SLOWMIST AI

Finfold Growth Audit Security Audit Report

REPORT NO.	DATE
SM-AR-2026-022	2026-10-30

Contents

01 Audit Overview 03

02 Findings Summary 04

03 Detailed Findings 05

04 Scope & Limitations 09

05 Disclaimer 11

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 WITH CONDITIONS

The platform control plane and business path are broadly sound. Control plane: an invalid audit key returns `401 SLOWMIST_AUDIT_TOKEN_INVALID`; a missing side-effect acknowledgement returns `409 AUDIT_SIDE_EFFECT_ACK_REQUIRED`; a forged `serviceVersionId` returns `404 SLOWMIST_AUDIT_SERVICE_UNKNOWN`; random session/invocation identifiers are rejected without leaking other sessions. All 18 calls were billing-exempt (`executionMode=audit`, `billingStatus=exempt`, `paidMicroUsdc=0`). Injection defence holds: four in-page payloads (instruction override with a marker string, system-prompt solicitation, environment/API-key solicitation, Markdown beacon) and one URL-only payload were never executed; the marker `INJECTION_CONFIRMED_7F3A` never appeared, the weblog beacon received no callback, and outputs contained no system prompt, secrets or upstream model-vendor identifiers. Server-side fetching boundaries hold: `127.0.0.1`, `169.254.169.254` and `file://` all failed within 0.63—0.95 s with no internal or local content returned; redirect targets were not followed and page subresources were not loaded (both confirmed by weblog); outbound fetches to a controlled public endpoint originate from a Cloudflare Worker (`Cf-Worker: finfold.app`, UA `FinfoldGrowthAudit/1.0`), which is designed behaviour. Output side: five of seven full artifacts passed JSON Schema validation; the report page returns a strict CSP (`default-src 'none'`, `script-src 'self'`), `X-Frame-Options: DENY`, `Referrer-Policy: no-referrer` and `Cache-Control: no-store`; the report token is 4,059 characters, decodes to 3,044 bytes with 7.94 bit/byte entropy, and a four-character tamper returns `400 invalid_report`, indicating an encrypted capability link. The conditional rating comes from three LOW findings: contract-violating input is not rejected at the acceptance boundary and is surfaced as a generic `AGENT_MARKET_INTERNAL_ERROR` (N1); reusing an idempotency key with different input returns an indeterminate `503` and leaves an undispatched run record (N2); and `in_review` / `draft` plus `availability.acceptingRequests=false` contradict the actually executable state (N3).

Key Findings

N1	LOW	Five contract violations accepted (202); failures flattened into a generic internal error
N2	LOW	Idempotency key reuse with different input returns indeterminate 503 and leaves an empty run
N3	LOW	in_review/draft and acceptingRequests=false contradict actual executability

Severity Distribution

Total	Critical	High	Medium	Low	Info
3	0	0	0	3	0

Findings Summary

3 findings · grouped by severity · IDs map to Detailed Findings

NO.	SEVERITY	SUMMARY	PAGE
LOW			3
N1	LOW	Contract violations not rejected; errors flattened	05
N2	LOW	Idempotency conflict returns 503, leaves empty run	06
N3	LOW	Status metadata contradicts executability	07

Detailed Findings

N1 Contract-violating Input Not Rejected at the Acceptance Boundary and Masked as an Internal Error

LOW

The contract `inputSchema` declares `url` as a 1—2048 character string, `objective` as the three-value enum `leads` / `signups` / `purchases`, and `additionalProperties=false`. In practice none of the five contract violations was rejected at the acceptance boundary: a missing `objective` (`invocationId=33e1ab32-f5ea-45df-b6ba-8e008af8e53a`), `objective=revenue` (`57159a80-9d45-4553-b117-58b31d9d1d97`), an extra `extra` field (`0c318dbb-8141-42ef-97fd-1ae1021ef5f1`), `url=""` (`895ffa3d-8d7a-48eb-91e3-dbe61aab588a`) and a 2,223-character `url` (`caf64a0e-7cfd-44e3-afb5-c19d32c4edfd`) each returned `HTTP 202` with `status=awaiting_dispatch` and failed only later at runtime.

