

maltrail

github.com/stamparm/maltrail

Scanned 29 October 2026 at 09:30 · Report generated 29 October 2026 · Python · 8,646 stars

OVERALL ASSESSMENT

Critical

Combined score 313 · 197 findings after filtering

Summary

110 findings likely need attention, 75 look like noise. Most notable: The application builds a file path from potentially untrusted data, which can lead to a... (100% confidence).

Generated from finding counts. A plain-English summary was not available for this scan.

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 (202)

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

MALICIOUS PATTERN

Critical (111)

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	7	83	0	90
bandit	Insecure coding patterns in Python source	0	11	49	60
yara	Signatures of malicious behaviour in source files	13	0	13	26
guarddog	Malicious package and install-time behaviour	7	13	1	21

110 of 197 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
HIGH	semgrep	The application builds a file path from potentially untrusted data, which can lead to a path traversal vulnerability. An attacker can manipulate the path which the application uses to access files. If the application does not validate user input and sanitize file paths, sensitive files such as confi
HIGH	semgrep	Unverified SSL context detected. This will permit insecure connections without verifying SSL certificates. Use 'ssl.create_default_context' instead.
HIGH	guarddog	Detects suspicious autostart persistence mechanisms [threat-filesystem-autostart]
HIGH	guarddog	Detects cryptocurrency mining activity [threat-process-cryptomining]
HIGH	guarddog	Detects download-and-execute patterns: fetching a remote file then executing it [threat-process-download-exec]
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
HIGH	semgrep	By not specifying a USER, a program in the container may run as 'root'. This is a security hazard. If an attacker can control a process running as root, they may have control over the container. Ensure that the last USER in a Dockerfile is a USER other than 'root'.
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
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed

SEVERITY	DETECTOR	FINDING
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone
HIGH	yara	YARA rule matched: detect_obfuscated_string_execution — Reconstructs a string character-by-character or from hex escapes, then immediately executes it - fromCharCode/chr() alone are common in ordinary text-processing code, so this requires both the reconstruction and an execution call together
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this
MEDIUM	guarddog	Detects extensive system/network enumeration activities [threat-runtime-enumeration]
MEDIUM	guarddog	Detects URLs to URL shorteners, file sharing, and suspicious services [threat-network-outbound-shady-links]
MEDIUM	guarddog	Detects access to sensitive files (credentials, configs, keys) [threat-filesystem-read]
MEDIUM	bandit	Use of insecure cipher mode cryptography.hazmat.primitives.ciphers.modes.ECB.
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.
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.
MEDIUM	semgrep	Detected possible formatted SQL query. Use parameterized queries instead.
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

SEVERITY	DETECTOR	FINDING
MEDIUM	bandit	Use of weak MD5 hash for security. Consider usedforsecurity=False
MEDIUM	bandit	Use of weak SHA1 hash for security. Consider usedforsecurity=False
MEDIUM	bandit	Audit url open for permitted schemes. Allowing use of file:/ or custom schemes is often unexpected.
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()'.
MEDIUM	bandit	Possible binding to all interfaces.
LOW	bandit	Possible hardcoded password: '9ab3cd9d67bf49d01f6a2e33d0bd9bc804ddbe6ce1ff5d219c42624851db5dbc'
LOW	guarddog	Detects fully silent process execution (suppressing all output channels) [threat-process-spawn-silent]
LOW	bandit	Starting a process without a shell.
LOW	bandit	Try, Except, Continue detected.
LOW	bandit	subprocess call with shell=True seems safe, but may be changed in the future, consider rewriting without shell
LOW	bandit	Consider possible security implications associated with the subprocess module.
LOW	bandit	Starting a process with a partial executable path
LOW	bandit	subprocess call - check for execution of untrusted input.

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
LOW	bandit	Use of assert detected. The enclosed code will be removed when compiling to optimised byte code.

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 October 2026.