Capapult
Money Market Contracts & Private Token
Audit Retainer Report

Prepared for Capapult, 8th May 2023

Table of Contents

Introduction

SCV was engaged by Capapult to assist in identifying security threats and vulnerabilities that have the potential to affect their security posture. Additionally, SCV will assist the team in understanding the risks and identifying potential mitigations.

Scope

SCV performed the security assessment on the following codebase:

- https://github.com/capapult-finance/capa-money-market/tree/development
  - Code Freeze: *ed7a9e2188c97af2403a568e946e66d9318ab0a6*
- https://github.com/capapult-finance/old-money-market-contracts/compare/market-development...fixed-interest
  - Code Freeze: *cbb8583bc451cd1a44e77ff987f6f490162add25*
- https://github.com/capapult-finance/private-capa-token/tree/spino-rew-traditional
  - Code Freeze: *6f5303ce89052b2d32ba820741b39127d10b412a*

Remediations were applied by the team and verified by SCV on the following:

- https://github.com/capapult-finance/capa-money-market/tree/development
  - Code Freeze: *92005833ab50916f49eef49bb2be70a43f3c4e3b*
- https://github.com/capapult-finance/capa-money-market
  - Code Freeze: `92005833ab50916f49eef49bb2be70a43f3c4e3b`
- https://github.com/capapult-finance/private-capa-token/tree/spino-rew-traditional
  - Code Freeze: `a258b7c104598af7cf69348f7497d8e792b2a43f`

Note that this security assessment was conducted as part of an ongoing retainer agreement with the client that began with a previous audit engagement. The previous engagement covered *Capapult - Money Market Contracts Updates - Audit Report v1.0*, and the current engagement focused on the codebases listed above.

Methodologies

SCV performs a combination of automated and manual security testing based on the scope of testing. The testing performed is based on the extensive experience and knowledge of the auditor to provide the greatest coverage and value to Capapult. Testing includes, but is not limited to, the following:

- Understanding the application and its code base purpose;
- Deploying SCV in-house tooling to automate dependency analysis and static code review;
- Analyse each line of the code base and inspect application security perimeter;
- Review underlying infrastructure technologies and supply chain security posture;

Code Criteria and Test Coverage

This section below represents how *SUFFICIENT* or *NOT SUFFICIENT* each code criteria was during the assessment

Criteria	Status	Notes
Provided Documentation	**SUFFICIENT**	Documentation is available at https://www.capapult.finance/whitepaper-v2.pdf and https://medium.com/@capapult
Code Coverage Test	**SUFFICIENT**	N/A
Code Readability	**SUFFICIENT**	The codebase had good readability and utilized many Rust and CosmWasm best practices.
Code Complexity	**SUFFICIENT**	N/A

Vulnerabilities Summary

#	Summary Title	Risk Impact	Status
1	Solid contract balance is calculated incorrectly	**Critical**	**Resolved**
2	Updating denom per block does not accrue previous rewards	**Critical**	**Resolved**
3	Incorrect `MAX_BLOCKS_PER_YEAR` constant	**Medium**	**Acknowledged**
4	Potential division by zero error when computing fees	**Low**	**Acknowledged**
5	Initializing base borrow and flash mint fee is not enforced to be lower than 100%	**Low**	**Acknowledged**
6	One-time borrow fees cannot be estimated	**Low**	**Partially Resolved**
7	Incorrect denom comments in `custody_lunax` contract	**Informational**	**Acknowledged**
8	Borrowers cannot query their original loan amount	**Informational**	**Acknowledged**
9	Total bond and supply store the same value	**Informational**	**Acknowledged**
10	Incorrect error when unbonding zero amount	**Informational**	**Acknowledged**
11	burn and redeem amount emits the same value	**Informational**	**Acknowledged**
12	Outstanding TODO comments in the codebase	**Informational**	**Acknowledged**
13	Improve integration testing with `cw-multi-test` adoption	**Informational**	**Acknowledged**

Audit Observations

- The comprehensive testing of the interaction with the governance contract was not performed due to the constraints of this audit retainer.

Detailed Vulnerabilities

1 – Solid contract balance is calculated incorrectly

Risk Impact: Critical - **Status**: Resolved

Description

To fund `SOLID` in the contract, the governance contract will send a `Cw20HookMsg::FundReserve` message to call the `fund_solid` function. However, the `solid_balance` incorrectly has the sent `amount` added in `contracts/staking-reward/src/contract.rs:175`, causing the queried balance to be larger than intended.

