Cavern Protocol
Core Contracts
Audit Report

Prepared for Cavern Protocol, Report – 3rd April 2023

Table of Contents

Table of Contents 2
Introduction 4
Scope 4
Methodologies 5
Code Criteria and Test Coverage 5
Threat Modeling 6
Vulnerabilities Summary 7
Detailed Vulnerabilities 9
1 - Swapping will fail due to calling ExecuteSwapOperation message 9
2 - Fund reserve will never record sent funds 10
3 - Liquidations will fail when one of the collaterals has no bids 11
4 - Specifying minimum receive as None allows sandwiching previous trades 12
5 - RegisterContracts can be executed by anyone 13
6 - Executing epoch operations will only call DistributeRewards to the first ten collaterals 14
7 - Interest buffer is not reduced when reserve bucket balance is equal or higher to max borrow reserves 15
8 - Redelegating to unregistered validators causes the validator to be left out 16
9 - Modifying the unbonding period affects previous epochs' withdrawal rate 17
10 - Marketplace index will be incorrect if the query fails 18
11 - Providing initial balances when instantiating the bAsset contract would be ineffective 19
12 - Adding too many validators would cause query_validators to fail 20
13 - Feeding price as zero causes potential division by zero error 21
14 - Removing collateral prevents liquidation attempts 22
15 - Incorrect blocks per year constant 23
16 - Peg recovery fee can be updated to be higher than 100% 24
17 - Config query message does not return validators_registry_contract 25
18 - execute_withdraw_unbonded function emits incorrect from attribute key 26
19 - Validator address is not validated upon delegation 27
20 - Stable token’s CW20 address is not validated 28
21 - Unvalidated decimal values 29
22 - Updating fees might fail due to incorrect fee assertion 30
23 - query_next_borrower_incentives fails when no borrows 31
24 - Misconfiguring known_cw20_tokens might cause swaps to fail 32
25 - Contracts should use two-step ownership transfer 33
26 - Configuration update does not emit attributes 34

27 - Validating info.sender consumes unnecessary gas costs 35
28 - Misleading query_accrued_rewards function 36
29 - Decompound emits incorrect action attribute 37
30 - Usage of canonical and non-canonical address 38
31 - Deprecated deduct tax functionality 39
32 - Unimplemented ClaimRewards message 40
33 - register_feeder overwrites existing feeder contract for a specified asset if it already exists 41
Document control 42
Appendices 43

Introduction

SCV was engaged by Cavern Protocol 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/CavernPerson/cavern-aAsset-contracts
  - Hash: `bdf1b81cb37fe9f9f0476b004faad4a9b5939676`
- https://github.com/CavernPerson/cavern-lsd-wrapper
  - Hash: `26d3522ec0f47f767fc54872373fe2b63cf0e11b`
- https://github.com/CavernPerson/money-market-contracts
  - Hash: `efb8492fe6b4926f8b0bed042e71709d564322c9`

SCV performed revisions on remediations applied by Cavern team on the following:

- https://github.com/CavernPerson/cavern-aAsset-contracts
  - Hash: `328701fe6a007a9706fb9f11ac9052b8a8a5b8a6`
- https://github.com/CavernPerson/cavern-lsd-wrapper
  - Hash: `3c75c90435f37a4ad04fa49a95229920ddd9a9f8`
- https://github.com/CavernPerson/money-market-contracts
  - Hash: `92dc1a12058eb37263792cb1799b35ac655ed06b`

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 Cavern Protocol. 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.
- Analyze 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 were available at the following url: https://docs.cavernprotocol.com

Code Coverage Test	**SUFFICIENT**	`cavern-aAsset-contracts: 93.99%`
		`cavern-lsd-wrapper: 68.03%`
		`money-market-contracts: 92.88%`

Code Readability	**SUFFICIENT**	The codebase had good readability and utilised many Rust and CosmWasm best practices.
Code Complexity	**SUFFICIENT**	N/A

Threat Modeling

The goal of threat modeling is to identify and evaluate potential threats to a system or application and to develop strategies to mitigate or manage those threats. Threat modeling is an important part of the software development life cycle, as it helps developers and security professionals to proactively identify and address security risks before they can be exploited by attackers.

