AUDIT REPORT

Hydro Finance

Hydro Protocol

Prepared by SCV-Security

On 23rd November 2023

Table of Contents

Introduction

SCV has been engaged by Hydro Finance to conduct a comprehensive security review with the goal of identifying potential security threats and vulnerabilities within the codebase. The purpose of this audit is to evaluate the security posture of the codebase and provide actionable recommendations to mitigate any identified risks. This report presents an overview of the findings from our security audit, outlining areas of concern and proposing effective measures to enhance the codebase's security.

Scope Functionality

Hydro is a Liquid Staking Derivatives (LSD) protocol on the Injective Network. The contracts in scope include the following:

- Auto-compound contract that compounds staking rewards.
- Community contract that holds community funds.
- Farm contract that lets users stake wrapped tokens and boost tokens.
- Governance contract that lets users create, vote, and execute proposals.
- LP staking contract that lets users stake liquidity pool tokens.
- LSD contract that mints wrapped tokens from native funds and stakes them to validators.
- LSD reward contract that rewards wrapped token holders with staking rewards.
- Wrapped token contract that is used by LSD reward contract.
- X hydro token contract that represents the governance token.

Submitted Codebase

Hydro Protocol Contracts	
Repository	https://github.com/hydro-protocol/contracts
Commit	`cb87f108da0832c1f867f6d44598dbf1e5ea0579`
Branch	`main`

Revisions Codebase

Hydro Protocol Contracts	
Repository	https://github.com/hydro-protocol/contracts
Commit	`51bb6a5e944543408cdc87687dc95d61b6054dfa`
Branch	`main`

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 Hydro Protocol. Testing includes, but is not limited to, the following:

- Understanding the application and its functionality 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

This section provides an evaluation of specific criteria aspects as described below:

- **Documentation:** Evaluating the presence and comprehensiveness of publicly available or provided explanatory information, diagram flowcharts, comments, and supporting documents to enhance code understanding.
- **Coverage:** Evaluating whether the code adequately addresses all necessary cases and scenarios, ensuring that the intended functionality or requirements are sufficiently covered.
- **Readability:** Assessing how easily the code can be understood and maintained, considering factors such as code structure, naming conventions, and overall organisation.
- **Complexity:** Evaluating the complexity of the code, including factors such as, number of lines, conditional statements, and nested structures.

The status of each criteria is categorised as either **SUFFICIENT** or **NOT-SUFFICIENT** based on the audit assessment. This categorisation provides insights to identify areas that may require further attention and improvement.

Criteria	Status	Notes
Documentation	**SUFFICIENT**	N/A
Coverage	**SUFFICIENT**	Testing coverage is considered sufficient, although there is room for improvement as the current coverage only extends to 39.00% of the code.
Readability	**SUFFICIENT**	The codebase had good readability overall and utilised many Rust and CosmWasm best practices.
Complexity	**SUFFICIENT**	N/A

Findings Summary

Summary Title	Risk Impact	Status
Share inflation attack in `auto_compound` contract	CRITICAL	RESOLVED
Unstaked amount will get auto-compounded, preventing users from claiming	SEVERE	RESOLVED
Reserved amount will be sent to reward contract, causing users unable to claim funds	SEVERE	RESOLVED
`in_undelegation_amount` is not added during undelegation	SEVERE	RESOLVED
Batch executions might not undelegate the full amount, causing users unable to claim their funds	SEVERE	RESOLVED
Incorrect computation of reward distribution	SEVERE	RESOLVED
Anyone can update user reward type	SEVERE	RESOLVED
Incorrect assignment of address parameter	SEVERE	RESOLVED
Incorrect calculation of `total_bonded`	SEVERE	RESOLVED
Rewards should be computed and distributed before updating `distribution_schedule`	MODERATE	RESOLVED
Swap transaction will not revert, causing funds stuck in contract	MODERATE	RESOLVED
`fee_rate` not validated causing underflow	LOW	RESOLVED
Potential misconfiguration of validators	LOW	RESOLVED
Overflow and division by zero error due to `boost_ratio` misconfiguration	LOW	RESOLVED
Incorrect undelegate amount	LOW	RESOLVED
Underflow if `start_time_seconds` is larger than `end_time_seconds`	LOW	RESOLVED
Incorrect user reward computed	LOW	RESOLVED
State inconsistencies due to non-updatable values	LOW	RESOLVED
Poll deposit is stuck in contract if the quorum is not reached	LOW	RESOLVED
`validate_executions` function can be bypassed	LOW	RESOLVED

