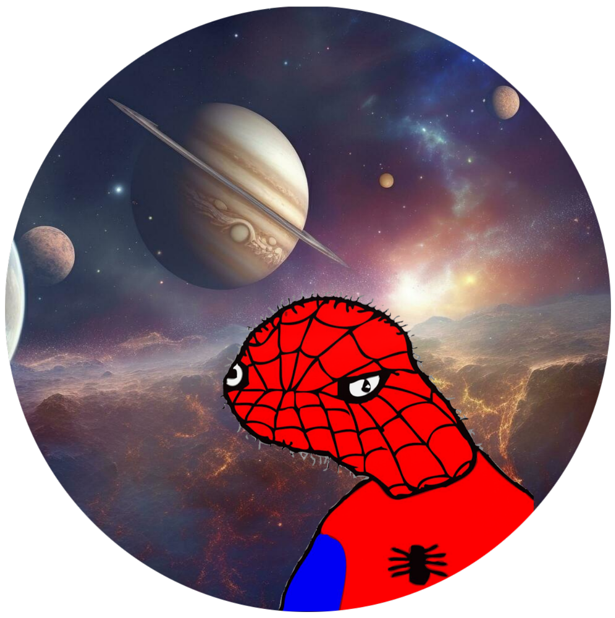

Solomon Labs - Vault program
Audit Report

Version 1.1

Zigtur

July 22, 2024

Solomon Labs - Vault program - Audit Report

Zigtur

July 22, 2024

Prepared by: Zigtur

Table of Contents

Introduction

Disclaimer

A smart contract security review cannot guarantee the complete absence of vulnerabilities. This effort, bound by time, resources, and expertise, aims to identify as many security issues as possible. However, there is no assurance of 100% security post-review, nor is there a guarantee that the review will uncover all potential problems in the smart contracts. It is highly recommended to conduct subsequent security reviews, implement bug bounty programs, and perform on-chain monitoring.

About Zigtur

Zigtur is an independent blockchain security researcher dedicated to enhancing the security of the blockchain ecosystem. With a history of identifying numerous security vulnerabilities across various protocols in public audit contests and private audits, **Zigtur** strives to contribute to the safety and reliability of blockchain projects through meticulous security research and reviews. Explore previous work here or reach out on X @zigtur.

About Solomon Labs

Solomon Labs is building a stablecoin-like yield protocol powered by perpetual funding payments.

Security Assessment Summary

Review commit hash - 2819c01d0fc59d14b3c495f8d64254ec35adea93

Fixes review commit hash - 030cacf93095f28621cbec476b3d5d5441bd04a9

Deployment chains

- Solana

Scope

This audit focuses on the Vault program. The following files are in scope of the review:

- vault/src/lib.rs
- vault/src/context.rs

Risk Classification

	Impact: High	**Impact:** Medium	**Impact:** Low
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Issues Count

A total of **15 issues** have been identified and can be classified as:

- **1 HIGH**
- **2 MEDIUM**
- **6 LOW**
- **6 INFO**

The mitigation review shows that 11 issues were fixed.

4 issues are acknowledged by Solomon Labs (`INFO-01` , `INFO-03` , `INFO-04` , `INFO-06`).

Issues

HIGH-01 - Deposit rate is used instead of redeem rate

Description

Scope:

- lib.rs#L181

The `redeem` function calculates the output amount of collateral tokens with the `deposit_rate` value instead of the `redeem_rate` value.

This will calculate an incorrect output amount of collateral tokens, leading to potential loss of protocol funds.

Code snippet

The `redeem` function uses the `deposit_rate` instead of the `redeem_rate`.

```
  pub fn redeem(ctx: Context<Redeem>, amt: u64) -> Result<()> {
    let rate = ctx.accounts.exchange_rate.deposit_rate as u128;
```

Recommendation

`redeem` should use the `redeem_rate` value.

```
  pub fn redeem(ctx: Context<Redeem>, amt: u64) -> Result<()> {
    let rate = ctx.accounts.exchange_rate.redeem_rate as u128;
```

Resolution

SolomonLabs Team: Fixed following recommendation.

Zigtur: Fix reviewed and approved.

MEDIUM-01 - Unsafe casting to `u64` in `convert_to_shares` and `convert_to_assets`

Description

- lib.rs#L133
- lib.rs#L183

The `deposit` and `redeem` functions are doing calculations with `u128` type. The result of the calculations are then casted into an `u64` type.

However, these casting are not safe.

Note: This issue is not likely to happen, but it would have high impact on the protocol.

Code snippet

The casting from `u128` to `u64` is done with `as` keyword.. This casting is not safe.

```rust
    let amt = (collat as u128 * rate / DECIMALS_SCALAR) as u64;
```

Recommendation

Consider replacing the `as u64` casting with a casting that reverts if the value is greater than `u64::MAX`. For example, `try_into().unwrap()` will do this check.

```rust
    let amt: u64 = (collat as u128 * rate / DECIMALS_SCALAR).try_into().unwrap();
```

A patch is given in Appendix to fix this issue.

