Audit Report

Sei Cosmos

v1.0

May 15, 2023

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Sei Labs Inc to perform a security audit of Sei Cosmos.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following GitHub repository:

https://github.com/sei-protocol/sei-cosmos

Commit hash: `afe957cab74dd05c213d082d50cae02dd4cb6b73`

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

Sei Cosmos is a fork of Cosmos SDK, which is an open-source framework for building multi-asset public Proof-of-Stake (PoS) blockchains, like the Cosmos Hub, as well as permissioned Proof-of-Authority (PoA) blockchains. The Sei team added optimizations to Cosmos SDK, such as block proposal processing and parallel transaction message processing. They are implemented with the help of ABCI++ interface integration that is aligned with the Sei Tendermint Core.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium-High	Readability is generally in line with the upstream Cosmos SDK repository.
Level of documentation	Medium-High	-
Test coverage	Medium	Go test reports a 57.9% code coverage excluding simulation packages and automatically generated files.

Summary of Findings

No	Description	Severity	Status
1	Non-deterministic iteration in `BuildDependencyDag` may break consensus	Critical	Resolved
2	`RegisterWasmDependency` allows anyone to register contract dependency mappings that can increase gas consumption	Major	Resolved
3	Several errors are not handled	Major	Resolved
4	Missing `LegacyAmino` codec registration	Minor	Resolved
5	Use of deprecated function	Informational	Resolved
6	Resolve TODOs before release	Informational	Resolved
7	Unnecessary lock acquisition might negatively impact performance	Informational	Resolved
8	`MintCoins` does not perform a nil pointer validation in case the module name is passed incorrectly	Informational	Resolved
9	Unused assignment in `GetSigners`	Informational	Resolved
10	Dead code in `encryptPrivKey`	Informational	Resolved
11	Missing usage description for transaction and query CLI commands	Informational	Resolved
12	Miscellaneous comments	Informational	Resolved

Detailed Findings

1. Non-deterministic iteration in `BuildDependencyDag` may break consensus

Severity: Critical

In `x/accesscontrol/keeper/keeper.go:333`, nodes iterate over `anteDepSet`. Due to the non-deterministic behavior of iteration over maps in Go, there could be two different types of errors from `ValidateAccessOp`. This could lead to consensus failure.

Recommendation

We recommend modifying the data structure that holds the set of dependencies so that iteration becomes deterministic. For example, an array could be used which could be iterated over in the deduplication step in line `326`. Alternatively if dependencies can be ordered, the set could be converted into an array.

Status: Resolved

2. `RegisterWasmDependency` allows anyone to register contract dependency mappings that can increase gas consumption

Severity: Major

The `RegisterWasmDependency` function in `x/accesscontrol/keeper/msg_server.go:22` allows anyone to register contract dependency mappings. It does not validate if the transaction sender is the contract owner. These contract dependency mappings are used by the Sei Chain where the `GetWasmDependencyMapping` function defined in `x/accesscontrol/keeper/keeper:126` gets a contract's dependency mapping information. This may create opportunities for attackers to exploit applications on top of Sei Chain. For instance, an exploiter can make the `getMessageMultiplierDenominator` function in `app/antedecorators/gas.go:34` of Sei Chain return `WasmCorrectDependencyDiscountDenominator`. This will lead to an increase of the gas consumption for that particular contract. Moreover, the `DispatchMsg` function in `wasmbinding/message_plugin.go:118` of Sei Chain does not validate message dependencies if a contract's `WasmDependencyMapping` is disabled.

Recommendation

We recommend adding validation to assert that the transaction sender is the contract owner before allowing registration of contract dependency mappings.

Status: Resolved

3. Several errors are not handled and may cause an inconsistent state

Severity: Major

In several places of the codebase, there are unhandled errors that allow execution to continue even when operations have failed. This may produce inconsistent states and unexpected panics.

Instances of unhandled errors can be found in:

- `x/bank/keeper/keeper.go:47`
- `x/bank/keeper/keeper.go:456`
- `x/bank/keeper/keeper.go:434`
- `x/bank/keeper/keeper.go:378`
- `x/accesscontrol/handler.go:14`
- `baseapp/abci.go:31`

As an example, the `UpsertDeferredSends` function may return an error, which is not checked. The calling function, `DeferredSendCoinsFromAccountToModule`, should propagate the Error upstream, but since the Error is not handled this will not happen. Since the execution would continue, it may leave the network in an inconsistent state or cause an unexpected panic.

