AUDIT REPORT

Apollo DAO

Neutron Vault

Prepared by SCV-Security

Prepared on 1st September 2023

Table of Contents

Introduction

SCV has been engaged by Apollo DAO 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

The Apollo Astroport vault contract is a vault contract that auto-compounds the rewards generated by the Astroport Generator contract. Users can deposit liquidity tokens in exchange for the vault share tokens and vice versa. An optional lockup period can be configured by the owner only to allow users to withdraw tokens after the lockup has matured.

The Neutron Astroport reward distributor contract holds vault tokens and distributes them over a period of time. The vault tokens are redeemed into liquidity tokens to withdraw the assets from the pool. The assets are sent to the configured distribution address as part of the reward distribution flow.

Submitted Codebase

locked-astroport-vault	
Repository	https://github.com/apollodao/locked-astroport-vault
Commit	`b68a2091b8d5a555f0da0a1e2e941071f914256f`
neutron-reward-distributor	
Repository	https://github.com/apollodao/neutron-reward-distributor
Commit	`2e1366324a36049048ac94684945305b46a4281f`

Revisions Codebase

locked-astroport-vault	
Repository	https://github.com/apollodao/locked-astroport-vault
Commit	193b3e5d842819262047b64712dae47d506f0c3d
neutron-reward-distributor	
Repository	https://github.com/apollodao/neutron-reward-distributor
Commit	c2fda3fb46f4102c9253fceabe8e830684e4aad3

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 Apollo DAO. 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**	Documentation was not provided. However, in-line comments were sufficient.
Coverage	**SUFFICIENT**	Coverage tests were provided covering both locked-astroport-vault and neutron-reward-distributor core contracts functionalities.
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
Auto-compounding will fail if the reward asset equals to reward liquidation target	SEVERE	RESOLVED
The reward distributor contract cannot integrate with vaults that have a lockup period	SEVERE	ACKNOWLEDGED
Possible sandwich attacks when performing swaps	SEVERE	ACKNOWLEDGED
Updating emission per second causes incorrect reward distribution	MODERATE	RESOLVED
Owner may accidentally block users from claiming rewards	MODERATE	RESOLVED
Configuring performance fees to 100% or 0% values would cause auto-compounds to fail	LOW	RESOLVED
Rewards can be converted into assets that are not in pool assets	LOW	RESOLVED
Misleading `CanOnlyDistributeOncePerBlock` error	INFO	RESOLVED
Incorrect event emitted in `execute_internal_lp_redeemed`	INFO	RESOLVED
Reward tokens parameter can be retrieved from the generator contract	INFO	ACKNOWLEDGED
Events are not emitted for important executions	INFO	RESOLVED
Incorrect variable naming	INFO	RESOLVED
Unnecessary `BASE_TOKEN` storage	INFO	ACKNOWLEDGED
Added whitelist address can be removed in the same transaction	INFO	RESOLVED
Mismatched documentation and implementation	INFO	RESOLVED
Unnecessary gas consumption when emitting `performance_fee` and `sold_token` attributes	INFO	RESOLVED

Findings Technical Details

1. Auto-compounding will fail if the reward asset equals to reward liquidation target

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

When selling the reward tokens in `locked-astroport-vault/src/execute/compound.rs:42`, all assets inside the configured reward tokens are computed to find the token amount to swap after deducting the performance fees. The tokens are then sent to the router contract to swap into `cfg.reward_liquidation_target`, which is one of the assets in the pool contract.

The problem arises when one of the reward assets is already denominated as `cfg.reward_liquidation_target`. For example, the ASTRO-USDC pool rewards the liquidity token holders with ASTRO. Since ASTRO is already one of the ASTRO-USDC pool assets, swapping in the router contract is unnecessary. As a result, the auto-compound will fail because the router contract tries to swap the ASTRO token into itself.

Recommendation

Consider modifying the `execute_sell_tokens` function so it does not swap pool assets if it equals `cfg.reward_liquidation_target`.

2. The reward distributor contract cannot integrate with vaults that have a lockup period

RISK IMPACT: SEVERE	STATUS: ACKNOWLEDGED

Revision Notes

The team mentioned that they do not intend to support locked vaults for the reward distributor contract.

Description

When distributing rewards in the reward distributor contract in `contracts/reward-distributor/src/execute.rs:53`, the computed vault token shares are redeemed from the vault. After that, the liquidity tokens are used to withdraw the pool assets and sent to the distribution contract, all in a single transaction.

This implies that the reward distributor contract expects the liquidity tokens to be returned directly after redeeming the vault tokens. However, not all vaults are guaranteed to release the tokens without a lockup period. For example, the locked Astroport vault contract in `locked-astroport-vault/src/execute/basic_vault.rs:85-90` will create a claim with a lockup period and allow the caller to claim the liquidity tokens after the lockup is matured.

As a result, the reward distribution will fail because no liquidity tokens are sent to the reward distribution contract. Additionally, there is no entry point to claim the matured lockup in the vault contract to retrieve the unlocked tokens.

