Kujira - BlackWhale Contracts - Audit Report

Prepared for BlackWhale, 19 September 2022

Table of Contents

Introduction

SCV was engaged by BlackWhale 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/blackwhale2/black-whale-contract
- Code Freeze: *b248f15aa7b8dc8de29c8065a2ed046322aee15c*

Remediations were applied into several commits up to the following hash commit:

- Code Freeze *a45f14927fc2038f5ac1440ef8a60d5041ac2775*

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 BlackWhale. 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(6-10)** or **NOT SUFFICIENT(0-5)** each code criteria was during the assessment:

Criteria	Status	Scale Range	Notes
Provided Documentation	**Sufficient**	7-8	N/A
Code Coverage Test	**Sufficient**	7-8	N/A
Code Readability	**Sufficient**	6-8	N/A
Code Complexity	**Sufficient**	7-8	N/A

Vulnerabilities Summary

	Title and Summary	Risk	Status
1	CW20 tokens are not validated when providing liquidity	**High**	**Remediated**
2	Fees are incorrectly charged when providing liquidity for the first time	**Medium**	**Remediated**
3	Asset information is not deduped and validated during contract instantiation	**Low**	**Acknowledged**
4	Consider halting the execution if the bot submitted an incorrect denom order	**Low**	**Remediated**
5	Consider verifying the fee percentage to be within the appropriate value	**Low**	**Remediated**
6	Extra funds sent by the user are lost in the contract	**Low**	**Acknowledged**
7	Lack of validation during contract instantiation and update can lead to misconfigurations	**Low**	**Acknowledged**
8	Consider changing the Token query message name into TotalSupply	**Informational**	**Remediated**
9	Consider enforcing denom validation when bots submit an order	**Informational**	**Acknowledged**
10	Consider refactoring the contract administrator into a single storage state	**Informational**	**Acknowledged**
11	Consider retrieving the sent amount directly from sent funds when withdrawing liquidity	**Informational**	**Acknowledged**
12	General code inefficiencies found in the codebase	**Informational**	**Remediated**

Detailed Vulnerabilities

1. CW20 tokens are not validated when providing liquidity

Likelihood	Impact	Risk
Possible	Severe	**High**

Description

When providing liquidity to the contract, only native assets are checked in `contracts/vault/src/contract.rs:172` to ensure the user sent native funds as provided in the `assets` vector. As seen in `packages/astroport/src/asset.rs:126-128`, the `assert_sent_native_token_balance` only validates the user sent the expected amount of native funds to the contract but skips the check if the asset is a token address.

As a result, if the contract decides to support any token address, users can provide “fake” liquidity to the pool without having the tokens. This can be further exploited to mint a large amount of liquidity pool tokens and use it to steal available funds inside the pool via `withdraw_liquidity`.

Recommendations

Consider executing a `TransferFrom` message when the contract expects the user to provide liquidity in CW20 tokens. An example from Astroport can be found in https://github.com/astroport-fi/astroport-core/blob/b47f5ad5fa546b0e176401181fb31c5ca0412d37/contracts/pair/src/contract.rs#L363-L373. If there is no plan to support CW20 tokens for now, consider reverting an error if any of the `msg.asset_infos` provided is a token address.

2. Fees are incorrectly charged when providing liquidity for the first time

Likelihood	Impact	Risk
Possible	Moderate	**Medium**

Description

When providing liquidity for the first time in `contracts/vault/src/contract.rs:220-237`, the total supply is increased first via `mint_liquidity_token_message` before processing the fees via `get_split_fee`. This would cause the protocol to charge a maintenance fee even though there are no tokens supplied beforehand, causing lesser profit for the users.

Please see the https://gist.github.com/scvsecurity/e49277c5989a6e3407a80449328e5df2 test case to reproduce the vulnerability.

Recommendations

Consider moving the code in lines 220-224 below line 237 to correctly process the maintenance fee before increasing the total supply.