The main objectives of threat modeling includes (not limited to) the following :

- **Identify threats**: The first objective of threat modeling is to identify potential threats that could affect the security posture of the underlying smart contracts or application. This can include threats from external attackers, internal actors, or even accidental events that could happen.
- **Evaluate risks:** Once potential threats have been identified, the next objective is to evaluate the risks associated with each threat. This involves assessing the likelihood of each threat occurring and the potential impact it could have overall.
- **Mitigation strategies:** After identifying potential threats and evaluating the associated risks, the next objective is to develop strategies to mitigate or reduce the impact of threats. This can include implementing technical controls, such as access controls or further security measures around developing policies and procedures to reduce the likelihood or impact of a threat.
- **Communicate findings:** The final objective of threat modeling is to communicate the findings and recommendations to relevant stakeholders, such as developers, security teams, and management. This helps ensure that everyone involved in the development and maintenance understands the potential risks and the best strategies for addressing them.

Vulnerabilities Summary

#	Summary Title	Risk Impact	Status
1	Swapping will fail due to calling `ExecuteSwapOperation` message	**Critical**	**Resolved**
2	Fund reserve will never record sent funds	**Critical**	**Resolved**
3	Liquidations will fail when one of the collaterals has no bids	**Critical**	**Resolved**
4	Specifying minimum receive as `None` allows sandwiching previous trades	**Severe**	**Acknowledged**
5	`RegisterContracts` can be executed by anyone	**Severe**	**Resolved**
6	Executing epoch operations will only call `DistributeRewards` to the first ten collaterals	**Severe**	**Resolved**
7	Interest buffer is not reduced when reserve bucket balance is equal or higher to max borrow reserves	**Severe**	**Resolved**
8	Redelegating to unregistered validators causes the validator to be left out	**Medium**	**Resolved**
9	Modifying the unbonding period affects previous epochs' withdrawal rate	**Medium**	**Resolved**
10	Marketplace index will be incorrect if the query fails	**Medium**	**Resolved**
11	Providing initial balances when instantiating the `bAsset` contract would be ineffective	**Medium**	**Acknowledged**
12	Adding too many validators would cause `query_validators` to fail	**Medium**	**Resolved**
13	Feeding price as zero causes potential division by zero error	**Medium**	**Resolved**
14	Removing collateral prevents liquidation attempts	**Medium**	**Resolved**
15	Incorrect blocks per year constant	**Medium**	**Resolved**
16	Peg recovery fee can be updated to be higher than	**Low**	**Resolved**

	100%		
17	`Config` query message does not return `validators_registry_contract`	**Low**	**Resolved**
18	`execute_withdraw_unbonded` function emits incorrect from attribute key	**Low**	**Resolved**
19	Validator address is not validated upon delegation	**Low**	**Resolved**
20	Stable token's CW20 address is not validated	**Low**	**Resolved**
21	Unvalidated decimal values	**Low**	**Resolved**
22	Updating fees might fail due to incorrect fee assertion	**Low**	**Resolved**
23	`query_next_borrower_incentives` fails when no borrows	**Low**	**Resolved**
24	Misconfiguring `known_cw20_tokens` might cause swaps to fail	**Low**	**Resolved**
25	Contracts should use two-step ownership transfer	**Low**	**Acknowledged**
26	Configuration update does not emit attributes	**Informational**	**Resolved**
27	Validating `info.sender` consumes unnecessary gas costs	**Informational**	**Resolved**
28	Misleading `query_accrued_rewards` function	**Informational**	**Acknowledged**
29	Decompound emits incorrect `action` attribute	**Informational**	**Resolved**
30	Usage of canonical and non-canonical address	**Informational**	**Resolved**
31	Deprecated deduct tax functionality	**Informational**	**Resolved**
32	Unimplemented `ClaimRewards` message	**Informational**	**Resolved**
33	`register_feeder` overwrites existing feeder contract for a specified asset if it already exists	**Informational**	**Resolved**

