White Whale – Migaloo
Audit Report

Prepared for White Whale, 16th November 2022

Table of Contents

Introduction

SCV was engaged by White Whale 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/White-Whale-Defi-Platform/migaloo-core
- Code Freeze: *0b8673a6cb2e1662cf207c6e6b7e9294304ae96c*

Several remediations were performed and reviewed by SCV up to the following hash :

- Code Freeze: *587597a53bf8c0b9d17ec742522698177597edce*

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 White Whale. 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;
- Analyse 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

SCV used a scale from 0 to 10 that represents how *SUFFICIENT* or *NOT SUFFICIENT* each code criteria was during the assessment:

Criteria	Status	Notes
Provided Documentation	SUFFICIENT 🟢	N/A
Code Coverage Test	SUFFICIENT 🟢	N/A
Code Readability	SUFFICIENT 🟢	The codebase had good readability and utilised many Rust and CosmWasm best practices.
Code Complexity	SUFFICIENT 🟢	N/A

Vulnerabilities Summary

#	Summary Title	Risk Impact	Status
1	Deposit function incorrectly calculates share	Critical	Resolved
2	Protocol fees are not excluded when calculating total asset balance	Critical	Resolved
3	The first depositor can prevent small liquidity providers from joining the pool	Medium	Resolved
4	Pair contract update_config is unreachable	Medium	Resolved
5	Update_Config lacking parameter validation	Low	Resolved
6	Contracts should use two-step ownership transfer	Low	Acknowledged
7	Vault contract sets incorrect CONTRACT_NAME	Low	Resolved
8	Computing offer amount would fail if swap and pool fees sum over 100%	Low	Resolved
9	Spelling error of minimum_receive variable	Low	Resolved
10	Querying fees with duplicate contract addresses would return incorrect result	Informational	Resolved
11	Consider deduplicating fees for overall query fees functionality	Informational	Resolved
12	Default contract names are used	Informational	Resolved

Audit observations

- Different pair contracts can have different native decimal values due to the `AddNativeTokenDecimals` functionality allowing the admin to overwrite decimal values for existing native tokens. This might happen if a new pair contract with the native token is instantiated after the admin updates its corresponding decimal values.

- Any user can siphon any funds inside the vault router. The possibility of this happening should be low since all profit will be returned to the flash loan caller.

Detailed Vulnerabilities

1 – Deposit function incorrectly calculates share

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

Description

The `deposit` function in `migaloo-core/contracts/liquidity_hub/vault-network/vault/src/execute/deposit.rs:74` incorrectly mints one liquidity token per one deposited asset. This is problematic because it differs from how user share is calculated within the withdraw function in `migaloo-core/contracts/liquidity_hub/vault-network/vault/src/execute/receive/withdraw.rs:43` where the `withdraw_amount` based on a user share of the total supply. This effect is those earlier depositors will dilute their share as the contract continues to receive deposits because the total supply has grown.

Recommendations

We recommend minting an equal amount of shares if the total supply is zero. If not, consider calculating a user's share based on the following formula:

```
liquidity pool token to mint = native amount sent / (total balance
in the contract - protocol fee - native sent amount) * total lp
token supply
```

2 – Protocol fees are not excluded when calculating total asset balance

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

Description

In the vault and pair contract, both contracts charge a protocol fee in addition to the swap and vault fees. The protocol fees are temporarily left in the contract until someone calls `CollectProtocolFees` to transfer the protocol fees. This means that, unlike the swap and vault fees, the protocol fees should not be calculated as part of the contract's balance.

However, this situation is only correctly handled during the pair's withdrawal function, as seen in `migaloo-core/contracts/liquidity_hub/pool-network/terraswap_pair/src/commands.rs:242`. Ideally, all computations that use the current contract's balance should deduct the protocol fees first before using the value for calculations.

As a result, the swap contract would have an inflated k value, while the queries in the vault contract would return a false result.

Affected contracts and functionalities:

- Pair contract
    - `provide_liquidity`
    - `swap`
    - `query_pool`
    - `query_simulation`
    - `query_reverse_simulation`
- Vault contract
    - `withdraw`
    - `get_share`

Recommendations

We recommend deducting protocol fee from the contract balance to prevent miscalculations.

3 – The first depositor can prevent small liquidity providers from joining the pool

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

Description

In the `provide_liquidity` function of the pair contract, there are no preventive measures to prevent the first liquidity provider from minting a minimal amount of LP tokens. Consequently, an attacker can prevent small liquidity providers from joining the pool.

Specifically, the attacker can provide liquidity until it returns exactly one liquidity pool token to the attacker. After that, an attacker can transfer many assets to the pool to cause the final shares minted to be 0. This is possible because the formula in `migaloo-core/contracts/liquidity_hub/pool-network/terraswap_pair/src/commands.rs:170-173` would return 0 if the multiplied value of total share and deposit is lower than the pool's asset balance due to division and integer rounding.