Possible sandwich attack in lsd reward staking contract	LOW	RESOLVED
Unnecessary rounding causes lesser funds claimed	LOW	RESOLVED
`check_available_balance` function can be bypassed	LOW	RESOLVED
`ADMIN_NOMINEE` is immediately updated without being approved	LOW	RESOLVED
Contract migration does not set migration version after migration is complete	INFO	RESOLVED
Out-of-gas due to recursive function call	INFO	RESOLVED
`ADMIN_NOMINEE` is not removed after being approved	INFO	RESOLVED
Potential out-of-gas for large distribution schedules	INFO	RESOLVED
Incorrect query due to claimed_seconds not being set	INFO	RESOLVED
Instantiate response is not emitted	INFO	RESOLVED
Incorrect response action emitted	INFO	RESOLVED
Unimplemented entry point in lsd_contract	INFO	RESOLVED
Misleading `seconds` parameter used	INFO	RESOLVED
Incomplete fields returned from query	INFO	RESOLVED
Poll execution delay can be bypassed	INFO	RESOLVED

Findings Technical Details

1. Share inflation attack in `auto_compound` contract

RISK IMPACT: CRITICAL	STATUS: RESOLVED

Description

In the `delegate_hook` function in `contracts/auto_compound/src/executions/delegate.rs:73`, a share inflation attack can be carried out to steal funds from users. After the contract is instantiated, the attacker will deposit 1 fund for 1 share.

When a victim is about to deposit 1000 `inj`, the attacker will front-run them and transfer 1001 `inj` to the contract. This would cause the `state.total_bonded` amount to be inflated because it includes the attacker sent funds in `contracts/auto_compound/src/executions/auto_compound.rs:82`.

When the `delegate_hook` function is entered, the share calculation will be computed as `shares = 1000 * 1 / (1001+1)`, which evaluates as 0.99. The amount will be rounded into 0, causing the victim to receive 0 shares from their 1000 `inj` deposit.

Ultimately, the attacker can redeem their funds plus the victim's funds with their share, causing a loss of funds for the victim.

Recommendation

Consider implementing virtual shares and decimal offsets to prevent the share inflation attack.

2. Unstaked amount will get auto-compounded, preventing users from claiming

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

Upon completion of the unstake process, `inj` tokens are returned to this contract. This contract automatically compounds `inj` tokens by capturing the available amount. When a user claims the undelegated funds, the `claim` function in `contracts/auto_compound/src/executions/claim.rs:31` will be called to deduct the `state.in_undelegation_amount`.

The issue arises from the `auto_compound_hook` function, situated at `contracts/auto_compound/src/executions/auto_compound.rs:52`, where the `total_reward` encompasses the `21` days unbonded amount. Following the deduction of the `fee_amount` from the `total_reward`, the resulting amount is redelegated, depleting the contract's available funds. Consequently, when a user attempts to claim, the process fails at `contracts/auto_compound/src/executions/claim.rs:36-39` due to insufficient funds within the contract.

It is essential to note that the `auto_compound` function in `contracts/auto_compound/src/executions/auto_compound.rs:59` should take into account `state.in_undelegation_amount` while processing.

Recommendation

Consider subtracting the batch amount (`state.in_undelegation_amount`) from the total amount to ensure that users can successfully claim their rewards after the undelegation ends.

3. Reserved amount will be sent to reward contract, causing users unable to claim funds

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

In the `claim_reward` function in `contracts/lsd/src/executions/claim_reward.rs:34`, upon completion of the undelegation process and `21` days have passed, the user will claim their undelegated `inj` tokens in the lsd contract. However, all undelegations are currently being sent to the reward contract, thereby preventing the user from claiming them. Consequently, users will be unable to claim their rewards due to insufficient contract balance.

