verichains

SECURITY AUDIT OF

CRYPTOCARS SMART CONTRACT

Public Report

Dec 01, 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 01, 2021. We would like to thank the CryptoCars 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 CryptoCars 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 **5**

1.1. About CryptoCars Smart Contract **5**

1.2. Audit scope **5**

1.3. Audit methodology **6**

1.4. Disclaimer **7**

2. AUDIT RESULT **8**

2.1. Overview **8**

2.2. Contract codes **8**

2.2.1. ERC20 contract 8

2.2.2. Pausable abstract contract 8

2.2.3. Ownable abstract contract 8

2.2.4. CryptoCars contract 9

2.3. Findings **9**

3. VERSION HISTORY **11**

1. MANAGEMENT SUMMARY

1.1. About CryptoCars Smart Contract

CryptoCars is inspired by:

- Movie Cars - A 2006 American computer-animated sports comedy film produced by Pixar Animation Studios and released by Walt Disney Pictures.
- The NFT Blockchain Technology helps to prove your ownership of digital assets.

We know that many of us, especially man players, love role-playing games and intensive racing matches with other players. We do too, this is why we create the CryptoCars Blockchain-based game to make your cars more unique and special from others. Going along with that is the diverse racing mode for you to enjoy every moment with CryptoCars from Virtual Race, Players vs. Computers, Players vs. Players, Tournaments.

1.2. Audit scope

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

The audited contract is the CryptoCars Smart Contract that deployed on Binance Smart Chain Mainnet at address 0x50332bdca94673f33401776365b66cc4e81ac81d. The details of the deployed smart contract are listed in Table 1.

FIELD	VALUE
Contract Name	CryptoCars
Contract Address	0x50332bdca94673f33401776365b66cc4e81ac81d
Compiler Version	v0.8.4+commit.c7e474f2
Optimization Enabled	No with 200 runs
Explorer	*https://bscscan.com/address/0x50332bdca94673f33401776365b66cc4e81ac81d*

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 CryptoCars Smart Contract (as of the report writing time).

PROPERTY	VALUE
Name	CryptoCars
Symbol	CCAR
Decimals	18
Total Supply	100,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 100,000,000 or 100 million.

Table 3. The CryptoCars Smart Contract properties

2.2. Contract codes

The CryptoCars Smart Contract was written in Solidity language, with the required version to be 0.8.2.

The source codes consist of two contracts, two abstract contracts. Almost all source codes in the CryptoCars 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. Pausable abstract contract

Contract module allows children to implement an emergency stop mechanism that can be triggered by an authorized account. The source code is referenced from OpenZeppelin's implementation.

2.2.3. Ownable abstract contract

This abstract contract makes the CryptoCars Smart Contract ownable. The source code is referenced from OpenZeppelin's implementation.

2.2.4. CryptoCars contract

This is the main contract in the CryptoCars Smart Contract, which extends ERC20, Pausable, Ownable abstract contract.

2.3. Findings

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

APPENDIX

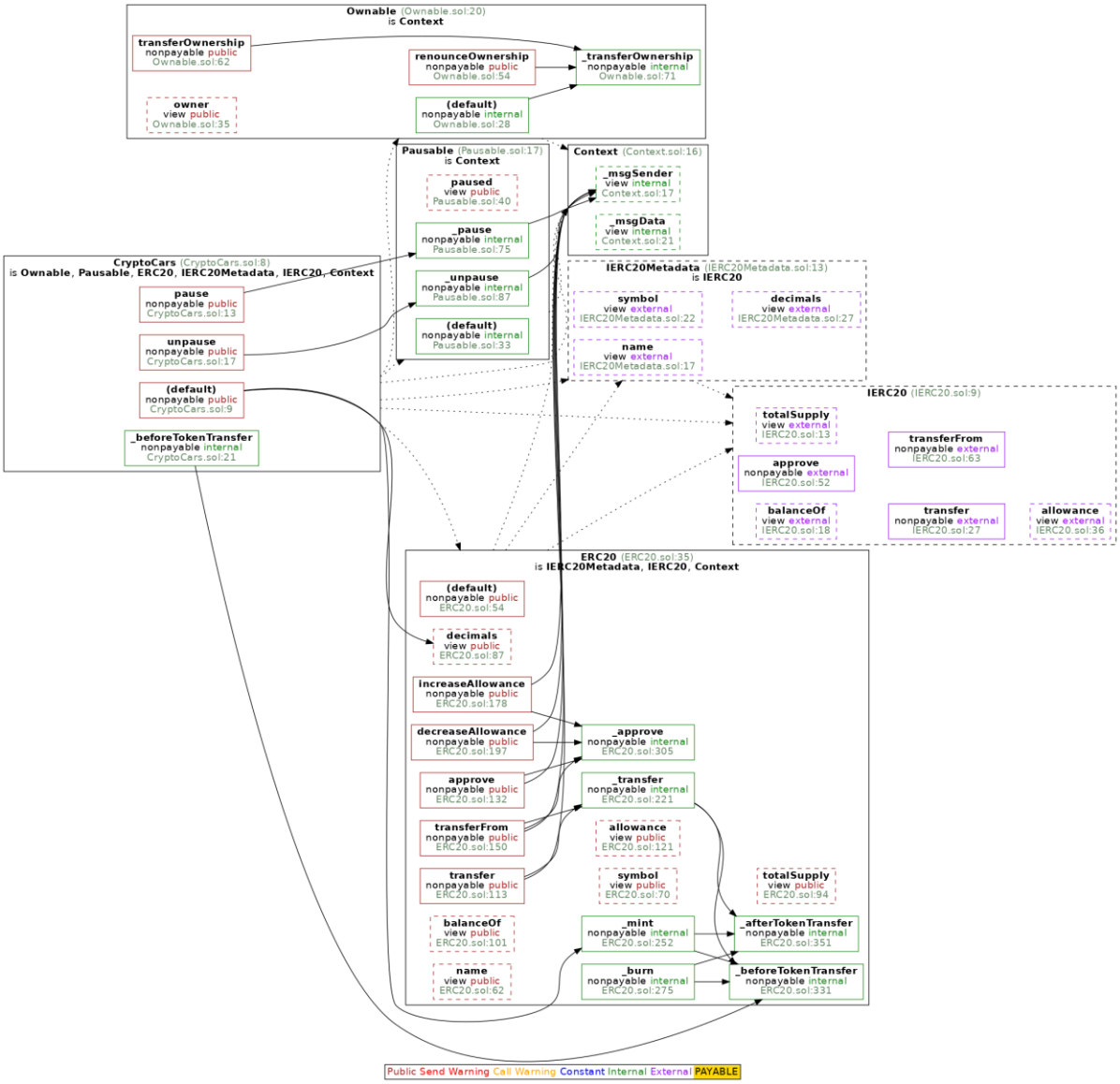

Image 1. CryptoCars Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Nov 29, 2021*	Public Report	Verichains Lab
1.1	*Dec 01, 2021*	Public Report	Verichains Lab

Table 4. Report versions history