verichains

SECURITY AUDIT OF

SSS TOKEN

Public Report

Mar 06, 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 Mar 06, 2024. We would like to thank the Super Sushi Samurai 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 SSS 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 a vulnerable issue in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About SSS Token

SSS Token will launch with fixed-supply token used in Super Sushi Samurai to facilitate upgrades, rewards and transactions.

Following the Super Sushi Samurai, there is a 2% tax on SSS every time someone swaps SSS on Thruster. 1.6% goes to game rewards, and 0.4% goes towards sustaining dev operations.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the SSS Token.

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

SHA256 Sum	File
f63879ffd5eb49a5b0ed6b45b4f2273e738bd6716bc016f61db78066e6724eb8	contracts/ERC20.sol
78e585e7de48931131a0c04394344343c7382239745e2e4124fdffb407b23894	contracts/SSS.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

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

2. AUDIT RESULT

2.1. Overview

The SSS Token was written in `Solidity` language, with the required version to be `0.8.20`.

The contract extends `ERC20` and `Ownable`. However, the `ERC20` contract is modified to implement the `IERC20Permit`. Below table describes some properties of the audited SSS Token (as of the report writing time).

PROPERTY	VALUE
Name	SSS
Symbol	SSS
Decimals	18
Total Supply	555,555,555,555,555x10^{18} Note: the number of decimals is 18, so the total representation token will be more than 555 trillion.
DEX Supply	80%
Ecosystem Supply	5%
Booster Supply	5%
Airdrop Supply	5%
Dev Supply	5%
Buy Tax	2% (max 5%)
Sell Tax	2% (max 5%)

Table 2. The SSS Token properties

2.2. Findings

During the audit process, the audit team found a vulnerability in the given version of SSS Token.

2.2.1. LOW - Do not revoke old community/dev wallet from excluded tax and unlimited lists.

Position:

- `contracts/SSS.sol#setCommunityAddress()`
- `contracts/SSS.sol#setDevAddress()`

Description

The `setCommunityAddress` and `setDevAddress` functions are used to set the community and dev wallet addresses. The new wallet address is added to the `excludeFromTaxes` and `unlimiteds` lists. The old wallet address is not removed from them.

```
function setCommunityAddress(address community) external onlyOwner {
    communityAddress = community;
    _setExcludeFromTax(community, true);
    _setUnlimited(community, true);
}

function setDevAddress(address devTaxReceiver, address devTokenReceiver) external onlyOwner
{
    devTaxReceiverAddress = devTaxReceiver;
    devTokenReceiverAddress = devTokenReceiver;
    _setExcludeFromTax(devTaxReceiver, true);
    _setExcludeFromTax(devTokenReceiver, true);
    _setUnlimited(devTaxReceiver, true);
    _setUnlimited(devTokenReceiver, true);
}
```

RECOMMENDATION

We recommend set the old wallet address is `false` in set new address functions.

Notes: Although the `setExcludeFromTax` and `setUnlimited` is existed, but it spent more gas when execute 2 transactions instead of 1 transaction.

UPDATES

- **Mar 01, 2024**: The issue has been fixed by SSS Token team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Feb 28, 2024*	Public Report	Verichains Lab
1.1	*Mar 01, 2024*	Public Report	Verichains Lab
1.2	*Mar 06, 2024*	Public Report	Verichains Lab

Table 3. Report versions history