Capapult
Money Market Contracts
Audit Report

Prepared for Capapult, 12th January 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/money-market-contracts
- Code Freeze: `34ea7d99982a35f84fecc2c7c51b5eb6f7e72e10`

Remediations were applied into several commits up to the following hash:

- Code Freeze: `77af89a5e303bdf48bbed6ca37085088d4f94f03`

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**	N/A
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	Liquidators can drain funds from contract	**Critical**	**Resolved**
2	Liquidate collateral incorrectly decreases contract balance	**Critical**	**Resolved**
3	`RegisterContracts` can be executed by anyone	**Severe**	**Resolved**
4	Executing epoch operations will only call `DistributeRewards` to the first ten collaterals	**Severe**	**Resolved**
5	Overseer `State` query message will always fail	**Medium**	**Resolved**
6	Oracle time constraints are not enforced in market contract	**Medium**	**Resolved**
7	Max slippage is not validated to be lower than the `MAX_ALLOWED_SLIPPAGE`	**Low**	**Resolved**
8	Collateral maximum LTV ratio is not validated	**Low**	**Resolved**
9	Maximum fees would prevent successful liquidations	**Low**	**Resolved**
10	Configuration update does not emit attributes	**Informational**	**Resolved**
11	General inefficiencies and informational issues in the codebase	**Informational**	**Resolved**
12	Contracts should use two-step ownership transfer	**Informational**	**Acknowledged**
13	Use of `CanonicalAddr` is deprecated	**Informational**	**Resolved**
14	`ExecuteEpochOperations` never updates `last_executed_height`	**Informational**	**Resolved**
15	`register_feeder` overwrites existing feeder contract for a specified asset if it already exists	**Informational**	**Resolved**
16	Remove duplicate condition	**Informational**	**Resolved**

Audit observations

- The money market contract relies on StaderLab's staking contract. The implementation between them is out-of-scope of the audit.

Detailed Vulnerabilities

1 – Liquidators can drain funds from contract

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

Description

The `ActivateBids` and `ClaimLiquidations` messages in `contracts/liquidation_queue/src/bid.rs:99` and `contracts/liquidation_queue/src/bid.rs:422` collect user's `bids_idx` input into a vector of `Bid` struct without deduping them.

The former would inflate the `total_activated_amount` and `available_bids` amount in lines `149` to `150`. This causes the emitted `total_activated_amount` amount to be inflated in line `157` and causes the collateral token to have more bids than intended in line `153`.

The latter would inflate the `bid_pool.residue_collateral`, `bid_pool.residue_bid`, and `claim_amount` value in lines `455`, `456`, and `459`. All three variables would cause a loss of funds.

Please see the test case in the following link to reproduce the funds draining issue.

Recommendations

Consider deduping the `bids_idx` vector provided by the user. An example can be found here.

2 – Liquidate collateral incorrectly decreases contract balance

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

Description

When liquidating collateral in the overseer contract in `contracts/custody_lunax/src/collateral.rs:218`, the contract's balance deducts the total borrower's balance instead of the amount to liquidate. This causes the contract to deduct more balance than expected. As a result, this causes an issue where users cannot withdraw their collateral due to insufficient balance in line 87.

Recommendations

Consider deducting the amount to liquidate instead of the borrower's total balance.

3 – `RegisterContracts` can be executed by anyone

Risk Impact: Severe - **Status**: Resolved

Description

The `register_contracts` function in `contracts/market/src/contract.rs:214` allows any caller to set the values for `overseer_contract`, `interest_model`, `collector_contract`, `liquidation_contract`, and `oracle_contract`.

After the contract is instantiated, the owner is expected to execute `RegisterContracts` to set up the contracts. This function can only be executed once due to the if statement in line `230`.

Suppose an attacker calls this function before the contract administrator. In that case, they can perform a range of actions, such as setting the addresses to malicious contracts, providing invalid addresses that will cause errors, or even passing some as empty parameters, which will cause the validation on lines `224-229` to fail.

Recommendations

Consider adding an authentication check to ensure only the contract owner can execute the message.