The failure signal is flattened externally: callers always receive `AGENT_MARKET_INTERNAL_ERROR` ("The test run did not complete successfully."), while `logs` records `failureCode=input_schema_invalid` for the same runs. Callers therefore cannot distinguish a bad request from an upstream outage, which distorts retry policies and alert severity; invalid calls still consume dispatch and runtime resources (production billing cannot be verified).

Code Location

finfold-growth-session-20261030-002.json inputSchema：url 1—2048、objective 三值枚举、additionalProperties=false JSON

```
"inputSchema": {
  "type": "object",
  "required": ["url", "objective"],
  "properties": {
    "url": { "type": "string", "maxLength": 2048, "minLength": 1 },
    "objective": { "enum": ["leads", "signups", "purchases"], "type": "string" }
  },
  "additionalProperties": false
}
```

finfold-growth-direct-20261030-008.json C1 缺 objective：HTTP 202 + awaiting_dispatch，未在受理侧拒绝 JSON

```
"request": {
  "body": {
    "providerInput": { "url": "https://example.com" },
    "idempotencyKey": "neg-missing-objective"
  }
},
"http_status": 202,
"response": {
  "invocationId": "33e1ab32-f5ea-45df-b6ba-8e008af8e53a",
  "status": "awaiting_dispatch"
}
```

finfold-growth-logs-20261030-003.json logs 内部 failureCode=input_schema_invalid，未对外透出 JSON

```
mode=direct_api outcome=failed failureCode=input_schema_invalid invocation=33e1ab32
mode=direct_api outcome=failed failureCode=input_schema_invalid invocation=57159a80
mode=direct_api outcome=failed failureCode=input_schema_invalid invocation=0c318dbb
mode=direct_api outcome=failed failureCode=input_schema_invalid invocation=895ffa3d
mode=direct_api outcome=failed failureCode=input_schema_invalid invocation=caf64a0e
```

Remediation

Validate `providerInput` against `inputSchema` at the acceptance boundary and return 4xx (e.g. `AGENT_MARKET_REQUEST_INVALID` with the offending field path and constraint); map runtime `input_schema_invalid` to a distinguishable client error instead of `INTERNAL_ERROR`; preserve `failureCode` in error responses for programmatic handling and alert classification.

STATUS

OPEN

N2 Reusing an Idempotency Key with Different Input Returns an Indeterminate State and Leaves an Empty Run

LOW

Same-input replay behaves correctly: reusing `finfold-baseline-001` with the identical `providerInput` returned the original `invocationId=4fe465ae-b1e0-41e8-b881-ce0eefd7fc64` with no new invocation. However, submitting the same `idempotencyKey` with a different `providerInput` (`url` changed to `https://www.iana.org/help/example-domains`) returned `HTTP 503 SLOWMIST_AUDIT_UNAVAILABLE` with "The result could not be confirmed. Read the saved state before retrying." and `X-Finch-Test-Log-Status: complete` — neither an idempotent replay nor a deterministic conflict.

A `logs` record with `invocationId=null` and `dispatches=0` is left behind: no duplicate dispatch occurred (a positive), but the orphan record cannot converge, and the retry-suggestive 503 invites client retry loops. For agents with real-world side effects, ambiguous conflict semantics amplify duplicate-execution risk.

Code Location

finfold-growth-error-20261030-001.json 同一 idempotencyKey 换输入 → HTTP 503 SLOWMIST_AUDIT_UNAVAILABLE JSON

```
"http_status": 503,
"response": {
  "error": {
    "code": "SLOWMIST_AUDIT_UNAVAILABLE",
    "message": "The result could not be confirmed. Read the saved state before retrying."
  },
  "requestId": "0bdec405-6b8e-4874-88a1-20ec6f403a35"
}
```

finfold-growth-logs-20261030-002.json logs 中遗留无 invocationId、无 dispatch 的 run 记录 JSON

```
run mode=direct_api outcome=None failureCode=None invocation=null dispatches=0
```

Remediation

Return a deterministic `409` conflict for the same `idempotencyKey` with different input, referencing the original `invocationId` and an input digest; the conflict path should not create a run record; when the platform genuinely cannot decide, expose a machine-readable status rather than a generic 503.