Detailed Vulnerabilities

1 – Swapping will fail due to calling `ExecuteSwapOperation` message

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

Affected code lines

- `cavern-money-market-contracts/contracts/custody_bluna/src/swap.rs:117-125`
- `cavern-money-market-contracts/contracts/custody_bluna/src/swap.rs:131-140`
- `cavern-money-market-contracts/contracts/custody_bluna/src/swap.rs:144-153`

Description

When creating a swap message, `AstroportExecuteMsg::ExecuteSwapOperation` is used instead of `AstroportExecuteMsg::ExecuteSwapOperations`. As `ExecuteSwapOperation` is only reserved for internal operations, the execution will revert due to an unauthorized error.

Recommendations

Consider using `ExecuteSwapOperations` instead.

2 – Fund reserve will never record sent funds

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

Affected code lines

- `cavern-money-market-contracts/contracts/overseer/src/contract.rs:721-724`

Description

The `fund_reserve` function determines the stable amount sent by checking the denom name to be "*uusd*". As Cavern protocol uses axlUSDC as the stable denom, the `prev_interest_buffer` and `prev_yield_reserve` will never increase despite the caller including funds when calling the `FundReserve` message.

Recommendations

Consider modifying the denom to check for `config.stable_denom` instead of "*uusd*".

3 – Liquidations will fail when one of the collaterals has no bids

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

Description

When performing liquidations, the `query_liquidation_amount` function in `cavern-money-market-contracts/contracts/liquidation_queue/src/query.rs:38` will be executed from the overseer contract to determine the number of collaterals that must be liquidated. If more than one collateral is provided, the amount to liquidate will be distributed evenly depending on the weight.

However, the liquidation will fail when one of the collateral has no bids in the pool. This might happen when all of the bids are consumed. In this case, the query will return the collateral's amount as zero in `cavern-money-market-contracts/contracts/overseer/src/collateral.rs:156`.

As a result, the `CustodyExecuteMsg::LiquidateCollateral` message will be executed with zero amount in line `189`, causing the whole execution to fail in `cavern-money-market-contracts/contracts/custody_base/src/collateral.rs:191` since `CW20` tokens do not allow zero-amount transfers.

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

Recommendations

Consider only executing `LiquidateCollateral` if `collateral.1` is higher than zero in `cavern-money-market-contracts/contracts/overseer/src/collateral.rs:180`.

4 – Specifying minimum receive as `None` allows sandwiching previous trades

Risk Impact: Severe - **Status**: Acknowledged

Affected code lines

- `cavern-aAsset-contracts/contracts/anchor_basset_reward/src/swap.rs:131`
- `cavern-lsd-wrapper/contracts/cavern_aAsset_lsd_reward/src/swap.rs:142`
- `cavern-lsd-wrapper/contracts/cavern_aAsset_lsd_reward/src/swap.rs:153`

Description

When performing `ExecuteSwapOperations` in decentralized exchanges, the `minimum_receive` is specified as `None`. In the Astroport router contract, spread assertions are not enforced for previous trades, and only the last trade will be validated against the provided spread assertion.

An attacker can perform a sandwich attack if the trade is performed within multiple hops. As a result, the sandwiched trade will succeed since there is no minimum received validation.

See the router's contract `router_does_not_enforce_spread_assertion` test case in Astroport for more information.

Recommendations

Consider setting the minimum receive amount as the simulated swap amount based on the best price result.

Revision Notes

The Cavern team acknowledged this issue as critical. However, the team states that the operation cannot fail, as doing so it would prevent the protocol from operating correctly.

5 – `RegisterContracts` can be executed by anyone

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

Affected code lines

- `cavern-money-market-contracts/contracts/market/src/contract.rs 274`

Description

The `register_contracts` function 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 291.

Suppose an attacker calls this function before the contract administrator. In that case, they can perform various actions, such as setting the addresses to malicious contracts, providing invalid addresses that will cause errors, or even passing some as empty parameters.

