AUDIT REPORT

Choice Exchange

Choice Contracts

Prepared by SCV-Security

On 29th April 2025

Table of Contents

Introduction

SCV has been engaged by Choice Exchange 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

The scope for this report includes the following:

- `choice_factory`: This contract allows users to create `choice_pair` contracts as AMM pools and records their address in the state.
- `choice_pair`: This contract facilitates liquidity pools and swaps between two assets in Injective. It is responsible for handling liquidity provision, swaps, and queries while integrating with Injective-specific modules.
- `choice_router`: This contract facilitates multi-hop swap operations across several `choice_pair` contracts.
- `choice_send_to_auction`: This contract facilitates sending CW20 and native tokens to the Injective burn auction subaccount by integrating with Injective's CW20 adapter contract.

Submitted Codebase

Choice Exchange	
Repository	https://github.com/choice-exchange/choice_exchange
Commit	`54f844c7bf5c33c37904109f7ec64b26138b6fe2`
Branch	`main`

Revisions Codebase

Choice Exchange	
Repository	https://github.com/choice-exchange/choice_exchange
Commit	`9654ed160cab0d7ce53713c7081a29999430e6b3`
Branch	`audit-fixes`

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 Choice Exchange . 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 are provided in the `README` files.
Coverage	**SUFFICIENT**	`cargo llvm-cov` reports a line coverage of 94.40% (8276/8767).
Readability	**SUFFICIENT**	The codebase had good readability overall and utilised many Rust and CosmWasm best practices.
Complexity	**SUFFICIENT**	N/A

Findings Summary

Summary Title	Risk Impact	Status
Funds sent for verifying native token decimals can be stolen	MODERATE	RESOLVED
Incorrect refund asset event emitted for CW20 tokens	LOW	RESOLVED
Addresses and subaccounts are not validated	LOW	RESOLVED
Potential incorrect `lp_amount` value emitted	INFO	RESOLVED
Two-step ownership transfer is not implemented	INFO	RESOLVED
Misleading error message	INFO	RESOLVED
Duplicate message interface	INFO	RESOLVED
Miscellaneous code quality improvements	INFO	RESOLVED

Findings Technical Details

1. Funds sent for verifying native token decimals can be stolen

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

The `execute_add_native_token_decimals` function in `contracts/choice_factory/src/contract.rs:244-249` attempts to record native tokens' decimal values into the `ALLOW_NATIVE_TOKENS` state after verifying its existence by checking the contract's balance. This means that the caller must send native tokens to the contract first before calling the `execute_add_native_token_decimals` function.

The issue is that once the funds are sent to the contract, there is no entry point to withdraw them. This is problematic because it allows malicious actors to steal funds by specifying the native token amount in the `CreatePair` message, causing them to be sent to the `choice_pair` contract in `contracts/choice_factory/src/contract.rs:349`. After that, the actor will call `WithdrawLiquidity` with the LP tokens minted in `contracts/choice_factory/src/contract.rs:378`, eventually receiving the funds sent for the `execute_add_native_token_decimals` function.

Recommendation

Consider implementing a refund mechanism for the contract owner to withdraw the native tokens.

2. Incorrect refund asset event emitted for CW20 tokens

RISK IMPACT: LOW	STATUS: RESOLVED

Description

The `provide_liquidity` function in `contracts/choice_pair/src/contract.rs:366` emits the "*refund_assets*" action to indicate the funds are refunded to the user when they have provided excess liquidity.

However, the calculation in `contracts/choice_pair/src/contract.rs:301-325` incorrectly computes the refund amount if the sent asset is a CW20 token. For native tokens, funds are sent along with the transaction, and if an excess amount is sent, the remaining amount is refunded to the user in `contracts/choice_pair/src/contract.rs:327-333`. This scenario is different for CW20 tokens, as the exact amount is taken directly with the `Cw20ExecuteMsg::TransferFrom` message in `contracts/choice_pair/src/contract.rs:334-344`, which means sending excess amounts of CW20 tokens is not possible.

As a result, the `refund_assets` variable will indicate that the contract has refunded excess CW20 tokens to the user, which is incorrect.

Recommendation

Consider setting the `remain_amount` to zero in `contracts/choice_pair/src/contract.rs:324` if the asset is a CW20 token.

