SECURITY AUDIT OF

KART BOX SMART CONTRACT

Public Report

Mar 22, 2023

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 22, 2023. We would like to thank the Kart Box 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 Kart Box 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 issue in the contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Kart Box Smart Contract

Kart Box is the first 3D NFT P2E battle racing game built on BSC, inspired by the well-known PikaLong comic series with millions of fans worldwide.

1.2. Audit scope

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

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

SHA256 Sum	File
fde85516a7caacf4611f93082239fed9248ea1c658e95589ab9a89c4155e66f5	KartBox.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

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

The Kart Box Smart Contract was written in `Solidity` language, with the required version to be `^0.8.8`. The source code was written based on OpenZeppelin's library.

2.1.1. KartBox.sol contract

The `KartBox` contract extends `AccessControlEnumerable`, `ERC20Pausable`, `ERC20Capped` and `ERC20Burnable` contracts. With `AccessControlEnumerable`, by default, the contract deployer has `DEFAULT_ADMIN_ROLE`, `MINTER_ROLE`, `PAUSER_ROLE` and `MAINTAINER_ROLE` roles. The user has `PAUSER_ROLE` can pause/unpause contract using `ERC20Pausable` contract, and users can only transfer tokens when the contract is not paused. The user has `MAINTAINER_ROLE` can enable/disable and update the `BotProtector` contract address. `ERC20Burnable` allows token holders to destroy both their own tokens and those that they have an allowance for.

2.2. Findings

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

2.3. Additional notes and recommendations

2.3.1. BotProtector contract INFORMATIVE

Since we do not control the logic of the `BotProtector` contract, there is no guarantee that `BotProtector` will not contain any security related issues. With the current context, in case the `BotProtector` contract is compromised, there is not yet a way to exploit the Kart Box 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 function `protect()` in the `BotProtector` contract can be easily disabled at any time by a user with the `MAINTAINER_ROLE` using the `enableBP()` function.

APPENDIX

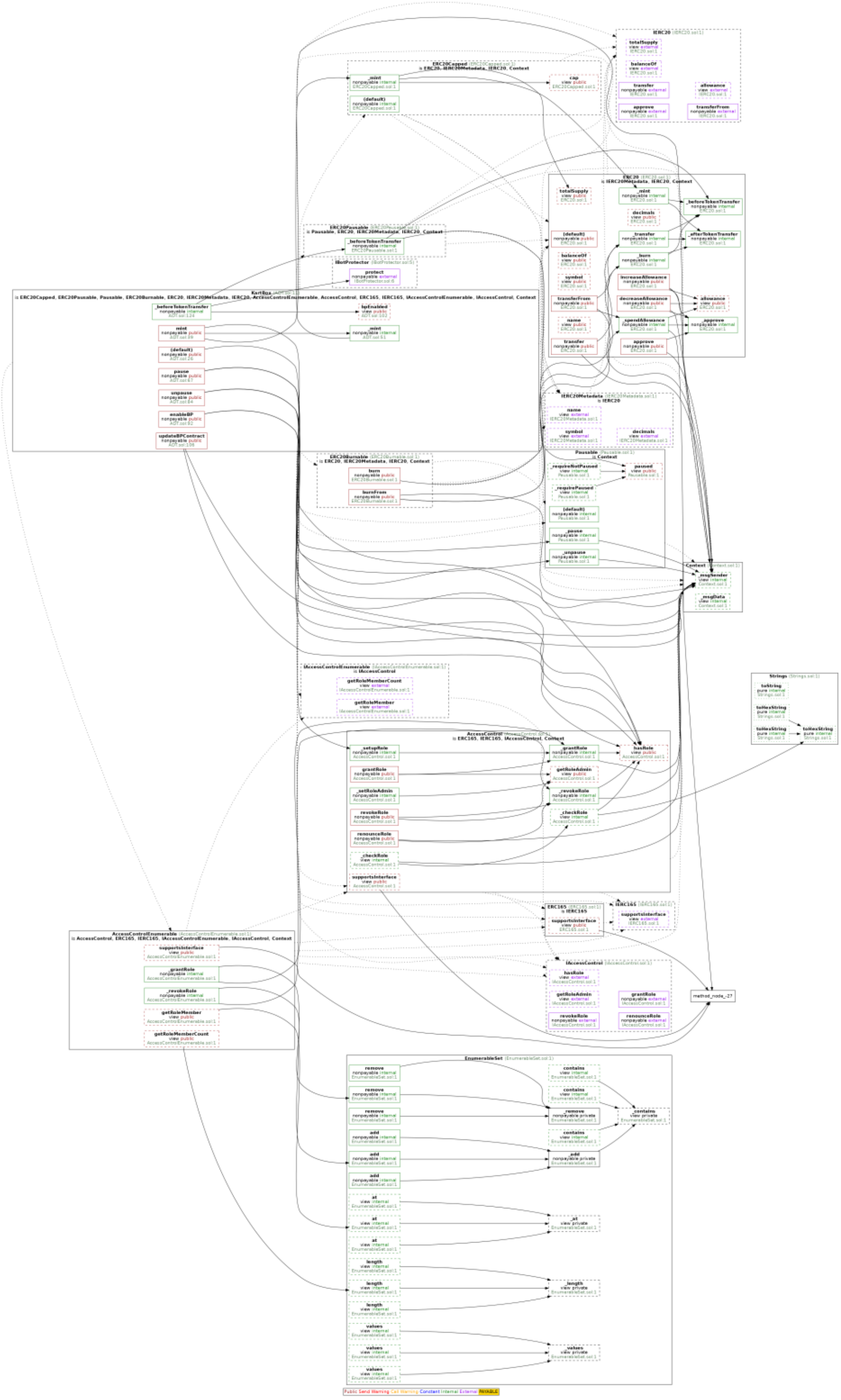

Image 1. Kart Box Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Mar 22,2023*	Public Report	Verichains Lab

Table 2. Report versions history