SCV SECURITY

AUDIT REPORT

Mito Finance

Mito Contracts

Prepared by SCV-Security

On 7th March 2024

Table of Contents

Table of Contents....................2
Introduction....................3
Scope Functionality....................3
Submitted Codebase....................4
Revision Codebase....................4
Methodologies....................4
Code Criteria....................6
Findings Summary....................7
Findings Technical Details....................8
1. Users can claim multiple times to receive more than the allotted amount.......8
2. Malicious vaults can steal funds from the Mito Master contract....................9
3. The Mito Master contract bears the token creation fee....................12
4. Total subscription amount is not deducted when removing user subscription amount....................13
5. The previous owner holds existing migration permissions....................14
6. Permissionless vault creation may allow attackers to deploy malicious vaults. 15
7. Canceling IDO fails due to insufficient funds....................16
8. Fetched Pyth price may be stale....................17
9. IS_ALLOWED_VAULT_REGISTRATION_ADDRESS cannot be removed....................18
10. Two-step ownership transfer is not implemented....................19
11. set_quote_price differs from documentation....................20
12. Remove duplicate code block....................21
13. Remove debug messages....................22
Document Control....................23
Appendices....................24
A. Appendix - Risk assessment methodology....................24
B. Appendix - Report Disclaimer....................25

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 Functionality

Mito Finance is a Defi platform that provides automated strategy vaults for passive yield generation and token launchpads deployed on the Injective chain.

The in-scope contracts include:

- `cw-vesting` contract that enables the creation of native token streams, allowing payment to be vested continuously over time.
- `mito-launchpad` contract that facilitates fair and decentralized token sales for projects.
- `mito-launchpad-factory` contract that facilitates the deployment of new token launchpads.
- `mito-master` contract that manages the orchestration of the protocol.
- `vault-cpmm-spot` vault contract that implements the x*y=k strategy for spot products.

Submitted Codebase

mito-finance-contracts	
Repository	https://github.com/MitoFinance/mito-contracts
Commit	a1f13a44db3c43c39a605624ab5d7b7211bbe8b9
Branch	dev
Contracts	• contracts/cw-vesting • contracts/mito-launchpad • contracts/mito-launchpad-factory • contracts/mito-master • contracts/vault-cpmm-spot

Revision Codebase

Revisions were broken down in several PRs and commits. The SCV team performed a revision on each up to the commit mentioned below.

mito-finance-contracts	
Repository	https://github.com/MitoFinance/mito-contracts
Commit	c7c5a0e0c89bd7e0e7d116d6e36d216935be585c
Branch	dev
Contracts	• contracts/cw-vesting • contracts/mito-launchpad • contracts/mito-launchpad-factory • contracts/mito-master • contracts/vault-cpmm-spot

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**	Documentation is available on the README pages and https://docs.mito.fi/.
Coverage	**SUFFICIENT**	The mito contracts had extensive test coverage in both unit and integration tests. Mito utilizes the injective test tube for integration testing.
Readability	**SUFFICIENT**	The codebase had good readability overall and utilized many Rust and CosmWasm best practices.
Complexity	**SUFFICIENT**	N/A

Findings Summary

Summary Title	Risk Impact	Status
Users can claim multiple times to receive more than the allotted amount	CRITICAL	RESOLVED
Malicious vaults can steal funds from the Mito Master contract	SEVERE	ACKNOWLEDGED
The Mito Master contract bears the token creation fee	SEVERE	RESOLVED
Total subscription amount is not deducted when removing user subscription amount	SEVERE	RESOLVED
The previous owner holds existing migration permissions	MODERATE	ACKNOWLEDGED
Permissionless vault creation may allow attackers to deploy malicious vaults	MODERATE	ACKNOWLEDGED
Canceling IDO fails due to insufficient funds	MODERATE	ACKNOWLEDGED
Fetched Pyth price may be stale	MODERATE	RESOLVED
`IS_ALLOWED_VAULT_REGISTRATION_ADDRESS` cannot be removed	MODERATE	RESOLVED
Two-step ownership transfer is not implemented	LOW	ACKNOWLEDGED
`set_quote_price` differs from documentation	LOW	RESOLVED
Remove duplicate code block	INFO	RESOLVED
Remove debug messages	INFO	ACKNOWLEDGED

Findings Technical Details

1. Users can claim multiple times to receive more than the allotted amount

RISK IMPACT: CRITICAL	STATUS: RESOLVED

Description

The `claim` function in `contracts/mito-launchpad/src/handle.rs:365` allows a subscriber to perform a claim multiple times to receive more funds than they originally subscribed.