3. Addresses and subaccounts are not validated

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In a few instances of the codebase, validations are not performed on the addresses and subaccounts:

- `msg.admin` and `msg.adapter_contract` should be validated as valid addresses with the `deps.api.addr_validate` function in `contracts/choice_send_to_auction/src/contract.rs:41-42`.
- `admin` should be validated as a valid address with the `deps.api.addr_validate` function in `contracts/choice_send_to_auction/src/contract.rs:137`.
- `msg.burn_auction_subaccount` can be validated with `SubaccountId::new`, similar to `contracts/choice_send_to_auction/src/contract.rs:29`.

Recommendation

Consider adding validation to the instances above.

4. Potential incorrect `lp_amount` value emitted

RISK IMPACT: INFORMATIONAL	STATUS: RESOLVED

Description

In `contracts/choice_pair/src/contract.rs:518`, the `lp_amount` variable represents the fee amount for liquidity providers after deducting the `fee_wallet_amount` and `burn_amount`, which are distributed in `contracts/choice_pair/src/contract.rs:528-571`.

The issue is that the `lp_amount` variable is currently computed as ⅔ of the `total_fee` directly, which will be truncated due to integer rounding. As a result, the "*pool_amount*" key will emit an incorrect value in line 587.

Recommendation

Consider computing the `lp_amount` variable as `total_fee - fee_wallet_amount - burn_amount`.

5. Two-step ownership transfer is not implemented

RISK IMPACT: INFORMATIONAL	STATUS: RESOLVED

Description

The codebase does not implement two-step ownership transfer in `contracts/choice_factory/src/contract.rs:97` and `contracts/choice_send_to_auction/src/contract.rs:137`.

Using a two-step ownership transfer mechanism helps provide a window of opportunity for the current owner to cancel the transfer if they did not intend to initiate it or if any unintended actions occurred.

As a result, ownership will be lost and cannot be recovered if transferred to an incorrect address that belongs to no one.

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.

6. Misleading error message

RISK IMPACT: INFORMATIONAL	STATUS: RESOLVED

Description

The `execute_add_native_token_decimals` function in `contracts/choice_factory/src/contract.rs:233` returns an error of "*unauthorized: sender does not match owner in denom*" if the caller is not the denom owner or the contract owner. This is misleading because the error message does not include the scenario where the caller is not the contract owner.

Recommendation

Consider modifying the error message to include the scenario where the caller is not the contract owner, such as "*unauthorized: sender does not match owner in denom and contract owner*".

7. Duplicate message interface

RISK IMPACT: INFORMATIONAL	STATUS: RESOLVED

Description

The `SendNative` message is implemented in the `BurnManagerMsg` enum in `contracts/choice_pair/src/contract.rs:442`. However, this message interface is incomplete and should use the correct interface defined in `packages/choice/src/send_to_auction.rs:16-20`.

Recommendation

Consider removing the `BurnManagerMsg` enum and importing the `ExecuteMsg` enum from `packages/choice/src/send_to_auction.rs:16-20`.

8. Miscellaneous code quality improvements

RISK IMPACT: INFORMATIONAL	STATUS: RESOLVED

Description

The following code line illustrates instances where code quality can be improved:

- The `burn_handler_address` variable can be declared outside the if-else statement in `contracts/choice_pair/src/contract.rs:536` and `contracts/choice_pair/src/contract.rs:549` to prevent duplicate code.
- The `optional_addr_validate` validation for the `to` parameter in `contracts/choice_router/src/contract.rs:85` is not required because the validation is already performed in the `ExecuteSwapOperations` entry points (`contracts/choice_router/src/contract.rs:70` and `contracts/choice_router/src/contract.rs:135`).
- A `break` statement can be added in the `for` loop in `contracts/choice_pair/src/contract.rs:171-177` when `authorized = true` to reduce gas consumption.
- A typo of `minium_receive` is found in the `assert_minimum_receive` function in `contracts/choice_router/src/contract.rs:206`. It should be modified into `minimum_receive`.
- The `add_message` function should be used instead of `add_messages` in `contracts/choice_router/src/operations.rs:68`, as the number of messages to dispatch is only one.
- The `subaccount_id` variable can be declared outside the if-else statement in `contracts/choice_send_to_auction/src/contract.rs:185` and `contracts/choice_send_to_auction/src/contract.rs:221` to prevent duplicate code.

Recommendation

Consider applying the recommendations above.

Document Control

Version	Date	Notes
-	30th March 2025	Security audit commencement date.
0.1	25th April 2025	Initial report with identified findings delivered.
0.5	26th - 29th April 2025	Fixes remediations implemented and reviewed.
1.0	29th April 2025	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