As a result, the `state.solid_per_block` calculation will be incorrect, rendering more rewards to be distributed to stakers.

Recommendations

Consider removing the sent amount when querying the contract balance.

2 – Updating denom per block does not accrue previous rewards

Risk Impact: Critical - **Status**: Resolved

Description

When updating `capa_per_block` or `solid_per_block`, old rewards are not accrued immediately. This is problematic because accrued rewards are determined as `passed_blocks` multiplied by `state.solid_per_block` or `state.capa_per_block`, causing the computed global reward index to be incorrect if an update was performed during the middle of the staking process.

The following code lines are affected:

- `contracts/staking-reward/src/contract.rs:105`
- `contracts/staking-reward/src/contract.rs:149`
- `contracts/staking-reward/src/contract.rs:177`

Recommendations

Consider accruing the previous rewards before updating `capa_per_block` or `solid_per_block`.

3 – Incorrect `MAX_BLOCKS_PER_YEAR` constant

Risk Impact: Medium - **Status**: Acknowledged

Description

The `MAX_BLOCKS_PER_YEAR` constant in `contracts/staking-reward/src/contract.rs:21` is hard-coded to 5195387. As one block time in Terra is 6 seconds, this would evaluate to 31172322 seconds, equivalent to 360 days, which is incorrect.

Recommendations

Consider modifying the value to 5256000, which is equivalent to 365 days.

Revision Notes

Capapult team suggests that block time fluctuates around 6 seconds and not necessarily need to pin at 5256000.

4 – Potential division by zero error when computing fees

Risk Impact: Low - **Status**: Acknowledged

Description

The `compute_borrow_fee` function in `contracts/market/src/borrow.rs:178` divides the value of `config.fee_increase_factor` if the price is equal to or lower than `$1`. If the contract owner configures the amount as zero, a division by zero error will occur when computing the borrow fees, preventing users from borrowing SOLID.

Recommendations

Consider handling this error gracefully by checking whether `config.fee_increase_factor` is zero.

5 – Initializing base borrow and flash mint fee is not enforced to be lower than 100%

Risk Impact: Low - **Status**: Acknowledged

Description

When instantiating the market contract in `contracts/market/src/contract.rs:43` and `45`, the `msg.base_borrow_fee` and `msg.flash_mint_fee` is not validated to be below 100%. This is inconsistent with the invariant when performing a configuration update, as seen in lines `275` to `277` and `285` to `287`.

Recommendations

Consider validating the `msg.base_borrow_fee` and `msg.flash_mint_fee` to be lower than 100% when instantiating the market contract.

6 – One–time borrow fees cannot be estimated

Risk Impact: Low - **Status**: Partially Resolved

Description

The `compute_borrow_fee` function in `contracts/market/src/borrow.rs:158` calculates the fees using the storage's `config.base_borrow_fee` and `config.fee_increase_factor` values. The current implementation does not expose or return the above values from queries. As a result, borrowers cannot estimate the one-time fee values from their intended borrowing amount.

Recommendations

Consider applying the following recommendations:

1. Emit the `one_time_borrow_fee` value in the `borrow_stable` function so users know their fee amount.
2. Include `config.fee_increase_factor` and `config.base_borrow_fee` values in the `query_config` function to allow users to query the current configuration values.
3. Expose a custom `QueryMsg::EstimateFee { borrow_amount }` query, which returns the estimated `one_time_borrow_fee` value computed from the `compute_borrow_fee` function.

7 – Incorrect denom comments in `custody_lunax` contract

Risk Impact: Informational - **Status**: Acknowledged

Description

The issue lies in `contracts/custody_lunax/src/collateral.rs:71`, in which the comment indicates `contract_balance_info.balance` as the contract's Luna balance. Since the new implementation will no longer sell LSD-generated proceeds, the Luna keyword in the comment should be modified into LunaX.

Recommendations

Consider modifying the comment into LunaX balance instead.

8 – Borrowers cannot query their original loan amount

Risk Impact: Informational - **Status**: Acknowledged

Description

The `query_borrower_info` function in `contracts/market/src/borrow.rs:186-194` does not include `borrower_info.loan_amount_without_interest` as part of the query. This prevents borrowers from querying their original borrowed amount that excludes fees.