Recommendations

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

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

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

Description

The `execute_epoch_operations` function in `cavern-money-market-contracts/contracts/overseer/src/contract.rs:599` 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 `cavern-money-market-contracts/contracts/overseer/src/state.rs:156`.

This is problematic because if the owner registered more than ten collaterals, the overseer contract would not execute the `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.

7 – Interest buffer is not reduced when reserve bucket balance is equal or higher to max borrow reserves

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

Description

In `cavern-money-market-contracts/contracts/overseer/src/contract.rs:562`, the `interest_buffer` that represents the contract balance is deducted by `borrow_incentives_amount` to correctly distribute funds and update the storage's previous interest buffer.

Suppose a scenario where the `borrow_incentives_amount` is higher than zero, but `reserve_bucket_balance` is higher than `max_borrow_reserves`. The second condition in line `553` will not be fulfilled, causing the interest buffer not to deduct the borrow incentives amount.

This is problematic because the previous interest amount will be incorrectly recorded without deducting the borrow incentives amount, as seen in line `614`.

Additionally, the `distribution_buffer` calculation in line `579` will be incorrect since it uses an outdated value.

Recommendations

Consider deducting the borrow incentives amount from the interest buffer regardless of the `reserve_bucket_balance` and `max_borrow_reserves` values.

8 – Redelegating to unregistered validators causes the validator to be left out

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

Description

The `execute_redelegate_proxy` function in `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/contract.rs:133` can be executed manually by the contract creator to redelegate funds to another validator.

No validation ensures the validator addresses to redelegate to are registered in the validators registry contract. This is problematic because delegations are distributed based on registered validators only, causing the unregistered validator to be left out and not receive future delegations.

Specifically, the `query_validators` function in `cavern-aAsset-contracts/contracts/lido_terra_validators_registry/src/contract.rs:240-242` only includes the validator if stored in the `REGISTRY` storage state. The delegated amount will be omitted if the redelegated validator is not recorded.

This forces the contract owner to manually call the `AddValidator` message so the validator can receive equal distributions with other registered validators.

Recommendations

Consider adding validations to ensure the destination validator address is registered in the validator registry contract.

9 – Modifying the unbonding period affects previous epochs' withdrawal rate

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

Description

In line `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/unbond.rs:155`, the historical time is determined from the current time minus the configurable unbonding period. The value is then fed into the `process_withdraw_rate` function to determine whether the unbonding epoch has been processed.

Since the withdrawal rate is determined based on the current and previous contract balance in line `295`, modifying the unbonding period would cause the withdrawal rate to be incorrectly calculated.

For example, increasing the unbonding period would cause previous undelegations to get incorrectly processed because the historical time will be higher. The withdrawal rate would be calculated to a low value as the funds are still undergoing unbonding, causing a loss of funds to the users. Once the funds are unbonded, the next epoch's withdrawal rate will be inflated.

Recommendations

Consider removing the functionality to update the unbonding period.

10 – Marketplace index will be incorrect if the query fails

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

Affected code lines

- `cavern-aAsset-contracts/contracts/anchor_basset_reward/src/swap.rs:50-51`
- `cavern-lsd-wrapper/contracts/cavern_aAsset_lsd_reward/src/swap.rs:46`
- `cavern-money-market-contracts/contracts/custody_bluna/src/swap.rs:36`
- `cavern-money-market-contracts/contracts/custody_lsd/src/swap.rs:44`

Description

The swap message calculates the best price among all DEXs before performing the swap. This is done by performing a `SimulateSwapOperations` query message to all DEXs and retrieving the highest return value.

If the query fails, the iteration will ignore it, as seen in line `36`. This could cause the incorrect marketplace to be retrieved, causing a potentially unprofitable or fail swap.

Suppose a scenario where the query from Phoenix failed for some reason, and TerraSwap returns the second most profitable swap. Ideally, the swap should be executed in TerraSwap.

Since the `best_price_result` ignores errors from the swap result, the tuple will record the second index as the best marketplace. However, the second index will be seen incorrectly as Phoenix marketplace in line `51`, causing the swap to be performed on the incorrect marketplace.

