THE ARCADIA GROUP

Security Audit Report

Hubble

12/22/2021

PREPARED FOR:
Axiom Markets LTD, AKA Hubble

ARCADIA CONTACT INFO
Email: audits@arcadiamgroup.com
Telegram: https://t.me/thearcadiagroup

THE ARCADIA GROUP

Table of Contents

Executive Summary

A Representative Party of **Axiom Markets LTD** ("**CLIENT**") engaged The Arcadia Group ("Arcadia"), a software development, research, and security company, to conduct a review of the following **Hubble Markets** smart contracts on the **hubble-markets (hubble-markets/hubble/tree/master/programs)** github repository at Commit #**e2c322f999ed2e53505ba532eff24a4d6a38b9a4**.

The scope of this audit included the following files:

1. **Borrowing/***
2. **token-derive/***

Arcadia completed this security review using various methods primarily consisting of dynamic and static analysis. This process included a line-by-line analysis of the in-scope contracts, optimization analysis, analysis of key functionalities and limiters, and reference against intended functionality.

There were **16** issues found, **01** of which were deemed to be ‘critical’ (“HUBBLE-5”), and **00** of which were rated as ‘high’.

Severity Rating	Number of Original Occurrences	Number of Remaining Occurrences
CRITICAL	01	0
HIGH	0	0
MEDIUM	01	0
LOW	0	0
INFORMATIONAL	14	6

Findings

1. Use token accounts of pda address in 'initialize'

Issue: **HUBBLE-1**
Severity: **INFORMATIONAL**

Target: **handler_initialize_borrowing_market.rs**
Finding Type: **DYNAMIC**

In `initialize_borrowing_market` instruction, there is a process to transfer token authority to the pda address.

These token accounts are accounts which do not hold any tokens before initialization, so we can directly create an account with a pda address, and set them in `initialize_borrowing_market`.

In `initialize_stability_pool`, and `staking_initialize`, we can optimize in the same way.

https://github.com/step-finance/reward-pool/blob/main/programs/reward-pool/src/lib.rs#L683-L689

https://github.com/step-finance/reward-pool/blob/028f1a2d911ffc3eba2aa6f762f8d60101b5ec8d/tests/user.js#L74

You can check how Step Finance does initialization with pda owned accounts.

Note that finding the program address on the chain and transferring authority to it will take an extra transaction fee.

Action Recommended:

Remove transferring authority and create token accounts for pda addresses before initialising.

Remediation:

This issue has been added to Hubble Protocol's backlog in github issue #87

2. `assert_permissions in handler_approve.trove.rs` can be handled by anchor annotations.

Issue: **HUBBLE-2**
Severity: **INFORMATIONAL**

Target: **handler_approve_trove.rs**
Finding Type: **DYNAMIC**

In `handler_approve.trove.rs`, there is an `assert_permissions` function which checks if `stablecoin_ata` is the correct one.

This can be checked by anchor annotations in account definition using constraint.

Action Recommended:

Check possible permissions by using constraint in account definition. And please consider `asset_permissions` in other handlers as well.

Remediation:

Hubble now utilizes anchor constraints when possible and assert_permissions has been stripped to all but more dynamic and ATA checks.

3. `Initialize_borrowing_market` does not set `market_state` version

Issue: **HUBBLE-3**
Severity: **INFORMATIONAL**

Target: **handler_initialize_borrowing_market.rs**
Finding Type: **DYNAMIC**

initialize_borrowing_market initializes global state and borrowing market state. But there is no code to set the version.

Since the version is zero, it does not matter. But for better practice, there should be a line to set it.

Action Recommended:

Set version of `borrowing_market_state` in initialize handler

Remediation:

All schema versions are now set when accounts with a version field are initialised.

4. No need to `compute_new_stake in approve_trove`

Issue: **HUBBLE-4**
Severity: **INFORMATIONAL**

Target: **borrowing_operations.rs**
Finding Type: **DYNAMIC**

The `approve_trove()` function initializes `borrow_stablecoin` as zero, and then updates user stake and total staked amount, but since `borrow_stablecoin` is zero, there is nothing updated in `update_user_stake_and_total_stake` function.

Action Recommended:

Remove `redistribution::update_user_stake_and_total_stakes` in `approve_trove` function.