3. Asset information is not deduped and validated during contract instantiation

Likelihood	Impact	Risk
Possible	Low	**Low**

Description

In the contracts/vault/src/contract.rs:44, the msg.asset_infos parameter is not validated to ensure the assets are not duplicated and the CW20 token address is valid.

As a result, having the same native asset in a pool allows the user to deposit funds that will be credited twice to the pool due to contracts/vault/src/contract.rs:180-191. An invalid CW20 token address configured would cause the provide_liquidity functionality to fail in contracts/vault/src/contract.rs:168-169 and the withdraw_liquidity functionality to fail in contracts/vault/src/contract.rs:331.

Recommendations

Consider using Astroport's functionality to prevent duplicate pool assets and invalid assets instantiated.

- https://github.com/astroport-fi/astroport-core/blob/b47f5ad5fa546b0e176401181fb31c5ca0412d37/contracts/fa
  L133

4. Consider halting the execution if the bot submitted an incorrect denom order

Likelihood	Impact	Risk
Unlikely	Low	**Low**

Description

When a bot submits an order in `contracts/vault/src/fin.rs:33-43`, if the denom submitted by the bot does not equal to the base or quote denom from the `BookResponse`, the execution will still continue without verifying the submitted price.

As the source code for Kujira Fin is not public at the date of writing, we are unable to determine the impact of this. However, due to the possibility of bypassing sensitive price checks, we decided to set the impact to low.

Recommendations

Consider adding an else statement in line 43 to return an error if the bot specified denom does not equal the expected base and quote offer denom.

5. Consider verifying the fee percentage to be within the appropriate value

Likelihood	Impact	Risk
Possible	Low	**Low**

Description

When updating the configuration, no validations are being performed on the `fee_percentage` parameter to ensure the fee percentage is not over 100%.

Affected code line:

- `contracts/vault/src/contract.rs:454`

As a result, a misconfiguration would cause the protocol to charge too many fees from the user, rendering users to have little to no profit.

Recommendations

Consider validating the `fee_percentage` decimal value is below or equal to `Decimal::one()`.

6. Extra funds sent by the user are lost in the contract

Likelihood	Impact	Risk
Unlikely	Low	**Low**

Description

In several instances of the codebase, there is no check the length of `info.funds` sent by the user does not exceed the expected fund's length. If a user sent more funds than intended, the excess funds would be stuck in the contract.

Affected code lines:

- `contracts/vault/src/contract.rs:172` (`provide_liquidity`)
- `contracts/vault/src/contract.rs:303` (`withdraw_liquidity`)

Recommendations

Consider modifying the `provide_liquidity` functionality to check the length of `info.funds` equals the number of native assets provided in the `assets` vector. As for `withdraw_liquidity`, consider verifying the expected length of `info.funds` sent by the user is equal to one.

7. Lack of validation during contract instantiation and update can lead to misconfigurations

Likelihood	Impact	Risk
Possible	Low	**Low**

Description

When performing contract instantiation and updating the configuration, no validations are performed on the message parameters that ensure the address supplied is valid.

Affected code lines and variables:

- `contracts/vault/src/contract.rs:36, 57` (`msg.owner_address`)
- `contracts/vault/src/contract.rs:39` (`msg.fin_address`)
- `contracts/vault/src/contract.rs:459` (`owner_address`)

Recommendations

Consider validating the `Addr` string before saving it into storage for the addresses mentioned above.

8. Consider changing the Token query message name into TotalSupply

Likelihood	Impact	Risk
Unlikely	Informational	**Informational**

Description

In `contracts/vault/src/contract.rs:490-495`, the `QueryMsg::Token` returns the total supply of minted tokens in the contract. As the `Token` query message name normally represents the token information, the name itself might not be intuitive and might confuse users since the returned response is the token's total supply.

Recommendations

Consider changing the `QueryMsg::Token` query name into `QueryMsg::TotalSupply`.

