SuiDollar

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Wed Oct 09 2024

SuiDollar Audit Report

1 Executive Summary

1.1 Project Information

Description	SuiDollar is a financial management protocol.
Type	DeFi
Auditors	MoveBit
Timeline	Tue Oct 08 2024 - Wed Oct 09 2024
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/suidollar/suidollar-usdc-vault/
Commits	e2a3dea0ee7b072d7487f68aa94f433a00d2a44b

1.2 Files in Scope

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

ID	File	SHA-1 Hash
SUI2	sources/suidollar.move	9ed5dbd6f16a0e87edeef0510383a77b075ecb78

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	3	2	1
Informational	1	1	0
Minor	0	0	0
Medium	1	1	0
Major	1	0	1
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

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 SuiDollar to identify any potential issues and vulnerabilities in the source code of the SuiDollar 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 3 issues of varying severity, listed below.

ID	Title	Severity	Status
SUI-1	Centralization Risk	Major	Acknowledged
SUI-2	Missing Parameter Validation	Medium	Fixed
SUI-3	Lack of Events Emit	Informational	Fixed

3 Participant Process

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

Owner

- The owner can call the `deposit` function to deposit coin to the pool.
- The owner can call the `withdraw` / `userWithdraw` to withdraw the coin to the `BLUEFIN_ADDRESS` or user.
- The owner can call the `withdrawFees` function to withdraw the fee.
- The owner can use the `depositNavi` / `withdrawNavi` / `borrow` / `repay` function to manage the coin in navi protocol.

User

- Users can call the `userDeposit` function to deposit coin.

4 Findings

SUI-1 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

sources/suidollar.move#63

Descriptions:

Centralization risk was identified in the smart contract.

- The `BLUEFIN_ADDRESS` address can call the `withdraw` function to extract the coin from the `_pool` .
- The `BLUEFIN_ADDRESS` address can use the `depositNavi` / `withdrawNavi` / `borrow` / `repay` function to manage the coin in the pool with navi protocol.
- When users call the `userDeposit` function in the contract to stake, they will not obtain any certificates or records. When users withdraw funds, they need to call the `userWithdraw` function proxy with `BLUEFIN_ADDRESS` to withdraw funds. There is no correlation check between the withdrawn amount and the staked amount. The contract owner can withdraw funds from the pool through the `withdraw` function, which may affect the user's fund security.

Suggestion:

It is recommended to take ways to reduce the risk of centralization.

Resolution:

Client response: The contract uses a centralized processing method to enable the contract to interact with the Navi protocol and process user deposits/withdrawals.

SUI-2 Missing Parameter Validation

Severity: Medium

Status: Fixed

Code Location:

sources/suidollar.move#53

Descriptions:

The `userDeposit` function does not check if the `amount` is 0, it may cause meaningless transactions or events. And if `total_amount` is less than `10000/25`, the user can bypass the `fee_amount`.

Suggestion:

It is recommended to limit the minimum value of `total_amount`.

SUI-3 Lack of Events Emit

Severity: Informational

Status: Fixed

Code Location:

sources/suidollar.move

Descriptions:

The contract lacks appropriate events for monitoring operations, such as `userDeposit`, `userWithdraw` functions which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for the function.

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.