verichains

SECURITY AUDIT OF

HIMOWORLD SMART CONTRACTS

Public Report

Dec 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 Dec 29, 2021. We would like to thank the HimoWorld 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 HimoWorld Smart Contracts. 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 contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About HimoWorld Smart Contracts

Himo World is an NFT game with a Play-to-Earn feature, in which players can engage in battles with others, build their team to their favourite to explore the universe, or choose to become observers of the war. This game is also providing a true Free-to-Play experience, which a lot of other NFT games in the market is lacking at the moment. Setting up in a distant realm, the player just awoke his summoning power, and realize they now are in the middle of the war between realms, where they fight against each other in a battle of strengths and wits.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the token and vesting part of the HimoWorld Smart Contracts. It was conducted on the source code provided by the HimoWorld team.

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

SHA256 SUM	FILE
d6b93daf385ed22a78cfb5bbf8626a0775e310044f781ec9b4a5d98f04777f9d	BPContract.sol
cccbc5ad2abf6038a60d3487f8b2911ba11b0c99f4240e85b76d3804b10967c5	HIMOToken.sol
42aab4adeea4e9529a0069763a4105b071f525b24bde9061b928092b580a7509	Investing.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 HimoWorld Smart Contracts was written in Solidity language, with the required version to be ^0.8.3.

2.1.1. HIMOToken contract

HIMOToken contract extends ERC20, Pausable and Ownable contracts. With Ownable, by default, Token Owner is contract deployer but he can transfer ownership to another address at any time. He can pause/unpause contract using Pausable contract, user can only transfer unlocked tokens and only when contract is not paused.

The contract also supports bot protection by BPContract, which will make limit on buy/sell, add/remove blacklist, whitelist addresses and prevent sandwich attack. The owner can enable/disable bot protection at any time.

2.1.2. Investing contract

This is the vesting contract in the HimoWorld Smart Contracts, which extends OwnableUpgradeable and PausableUpgradeable contract. With OwnableUpgradeable, by default, Token Owner is contract deployer, but he can transfer ownership to another address at any time. He can pause/unpause contract using PausableUpgradeable contract, user can only claim tokens when contract is not paused.

The owner can change beneficiary address of a vesting plan to a new one. He can also withdraw all tokens from this contract (in case of emergency situation).

2.2. Findings

This section contains a detailed analysis of all the vulnerabilities that were discovered by the audit team during the audit process.

HimoWorld fixed the code, according to Verichains's private report, in commit 837292d5d6a9fb4a1265177d1295b57de0a3dc7e.

2.2.1. BPContract.sol - Wrong listingTime when transfer zero amount HIGH

In protect function, if someone transfers 0 token to lpPair before adding LP, listingTime will be set and can not be reset so fastestValidIn feature of bot protection will not work as expected.

```
function protect(
    address from,
    address to,
```

```
    uint256 amount
) external onlyToken {
    ...
    if (to == lpPair && listingTime == 0) {
        listingTime = block.timestamp;
    }
    ...
    if (listingTime > 0 && !fastestBuyerExists[to]) {
        require(block.timestamp >= listingTime + fastestValidIn);
    }
    ...
}

function _beforeTokenTransfer(
    address from,
    address to,
    uint256 amount
) internal virtual override {
    super._beforeTokenTransfer(from, to, amount);

    require(!paused(), "ERC20Pausable: token transfer while paused");

    if (bpEnabled) {
        BP.protect(from, to, amount);
        emit BPTransfer(from, to, amount);
    }
}
```

RECOMMENDATION

Adding amount > 0 check.

```
function protect(
    address from,
    address to,
    uint256 amount
) external onlyToken {
    ...
    if (to == lpPair && listingTime == 0 && amount > 0) {
        listingTime = block.timestamp;
    }
    ...
}
```

UPDATES

- *Dec 29, 2021*: This issue has been acknowledged and fixed by the HimoWorld team.

2.2.2. Investing.sol - Low version of solidity does not support custom:oz-upgrades-unsafe-allow INFORMATIVE

Initializable is using custom:oz-upgrades-unsafe-allow and it's only supported from solidity version 0.8.3 and above.

```
pragma solidity ^0.8.0;
```

RECOMMENDATION

Changing the solidity version to ^0.8.3.

```
pragma solidity ^0.8.3;
```

UPDATES

- *Dec 29, 2021*: This issue has been acknowledged and fixed by the HimoWorld team.