Recommendation

Consider modifying the `claim_reward` function to exclude the undelegation amount in the contract for the user to claim and only send the delegated rewards to the rewards contract.

4. `in_undelegation_amount` is not added during undelegation

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

In the `claim` function in `contracts/lsd/src/executions/claim.rs:31`, the claim will fail when deducting the `state.in_undelegation_amount` value. This is because the value will always be 0 as it is not incremented when calling the `Undelegate` message, preventing users from claiming correctly.

Recommendation

Consider modifying the `undelegate` function in `contracts/lsd/src/executions/undelegate.rs:10` to increment the `state.in_undelegation_amount` by `return_amount`.

5. Batch executions might not undelegate the full amount, causing users unable to claim their funds

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

When an admin configures new validators, they must call `ReDelegate` to move the delegated funds from the old validator to the new validator so unbondings works as expected. However, redelegations may not be completed instantly and may require more time.

In this case, if a batch is ready to be executed in `contracts/lsd/src/executions/execute_batch.rs:37`, the validators might not have enough funds to satisfy the required `batch.amount`. This would cause the batch to not undelegate the full amount.

Consequently, users will be unable to claim their funds due to insufficient contract balance.

Recommendation

Consider adding validation to ensure that the `left_amount` is `0` after undelegation finishes in the `execute_batch` function.

6. Incorrect computation of reward distribution

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

In the `distribute_reward` function in `contracts/lsd_reward_staking/src/executions/distribute_reward.rs:35`, the `claimed_amount` is currently determined as the `delegation_rewards`. However, when `ClaimReward` is called in the lsd contract, the `total_reward` includes the lsd contract balance and the delegations rewards.

This would cause the `reward_without_fee` computed in `contracts/lsd_reward_staking/src/executions/distribute_reward.rs:39` to be incorrect because it does not include the lsd contract's balance. Consequently, the reward computed will be less than intended.

Recommendation

Consider modifying the `distribute_reward` function to include the lsd contract's balance when computing rewards.

7. Anyone can update user reward type

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

Anyone can update a user's reward type in the `update_user_reward_type_hook` function in `contracts/lsd_reward_staking/src/executions/update_config.rs:43`, as the function does not have authorization checks in place to ensure that the caller is the lsd reward staking contract. This unauthorized alteration can result in a reduction of the total tokens staked within the contract, thereby enabling the attacker to benefit from the increased rewards.

Recommendation

Consider adding a validation check to ensure that the caller is the lsd reward staking contract itself.

8. Incorrect assignment of address parameter

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

In the `execute` function in `contracts/x_hydro/src/entrypoints.rs:96,105`, and `109`, the address input parameter is incorrectly assigned to `info.sender` instead of the owner's address. This is incorrect because when calling `TransferFrom`, `SendFrom`, and `BurnFrom`, the funds are consumed by the owner, not the `info.sender`.

This will also cause the governance contract to fail during stake token redemption from the governance token, given the governance contract should not possess ownership of tokens.

Recommendation

Consider modifying the `address` parameters to use the owner's address.

9. Incorrect calculation of `total_bonded`

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

In the `auto_compound_hook` function in `contracts/auto_compound/src/executions/auto_compound.rs:82`, the addition of `total_bonded` incorrectly calculate both the `delegate_amount` and `fee_amount`, whereas the intended behavior dictates that only the `delegate_amount` should contribute to the calculation of `total_bonded`.

This miscalculation results in an inaccurate representation of `total_bonded`, surpassing the correct value. In scenarios where `total_bonded` is utilized as the denominator in calculations, such as when calculating a user's share, the inflated value can result in users receiving a smaller share than intended.

Recommendation

Consider modifying the calculation of `total_bonded` to exclude `fee_amount` by using the `delegate_amount`.

10. Rewards should be computed and distributed before updating `distribution_schedule`

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

In the `update_config` function in `contracts/farm/src/executions/update_config.rs:44`, the existing user's rewards should be computed and distributed before the `distribution_schedule` is updated. Failure to do so would cause the contract to distribute an incorrect amount of rewards.

