AUDIT REPORT

Zignaly
Zigchain
(Cosmos-SDK)

Prepared by SCV-Security

On 30th January 2025

Table of Contents

Introduction

SCV has been engaged by Zignaly 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

Zigchain implements the `x/dex` and `x/factory` modules. The `x/dex` module allows users to provide liquidity into pools, swap assets, and withdraw liquidity. The `x/factory` module enables users to create factory tokens and mint/burn them. Features such as updating the token's metadata and max supply are also supported.

Submitted Codebase

The Zignaly team provided the codebase compacted in a zip file. The file has the following SHA-1 hash.

Zig chain – Cosmos-SDK	
File	Zigchain_20250103.zip
Commit	`66b738884ae5e99c8d3ae50dbd8b08a783922b9d`

Revisions Codebase

The Zignaly team provided the codebase compacted in a zip file. The file has the following SHA-1 hash.

Zig chain – Cosmos-SDK	
File	zigchain_20250127.zip
Commit	`a6644471abe7d23cb4b280fae84d03a16f197340`

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 Zignaly. 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**	N/A
Coverage	**SUFFICIENT**	N/A
Readability	**SUFFICIENT**	N/A
Complexity	**SUFFICIENT**	N/A

Findings Summary

Summary Title	Risk Impact	Status
Attackers can steal funds due to incorrect sender field	CRITICAL	RESOLVED
No slippage protection when swapping tokens, adding liquidity, or removing liquidity	SEVERE	ACKNOWLEDGED
Fees are stuck in module accounts	MODERATE	RESOLVED
Missing validations of metadata URI	LOW	RESOLVED
Incomplete genesis validations for x/dex and x/factory	LOW	RESOLVED
Creator address is left out when updating the maximum denom supply	INFO	RESOLVED
Incorrect function used when validating token denom	INFO	RESOLVED
Denom descriptions are not configurable on creation	INFO	PARTIALLY RESOLVED

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. Users cannot provide single-sided liquidity

When adding liquidity into pools, users may choose to add single-sided liquidity. The `x/dex` module intends to support this by allowing users to provide empty base or quote assets in `x/dex/types/message_add_liquidity.go:39-45`.

However, the actual logic in `MsgAddLiquidity` will not work correctly if the user intends to add single-sided liquidity:

- If any of the tokens provided is empty, `sdk.NewCoins` will remove them in `x/dex/keeper/msg_server_add_liquidity.go:45`. This causes an index out-of-range error when the `tempCoins` variable is validated with `pool.Coins[n].Denom`.
- The LP tokens minted will become zero because the `CalculateLiquidityShares` function will compute the `incomingProduct` variable as zero value in `x/dex/keeper/msg_server_add_liquidity.go:174`.

Potential Remediation:

Consider adding support for users to provide single-sided liquidity.

Revision Notes

The team advised that they are working on an improved `x/dex` version where this will be addressed alongside other changes.

2. `x/factory` tokens can be burnt via CosmWasm contracts

The `MsgBurnTokens` entry point allows the bank admin to burn the tokens and decrease the denom's supply state in `x/factory/keeper/msg_server_burn_tokens.go:84`. However, since native tokens can be burnt directly via [CosmWasm contract's `BankMsg::Burn` message](), the intended design of the protocol may not work properly.

Specifically, users other than the bank admin can burn the tokens as long they own them. This would cause the `Denom.Supply` state (`x/factory/types/denom.pb.go:36`) to record incorrect values because it does not consider the tokens burnt via CosmWasm contracts. This may cause unintended failures when minting new tokens due to incorrectly triggering the max supply error in `x/factory/keeper/msg_server_mint_and_send_tokens.go:39-50`.

Potential Remediation:

Consider modifying the implementation not to record the token's total supply in the storage state. Instead, the bank keeper's `TotalSupply` function should always be used to correctly determine the total supply of an `x/factory` token.

Revision Notes

The Zignaly team effectively addressed this finding as part of our recommendation.

Findings Technical Details

1. Attackers can steal funds due to incorrect sender field

RISK IMPACT: CRITICAL	STATUS: RESOLVED

Description

The `CreatePool` function in `x/dex/keeper/msg_server_create_pool.go:165` allows users to create a pool by providing base and quote assets in exchange for LP tokens.

The issue is that the funds are incorrectly taken from the `receiver` instead of the `sender`. This is problematic because the transaction signer is the sender (`proto/zigchain/dex/tx.proto:50`), not the recipient. Additionally, the LP tokens are incorrectly minted to the sender instead of the recipient.

This combination allows an attacker to steal funds by specifying the victim's address as the recipient to receive the LP tokens. After that, the attacker would obtain the victim's assets after burning the LP tokens with `MsgRemoveLiquidity`, causing a loss of funds.

Recommendation

Consider performing the following recommendations:

- Update the `SendCoinsFromAccountToModule` function parameter to transfer the assets from the `sender`.
- Update the `SendCoinsFromModuleToAccount` function parameter to transfer the assets to the `recipient`.