Resolution

SolomonLabs Team: Fixed. Provided patch applied.

Zigtur: Fix reviewed and approved.

MEDIUM-02 - Rates are not on the same scale

Description

- lib.rs#L132-L133
- lib.rs#L181-L183

The `deposit_rate` and the `redeem_rate` are not on the same scale. This issue is not a problem if the administrator correctly sets these two values.

However, using different scaling for these two rates is prone to errors from the admin.

Incorrect Scenarios

According to the tests, both the `deposit_rate` and `redeem_rate` are set to `1_000_000_000` (1e9). Here are the calculations with these values.

A scenario with **incorrect** deposit rate for 1:1 value:

- deposit 1000 USDC => `1000_000_000` (1000e6)
- `deposit_rate = 1_000_000_000` (**1e9**)
- `DECIMALS_SCALAR = 1_000_000_000` (1e9)
- vault token output amount = 1000e6 * 1e9 / 1e9 = 1000e6

This 1000e6 output amount is incorrect. As the vault token is 9 decimals based, this amount corresponds to 1 vault token for 1,000 USDC.

A scenario with **incorrect** redeem rate for 1:1 value (same as deposit rate):

- redeem 1000 vault token => `1000_000_000_000` (1000e9)
- `redeem_rate = 1_000_000_000` (**1e9**)
- collateral decimals (USDC) = 6
- collateral output amount = 1000e9 * 1e9 / 1e6 = 1000e12

This 1000e9 collateral amount is incorrect. As the vault token is 9 decimals based, this amount corresponds to 1,000,000 USDC for 1,000 vault tokens.

Correct Scenarios

A scenario with **correct** deposit rate for 1:1 value:

- deposit 1000 USDC => `1000_000_000` (1000e6)
- `deposit_rate = 1_000_000_000_000` (**1e12**)
- `DECIMALS_SCALAR = 1e9`
- vault token output amount = 1000e6 * 1e12 / 1e9 = 1000e9

This 1000e9 output amount is correct. As the vault token is 9 decimals based, it gives 1,000 vault tokens for 1,000 USDC.

A scenario with **correct** redeem rate for 1:1 value:

- redeem 1000 vault token => `1000_000_000_000` (1000e9)
- `redeem_rate = 1_000` (**1e3**)
- collateral decimals (USDC) = 6
- collateral output amount = 1000e9 * 1e3 / 1e6 = 1000e6

This 1000e3 collateral amount is correct. As the vault token is 9 decimals based, it gives 1,000 USDC for 1,000 vault tokens.

As we can see, for interaction with USDC tokens (6 decimals based), the `deposit_rate` must be based on 12 decimals and the `redeem_rate` must be based on 3 decimals to get a result close to a 1:1 valuation.

Recommendation

The deposit and redeem calculations should be reviewed to be based on the same scale. This will help avoiding errors from the admin and will provide sufficient precision.

These calculations should be based on the decimals of both tokens to correctly scale the rate and resulting amounts.

Note: These fixes are heavy and a patch couldn't be provided during the audit.

Resolution

SolomonLabs Team: Fixed. Scales have been adjusted for deposit and redeem to keep a rate with 9 decimals (`1_000_000_000` gives a 1:1 rate valuation).

Zigtur: Fix reviewed and approved. Note that tokens with decimals greater than 9 **are not supported and must not be used**.

LOW-01 - `add_withdraw_address` function doesn't cap the number of withdraw addresses

Description

- lib.rs#L415-L436

The `add_withdraw_address` is called by the admin to add a whitelisted address to which funds can be withdrawn.

However, the function does not cap the number of withdraw addresses.

This means that more addresses than expected can be set in `vault_state.withdraw_addresses`, and this even if the vector is defined with a capacity of 50.

Proof of Concept

A PoC is available in Appendix to show the issue.

Recommendation

The length should be ensure to not exceed 50 for `vault_state.withdraw_addresses`.

A patch is given in Appendix to fix this issue along with `LOW-02` and `LOW-03`.

Resolution

SolomonLabs Team: Fixed. Provided patch applied.

Zigtur: Fix reviewed and approved.

LOW-02 - `add_asset_manager` function doesn't cap the number of asset managers

Description

- lib.rs#L327-L348

The `add_asset_manager` is called by the admin to add an asset manager address.

However, the function does not cap the number of asset manager addresses.

This means that more addresses than expected can be set in `vault_state.asset_managers`, and this even if the vector is defined with a capacity of 20.

Recommendation

The length should be ensure to not exceed 20 for `vault_state.asset_managers`.

A patch is given in Appendix to fix this issue along with `LOW-01` and `LOW-03`.

Resolution

SolomonLabs Team: Fixed. Provided patch applied.

Zigtur: Fix reviewed and approved.

LOW-03 - `add_role_manager` function doesn't cap the number of role managers

Description

- lib.rs#L350-L370

The `add_role_manager` is called by the admin to add a role manager address.

However, the function does not cap the number of manager addresses.

