verichains

SECURITY AUDIT OF

WORLD OF DEFISH COIN VESTING SMART CONTRACT

Public Report

Mar 16, 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 16, 2022. We would like to thank the World of Defish 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 World of Defish Coin Vesting 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 issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About World of Defish Coin Vesting Smart Contract

World of Defish is a decentralized NFT gaming universe running on BSC where players immerse themselves in the delightful underwater universe.

With the care for best gaming quality in mind, World of Defish offers players the astonishing NFT world where they can set off their journey to the seven seas to look for the desired NFT fish.

Brave fishermen can upgrade their equipment and skills to improve fishing production. If the Fishermen don't have too much time on their hands, buying territories to earn passive income would surely be the way to go. Competitions and trading are also available to those who want to maximize their hard-earned profit. World of Defish has got you all covered!

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the World of Defish Coin Vesting Smart Contract.

It was conducted on commit e6cde32a396203ab3b316e43f984ac8457991455 from git repository *https://github.com/wodtech/contracts/*.

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

SHA256 SUM	FILE
05847d5041507fca72d09c93d9037f3792ad489e2a351c0ee588b21d258de83a	CoinVesting.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 World of Defish Coin Vesting Smart Contract was written in Solidity language, with the required version to be 0.8.4. It is a token holder contract that will allow a beneficiary to extract the tokens after a given release time.

2.2. Findings

During the audit process, the audit team found no vulnerability in the given version of World of Defish Coin Vesting Smart Contract.

2.3. Additional notes and recommendations

2.3.1. Possible issue when using third party token contract INFORMATIVE

Contract uses transfer and transferFrom to send/receive tokens without require. If the token_contract is third party token contract, and it returns false instead of revert (like the OpenZeppelin erc20 implement in WOD token) when transfer failed, beneficiary will not receive tokens but the vesting is still mark as released. It's the same with lock, vesting contract will not receive tokens but it is still mark as locked and user can claim free tokens when it is released.

Another problem with third party token contract is that if it is ERC777 token contract. Since ERC777 supports tokensToSend and tokensReceived hooks (*https://docs.openzeppelin.com/contracts/2.x/api/token/erc777#IERC777Recipient*), attacker can deploy a smart contract with these hooks as a receiver and make a reentrancy attack by calling release (locks[_lock_id].is_released = true is not set at this time) again and again, drain all the remaining tokens in the contract.

```
function lock(address _beneficiary, uint _amount, uint64 _release_time) p…
  ublic {

    require(_release_time > block.timestamp, 'You cannot set the release …
  time retroactively');
    require(_amount <= token_contract.balanceOf(msg.sender), 'Not enought…
   balance');

    token_contract.transferFrom(msg.sender, address(this), _amount);

    Lock storage _lock = locks.push();
    _lock.beneficiary = _beneficiary;
    _lock.amount = _amount;
```

```
    _lock.release_time = _release_time;
    _lock.exists = true;

    uint _id = locks.length - 1;

    beneficiary_locks[_beneficiary].push(_id);
}

function release(uint _lock_id) public {

require(locks[_lock_id].exists, 'Lock does not exist');
require(block.timestamp >= locks[_lock_id].release_time, 'It is not time …
  yet');
require(!locks[_lock_id].is_released, 'Already released');

token_contract.transfer(locks[_lock_id].beneficiary, locks[_lock_id].amou…
  nt);

locks[_lock_id].is_released = true;
}
```

RECOMMENDATION

If the vesting contract does support third party tokens.Use require or SafeERC20 when transferring tokens.To avoid reentrancy attack with ERC777, use the Checks-Effects-Interactions Pattern and/or ReentrancyGuard.

UPDATES

- *Mar 16, 2022*: This issue has been acknowledged by the World of Defish team.