Recommendation

We recommend checking all errors and aborting the execution of message handlers if necessary.

Status: Resolved

4. Missing `LegacyAmino` codec registration

Severity: Minor

In `x/accesscontrol/module.go:45`, the `RegisterLegacyAminoCodec` function does not register the module's `LegacyAmino` codec. This is necessary for JSON serialization to support hardware devices like Ledger since these devices do not support proto transaction signing.

Recommendation

We recommend registering the `LegacyAmino` codec for the module using the `RegisterCodec` function in `x/accesscontrol/types/codec.go:10`.

Status: Resolved

5. Use of deprecated function

Severity: Informational

In `x/accesscontrol/client/utils/utils.go:15` and `x/accesscontrol/client/utils/utils.go:28`, the `ReadFile` function is called from the `ioutil` package, yet that package is deprecated as of Go 1.16. The function should be used from the `os` package instead.

Recommendation

We recommend using the `ReadFile` function from the os package.

Status: Resolved

6. Resolve TODOs before release

Severity: Informational

There are multiple TODOs in the codebase that may refer to sensitive or important logic. Instances are:

- `baseapp/abci.go:170`
- `baseapp/baseapp.go:717`
- `x/accesscontrol/keeper/keeper.go:406`
- `types/module/module.go:289`

Recommendation

We recommend resolving all TODOs in the codebase before production release, especially those which may relate to sensitive or important logic.

Status: Resolved

7. Unnecessary lock acquisition might negatively impact performance

Severity: Informational

In `types/context_cache.go:68`, there is a check to circuit break in case the amount is not valid. However, before that check, a lock is acquired which is not needed for this specific validation.

Recommendation

We recommend deferring the lock acquisition until the check of the amount is performed.

Status: Resolved

8. `MintCoins` does not perform a nil pointer validation in case the module name is passed incorrectly

Severity: Informational

In `x/bank/kepper/keeper.go:561`, the module account is retrieved to add coins to it through the `createCoins` function in line `565`. However, if the `moduleName` does not exist, it would return a nil pointer from the `GetModuleAccountAndPermissions` function.

Currently, the only place that the function is called is in `x/mint/keeper/keeper.go:103`, and the module name is a constant. Therefore, unless that specific module name does not exist, there is no attack vector with the current design.

Recommendation

We recommend adding a check that the account is not nil to avoid issues in the future in case the `MintCoins` function is called directly with a variable module name.

An alternative is to remove the module name argument and use a global constant instead.

Status: Resolved

9. Unused assignment in `GetSigners`

Severity: Informational

In `types/tx/types.go:125`, there is an assignment to the variable `seen` which is never used thereafter.

Recommendation

We recommend removing the assignment completely.

Status: Resolved

10. Commented code in `encryptPrivKey`

Severity: Informational

In `encryptPrivKey` in `crypto/armor.go:149`, there is an assignment that has been commented out.

Recommendation

We recommend removing the commented assignment from the codebase.

Status: Resolved

11. Missing usage description for transaction and query CLI commands

Severity: Informational

The transaction and query commands of the accesscontrol module in `x/accesscontrol/client/cli/tx.go:18` and `x/accesscontrol/client/cli/query.go:13` are missing a long message to describe their usage, which could be helpful for users and external developers.

Recommendation

We recommend specifying a long message for all transaction and query CLI commands. Each command should provide a description of how to correctly use the command.

Status: Resolved

12. Miscellaneous comments

Severity: Informational

Across the codebase, various instances of unused code and misleading comments have been found.

Recommendation

The following are some recommendations to improve the overall code quality and readability:

- The range expression in `types/accesscontrol/access_operation_map.go:39` can be simplified by omitting the `, _`.
- The `println` function in `x/accesscontrol/module.go:62` can be removed as it is there for testing only.
- Remove the unused sentinel error `ErrWasmDependencyRegistrationFailed` in `x/accesscontrol/errors.go:14`.
- Remove the unused function `GetMessageAccessOps` in `types/accesscontrol/access_operation_map.go`.
- Fix the comment in `types/module/module.go:514` which describes another function, `EndBlock`.
- Remove the unused function `NewProposalHandler` in `x/accesscontrol/handler.go:19`.
- Remove the unused function `IsDefaultSynchronousAccessOps` in `x/accesscontrol/types/message_dependency_mapping.go:93`.

Status: Resolved