Similarly, this issue exists in the `update_config` function in `contracts/lsd_reward_staking/src/executions/update_config.rs:112`.

Recommendation

Consider computing and distributing the rewards before updating the `distribution_schedule`, such as in `contracts/lp_staking/src/executions.rs:152`.

11. Swap transaction will not revert, causing funds stuck in contract

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

In the `distribute_reward` function in `contracts/lsd_reward_staking/src/executions/distribute_reward.rs:91`, in the event where a swap fails due to the maximum spread limit in Astroport, the transaction will not automatically revert due to `ReplyOn::Always`.

While the sent `inj` tokens will be refunded within the contract, they remain unused.

Recommendation

Consider parsing the `msg` result in `contracts/lsd_reward_staking/src/executions/distribute_reward.rs:109` such that in the event of a swap failure, the corresponding amount is stored and can be later by the admin.

12. `fee_rate` not validated causing underflow

RISK IMPACT: LOW	STATUS: RESOLVED

Description

During contract instantiation In `contracts/auto_compound/src/entrypoints.rs:38`, the configured `fee_rate` lacks validation to ensure it is less than or equal to `100%`. This absence of validation poses a significant risk, as exceeding `100%` could result in an underflow scenario at `contracts/auto_compound/src/executions/auto_compound.rs:63`. Specifically, the `fee_amount`, being larger than the `amount` from which it is subtracted, would trigger an underflow condition.

Additional affected code lines:

- `contracts/lsd_reward_staking/src/entrypoints.rs:49`
- `contracts/lsd_reward_staking/src/executions/distribute_reward.rs:39`
- `contracts/lsd_reward_staking/src/executions/update_config.rs:81`

Recommendation

Consider adding a validation check to ensure that the configured `fee_rate` does not surpass the threshold of `100%`.

13. Potential misconfiguration of validators

RISK IMPACT: LOW	STATUS: RESOLVED

Description

During contract instantiation In `contracts/auto_compound/src/entrypoints.rs:39`, the configured `validators` lack validation to ensure it is not duplicated, empty, and valid address. This is risky as a duplicate validator being configured would potentially cause the `pick_validator` function to pick a validator that has been previously chosen. Having a duplicate validator would cause the `StakingMsg::Undelegate` of the `execute_batch` function in `contracts/auto_compound/src/executions/execute_batch.rs:57` to not be called, causing users to fail to claim their funds due to insufficient balance.

Additionally, if the validators being configured are empty, a division by 0 error in `contracts/auto_compound/src/states.rs:165`, This will cause the pick_validator function to fail, as the function will panic when the error occurs.

Additional affected code lines:

- `contracts/lsd/src/entrypoints.rs:39`
- `contracts/lsd/src/states.rs:171`
- `contracts/lsd/src/executions/update_config.rs:32`

Recommendation

Consider adding a validation check to ensure that the configured `validators` have no duplicates, not empty and have valid addresses.

14. Overflow and division by zero error due to `boost_ratio` misconfiguration

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In the `instantiate` function in `contracts/farm/src/entrypoints.rs:130`, the `boost_ratio` being instantiated is not validated to be less than `100%`. This will result in the calculation of `reward_distributed_amount` in the `compute_reward` function in `contracts/farm/src/states.rs:109` to overflow.

Additionally, in the `get_user_boost_max` function in `contracts/farm/src/states.rs:229`, having a `config.boost_ratio` of `100%` will result in a division by `0` error.

Recommendation

Consider adding a validation check to ensure that the configured `boost_ratio` is less than `100%`.

15. Incorrect undelegate amount

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In the `undelegate` function in `contracts/auto_compound/src/executions/undelegate.rs:42,44,47,51`, it is imperative to decrement each value of `amount` by `1`. This adjustment is necessary to accommodate the rounding check implemented in `contracts/auto_compound/src/states.rs:192`. This will result in extra funds not being claimed by anyone in the `in_undelegation_amount`, resulting in the extra funds being bonded for rewards.

Recommendation

Consider adjusting the values of the aforementioned amount to accurately reflect the correct value.

