DeagentAI Bridge

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Fri Sep 12 2025

DeagentAI Bridge Audit Report

1 Executive Summary

1.1 Project Information

Description	DeagentAI Bridge is a bridge from BSC to Sui.
Type	Infrastructure, SocialFi
Auditors	MoveBit
Timeline	Tue Sep 09 2025 - Fri Sep 12 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
BBS	bridge-bsc.sol	ae4b181338983f510fb3ff2090e378f3e2b0f760
BSU	bridge-sui.move	89a46590bb353951f36f6da282cd72c847c5ea4e

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	7	4	3
Informational	2	0	2
Minor	3	2	1
Medium	2	2	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 Bridge 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 7 issues of varying severity, listed below.

ID	Title	Severity	Status
BBS-1	Cross-chain Bridge `amount` Numeric Type Mismatch	Medium	Fixed
BBS-2	Inconsistency Between The Comment And The Actual Implementation	Minor	Fixed
BBS-3	BSC Bridge Contract Sui Address Format Validation Missing	Minor	Fixed
BBS-4	Bridge Contract Lacks Emergency Pause Mechanism	Minor	Acknowledged
BBS-5	Lack of Fund Refund Processing	Informational	Acknowledged
BBS-6	Useless `signature` Parameter	Informational	Acknowledged
BSU-1	Third Parties Can Forge Sui `TransferEvent`	Medium	Fixed

3 Participant Process

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

user

- call "bridge" in "bridge-bsc.sol" to start transfer token.

bridge-offchain-server

- call "transfer_token" in "bridge-sui.move" to finish the cross-chain transfer.

4 Findings

BBS-1 Cross-chain Bridge `amount` Numeric Type Mismatch

Severity: Medium

Status: Fixed

Code Location:

bridge-bsc.sol#61;

bridge-sui.move#79

Descriptions:

There exists a numeric type mismatch issue between the BSC bridge contract and Sui bridge contract. The BSC side `bridge` function uses `uint256` type to handle token amounts, while the Sui side `transfer_token` function uses `u64` type. This type mismatch may cause large cross-chain transfers to fail or produce unexpected behavior.
For most ERC20 tokens, u64::MAX is already a very large number (approximately 18.44 quadrillion), but certain tokens with extremely high precision or special designs might exceed this limit.

Suggestion:

Add Numeric Range Check on BSC Side

- Add `amount <= type(uint64).max` check in `bridge` function

Resolution:

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

BBS-2 Inconsistency Between The Comment And The Actual Implementation

Severity: Minor

Status: Fixed

Code Location:

bridge-bsc.sol#103

Descriptions:

In the `bridge-bsc.sol` contract, there is an inconsistency between the comment and the actual implementation, which could lead to a critical security issue:

```
// ... existing code ...
// Transfer tokens from user to zero address for burning
token.safeTransferFrom(msg.sender, address(this), amount);
// ... existing code ...
```

Comment indicates tokens should be transferred to the zero address (address(0)) for burning.

Actual implementation transfers tokens to the contract itself (address(this)).

Suggestion:

`ERC20.safeTransferFrom` cannot transfer tokens to `address(0)`, otherwise it will directly revert. In real-world projects, if you want to simulate burning, it is common to transfer tokens to an uncontrollable “dead address”. Tokens sent there can never be recovered, which effectively achieves burning.

```
// Transfer tokens from user to a burn address (irrecoverable)
token.safeTransferFrom(msg.sender, 0xdead, amount);
```

Or, if the design intention is indeed to retain tokens in the contract instead of burning, then update the comment to reflect the actual implementation:

```
// Transfer tokens from user to this contract
token.safeTransferFrom(msg.sender, address(this), amount);
```

Resolution:

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

BBS-3 BSC Bridge Contract Sui Address Format Validation Missing

Severity: Minor

Status: Fixed

Code Location:

bridge-bsc.sol#60

Descriptions:

In the `bridge` function of the `bridge-bsc.sol` contract, the `to_addr` parameter lacks Sui address format validation. Currently, it only checks if the address is empty, but does not validate whether the address conforms to Sui blockchain address format requirements, which may cause tokens to be sent to invalid addresses. **Sui Address Format Requirements**:

- Sui addresses are typically 32-byte hexadecimal strings
- Start with "0x"
- Should be 66 characters long (0x + 64 hexadecimal characters)

Potential Risks:

- Users may input incorrectly formatted addresses
- Tokens may be sent to invalid addresses
- Funds may be permanently lost

Suggestion:

Add Sui address format validation function

Resolution:

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

BBS-4 Bridge Contract Lacks Emergency Pause Mechanism

Severity: Minor

Status: Acknowledged

Code Location:

bridge-bsc.sol#1;

bridge-sui.move#1

Descriptions:

The `bridge-bsc.sol` and `bridge-sui.move` contracts completely lack an emergency pause mechanism. The contract does not implement any pause functionality, administrator access control, or emergency stop mechanisms, which means that once security vulnerabilities are discovered, attacks occur, or abnormal situations arise, critical contract operations cannot be immediately stopped.

Some scenarios

- Need to pause immediately when smart contract vulnerabilities are discovered
- Need to stop urgently when under hacker attacks
- Need to pause services during system upgrades or maintenance
- Need to stop operations due to regulatory requirements or legal disputes

Suggestion:

Add pause mechanism.

BBS-5 Lack of Fund Refund Processing

Severity: Informational

Status: Acknowledged

Code Location:

bridge-bsc.sol#60

Descriptions:

When a cross-chain operation on the BSC chain successfully executes (funds are transferred from the user account to the contract), but the corresponding transfer operation on the SUI chain fails, the system does not provide a fund return mechanism, resulting in the user's funds being permanently lost.

Suggestion:

Adding an off-chain refund mechanism.

BBS-6 Useless `signature` Parameter

Severity: Informational

Status: Acknowledged

Code Location:

bridge-bsc.sol#64

Descriptions:

In the `bridge` function of the `bridge-bsc.sol` contract, the signature parameter is defined but never validated or used. This parameter is only checked for emptiness and then directly included in events without any security validation functionality, making it a useless redundant parameter.

Suggestion:

Remove useless parameter.

BSU-1 Third Parties Can Forge Sui `TransferEvent`

Severity: Medium

Status: Fixed

Code Location:

bridge-sui.move#108

Descriptions:

Since the `init_token_bridge` function is public, any third party can create their own `StateObject` object and then call the `transfer_token` function by adding funds, thereby forging `TransferEvent` events.

And since "CoinType" is not in the "TranferEvent", attacks can even forge `TransferEvent` events without any cost.

```
// Event fields that can be controlled by attacker
event::emit(TransferEvent {
    recipient,  // forgeable
    amount,     // forgeable
    tx_hash,    // forgeable
    sender,     // not forgeable (from tx_context::sender)
    // it's better to add "CoinType" to the event !!!!!!
});
```

Attack Flow

1. Attacker calls `init_token_bridge` to create their own `StateObject`
2. Attacker adds token funds to the pool through `add_funds`
3. Attacker calls `transfer_token` function, controlling the following parameters:
    - `recipient` : arbitrary recipient address
    - `amount` : arbitrary amount (limited by pool balance)

    - `tx_hash` : arbitrary transaction hash string

4. The function emits a `TransferEvent` where only the `sender` field cannot be forged (from `tx_context::sender(ctx)`)

Impact

- **Monitoring Systems**: Event-based monitoring systems may receive false data

Suggestion:

Always check `TransferEvent.sender` offchain.And add "CoinType" to the event

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.