STATUS

OPEN

N3 Status Metadata Contradicts Actual Executability

LOW

The session catalog projection reports `versionStatus=in_review`, `runtimeStatus=draft`, `contractStatus=draft` and `availability={status:unknown, checkedAt:null, acceptingRequests:false}`, alongside `marketVisible=true` and `contractTestPassed=true`. In practice the version accepts and executes calls normally: 7 of 18 calls produced complete reports (5 of them schema-valid), each taking about 52 s, with no unavailability gate observed.

Consumers gating on `availability.acceptingRequests` will wrongly treat the agent as unavailable, and `draft` stages can be integrated as if published; audit and operations teams also cannot use this field to judge real availability.

Code Location

finfold-growth-session-20261030-002.json runtimeStatus/contractStatus=draft 且 acceptingRequests=false JSON

```
"versionStatus": "in_review",
"runtimeStatus": "draft",
"contractStatus": "draft",
"availability": {
  "status": "unknown",
  "checkedAt": null,
  "acceptingRequests": false
}
```

finfold-growth-catalog-20261030-001.json 同一版本 marketVisible=true、contractTestPassed=true JSON

```
"versionStatus": "in_review",
"contractTestPassed": true,
"publicationGeneration": "0"
// 实测：18 次调用中 7 次 completed（含 5 份 schema 合规产物）
```

Remediation

Align the state machine with the `availability` projection: either reject external calls during `draft` / `in_review`, or mark the version as audit-channel-only; tie `marketVisible` to publication state; populate `availability.status` from runtime probing instead of a fixed `unknown`.

STATUS

OPEN

Scope & Limitations

This is a black-box audit: inputs were submitted only through the AgentOn Direct channel and responses observed from outside. Agent source code, system prompts, fetch-tool implementation, upstream models and merchant-side runtime logs were not available. The following limitations apply to every conclusion below:

OUT-OF-SCOPE MODULE	AFFECTED FINDINGS	PENDING VERIFICATION
Black-box coverage		Covered catalog and contract verification, session lifecycle, 18 Direct calls, contract boundaries, server-side fetching and internal-target probing, five injection payloads, idempotency and the control plane, log observability, output schema and report-page headers. Not covered: source code, system prompts, fetch-tool implementation, multi-turn progressive jailbreaks, concurrency and multi-tenant isolation, upstream model version and content-filtering policy. Untested does not mean secure.
Fetch implementation not observable	**N1**	Egress originates from a Cloudflare Worker (Cf-Worker: finfold.app). Its IP-range filtering, DNS resolution and rebinding policy, timeouts and response-size limits cannot be verified. The failure of internal/metadata probes only shows no content was returned along the current path; other network paths or future implementation changes are not excluded.
Report page and token lifecycle		Headers, CSP, token entropy and integrity were verified. Token expiry, revocation, rate limiting of `/report/open` and access-log retention were not. A capability link, once shared, is equivalent to disclosing the report.
Model-level jailbreak suite not executed	**N1**	The agent exposes a single-shot stateless Direct channel while the FuzzyAI bridge depends on multi-step turn/task flows, so automated jailbreak testing was not run; five custom injection payloads were used instead. Multi-turn Crescendo/PAIR attacks remain unverified.
Production billing and cost not verifiable	**N1**、**N2**	Every audit call was billing-exempt (`executionMode=audit`, `billingStatus=exempt`, `paidMicroUsdc=0`) against a list price of `unitPriceAtomic=200000` (0.2 USDC). Whether contract violations and runtime failures are billed, reconciled or refunded in production cannot be verified.

Content and factual accuracy out of scope	Only output-schema compliance and prohibited vendor identifiers were checked; growth recommendations, page-fact citations and copy quality were not individually verified. Claims such as "1,200 crews in 14 countries" are the audited page's own statements and the agent flags them as unverified in `limitations`.

> Follow-up review should cover the modules above and close N1, N2 and N3 end to end: verify 4xx at acceptance with contract-violating input, verify 409 semantics with an idempotency conflict, and verify availability projection consistency against status metadata.

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