MOVEBIT
Securing the Move Ecosystem

Suia Smart Contract Audit Report

https://twitter.com/movebit_

contact@movebit.xyz

Suia Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	Suia is a social dApp built on SuiNetwork
Type	NFT
Auditors	MoveBit
Timeline	May 4, 2023 – May 9, 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Mynft/suia https://github.com/Mynft/suia-token
Commits	7805f060b67db624e02cd114fa0523aa6173fe81 c33d2beea1a3a4b2ca4e0b72525085bbd359e532

1.2 Files in Scope

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

ID	Files	SHA-1 Hash

SUA	suia/move_packages/suia/sources/suia.move	2e4b5d320ec9c12902ccf3266fecb7dabe5c6af9
SCY	suia/move_packages/suia_capy/sources/suia_capy.move	d627d84ab2d83603bcb6c84a302917464ceb9e05
STK	suia-token/sources/suia_token.move	6cc656fec1826cbda67f9fc1faeb6148bd1f18b5

1.3 Issue Statistic

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

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 **Suia** to identify any potential issues and vulnerabilities in the source code of the **Suia** 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
STK-01	Centralization Risk	Major	Acknowledged
SCY-02	Unused Constant	Minor	Acknowledged
SCY-03	Code Can't be Compiled Correctly	Minor	Acknowledged

3 Participant Process

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

Admin

- Admin can create an `item` and send it to receipt through `create_and_send_item()`.
- Admin can mint `SUIA_TOKEN` through `mint()`.
- Admin can burn `SUIA_TOKEN` through `burn()`.

Whitelist User

- Whitelist User can claim the `Medal` and get a `PersonalMedal` through `claim_medal()`.

User

- User can create a new `Medal` through `create_medal()`.
- User can wrap the `Capy` with `SuiaCapyItem` through `wrap_capy_with_item()`.
- User can wrap the `SuiaCapy` with `SuiaCapyItem` through `wrap_suia_capy_with_item()`.

4 Findings

STK–01 Centralization Risk

Severity: Major

Status: **Acknowledged**

Code Location: suia–token/sources/suia_token.move.

Descriptions: There is a risk of centralization, with privileged accounts able to mint unlimited tokens and burn their token.

Suggestion: It is recommended that multi–signature accounts should be set as privileged accounts.

SCY–02 Unused Constant

Severity: Minor

Status: **Acknowledged**

Code Location: suia/move_packages/suia_capy/sources/suia_capy.move#L15, L16.

Descriptions: The constants `ENOT_ADMIN` and `EIINVALID_SUIA` are not used in `suia_capy.move`.

Suggestion: It is recommended that unused variables should be removed.

SCY–03 Code Can't be Compiled Correctly

Severity: Minor

Status: **Acknowledged**

Code Location: suia/move_packages/suia_capy/sources/suia_capy.move.

Descriptions: The `suia_capy.move` can not be compiled correctly with the local `sui` and `sui–capybaras` dependencies.

Suggestion: It is recommended to use the dependencies from the official `sui` and `sui–capybaras` repositories.

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.