2.2.3. 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. Unless you want to write to the variable, always using calldata for external function.

```
// BPContract.sol
function multiRemoveFromBlackList(address[] memory addressesToBlackList)
function multiAddFastestBuyer(address[] memory addressToFastestBuyer)
function multiAddWhiteList(address[] memory addressToWhiteList)
function multiRemoveFromWhiteList(address[] memory addressesToWhiteList)

// Investing.sol
function addMultiBeneficicary(VestingInfo[] memory infors)
```

RECOMMENDATION

Changing memory to calldata for gas saving in all external functions.

```
function multiRemoveFromBlackList(address[] calldata addressesToBlackList)
function multiAddFastestBuyer(address[] calldata addressToFastestBuyer)
function multiAddWhiteList(address[] calldata addressToWhiteList)
function multiRemoveFromWhiteList(address[] calldata addressesToWhiteList)
```

```
// Investing.sol
function addMultiBeneficicary(VestingInfo[] calldata infors)
external
onlyOwner
{
    for (uint256 i = 0; i < infors.length; i++) {

    _addBeneficicary(
        infors[i].beneficiary,
        infors[i].genesisTimestamp,
        infors[i].totalAmount,
        infors[i].participant,
        infors[i].cliff,
        infors[i].period,
        infors[i].duration,
        infors[i].tgeAmount
    );

}
}
```

UPDATES

- *Dec 29, 2021*: This issue has been acknowledged and fixed by the HimoWorld team.

2.2.4. Investing.sol - PausableUpgradeable is extended but not init INFORMATIVE

The contract extended PausableUpgradeable but it is not init.

```
function initialize(address _token) public initializer {
    require(
        _token != address(0),
        "Vesting::constructor: _token is the zero address!"
    );

    token = IERC20Upgradeable(_token);
    __Ownable_init();
}
```

RECOMMENDATION

Adding __Pausable_init() to initialize function.

```
function initialize(address _token) public initializer {
    require(
        _token != address(0),
        "Vesting::constructor: _token is the zero address!"
    );

    token = IERC20Upgradeable(_token);
    __Ownable_init();
    __Pausable_init();
}
```

UPDATES

- *Dec 29, 2021*: This issue has been acknowledged and fixed by the HimoWorld team.

2.2.5. Investing.sol - Typo in Beneficicary INFORMATIVE

There are some typo in some functions with Beneficicary, the correct should be Beneficiary.

RECOMMENDATION

Fixing the typo

UPDATES

- *Dec 29, 2021*: This issue has been acknowledged and fixed by the HimoWorld team.

