DeagentAI Token

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Mon Sep 01 2025

DeagentAI Token Audit Report

1 Executive Summary

1.1 Project Information

Description	DeAgentAI is the largest AI Agent infrastructure across Sui, BSC, and BTC ecosystems, empowering AI Agents with trustless autonomous decision-making capabilities on-chain.
Type	Infrastructure, SocialFi
Auditors	MoveBit
Timeline	Wed Aug 27 2025 - Mon Sep 01 2025
Languages	Solidity, Move
Platform	BSC,Sui
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
DTO	deagentai/token/sui/deagent_token.move	7ca5ef1b19aff8282dd567b096a238875aa2a29a
AIA	deagentai/token/bsc/AIA.sol	775203705804a6c0aad96da27528fb9a9a790a39

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	2	2	0
Informational	2	2	0
Minor	0	0	0
Medium	0	0	0
Major	0	0	0
Critical	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 Token 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	Unnecessary `Ownable()` Constructor Call in AIA Contract	Informational	Fixed
DTO-1	`batch_transfer` Becomes Inoperable If `TreasuryCap` is Frozen	Informational	Fixed

3 Participant Process

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

User

- `User` can transfer tokens to another address via `transfer()` / `transferFrom()` .
- `User` can approve another address to spend tokens on their behalf via `approve()` .

Owner

- `Owner` can transfer ownership to a new address via `transferOwnership()` .

4 Findings

AIA-1 Unnecessary `Ownable()` Constructor Call in AIA Contract

Severity: Informational

Status: Fixed

Code Location:

deagentai/token/bsc/AIA.sol#1

Descriptions:

The `AIA` contract inherits from OpenZeppelin's Ownable contract, but the ownership functionality is not used anywhere in the contract. The explicit call to `Ownable()` in the constructor is redundant, as the parent constructor is automatically invoked by Solidity when not explicitly called. This adds unnecessary boilerplate and can slightly increase contract bytecode size.

Suggestion:

Remove the explicit `Ownable()` call from the constructor:

```
constructor(string memory _name, string memory _symbol) ERC20(_name, _symbol) {
    _mint(msg.sender, 200_000_000 * 10 ** decimals());
}
```

Optionally, if no owner-controlled functionality is needed, consider removing Ownable inheritance entirely to simplify the contract.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

DTO-1 `batch_transfer` Becomes Inoperable If `TreasuryCap` is Frozen

Severity: Informational

Status: Fixed

Code Location:

deagentai/token/sui/deagent_token.move#1

Descriptions:

The `TreasuryCap` in Sui is a capability that can be frozen. If the `TreasuryCap` for `DEAGENT_TOKEN` is frozen, any subsequent calls to `coin::mint_and_transfer` will fail. This means that if the `TreasuryCap` is frozen, the `batch_transfer` function will always revert and be impossible to execute. This function, which is likely intended for important operations like airdrops or reward distributions, becomes permanently disabled.

Suggestion:

The logic for managing the `TreasuryCap` should be carefully designed to prevent unauthorized or accidental freezing. No direct code change in `batch_transfer` can prevent this vulnerability; the solution lies in the operational security and access control surrounding the `TreasuryCap` object.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

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.