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

Recommendations

Consider handling the best price index correctly based on the marketplace.

11 – Providing initial balances when instantiating the `bAsset` contract would be ineffective

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

Description

In `cavern-aAsset-contracts/contracts/anchor_basset_token/src/contract.rs:37`, the `bAsset` contract allows providing an initial balance when creating the token. However, the is unusable as an increase in a user's balance should be reflected in the reward contract by performing an `IncreaseBalance` message as seen in `cavern-aAsset-contracts/contracts/anchor_basset_token/src/handler.rs:94`.

As the user's balance in `bAsset` contract is not increased, they would be unable to transfer, burn or send their tokens. Additionally, the total supply amount in the `bAsset` contract and the actual balance in the reward contract will be inconsistent.

Recommendations

Consider increasing the user's balance in the reward contract based on the provided initial balance.

12 – Adding too many validators would cause `query_validators` to fail

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

Description

In `cavern-aAsset-contracts/contracts/lido_terra_validators_registry/src/contract.rs:232`, the validators registry contract does not validate the validator's address is registered before storing them into the `REGISTRY` storage. This happens during contract instantiation and adding the validator phase.

If too many validators are registered, an out-of-gas error will occur when processing the loop. For example, removing a validator will fail in `cavern-aAsset-contracts/contracts/lido_terra_validators_registry/src/contract.rs:139`, and bonding will fail in `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/bond.rs:85`.

Recommendations

Consider implementing a max limit of validators that can be registered when adding validator.

13 – Feeding price as zero causes potential division by zero error

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

Description

In `cavern-money-market-contracts/contracts/oracle/src/contract.rs:112`, registered price feeders can feed prices for specific assets. However, no validations ensure the price fed is not zero.

Additionally, if the zero price is being used as a quote, a division by zero error will occur in line 173.

Recommendations

As zero price for an asset is not possible, consider adding a check to ensure the price fed is not zero.

14 – Removing collateral prevents liquidation attempts

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

Description

In `cavern-money-market-contracts/contracts/overseer/src/contract.rs:395`, the `remove_whitelist` function allows the contract owner to remove any collateral tokens from the protocol.

If the removed token is used as collaterals, this will cause liquidations to fail due to the `read_whitelist_elem` function failing in `cavern-money-market-contracts/contracts/overseer/src/state.rs:146-151`, potentially affecting the protocol's solvency.

Recommendations

Consider removing the `remove_whitelist` function.

15 – Incorrect blocks per year constant

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

Description

In `cavern-money-market-contracts/contracts/overseer/src/contract.rs:34`, the value of `BLOCKS_PER_YEAR` constant is `4656810`. In Terra, an average block is produced in 6 seconds. This means a year will be incorrectly calculated as `323` instead of `365` days.

Recommendations

Consider modifying `BLOCKS_PER_YEAR` to `5256000`. This calculation is derived from the total seconds per year divided by 6 seconds (`31536000/6=5256000`).

16 – Peg recovery fee can be updated to be higher than 100%

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

Description

In `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/config.rs:36`, the `execute_update_params` function allows the creator to update the `peg_recovery_fee` value to be higher than 100%.

However, this is prevented during the contract instantiation phase in `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/contract.rs:60-64`, which checks the peg recovery fee is not higher than `Decimal::one()`.

Recommendations

Consider adding validation to ensure the peg recovery fee is lower or equal to `Decimal::one()`.

17 – `Config` query message does not return `validators_registry_contract`

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

Description

In `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/contract.rs:345-350`, the `query_config` function that returns a `ConfigResponse` struct does not include the value of `validators_registry_contract`.

This implies that off-chain users cannot retrieve the address of the validator's registry contract.

Recommendations

Consider including `validators_registry_contract` in the `ConfigResponse` struct.

18 – `execute_withdraw_unbonded` function emits incorrect from attribute key

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

Description

In `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/unbond.rs:196`, the `from` attribute key's value is incorrectly set to the contract address. This is incorrect as the caller is the `sender_human` variable, not the current contract address.