APPENDIX

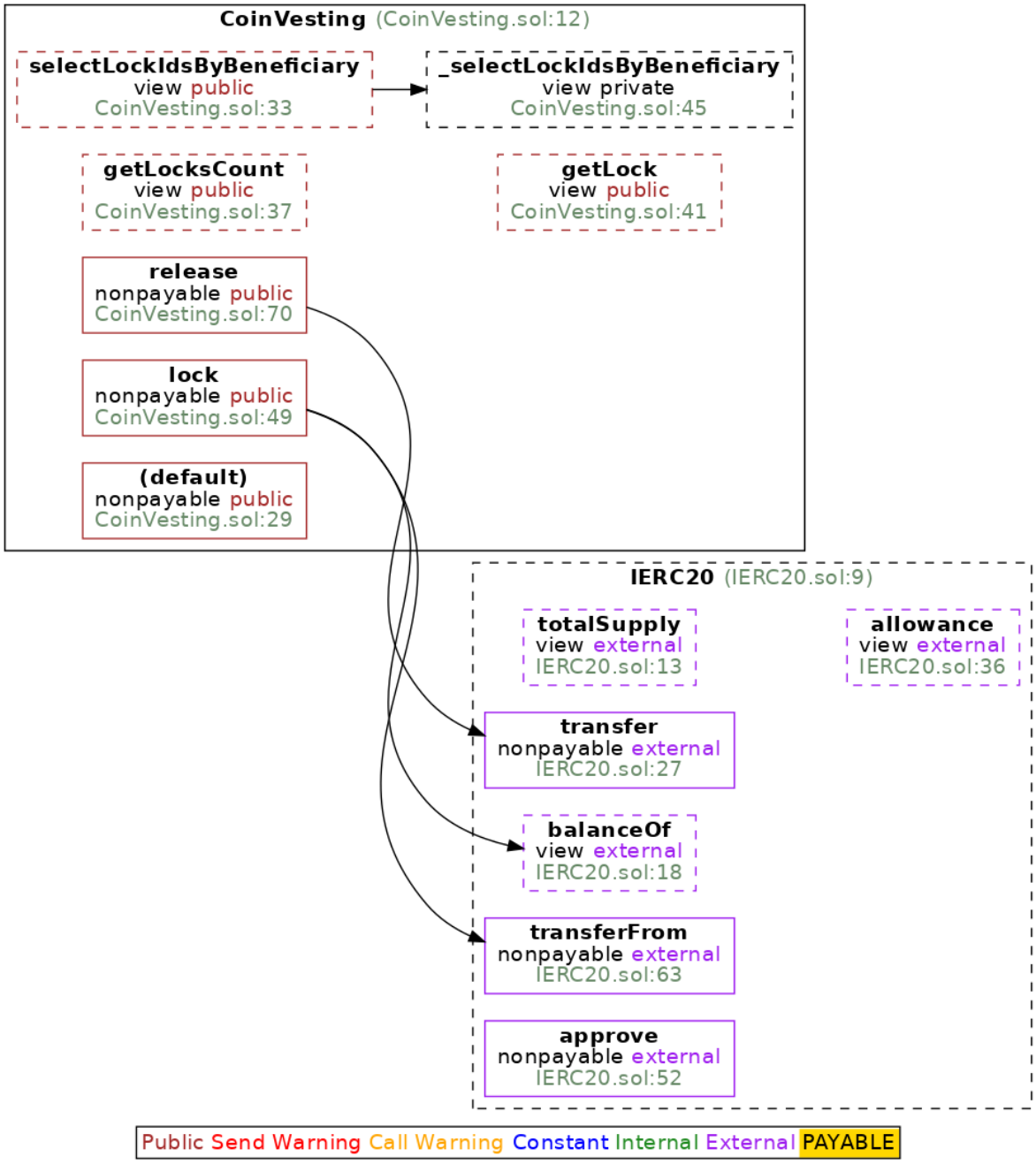

Image 1. World of Defish Coin Vesting Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 16, 2021*	Private Report	Verichains Lab
1.1	*Mar 16, 2022*	Public Report	Verichains Lab

Table 2. Report versions history