However, the `claim` function simply checks the user's amount in `USER_SUBSCRIPTION_AMOUNTS` and then issues a refund to the caller's address. This is problematic because the amount is not removed from `USER_SUBSCRIPTION_AMOUNTS`. This applies to both successful and unsuccessful launches.

As a result, the same subscriber can call the function multiple times so long as the contract has a balance to receive more funds than their allotted amount from the launchpad contract.

A proof of concept for this finding can be found here.

Recommendation

Consider removing the subscriber from `USER_SUBSCRIPTION_AMOUNTS` after a claim has been processed.

2. Malicious vaults can steal funds from the Mito Master contract

RISK IMPACT: SEVERE	STATUS: ACKNOWLEDGED

Revision Notes

The team mentions that the root cause can be fixed by a chain upgrade. The requirement for allowing `BankMsg::Send` messages can be avoided by adding chain support in `MsgExternalTransfer` to transfer to a default sub account ID. Currently, this is only possible with the `MsgWithdraw` and `BankMsg::Send` messages (see injective-cosmwasm).

The team notes that there is an inherent risk in allowing such open registrations with any code ID. They have no open registration support yet, but will potentially adopt such a code ID whitelist approach and enforce Mito governance to be the migration owner before allowing open registrations.

Description

The `handle_forwarded_message_reply` function in `contracts/mito-master/src/forward.rs:111` parses the response data from the vault and dispatches messages based from the `ForwardedActionResponse`. One of the messages that can be dispatched is `BankMsg::Send`, which transfers funds to the trader's address. This means that a malicious vault can set the response data to an arbitrary value to trick the Mito Master contract to send funds to them.

Consider the following attack scenario:

1. An attacker calls the `register_vault` function in `contracts/mito-master/src/handle.rs:91` to create a vault. They will purposely set the contract migration admin (`msg.vault_upgrade_admin`) to their own address so they can migrate the vault.
2. Once the vault contract is fully instantiated and registered in the Mito Master contract, the attacker migrates the vault to their malicious code ID.

3. The malicious vault dispatches MsgPrivilegedExecuteContract to the InjectiveExec entry point in the Mito Master contract in contracts/mito-master/src/contract.rs:83. The vault_subaccount_id will be set to the malicious vault subaccount ID and the trader_subaccount_id will be set to the attacker's address.
4. The Mito Master contract dispatches BaseVaultExecuteMsg::ForwardedMasterMessage to the malicious vault and handles the response with the FORWARD_MESSAGE_REPLY_ID reply ID in contracts/mito-master/src/forward.rs:70.
5. When the malicious vault contract is executed, it sets an arbitrary value in the Response data. For example, the malicious vault sets the messages field so the Mito Master contract transfers the attacker some native tokens.

```
let mut response : Response<InjectiveMsgWrapper> = Response::new();
let action_response = ForwardedActionResponse {
    tokens_to_mint: None,
    tokens_to_burn: Some(Uint128::new(1_000_000)),
    privileged_position_transfer: None,
    messages: vec![
        CosmosMsg::Bank(cosmwasm_std::BankMsg::Send {
            to_address:
subaccount_id_to_unchecked_injective_address(&trader_subaccount_id),
            amount: vec![
  Coin { denom: "inj".to_string(), amount: Uint128::new(1_000_000) }
            ]
        })
    ],
    submessages: vec![],
};
response.data = Some(to_json_binary(&action_response).unwrap());
```

6. Once the Mito Master contract parses the data in contracts/mito-master/src/forward.rs:129, the attacker will receive the funds owned by the contract in line 174.