16. Underflow if `start_time_seconds` is larger than `end_time_seconds`

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In the `compute_reward` function in `contracts/farm/src/states.rs:102`, the `end_time_seconds` is not validated to be larger than or equal to `start_time_seconds`. This will result in the calculation of `num_seconds` in the `compute_reward` to underflow.

Additionally, if the `end_time_seconds` is equal to `start_time_seconds`, a division by `0` will occur in `contracts/farm/src/states.rs:103` as the `num_seconds` being used as the denominator to calculate `distribution_amount_per_second` is `0`.

Recommendation

Consider adding a validation check to ensure that `end_time_seconds` is larger than `start_time_seconds`.

17. Incorrect user reward computed

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In the `get_pending_reward` function in `contracts/farm/src/states.rs:308`, if the `config.distribution_schedule` or the `boost_ratio` is updated, it can result in an incorrect computation of the user's reward. This is because the compute_reward function relies on both values when computing the rewards, as seen in lines 93 and 109.

This scenario causes the `compute_user_reward` function to produce an inaccurate calculation of the user's pending reward, leading to an incorrect amount being returned by the `query_boost` function when called.

Recommendation

Consider modifying the function to use a snapshot of the `Config` to ensure that the user's rewards are not affected by config updates.

18. State inconsistencies due to non-updatable values

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In the `update_config` function in `contracts/farm/src/executions/update_config.rs:33`, the value of `wrapped_token` can be updated after the contract's instantiation. Changing the `wrapped_token` could introduce unintended state inconsistencies because `state.total_bond_amount` records the amount with a different token denom.

Similarly in the `update_config` function in `contracts/lp_staking/src/executions.rs:140`, the value of `lp_token` can be updated after the contract's instantiation which can cause state inconsistencies as mentioned above.

Recommendation

Consider removing the ability to update the value of `wrapped_token` and `lp_token` to prevent unintended behaviors from occurring.

19. Poll deposit is stuck in contract if the quorum is not reached

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In the `end_poll` function in `contracts/governance/src/poll/executions.rs:254`, if the quorum is not reached and the `end_poll` function is called, the poll deposit remains in the contract. The funds will be stuck in the contract and cannot be withdrawn.

Recommendation

Consider modifying the `end_poll` function to burn the remaining poll deposits if the quorum is not reached.

20. `validate_executions` function can be bypassed

RISK IMPACT: LOW	STATUS: RESOLVED

Description

The `validate_executions` function in `contracts/governance/src/poll/executions.rs:455`, can be bypassed by calling the blocked executions within an `ExecuteMsg::RunExecution`.

```
#[test]
fn test_bypass_execution_whitelist() {
    // test in contracts/governance/src/poll/executions.rs
    let contract_addr = "env.contract.address";
    let execs = vec![ExecutionMsg {
        order: 1,
        contract: contract_addr.to_string(),
        msg: cosmwasm_std::to_binary(&ExecuteMsg::RunExecution {
            executions: vec![ExecutionMsg {
                order: 1,
                contract: contract_addr.to_string(),
                msg: cosmwasm_std::to_binary(&Cw20ExecuteMsg::Transfer {
                    recipient: "someone".to_string(),
                    amount: Uint128::new(1000),
                })
                .unwrap(),
            }],
        })
        .unwrap(),
    }];
    validate_executions(&execs).unwrap();
}
```

Recommendation

Consider adding validation in the `validate_executions` function to prevent running `ExecuteMsg::RunExecution`.

21. Possible sandwich attack in lsd reward staking contract

RISK IMPACT: LOW	STATUS: RESOLVED

Description

The `distribute_reward` function in `contracts/lsd_reward_staking/src/executions/distribute_reward.rs:86` interacts with DEX to perform a swap. However, the max default spread is set to `50%`. This allows attackers to perform a sandwich attack and skim the reward amount.

Recommendation

Consider setting the slippage to a lower value (e.g., `5%` or `10%`) and allowing the contract admin to configure default slippage based on market conditions.

22. Unnecessary rounding causes lesser funds claimed

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In `contracts/lsd/src/states.rs:145` of the `add_undelegation` function, the rounding is not needed as it is a direct minting. This would result in the user's undelegated amount being lesser than the intended value.

