

h4cker

github.com/The-Art-of-Hacking/h4cker

Scanned 07 October 2026 at 04:13 · Report generated 29 October 2026 · Jupyter Notebook · 29,639 stars

OVERALL ASSESSMENT

High

Combined score 148 · 98 findings after filtering

Summary

This repository has a medium vulnerability score of 75 and a high malicious-pattern score of 73, which means the code contains several serious security flaws that could be exploited by attackers. The most critical issues involve using deprecated encryption libraries like pyCrypto and hardcoding sensitive credentials such as AWS keys directly in the source files, both of which significantly increase the risk of data theft or system compromise. While some findings appear to be ordinary mistakes like missing JWT verification or exposing a server on all network interfaces, others are highly suspicious because they match known patterns used by malware to execute reverse shells, inject code into running processes, or steal credentials from the environment. These malicious indicators suggest that the code may have been intentionally written to bypass security checks rather than just containing bugs, so you should treat it with extreme caution and avoid using it in any production environment without a thorough review.

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 (75)

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

MALICIOUS PATTERN

High (73)

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
yara	Signatures of malicious behaviour in source files	24	0	14	38
bandit	Insecure coding patterns in Python source	3	6	20	29
guarddog	Malicious package and install-time behaviour	0	15	5	20
semgrep	Insecure coding patterns across many languages	2	8	1	11

40 of 98 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	Detected JWT token decoded with 'verify=False'. This bypasses any integrity checks for the token which means the token could be tampered with by malicious actors. Ensure that the JWT token is verified.	1	/ cybersecurity-domains/application-security/web-application-testing/api_security_assessment.py:170
HIGH	bandit	The pyCrypto library and its module AES are no longer actively maintained and have been deprecated. Consider using pyca/cryptography library.	1	/ cybersecurity-domains/labs-practice/capture-the-flag/challenges/heavy_computation/encrypt.py:2

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	bandit	The pyCrypto library and its module pad are no longer actively maintained and have been deprecated. Consider using pyca/cryptography library.	1	/cybersecurity-domains/labs-practice/capture-the-flag/challenges/heavy_computation/encrypt.py:3
HIGH	bandit	The pyCrypto library and its module bytes_to_long are no longer actively maintained and have been deprecated. Consider using pyca/cryptography library.	1	/cybersecurity-domains/labs-practice/capture-the-flag/challenges/heavy_computation/encrypt.py:1
HIGH	semgrep	AWS Access Key ID Value detected. This is a sensitive credential and should not be hardcoded here. Instead, read this value from an environment variable or keep it in a separate, private file.	1	/cybersecurity-domains/defensive-security/dfir/snort/README.md:5
HIGH	yara	YARA rule matched: detect_reverse_shell — Reverse shell command patterns	4	/cybersecurity-domains/offensive-security/more-payloads/server-side-template-injection.txt

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	yara	YARA rule matched: detect_process_injection — Windows process-injection API sequence	1	/ cybersecurity-domains/application-security/programming-and-scripting-for-cybersecurity/exploitation/dll_injection_example.py
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	1	/ cybersecurity-domains/defensive-security/threat-intelligence/malware_bazaar_to_stix.py
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	2	/ cybersecurity-domains/offensive-security/more-payloads/more-xxs-payloads.txt
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	/ai/llm-engineering/ollama-labs/lab-01-installation-and-basics.md
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	3	/ cybersecurity-domains/offensive-security/more-payloads/xxs_obfuscation_vectors.txt

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	7	/ cybersecurity- domains/ defensive- security/ macos- hardening/ detecting- compromise.md
MEDIUM	semgrep	Running flask app with host 0.0.0.0 could expose the server publicly.	1	/ cybersecurity- domains/ application- security/ programming- and- scripting- for- cybersecurity / exploitation/ cookie_steal er.py:31
MEDIUM	guarddog	Detects extensive system/network enumeration activities [threat-runtime-enumeration]	3	/ cybersecurity- domains/ application- security/ programming- and- scripting- for- cybersecurity / recon_scripts /scanning/ python_nmap. py:13
MEDIUM	guarddog	Detects URLs to suspicious domains often used for exfiltration or C2 [threat-network-exfiltration]	3	/ cybersecurity- domains/ application- security/web- application- testing/ ssrf_ywing.py :28

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	guarddog	Detects URLs to URL shorteners, file sharing, and suspicious services [threat-network-outbound-shady-links]	5	/ cybersecurity- domains/ application- security/web- application- testing/ ssrf_ywing.py :28
MEDIUM	guarddog	Detects access to sensitive files (credentials, configs, keys) [threat-filesystem-read]	4	/ cybersecurity- domains/ application- security/web- application- testing/ api_security_ assessment.py :253
MEDIUM	semgrep	The 'ssl' module disables insecure cipher suites by default. Therefore, use of 'set_ciphers()' should only be used when you have very specialized requirements. Otherwise, you risk lowering the security of the SSL channel.	1	/ cybersecurity- domains/ application- security/ programming- and- scripting- for- cybersecurity / recon_scripts / certificate_ information/ check_weak_c rypto.py:26
MEDIUM	semgrep	The application exports an activity. Any application on the device can launch the exported activity which may compromise the integrity of your application or its data. Ensure that any exported activities do not have privileged access to your application's control plane.	1	/ cybersecurity- domains/ hardware- embedded- security/ mobile- security/ OmarsApplica tion/app/src/ main/ AndroidManif est.xml:12

