Cyberscope

# Audit Report
# Hubinio

October 2022

| | |
|---|---|
| Type | BEP20 |
| Network | BSC |
| Address | 0xd2bCF93413E3996b8422DD3e320AE0Ed394e573F |
| Audited by | © cyberscope |

# Table of Contents

## Contract Review

| | |
|---|---|
| **Contract Name** | Hubinio |
| **Compiler Version** | v0.8.7+commit.e28d00a7 |
| **Optimization** | 200 runs |
| **Licence** | None |
| **Explorer** | https://bscscan.com/token/0xd2bCF93413E3996b8422DD3e320AE0Ed394e573F |

## Source Files

| **Filename** | **SHA256** |
|---|---|
| **contract.sol** | daa70b74b5690a2018f79782da441c1600fe3191e13ec64c121cfd97b049cf74 |

## Audit Updates

| | |
|---|---|
| **Initial Audit** | 25th October 2022 |
| **Corrected** | |

## Introduction

The Contract Hubinio implements a rewards claim mechanism. The contract predefines the winners off-chain and produces signatures. These signatures are used in the contract to validate the applicable winner.

## Roles

The contract has an owner role. The owner has the authority to:

- Configure times between claims and tokens per claim.
- Add a new banker.
- Change ownership.
- Recover the contract balance
- Recover the contract tokens.

The users have the ability to claim the rewards if they are eligible.

## Contract Diagnostics

🔴 Critical 🟠 Medium ● Minor / Informative

| Severity | Code | Description | Status |
|---|---|---|---|
| ● | CCT | Contract Claim Threshold | Unresolved |
| ● | OCTD | Transfers Contract's Tokens | Unresolved |
| ● | ULTW | Transfers Liquidity to Team Wallet | Unresolved |
| ● | STC | Succeeded Transfer Check | Unresolved |
| ● | ME | Missing Events | Unresolved |
| ● | MC | Missing Check | Unresolved |

## CCT - Contract Claim Threshold

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L47 |
| **Status** | Unresolved |

### Description

The contract could embed some off-chain checks inside the contract as an extra layer of safety. For instance, the claimReward method could also check for the timeBetweenClaims restriction. Hence, the contract will not relly celery from the off-chain logic.

```
function claimReward(bytes memory signature) external {
        bytes32 hashChallenge = hashPrefixed(keccak256(abi.encodePacked(msg.sender,
_prizeClaims[msg.sender].length)));
        address signer = recoverSigner(hashChallenge, signature);
        require(_bankers[signer], "Not signed by a banker");

        _tokenContract.transfer(msg.sender, tokensPerClaim);
        _prizeClaims[msg.sender].push(block.timestamp);
}
```

### Recommendation

The claimReward method could be reused the canClaim method. Hence, it will be guaranteed that the time restriction will be taken into consideration.

## OCTD - Transfers Contract's Tokens

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L97 |
| **Status** | Unresolved |

### Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the removeEth function.

```
function removeEth() external onlyOwner {
        uint256 balance = address(this).balance;
        payable(owner).transfer(balance);
}
```

### Recommendation

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

## ULTW - Transfers Liquidity to Team Wallet

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L102 |
| **Status** | Unresolved |

### Description

The contract owner has the authority to transfer funds without limit to the team wallet. These funds have been accumulated from fees collected from the contract. The owner may take advantage of it by calling the removeTokens.

```
function removeTokens(address token) external onlyOwner {
        uint256 balance = IERC20(token).balanceOf(address(this));
        IERC20(token).transfer(owner, balance);
}
```

### Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped. Since a huge amount may volatile the token’s price.

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

## STC - Succeeded Transfer Check

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L104 |
| **Status** | Unresolved |

### Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IERC20(token).transfer(owner, balance);
```

### Recommendation

The contract should check if the result of the transfer methods is successful.

## ME - Missing Events

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L47,81,87,92,97,102 |
| **Status** | Unresolved |

### Description

Detected missing events for critical access control functions. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
function claimReward(bytes memory signature) external

function setTimeAndTokens(uint256 time, uint256 tokens) external onlyOwner

function setBanker(address who, bool enabled) external onlyOwner {

function setOwner(address who) external onlyOwner

function removeEth() external onlyOwner

function removeTokens(address token) external onlyOwner
```

### Recommendation

Emit an event for critical parameter changes.

## MC - Missing Check

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L81,87 |
| **Status** | Unresolved |

### Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

The contract does not sanitize function arguments.

```
function setTimeAndTokens(uint256 time, uint256 tokens) external onlyOwner {
    timeBetweenClaims = time;
    tokensPerClaim = tokens;
}

function setBanker(address who, bool enabled) external onlyOwner {
    _bankers[who] = enabled;
}
```

The contract sends the awarded amount to the recipient without pre-validating that the amount is sufficient from the transaction.

```
function claimReward(bytes memory signature) external {
     //..
    _tokenContract.transfer(msg.sender, tokensPerClaim);
}
```

## Recommendation

The contract should properly check the variables according to the required specifications.

