verichains

SECURITY AUDIT OF

SENSEIFI STAKING SMART CONTRACT

Public Report

May 25, 2023

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Sei	A decentralized blockchain built to enable scalable, user-friendly apps for the world.
DAO	A digital Decentralized Autonomous Organization and a form of investor-directed venture capital fund.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on May 25, 2023. We would like to thank the SenseiFi for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the SenseiFi Staking Smart Contract. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified no vulnerable issue in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About SenseiFi Staking Smart Contract

SenseiFi Leading gamified Defi on Sei network with the spiciest rewards.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the SenseiFi Staking Smart Contract. It was conducted on commit `7c4b9304fe1520d37e0b1b0dfec7e3c1b469111c` from git repository link: *https://github.com/senseifi/staking-rewards-nll*.

1.3. Audit methodology

Our security audit process for Sei smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the Sei smart contract:

- Arithmetic Overflow and Underflow
- Ownership checks
- Numerical precision errors
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.

SEVERITY LEVEL	DESCRIPTION
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The SenseiFi Staking Smart Contract was written in `Rust` programming language. It's a staking contract in Sei chain which helps users stake tokens into validator and received the ticket for winning reward.

The Staking contracts of SenseiFi allow users to stake tokens and receive tickets in return. These tokens will be staked into the validator for bonus tokens. The contract admin has the authority to choose a user who will receive 70% of the reward by calling the `set_winner` function, while the remaining 30% will be transferred to the admin account.

Additionally, the admin has the option to redelegate the tokens into a new validator by calling the `redelegate` function.

2.2. Findings

During the audit process, the audit team found no issue in the given version of SenseiFi Staking Smart Contract.

2.3. Additional notes and recommendations

2.3.1. Unused `non_winner` variable in `UserState` INFORMATIVE

The `non_winner` variable can be set by staking user when calling stake function without any restriction. There is no functionality of this variable in the contract.

```
pub fn stake(
        deps: DepsMut,
        env: Env,
        info: MessageInfo,
        non_winner: bool,
    ) -> Result<Response, ContractError> {
    ...
    user_state.non_winner = non_winner;
    ...
}
```

RECOMMENDATION

We should remove the `non_winner` variable or adding the document about this variable.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*May 25, 2023*	Public Report	Verichains Lab

Table 2. Report versions history