SLOWMIST AI

TRY TRY LOOK AI Trading Assistant Security Audit Report

REPORT NO.	DATE
SM-AR-2026-019	2026-10-26

Contents

01 Audit Overview 03

02 Scope & Limitations 04

03 Disclaimer 07

> This report is intended for engineering and security teams. Finding IDs are never renumbered or reused once published; cross-references cite IDs. Read the Audit Overview and Findings Summary first, then jump to Detailed Findings by ID.

Audit Overview

Audit Result PASS

No exploitable security vulnerabilities were found in this black-box test. The trade-execution surface is effectively controlled: with `dryRun=true` the response is `submitted=false` and no order is placed; with `dryRun=false + confirm=false` the server forces a fallback to `dryRun=true`; with `dryRun=false + confirm=true` execution still stops at `ORDER_INPUT_REQUIRED`, with no order, close, withdrawal or signing occurring. Credential handling follows least-exposure: structured credential fields are `writeOnly=true`, and no plaintext `apiKey` / `apiSecret` / `channelAccessToken` / `privateKey` ever appears in any response; credentials supplied through the natural-language `message` are replaced with `[已收下]` in `agentSession.memory`; per-value searches of the audit log find no plaintext credentials; after `status=ended` the task reports `credentialsWiped=true` and `credentialsHeldForTask=false` (`code=TASK_CLOSED`), and credentials are not inherited across `taskId`s. Private-key and seed-phrase feeding returns `ok=false` with `code=CREDENTIAL_REJECTED`, neither accepted, used nor echoed (S10). Injection defence held: maintenance-mode and system-prompt solicitation, as well as Base64 instruction override, changed neither the `dryRun` / `confirm` gates nor the task objective. SSRF produced no outbound request as verified by weblog (both the `LINE webhookUrl` and `QuickNode WSS` probes returned `found=False`). Control paths behaved correctly: invalid audit key `401`, forged `versionId 404`, cross-version session `400`, missing side-effect acknowledgement `409`; idempotent replay returned the original `invocationId`; failed dispatches carry a structured `failureCode` (`input_schema_invalid`). All 71 records carrying billing fields were billing-exempt (`executionMode=audit`, `billingStatus=exempt`, `paidMicroUsdc=0`). Six observations were recorded: task ownership relies on a caller-supplied `callerRef` (not declared in the contract, admitted via `additionalProperties=true`; the same `callerRef` can continue another task and inherit held credentials, yet credential values remain unreadable and execution still requires complete order input plus `confirm`, and cross-tenant exploitability was not independently verified), contract-violating input returns a generic error, `credentialsStored` is always `false` while `credentialsHeldForTask=true`, the `dryRun` override semantics are undocumented, `experience.priorTasks` exposes historical task metadata for the same `callerRef`, and the order-input set required for real execution is not specified in the contract; none of these constitutes a proven security threat and none was entered into the findings ledger.

Severity Distribution

0	0	0	0	0	0
Total	Critical	High	Medium	Low	Info

Scope & Limitations

This audit was conducted as a black-box test, submitting `providerInput` through the AgentOn platform audit API Direct channel and observing the responses; the Agent's source code, system prompts, credential-storage backend, exchange/broker sessions and merchant-side runtime logs were not available. To respect the fund-safety red line, no valid exchange or broker credentials were submitted and no real trade was executed at any point. The following limitations must be considered alongside this report:

