AUDIT REPORT

Dojo Trading

DojoSwap Factory

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:

- `dojoswap_factory`: Fork of TerraSwap Factory contract that enables the creation of `dojoswap_pair` contracts.
- `dojoswap_pair`: Fork of TerraSwap Pair contract that enables users to provide liquidity, withdraw liquidity, swap, and receive assets.
- `dojoswap_router`: Fork of Terraswap Router contract that enables users to perform multi-hop swapping across multiple `dojoswap_pair` contracts.
- `dojoswap_staking`: Fork of Anchor's staking contract that distributes rewards to stakers based on distribution schedules.
- `dojoswap_token`: Fork of TerraSwap Token contract that represents the liquidity token used in `dojoswap_pair` contracts. It can also be used as a standard CW20 token contract.

Submitted Codebase

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

Revision Codebase

dojoswap-contracts	
Repository	https://github.com/dojo-trading/dojoswap-contracts
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**	There are compiler errors when running the test cases. After fixing the error by adding the `custom_query_type` parameter in `contracts/dojoswap_staking/src/mock_querier.rs:19`, the `test_migrate_staking` and `test_update_config` test cases fail.
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
Funds in the factory contract can be stolen	MODERATE	ACKNOWLEDGED
Updating `asset_infos` causes losses for liquidity providers	MODERATE	RESOLVED
Governance address is not validated	LOW	RESOLVED
Admin and offering token addresses are not validated	LOW	ACKNOWLEDGED
Swaps will fail if the commission amount is one	LOW	RESOLVED
Misleading attributes emitted	INFO	RESOLVED
Usage of deprecated functions	INFO	RESOLVED
Contracts should use two-step ownership transfer	INFO	ACKNOWLEDGED
Outdated workspace optimizer	INFO	ACKNOWLEDGED
Usage of canonical addresses decreases code readability	INFO	ACKNOWLEDGED
Inadequate labelling of liquidity pool tokens	INFO	ACKNOWLEDGED

Audit Observations

The audit observations section is intended to present potential findings that are related to the underlying design of the protocol and would require underlying design changes to remediate that may change the overall functioning of the protocol. SCV asks that the client formulate responses to add context to validate or invalidate the following concerns.

1. The factory contract owner can modify the asset decimals into arbitrary values

The `execute_admin_config` function in `contracts/dojoswap_factory/src/contract.rs:107` allows the factory contract owner to update the asset decimals values into arbitrary values. This functionality is powerful and arbitrary, as the asset decimal values do not need to be updated.

For example, the native token decimal values should follow the configured value in the `execute_add_native_token_decimals` function stored in `ALLOW_NATIVE_TOKENS`. For CW20 tokens, the decimal values should follow the token contract's decimal values in `packages/dojoswap/src/asset.rs:171-174`.

Potential Remediation

Consider removing the `execute_admin_config` function.

Revision Notes

The client provided an effective remediation to this observation. This particular observation is now mitigated, posing no further risks.

2. Hardcoded governance address during contract migration

The `migrate` function in `contracts/dojoswap_staking/src/contract.rs:486-488` updates the governance address to `inj12qy3algm6e0zdpv8zxvauzquumuvd39ccdcdjt` when performing a contract migration. This can introduce unintended consequences when the current governance address is not the hardcoded address.

For example, assume a scenario where the hardcoded address is compromised and the governance address is not the hardcoded address. The contract migration admin then performs a migration, causing the governance address to be updated back to `inj12qy3algm6e0zdpv8zxvauzquumuvd39ccdcdjt`, resulting in the ability for the attacker to call the `update_config` and `migrate_staking` functions.

Potential Remediation

Consider adding the governance address parameter to the initialization message in line 27 and only calling `store_gov` if the current contract version is lower than the migrated contract version in line 486.

This can be achieved by using the `get_contract_version` function to retrieve the contract version and call `set_contract_version` after the migration is successful, ensuring the `store_gov` function is only called once when migrating from a specific contract version.

Revision Notes

The client provided an effective remediation to this observation. This particular observation is now mitigated, posing no further risks.

3. Hardcoded and privileged fee collector address

The `FEE_COLLECTOR` address is hardcoded in `contracts/dojoswap_pair/src/contract.rs:40-41` as `inj18xg2xfhv36v4z7dr3ldqnm43fzukqgsafyyg63` instead of being a configurable value. The fee collector is responsible for receiving half of the commission fees in line `581` and also calling the `admin_configure` function in lines `155` to `157` to update the pool's assets and decimal values.

Since the fee collector address is hardcoded directly in the contract, it is not possible to update the address to restrict the damage in case of a compromised account. Due to the admin privilege, an attacker can weaponize the `admin_configure` function in line `164` to manipulate the `asset_infos` array to be their fake tokens, allowing them to mint arbitrary amounts of LP tokens when calling `ProvideLiquidity`. To steal the funds from the pool, the attacker updates the `asset_infos` back to the initial pool assets and then burns their liquidity tokens with `WithdrawLiquidity`.

