

commix

github.com/commixproject/commix

Scanned 18 September 2026 at 19:41 · Report generated 29 October 2026 · Python · 5,848 stars

OVERALL ASSESSMENT

Critical

Combined score 349 · 136 findings after filtering

Summary

This repository has two distinct security issues that are clearly not caused by malicious intent but rather by poor coding practices. The most serious problem is an unverified SSL context, which means the code could allow connections without checking if the website's certificate is valid, potentially letting attackers hijack your network traffic. Additionally, there are multiple instances where developers use `shell=True` with the `subprocess` function, which runs commands using the current user's shell settings and variables instead of a safe environment, making it easy for attackers to execute arbitrary code. While some other findings like weak hash algorithms or suspicious GitHub Actions steps are present, they appear to be standard security weaknesses rather than signs that someone is trying to break into your system. The overall vulnerability risk score of 344.0 indicates significant exposure to exploitation if these issues are not fixed immediately.

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

Critical

 (344)

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

MALICIOUS PATTERN

Low

 (5)

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	10	36	16	62
semgrep	Insecure coding patterns across many languages	41	8	0	49
yara	Signatures of malicious behaviour in source files	8	0	17	25

47 of 136 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	Unverified SSL context detected. This will permit insecure connections without verifying SSL certificates. Use 'ssl.create_default_context' instead.	1	/src/utils/settings.py:1122
HIGH	semgrep	Found 'subprocess' function 'call' 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.	6	/src/utils/install.py:109
HIGH	semgrep	Found 'subprocess' function 'Popen' 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.	4	/src/utils/update.py:58
HIGH	semgrep	Avoiding SQL string concatenation: untrusted input concatenated with raw SQL query can result in SQL Injection. In order to execute raw query safely, prepared statement should be used. SQLAlchemy provides TextualSQL to easily used prepared statement with named parameters. For complex SQL composition	30	/src/utils/session_handler.py:221

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	yara	YARA rule matched: detect_reverse_shell — Reverse shell command patterns	1	/src/core/shells/reverse_tcp.py
HIGH	bandit	subprocess call with shell=True identified, security issue.	10	/src/utils/install.py:87
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	1	/src/core/controller/checks.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	6	/src/core/modules/shellshock/shellshock.py
MEDIUM	bandit	By default, Python will create a secure, verified ssl context for use in such classes as HTTPConnection. However, it still allows using an insecure context via the _create_unverified_context that reverts to the previous behavior that does not validate certificates or perform hostname checks.	1	/src/utils/settings.py:1122
MEDIUM	semgrep	Detected SHA1 hash algorithm which is considered insecure. SHA1 is not collision resistant and is therefore not suitable as a cryptographic signature. Use SHA256 or SHA3 instead.	1	/src/utils/session_handler.py:77
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	3	/.github/workflows/builds.yml:15
MEDIUM	bandit	Use of weak MD5 hash for security. Consider usedforsecurity=False	1	/src/utils/common.py:246

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Untrusted user input in <code>`importlib.import_module()`</code> function allows an attacker to load arbitrary code. Avoid dynamic values in <code>`importlib.import_module()`</code> or use a whitelist to prevent running untrusted code.	3	/src/core/controller/checks.py:2506
MEDIUM	bandit	Use of weak SHA1 hash for security. Consider <code>usedforsecurity=False</code>	1	/src/utils/session_handler.py:77
MEDIUM	bandit	Possible SQL injection vector through string-based query construction.	25	/src/core/controller/handler.py:149
MEDIUM	semgrep	Found non static data as an index to <code>'globals()'</code> . This is extremely dangerous because it allows an attacker to execute arbitrary code on the system. Refactor your code not to use <code>'globals()'</code> .	1	/src/utils/settings.py:990
MEDIUM	bandit	Use of <code>exec</code> detected.	1	/src/thirdparty/six/__init__.py:735
MEDIUM	bandit	Possible binding to all interfaces.	4	/src/thirdparty/socks/socks.py:804
MEDIUM	bandit	Probable insecure usage of temp file/directory.	3	/src/core/controller/checks.py:4434
LOW	bandit	Possible hardcoded password: <code>'/etc/passwd'</code>	1	/src/utils/settings.py:895
LOW	bandit	Possible hardcoded password: <code>"</code>	1	/src/utils/settings.py:1077
LOW	bandit	Try, Except, Continue detected.	5	/src/core/controller/handler.py:513
LOW	bandit	subprocess call with <code>shell=True</code> seems safe, but may be changed in the future, consider rewriting without shell	1	/src/utils/update.py:44

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	Consider possible security implications associated with the subprocess module.	4	/src/utils/update.py:18
LOW	bandit	Starting a process with a partial executable path	1	/src/utils/update.py:44
LOW	bandit	subprocess call - check for execution of untrusted input.	1	/src/utils/requirements.py:26
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	17	/src/core/requests/redirection.py
LOW	bandit	Use of assert detected. The enclosed code will be removed when compiling to optimised byte code.	2	/src/thirdparty/flatten_json/ flatten_json.py:49

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 18 September 2026.