Recommendations

Consider modifying `contract_address` into `sender_human` to indicate the caller address correctly.

19 – Validator address is not validated upon delegation

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

Affected code lines

- `cavern-aAsset-contracts/contracts/lido_terra_validators_registry/src/contract.rs:47`
- `cavern-aAsset-contracts/contracts/lido_terra_validators_registry/src/contract.rs:118`

Description

The validators registry contract does not validate the validator's address is registered before storing them into the `REGISTRY` storage. This happens during contract instantiation and adding the validator phase.

Recommendations

Consider ensuring that the validator is registered using the `deps.querier.query_validator` function.

20 – Stable token's CW20 address is not validated

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

Description

In `cavern-money-market-contracts/contracts/custody_lsd/src/contract.rs:39`, the provided `msg.stable_token` is not validated to be a CW20 token address.

Recommendations

Consider validating the stable token to be a valid address.

21 – Unvalidated decimal values

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

Affected code lines

- `safe_ratio`
  - `cavern-money-market-contracts/contracts/liquidation_queue/src/contract.rs:39, 176`
- `max_ltv`
  - `cavern-money-market-contracts/contracts/overseer/src/contract.rs:344, 379`
- `buffer_distribution_factor`
  - `cavern-money-market-contracts/contracts/overseer/src/contract.rs:57, 273`

Description

Some of the decimal values are not validated to be lower than `Decimal256::one()`, which might cause unintended consequences if misconfigured.

Firstly, the `safe_ratio` should be validated lower than 100% to prevent borrowers from taking out an undercollateralized loan.

Secondly, the `max_ltv` should be validated lower than 100% to prevent borrowers from taking out an undercollateralized loan and higher than 0% to prevent a division by zero error when computing collateral weights in `cavern-money-market-contracts/contracts/liquidation_queue/src/query.rs:172`.

Lastly, the `buffer_distribution_factor` should be validated below 100% to prevent the contract from distributing too many funds in `cavern-money-market-contracts/contracts/overseer/src/contract.rs:582`.

Recommendations

Consider applying the recommendations mentioned above.

22 – Updating fees might fail due to incorrect fee assertion

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

Description

In `cavern-money-market-contracts/contracts/liquidation_queue/src/contract.rs:180` and `185`, incorrect fee assertions may cause errors during updates.

For example, this does not consider whether `config.liquidator_fee` and `config.bid_fee` are updated within the same call. Consequently, the execution will revert, although the new fees are valid.

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

Recommendations

Consider only asserting the fees after updating the bid and liquidator fees.

23 – `query_next_borrower_incentives` fails when no borrows

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

Description

In `cavern-money-market-contracts/contracts/market/src/querier.rs:80`, the `query_next_borrower_incentives` function attempts to divide all the liabilities in the protocol when computing the rate.

If no borrows exist in the protocol, the value will become zero, causing a division by zero error.

Recommendations

Consider only computing the rate when `state.total_liabilities` is higher than zero.

24 – Misconfiguring `known_cw20_tokens` might cause swaps to fail

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

Description

In `cavern-lsd-wrapper/cavern_aAsset_lsd_reward/src/contract.rs:34` and `87`, `known_cw20_tokens` is not validated not to contain duplicates. A misconfiguration of duplicate tokens would cause `SwapToRewardDenom` to fail because it attempts to double-send the total `CW20` tokens in the contract.

Recommendations

Consider adding validation to ensure `config.known_cw20_tokens` does not contain duplicates.

25 – Contracts should use two–step ownership transfer

Risk Impact: Low - **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 the new admin must accept ownership before the transfer and config changes occur.

