AUDIT REPORT

Mito Finance

Staking Contract

Prepared by SCV-Security on 25th May 2023

Table of Contents

Introduction

SCV has been engaged by Mito 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

The following CosmWasm contract was in scope:

- staking-contract

Functionality

The Mito Staking Incentives Allocator Contract operates as a custody for reward tokens that are awaiting distribution to stakers. It utilises gauges, which encapsulate an LP token, a timeframe, and a portion of the reward tokens pool that is distributed proportionally during that time frame. To allocate reward tokens, an array of gauges and the reward tokens pool must be supplied via a transaction. The staking contract periodically fetches rewards from the allocator contract, and the allocator contract sends a portion of each gauge's rewards in response. The contract includes API methods for allocating reward tokens via new gauge creation, adding rewards to an existing gauge, and querying gauges.

Submitted Codebase

- **Repository**
    - https://github.com/MitoFinance/mito-contracts/tree/dev/contracts/staking-contract
- **Commit**
    - `fe1747451454363f98e41b4240996e4c26c4b552`

Revisions Remediations

Remediations recommendations were applied by Mito Finance and revised by SCV team on the following:

- **Repository**
    - https://github.com/MitoFinance/mito-contracts/pull/264
- **Commit**
    - `383c4c9b13983cea1df5527e5288fddadd95a0e8`

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 Mito Finance. 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**	Documentations were comprehensive and detailed. A diagram chart was also provided.
Coverage	**SUFFICIENT**	Testing coverage was up to 73 . 71%, making it sufficient.
Readability	**SUFFICIENT**	The codebase had good readability overall and utilised many Rust and CosmWasm best coding practices.
Complexity	**SUFFICIENT**	N/A

Findings Summary

#	Summary Title	Risk Impact	Status
1	Revoking gauges does not remove them	**SEVERE**	**RESOLVED**
2	Reward distribution might fail for a large number of reward denoms	**MODERATE**	**ACKNOWLEDGED**
3	`query_user_stakes` might fail due to out-of-gas	**LOW**	**ACKNOWLEDGED**
4	Contract does not emit custom events or attributes	**INFO**	**ACKNOWLEDGED**
5	Contract should use two-step ownership transfer	**INFO**	**ACKNOWLEDGED**

Findings Technical Details

1 – Funds can become temporarily or permanently locked in contract

Risk Impact: SEVERE	**Status: RESOLVED**

Description

The `revoke_reward_gauge` function in `contracts/staking-allocator/src/contract.rs:162` allows the contract owner to revoke any gauges and refunds the reward tokens to the gauge owner. However, the function does not remove the gauge after sending the funds.
As a result, gauges that are not removed will be used to distribute rewards using the `DistributeCurrentPeriodRewards` message. Since the reward tokens are refunded back to the gauge owner, the contract might have insufficient funds to distribute the required rewards. Moreover, the contract owner might accidentally call `revoke_reward_gauge` multiple times to refund the gauge owner more funds than expected.

Note that this issue is out of scope as the `staking-allocator` contract is not included in the original audit scope.

Recommendation

Consider removing the gauges after refunding the reward tokens to the gauge owner.

2 – Reward distribution might fail for a large number of reward denoms

RISK IMPACT: MODERATE	**STATUS: ACKNOWLEDGED**

Revision Notes

The client advises that they are enforcing a limit of maximum denoms per gauge and the maximum number of gauges at any point in time, which limits the total number of reward denoms.

Description

The staking contract handles the reward distribution using periods that stores the current rewards per liquidity pool token. If new reward denoms are added for the specific liquidity pool token, `contracts/staking-contract/src/contract.rs:290` creates an entry for the new denom and then stores it using `store_rewards_per_lp_token`.
When calculating the rewards for the user, a loop is iterated for all rewards, as seen in line 305. An attacker can exploit this to create a gauge for the liquidity pool token with many different reward denoms, potentially causing `get_accumulated_rewards` to fail due to an out-of-gas error.

The attack difficulty of this is rated as high due to the various fees the attacker must pay when carrying out the attack (e.g., token creation fee, gauge fees, and gas fees). This issue might also simply happen if the liquidity pool token is popular until many reward denoms are created.

Recommendation

Consider limiting the maximum reward denoms allowed for a liquidity pool token.

3 – `query_user_stakes` might fail due to out-of-gas

RISK IMPACT: LOW	STATUS: ACKNOWLEDGED

Revision Notes

The client advises that the number of vaults will be relatively small in the beginning and will consider pagination at a later stage.

Description

The `query_user_stakes` function in `contracts/staking-contract/src/queries.rs:40` iterates all stakes performed by the user without any pagination. If the user staked too many liquidity pool tokens, the query would fail due to an out-of-gas error.
This issue would only exist if many gauges were created from the `staking-allocator` contract.

Recommendation

Consider implementing a pagination mechanism for the `query_user_stakes` function.

4 – Contract does not emit custom events or attributes

RISK IMPACT: **INFORMATIONAL**	**STATUS:** **ACKNOWLEDGED**

Revision Notes

Client advises that custom events will be added at a later stage once requirements are completed.

Description

The current implementation does not emit custom events or attributes during the instantiate and execute phase. This lack of custom events or attributes can make it difficult for users and event indexers to monitor and track the progress of contract execution. Implementing custom events and attributes will provide better transparency and control over the contract execution.

Recommendation

Consider adding custom events and attributes to the instantiate and execute phases.

5 – Contract should use two-step ownership transfer

RISK IMPACT: INFORMATIONAL	STATUS: ACKNOWLEDGED

Revision Notes

As Injective blockchain is permissioned, the owner of the contract will be the governance contract, and the governance process is sufficient for checking the new owner address.

Description

The current ownership transfer for the contract 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.

Recommendation

Consider implementing a two-step ownership transfer where the current owner proposes a new owner address, and then that new owner address must call the contract to accept ownership within a finite time frame. SCV suggests the following implementation:

- https://docs.rs/cw-controllers/latest/cw_controllers/index.html.

Document Control

Version	Date	Notes
0.1	3rd May 2023	Initial report with identified findings delivered.
0.5	24th May 2023	Fixes remediations implemented and reviewed.
1.0	25th May 2023	Audit completed, final report delivered.
-	13th July 2023	Modified report template format and public 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	MODERATE	SEVERE	CRITICAL	CRITICAL
Severe	LOW	MODERATE	SEVERE	SEVERE
Moderate	LOW	MODERATE	SEVERE	SEVERE
Low	LOW	LOW	LOW	MODERATE
Informational	INFO	INFO	INFO	INFO

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.

THANK YOU FOR CHOOSING

SCV
SECURITY

🌐 scv.services

✉ contact@scv.services