Remediation:

Hubble has removed the redundant calculation in the pull request [234]

5. No verification about pyth price account

Issue: **HUBBLE-5**
Severity: **CRITICAL**

Target: **handler_borrow_stablecoin.rs**
Finding Type: **DYNAMIC**

There is no enough verification to get price from pyth client.

At least, there must be a verification for the selected pyth is the correct account for selected token.

Action Recommended:

Add more verification of price before borrowing stablecoin.

Remediation:

- Price accounts are now set in the open, but verifiable endpoint - update_oracle_mapping
- Oracle mappings are now validated when passed using anchor constraints
- Confidence checking implemented when using the price -

6. Double calculation of collateral value

Issue: **HUBBLE-6**
Severity: **INFORMATIONAL**

Target: **finance.rs, liquidation_calc.rs**
Finding Type: **DYNAMIC**

In the \`calculate_collateral_value\` function, it calculates \`collateral_value\` and \`collateral_ratio\`.

But in \`collateral_ratio\` calculation (\`calc_coll_ratio\` function), there is calculation of \`collateral_value\` again.

This will use more gas because of double calculation.

And in the \`calc_system_mode\` function in \`liquidation_calc.rs\`, it only uses \`collateral_ratio\` from the returned value of \`calculate_collateral_value\`.

Action Recommended:

Optimize calculation of the \`collateral_value\`.

Remediation:

Functions now return tuples so values can be reused

7. Refresh base rate might not work for frequent borrowing

Issue: **HUBBLE-7**
Severity: **MEDIUM**

Target: **borrowing_rate.rs, stability_pool_operations.rs**
Finding Type: **DYNAMIC**

In the `decay_base_rate` function, `decay_factory` depends on different minutes from the last fee event.

If `borrow_stablecoin` is called every 40 seconds, the base rate cannot be updated because `decay_factor` is always set to 1, however, `last_fee_event` is successfully updated because its time is set in seconds.

Action Recommended:

Decay base rate based on different seconds, or find any other solution.

In `expected_issuance_since_start` function in `stability_pool_operations` has the same issue.

Remediation:

Client Response: No implementation needed for hbb emissions as the function is cumulative from the beginning of time, not incremental.

IMPLEMENTATION -

https://github.com/hubbleprotocol/hubble/blob/master/programs/borrowing/src/borrowing_market/borrowing_rate.rs#L58

TEST -

https://github.com/hubbleprotocol/hubble/blob/master/programs/borrowing/tests/tests_borrowing_operations.rs#L808

8. Incorrect argument name in `BorrowSplit::split_fees` function

Issue: **HUBBLE-8**
Severity: **INFORMATIONAL**

Target: **borrowing_rate.rs**
Finding Type: **DYNAMIC**

The `split_fees` function has two arguments, the second one of which is `borrowing_rate`.

The second argument's actual behaviour is to calculate the fee, thus, it can be confusing to understand the intent of the function's needed argument.

Action Recommended:

Consider a more closely intent-aligned second argument name in `split_fees` function.

Remediation:

Hubble has renamed the parameter to borrowing_fees_rate

9. Incorrect argument name in `BorrowSplit::split_fees` function

Issue: **HUBBLE-9**
Severity: **INFORMATIONAL**

Target: **borrowing_operations.rs**
Finding Type: **DYNAMIC**

In line 187, there is a condition to update user status to active after borrowing.

In this condition, new_debt > 0 will always be true, because there is a check for a 0 at the top, and also new debt must be greater than `BORROW_MIN` value.

Action Recommended:

Remove new_debt > 0 condition.

Remediation:

Client Response: Hubble has removed the redundant condition

10. Optimize calculation of `amount_mint_to_user`

Issue: **HUBBLE-10**
Severity: **INFORMATIONAL**

Target: **borrowing_operations.rs**
Finding Type: **DYNAMIC**

`amount_mint_to_user` is the same value as `requested_amount_borrow`, but in line 193, it returns value by calculating `borrow_and_fee.amount_to_borrow - borrow_and_fee.fees_to_pay`.

Action Recommended:

Change Line 193 to `amount_mint_to_user: requested_amount_borrow`

Remediation:

11. Optimize stablecoin mint operation when borrowing

Issue: **HUBBLE-11**
Severity: **INFORMATIONAL**

Target: **handler_borrow_stablecoin.rs**
Finding Type: **DYNAMIC**

In `handler_borrow_stablecoin::process` function, there are 3 calls of `stablecoin::mint` to mint, stablecoin to user, borrowing fee vault, and treasury.

