

public-pentesting-reports

github.com/juliocesarfort/public-pentesting-reports

Scanned 29 July 2026 at 15:18 · Report generated 29 October 2026 · HTML · 9,546 stars

OVERALL ASSESSMENT

Medium

Combined score 62 · 40 findings after filtering

Summary

This repository has a medium vulnerability score of 60, which means there are several insecure coding patterns that an attacker could exploit to run commands or steal data. However, the code does not appear to be intentionally malicious; instead, it contains common mistakes like missing security attributes and using unencrypted links, but these look like standard errors rather than deliberate attacks. The most serious issue is a command injection vulnerability found in one file, which could allow an attacker to execute arbitrary commands if they are not careful about what input the code receives. While there is also a low score indicating some signs of malicious intent, this is likely just a false positive because the specific patterns detected are very rare and do not match typical malware evasion techniques used by real attackers. Overall, while the code has some security gaps that need to be fixed to prevent exploitation, it does not look like a site designed to harm users directly.

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 (60)

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
semgrep	Insecure coding patterns across many languages	2	37	0	39
yara	Signatures of malicious behaviour in source files	1	0	0	1

2 of 40 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	Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing OS commands with user input	2	/Openwall/OpenVZ-audit/progs/sched/sched_fifo.c:20
HIGH	yara	YARA rule matched: detect_anti_analysis — Checks for a debugger, VM, or sandbox before continuing - a common malware evasion technique, essentially never present in ordinary application code	1	/iSEC/NCC_Group_-_Ricochet.pdf
MEDIUM	semgrep	This link points to a plaintext HTTP URL. Prefer an encrypted HTTPS URL if possible.	1	/RadicallyOpenSecurity/2018-CodeAudit+Pentest-F-Droid-v1.0.html:1504
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	36	/RadicallyOpenSecurity/2018-CodeAudit+Pentest-F-Droid-v1.0.html:51

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 29 July 2026.