Random-Vault

Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

Fri Feb 02 2024

Random-Vault Audit Report

1 Executive Summary

1.1 Project Information

Description	This is a Lottery project
Type	DeFi
Auditors	MoveBit
Timeline	Mon Jan 22 2024 - Fri Feb 02 2024
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/haedallsd/random_vault/ https://github.com/haedallsd/random_vault
Commits	https://github.com/haedallsd/random_vault/: d736a881321d9610a2cbcaa414ce0839b4f2de55 fc916c637ddab32d05b16af8aa39cb74d50bc3f2 https://github.com/haedallsd/random_vault: 5ffcf8db769bda8dde698b92a8bfca0bbe9e83ab f96938886c5b6de91cccae7bebff225c9da9b465

1.2 Files in Scope

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

ID	File	SHA-1 Hash
TLO	tests/test_lottery.move	5e33c11a5a67d9a0e537779a139d6df8684776ac
MOV	Move.toml	37fca438f2191cb38e6e70ffff64b0ad74120837
LOT	sources/lottery.move	2609a631d9745b3dab53f8876132218f919999e3
ADM	sources/admin.move	6215275fc4f328c5073be6b33aa8e333ebea5d42
CON	sources/config.move	d9af5bac87c35ff1542d703bb2ad972cac4d96ca
USE	sources/user.move	fa31e9e7cbf041c2cfc16f290a3cd7f9ca230b5a

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	6	6	0
Informational	0	0	0
Minor	4	4	0
Medium	1	1	0
Major	1	1	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

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 Haedal to identify any potential issues and vulnerabilities in the source code of the lottery 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 6 issues of varying severity, listed below.

ID	Title	Severity	Status
CON-1	`set_prize_rate` Should Check If It Is New Rate	Minor	Fixed
LOT-1	`deposit` Does Not Update User's Share If He Deposits Multiple Times	Major	Fixed
LOT-2	Valid Prizes Can Be Collected As Expired	Medium	Fixed
LOT-3	`migrate` Cannot Be Called	Minor	Fixed
LOT-4	`pause` and `resume` Functions Should Check States First	Minor	Fixed
LOT-5	`queryWinRate` Does Not Handle User Withdrawed	Minor	Fixed

3 Participant Process

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

Admin

- `Admin` can add an operator through `add_operator`.
- `Admin` can collect fees through `collect_fee`.
- `Admin` can migrate the protocol through `migrate`.

Operator

- `operator` can destroy `OperatorCap` through `destroy_operator`.
- `operator` can start first round through `start_first_round`.
- `operator` can pause the protocol through `pause`.
- `operator` can resume the protocol through `resume`.
- `operator` can draw a vault through `draw`.
- `operator` can set round duration through `set_round_duration`.
- `operator` can set the prize rate through `set_prize_rate`.
- `operator` can set the withdraw fee rate through `set_withdraw_fee_rate`.
- `operator` can clear rounds through `clear_round_record`.

User

- `User` can collect expired prizes via `collect_expired_prize`.
- `User` can deposit SUI via `desposit`.
- `User` can withdraw haSUI via `withdraw`.
- `User` claims the prizes via `claim`.

4 Findings

CON-1 `set_prize_rate` Should Check If It Is New Rate

Severity: Minor

Status: Fixed

Code Location:

sources/config.move#38-41

Descriptions:

In `config.move`, there should be assersion to check that in `set_prize_rate` function, `config.prize_rates != rates`.

Suggestion:

It is suggested to add the assertion.

Resolution:

It is fixed by the client by adding the checks.

LOT-1 `deposit` Does Not Update User's Share If He Deposits Multiple Times

Severity: Major

Status: Fixed

Code Location:

sources/lottery.move#363-365

Descriptions:

In `lottery.move`, `deposit` function will allow users to deposit certain `SUI` and increase their share for the price calculation.

However, if a user have deposited multiple times with no withdraw, his share would not be updated while `round.total_share` would.

This result in great loss of the user since he deposits but gets no new shares.

```
// if user has withdrawed before, he has no share in this round, just ignore him
        if (ui.withdrawed_amount == 0) {
            round.total_share = round.total_share + increased_share; //@audit ui.share is not updated
        };
```

Suggestion:

It is suggested to add the `increased_share` to the `ui.share` inside the if condition.

Resolution:

It is fixed by the client to update the `ui.share` when the user deposits several times.

LOT-2 Valid Prizes Can Be Collected As Expired

Severity: Medium

Status: Fixed

Code Location:

sources/lottery.move#250-252;

sources/lottery.move#598

Descriptions:

In the `lottery.move`, `update_old_round` will be used to select the winners and deliver the prize.

And in the creation of the winner ticket, the time of the expiration is the time of calling the `update_old_round` function, which may be later than `round.end_time`.

```
assert!(round.end_time<=now, EInvalidArgument);
expire_time: now + 86400 * 60, // must be claimed within 60 days
```

However, in the `collect_expired_prize` function, anyone can collect the prize that is over the `round.end_time`

```
if (now < round.end_time + 86400 * 60)
```

This means, that if the time difference is large enough, the valid prize may be collected as expired.

Suggestion:

It is suggested to use the same time for claiming and collecting.

Resolution:

It is fixed by the client by changing the `expire_time` from `now+86400*60` to `round.end_time+86400*60`.

LOT-3 `migrate` Cannot Be Called

Severity: Minor

Status: Fixed

Code Location:

sources/lottery.move#304-308

Descriptions:

In `lottery.move`, `migrate` friend function is used to migrate the old `lottery_object.version` to `PROGRAM_VERSION`, but in the only friend module `admin.move` there is no such function that can call `migrate`. Resulting in this uncalleable situation.

Suggestion:

It is suggested to add a function in `admin.move` that can call `migrate` function.

Resolution:

It is fixed by the client by adding a `migrate` function in `admin.move`.

LOT-4 `pause` and `resume` Functions Should Check States First

Severity: Minor

Status: Fixed

Code Location:

sources/lottery.move#188-194

Descriptions:

The `lottery.move`, `pause`, and `resume` functions are used in emergencies to pause the protocol and resume it.

However, typically it should be checked if it's already paused or unpaused first. And the related event would be emitted as well.

Suggestion:

It is suggested to check the state of `lottery_object` and emit events for pausing and resuming the `lottery`.

Resolution:

It is fixed by the client by adding the events for both functions.

LOT-5 `queryWinRate` Does Not Handle User Withdrawed

Severity: Minor

Status: Fixed

Code Location:

sources/lottery.move#495-507

Descriptions:

In the design of the `lottery.move` once a user withdraws any amount in a round, he will not be able to participate in the lottery, which means he has a win rate of 0.

However, the `queryWinRate` function, does not handle such a case but directly computes the win rate using its share.

This will lead to confusion of the user and a poor user experience.

Suggestion:

It is suggested to modify the `queryWinRate` function so that users who withdraw before will have no win rate.

Resolution:

It is fixed by the client by adding the filter.

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.