

accelerated-intelligent-document-processing-on-aws

github.com/aws-solutions-library-samples/accelerated-intelligent-document-processing-on-aws

Scanned 24 September 2026 at 08:37 · Report generated 29 October 2026 · Python · 306 stars

OVERALL ASSESSMENT

High

Combined score 101 · 170 findings after filtering

Summary

This repository has been flagged as having high malicious-pattern risk, which means there are strong signs that the code was intentionally written to be dangerous rather than just containing ordinary insecure coding mistakes. While the vulnerability risk score is low and indicates only minor issues like unsafe YAML loading or XML parsing, the high malicious pattern score suggests the presence of specific threats such as credential harvesting, memory scraping, and deliberate attempts to download and execute remote files. You should treat this code with extreme caution because it contains multiple dangerous patterns that could be exploited by attackers, even though the overall vulnerability count is relatively small. The scan identified several critical issues including a YARA rule that detects credential harvesters, a pattern for downloading files and running them, and memory scraping techniques used to steal data from running processes. Given these specific high-risk indicators, it is not safe to use this code in production environments without thorough manual review and remediation of the identified threats.

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 (17)

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

MALICIOUS PATTERN

High (84)

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	6	95	101
yara	Signatures of malicious behaviour in source files	27	0	33	60
guarddog	Malicious package and install-time behaviour	4	1	3	8
dep_checker	Vulnerable, unpinned or typosquatted dependencies	0	0	1	1

10 of 170 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	guarddog	Detects HTTP/HTTPS URL dependencies in package.json (dependency confusion, untrusted sources) [threat-npm-http-dependency]	1	/src/ui/package.json:62
HIGH	guarddog	Detects download-and-execute patterns: fetching a remote file then executing it [threat-process-download-exec]	3	/scripts/srt/setup.py:19
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	12	/src/lambda/discovery_processor/index.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	12	/scripts/setup/dev_setup_ubuntu.sh
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	3	/scripts/setup/dev_setup_al2023.sh

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	bandit	Use of unsafe yaml load. Allows instantiation of arbitrary objects. Consider yaml.safe_load().	1	/scripts/ check_data_ plane_tags. py:195
MEDIUM	guarddog	Detects heavy code obfuscation techniques [threat-runtime-obfuscation-general]	1	/lib/ idp_common_ pkg/ idp_common/ ocr/ document_co nverter.py:1 823
MEDIUM	bandit	Audit url open for permitted schemes. Allowing use of file:/ or custom schemes is often unexpected.	2	/scripts/ ux_recorder. py:610
MEDIUM	bandit	Possible SQL injection vector through string-based query construction.	1	/scripts/ check_threa t_model_cur rency.py:218
MEDIUM	bandit	Probable insecure usage of temp file/directory.	2	/scripts/ ux_recorder. py:130
LOW	bandit	Using escape to parse untrusted XML data is known to be vulnerable to XML attacks. Replace escape with the equivalent defusedxml package, or make sure defusedxml.defuse_stdlib() is called.	1	/scripts/ ux_recorder_ render.py:80
LOW	guarddog	Detects memory scraping and credential dumping from process memory [threat-process-memory]	3	/lib/ idp_common_ pkg/ idp_common/ ocr/ service.py:1 409
LOW	bandit	Try, Except, Continue detected.	4	/scripts/ sdlc/ codebuild_d eployment.py :1534
LOW	bandit	Consider possible security implications associated with the subprocess module.	22	/scripts/ ux_recorder_ render.py:74

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	Starting a process with a partial executable path	20	/lib/ idp_sdk/ idp_sdk/ operations/ publish.py:52
LOW	bandit	subprocess call - check for execution of untrusted input.	36	/scripts/ srt/ setup.py:178
LOW	dep_checker	Loose version constraint: 'basedpyright^1.32.1' — attacker could push malicious update	1	package.json
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	33	/feature- platform/ sample- feature/ui- deployer/ handler.py
LOW	bandit	Use of assert detected. The enclosed code will be removed when compiling to optimised byte code.	12	/notebooks/ misc/ standalone_ sharded_ext raction_demo .py:148

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.