# AUDIT REPORT

## Dojo Trading

## DojoSwap Launchpad

Prepared by SCV-Security

On 30th January 2024

# Table of Contents

# Introduction

SCV has been engaged by Dojo Trading 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 DojoSwap protocol is a decentralised finance platform that implements automated money-market trading, a reward staking mechanism, and a launchpad contract for token sales.

Note that several modifications are made to the forked contracts to match the protocol architecture and tokenomics design.

The following contracts are submitted for audit:

- `launchpad`: This contract enables projects to raise funds by offering CW20 tokens in exchange for native tokens.

## Submitted Codebase

| dojoswap-contracts | |
|---|---|
| **Repository** | https://github.com/dojo-trading/dojoswap-contracts/tree/main/contracts/launchpad |
| **Commit** | `80f033804478a8ef092a457441d26c22f36dbd4e` |
| **Branch** | `main` |

## Revision Codebase

| dojoswap-contracts | |
|---|---|
| **Repository** | https://github.com/dojo-trading/dojoswap-contracts/tree/audit-fixes/contracts/launchpad |
| **Commit** | `e4f3d90abc8ba419c397cce06e75740b4e2769d8` |
| **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 Dojo Trading. 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 at https://docs.dojo.trading and https://medium.com/@dojoswap. |
| **Coverage** | **NOT-SUFFICIENT** | Test coverage can be heavily improved to increase test suite prior deployments. |
| **Readability** | **NOT-SUFFICIENT** | There are several commented codes that can be removed to improve code readability. |
| **Complexity** | **SUFFICIENT** | N/A |

## Findings Summary

| Summary Title | Risk Impact | Status |
|---|---|---|
| Offering tokens cannot be transferred to users and contract admin | CRITICAL | RESOLVED |
| Permissionless `FlipAllowClaim` message allows user to instantly harvest after end time | SEVERE | RESOLVED |
| Centralization risks in the `FinalWithdraw` message | MODERATE | ACKNOWLEDGED |
| Updating launchpad configuration introduces state inconsistency and incorrect computations | MODERATE | ACKNOWLEDGED |
| Admin and offering token addresses are not validated | LOW | ACKNOWLEDGED |
| The start and end times of launchpads are not validated | LOW | ACKNOWLEDGED |
| Usage of deprecated functions | INFO | RESOLVED |
| Outdated workspace optimizer | INFO | ACKNOWLEDGED |
| Usage of canonical addresses decreases code readability | INFO | ACKNOWLEDGED |

## Findings Technical Details

### 1. Offering tokens cannot be transferred to users and contract admin

| RISK IMPACT: CRITICAL | STATUS: RESOLVED |
|---|---|

#### Description

The `harvest` and `final_withdraw` functions in `contracts/launchpad/src/contract.rs:245` and `323` set the recipient of `Cw20ExecuteMsg::Transfer` to be `env.contract.address`, which is the contract address. This is incorrect because the offering tokens should be sent to the caller, not the launchpad contract address.

Consequently, users will not receive their offering tokens after depositing native tokens in the contract, and the contract admin cannot withdraw excess offering tokens.

#### Recommendation

Consider modifying the recipient to be `info.sender` in lines `245` and `323`.

## 2. Permissionless `FlipAllowClaim` message allows user to instantly harvest after end time

| RISK IMPACT: SEVERE | STATUS: RESOLVED |
|---|---|

### Description

The `flip_allow_claim` function in `contracts/launchpad/src/contract.rs:337` does not implement any authentication check to ensure the caller is the contract admin. This is an issue because anyone can call this message to enable `state.allow_claim` to true so the `harvest` function can be called in line `213`.

Furthermore, anyone can disable `state.allow_claim` to false in order to prevent users from harvesting and claiming their allocation of total offering tokens.

### Recommendation

Consider adding validation in the `flip_allow_claim` function to ensure the caller is the contract admin and `state.allow_claim` can only be changed from false to true and not vice versa. This improves transparency so that the contract admin cannot block users from claiming the allocation tokens after the end of a token sale.

## 3. Centralization risks in the `FinalWithdraw` message

| **RISK IMPACT: MODERATE** | **STATUS: ACKNOWLEDGED** |
|---|---|

### Revision Notes

The client mentions that they understand centralization risks. In the future, fund withdrawals from launchpads will be conducted through multi-sigs to reduce centralization risks.

### Description

The `final_withdraw` function in `contracts/launchpad/src/contract.rs:272` allows the contract admin to withdraw arbitrary amount of `state.raising_denom` (native funds the user deposits) and `state.offering_token` (CW20 tokens to offer to users). This feature is powerful in such that the withdrawals can be executed multiple times without restrictions by the contract admin.