Recommendations

Consider including `borrower_info.loan_amount_without_interest` in the query response.

9 – Total bond and supply store the same value

Risk Impact: Informational - **Status**: Acknowledged

Description

The `total_bond` and `total_supply` store the same value when bonding funds in `contracts/stride_wrapper/src/bond.rs:37-38` and unbonding funds in lines `68` to `69`. This is inefficient as having one variable to hold the value is enough.

Note that the native wrapper contract is also affected, as seen in `contracts/native_wrapper/src/bond.rs:68-69`.

Recommendations

Consider removing either `total_bond` or `total_supply` and use only one instead.

10 – Incorrect error when unbonding zero amount

Risk Impact: Informational - **Status**: Acknowledged

Description

A `ZeroRepay` contract error will happen in `contracts/stride_wrapper/src/bond.rs:65` if the unbond amount is zero. This is incorrect, as the error message mentions the repayment amount, not the unbond amount.

Note that the native wrapper contract is also affected, as seen in `contracts/native_wrapper/src/bond.rs:65`.

Recommendations

Consider adding a new error message of “Unbond amount” instead.

11 – burn and redeem amount emits the same value

Risk Impact: Informational - **Status**: Acknowledged

Description

The `unbound` function in `contracts/native_wrapper/src/bond.rs:90` and `92` implements different attribute names that emit the same value, specifically `burn_amount` and `redeem_amount`. This is inefficient as only one attribute is required to emit the value.

Recommendations

Consider only using one of the attributes instead.

12 – Outstanding TODO comments in the codebase

Risk Impact: Informational - **Status**: Acknowledged

Description

The following code lines contain outstanding TODO comments:

- `contracts/stride_wrapper/src/contract.rs:59`
- `contracts/stride_wrapper/src/contract.rs:215`
- `contracts/native_wrapper/src/contract.rs:55`

Recommendations

Consider resolving the TODO comments.

13 – Improve integration testing with `cw-multi-test` adoption

Risk Impact: Informational - **Status**: Acknowledged

Description

The current testing approach in the smart contract relies on the native CosmWasm testing framework. While this is effective for basic testing, it is recommended to adopt the `cw-multi-test` library for a more comprehensive integration testing process.

As the existing testing framework may not provide a sufficiently realistic simulation environment for contract-to-contract and contract-to-SDK interactions, it could lead to potential issues being overlooked during testing.

Recommendations

Consider adopting the `cw-multi-test` library to enhance the integration testing of the smart contract.

Document control

Version	Date	Approved by	Changes
0.1	26/12/2022	Vinicius Marino	Document Pre-Release
0.2	11/01/2023	SCV Team	Remediation Revisions
1.0	12/01/2023	Vinicius Marino	Document Release (1.0)
1.1	17/04/2023	SCV Team	MM Audit Retainer
1.2	23/04/2023	SCV Team	private-capa-token Audit Retainer
1.5	24/04/2023	Vinicius Marino	Report Release
1.6	03/05/2023	SCV Team	Remediation Revisions
2.0	08/05/2023	Vinicius Marino	Document Release

Appendices

A. Appendix – Risk assessment methodology

A qualitative risk assessment is performed on each vulnerability to determine the impact and likelihood of each.

Risk rate will be calculated on a scale. As per criteria Likelihood vs Impact table below:

	Rare	Unlikely	Possible	Likely
Critical	Medium	Severe	Critical	Critical
Severe	Low	Medium	Severe	Severe
Moderate	Low	Medium	Medium	Severe
Low	Low	Low	Low	Medium
Informational	Informational	Informational	Informational	Informational

LIKELIHOOD

- Likely: likely a security incident will occur;
- Possible: It is possible a security incident can occur;
- Unlikely: Low probability a security incident will occur;
- Rare: In rare situations, a security incident can occur;

IMPACT

- Critical: May cause a significant and critical impact;
- Severe: May cause a severe impact;
- Moderate: May cause a moderated impact;
- Low: May cause low or none impact;
- Informational: May cause very low impact or none.

B. Appendix – Report Disclaimer

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts SCV-Security to perform a security review. The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The content of this audit report is provided “as is”, without representations and warranties of any kind, and SCV-Security disclaims any liability for damage arising out of, or in connection with, this audit report.