

VectorDB-Plugin

github.com/BBC-Esq/VectorDB-Plugin

Scanned 24 September 2026 at 10:02 · Report generated 29 October 2026 · Python · 370 stars

OVERALL ASSESSMENT

High

Combined score 158 · 141 findings after filtering

Summary

The repository shows two high-risk issues that look like serious security problems rather than normal mistakes. One issue involves using Python's subprocess command with the shell=True flag, which allows attackers to run arbitrary commands by injecting code into your shell environment. Another major concern is the use of pickle for data serialization, a method known to execute dangerous code when files are loaded incorrectly. These findings indicate that the code contains significant vulnerabilities that could be exploited if someone with access to this repository runs it on their systems. While some rules were triggered by unrelated files like system executables or environment variables, the core issues stem from how the application handles external commands and data storage. The developer should treat these as critical security flaws requiring immediate review and remediation before deploying the code anywhere.

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

High (116)

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

MALICIOUS PATTERN

High (42)

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	1	41	56	98
semgrep	Insecure coding patterns across many languages	4	29	0	33
yara	Signatures of malicious behaviour in source files	3	0	2	5
guarddog	Malicious package and install-time behaviour	4	0	1	5

14 of 141 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	the `encoding` argument to Popen is only available on Python 3.6+	1	/db/database_interactions.py:130
HIGH	semgrep	the `errors` argument to Popen is only available on Python 3.6+	1	/db/database_interactions.py:130
HIGH	guarddog	Detects download-and-execute patterns: fetching a remote file then executing it [threat-process-download-exec]	3	/setup_windows.py:201
HIGH	guarddog	Detects steganography decode followed by code execution [threat-runtime-obfuscation-steganography]	1	/modules/process_images.py:202
HIGH	semgrep	Found 'subprocess' function 'run' with 'shell=True'. This is dangerous because this call will spawn the command using a shell process. Doing so propagates current shell settings and variables, which makes it much easier for a malicious actor to execute commands. Use 'shell=False' instead.	1	/setup_windows.py:20
HIGH	semgrep	Functions reliant on pickle can result in arbitrary code execution. Consider loading from `state_dict`, using fickling, or switching to a safer serialization method like ONNX	1	/modules/kokoro.py:69

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	bandit	subprocess call with shell=True identified, security issue.	1	/setup_windows.py: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	2	/Assets/espeak_windows_files_extracted/libespeak-ng.dll
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	/tools/check_packages.py
MEDIUM	semgrep	Detected a dynamic value being used with urllib. urllib supports 'file:/' schemes, so a dynamic value controlled by a malicious actor may allow them to read arbitrary files. Audit uses of urllib calls to ensure user data cannot control the URLs, or consider using the 'requests' library instead.	4	/tools/check_packages.py:22
MEDIUM	semgrep	Avoid using `pickle`, which is known to lead to code execution vulnerabilities. When unpickling, the serialized data could be manipulated to run arbitrary code. Instead, consider serializing the relevant data as JSON or a similar text-based serialization format.	22	/db/database_interactions.py:669
MEDIUM	bandit	Use of weak MD5 hash for security. Consider usedforsecurity=False	2	/db/create_symlinks.py:33
MEDIUM	bandit	Audit url open for permitted schemes. Allowing use of file:/ or custom schemes is often unexpected.	4	/tools/check_packages.py:22
MEDIUM	semgrep	Untrusted user input in `importlib.import_module()` function allows an attacker to load arbitrary code. Avoid dynamic values in `importlib.import_module()` or use a whitelist to prevent running untrusted code.	1	/tools/check_packages.py:341

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Found non static data as an index to 'globals()'. This is extremely dangerous because it allows an attacker to execute arbitrary code on the system. Refactor your code not to use 'globals()'.	2	/chat/ base.py:470
MEDIUM	bandit	Pickle and modules that wrap it can be unsafe when used to deserialize untrusted data, possible security issue.	10	/db/ stage_split. py:258
MEDIUM	bandit	Unsafe Hugging Face Hub download without revision pinning in snapshot_download()	5	/core/ utilities.py :366
MEDIUM	bandit	Unsafe Hugging Face Hub download without revision pinning in hf_hub_download()	5	/modules/ ocr.py:278
MEDIUM	bandit	Unsafe Hugging Face Hub download without revision pinning in from_pretrained()	15	/tools/ build_user_ manual_db.py :82
LOW	bandit	Possible hardcoded password: '[PAD]'	1	/db/ embedding_m odels.py:401
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	1	/db/ database_in teractions. py:278
LOW	bandit	Starting a process without a shell.	4	/core/ utilities.py :752
LOW	bandit	Consider possible security implications associated with the subprocess module.	15	/core/ utilities.py :94
LOW	bandit	Starting a process with a partial executable path	9	/core/ utilities.py :162
LOW	bandit	subprocess call - check for execution of untrusted input.	20	/ setup_windo ws.py:88
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	/chat/ lm_studio.py

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	Use of assert detected. The enclosed code will be removed when compiling to optimised byte code.	7	/Assets/core.py:439

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.