Codex Protocol Registry Contract

Audit Issue Summary

Prepared by Hosho
June 26th, 2018

NOTICE: *This document is a draft and is not representative of a completed audit.*

Overview

The Hosho team completed a thorough audit of the Registry contracts for the Codex Protocol project and the results are found in this report. The Codex Protocol team has been quick to respond and make plans to remediate each of this issues. As such, this report should not reflect a neglect on behalf of the development team but rather the first pass in the auditing process. A particular note should be made of Issue 1.6 which deals with a delegate call which is informational only to indicate that they are utilizing a potentially vulnerable function but that there are protections in place to prevent issues arising.

Issues Found

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged "Resolved" or "Unresolved" depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

- **Critical** - The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.
- **High** - The issue affects the ability of the contract to compile or operate in a significant way.
- **Medium** - The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.
- **Low** - The issue has minimal impact on the contract's ability to operate.
- **Informational** - The issue has no impact on the contract's ability to operate.

1.1 Unresolved, Critical: Repeat Ownership Changes

Contract: DelayedOwnable

Explanation

The `initializeOwnable` function should only be called once. This function does not set the `isInitialized` state after an owner has been assigned, allowing ownership to be changed repeatedly.

1.2 Unresolved, Critical: Potential Ability to Lose Ownership

Contracts: CodexStakeContainer and CodexRecord

Explanation

These contracts use `DelayedOwnable` to manage its ownership, which means there is no explicit initial setup for ownership. As this is normally handled by `initializeOwnable` in the `DelayedOwnable` contract, the contract does not initialize with an owner. If the owner forgets to initialize ownership, or is raced for ownership by a third party, anyone could call `initializeOwnable` to become the owner of this contract.

1.3 Unresolved, High: Incorrect Type Comparison

Contract: CodexRecordMetadata

Explanation

Within the `modifyMetadataHashes` function there is an integer comparison to verify that a *bytes32* is empty. If this remains not casted to the proper type, it will cause hard reverts and errors within the EVM.

1.4 Unresolved, High: Value Mismatch

Contract: RC900BasicStakeContainer

Explanation

The `annualizedInterestRate` does not match the code comments. If `interestRate` is 10 after 1 year, the perceived stake would be 1/1e17 more valuable, instead of 10% more valuable as noted in the contract.

1.5 Unresolved, High: Gas Limit

Contract: ERC900BasicStakeContainer

Explanation

Due to the loops built into the `updatePerceivedStakeAmounts` function, it is possible that this function can exceed the gas limit for the block, causing it to be unable to finalize.

1.6 Unresolved, Informational: Lack of Ownership Modifiers

Contract: CodexRecordProxy

Explanation

Delegate call utilization should be protected behind ownership modifiers, as it allows remote contracts to execute code that can modify the local storage state of the contract. The severity level of this risk can potentially be reduced if implementation details are provided by the Codex Protocol team about the expected protections around this call.

Note

The Codex Protocol team has protections in place to prevent this function from being utilized to gain unwanted control of the contract.

1.7 Unresolved, Informational: Unnecessary Import

Contract: CodexStakeContainer

Explanation

The CodexStakeContainer contract inherits the Pausable contract, but it does not use any pausable functions or modifiers.

1.8 Unresolved, Informational: Allows transferFrom to Owner

Contract: ERC721BasicToken

Explanation

The `transferFrom` function allows for tokens to be transferred from the owner of the token, back to themselves.

Test Results

Failing Tests

1. Contract: CodexRecord `modifyMetadataHashes` should able to use `modifyMetadataHashes` to modify token (See Issue 1.3 Incorrect Type Comparison)
2. Contract: ERC-900 Tests for ERC900BasicStakeContainer `updatePerceivedStakeAmounts` should know the `annualizedInterestRate` is in 1/1e18 instead of % (See Issue 1.4 Value Mismatch)
3. Contract: ERC-900 Tests for ERC900BasicStakeContainer `updatePerceivedStakeAmounts` has vulnerability of exceeds gas cap (See Issue 1.5 Gas Limit)
4. Contract: ERC-721 Tests for ERC-721TokenMock using `transferFrom` should not be able to transfer token to owner (See Issue 1.8 Allows `transferFrom` to Owner)
5. Contract: ERC-721 Tests for ERC-721BasicTokenMock using `transferFrom` should not be able to transfer token to owner (See Issue 1.8 Allows `transferFrom` to Owner)