SECURITY AUDIT OF

LAST SURVIVOR TOKEN SMART CONTRACT

Public Report

Dec 06, 2021

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 prepared by Verichains Lab on Dec 06, 2021. We would like to thank the Last Survivor 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 Last Survivor Token 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 issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Last Survivor Token Smart Contract

Last Survivor is a metaverse project based on binance smart chain.

Your goal is to be the last on the battlefield to win the game. Last

Survivor is the newest free-to-earn battle royale game, fast and easy to play.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Last Survivor Token Smart Contract.

The audited contract is the Last Survivor Token Smart Contract that deployed on Binance Smart Chain Mainnet at address 0x74e2572a1C579B4DF80D57e9698098b255F23c9e. The details of the deployed smart contract are listed in Table 1.

FIELD	VALUE
Contract Name	LastSurvivorToken
Contract Address	0x74e2572a1C579B4DF80D57e9698098b255F23c9e
Compiler Version	v0.8.0+commit.c7dfd78e
Optimization Enabled	Yes with 200 runs
Explorer	*https://bscscan.com/address/0x74e2572a1C579B4DF80D57e9698098b255F23c9e*

Table 1. The deployed smart contract details

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 was 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 2. 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

This table lists some properties of the audited Last Survivor Token Smart Contract (as of the report writing time).

PROPERTY	VALUE
Name	Last Survivor
Symbol	LSC
Decimals	18
Total Supply	1,200,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 1,200,000,000 or 1.2 billion.

Table 3. The Last Survivor Token Smart Contract properties

2.2. Contract codes

The Last Survivor Token Smart Contract was written in Solidity language, with the required version to be ^0.8.0.

The source codes consist of two contracts and one abstract contract. Almost all source codes in the Last Survivor Token Smart Contract imported OpenZeppelin contracts.

2.2.1. ERC20 contract

This is the contract implement ERC20 token. The source code is referenced from OpenZeppelin's implementation.

2.2.2. Ownable abstract contract

Contract module which provides a basic access control mechanism, where there is an account (an owner) that can be granted exclusive access to specific functions. The source code is referenced from OpenZeppelin's implementation.

2.2.3. LastSurvivorToken contract

This is the token contract in the Last Survivor Token Smart Contract, which is extended from ERC20 contract. It also supports bot protection from another BP contract which is not in our audit scope.

2.3. Findings

During the audit process, the audit team found no vulnerability in the given version of Last Survivor Token Smart Contract.

2.4. Additional notes and recommendations

2.4.1. BPContract function INFORMATIVE

Since we do not control the logic of the bpContract, there is no guarantee that bpContract will not contain any security related issues. With the current context, in case the bpContract is compromised, there is not yet a way to exploit the Last Survivor Token Smart Contract, but we still note that here as a warning for avoiding any related issue in the future.

By the way, if having any issue, the bpContract function can be easily disabled anytime by the contract owner using the setBPEnabled function. In addition, bpContract is only used in a short time in token public sale IDO then the contract owner will disable it forever by the setBPDisableForever function.

