Seapad

Smart Contract

Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

05/30/2023

Seapad Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	A standard token on Sui.
Type	Token
Auditors	MoveBit
Timeline	May 29, 2023 – May 30, 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/seapad-fund/sui-contracts/blob/mainnet/tokens
Commits	8b9085eab5e131d35beaaed6cfe42488a669b9d4 ffaf0d6018a514a3aa991af72d7f81c81c9cdd71

1.2 Files in Scope

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

ID	Files	SHA-1 Hash
SPT	tokens/sources/spt.move	42f2fb46558a10b951d2ae60a7270624169af701
MOV	tokens/Move.toml	f2f828c3bb29ff12b60478d728065ddf8ab404f9

1.3 Issue Statistic

Item	Count	Fixed	Partially Fixed
Total	3	2	1
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 **Seapad** to identify any potential issues and vulnerabilities in the source code of the **Seapad 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 have identified **3** issues of varying severity, listed below.

ID	Title	Severity	Status
SPT-01	Missing Emit Event	Minor	Fixed
SPT-02	Centralization Risk	Major	Partially Fixed
MOV-03	Redundant Third-party Dependency References	Minor	Fixed

3 Participant Process

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

Admin

- Admin can mint unlimited tokens through `mint_balance()` .
- Admin can burn itself tokens through `burn()` .
- Admin can freeze the mint capability through `burn_cap()` .

4 Findings

SPT-01 Missing Emit Event

Severity: Minor

Status: Fixed

Code Location: tokens/sources/spt.move#L37

Descriptions: The smart contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track important actions or detect potential issues.

Suggestion: It is recommended to emit events for the function `burn_cap` .

Resolution: The client followed our suggestion and addressed this issue.

SPT-02 Centralization Risk

Severity: Major

Status: Partially Fixed

Code Location: tokens/sources/spt.move

Descriptions: The contract's design and implementation introduce elements that could undermine the decentralized nature of the system.

- Admin can mint unlimited tokens through `mint_balance()` .
- Admin can burn itself tokens through `burn()` .
- Admin can freeze the mint capability through `burn_cap()` .

Suggestion: To mitigate the centralization risk in the smart contract, we recommend the following solutions:

- Foster community governance and participation to ensure decision-making power is distributed among the system's participants.
- Implement robust security measures to protect against potential attacks or malicious actions, such as multi-signature.

Removing all centralized methods can be considered to permanently address centralized risks.

Resolution: The client adopts Sui native multisign feature on the treasury wallet to mitigate this issue.

MOV-03 Redundant Third-party Dependency References

Severity: Minor

Status: Fixed

Code Location: tokens/Move.toml#L8

Descriptions: The project contains unused references to third-party dependencies. These references serve no purpose and can be considered unnecessary clutter within the codebase.

Suggestion: It is recommended to remove the unused dependency reference.

Resolution: The client followed our suggestion and addressed this issue.

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