- `cavern-money-market-contracts/contracts/borrow_incentives_bucket/src/contract.rs`
- `cavern-money-market-contracts/contracts/custody_base/src/contract.rs`
- `cavern-money-market-contracts/contracts/custody_bluna/src/contract.rs`
- `cavern-money-market-contracts/contracts/custody_lsd/src/contract.rs`
- `cavern-money-market-contracts/contracts/distribution_model/src/contract.rs`
- `cavern-money-market-contracts/contracts/interest_model/src/contract.rs`
- `cavern-money-market-contracts/contracts/liquidation_queue/src/contract.rs`
- `cavern-money-market-contracts/contracts/oracle/src/contract.rs`
- `cavern-lsd-wrapper/contracts/cavern_aasset_hub/src/config.rs`
- `cavern-aAsset-contracts/contracts/lido_terra_validators_registry/src/contract.rs`

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.

26 – Configuration update does not emit attributes

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

Affected code lines

- `cavern-money-market-contracts/contracts/borrow_incentives_bucket/src/contract.rs:100`
- `cavern-money-market-contracts/contracts/liquidation_queue/src/contract.rs:206`
- `cavern-money-market-contracts/contracts/oracle/src/contract.rs:63`

Description

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

Consider emitting relevant attributes.

27 – Validating `info.sender` consumes unnecessary gas costs

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

Affected code lines

- `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/contract.rs:37`
- `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/config.rs:25`
- `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/config.rs:60`
- `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/contract.rs:41`
- `cavern-aAsset-contracts/contracts/anchor_basset_hub/src/contract.rs:140`
- `cavern-aAsset-contracts/contracts/anchor_basset_reward/src/user.rs:132`
- `cavern-lsd-wrapper/contracts/cavern_aasset_hub/src/config.rs:19`
- `cavern-lsd-wrapper/contracts/cavern_aasset_hub/src/contract.rs:30`
- `cavern-lsd-wrapper/contracts/cavern_aasset_hub/contract.rs:26`

Description

In some instances of the codebase, `info.sender` is validated as a valid address. This is unneeded as Cosmos SDK ensures the signer is a valid address when interacting with the contract.

Recommendations

Consider removing mentioned code lines above to reduce computation costs.

28 – Misleading `query_accrued_rewards` function

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

Description

In `cavern-lsd-wrapper/contracts/cavern_aAsset_lsd_reward/src/user.rs:54-69`, the `query_accrued_rewards` function expects the caller to provide an address argument value. Essentially, it hints to the caller that it is used to query accrued rewards for a specific user.

However, this is not the case as the function returns the total balance of reward denom in the contract.

Recommendations

Consider modifying the function name into something more appropriate, such as `query_total_reward_denom`.

29 – Decompound emits incorrect `action` attribute

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

Description

In `packages/cavern_lsd_wrapper_token/src/contract.rs:246`, the `action` attribute value emitted for `execute_decompound` function is *"`claim_reward`"*, which is incorrect based on the function name.

Recommendations

Consider modifying the value into *"`execute_decompound`"* instead.

30 – Usage of canonical and non–canonical address

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

Description

In `cavern-aAsset-contracts/contracts/anchor_basset_reward/src/contract.rs:28` and `45`, usages of both canonical and non-canonical addresses could lend to future error and code complexity.

Recommendations

Consider converting all `CanonicalAddr` into `Addr` instead.

31 – Deprecated deduct tax functionality

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

Description

In `cavern-aAsset-contracts/contracts/anchor_basset_reward/src/user.rs:56`, the `deduct_tax` function is used even though it is deprecated.

Recommendations

As Terra does not implement any tax charges now, consider removing the `deduct_tax` function.

32 – Unimplemented `ClaimRewards` message

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

Description

In `cavern-money-market-contracts/contracts/market/src/contract.rs:214`, the `ClaimRewards` message is not implemented.

Recommendations

Consider removing the unused function to reduce code complexity.

33 – `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 `cavern-money-market-contracts/contracts/oracle/src/contract.rs:77` 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.

However, 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 entry point that reflects the state change occurring and will emit the proper attributes.

Recommendations

Consider creating a separate update feeder entry point.

Document control

Version	Date	Approved by	Changes
0.1	23/03/2023	Vinicius Marino	Document Pre-Release
0.2	-	SCV Team	Remediation Revisions
1.0	-	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.