SECURITY AUDIT OF

DEFIHORSE IBCO SMART CONTRACT

Public Report

Mar 08, 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 Mar 08, 2022. We would like to thank the DeFiHorse 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 DeFiHorse IBCO 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 some vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About DeFiHorse IBCO Smart Contract

The DeFiHorse is a metaverse e-sport game based on Blockchain technology and NFTs, empowering the players and creators to the next level of the horse race.

The game brings you the gorgeous legendary horses to enter the limitless cyberpunk horserace. Overwhelming yourself in every minute of the battle, using your personal and teamwork skills to earn token rewards, and you will end up with a 1,000,000 dollar prize pool.

The DefiHorse IBCO is the smart contract used to support the release of tokens in the Initial Bonding Curve Offering.

1.2. Audit scope

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

It was conducted on commit c1841780f6aa49871a6589abf2c3b1d1f9bab1ea from git repository *https://github.com/DefiHorse/IBCO-Event-Smartcontract/*.

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

SHA256 Sum	File
67f5be60db1e7499d6e2cd7b833cc8af2ec2e3f79de70acc224e3462efb3959c	**DefiHorseIBCO.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 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 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 DeFiHorse IBCO Smart Contract was written in Solidity language, with the required version to be ^0.8.2. The source code was written based on OpenZeppelin's library.

The main contract in the audit scope is DefiHorseIBCO, which implements the Initial Bonding Curve Offering (IBCO) for DefiHorse Token. The IBCO process can be summarized as below:

- The IBCO event will last for 3 days, users will need to deposit BUSD tokens to join the IBCO event.
- After 30 minutes from the end of the IBCO event (3 days + 30 minutes from the start), users can claim their DefiHorse tokens based on their share amount of BUSD with the price settled by the bonding curve.

The DefiHorseIBCO contract extends the Ownable contract from OpenZeppelin. With Ownable, by default, Contract Owner is also the contract deployer, but he can transfer ownership to another address at any time.

Based on the current logic of the IBCO smart contract, Contract Owner can withdraw all the BUSD tokens to their wallet after the IBCO event ends (3 days from the start). Moreover, after 30 days from the end of the IBCO event (33 days from the start), all the remaining DefiHorse tokens may be returned to the contract owner so that users cannot claim anymore.

2.2. Findings

During the audit process, the audit team found some vulnerabilities in the given version of DeFiHorse IBCO Smart Contract.

DeFiHorse fixed the code, according to Verichains's draft report, in commit dba0994c1223749566de787b6a1a8809f1ff76ca.

2.2.1. DefiHorseIBCO.sol - Wrong formula for takeBackPercentage calculation CRITICAL

In the _withdrawCap function, the formula to calculate takeBackPercentage is wrong. Based on the provided description, the value of the takeBackPercentage variable should be range from 70% to 3% when the userAccumulated amount is changed from 2,000 to 200,000. However, when we try to input the value of 200,000 * DECIMALS to the userAccumulated parameter, we got a negative number result (approximately -64%).

```
function _withdrawCap(uint256 userAccumulated) internal pure returns (uin…
  t256 withdrawableAmount) {
    if (userAccumulated <= MINIMAL_USER_AMOUNT) {
        return (userAccumulated * 70) / 100;
    }

    if (userAccumulated <= THRESHOLD_USER_AMOUNT) {
        uint256 accumulatedTotal = userAccumulated / DECIMALS;
        uint256 baseNum = (accumulatedTotal*1709)/(10**9);
        uint256 secondNum = (68*accumulatedTotal)/100;
        uint256 takeBackPercentage = (baseNum + 71360 - secondNum) / 1000…
  ;
        return (userAccumulated * takeBackPercentage) / 100;
    }

    if (userAccumulated > THRESHOLD_USER_AMOUNT){
        return (userAccumulated * 3) / 100;
    }
}
```

RECOMMENDATION

The dev team should review and fix the above formula.

UPDATES

- *Mar 08, 2022*: This issue has been acknowledged and fixed by the DeFiHorse team.

2.3. Additional notes and recommendations

2.3.1. DefiHorseIBCO.sol - Missing token transfer result checking in deposit function INFORMATIVE

In the transfer function of the DefiHorseIBCO contract, the transfer result is not checked. So, if the transferFrom function of the BUSD token does not revert when the transferring is failed, users may get free DEFIHORSE tokens.

```
function deposit(uint256 amount) external{
    require(START <= block.timestamp, "The offering has not started yet")…
  ;
    require(block.timestamp <= END, "The offering has already ended");
    require(DEFIHORSE.balanceOf(address(this)) == TOTAL_DISTRIBUTE_AMOUNT…
  , "Insufficient DEFIHORSE token in contract");
    require(BUSD.allowance(msg.sender, address(this)) >= amount, 'Caller …
  must approve first');
```

```
    // grab the tokens from msg.sender.
    BUSD.transferFrom(msg.sender, address(this), amount); // using safeTr…
  ansferFrom instead
    totalProvided += amount;
    provided[msg.sender] += amount;
    accumulated[msg.sender] = Math.max(accumulated[msg.sender], provided[…
  msg.sender]);
    emit Deposit(msg.sender, amount);
}
```

RECOMMENDATION

We should use the safeTransferFrom function from SafeERC20 library instead of transferFrom function when transferring token.

UPDATES

- *Mar 08, 2022*: This issue has been acknowledged and fixed by the DeFiHorse team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Mar 08, 2022*	Public Report	Verichains Lab

Table 2. Report versions history