SECURITY AUDIT OF

BIKEARN TOKEN SMART CONTRACT

Public Report

May 11, 2022

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 May 11, 2022. We would like to thank the Bikearn 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 Bikearn 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 Bikearn Token Smart Contract

Bikearn is a Web3 Exercise app based on Move-to-Earn and Game-Fi system. Users equipped with NFT Bike can earn tokens by cycling the bicycle to school, workplace, trips, or training.

1.2. Audit scope

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

The audited contract is the Bikearn Token Smart Contract that deployed on Binance Smart Chain Mainnet at address 0x83AD7F46454d28c8363035ce1f3d5eb629991ee2. The details of the deployed smart contract are listed in Table 1.

FIELD	VALUE
Contract Name	RTE
Contract Address	0x83AD7F46454d28c8363035ce1f3d5eb629991ee2
Compiler Version	v0.8.0+commit.c7dfd78e
Optimization Enabled	No with 200 runs
Explorer	*https://bscscan.com/address/0x83AD7F46454d28c8363035ce1f3d5eb629991ee2*

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

2.2. Contract codes

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

The contract extends ERC20, ERC20Burnable, Ownable and Pausable contracts. With Ownable, by default, Token Owner is contract deployer, but he can transfer ownership to another address at any time. ERC20Burnable allows token holders to destroy both their own tokens and those that they have an allowance for. Token Owner can pause/unpause contract using Pausable contract, user can only transfer tokens when contract is not paused.

The contract also implements the mint public function which allows the owner of the contract to create new tokens with a limitation that the totalSupply is lower than or equal the cap value.

Table 2 lists some properties of the audited Bikearn Token Smart Contract (as of the report writing time).

PROPERTY	VALUE
Name	RTE
Symbol	RTE
Decimals	18
Total Supply	500,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 500,000,000 or 500 million.