Recommendation

Consider adding support to distribute rewards if the vault contract implements a lockup mechanism. Another option is to add validation in the contract instantiation phase to ensure the vault contract does not require a lockup period when redeeming the vault tokens.

3. Possible sandwich attacks when performing swaps

RISK IMPACT: SEVERE	STATUS: ACKNOWLEDGED

Revision Notes

Team is aware and decided to acknowledge it as the risk of sandwiching is low and the workarounds to remediate are currently cumbersome.

Description

When selling the reward tokens in `locked-astroport-vault/src/execute/compound.rs:63`, the `minimum_receive` parameter is hardcoded to `None`. This implies that no slippage checks will be performed when the swap is being performed. As a result, an attacker can perform a sandwich attack to extract profit from the yield, causing a loss of profit to vault token holders.

Recommendation

Consider allowing the owner to configure a `mininum_receive` value and use it in the `execute_sell_tokens` function. The owner should update the value to adapt to different market conditions.

4. Updating emission per second causes incorrect reward distribution

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

When distributing rewards in the reward distributor contract in `contracts/reward-distributor/src/execute.rs:28`, the `redeem_amount` value is computed by multiplying the `config.emission_per_second` and the seconds elapsed since the last distribution. The computed value represents the reward that needs to be sent to the distribution contract.

This implies that the `emission_per_second` configuration influences the reward amount directly. Suppose the owner updates the `config.emission_per_second` value using the `execute_update_config` function, the resulting reward amount will be manipulated. As a result, the new `emission_per_second` configuration will be applied to the old time elapsed value, causing excess or lesser reward distribution based on the configuration changes.

Recommendation

Consider modifying the `execute_update_config` function to compute the reward amount with the old `emission_per_second` value before updating it.

5. Owner may accidentally block users from claiming rewards

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

When retrieving rewards from a lockup position in `locked-astroport-vault/src/contract.rs:163-186`, there is a validation that errors if the configured lock duration is zero. The lock duration represents the period of time that the users need to wait before claiming the liquidity tokens.

Since the owner can update the lock duration using the `execute_update_config` function, the owner might accidentally block users from accessing their matured lockups. For example, assume the lock duration was configured to one day. A user redeems their vault tokens and waits for the lockup to mature. The owner updates the lock duration to zero, so future liquidity tokens are released instantly without a lockup period.

However, this prevents the user from claiming their matured lockup due to the `cfg.lock_duration.is_zero()` validation. Additionally, it may also prevent third-party integrations from failing to work correctly. For instance, the comment in `locked-astroport-vault/src/state.rs:36-39` hints that the `ForceRedeem` and `ForceWithdrawUnlocking` messages are used when liquidating vault tokens from the Mars protocol. As a result, the owner might block users and third-party liquidations from working correctly after updating the lock period value to zero.

Recommendation

Consider removing the validations in `locked-astroport-vault/src/contract.rs:150-152` and lines 172 to 174.

6. Configuring performance fees to 100% or 0% values would cause auto-compounds to fail

RISK IMPACT: LOW	STATUS: RESOLVED

Description

When selling the reward tokens in `locked-astroport-vault/src/execute/compound.rs:38-43`, the configured performance fee is used to calculate the fees to be sent to the treasury address while the remaining amount acts as the number of tokens to sell.

The problem arises when the owner configures the performance fee to 100% or 0%. In this case, either the fee or the token to sell amount will become zero. Since Cosmos SDK prevents zero native amount transfers, the `execute_sell_tokens` function will fail in line 46 when transferring funds to the treasury contract or in line 60 when swapping in the router contract.

Recommendation

Consider only adding the asset to the `performance_fees` and `tokens_to_sell` asset list if the amount exceeds zero.

7. Rewards can be converted into assets that are not in pool assets

RISK IMPACT: LOW	STATUS: RESOLVED

Description

When instantiating the vault contract in `locked-astroport-vault/locked-astroport-vault/src/contract.rs:52`, the `msg.reward_liquidation_target` is not validated to be in `pair_info.asset_infos`. If the `msg.reward_liquidation_target` parameter provided during instantiation is not in `pair_info.asset_infos`, the rewards would be converted into an asset that is not one of the pool assets, ultimately causing the auto-compounding to fail.

Recommendation

Consider adding validation checks to ensure that the `msg.reward_liquidation_target` is in `pair_info.asset_infos`.

8. Misleading `CanOnlyDistributeOncePerBlock` error

RISK IMPACT: INFO	STATUS: RESOLVED

Description

When distributing rewards in the reward distributor contract in `contracts/reward-distributor/src/execute.rs:23`, a `CanOnlyDistributeOncePerBlock` error will occur if the current timestamp equals the last distributed timestamp retrieved from the `LAST_DISTRIBUTED` storage state. The error message is misleading because it mentions that distribution can only happen once per block, but the comparison actually means the distribution can only happen once per second.

Recommendation

Consider modifying the error message to mention distribution can only happen once per second.