This means that more addresses than expected can be set in `vault_state.role_managers`, and this even if the vector is defined with a capacity of 20.

Recommendation

The length should be ensure to not exceed 20 for `vault_state.role_managers`.

A patch is given in Appendix to fix this issue along with `LOW-01` and `LOW-02`.

Resolution

SolomonLabs Team: Fixed. Provided patch applied.

Zigtur: Fix reviewed and approved.

LOW-04 - New exchange rates are not included into event emission

Description

- lib.rs#L123-L126
- lib.rs#L485-L489

The `AssetModifiedEvent` structure used to emit event in `update_asset` does not include the new exchange rates.

Recommendation

Add two fields to `AssetModifiedEvent`: one for the new `deposit_rate` and one for the new `redeem_rate`.

A patch is given in Appendix to fix this issue.

Resolution

SolomonLabs Team: Fixed. Provided patch applied.

Zigtur: Fix reviewed and approved.

LOW-05 - `NewManagerEvent` and `ManagerRemovedEvent` events are not precise enough

Description

- lib.rs#L342-L345
- lib.rs#L365-L368
- lib.rs#L386-L389
- lib.rs#L407-L410

The `NewManagerEvent` event is emitted when a manager is added to `role_managers` or to `asset_managers`. The `ManagerRemovedEvent` event is emitted when a manager is removed from `role_managers` or from `asset_managers`.

These two events make no difference between role managers and asset managers.

Recommendation

Consider creating four (4) different events instead of two (2).

A patch is given in Appendix to fix this issue.

Resolution

SolomonLabs Team: Fixed. Provided patch applied.

Zigtur: Fix reviewed and approved.

LOW-06 - `DepositEvent`, `WithdrawEvent` and `RedeemEvent` events are not precise enough

Description

- lib.rs#L172-L175
- lib.rs#L222-L225
- lib.rs#L257-L260
- lib.rs#L491-L507

The `DepositEvent`, `WithdrawEvent` and `RedeemEvent` are used to emit events during `deposit`, `withdraw` and `redeem` functions.

However, each of this function supports multiple tokens simultaneously. The events don’t allow to identify which token is used.

Recommendation

A `token_mint` field should be added to all these event structures.

A patch is given in Appendix to fix this issue.

Resolution

SolomonLabs Team: Fixed. Provided patch applied.

Zigtur: Fix reviewed and approved.

INFO-01 - `initialize_vault_state` can be front-runned

Description

- lib.rs#L60

The `initialize_vault_state` function does not have access control set.

An attacker could front-run the initialization transaction.

Recommendation

An access control could be hardcoded for this function.

Another way to mitigate the issue is to ignore the deployed program if anyone front-runs the initialization.

Resolution

SolomonLabs Team: Acknowledged.

Zigtur: Acknowledged.

INFO-02 - Typo issue in comment

Description

- lib.rs#L42

In `ExchangeRate` structure, a comment indicates `sclaed` instead of `scaled`.

Recommendation

Replace `sclaed` with `scaled`.

Resolution

SolomonLabs Team: Fixed.

Zigtur: Fix reviewed and approved.

INFO-03 - `Managers`, `WithdrawAddresses` and `TransferAdmin` are the same structure

Description

- context.rs#L254-L288

The `Managers`, `WithdrawAddresses` and `TransferAdmin` structures are used by different functions.

However, all these structures are the same. They only define two fields: `caller` and `vault_state`.

Recommendation

All these can be merged into a generic one to reduce the codebase size.

Resolution

SolomonLabs Team: Acknowledged.

Zigtur: Acknowledged.

INFO-04 - Admin needs to be trusted

Description

- lib.rs#L230

There is a `withdraw` functionality which allows an asset manager to withdraw funds to one of the `withdraw_addresses`.

Because the admin can set both the asset managers and the withdraw addresses, the admin can drain all funds from the vault program.

Recommendation

None.

According to documentation, the admin address will be a multisig wallet with timelock functionalities.

Resolution

SolomonLabs Team: Acknowledged.

Zigtur: Acknowledged.

INFO-05 - Incorrect comments and variable names indicate ATA when it is not

Description

- context.rs#L106
- context.rs#L166
- lib.rs#L18

In multiple comments and lines of code, the term “ATA” is used.

However, it is incorrectly used. Simple “Token Accounts” are used in the current codebase and not “Associated Token Accounts”.

Note: An ATA can be initialized by anyone while a Token Account can only be initialized by the owner. This difference can have security impacts.

Recommendation

Consider fixing the comments to indicate “Token Account” instead of “ATA”.

Resolution

SolomonLabs Team: Fixed.

Zigtur: Fix reviewed and approved.

INFO-06 - Disabling a collateral is not direct

Description

- lib.rs#L120-L121
- lib.rs#L135-L137
- lib.rs#L185-L187

The vault program does not have a function to directly disable a previously supported collateral token.

Recommendation

None.

The way to get this functionality is to use `update_asset` and set the deposit and redeem rates to `0`.

Resolution

SolomonLabs Team: Acknowledged.

Zigtur: Acknowledged.