Steak & Fee Split Audit Report

Prepared for PFC, 8th May 2023

Table of Contents

Introduction

SCV was engaged by PFC 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/PFC-Validator/PFC-fee-split/tree/0.2.9/contracts/pfc-vault-contract
    - Code Freeze: `38109c99754562fc961d2c189855f139e66b1136`
- https://github.com/PFC-Validator/PFC-fee-split/tree/0.2.9/contracts/pfc-astroport-generator
    - Code Freeze: `9cc64147ff8ee6ac4e4d322f5501513c222ff404`
- https://github.com/PFC-Validator/PFC-fee-split/tree/main
    - Code Freeze: `b3a9eb2fa5d5cbac7d5e7ec27868ac7ee5ae1da9`
- https://github.com/PFC-developer/steak-contracts/tree/v3.0.3-cargo
    - Code Freeze: `9cc64147ff8ee6ac4e4d322f5501513c222ff404`

Revisions were performed by SCV on the up to the following:

- https://github.com/PFC-Validator/PFC-fee-split
    - Code Freeze: `18a2ca47dbd4cacd640b5f5a48a1fb04087e472a`

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 PFC. 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

This section below 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**	While the current coverage mark is at 48.04% (565 lines out 1176). SCV recommends increasing the testing coverage to at least 70% using cw-multi-test rather than mock-testing.
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	Updating lp token can cause inconsistencies	**Severe**	**Resolved**
2	`do_deposit` can fail	**Severe**	**Resolved**
3	`SendType` address not validated in `execute_add_allocation_detail`	**Medium**	**Resolved**
4	`Send_after` not validated in `execute_add_allocation_detail`	**Low**	**Resolved**
5	Updating token can orphan some rewards	**Low**	**Resolved**
6	`Migrate_reward` function missing validation to ensure contract has enough rewards to migrate	**Low**	**Acknowledged**
7	Potential future unbounded iteration over reward tokens	**Informational**	**Acknowledged**
8	No limit set for `change_gov_contract_by_height`	**Informational**	**Acknowledged**
9	Emit event upon unbonding	**Informational**	**Resolved**
10	Account address not validated	**Informational**	**Resolved**
11	Multiple references to VKR in the astroport generator contract	**Informational**	**Acknowledged**
12	Remove unused code	**Informational**	**Resolved**
13	Unused reply id on steak hub contract	**Informational**	**Acknowledged**
14	Reuse duplicated code	**Informational**	**Acknowledged**

Detailed Vulnerabilities

1 – Updating lp token can cause inconsistencies

Risk Impact: Severe - **Status**: Resolved

Description

The vault contract's `update_config` function in `contracts/pfc-vault-contract/src/executions.rs:216` allows the admin to update the value of `config.lp_token`, even if there are existing bonds using the old asset. This is problematic because if the `lp_token` address were to be updated after existing deposits were made it would cause inconsistencies in user's share value.

For example, if Alice had bonded 100 of an lp token with an equivalent value of $100 USD, then the Admin changed the lp token and Bob bonded 100,000 of the new lp token also with an equivalent value of $100, Bob stake would significantly dilute the amount of share Alice has in the vault. This is just one example, but there are many inconsistent states that would arise if this value was updated after bonds were made. `Staker_info.bond_amount` is a simple Uint128 value that does not differentiate between the different bonded tokens so it would directly affect any functionality that deals with the claims.

Recommendations

We recommend removing `config.lp_token` as an updatable parameter in the update_config function.

2 – `do_deposit` can fail

Risk Impact: Severe - **Status**: Resolved

Description

The `do_deposit` function in `contracts/pfc-fee-splitter/src/handler/exec.rs:417` can potentially fail in the event that a small allocation exists and a flush deposit is specified with a small amount of input funds. This can be caused if the `determine_allocation` function returns a coin / coins with a zero amount *(See test case provided here)* . This is possible because in `contracts/pfc-fee-splitter/src/handler/exec.rs:363` if the portion is zero or sufficiently small, the coin will be set to zero.

If the transaction is a flush deposit, the `SendType::SteakRewards` will fail which will result in the entire flush being blocked because an attempt to pass a bond message to the steak hub will error. For a wallet send type, the bank message will not fail because that can be sent with a 0 amount.

Recommendations

In `determine_allocation`, funds should not be added to `send_coins` if they have a 0 amount.

3 – `SendType` address not validated in `execute_add_allocation_detail`

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

Description

In the `execute_add_allocation_detail` function in `contracts/pfc-fee-splitter/src/handler/exec.rs:69`, the `SendType`'s verify function does not validate the address specified by the caller. Currently, the verify function only confirms that the receiver address specified is not the contract address to avoid recursion. This issue is also present in `execute_modify_allocation_detail` which also does not verify the address specified within the `SendType`.

If an improperly configured allocation was added, it would highly impact the contract's functionality. If this is saved as an invalid address it will cause an error to be thrown in the `generate_cosmos_msg` function. This will have a high impact because it will cause the `do_deposit` function to error when a message is sent to the invalid allocation. Additionally this would impact `execute_reconcile`.

We classify this finding as a medium severity because while it would have a high impact, only the contract's admin can configure an invalid allocation.

Recommendations

We recommend validating the address specified in `contracts/pfc-fee-splitter/src/handler/exec.rs:69`.

