Security Audit Report

Anichess Ethereum Contracts

v1.2

June 26, 2025

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

THIS AUDIT REPORT WAS PREPARED EXCLUSIVELY FOR AND IN THE INTEREST OF THE CLIENT AND SHALL NOT CONSTRUE ANY LEGAL RELATIONSHIP TOWARDS THIRD PARTIES. IN PARTICULAR, THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THIRD PARTIES AND PROVIDE NO WARRANTIES REGARDING THE FACTUAL ACCURACY OR COMPLETENESS OF THE AUDIT REPORT.

FOR THE AVOIDANCE OF DOUBT, NOTHING CONTAINED IN THIS AUDIT REPORT SHALL BE CONSTRUED TO IMPOSE ADDITIONAL OBLIGATIONS ON COMPANY, INCLUDING WITHOUT LIMITATION WARRANTIES OR LIABILITIES.

This audit has been performed by

Oak Security GmbH

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

Introduction

Purpose of This Report

Oak Security GmbH has been engaged by New Frame Limited to perform a security audit of Audit of the Animoca Staking Pool smart contracts.

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 target:

Repository	https://github.com/animoca/anichess-ethereum-contracts/tree/main/contracts/staking
Commit	`71015e6efb4afa13f30c388d2fc0312d08bdb844`
Scope	All contracts were in scope.
Fixes verified at commit	`75e0894104c5a71297bc8ea568388202abd80847` Note that only fixes to the issues described in this report have been reviewed at this commit. Any further changes such as additional features have not been reviewed.

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

Anichess implements a specialized staking solution built upon the Animoca Staking Pool. It focuses on enabling the staking of ERC20 tokens in exchange for "Points" rewards, integrating with an external points contract. This implementation introduces features such as restricted staking via a claim contract and custom reward distribution logic specifically tailored for the needs of the Anichess application.

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**, **Partially Resolved**, 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**	-
Test coverage	**High**	`hardhat coverage` reports a test coverage of `100%`

Summary of Findings

No	Description	Severity	Status
1	Missing event emission to distinguish stake sources	Informational	Acknowledged
2	Misleading error message	Informational	Acknowledged

Detailed Findings

1. Missing event emission to distinguish stake sources

Severity: Informational

In `anichess-ethereum-contracts/contracts/staking/ERC20StakingPointsRewardsLinearPool.sol:27-37`, and `ERC20StakingPointsRewardsLimitedLinearPool.sol:24-31`, while `Staked` events are emitted for both direct user stakes and claim contract stakes, there is no immediate way to distinguish between these two staking methods in off-chain monitoring without examining the `stakeData` parameter encoding.

Recommendation

We recommend adding a boolean event parameter to the existing `Staked` event to clearly indicate when stakes are performed via the claim contract, thereby improving observability and monitoring.

Status: Acknowledged

2. Misleading error message

Severity: Informational

In `anichess-ethereum-contracts/contracts/staking/ERC20StakingPointsRewardsLimitedLinearPool.sol:32`, the error `OnlyReceiverInterface` is reused in a context where it does not accurately describe the reason for reverting. This can cause confusion for users attempting to stake, developers building interfaces, and readers of the contract, as the error message does not clearly indicate the actual problem encountered.

Recommendation

We recommend defining a custom error that accurately describes the specific revert reason or using a plain `revert(<reason>)` statement with a clear and descriptive message.

Status: Acknowledged