/ api_clients/ grok_consumer_client/src/ credentials/ _metadata/ crypto_signature/ xStatsigIDGenerator.py:218

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	subprocess call - check for execution of untrusted input.	3	/crates/ api_clients/ grok_consumer_ client/src/ credentials/ _metadata/ crypto_signatu re/ xStatsigIDGene rator.py:122
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	2	/_docs/ artcraft_omni_ api.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 13 October 2026.