In `stablecoin::mint` function, there are steps to get seeds, pda, and other data to invoke token::mint_to

So, at every one of the `stablecoin::mint` calls, it will call all required steps in duplicate, yet only `token::mint_to` is different; This will use more gas.

To optimize, it is possible to make a function to mint stablecoin for several recipients.

Action Recommended:

Optimize minting stable coin when borrowing

Remediation:

Added mint_many function - https://github.com/hubbleprotocol/hubble/blob/master/programs/borrowing/src/token_operations/stablecoin.rs#L7

12. Not sure if `deposit_collateral_and_borrow_stablecoin` is required

Issue: **HUBBLE-12**	Target: **lib.rs**
Severity: **INFORMATIONAL**	Finding Type: **DYNAMIC**

It looks like the `deposit_collateral_and_borrow_stablecoin` instruction is used to handle both depositing and borrowing collateral.

But there are `deposit_collateral` and `borrow_stablecoin` instructions already, and Solana allows sending multiple instructions in one instruction.

Action Recommended:

Consider if the `deposit_collateral_and_borrow_stablecoin` instruction is required.

Remediation:

NO ACTION - this is an atomic version of the methods and is required for borrowing when in recovery mode, as depositing collateral can further decrease the system collateral ratio we will not allow it.

13. Optimize when changing user status

Issue: **HUBBLE-13**	Target: **borrowing_operations.rs**
Severity: **INFORMATIONAL**	

Finding Type: **DYNAMIC**

When `repay_loan`, if the borrowed stable coin is zero, it changes user status to Inactive.

If user status was active, `user.inactive_collateral` is zero, so no need to add `user.deposited_collateral` to `user.inactive_collateral`.

In the `borrow_stablecoin` function, it is also a similar case.

Action Recommended:

Change Line 171 to `user.deposited_collateral = user.inactive_collateral` and change Line 239 to `user.inactive_collateral = user.deposited_collateral`

Remediation:

Planned for a future version, github issue #88.

14. RepayLoanEffects returns same value

Issue: **HUBBLE-14**
Severity: **INFORMATIONAL**

Target: **borrowing_operations.rs**
Finding Type: **DYNAMIC**

RepayLoanEffects is only calculated in the `repay_loan` function in `borrowing_operations.rs` and its `amount_to_burn` and `amount_to_transfer` are the same value.

Action Recommended:

Consider variables in RepayLoanEffects and if they will always be the same, remove one variable.

Remediation:

Planned for a future version, github issue #89

15. Optimize `withdraw_collateral` function depending upon user status

Issue: **HUBBLE-15**
Severity: **INFORMATIONAL**

Target: **borrowing_operations.rs**
Finding Type: **DYNAMIC**

Users can hold either `inactive_collateral` or `deposited_collateral`.

So `requested_amount - user_inactive_token` does not have any meaning.

Action Recommended:

Check user status, and if active, then withdraw from `deposited_collateral`, and call `liquidation:calcs::try_withdraw`, and update `market.deposited_collateral`, and `user.deposited_collateral`.

If user status is inactive, then withdraw from `inactive_collateral`, and update `market.inactive_collateral` and `user.inactive_collateral`.

Remediation:

Planned for a future version, github issue #90.

16. Unrequired condition in `withdraw_collateral` function

Issue: **HUBBLE-16**
Severity: **INFORMATIONAL**

Target: **borrowing_operations.rs**
Finding Type: **DYNAMIC**

If user status is inactive, `user.borrow_stablecoin` is zero.

If user status is active, and there is borrowed stablecoin, the user cannot withdraw all collateral.

So no need to check if `user.borrow_stablecoin` == 0 to close user metadata.

Action Recommended:

Remove line 300.

Remediation:

Planned for future version, github issue #91

Conclusion

Arcadia identified issues that occurred at hash **#6e0f7d2b63b02f46eefd0b4f7de024e1f2412c2b**

Some of the relevant issues were resolved in the following pull requests: #234, some issues were resolved in unreviewed commits and some of which are scheduled to be added in later versions

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, The Arcadia Group and the Authors assume no responsibility for errors, omissions, or damages resulting from the use of the provided information. Additionally, Arcadia would like to emphasize that the use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period.