Recommendation

Consider removing the `add_undelegation` function so it does not remove the rounding from the user's undelegated amount.

23. check_available_balance function can be bypassed

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In the check_available_balance function in contracts/x_hydro/src/state.rs:99, the check can be bypassed by executing transferring funds before casting a vote. This is due to the fact that, during the governance's vote-casting process, the token balance is queried based on snapshot balance in contracts/governance/src/poll/executions.rs:201.

Recommendation

Consider revising the implementation to ensure users cannot transfer funds after voting governance funds.

24. ADMIN_NOMINEE is immediately updated without being approved

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In the `update_config` function in `contracts/community/src/executions.rs:51`, the admin is immediately updated without going through the `approve_admin_nominee` approval process.

Recommendation

Consider modifying the function to only update the admin address after going through the `approve_admin_nominee` approval process.

25. Contract migration does not set migration version after migration is complete

RISK IMPACT: INFO	STATUS: RESOLVED

Description

During contract migration, the target version of the migration is not set. End users invoking the `get_contract_version` function to inquire about the contract version will receive inaccurate information due to the absence of the correct migration target version.

There are several instances throughout the codebase that this occurs:

- `contracts/auto_compound/src/entrypoints.rs:139`
- `contracts/community/src/entrypoints.rs:61`
- `contracts/farm/src/entrypoints.rs:112`
- `contracts/governance/src/entrypoints.rs:159`
- `contracts/lp_staking/src/entrypoints.rs:107`
- `contracts/lsd/src/entrypoints.rs:130`
- `contracts/lsd_reward_staking/src/entrypoints.rs:123`
- `contracts/wrapped_token/src/entrypoints.rs:143`
- `contracts/x_hydro/src/entrypoints.rs:157`

Recommendation

Consider calling `set_contract_version` when performing contract migration.

26. Out-of-gas due to recursive function call

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In `contracts/auto_compound/src/queries.rs:73`, the `query_reward_distribution` function exhibits a recursive behavior that leads to execution failure as it exhausts gas resources causing an out-of-gas to occur.

Recommendation

Consider completing the `query_reward_distribution` function so it returns the reward distribution schedules correctly.

27. ADMIN_NOMINEE is not removed after being approved

RISK IMPACT: INFO	STATUS: RESOLVED

Description

The `approve_admin_nominee` function throughout the codebase does not remove the `ADMIN_NOMINEE` after the new `ADMIN_NOMINEE` has been approved.

Affected code lines:

- `contracts/auto_compound/src/executions/update_config.rs:57`
- `contracts/community/src/executions.rs:60`
- `contracts/farm/src/executions/update_config.rs:68`
- `contracts/lp_staking/src/executions.rs:198`
- `contracts/lsd/src/executions/update_config.rs:58`
- `contracts/lsd_reward_staking/src/executions/update_config.rs:122`
- `contracts/wrapped_token/src/executions.rs:41`
- `contracts/x_hydro/src/executions.rs:53`

Recommendation

Consider removing the `ADMIN_NOMINEE` after it has been approved and admin is updated.

28. Potential out-of-gas for large distribution schedules

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In `contracts/farm/src/states.rs:93`, the `distribution_schedule` in the `compute_reward` is implemented as a vector. If the vector is configured too large, transaction failures may occur due to running out of gas.

Recommendation

Consider modifying the function to use a minimum limit so that it validates within a reasonable limit.

29. Incorrect query due to `claimed_seconds` not being set

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In the `claim` function in `contracts/farm/src/executions/claim.rs:41`, when a user executes a claim, the attribute `claim.claimable_seconds` is not assigned to `amount_list[0].claimable_seconds` before being returned. This will lead to the `claim.claimable_seconds` defaulting as `0`, generating inaccurate query results for the user.

Recommendation

Consider setting `claim.claimable_seconds` to `amount_list[0].claimable_seconds`.

30. Instantiate response is not emitted

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In `contracts/governance/src/entrypoints.rs:44`, the `instantiate` function does not emit the responses of each individual instantiation that resides within lines 25, 31, and 37 as the response from each of these instantiations is not captured and emitted.

