SECURITY AUDIT OF

MULTIPLIER BOX CONTRACT

Public Report

Apr 20, 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 Apr 20, 2023. We would like to thank the Billion 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 Multiplier Box 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 one vulnerable issue in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Multiplier Box contract

Billion Box is a decentralized gaming platform that offers users the opportunity to win big rewards by participating in a range of exciting prediction-based games. Our platform uses blockchain technology and smart contracts to ensure that all games are transparent, secure, and tamper-proof. We have three main games, including Prediction Classic, Decimal Box, and BB Max, each with unique rules and rewards.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of Multiplier Box contract.

It was conducted on commit `c4de23f10fc6aea54413cc1cb6ba23277fd3e3a4` from git repository *https://github.com/billion-box/multiplier-contracts*.

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

SHA256 Sum	File
d782a3a724d19d8b990431ca093517056c26ccd1bf7a8ffadfc88b022e698399	contracts/MultiplierBox.sol
ad5a797d7cf4630d52000db11119a27ebb786f2311c70777afafd49556a1529e	contracts/games/HighLowGame.sol
dbc307a50f7f9dd8a9a3a534f17950106c4026cb4e1200c9b7133d1acca9f285	contracts/games/IGame.sol
23d6aad7f5b58040af710d3510a87504f37ea2901539e746f481ab01035575c2	contracts/games/LotteryGame.sol
bb9da2c02e77618a7de82a0ba9db6addc1765405cfae9066a854b2bb8880d041	contracts/games/OddEvenGame.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 Multiplier Box contract was written in Solidity language, with the required version to be ^0.8.9

2.1.1. MultiplierBox contract

This contract enables a number guessing game where users can participate in a pool by submitting a number and some asset (such as whitelist tokens or native coins). The contract operator has the duty of updating the winning number each round. The users who matched the winning number can claim their reward, which is a portion of the total assets in the contract.

The contract uses withdrawReserve mechanism which allows admin to withdraw token from the contract in two steps.

2.2. Findings

During the audit process, the audit team found one vulnerability issue in the given version of the Multiplier Box contract.

2.2.1. Missing init ReentranceGuard contract in `initialize` function MEDIUM

MultiplierBox contract is a proxy contract implementation that uses ReentranceGuard to prevent reentrancy attacks. However, the contract does not initialize ReentranceGuard, which may cause the contract to malfunction if the ReentranceGuard state conflicts with the previous logic state of the proxy contract.

RECOMMENDATION

Add ReentrancyGuard_init statement to the initialize function.

UPDATES

- *Apr 20, 2023*: This issue has been acknowledged and fixed by the Billion Box team.

2.3. Additional notes and recommendations

2.3.1. Use `calldata` instead of `memory` for gas saving INFORMATIVE

In external function with array arguments, using memory will force solidity to copy that array to memory thus wasting more gas than using directly from calldata.

```
function playNativeToken(uint256[] memory amounts, uint256[] memory numbers, address game,
uint256 roundId)
```

RECOMMENDATION

Change `memory` to `calldata` for gas saving in all external functions.

```
function playNativeToken(uint256[] calldata amounts, uint256[] calldata numbers, address
game, uint256 roundId)
```

UPDATES

- *Apr 20, 2023*: This issue has been acknowledged and fixed by the Billion Box team.

2.3.2. Centralization Related Risks INFORMATIVE

In the MultiplierBox, the role `DEFAULT_ADMIN_ROLE` and `OPERATOR_ROLE` have control some important features so any compromise to the these accounts may allow hackers to exploit the system.

RECOMMENDATION

We should use Multi sign or DAO mechanism to manage the admin and operator account.

UPDATES

- *Apr 20, 2023*: This issue has been acknowledged by the Billion Box team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Apr 12, 2023*	Private Report	Verichains Lab
1.1	*Apr 20, 2023*	Public Report	Verichains Lab

Table 2. Report versions history