2. No slippage protection when swapping tokens, adding liquidity, or removing liquidity

RISK IMPACT: SEVERE	STATUS: ACKNOWLEDGED

Revision Notes

The team advised that they are working on an improved `x/dex` version where this will be addressed alongside other changes.

Description

When calling `MsgAddLiquidity`, `MsgRemoveLiquidity`, and `MsgSwap` in the `x/dex` module, there is no parameter defining a minimum amount of coins or LP tokens that are necessary for the user to receive for the message execution to be successful.

This creates opportunities for malicious actors to target users with front-running and sandwich attacks, moving prices significantly and causing unexpected losses for protocol users.

Recommendation

Consider adding a parameter specifying the minimum amount of coins or LP tokens to receive when calling `MsgAddLiquidity`, `MsgRemoveLiquidity`, and `MsgSwap` to protect against slippage.

3. Fees are stuck in module accounts

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

When users create a pool or token factory denom, they must pay the fees configured in `params.CreationFee` or `params.CreateFeeDenom` and `params.CreateFeeAmount`. These fees are transferred to the `x/dex` and `x/factory` module accounts in `x/dex/keeper/msg_server_create_pool.go:115` and `x/factory/keeper/msg_server_create_denom.go:74`, which will be stuck and cannot be withdrawn.

Recommendation

Consider implementing a fee collector address configuration and transferring the fees to there instead.

4. Missing validations of metadata URI

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In `x/factory/types/message_set_denom_metadata.go:33`, the `ValidateBasic` function of `MsgSetDenomMetadata` does not validate the `msg.Metadata.URI` and `msg.Metadata.URIHash` fields. Specifically, the request URI, URI length, and URI hash lengths should be validated, similar to the implementation in `x/factory/types/msg_create_denom.go:62-84`.

Recommendation

Consider applying the validations implemented in `x/factory/types/msg_create_denom.go:62-84`.

5. Incomplete genesis validations for x/dex and x/factory

RISK IMPACT: LOW	STATUS: RESOLVED

Description

In genesis, x/dex and x/factory accept a GenesisState to set up the initial state of the modules. While the Params field is properly validated for both modules at genesis, validations are not implemented to ensure that the modules are correctly set up and will not lead to any unexpected or incorrect behavior.

In x/dex/types/genesis.go:26, there should be additional genesis validations to ensure that each Pool entry in the PoolsList has valid values for PoolId, LPToken, Creator, Fee, and Coins, and that each PoolUids entry in the PoolUidsList has a valid PoolUid value and a PoolId that matches a corresponding Pool entry.

In x/factory/types/genesis.go:22, there should be additional genesis validations to ensure that each Denom entry in the DenomList has valid values for Creator and Denom and that the Supply value is less than or equal to the MaxSupply. Also, there should be additional validations to ensure that each DenomAuth entry in the DenomAuthList has a valid Denom, BankAdmin, and MetadataAdmin value.

Recommendation

Consider adding the additional GenesisState validations suggested above for the x/dex and x/factory modules.

6. Creator address is left out when updating the maximum denom supply

RISK IMPACT: INFORMATIONAL	STATUS: RESOLVED

Description

The UpdateDenomMaxSupply function in x/factory/keeper/msg_server_update_denom_max_supply.go:52-62 creates a types.Denom struct and stores the new denom information into it. However, the Creator field (x/factory/types/denom.pb.go:33) is not included in the struct, causing the original creator's address to be left out. As a result, the creator field will be missing when the denom is queried, causing a loss of information.

Recommendation

Consider setting the Creator field as currentDenom.Creator for the new denom.

7. Incorrect function used when validating token denom

RISK IMPACT: INFORMATIONAL	STATUS: RESOLVED

Description

In x/factory/types/message_update_denom_metadata_auth.go:26, the ValidateBasic function of MsgUpdateDenomMetadataAuth validates the msg.Denom parameter with the CheckDenomString function. This is incorrect because the msg.Denom parameter will be provided as an x/factory denom when updating the denom metadata.

Recommendation

Consider validating the msg.Denom parameter with the DeconstructDenom function implemented in x/factory/types/denoms.go:51.

8. Denom descriptions are not configurable on creation

RISK IMPACT: INFORMATIONAL	STATUS: PARTIALLY RESOLVED

Revision Notes

The team advised that they edited the default denom description to be an empty string since they expect users to mainly rely on the URI and URI Hash. The denom description is still configurable through `SetDenomMetadata`.

Description

Whenever a user creates a new denom with `MsgCreateDenom` in `x/factory`, the description field is hardcoded as *“Denom created on zigchain factory with love”*. While this description can be updated by calling `SetDenomMetadata`, however, it requires an additional message execution and gas usage for the denom creator.

Recommendation

Consider adding a parameter to `MsgCreateDenom` to allow a denom creator to specify their denom description upon creation instead of requiring them to call `SetDenomMetadata`.

Document Control

Version	Date	Notes
-	7th January 2025	Security audit commencement date.
0.1	18th January 2025	Initial report with identified findings delivered.
0.5	29th January 2025	Fixes remediations implemented and reviewed.
1.0	30th January 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