Potential Remediation

Consider adding the fee collector address parameter to the initialization message in line `47` to be a configurable state and remove the hardcoded address in line `40`.

Findings Technical Details

1. Funds in the factory contract can be stolen

RISK IMPACT: MODERATE	STATUS: ACKNOWLEDGED

Revision Notes

The client advises that the purpose of having funds inside the factory is originally meant to allow for adding a new native token via `execute_create_pair`. They state that only 1 WEI of denomination is required. After adding via the aforementioned function call, there is no more use for the denom inside the factory. In fact, it was never intended for the denom that was added into the factory to be withdrawn after the operation was done. 1 WEI of denomination hardly makes for any financial damage, and they accept the risks of this.

Description

The `execute_create_pair` function in `contracts/dojoswap_factory/src/contract.rs:151` allows anyone to create a pair contract. After the pair contract is instantiated, the `CREATE_PAIR_REPLY_ID` reply ID will be executed to provide liquidity based on the amount of the specified assets in line 324.

The problem occurs when the `execute_create_pair` function does not validate that the caller actually sent the native funds if they intend to create a pair contract that consists of native tokens. If the factory contract contains the funds, it will be used instead. The probability of the factory contract holding native tokens is high because the `execute_add_native_token_decimals` function only allows updating the `ALLOW_NATIVE_TOKENS` state if the balance of native token in the contract is not zero, as seen in lines 241 to 246.

Consequently, an attacker can steal the funds from the factory contract by creating a new pair contract that consists of the native token, and then receive the funds after burning the liquidity tokens.

Recommendation

Consider adding validation to ensure the caller provided the native funds in the `execute_create_pair` function.

2. Updating `asset_infos` causes losses for liquidity providers

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

The `admin_configure` function in `contracts/dojoswap_pair/src/contract.rs:164-166` allows the hardcoded fee collector address to modify the `asset_infos` and `asset_decimals` values. This causes incorrect computations when providing and withdrawing LP tokens in the contract.

For example, assume the contract holds INJ-USDT assets, and the fee collector address updates it to WETH-USDT assets. In this case, the pool liquidity will be determined by the balance of the WETH-USDT assets in the contract instead of INJ-USDT assets. If a liquidity provider decides to redeem their LP tokens, they will not receive a fair share of WETH-USDT assets based on the amount they initially provided with INJ-USDT assets. The same happens when providing liquidity, as the LP token to mint will be computed with the contract balance of the WETH-USDT assets, causing it to have different costs compared to existing liquidity providers who provide liquidity with INJ-USDT assets.

Recommendation

Consider removing the `admin_configure` function. If there is a requirement for new assets to be traded, call the `CreatePair` message in the factory contract instead.

3. Governance address is not validated

RISK IMPACT: LOW	STATUS: RESOLVED

Description

The `update_gov` function in `contracts/dojoswap_staking/src/contract.rs:241` does not validate whether the `gov` string is a valid address. If the provided governance address is not a valid address, the contract will operate without a valid governance account, preventing the `update_config` and `migrate_staking` functions from being called.

Recommendation

Consider validating the governance address with `addr_validate` in the `update_gov` function.

4. Swaps will fail if the commission amount is one

RISK IMPACT: LOW	STATUS: RESOLVED

Description

The `swap` function in `contracts/dojoswap_pair/src/contract.rs:577` divides the commission amount by two and sends the funds to the fee collector address. In an edge case where the commission amount is one, the amount will become zero after division due to rounding, causing the swap to fail because Cosmos SDK prevents sending zero native token amount.

Recommendation

Consider skipping the fee distribution if the commission amount is one to prevent the swap from failing.

5. Misleading attributes emitted

RISK IMPACT: INFORMATIONAL	STATUS: RESOLVED

Description

The `execute_admin_config` function in `contracts/dojoswap_factory/src/contract.rs:111` emits the action attribute as "*update_config*", which is the same as the `execute_update_config` function in line 147.

This would confuse event listeners and off-chain indexers as the same action is emitted for different functionalities.

Recommendation

Consider modifying the `execute_admin_config` function's attribute to emit as "*update_admin_config*".

6. 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.

7. Contracts should use two-step ownership transfer

RISK IMPACT: INFORMATIONAL	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 configuration changes occur.

- `contracts/dojoswap_factory/src/contract.rs:134`
- `contracts/dojoswap_staking/src/contract.rs:241`

Recommendation

Consider 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. SCV suggests the following implementation: https://docs.rs/cw-controllers/latest/cw_controllers/index.html.

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.

10. Inadequate labelling of liquidity pool tokens

RISK IMPACT: INFORMATIONAL	STATUS: ACKNOWLEDGED

Description

The `instantiate` function in `contracts/dojoswap_pair/src/contract.rs:79` labels the LP tokens without including the names of the tokens. This makes it hard to differentiate and identify the LP tokens in blockchain explorers and wallets due to the vague labelling.

Recommendation

Consider modifying the LP token label to include the names, symbols, and other relevant information for better clarity and usability in various interfaces.

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

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