SECURITY AUDIT OF

SENSEI LENDING PRIZE GAME SMART CONTRACT

Public Report

Aug 28, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Aug 28, 2024. We would like to thank the Savvio 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 Sensei Lending Prize Game 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 some vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Sensei Lending Prize Game Smart Contract

Savvio is a leading gamified decentralized finance (DeFi) platform that provides secure and reliable savings alternatives together with bonus token and NFT rewards. It blends the excitement of gamification with the stability of DeFi to provide users a one-of-a-kind and gratifying financial experience.

1.2. Audit Scope

This audit focused on identifying security flaws in code and the design of the Sensei Lending Prize Game Smart Contract.

It was conducted on commit `ade5d9c3389cc6c0ee7bbb597e601b212da92caf` from git repository link: *https://github.com/savvio-io/canto-smart-contracts/*

The latest version of the following files were made available in the course of the review:

SHA256 Sum	File
96963721701698de660901fbfba417398e2f00304bd82f925bafa22fc97edbbe	contracts/ERC20PrizeGameRewards.sol
f704d0389569528e3d39554cb45fc0b3f60d030b091a6d002fac6536e552d1a9	contracts/ERC4626PrizeGameRewards.sol
c658c26bc511cc9291df18b6ec8b9ea0ce3cd120e18f4be442e79397118cb86a	contracts/SenseiFiLendingPrizeGame.sol
0a91e0a9e3f854f336cb0d13f4d73f30981080d585a6794634811781f6234ea9	contracts/interfaces/ICErc20.sol

1.3. Audit Methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, 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 smart contract:

- Integer Overflow and Underflow

- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- 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.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Savvio acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. Savvio understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Savvio agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the Savvio will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Savvio, the final report will be considered fully accepted by the Savvio without the signature.

2. AUDIT RESULT

2.1. Overview

The Sensei Lending Prize Game Smart Contract is a smart contract written in Solidity, requiring version `>=0.8.0`. It's based on the ERC4626 standard, which is designed for tokenized vaults, and it includes a feature for distributing rewards to winners, as decided by the owner. The contract allows users to deposit assets and earn shares in return.

2.1.1. Actors & Assets

Actors

- **Owner**: The owner has significant control over the contract, such as setting the reward intervals and rates, which directly affects how rewards are distributed to users.
- **Users**: Users can deposit and withdraw assets. The assets they deposit are always matched by an equal amount of shares.

Assets

- **Tokens**: These are the digital assets managed by the contract, including both native and ERC-20 tokens. Users deposit these tokens, and they are managed and distributed according to the contract's rules.
- **Shares**: These represent the user's stake in the contract's assets.
- **Rewards**: These are additional assets or tokens given to users, based on decisions made by the owner.

2.1.2. Centralization Concerns

The primary centralization concern with this contract is the significant power held by the Contract Owner. The owner has the authority to determine the winner of games, which could lead to concerns about fairness. If the owner can influence or decide the outcome of games, it raises questions about the integrity of the process. This kind of centralized control might result in biased or manipulated results, making it unclear if all participants have an equal and fair chance, or if the outcomes are truly merit-based.

2.2. Findings

During the audit process, the audit team had identified some vulnerable issues in the contract code.

2.3. Draft Findings

#	Title	Severity	Status
1	Incorrect Implementation of `totalAssets` in ERC-4626	HIGH	RESOLVED
2	Incompatibility with Non-Compliant ERC-20 Tokens	MEDIUM	RESOLVED

2.3.1. HIGH - Incorrect Implementation of `totalAssets` in ERC-4626

Positions:

- `contracts/SenseiFiLendingPrizeGame.sol`#L48

The `totalAssets` function is supposed to return the total value of the underlying assets managed by the vault. However, in the current implementation, it returns the `totalSupply` of the ERC-20 token, which represents the number of shares issued by the vault rather than the actual value of the assets.

```
function totalAssets() public view override returns (uint256) {
    return totalSupply;
}
```

RECOMMENDATION

Update the `totalAssets` function to accurately reflect the total value of the assets managed by the vault. This typically involves calculating the value of the underlying assets that correspond to the shares in circulation, rather than simply returning `totalSupply`.

UPDATES

- ***Aug 28, 2024***: The issue has been resolved by the development team at commit `ade5d9c3389cc6c0ee7bbb597e601b212da92caf`.

2.3.2. MEDIUM - Incompatibility with Non-Compliant ERC-20 Tokens

Positions:

- contracts/SenseiFiLendingPrizeGame.sol#L57

The afterDeposit function relies on the approve function of the asset ERC-20 token to return true on success. However, not all ERC-20 tokens strictly adhere to this convention. Some tokens may success call without returning a value, leading to a failure in the require statement and causing the transaction to revert unnecessarily.

```
function afterDeposit(uint256 assets, uint256) internal override {
    require(asset.approve(address(cAsset), assets));
    uint256 result = cAsset.mint(assets);
    require(result == 0, result.toString());
}
```

RECOMMENDATION

To improve compatibility with non-compliant ERC-20 tokens, consider using a safer approach to handle the approval process. This might involve checking the return value of approve more flexibly, or using a low-level call to bypass the return value check, though this comes with trade-offs in terms of safety and transparency.

UPDATES

- ***Aug 28, 2024***: The issue has been resolved by the development team at commit ade5d9c3389cc6c0ee7bbb597e601b212da92caf.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Aug 28, 2024*	Public Report	Verichains Lab

Table 2. Report versions history