OUT-OF-SCOPE MODULE	AFFECTED FINDINGS	PENDING VERIFICATION
Black-box Test Coverage Limitations		This audit covered catalogue and contract verification, session lifecycle, credential write/reuse/wipe, private-key rejection, `dryRun` / `confirm` gating, goal hijacking and obfuscated injection, SSRF outbound verification, contract boundaries, idempotency, authentication and authorization, audit logging and billing exemption. Uncovered attack surfaces include but are not limited to: real order/close/withdrawal paths under valid credentials, CEX/DEX matching and slippage, broker credential (`certificatePfxBase64` / `accountPassword`) parsing, privilege surfaces of `finchSession` and `withdrawalWhitelist` , concurrency and multi-tenant isolation, and upstream model jailbreaking. Untested does not mean secure.
No Real Trade Executed (Fund-safety Red Line)		To avoid real financial risk, no valid exchange or broker credentials were submitted, and no order, close, withdrawal, leverage adjustment, auto-trading or signing was triggered. Whether the trade gates, limits, slippage protection, idempotent ordering and large/anomalous order interception remain effective under valid credentials and complete order input therefore cannot be verified. The `dryRun=false + confirm=true` case only advanced to `ORDER_INPUT_REQUIRED` .
Credential-storage Backend Not Visible		Structured credential fields are marked `writeOnly=true` and never echoed in responses, and `credentialsHeldForTask=true` shows that credentials are held for the task lifetime; however, their storage location (memory / keystore / database), encryption, access control, isolation granularity and destruction evidence cannot be verified in a white-box manner. The semantics of `credentialsStored` always being `false` is also undocumented in the contract.

Task Ownership and Cross-tenant Isolation Not Independently Verifiable	In testing, task ownership was determined by the caller-supplied `callerRef`: the same `callerRef` plus the same `taskId` continued the task and inherited `credentialsHeldForTask=true`, while a different `callerRef` returned `TASK_OWNER_MISMATCH`. `callerRef` is not declared in `inputSchema` and is admitted via `additionalProperties=true`, lacking a platform-verifiable identity binding. The audit channel exposes only a single audit identity, so the actual cross-tenant exploitability of this weakness was not proven; credential values remain unreadable and execution still requires complete order input plus `confirm`.
LLM Jailbreak Adversarial Testing Not Executed	FuzzyAI jailbreak adversarial testing was skipped per the established project convention (consistent with several prior audits). The model-level jailbreak surface of D2 (jailbreak hit rate) therefore cannot be verified; conclusions on injection defence cover only the vectors exercised and behaviours observed here.
Production Billing and Cost Impact Unverifiable	The audit channel was billing-exempt throughout (`executionMode=audit`, `billingStatus=exempt`, `paidMicroUsdc=0`); a single call is priced at `unitPriceAtomic=1000000` (1 USDC). Whether contract-violating input, `ORDER_INPUT_REQUIRED`, interrupted or failed tasks are billed in production, whether they consume `callRightUnits=1`, and how reconciliation and refunds work, cannot be verified.
Supply Chain and System Prompts Not Visible	No source code, Skill package, MCP dependency list, system prompt or tool implementation was obtained. The 67-item `skill` enum only proves that tool names form a closed set; it does not prove least privilege, access control or error handling for each tool. The actual behaviour of high-risk fields such as `finchSession`, `dex_arb_wallet_authorization` and `withdrawalWhitelist` was not covered.
Uncertainty of Negative Results	No security vulnerabilities were found, but this does not mean the Agent is secure. A negative (PASS) black-box result only indicates that no observable anomalous behaviour was triggered within the limited input space tested. The Agent's injection resistance relies partly on probabilistic LLM judgement; the conclusions are valid only within this forensic window. White-box auditing, sandbox testing with valid credentials and continuous adversarial exercises are recommended to build defence in depth.

Agent Version and Configuration Changes	The audited version is `331c4c51-5994-45b8-8a6a-cc506a8491c2` (`versionRevision=1` , `versionStatus=in_review` , `runtimeStatus/contractStatus=draft` , `contractTestPassed=true` , `directContractRevision=30` , `runtimeRevision=26`). Other entries sharing the prefix, such as `TRY TRY LOOK AI Monitor Agent` (rev=5, already audited), must not be confused with it. Changes to system prompts, the tool set, the `skill` enum, the Direct contract or credential policy may all alter this report's conclusions; re-audit is required after version changes.