9. Incorrect event emitted in `execute_internal_lp_redeemed`

RISK IMPACT: INFO	STATUS: RESOLVED

Description

The `execute_internal_lp_redeemed` function in `contracts/reward-distributor/src/execute.rs:107` emits the event type as "*apollo/neutron-astroport-reward-distributor/execute_internal_vault_tokens_redeemed*". This is incorrect because this event should only be emitted in the `execute_internal_vault_tokens_redeemed` function.

Recommendation

Consider emitting the event type as "*apollo/neutron-astroport-reward-distributor/execute_internal_lp_redeemed*", which aligns with the function name.

10. Reward tokens parameter can be retrieved from the generator contract

RISK IMPACT: INFO	STATUS: ACKNOWLEDGED

Description

When instantiating the locked Astroport vault contract in `locked-astroport-vault/src/contract.rs:47`, the `msg.reward_tokens` parameter is provided to indicate the rewards assets the Astroport generator contract will send. Instead of relying on the owner-supplied values that are vulnerable to misconfiguration, retrieving the assets using the `RewardInfo` query message in the Astroport generator contract is possible.

This approach eliminates the possibility of the owner misconfiguring incorrect reward assets, ensuring correct assets are auto-compounded and accrued to vault token holders.

Recommendation

Consider modifying the code to retrieve the reward assets by performing a `RewardInfo` query message to the `msg.astroport_generator` with the `lp_token` parameter as `pair_info.liquidity_token` value instead of requiring owner inputs.

11. Events are not emitted for important executions

RISK IMPACT: INFO	STATUS: RESOLVED

Description

When executing contract updates in both the `execute_update_config` and `execute_update_force_withdraw_whitelist` functions, there are no events or attributes emitted for the vault contract as seen in `locked-astroport-vault/locked-astroport-vault/src/execute/basic_vault.rs:97` and `locked-astroport-vault/locked-astroport-vault/src/execute/lockup.rs:33`.

As contract updates are important executions, relevant events, and attributes should be emitted for off-chain listeners to record and index the configured parameters.

Recommendation

Consider emitting the relevant attributes or events when updating the vault contract configuration based on configured parameters.

12. Incorrect variable naming

RISK IMPACT: INFO	STATUS: RESOLVED

Description

The constant variable `COMPOUNT_REPLY_ID` in `locked-astroport-vault/locked-astroport-vault/src/contract.rs:32` is incorrectly named.

Recommendation

Consider modifying the variable name to `COMPOUND_REPLY_ID`.

13. Unnecessary BASE_TOKEN storage

RISK IMPACT: INFO	STATUS: ACKNOWLEDGED

Description

The `BASE_TOKEN` storage is used to store the liquidity token address during contract instantiation phase in `locked-astroport-vault/src/contract.rs:75`. The token address is also recorded in line 87 in the `STAKING.lp_token_addr` storage.

Recommendation

Consider removing the `BASE_TOKEN` storage and using `STAKING.lp_token_addr` instead to retrieve the liquidity pool token address.

14. Added whitelist address can be removed in the same transaction

RISK IMPACT: INFO	STATUS: RESOLVED

Description

When updating the `FORCE_WITHDRAW_WHITELIST` storage in `locked-astroport-vault/src/execute/lockup.rs:44-56`, the newly added whitelist address might be unintentionally removed if the same address is contained in the `add_addresses` and `remove_addresses` vector.

Recommendation

Consider adding validation to ensure that the `add_addresses` and `remove_addresses` vectors do not contain the same address.

15. Mismatched documentation and implementation

RISK IMPACT: INFO	STATUS: RESOLVED

Description

The `LockupExecuteMsg::Unlock` documentation mentions, "*Emits an event with type `UNLOCKING_POSITION_CREATED_EVENT_TYPE` with an attribute with key `UNLOCKING_POSITION_ATTR_KEY` containing an u64 `lockup_id`*". This is not true because the events are not emitted.

Other than that, the `LockupExecuteMsg::EmergencyUnlock` documentation mentions, "*This call should simply unlock `amount` of vault tokens, without performing any other side effects that might cause the transaction to fail.*". This is incorrect because the transaction will fail in `locked-astroport-vault/src/execute/basic_vault.rs:66` if incorrect funds are sent.

Recommendation

Consider modifying the implementation so it aligns with the documentation.

16. Unnecessary gas consumption when emitting `performance_fee` and `sold_token` attributes

RISK IMPACT: INFO	STATUS: RESOLVED

Description

The `performance_fee` and `sold_token` attributes are emitted in their individual token asset list loops in `locked-astroport-vault/src/execute/compound.rs:51-56`. This creates unnecessary gas consumption as these attributes can be emitted along in lines 38 to 43 when computing the fee and token amount to sell.

Recommendation

Consider modifying the `execute_sell_tokens` function so the `performance_fee` and `sold_token` attributes are emitted together in lines 38 to 43.

Document Control

Version	Date	Notes
-	18th August 2023	Security audit commencement date.
0.1	25th August 2023	Initial report with identified findings delivered.
0.5	31st August 2023	Remediations Revisions.
1.0	1st September 2023	Final report Released

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