Sparkling Finance

Audit Report

contact@movebit.xyz

https://twitter.com/movebit_

Fri Apr 26 2024

Sparkling Finance Audit Report

1 Executive Summary

1.1 Project Information

Description	A protocol for pledging for incentives
Type	DeFi
Auditors	MoveBit
Timeline	Wed Apr 24 2024 - Fri Apr 26 2024
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Bucket-Protocol/v1-core
Commits	968c804e55614f5e9375d805af08caeadde4443d

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MBU	flask/tests/mock_buck.move	85936240c0e2b94e6f4d8e4b75f67ac6c6646be5
STE	flask/tests/sbuck_tests.move	76a96651070d800bda079c436e4b70f33ef3b5cd
MOV1	flask/Move.toml	9f2ec21e09af0bdb4efcd4405b2a98bf72bb5685
EVE	flask/sources/event.move	5abd29eea750a8ea6bbc4a2a0951fdd7f22e3ab5
UTI	flask/sources/utils.move	36c368f3e9bdc9483c566ade24a5ecf24c0166cf
SBU	flask/sources/sbuck.move	f2d25c0f6317ff905a4f2bdfc375cbdd37305487
WBC	WBC.sol	728809972317763dd9b9ef4d315b085e1c86c2b0

1.3 Issue Statistic

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

1.4 MoveBit Audit Breakdown

ScaleBit 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
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

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) 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 Sparkling Finance to identify any potential issues and vulnerabilities in the source code of the Sparkling Finance - Flask 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
SBU-1	Share Manipulation Vulnerability	Critical	Acknowledged
SBU-2	Missing The Functionality For Version Upgrades	Major	Acknowledged
SBU-3	Risk of Withdrawing Unearned Rewards Upon System Recharge	Major	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the Sparkling Finance - Flask Smart Contract :

Owner

- The owner can call the `initialize` function to initialize the flask.

User

- The user can call the `collect_rewards` function to add reward tokens to the contract
- The user can call the `deposit` function to pledge tokens into the contract.
- The user can call the `withdraw` function to withdraw the pledged tokens as well as the earned rewards.

4 Findings

SBU-1 Share Manipulation Vulnerability

Severity: Critical

Status: Acknowledged

Code Location:

flask/sources/sbuck.move#85-138

Descriptions:

Here exists a classic ERC4626 model share manipulation vulnerability. Although it is mitigated here by `assert!(minted_sbuck_val > 0, ERR_INSUFFICIENT_DEPOSIT);`, this is not an effective measure. The following example illustrates this:

1. Suppose the coin deposited is SUI, and initially, the attacker stakes 1 MIST to receive 1 SBUCK.
2. Bob, the victim, attempts to deposit 2 SUI into the contract. If the deposit were to occur at this point, Bob would receive (1 SBUCK * 2e9 MIST) / 1 MIST = 2e9 SBUCK.
3. Before Bob's transaction, the attacker transfers 1 SUI through the `collect_rewards` function.
4. Bob's current deposit calculation is now (1 SBUCK * 2 SUI) / (1 SUI + 1 MIST).
5. Due to truncation, the calculation yields 1 SBUCK share. This calculation passes the non-zero share check.
6. The attacker withdraws SUI using the `withdraw` function, now valued at approximately 1.5 SUI because the contract holds 3 SUI + 1 MIST, while the contract has only minted 2 SBUCK.

Suggestion:

It is recommended to implement permission controls on collect_rewards or incorporate a Burn mechanism for SBUCK shares during the initial deposit.

SBU-2 Missing The Functionality For Version Upgrades

Severity: Major

Status: Acknowledged

Code Location:

flask/sources/sbuck.move#28-30

Descriptions:

In this contract, the functionality for version upgrades is absent. Upon contract upgrade, the contract will become invalid.

```
fun assert_pacakge_version<T>(self: &Flask<T>){
    assert!(self.version == VERSIOIN, ERR_WRONG_VERSION);
}
```

Suggestion:

It is recommended to add the functionality for version upgrades.

SBU-3 Risk of Withdrawing Unearned Rewards Upon System Recharge

Severity: Major

Status: Acknowledged

Code Location:

flask/sources/sbuck.move#97

Descriptions:

When all participants have withdrawn their balances, `sbuck_supply` becomes 0, while rewards from previous deposits may still be pending. If the project deposits rewards (e.g., 10,000 tokens) at this point, `reserves` would be 10,000 while `sbuck_supply` remains at 0. If a user deposits tokens at this moment, they can immediately withdraw the deposited tokens along with the pending rewards.

Suggestion:

It is recommended to consider adding a limit in `collect_rewards()` to prevent rewards from being deposited if both `reserves` and `sbuck_supply` are 0.

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.