9. Consider enforcing denom validation when bots submit an order

Likelihood	Impact	Risk
Rare	Informational	**Informational**

Description

In `contracts/vault/src/fin.rs:17`, authorized bots can submit an order with the `Denom` enum as native denoms or CW20 token addresses. As the denom variable is used in line 22 to query the denom's balance via the `BankQuery` module, this means that only native denoms orders can be submitted.

If a bot tries to submit an order with the `Denom` enum as CW20 token address, the error returned is "Exceed vault asset balance" which won't hint to the bot operator that they used an incorrect `Denom` enum.

Recommendations

Consider validating the bot submitted valid `Denom` enum by checking with `denom_is_native` functionality in `packages/blackwhale/src/denom_utils.rs:11`. If the submitted `Denom` enum is a CW20 token address, return an appropriate error message such as "CW20 token is not supported".

10. Consider refactoring the contract administrator into a single storage state

Likelihood	Impact	Risk
Possible	Informational	**Informational**

Description

The current contract configuration holds the same admin address in two storage states, the `Config` and `Admin` struct. As privileged functions only check the caller is the intended owner from the `Config` struct, the admin address from the `Admin` storage state is left unused.

Recommendations

Consider removing the `Admin` storage state from the contract and the associated `SetAdmin` function to improve the readability and maintainability of the codebase.

11. Consider retrieving the sent amount directly from sent funds when withdrawing liquidity

Likelihood	Impact	Risk
Unlikely	Informational	**Informational**

Description

In `contracts/vault/src/contract.rs:303-309`, the `withdraw_liquidity` functionality attempts to verify the amount of receipt denom sent by the user equals the provided `amount` argument in line 298. This can be further refactored into removing the need for the user to provide the `amount` argument and retrieving it directly from funds the user sends instead.

Recommendations

Consider removing the `amount` argument and directly use the `info.funds[0].amount` as sent amount instead.

12. General code inefficiencies found in the codebase

Likelihood	Impact	Risk
Unlikely	Informational	**Informational**

Description

In several instances of the codebase, there exists code that is either not used or not needed as the result of the execution is still the same. As a result, this introduced inefficiencies such as unnecessary gas consumption and reduced maintainability and readability.

Affected code lines:

- `contracts/vault/src/state.rs:27-33`
- `contracts/vault/src/contract.rs:239`

Additionally, some match statements in the codebase prints the error to the console instead of halting the execution. This would cause an invalid execution to succeed even though there is an alarming error in the codebase, which might cause unintended consequences in the future.

Code lines	Suggested modification
`contracts/vault/src/contract.rs:142-145`	`FEE.save(storage, &fee).unwrap();`
`contracts/vault/src/contract.rs:152-155`	`FEE.save(storage, &fee).unwrap();`
`contracts/vault/src/contract.rs:275-278`	`add_supply(storage, amount).unwrap();`
`contracts/vault/src/contract.rs:357-360`	`remove_supply(deps.storage, amount).unwrap();`

Finally, it is possible to skip the if else statement in `contracts/vault/src/fin.rs:66-69` since the `RetractOrder` functionality is able to handle the `amount` variable type as `Option<Uint256>`.

Recommendations

- Removing inefficient code
- Modify the match statements above to the suggested modifications
- Modify the code in `contracts/vault/src/fin.rs:66-69` into `msg = to_binary(&RetractOrder { order_idx, amount })?;`

Document control

Document changes

Version	Date	Name	Changes
0.1	2022-09-16	Vinicius Marino	Initial report
0.2	2022-09-17	Vinicius Marino	Team communication and Pre-Release
1.0	2022-09-19	Vinicius Marino	Revisions and Document Release

Document contributors

Name	Role	Email address
Vinicius Marino	Security Specialist	vini@scv.services

Appendices

Appendix A: 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.

Appendix B: 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:

Likelihood / Impact	Rare	Unlikely	Possible	Likely
Critical	**Medium**	**High**	**Critical**	**Critical**
Severe	**Low**	**Medium**	**High**	**High**
Moderate	**Low**	**Medium**	**Medium**	**High**
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.