Recommendation

Consider modifying the `instantiate` entry point so instantiate event is emitted.

31. Incorrect response action emitted

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In `contracts/governance/src/staking/executions.rs:87`, the `stake` function has the value "`bond`" in the `make_response` function when it should be "`stake`".

Additionally, the `claim_hook` function in `contracts/lsd_reward_staking/src/executions/claim.rs:95` emits the value of "`info.sender`" for the key attribute "`owner`" when the value should be the claimer instead.

Recommendation

Consider modifying the responses to accurately emit their respective values.

32. Unimplemented entry point in `lsd_contract`

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In the `update_config` function in `contracts/lsd_reward_staking/src/executions/update_config.rs:71`, a validation check is implemented to ensure that the caller is the `lsd_contract`. However, there is no corresponding entry point to this function in the `lsd_contract`.

Recommendation

Consider removing the entry point if it is unused.

33. Misleading seconds parameter used

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In the `query` function in `contracts/wrapped_token/src/entrypoints.rs:124`, the `QueryMsg::BalanceAt` is given the `seconds` parameter. However, it returns the balance of the given address at the given block, not the given seconds.

Recommendation

Consider modifying the input parameter of seconds to block height.

34. Incomplete fields returned from query

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In the `query_user_state` function in `contracts/farm/src/queries.rs:23`, the `UserStateResponse` does not return the values of `pending_reward` and `last_distributed`.

Recommendation

Consider returning these values in the `query_user_state` function.

35. Poll execution delay can be bypassed

RISK IMPACT: INFO	STATUS: RESOLVED

Description

In the `execute_poll` function in `contracts/governance/src/poll/executions.rs:294`, the `poll.end_seconds` does not accurately reflect the actual end time of the poll, especially when compared to the time following the execution of the `end_poll` function. If the `end_poll` function is at a later time (after `poll.end_seconds` plus `poll_config.execution_delay_period`), the execution delay is effectively bypassed.

Recommendation

Consider implementing a timestamp that reflects the time the proposal ended and validates it in the `execute_poll` function.

Document Control

Version	Date	Notes
-	30th October 2023	Security audit commencement date.
0.1	20th November 2023	Initial report with identified findings delivered.
0.5	20th - 23rd November 2023	Fixes remediations implemented and reviewed.
1.0	23rd November 2023	Audit completed, final report delivered.

Appendices

A. Appendix – Risk assessment methodology

SCV-Security employs a risk assessment methodology to evaluate vulnerabilities and identified issues. This approach involves the analysis of both the LIKELIHOOD of a security incident occurring and the potential IMPACT if such an incident were to happen. For each vulnerability, SCV-Security calculates a risk level on a scale of 5 to 1, where 5 denotes the highest likelihood or impact. Consequently, an overall risk level is derived from combining these two factors, resulting in a value from 10 to 1, with 10 signifying the most elevated level of security risk

Risk Level	Range
CRITICAL	10
SEVERE	From 9 to 8
MODERATE	From 7 to 6
LOW	From 5 to 4
INFORMATIONAL	From 3 to 1

LIKELIHOOD and **IMPACT** would be individually assessed based on the below:

Rate	LIKELIHOOD	IMPACT
5	**Extremely Likely**	Could result in severe and irreparable consequences.
4	**Likely**	May lead to substantial impact or loss.
3	**Possible**	Could cause partial impact or loss on a wide scale.
2	**Unlikely**	Might cause temporary disruptions or losses.
1	**Rare**	Could have minimal or negligible impact.

B. Appendix – Report Disclaimer

This report should not be regarded as an "endorsement" or "disapproval" of any specific project or team. These reports do not indicate the economics or value of any "product" or "asset" created by a team or project that engages SCV-Security for a security review. The audit report does not make any statements or warranties about the code's utility, safety, suitability of the business model, regulatory compliance of the business model, or any other claims regarding the fitness of the implementation for its purpose or its bug-free status. The audit documentation is intended 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 damages arising from or in connection with this audit report. Copyright of this report remains with SCV-Security.

THANK YOU FOR CHOOSING

SCV SECURITY

scv.services

contact@scv.services