verichains

SECURITY AUDIT OF

SINGSING TOKEN

Public Report

Jul 10, 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 Jul 10, 2024. We would like to thank the SingSing 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 SingSing Token. 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 given version of SingSing Token, only some notes and recommendations. All the issues have been acknowledged and fixed in the latest version of the SingSing Token.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About SingSing Token

Users earn SingSing Token through vesting schedules or airdrops, which can be used for service fees, buying NFTs and tickets, minting new NFTs, playing mini-games, in-game services, redeeming vouchers and merchandise, and staking and governance within the SingSing ecosystem.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the SingSing Token. It was conducted on commit `829fc53836ccef7636a3e2979362548a8796fe4e` from git repository `https://github.com/phamsonha/SING-BEP20`.

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

SingSing 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. SingSing understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, SingSing 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 SingSing will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the SingSing, the final report will be considered fully accepted by the SingSing without the signature.

2. AUDIT RESULT

2.1. Overview

The SingSing Token was written in `Solidity` language, with the required version to be `0.8.0`.

The contract extends `Pausable` from the `OpenZeppelin` library. Below table describes some properties of the audited SingSing Token (as of the report writing time).

PROPERTY	VALUE
Name	SingSing Token
Symbol	SING
Decimals	18
Total Supply	2,400,000,000x10^{18} (it represents 2.4 billion tokens)

Table 2. The SingSing Token properties

The SingSing Token is an ERC20 token implementation that includes pausing feature. The total supply of the token is fixed at 2.4 billion SING tokens, which are minted to the owner's address upon deployment. Additionally, it extends OFT (Omnichain Fungible Token) to support cross-chain token transfers via LayerZero's infrastructure. The contract also allows the owner to pause and unpause token transfers as needed.

2.2. Findings

During the audit process, the audit team found no vulnerability in the given version of SingSing Token.

However, the audit team found some notes and recommendations that could improve the codebase and optimize the gas usage.

2.2.1. INFORMATIVE - `SafeMath` redundant in Solidity `0.8.+` version

The `SafeMath` library usage in the contract are for overflow checking, solidity `0.8.0+` already do that by default. Additionally, the contract does not use `SafeMath` anywhere else.

RECOMMENDATION

Remove the `SafeMath` library.

UPDATES

- ***Jul 10, 2024***: The issue has been acknowledged and fixed by the SingSing Token team.

2.2.2. INFORMATIVE - Unused imports

At the beginning of the source code, the contract imports unused abstract contracts as below:

- `import "@openzeppelin/contracts/token/ERC20/ERC20.sol";`
- `import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";`

RECOMMENDATION

Remove the unused imports to reduce the contract size and complexity.

UPDATES

- ***Jul 10, 2024***: The issue has been acknowledged and fixed by the SingSing Token team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.1	*Jul 10, 2024*	Public Report	Verichains Lab

Table 3. Report versions history