If the contract admin performs a full withdrawal of funds, users will not be able to receive any offering tokens or native token refunds after the completion of the token sale.

### Recommendation

Consider implementing a validation so that the `final_withdraw` function can only be executed after a period when the token sale finishes to act as a last withdrawal for unclaimed funds. For example, the following validation can be implemented:

```
env.block.time.seconds() >  state.end_time + HARVEST_DURATION
```

The `HARVEST_DURATION` constant represents the period for users to claim their offering tokens after participating in the token sales. The duration should be set to a considerable amount of time and clearly documented and communicated to users in order to maintain transparency and prevent misunderstandings.

## 4. Updating launchpad configuration introduces state inconsistency and incorrect computations

| RISK IMPACT: MODERATE | STATUS: ACKNOWLEDGED |
|---|---|

### Description

The `update_config` function in `contracts/launchpad/src/contract.rs:95` allows the contract admin to update several configurations, such as the raising denom, offering token address, and the target raising amount. However, updating these values with ongoing crowdfunding would introduce consequences.

For example, a state inconsistency issue will happen if the `raising_denom` is updated to a different denom. The state records that the contract has `state.total_amount` amount of `raising_denom`, but the actual contract balance does not have the recorded amount of funds. This would cause the `Harvest` and `FinalWithdraw` messages to fail due to insufficient funds.

Another example is updating `raising_amount`, which will cause incorrect computations to occur. This is because the `get_offering_amount` and `get_refunding_amount` functions in lines `224` and `231` use it to determine the amount of offering tokens and native tokens to refund to the user. Modifying it would cause some users to receive more or less funds compared to claimed users with the same deposit amount.

### Recommendation

Consider adding validation so `update_config` can only be called if there are no deposits in the contract. This can be achieved by checking `state.total_amount` is zero before allowing the configurations to be updated..

## 5. Admin and offering token addresses are not validated

| RISK IMPACT: LOW | STATUS: ACKNOWLEDGED |
|---|---|

### Revision Notes

The client notes that deployment scripts have been written to conduct such checks externally prior to the on-chain deployment.

### Description

The `instantiate` function in `contracts/launchpad/src/contract.rs:31` and `35` does not validate the addresses for `msg.admin` and `msg.offering_token`. If the provided admin or offering token address is not valid, the `update_config`, `harvest`, and `final_withdraw` functions will fail to work properly.

### Recommendation

Consider validating the admin and the offering token address with `addr_validate` in lines `31`, `33`, and `117` during the `instantiate` and `update_config` phases.

## 6. The start and end times of launchpads are not validated

| RISK IMPACT: LOW | STATUS: ACKNOWLEDGED |
|---|---|

### Description

The `instantiate` function in `contracts/launchpad/src/contract.rs:34-35` does not validate `msg.end_time` is larger than `msg.start_time`. If `msg.start_time` is larger than `msg.end_time`, the logic in line `150` cannot be satisfied, preventing users from depositing funds into the contract.

### Recommendation

Consider adding validation to ensure `msg.end_time` is larger than `msg.start_time` in lines `34`, `35`, `125`, and `129` during the `instantiate` and `update_config` phases.

## 7. Usage of deprecated functions

| RISK IMPACT: INFORMATIONAL | STATUS: RESOLVED |
|---|---|

### Description

The codebase uses multiple instances of deprecated functions, which are `to_binary`, `from_binary`, and `from_slice` functions.

### Recommendation

Consider replacing the deprecated function instances with `to_json_binary` and `from_json` functions instead.

## 8. Outdated workspace optimizer

| **RISK IMPACT:** INFORMATIONAL | **STATUS:** ACKNOWLEDGED |
|---|---|

### Description

The `build_release.sh` script uses `cosmwasm/workspace-optimizer:0.11.5`, which is outdated.

### Recommendation

Consider updating the workspace optimizer in the `build_release.sh` script to the latest version for improved performance and compatibility.

## 9. Usage of canonical addresses decreases code readability

| RISK IMPACT: INFORMATIONAL | STATUS: ACKNOWLEDGED |
|---|---|

### Description

The codebase uses multiple instances of canonical addresses instead of human-readable addresses. This decreases the code's readability and can be error-prone.

### Recommendation

Consider replacing the canonical addresses with human-readable addresses and validate them with `deps.api.addr_validate` for improved readability and reliability.

## Document Control

| Version | Date | Notes |
|---|---|---|
| **-** | 5th January 2024 | Security audit commencement date. |
| **0.1** | 15th January 2024 | Initial report with identified findings delivered. |
| **0.5** | 15th/30th January 2024 | Fixes remediations implemented and reviewed. |
| **1.0** | 30th January 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

THANK YOU FOR CHOOSING

SCV SECURITY

scv.services

contact@scv.services