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Data is being piped into `bash` from a `curl` command. An attacker with control of the server in the `curl` command could inject malicious code into the pipe, resulting in a system compromise. Avoid piping untrusted data into `bash` or any other shell if you can. If you must do this, consider checki	1	/cybersecurity-domains/cloud-container-security/docker-and-k8s-security/docker/run_cis_bench_locally.sh:28
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/greetings.yml:12
MEDIUM	semgrep	Checks for any usage of http servers instead of https servers. Encourages the usage of https protocol instead of http, which does not have TLS and is therefore unencrypted. Using http can lead to man-in-the-middle attacks in which the attacker is able to read sensitive information.	1	/cybersecurity-domains/cloud-container-security/docker-and-k8s-security/docker/server.js:6
MEDIUM	bandit	Possible binding to all interfaces.	1	/cybersecurity-domains/application-security/programming-and-scripting-for-cybersecurity/exploitation/cookie_stealer.py:31

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	bandit	Call to requests without timeout	5	/ cybersecurity- domains/ offensive- security/ osint/ quick_recon/ quick_recon. py:21
LOW	bandit	Possible hardcoded password: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...'	1	/ cybersecurity- domains/ application- security/web- application- testing/ api_security_ example.py:36
LOW	bandit	Possible hardcoded password: 'TR-token.txt'	1	/ certificatio ns/cisco/ scor-350-701/ threat- response-api- example.py:11
LOW	bandit	Possible hardcoded password: 'oauth2/token'	1	/ certificatio ns/cisco/ scor-350-701/ threat- response-api- example.py:24
LOW	bandit	Possible hardcoded password: 'Bearer xyz789'	1	/ cybersecurity- domains/ application- security/web- application- testing/ api_security_ example.py:10 8

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	Possible hardcoded password: '<INSERT YOUR API CLIENT PASSWORD HERE>'	1	/certifications/cisco/scor-350-701/threat-response-api-example.py:15
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	2	/cybersecurity-domains/application-security/web-application-testing/api_security_example.py:132
LOW	guarddog	Detects DLL injection and process injection techniques [threat-process-injection-dll]	3	/cybersecurity-domains/application-security/programming-and-scripting-for-cybersecurity/exploitation/dll_injection_example.py:61
LOW	bandit	Possible hardcoded password: 'Bearer your-token-here'	1	/cybersecurity-domains/application-security/web-application-testing/api_security_example.py:75

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	Possible hardcoded password: 'token123'	1	/cybersecurity-domains/application-security/web-application-testing/api_security_example.py:107
LOW	bandit	Possible hardcoded password: 'None'	1	/cybersecurity-domains/application-security/web-application-testing/api_security_example.py:106
LOW	bandit	Starting a process with a shell: Seems safe, but may be changed in the future, consider rewriting without shell	2	/cybersecurity-domains/application-security/programming-and-scripting-for-cybersecurity/exploitation/arp_cache_poisoner.py:75
LOW	semgrep	Detected a request using 'http://'. This request will be unencrypted, and attackers could listen into traffic on the network and be able to obtain sensitive information. Use 'https://' instead.	1	/cybersecurity-domains/application-security/web-application-testing/ssrf_galactic_archives.py:20

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	subprocess call with shell=True seems safe, but may be changed in the future, consider rewriting without shell	1	/cybersecurity-domains/application-security/programming-and-scripting-for-cybersecurity/recon_scripts/scanning/quick_scanner.py:17
LOW	bandit	Starting a process with a partial executable path	4	/cybersecurity-domains/application-security/programming-and-scripting-for-cybersecurity/recon_scripts/scanning/quick_scanner.py:17
LOW	bandit	Consider possible security implications associated with the subprocess module.	2	/cybersecurity-domains/application-security/programming-and-scripting-for-cybersecurity/recon_scripts/scanning/basic_ping_sweep.py:5

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	subprocess call - check for execution of untrusted input.	1	/ cybersecurity-domains/application-security/programming-and-scripting-for-cybersecurity/recon_scripts/scanning/basic_ping_sweep.py:13
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	14	/ cybersecurity-domains/offensive-security/osint/quick_recon/quick_recon.py
LOW	bandit	Use of assert detected. The enclosed code will be removed when compiling to optimised byte code.	2	/ cybersecurity-domains/labs-practice/capture-the-flag/challenges/heavy_computation/encrypt.py:36

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 07 October 2026.