Since Injective is a permissioned chain, the attacker cannot simply upload a malicious code ID, as it must be approved by the governance through proposals. Hence, the attacker must be stealthy to exploit this vulnerability, such as purposely concealing the exploit in their contract during the governance voting process. The attacker could also review legitimate vaults that allow manipulation of the `ForwardedActionResponse` to exploit this issue.

Recommendation

Consider modifying the `register_vault` function so the contract migration admin is the Mito Master contract and implementing an entry point for the contract owner to migrate the vaults to whitelisted code IDs.

3. The Mito Master contract bears the token creation fee

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

The `register_vault` function in `contracts/mito-master/src/handle.rs:178` does not validate the caller provided token factory creation fee if `is_subscribing_with_funds` is false. This means that the fees will be taken from the Mito Master contract's balance, causing a loss of funds.

This attack is feasible if `config.is_allowing_open_registrations` is set to true, allowing an attacker to repeatedly create dummy vaults to consume the contract's funds.

Recommendation

Consider adding validation to ensure the caller provided sufficient token factory creation fee.

4. Total subscription amount is not deducted when removing user subscription amount

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

The `remove_addresses_from_whitelist` function in `contracts/mito-launchpad/src/handle.rs:113` clears the `USER_SUBSCRIPTION_AMOUNTS` storage when removing a user from the whitelist and `should_revoke_subscription` is set to true.

However, the `TOTAL_USER_SUBSCRIPTION_AMOUNT` storage is not deducted accordingly to reflect the changes. As a result, incorrect calculations will occur when computing the `user_project_token_amount` and `user_quote_refund_amount` in `contracts/mito-launchpad/src/handle.rs:438-441`.

Recommendation

Consider deducting the `TOTAL_USER_SUBSCRIPTION_AMOUNT` storage from the user's subscription amount.

5. The previous owner holds existing migration permissions

RISK IMPACT: MODERATE	STATUS: ACKNOWLEDGED

Revision Notes

The team mentions that they can call `MsgUpdateAdmin` on the old launchpads if required. They also mentioned that launchpads typically have a distinct period of time after which they are not used anymore, and updating all launchpads programmatically is not easy due to a potential DOS.

Description

The `deploy_launchpad` function in `contracts/mito-launchpad-factory/src/handle.rs:159` sets the launchpad contract's migration admin to the current owner. If many launchpads have been instantiated and the ownership is transferred to a new owner, the previous owner still holds sufficient permissions to migrate the launchpad contracts.

Recommendation

Consider calling `WasmMsg::UpdateAdmin` when transferring the ownership so all launchpad contracts have their contract migration admin as the new owner.

6. Permissionless vault creation may allow attackers to deploy malicious vaults

RISK IMPACT: MODERATE	STATUS: ACKNOWLEDGED

Revision Notes

The team mentions that malicious vaults can always steal user funds, so it is not directly an issue. Adding the whitelist approach would, of course, prevent this.

Description

The `RegisterVault` function in `contracts/mito-master/src/handle.rs:91` takes in a caller-supplied param of `vault_code_id`. This would allow the caller to supply any `code_id` of their choosing. This opens up the opportunity for an attacker to deploy a modified vault contract that can potentially steal user funds with a backdoor. If the mito master contract has the `is_allowing_open_registrations` flag set to `true`, a malicious actor may create a malicious vault.

We classify this issue as moderate severity due to the fact that CosmWasm deployment on Injective is currently permissioned.

Recommendation

Consider maintaining a list of whitelisted vault code IDs to prevent malicious vault registration to the master contract.

7. Canceling IDO fails due to insufficient funds

RISK IMPACT: MODERATE	STATUS: ACKNOWLEDGED

Revision Notes

The team mentions that `CancelIdo` cannot be called anymore after `StartIdo` was called. Funds are only sent during `StartIdo`.

Description

The `start_ido` function in `contracts/mito-launchpad/src/handle.rs:175-190` distributes the funds from the `LAUNCH_FEES_FOR_TREASURY` storage to the treasury address.

However, when canceling the IDO in the `cancel_ido` function, the amount in the `LAUNCH_FEES_FOR_TREASURY` storage will be attempted to refund to the project owner in `contracts/mito-launchpad/src/handle.rs:204-212`. As a result, the transaction will fail due to insufficient funds in the contract.