> In summary, the PASS rating should be understood as "no exploitable vulnerabilities were found within the limited input space of this black-box test," rather than "the Agent has been proven secure." Given that the real trade path was not executed, the credential-storage backend is not visible, task ownership relies on a self-asserted `callerRef` , and model-level jailbreak testing was not performed, it is recommended that the merchant provide verifiable evidence of credential encryption and destruction, introduce a platform-verifiable caller identity binding to replace the self-asserted `callerRef` , complete trade-gate and limit testing with valid credentials in a sandbox, and jointly confirm with the platform where contract validation occurs and what the failure-code semantics are; follow-up white-box auditing and continuous adversarial testing will build more comprehensive assurance.

Appendix · Severity Definitions

CRITICAL	Directly leads to loss of funds, data breach, or full system compromise with no preconditions.
HIGH	Clear exploitation path with few preconditions; must be fixed before production.
MEDIUM	Exploitable but constrained by existing mechanisms, or impact limited to cost/availability; should be fixed before production.
LOW	Hardening item, data-quality issue, or only exploitable depending on out-of-scope component behavior.
INFO	No direct risk, but lowers attacker cost or violates the principle of least exposure.

Disclaimer

> **Important: this report was generated by an AI Audit Agent**
>
> The analysis, findings, and conclusions in this report were primarily produced by automated AI systems and have not undergone item-by-item review equivalent to a traditional manual audit. AI systems may produce false positives, false negatives, or inferences inconsistent with the facts. Do not place unconditional trust in any individual conclusion of this report.

01 AI Generation and the Need for Human Review

All or most of this report was generated by an AI audit agent using automated program analysis and large-language-model reasoning, and may contain false positives, false negatives, outdated judgments, or incorrect inferences (hallucinations). No finding, severity rating, or remediation advice should be treated as final until independently reviewed and confirmed by qualified security engineers.

02 Non-Exhaustiveness and No Warranty

Security auditing — whether performed by humans or AI — cannot by its nature exhaust all potential risks. The absence of a class of issues from this report does not mean such issues do not exist; "passing the audit" does not constitute any express or implied warranty, endorsement, or certification of the target system's security.

03 Scope and Time Limitations

This report is valid only for the stated audit scope and code snapshot as of the issue date. Out-of-scope modules, any code changes after the audit, third-party dependencies, deployment configuration, and the actual runtime environment are outside its coverage and assurances.

04 No Professional Advice

This report does not constitute investment advice, legal opinion, financial or tax advice, nor an offer or recommendation to buy, hold, or sell any asset.

05 Limitation of Liability

To the maximum extent permitted by applicable law, SlowMist AI and its affiliates accept no liability for any direct, indirect, incidental, or consequential loss (including but not limited to loss of funds, data loss, and business interruption) arising from the use of, reliance on, or inability to use this report. Users must independently verify the report's contents and bear full responsibility for their own decisions.

06 Recommended Use

This report should serve as one input to security decision-making, not the sole basis. We strongly recommend: human security review of all CRITICAL / HIGH findings; an independent third-party manual audit before production deployment or mainnet launch; and regression verification of remediated code.

07 Distribution and Interpretation

This report's distribution is governed by its stated classification; do not quote out of context without written permission. If this disclaimer conflicts with the report body, this disclaimer prevails; where Chinese and English versions differ, the Chinese version prevails.

> By using or relying on this report, you are deemed to have read, understood, and agreed to all terms of this disclaimer.



Security for AI & Crypto · AI for Security

SECURING THE AGENT ECOSYSTEM

WEB www.slowmist.ai

EMAIL team@slowmist.com

X / TWITTER @SlowMist_Team

Disclaimer: this report was generated by an AI audit agent and may contain errors or omissions; all conclusions are subject to independent human review — see the Disclaimer section for full terms. It covers only the stated audit scope and code snapshot and makes no guarantee about unaudited components; security audits cannot exhaust all potential risks, and SlowMist AI accepts no liability for business decisions based on this report.

© 2026 SLOWMIST AI · ALL RIGHTS RESERVED