If a small liquidity provider comes in and decides to supply a reasonable amount of liquidity to the pool, they would likely return an error in line `178`.

Recommendations

We recommend forcing the first liquidity provider to mint a minimum amount of liquidity tokens and transfer them to the contract itself. For reference, here is Astroport's implementation of `provide_liquidity`.

4 – Pair contract update_config is unreachable

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

Description

The `update_config` function in `migaloo-core/contracts/liquidity_hub/pool-network/terraswap_pair/src/commands.rs:392` is unreachable. This is because the pair contract's owner is the factory contract, and the factory contract does not include any functionality to call this entry point.

Recommendations

We recommend adding functionality to the factory contract that allows it to perform pair contract updates.

5 – Update_Config lacking parameter validation

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

Description

The `update_config` function in `migaloo-core/contracts/liquidity_hub/pool-network/terraswap_pair/src/commands.rs:392` does not properly validate `pool_fees`. This may allow the admin to introduce a misconfiguration potentially. This is also present in `migaloo-core/contracts/liquidity_hub/vault-network/vault_factory/src/execute/update_vault_config.rs:10`.

Recommendations

We recommend validating each `Fee` with the `Fee is_valid` function to ensure that the value is within an acceptable range.

6 – 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 it requires the new admin to accept ownership before the transfer and config changes occur.

Recommendations

We recommend 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.

7 – Vault contract sets incorrect CONTRACT_NAME

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

Description

The vault contract's `CONTRACT_NAME` is being set in `migaloo-core/contracts/liquidity_hub/vault-network/vault/src/contract.rs:21` to `"vault_factory"`. This will result in the vault contract having the same name as the White Whale Vault Factory Contract.

Recommendations

We recommend updating the `CONTRACT_NAME` for the vault contract.

8 – Computing offer amount would fail if swap and pool fees sum over 100%

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

Description

In `migaloo-core/contracts/liquidity_hub/pool-network/terraswap_pair/src/helpers.rs:66`, the total swap and pool fees are added together and deducted by `Decimal256::one()`. If both values exceed 100%, an overflow panic will occur. This is possible as the fee validations ensure that both fees do not exceed 200%.

Recommendations

We recommend implementing additional validation to ensure the sum of both fees does not exceed 100%.

9 – Spelling error of minimum_receive variable

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

Description

In `migaloo-core/contracts/liquidity_hub/pool-network/terraswap_router/src/contract.rs:184`, the `minium_receive` variable contains a spelling error which should be `minimum_receive`.

Recommendations

We recommend correcting the spelling error mentioned above.

10 – Querying fees with duplicate contract addresses would return incorrect result

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

Description

The *`query_fees`* function in *`migaloo-core/contracts/liquidity_hub/fee_collector/src/queries.rs:27-44`* does not filter duplicate contracts when querying the fees for a pool or vault. As a result, the returned asset vector would include the duplicated fee amount. This is problematic because the result will differ when performing `CollectFees` execution with the same parameters.

Recommendations

Consider deduplicating contract addresses in `migaloo-core/contracts/liquidity_hub/fee_collector/src/queries.rs:28`.

11 – Consider deduplicating fees for overall query fees functionality

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

Description

In `migaloo-core/contracts/liquidity_hub/fee_collector/src/queries.rs:127-136`, the fees are only accumulated and deduplicated if the query is specific for the pool factory. As a result, other queries that return the same type of asset fees in a vector would not be deduped and accumulated together. For example, performing a manual `QueryFeesFor::Contracts` query with the same parameters as `terraswap::factory::QueryMsg::Pairs` would cause duplicate asset fees inside the returned vector.

Recommendations

Instead of only deduping and accumulating fees for the pool factory, consider implementing it for the overall `query_fees` functionality in `migaloo-core/contracts/liquidity_hub/fee_collector/src/queries.rs:56`.

12 – Default contract names are used

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

Description

In the pool network folder, several contracts that are based on Terraswap's contracts uses the default contract names:

- `migaloo-core/contracts/liquidity_hub/pool-network/terraswap_factory/src/contract.rs:21`
- `migaloo-core/contracts/liquidity_hub/pool-network/terraswap_pair/src/contract.rs:24`
- `migaloo-core/contracts/liquidity_hub/pool-network/terraswap_router/src/contract.rs:24`
- `migaloo-core/contracts/liquidity_hub/pool-network/terraswap_token/src/contract.rs:13`

Recommendations

Consider modifying the contract name to include specific branding instead of using Terraswap's default contract name.

Document control

Version	Date	Approved by	Changes
0.1	01/11/2022	Vinicius Marino	Document Pre-Release
0.2	-	SCV Team	Remediation Revisions
1.0	16/11/2022	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

The content of this audit report is provided “As is”, without representations and warranties of any kind.

The author and their employer disclaim any liability for damage arising out of, or in connection with, this audit report.

Copyright of this report remains with the author.