verichains

SECURITY AUDIT OF

GOHORSE TOKEN SMART CONTRACT

Public Report

Mar 09, 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 09, 2022. We would like to thank the GoHorse 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 GoHorse 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 issue in the smart contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About GoHorse Token Smart Contract

GoHorse is a NFT game.

GoHorse contract is an ERC20 token that GoHorse players can use in the game.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the GoHorse Token Smart Contract. It was conducted on the source code provided by the GoHorse team.

The following file were made available in the course of the review:

FILE	SHA256 SUM
GoHorse.sol	a889756f94f66583b9c82a20927db6c0f72d5fc0149a9caf923ca16214e1ccfa

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

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

PROPERTY	VALUE
Name	Go Horse
Symbol	GOHORSE
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 2. The GoHorse Token Smart Contract properties

2.2. Contract codes

The GoHorse Token Smart Contract was written in Solidity language, with the required version to be 0.8.11. Almost all source codes in the GoHorse Token Smart Contract are imported from OpenZeppelin contracts.

The contract is a custom ERC20 token. Compared with a normal ERC20 token, the contract has a different logic in the _transfer function. The _transfer function allows the contract to take a fee amount from users every they interact with the pancakeswap.pair token. If the tokenBalance of this contract is greater than the threshold value, the transfer transaction may sell these tokens and use the received BNB to add liquidity.

The contract extends Ownable and Pausable contracts. With Ownable, by default, Token Owner is the contract deployer but he can transfer ownership to another address at any time. He can pause/unpause the contract using the Pausable contract, the user can only transfer unlocked tokens and only when the contract is not paused.

2.3. Findings

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

2.4. Additional notes and recommendations

2.4.1. The logic of addLiquidity flow allow users front-running INFORMATIVE

The contract uses numTokensToSwap variable like a threshold. If the tokenBalance of this contract is greater than numTokensToSwap value, the transfer transaction may sell these tokens and use the received BNB to add liquidity. The users may observe the tokenBalance variable to predict the sell token transaction and sell their tokens before the transaction call to take advantage.

RECOMMENDATION

We suggest calculating and setting the numTokensToSwap variable with an appropriate value to prevent the front-running action.

UPDATES

- *Mar 09, 2022*: This issue has been acknowledged and fixed by the GoHorse team in commit 0e384ee275672684099d075d8b832635d7ff3175.

APPENDIX

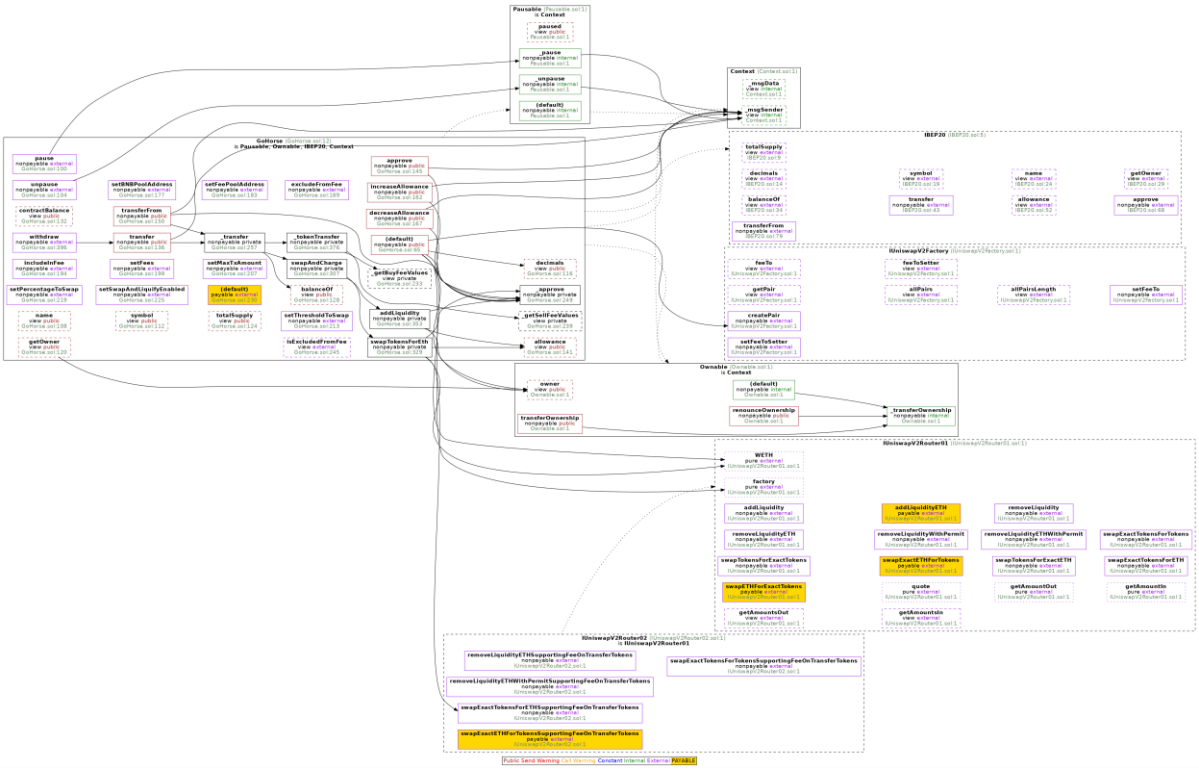

Image 1. GoHorse Token Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Mar 04, 2022*	Public Report	Verichains Lab
1.1	*Mar 09, 2022*	Public Report	Verichains Lab

Table 3. Report versions history