Recommendation

Consider setting the `LAUNCH_FEES_FOR_TREASURY` storage to zero if the amount in the `LAUNCH_FEES_FOR_TREASURY` storage is sent to the treasury address in the `start_ido` function.

8. Fetched Pyth price may be stale

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

The `get_quote_price` function in `contracts/mito-launchpad/src/utils.rs:163` queries the Pyth price and is used in `contracts/mito-launchpad/src/amm_subscriptions.rs:22` and `contracts/mito-launchpad/src/utils.rs:174` for calculations.

However, it does not validate that the fetched Pyth price is fresh within a specific period of time with high confidence intervals. As a result, stale prices may be used, leading to incorrect calculations.

Recommendation

Consider validating the fetched Pyth price is fresh with a high confidence interval.

9. `IS_ALLOWED_VAULT_REGISTRATION_ADDRESS` cannot be removed

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

The `add_allowed_vault_registration_address` function in `contracts/mito-master/src/handle.rs:55` allows the contract owner to set IS_ALLOWED_VAULT_REGISTRATION_ADDRESS status for an address.

However, there is no entry point for the contract owner to remove it. This means that whoever addresses that are set with `IS_ALLOWED_VAULT_REGISTRATION_ADDRESS` can always register vaults in `contracts/mito-master/src/handle.rs:103`.

For example, if the contract ownership is transferred to a new owner, the previous owner will always be able to register vaults due to `contracts/mito-master/src/handle.rs:55`.

Recommendation

Consider adding an entry point for the contract owner to remove `IS_ALLOWED_VAULT_REGISTRATION_ADDRESS`.

10. Two-step ownership transfer is not implemented

RISK IMPACT: LOW	**STATUS:** ACKNOWLEDGED

Description

The codebase does not implement two-step ownership transfer in the following code lines:

- `contracts/mito-launchpad-factory/src/handle.rs:69`
- `contracts/mito-launchpad/src/handle.rs:49`
- `contracts/mito-master/src/handle.rs:30`
- `packages/mito_shared/src/config.rs:68`

As a result, the ownership will be lost and cannot be recovered if transferred to an incorrect address that is not owned by anyone.

Recommendation

Consider implementing a two-step ownership transfer that proposes a new owner in the first step and requires the proposed owner to accept it as the second step.

11. `set_quote_price` differs from documentation

RISK IMPACT: LOW	STATUS: RESOLVED

Description

The `set_quote_price` function in `contracts/mito-launchpad/src/handle.rs:221` does not validate that the sender is the owner. This is defined in the README as a message that is only callable by the owner. While this currently does not appear to have any security implications, the logic does diverge from the project documentation.

Recommendation

Consider either updating the project documentation to correctly represent the authorized caller of the function or updating the function to ensure that the caller is only the owner.

12. Remove duplicate code block

RISK IMPACT: INFORMATIONAL	STATUS: RESOLVED

Description

The `update_config` function in `contracts/mito-launchpad-factory/src/handle.rs:77-79` is a repeated code of lines 71 to 73.

Recommendation

Consider removing the duplicate code block.

13. Remove debug messages

RISK IMPACT: INFORMATIONAL	STATUS: ACKNOWLEDGED

Description

In the `vault-cpmm-spot` contract, several debug messages are present:

- `contracts/vault-cpmm-spot/src/begin_blocker.rs:10`
- `contracts/vault-asmm-spot/src/contract.rs:207`
- `contracts/vault-cpmm-spot/src/mm_bot/bot.rs:39,44,79,104`
- `contracts/vault-cpmm-spot/src/mm_bot/order_management.rs:177,191,212`

It is best practice to remove debug messages before the contracts are deployed to mainnet.

Recommendation

Consider removing the debug messages mentioned above.

Document Control

Version	Date	Notes
-	29th January 2024	Security audit commencement date.
0.1	15th February 2024	Initial report with identified findings delivered.
0.5	6th March 2024	Fixes remediations implemented and reviewed.
1.0	7th March 2024	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