Table 3. The Bikearn Token Smart Contract properties

2.3. Findings

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

APPENDIX

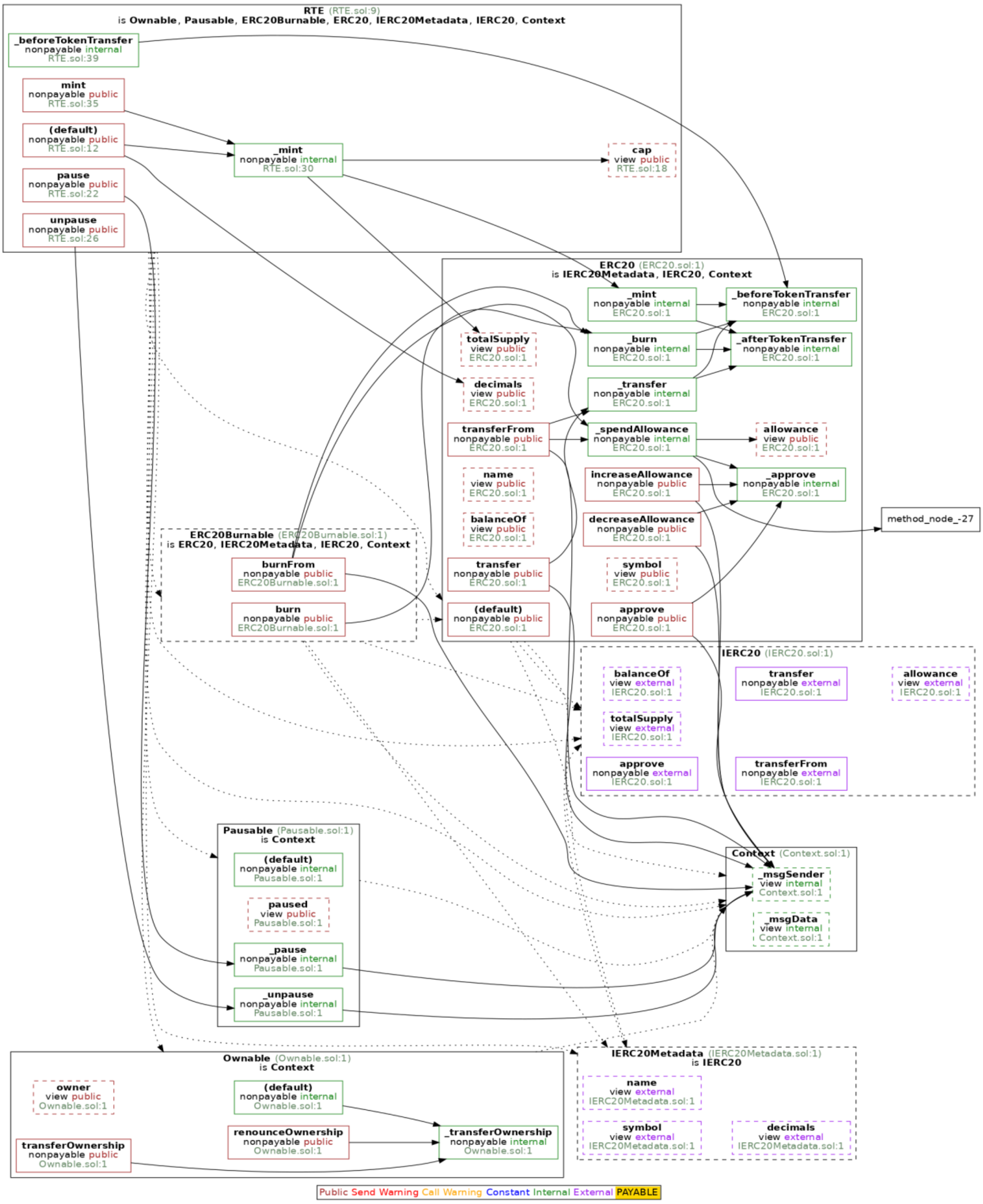
RTE (RTE.sol:9)
is Ownable, Pausable, ERC20Burnable, ERC20, IERC20Metadata, IERC20, Context
_beforeTokenTransfer nonpayable internal RTE.sol:39
mint nonpayable public RTE.sol:35
(default) nonpayable public RTE.sol:12
pause nonpayable public RTE.sol:22
unpause nonpayable public RTE.sol:26
_mint nonpayable internal RTE.sol:30
cap view public RTE.sol:18
ERC20 (ERC20.sol:1)
is IERC20Metadata, IERC20, Context
_mint nonpayable internal ERC20.sol:1
_beforeTokenTransfer nonpayable internal ERC20.sol:1
totalSupply view public ERC20.sol:1
_burn nonpayable internal ERC20.sol:1
_afterTokenTransfer nonpayable internal ERC20.sol:1
decimals view public ERC20.sol:1
_transfer nonpayable internal ERC20.sol:1
transferFrom nonpayable public ERC20.sol:1
_spendAllowance nonpayable internal ERC20.sol:1
allowance view public ERC20.sol:1
name view public ERC20.sol:1
increaseAllowance nonpayable public ERC20.sol:1
_approve nonpayable internal ERC20.sol:1
balanceOf view public ERC20.sol:1
decreaseAllowance nonpayable public ERC20.sol:1
method_node_-27
transfer nonpayable public ERC20.sol:1
symbol view public ERC20.sol:1
(default) nonpayable public ERC20.sol:1
approve nonpayable public ERC20.sol:1
ERC20Burnable (ERC20Burnable.sol:1)
is ERC20, IERC20Metadata, IERC20, Context
burnFrom nonpayable public ERC20Burnable.sol:1
burn nonpayable public ERC20Burnable.sol:1
IERC20 (IERC20.sol:1)
balanceOf view external IERC20.sol:1
transfer nonpayable external IERC20.sol:1
allowance view external IERC20.sol:1
totalSupply view external IERC20.sol:1
approve nonpayable external IERC20.sol:1
transferFrom nonpayable external IERC20.sol:1
Pausable (Pausable.sol:1)
is Context
(default) nonpayable internal Pausable.sol:1
paused view public Pausable.sol:1
_pause nonpayable internal Pausable.sol:1
_unpause nonpayable internal Pausable.sol:1
Context (Context.sol:1)
_msgSender view internal Context.sol:1
_msgData view internal Context.sol:1
Ownable (Ownable.sol:1)
is Context
owner view public Ownable.sol:1
(default) nonpayable internal Ownable.sol:1
renounceOwnership nonpayable public Ownable.sol:1
transferOwnership nonpayable internal Ownable.sol:1
transferOwnership nonpayable public Ownable.sol:1
IERC20Metadata (IERC20Metadata.sol:1)
is IERC20
name view external IERC20Metadata.sol:1
symbol view external IERC20Metadata.sol:1
decimals view external IERC20Metadata.sol:1
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 1. Bikearn Token Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*May 11, 2022*	Public Report	Verichains Lab

Table 4. Report versions history