APPENDIX

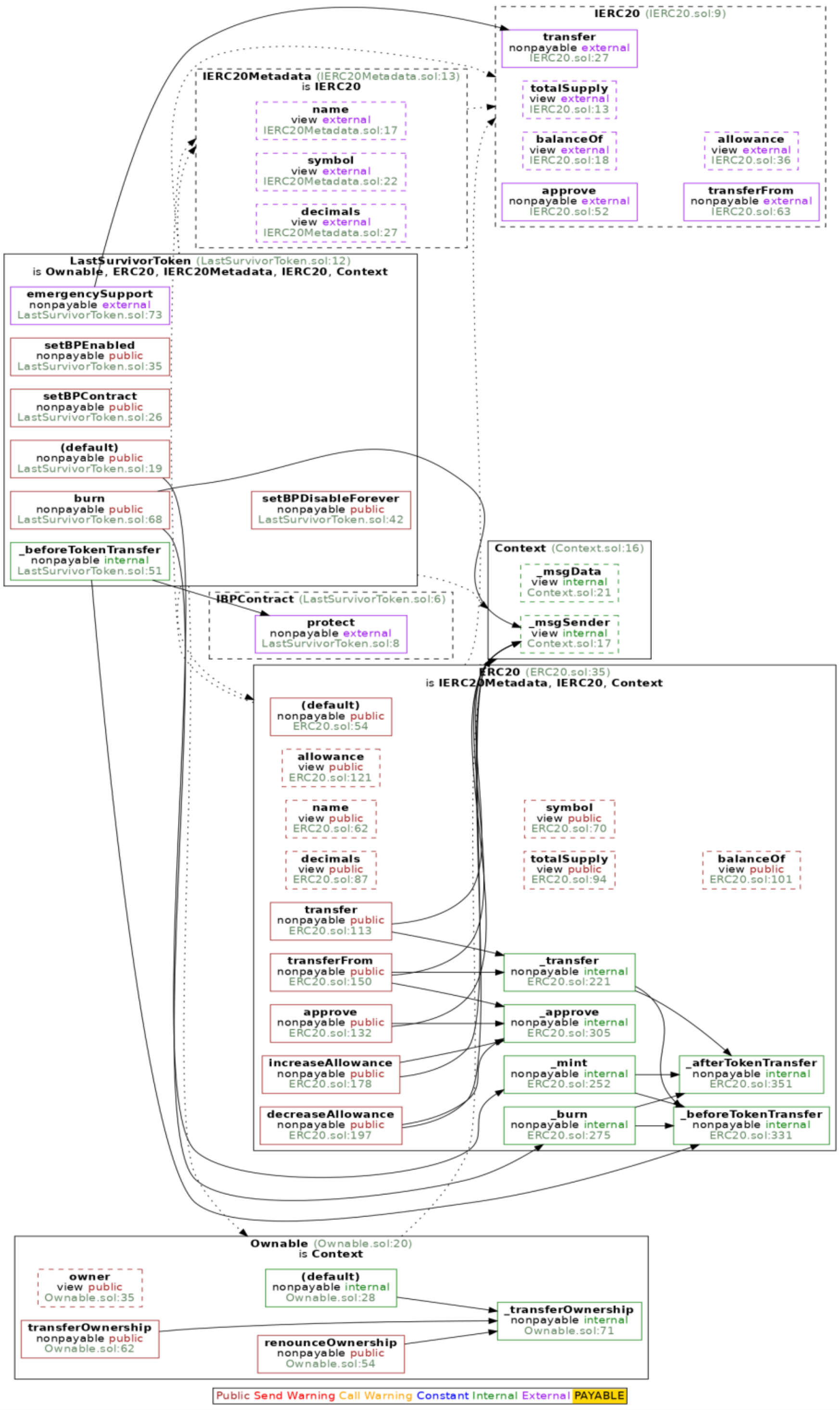
IERC20 (IERC20.sol:9)
transfer nonpayable external IERC20.sol:27
totalSupply view external IERC20.sol:13
balanceOf view external IERC20.sol:18
allowance view external IERC20.sol:36
approve nonpayable external IERC20.sol:52
transferFrom nonpayable external IERC20.sol:63
IERC20Metadata (IERC20Metadata.sol:13) is IERC20
name view external IERC20Metadata.sol:17
symbol view external IERC20Metadata.sol:22
decimals view external IERC20Metadata.sol:27
LastSurvivorToken (LastSurvivorToken.sol:12) is Ownable, ERC20, IERC20Metadata, IERC20, Context
emergencySupport nonpayable external LastSurvivorToken.sol:73
setBPEnabled nonpayable public LastSurvivorToken.sol:35
setBPContract nonpayable public LastSurvivorToken.sol:26
(default) nonpayable public LastSurvivorToken.sol:19
burn nonpayable public LastSurvivorToken.sol:68
setBPDisableForever nonpayable public LastSurvivorToken.sol:42
_beforeTokenTransfer nonpayable internal LastSurvivorToken.sol:51
Context (Context.sol:16)
_msgData view internal Context.sol:21
_msgSender view internal Context.sol:17
IBPContract (LastSurvivorToken.sol:6)
protect nonpayable external LastSurvivorToken.sol:8
ERC20 (ERC20.sol:35) is IERC20Metadata, IERC20, Context
(default) nonpayable public ERC20.sol:54
allowance view public ERC20.sol:121
name view public ERC20.sol:62
symbol view public ERC20.sol:70
decimals view public ERC20.sol:87
totalSupply view public ERC20.sol:94
balanceOf view public ERC20.sol:101
transfer nonpayable public ERC20.sol:113
transferFrom nonpayable public ERC20.sol:150
_transfer nonpayable internal ERC20.sol:221
approve nonpayable public ERC20.sol:132
_approve nonpayable internal ERC20.sol:305
increaseAllowance nonpayable public ERC20.sol:178
_mint nonpayable internal ERC20.sol:252
_afterTokenTransfer nonpayable internal ERC20.sol:351
decreaseAllowance nonpayable public ERC20.sol:197
_burn nonpayable internal ERC20.sol:275
_beforeTokenTransfer nonpayable internal ERC20.sol:331
Ownable (Ownable.sol:20) is Context
owner view public Ownable.sol:35
(default) nonpayable internal Ownable.sol:28
_transferOwnership nonpayable internal Ownable.sol:71
transferOwnership nonpayable public Ownable.sol:62
renounceOwnership nonpayable public Ownable.sol:54
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 1. Last Survivor Token Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 06, 2021*	Public Report	Verichains Lab

Table 4. Report versions history