4 – Executing epoch operations will only call `DistributeRewards` to the first ten collaterals

Risk Impact: Severe - **Status**: Resolved

Description

The `execute_epoch_operations` function in `contracts/overseer/src/contract.rs:266` performs epoch operations by first getting a list of the whitelisted collaterals using the `read_whitelist` function. Since the `start_after` and `limit` parameters are specified as `None`, the default limit of ten will be used as seen in `contracts/overseer/src/state.rs:99`.

This is problematic because if the owner registered more than ten collaterals, the overseer contract would not execute `DistributeRewards` message for the excess collaterals, causing the collector contract to receive no rewards.

Recommendations

Consider implementing a maximum limit when registering collaterals with regard to the limit retrieved by the `read_whitelist` function.

5 – Overseer `State` query message will always fail

Risk Impact: Medium - **Status**: Resolved

Description

In `contracts/market/src/contract.rs:399`, the `base` argument of the `query_price` function was hardcoded to "*solid*", which is incorrect because the correct value should be `config.stable_contract` instead. This is because the asset identifier for SOLID will likely be the contract address itself, as seen in `contracts/oracle/src/contract.rs:153` and `162`.

Recommendations

Consider modifying line `399` into `config.stable_contract`.

6 – Oracle time constraints are not enforced in market contract

Risk Impact: Medium - **Status**: Resolved

Description

In several instances of the market contract, the time constraints are not applied to the oracle queries. This could lead to out-of-price quotes, causing the contract to accept any outdated `SOLID/uusd` price response.

- `contracts/market/src/contract.rs:317,266`
- `contracts/market/src/borrow.rs:44,125`

Recommendations

Consider adding a fresh time frame to ensure the price retrieved is not stale.

7 – Max slippage is not validated to be lower than the `MAX_ALLOWED_SLIPPAGE`

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

Description

In `contracts/custody_lunax/src/contract.rs:45` and `contracts/custody_lunax/src/contract.rs:168`, the `max_slipage` value is not validated during contract instantiation and config update.

This is problematic because the value will be used in `contracts/custody_lunax/src/distribution.rs:112` where the max spread is supplied with configured max slippage amount. If the max slippage amount is configured higher than `50%`, the swap fails with the `AllowedSpreadAssertion` error.

Recommendations

Consider validating the value to be lower or equal to `50%`.

8 – Collateral maximum LTV ratio is not validated

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

Description

In `contracts/overseer/src/contract.rs:205` and `contracts/overseer/src/contract.rs:240`, the LTV is not validated to be higher than `0%` and lower than `100%`.

The former allows the user to take out an undercollateralized loan, while the latter will cause a division by zero error in `contracts/liquidation_queue/src/query.rs:165`, preventing a successful liquidation attempt.

Recommendations

Consider validating the LTV value to be higher than `0%` and lower than `100%`.

9 – Maximum fees would prevent successful liquidations

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

Description

In `contracts/liquidation_queue/src/bid.rs:361-375`, the total bid fee and liquidation fee can be specified into the max value of `100%` as seen in the `assert_fees` function in `contracts/liquidation_queue/src/asserts.rs:56`.

This will cause the repay amount to be zero when executing liquidation as seen in `contracts/liquidation_queue/src/bid.rs:363`. The SOLID contract will then try to send 0 funds in line `270`, which will fail because CW20 prevents it.

Recommendations

Consider only repaying stables if the amount is higher than zero. An example can be found here.

10 – Configuration update does not emit attributes

Risk Impact: Informational - **Status**: Resolved

Description

In `contracts/liquidation_queue/src/contract.rs:224`, no attributes are emitted when the contract owner updates the configuration. It is best practice to emit attributes to support event aggregation and block explorers.

Recommendations

We recommend emitting relevant attributes.

11 – General inefficiencies and informational issues in the codebase

Risk Impact: Informational - **Status**: Resolved

Description

In the following code lines, several inefficiencies and informational issues are discovered:

1. `contracts/market/src/contract.rs:74-77`

The `initial_balances` in line `74` to `77` tries to create an initial balance of zero for the current contract addresses. This is unneeded, as balances will be created when the balance increases. The current implementation will cost more gas as it needs to enter the validate the accounts created in a loop, increasing the code's complexity.

