Audit Report

Stride ICA Oracle

v1.0

August 20, 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 Stride Labs, Inc. to perform a security audit of Stride ICA Oracle.

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 targets:

Repository	https://github.com/Stride-Labs/ica-oracle
Commit	`15af6f1f7ec9d2be532619156a037245e8e8626c`
Scope	All code in this repository was in scope.
Identifier	In this report, all paths pointing to this repository are prefixed with `cw-ica-oracle:`

Repository	https://github.com/Stride-Labs/stride
Commit	`2a9ead570a02cb0ad34040fe6949814bbfcaa6ec`

Scope	Only the changes between the commits `2a9ead570a02cb0ad34040fe6949814bbfcaa6ec` and `4db410ed17e17988339d44afe97174ee663e8392` have been in scope of these audits. These changes include a new ICA oracle Cosmos SDK module in `x/icaoracle` and its integration with the Stride appchain.
Identifier	In this report, all paths pointing to this repository are prefixed with `stride:`

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

This audit covers the functionality associated with Stride's ICA oracle Cosmos SDK module and its accompanying oracle CosmWasm smart contract. The oracle module is designed to provide a generic metric oracle that is simple, integrates with Stride, and is easily deployable on chains that support CosmWasm while remaining isolated from Stride's Stakeibc module logic.

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	**Low-Medium**	-
Code readability and clarity	**Medium-High**	-
Level of documentation	**Medium-High**	The client provided extensive documentation as well as detailed diagrams that detailed all the expected flows.
Test coverage	**Medium-High**	`cw-ica-oracle:` `84.67%` coverage, `116/137` lines covered `stride:` `x/icaoracle/keeper` `91.2%` of statements covered

Summary of Findings

No	Description	Severity	Status
1	`MsgInstantiateOracle` is not registered	Minor	Resolved
2	Amino codec must be registered to support end users with hardware devices like Ledger	Minor	Resolved
3	Missing `GenesisState` validation	Minor	Resolved
4	Governance can activate invalid oracles	Minor	Resolved
5	Removing oracles does not remove associated metrics	Minor	Resolved
6	Oracles cannot be added again if they have an active channel and are removed through governance	Minor	Resolved
7	`RestoreOracleICA` does not ensure that channel is closed	Informational	Resolved
8	Duplicated `ValidateBasic` invocation in CLI	Informational	Resolved
9	Incorrect usage message for `GetCmdQueryOracle` command	Informational	Resolved
10	Unused code	Informational	Resolved
11	Events are not always emitted	Informational	Resolved
12	Misleading comments in the codebase	Informational	Resolved

Detailed Findings

1. `MsgInstantiateOracle` is not registered

Severity: Minor

In `stride:x/icaoracle/types/codec.go:18`, `MsgInstantiateOracle` is not registered as a concrete implementation of the `sdk.Msg` interface. While there are no issues when executing a `MsgInstantiate` transaction message without the registration, an error will occur if a transaction message contains code that involves marshaling or unmarshalling an unregistered message.

Recommendation

We recommend registering `MsgInstantiateOracle` in `RegisterInterfaces` in `stride:x/icaoracle/types/codec.go:18`.

Status: Resolved

2. `Amino` codec must be registered to support end users with hardware devices like Ledger

Severity: Minor

In `stride:x/icaoracle/types/codec.go:33`, `Amino` should be used to register all interfaces and concrete types for the `icaoracle` module. This is necessary for legacy binary and JSON serialization to support hardware devices like Ledger since these devices do not support proto transaction signing.

Recommendation

We recommend registering all interfaces and concrete types for the `icaoracle` module using `Amino` to support legacy binary and JSON serialization.

Status: Resolved

3. Missing `GenesisState` validation

Severity: Minor

In `stride:x/icaoracle/keeper/genesis.go:11`, `InitGenesis` is missing genesis state validation. Without validation, misconfigurations may go unnoticed and might eventually be difficult to correct.

Recommendation

We recommend adding `genState.Validate` in the `InitGenesis` function.

Status: Resolved

4. Governance can activate invalid oracles

Severity: Minor

In `stride:x/icaoracle/keeper/gov/gov.go:11-13`, the `ToggleOracle` function can be called by governance to enable or disable any oracles. However, no validation ensures that the oracle already has a contract instantiated on the host chain and the channel is still active. Enabling such invalid oracles is inefficient.

Recommendation

We recommend ensuring that an oracle has a contract instantiated on the host chain and the channel is still active when toggling the oracle as active.