4 – `Send_after` not validated in `execute_add_allocation_detail`

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

Description

In the `execute_add_allocation_detail` function in `contracts/pfc-fee-splitter/src/handler/exec.rs:57` the `send_after` parameter is not validated before being saved in `ALLOCATION_HOLDINGS`. This validation is inconsistent from the validation performed in the `instantiate` function in `contracts/pfc-fee-splitter/src/contract.rs:52`.

Recommendations

We recommend performing consistent validation of the `send_after` parameter.

5 – Updating token can orphan some rewards

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

Description

In the vault contract's `update_config` function in `contracts/pfc-vault-contract/src/executions.rs:207`, the admin can update `config.token`. If `config.token` is updated while there are existing rewards then the `MigrateReward` message will fail when called by the admin as it will specify the config value of the new token and will not be able to transfer the previous token.

Recommendations

We recommend disallowing any update of `config.token` if there are existing rewards for that asset.

6 – `Migrate_reward` function missing validation to ensure contract has enough rewards to migrate

Risk Impact: Low - **Status**: Acknowledged

Description

In the vault contract's `migrate_reward` function in `contracts/pfc-vault-contract/src/executions.rs:249`, the rewards being migrated in the `amount` variable are unchecked. This means that there is no validation to ensure that the contract has enough rewards to perform the migrate rewards action. This can lead to a situation where the contract tries to transfer more rewards than it actually has, resulting in an error or failed transaction.

Recommendations

We recommend checking the contract's reward balance and validating the `amount` variable before executing the transfer.

7 – Potential future unbounded iteration over reward tokens

Risk Impact: Informational - **Status**: Acknowledged

Description

The `recv_reward_token` function in `contracts/pfc-vault-contract/src/executions.rs:119` receives `config.token` and adds it to `TOTAL_REWARDS`. Currently, only `config.token` is supported. There is a comment in `contracts/pfc-vault-contract/src/entrypoints.rs:90` that specifies future support for multiple CW20 tokens.

This could introduce an opportunity for an attacker to create many CW20 tokens, and send them to the vault contract. This would then add many `TokenBalance` to the `TOTAL_REWARDS` which could cause out of gas errors in `get_current_claims` when they are all iterated over in `contracts/pfc-vault-contract/src/executions.rs:375`.

Recommendations

While this is not currently an issue, we advise caution in the future when deciding on how to support multiple reward tokens. A straightforward approach could be to create a reward token whitelist.

8 – No limit set for `change_gov_contract_by_height`

Risk Impact: Informational - **Status**: Acknowledged

Description

When updating the `gov_contract` in `execute_update_gov_contract` in `contracts/pfc-vault-contract/src/executions.rs:268`, the `blocks` variable in `change_gov_contract_by_height` is not validated. There is no time limit to how long the proposal of updating the gov contract is active or becomes active before accepting.

Recommendations

We recommend limiting the maximum allowed value for the `change_gov_contract_by_height` variable.

9 – Emit event upon unbonding

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

Description

When the `unbond` function is invoked in `contracts/pfc-vault-contract/src/executions.rs:112`, the function does not emit the “`action`” attribute “`unbond`” upon successful function call.

Recommendations

We recommend emitting relevant events or attributes based on configured parameters.

10 – Account address not validated

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

Description

When sending rewards in the `send_rewards` function in `contracts/pfc-astroport-generator/src/contract.rs:153`, the `account` address variable is not validated.

Recommendations

We recommend including a validation check for the recipient address to ensure that the address is valid before executing the transfer.

11 – Multiple references to `VKR` in the astroport generator contract

Risk Impact: Informational - **Status**: Acknowledged

Description

The astroport generator contract contains several references to `Valkyrie` and the `VKR` token.

Recommendations

Consider making the proper changes to the code comments and function docs to remove those references.

12 – Remove unused code

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

Description

Multiple instances of unused code such as debug statements can be found throughout the codebase, which impacts the readability. A few notable examples to this are listed as below:

- `contracts/pfc-vault-contract/src/executions.rs:369-374`
- `contracts/pfc-vault-contract/src/executions.rs:418-425`

Recommendations

Consider removing the unused code.

13 – Unused reply id on steak hub contract

Risk Impact: Informational - **Status**: Acknowledged

Description

When creating the steak token as a cw20 token on the `steak hub`, the `REPLY_INSTANTIATE_TOKEN` reply id is used in the `SubMsg`. However, when matching the reply id in the `reply` entry point, the constant is not being used but the equivalent value `1`.

Recommendations

We recommend using the defined constant `REPLY_INSTANTIATE_TOKEN` when matching the reply id to avoid potential errors in the future.

14 – Reuse duplicated code

Risk Impact: Informational - **Status**: Acknowledged

Description

Both the `steak hub` and the `steak hub token factory` contracts use the exact same copies of `math.rs`, which can lead to divergence between copies in the future as the contracts evolve, causing potential bugs.

Recommendations

We recommend moving the `math.rs` file into the `steak` package and reuse it in both contracts.

Document control

Version	Date	Approved by	Changes
0.1	03/05/2023	Vinicius Marino	Document Pre-Release
0.2	06/05/2023	SCV Team	Remediation Revisions
1.0	08/05/2023	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