verichains

SECURITY AUDIT OF

WINGSWAP VESTING CONTRACT

Public Report

Nov 29, 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 Nov 29, 2021. We would like to thank the WingSwap 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 WingSwap Vesting 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 found no vulnerability in the given version of WingSwap Vesting Contract.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About WingSwap Vesting Contract

WingSwap is the first DeFi platform that provides both AMM and intergrated NFT Farming functionalities on Fantom.

They provide automated, blazing fast and all-round solutions for the DeFi industry.

1.2. Audit scope

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

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 initial review was conducted on Nov 26, 2021 and a total effort of 3 working days was dedicated to identifying and documenting security issues in the code base of the WingSwap Vesting Contract.

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

FILE	SHA256 SUM
Vesting.sol	f13aa1ebfa3d40a515ef5e92449f17a470a2af17207d0d222bc568534c4caf0a

2.2. Contract code

The WingSwap Vesting Contract was written in Solidity language, with the required version to be 0.6.12. The source code was written based on OpenZeppelin's library.

The contract is used to create vesting schedules for users. The token distribution process can be summarized as below:

- Before TGE, all tokens are locked in the vesting contract.
- At TGE, all the TGE amount of tokens will be unlocked and claimable.
- Once the cliff time is reached, the remaining locked tokens will be released linearly and can be claimed at any point of time.

2.3. Findings

During the audit process, the audit team found no vulnerability in the given version of the WingSwap Vesting Contract.

2.4. Additional notes and recommendations

2.4.1. Outdated version of Solidity INFORMATIVE

The required version of Solidity in the WingSwap Vesting Contract source code is 0.6.12, which is outdated.

Updating the Solidity version from 0.6.12 to 0.8.10 will introduce many features and fix some bugs like:

- Automatic integer overflow/underflow checking for arithmetic operations.
- Allow overrides to have stricter state mutability: view can override nonpayable and pure can override view.

- Disallow public state variables overwriting pure functions.

UPDATES

- *2021-11-29*: This issue has been acknowledged by the WingSwap team.

2.5. Compatible chains

All the contracts providing in this audit is compatible with EVM-based chains, including Ethereum, Binance Smart Chain, and Fantom.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*2021-11-29*	Public Report	Verichains Lab

Table 2. Report versions history