DeagentAI

Audit Report

MOVEBIT

contact@bitslab.xyz

https://twitter.com/movebit_

Fri Nov 28 2025

DeagentAI Audit Report

1 Executive Summary

1.1 Project Information

Description	DeAgentAI is the AI Agent infrastructure across Sui, BSC, and BTC ecosystems, empowering AI Agents with trustless autonomous decision-making capabilities on-chain.
Type	Infrastructure, SocialFi
Auditors	Sprig,Leon@BitsLab
Timeline	Mon Nov 24 2025 - Fri Nov 28 2025
Languages	Move, Solidity
Platform	Sui,BSC
Methods	Architecture Review, Unit Testing, Manual Review

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
AIA	bsc/contracts/AIA.sol	775203705804a6c0aad96da27528fb9a9a790a39
CMI	sui_coin_migrate/coin_migrate/sources/coin_migrate.move	fb56f2cc23e11aaf3180ee788579206b4f4f2943
DTO	sui_coin_migrate/deagent_token/sources/deagent_token.move	7b2db8cc7aa784fdce7dca0850f8bcbd4390c9a4
DTO1	sui_coin_migrate/AIA/sources/deagent_token.move	741d24da220010ffc7c3bd23c9623a1c6a296995

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	2	0	2
Centralization	1	0	1
Critical	0	0	0
Major	0	0	0
Medium	0	0	0
Minor	1	0	1
Informational	0	0	0

1.4 MoveBit Audit Breakdown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Formal Verification(Optional)

Perform formal verification for key functions with the Move Prover.

(4) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by DeagentAI to identify any potential issues and vulnerabilities in the source code of the DeagentAI smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 2 issues of varying severity, listed below.

ID	Title	Severity	Status
AIA-1	Centralization Risk	Centralization	Acknowledged
CMI-2	Missing Events	Minor	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the DeagentAI Smart Contract :

Admin

- `Admin` can set users to migrate coin through `admin_set_users()` .
- `Admin` can delete users from migration list through `admin_del_users()` .
- `Admin` can toggle migration status (pause/resume) through `admin_toggle_migrate()` .
- `Admin` can update migration program version through `migrate()` .
- `Admin` can set TokenA type through `set_token_a_type()` .

User

- `User` can migrate their TokenA to TokenB through `user_migrate()` .
- `User` can check their remaining migration amount through `get_user_amount()` .
- `User` can check their total migrated amount through `get_user_migrated()` .
- `User` can batch query migration status through `get_users_migrated()` .

4 Findings

AIA-1 Centralization Risk

Severity: Centralization

Status: Acknowledged

Code Location:

bsc/contracts/AIA.sol#6

Descriptions:

Centralization risk was identified in the smart contract:

- Admin minted all the tokens at once and holds all the undistributed tokens.

Suggestion:

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution plan should be published in a public location that the community can access. The team should make efforts to restrict access to the private keys of the deployer account or EOAs. A multi-signature wallet can be used to prevent a single point of failure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vesting schedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greater accountability.

CMI-2 Missing Events

Severity: Minor

Status: Acknowledged

Code Location:

sui_coin_migrate/coin_migrate/sources/coin_migrate.move#69-107,109-119

Descriptions:

The functions `admin_set_users` , `admin_del_users` , `admin_toggle_migrate` , and `set_token_a_type` are used by administrators to fill in user redemption information. However, these functions do not emit events. The lack of events increases the difficulty of off-chain monitoring and.

Suggestion:

It is recommended to add the missing events.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.