2. `contracts/oracle/src/contract.rs:111`

The price set by the feeder in line `111` should be validated as not zero. If zero is allowed, all queries that received the rate should immediately invalidate the price by reverting the transaction, as an asset with 0 prices is not theoretically possible.

3. `contracts/overseer/src/collateral.rs:189`

The filter in line `189` can be removed because the `Result` is returned as `Ok()` in line `176`.

4. `contracts/custody_lunax/src/contract.rs:25-56`

Lines `25` and `26` can be removed since it's not used in the reply handler.

5. `contracts/market/src/borrow.rs:291-295`

The `BorrowerInfoResponse` does not return the original loan amount from a borrower. Consider returning `loan_amount_without_interest` in the response.

6. `contracts/market/src/borrow.rs:230-231`

Line `230` and `231` comments should be removed since it's copied from Anchor and not used.

7. `contracts/liquidation_queue/src/contract.rs:128`

There's an extra `OR` statement that checks whether the fee address is empty. It can be removed as it's previously checked already.

Recommendations

Consider resolving the inefficiencies and informational issues above.

12 – Contracts should use two–step ownership transfer

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

Description

The current ownership transfer for each of the contracts is executed in one step, which imposes a risk that if the new owner is incorrect, then the admin privileges of the contract are effectively transferred and lost. A two-step ownership transfer is best practice because it requires the new admin to accept ownership before the transfer and config changes occur.

- `contracts/market/src/contract.rs:269`
- `contracts/custody_lunax/src/contract.rs:147`
- `contracts/oracle/src/contract.rs:57`
- `contracts/interest_model/src/contract.rs:71`
- `contracts/liquidation_queue/src/contract.rs:185`
- `contracts/overseer/src/contract.rs:155`

Recommendations

Consider implementing a two-step ownership transfer where the current owner proposes a new owner address, and then that new owner address must call the contract to accept ownership within a finite time frame. SCV suggests the following implementation https://docs.rs/cw-controllers/latest/cw_controllers/index.html.

13 – Use of `CanonicalAddr` is deprecated

Risk Impact: Informational - **Status**: Resolved

Description

In several instances of the codebase, `CanonicalAddr` is used within the contracts in the scope of this audit. However, `CanonicalAddr` is deprecated.

Recommendations

Consider removing instances of `CanonicalAddr` within the codebase.

14 – `ExecuteEpochOperations` never updates `last_executed_height`

Risk Impact: Informational - **Status**: Resolved

Description

In `contracts/overseer/src/contract.rs:256`, the `execute_epoch_operations` function does not set the `state.last_executed_height` to the latest block height when it gets executed. Since `store_epoch_state` is only called once during contract instantiation, the `last_executed_height` will always reflect the outdated height, while the deposit rate will always reflect zero.

This issue is flagged as informational severity as the epoch operations performed on the market contract will update the market contract's internal state `last_interest_updated` value.

Recommendations

Consider setting the `state.last_executed_height` to the current height after executing the epoch operations.

15 – `register_feeder` overwrites existing feeder contract for a specified asset if it already exists

Risk Impact: Informational - **Status**: Resolved

Description

The `register_feeder` function in `contracts/oracle/src/contract.rs:65` allows the contract's owner to register a feeder for a specified asset. This then calls the `store_feeder` function which saves the specified feeder address for the specified asset. This function does not account for a situation where an asset already has a specified feeder address. In that scenario the old address will simply be overwritten. While this is only callable by the admin, it is best practice to create a proper entrypoint that reflects the state change that is occurring and that will emit the proper attributes.

Recommendations

We recommend creating a separate update feeder entrypoint.

16 – Remove duplicate condition

Risk Impact: Informational - **Status**: Resolved

Description

The `ExecuteBid` message type in `contracts/liquidation_queue/src/contract.rs:128` contains a duplicate condition that should be removed. The condition is checking twice if `fee_address` is empty.

Recommendations

We recommend removing the duplicate condition.

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

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.