- tokensPerClaim should be greater than zero

- timeBetweenClaims should be greater than zero and lower than the current time stamp. Otherwise the `block.timestamp - timeBetweenClaims;` expression will revert.

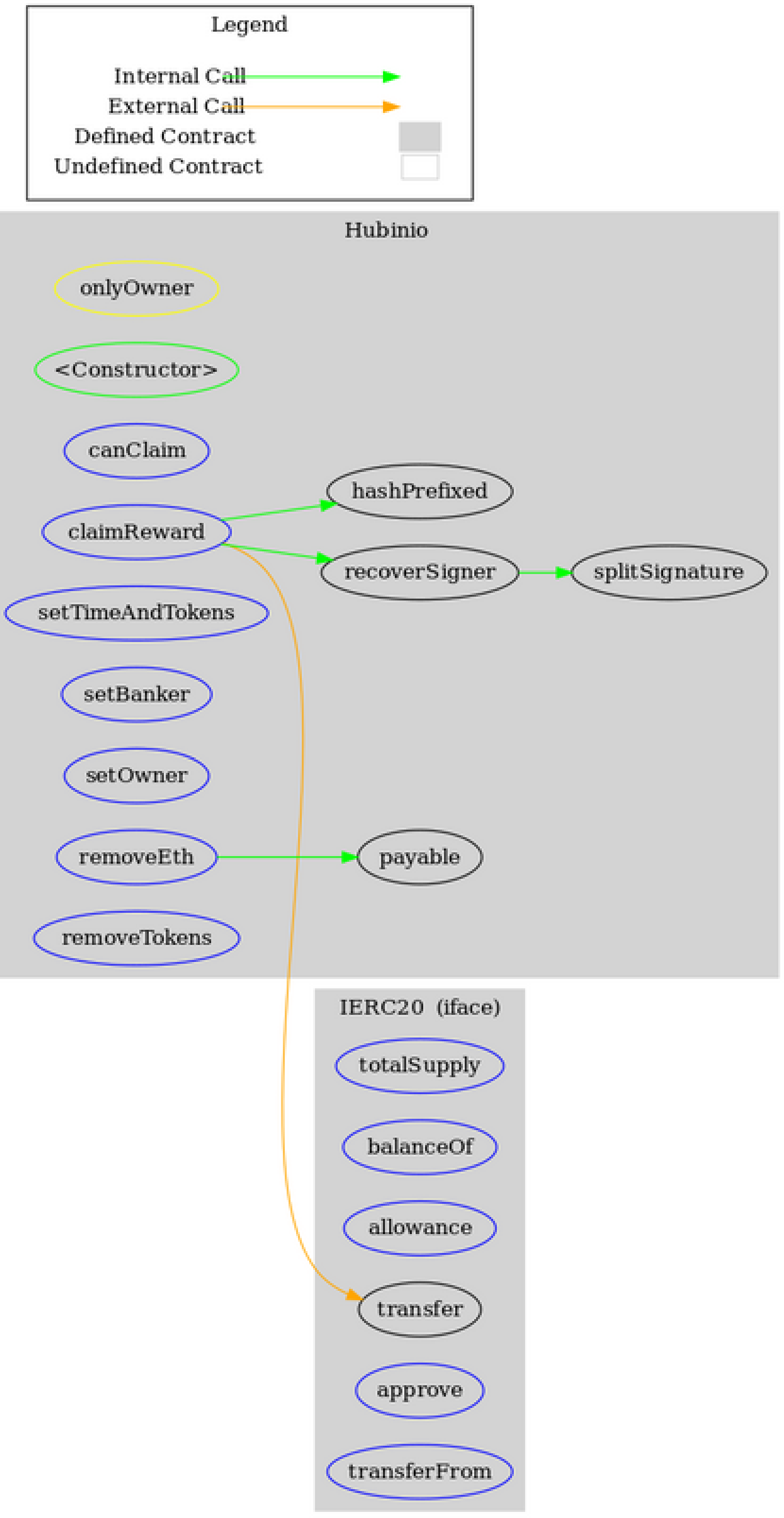
- who address should not be zero address.

It is recommended to pre-check if a user has sufficient balance to proceed with the reward and return a descriptive message.

# Contract Functions

| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **IERC20** | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | allowance | External | | - |
| | transfer | External | ✓ | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| **Hubinio** | Implementation | | | |
| | <Constructor> | Public | ✓ | - |
| | canClaim | External | | - |
| | claimReward | External | ✓ | - |
| | recoverSigner | Private | | |
| | hashPrefixed | Private | | |
| | splitSignature | Private | | |
| | setTimeAndTokens | External | ✓ | onlyOwner |
| | setBanker | External | ✓ | onlyOwner |
| | setOwner | External | ✓ | onlyOwner |
| | removeEth | External | ✓ | onlyOwner |
| | removeTokens | External | ✓ | onlyOwner |

## Contract Flow



## Summary

The Hubinio contract implements a rewards claim mechanism. This audit investigates security issues and mentions business logic concerns and potential improvements.

## Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

# About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io