Status: Resolved

5. Removing oracles does not remove associated metrics

Severity: Minor

In `stride:x/icaoracle/keeper/gov/gov.go:16-24`, the `RemoveOracle` function can be called by governance to remove any oracle as long as it exists in the storage. However, no validation ensures that the metrics associated with the oracles are removed too. If a channel times out and governance removes the oracle, the pending metrics will remain forever in the storage in an `IN_PROGRESS` state.

Recommendation

We recommend removing all metrics with the `DestinationOracle` set as `OracleChainId`.

Status: Resolved

6. Oracles cannot be added again if they have an active channel and are removed through governance

Severity: Minor

In `stride:x/icaoracle/keeper/gov/gov.go:16-24`, the `RemoveOracle` function can be called by governance to remove any oracle as long as it exists in storage. However, if

governance removed `OracleChainId` while the channel is still active, the chain identifier cannot be added back through the `AddOracle` function.

This is due the error occurring in `stride:x/icaoracle/keeper/msg_server.go:85` because the channel would still be active. Since interchain account channels cannot be closed manually by users, the oracle for the specific chain id cannot be added back anymore.

Recommendation

We recommend allowing governance to add oracle's chain identifier as long as the connection remains active.

Status: Resolved

7. `RestoreOracleICA` does not ensure that channel is closed

Severity: Informational

The `RestoreOracleICA` function in `stride:x/icaoracle/keeper/msg_server.go:150` does not explicitly check to ensure that the channel for the requested oracle is closed. If the channel is closed an error should be returned. In the case that this function is called for an open channel `RegisterInterchainAccount` will fail later in the function's execution, but will not return an error that clearly communicates its cause.

Recommendation

We recommend checking whether the channel is open in `RestoreOracleICA` and returning an error with a clear error message if not.

Status: Resolved

8. Duplicated `ValidateBasic` invocation in CLI

Severity: Informational

All transaction CLI commands registered in the `GetTxCmd` function in `stride:x/icaoracle/client/cli/tx.go:19` and `stride:x/icaoracle/client/cli/gov.go:55` call the `msg.ValidateBasic` function before calling `GenerateOrBroadcastTxCLI`.

As `msg.ValidateBasic` is already called inside the `GenerateOrBroadcastTxCLI` function this is a duplicated invocation, which is inefficient.

Recommendation

We recommend removing the duplicated `msg.ValidateBasic` invocation.

Status: Resolved

9. Incorrect usage message for `GetCmdQueryOracle` command

Severity: Informational

The usage message for `GetCmdQueryOracle` command in `stride:x/icaoracle/client/cli/query.go:47` is incorrect. The command expects to receive exactly one argument holding the chain id, but the usage message is not describing this.

Recommendation

We recommend adding a reference to the chain id argument in the usage message in the `GetCmdQueryOracle` command.

Status: Resolved

10. Unused code

Severity: Informational

Across the codebase, instances of unused errors and functions exist in the following locations. Unused code decreases the maintainability of the codebase.

- `ErrMarshalFailure` in `stride:x/icaoracle/types/error.rs:18`
- `ErrUnmarshalFailure` in `stride:x/icaoracle/types/error.rs:19`
- `CWTemplateContract` in `cw-ica-oracle:src/helpers.rs:10-27`

Recommendation

We recommend removing unused errors and functions.

Status: Resolved

11. Events are not always emitted

Severity: Informational

There are multiple functions within the scope of this audit that do not emit events or attributes. It is best practice to emit events and attributes to improve the usability of the contracts and to support off-chain event listeners and blockchain indexers.

The following functions do not emit events or attributes:

- In `cw-ica-oracle:src/contract.rs:15`, the `instantiate` function should emit an event containing the `admin_address` value.
- In `cw-ica-oracle:src/execute.rs:19`, the `post_metric` function should emit events containing the `new_metric` and `new_price` values.

Recommendation

We recommend emitting events as described above.

Status: Resolved

12. Misleading comments in the codebase

Severity: Informational

In `cw-ica-oracle:src/state.rs:127`, the `add` function for the `History` storage state mentions that "*Old items are removed from the back of the deque when capacity is reached*". However, this is incorrect because the old items are actually removed from the *front* of the deque when the capability is reached, as seen in line `137`.

Additionally, in `cw-ica-oracle:src/execute.rs:46`, the comment specifies that duplicated metrics will not be added to the store while in the current implementation duplicate new metrics silently overwrite existing metrics.

Recommendation

We recommend modifying the comments to reflect their functionality correctly.

Status: Resolved