APPENDIX

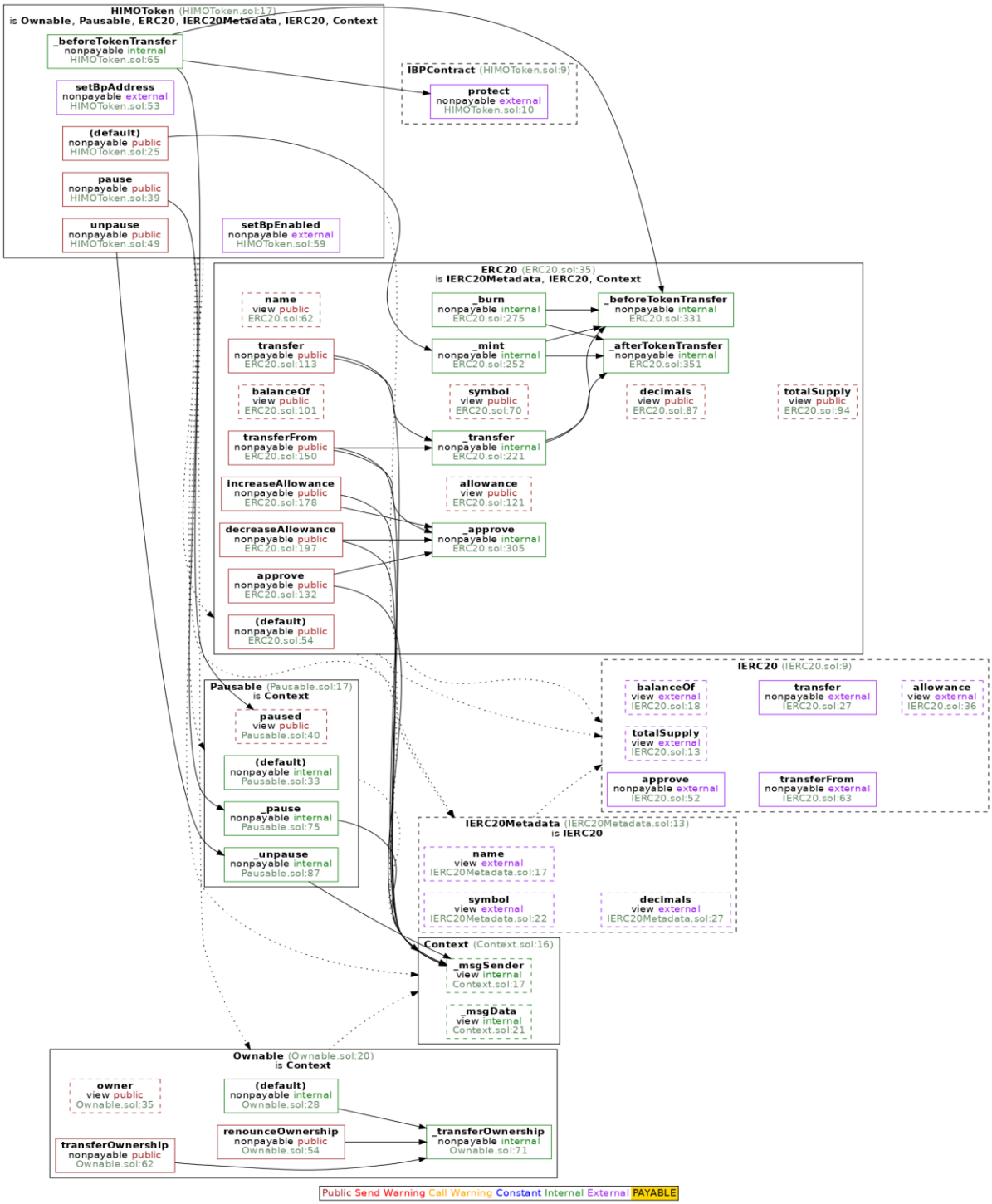

Image 1. HIMO Token call graph

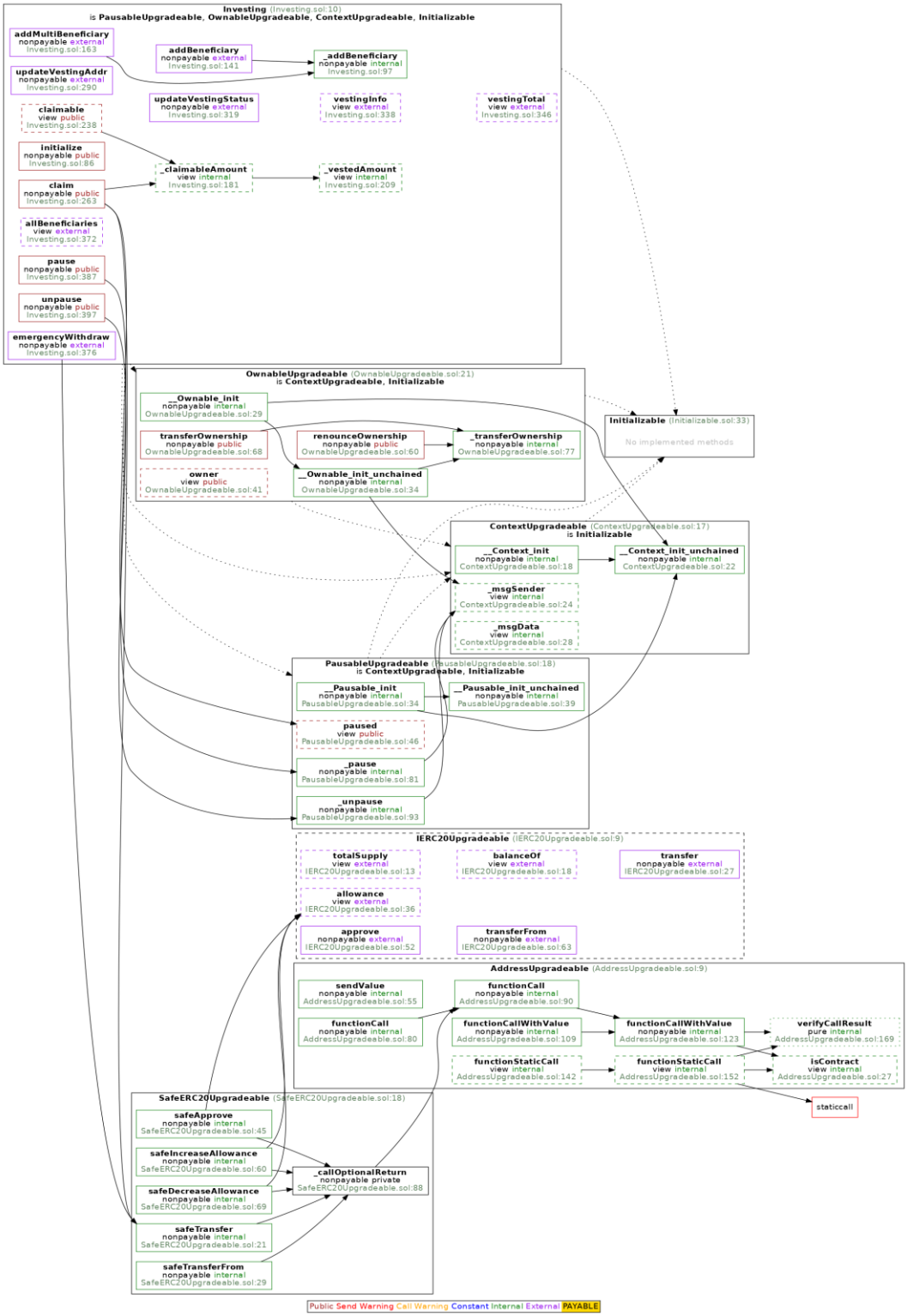

Image 2. Investing call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 24, 2021*	Private Report	Verichains Lab
1.1	*Dec 29, 2021*	Public Report	Verichains Lab

Table 2. Report versions history