auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

# Smart Contract Audit

FOR

# Coin31

DATED : 5 July 24'

# MANUAL TESTING

## Centralization – Enabling Trades

## Severity: High

## Function: EnableTrading

## Status: Open

**Overview:**
The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function enableTrading() external onlyOwner() {
    tradeEnable = !tradeEnable;
 }
```

**Suggestion:**
To reduce centralization and potential manipulation, consider one of the following approaches:
1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

# MANUAL TESTING

**Centralization – Owner can blacklist wallets.**
**Severity: High**
**Function:**
**Status: Open**

**Overview:**
The owner can blacklist wallets from transferring of tokens for an indefinite period of time which is not recommended. Which can lock user's token.

```
    ftrace | funcSig
    function addB(address addr↑, uint256 period↑) public onlyOwner {
        bl[addr↑] = block.timestamp + period↑;
    }

    ftrace | funcSig
    function removeB(address addr↑) public onlyOwner {
        bl[addr↑] = block.timestamp;
    }

}
```

# AUDIT SUMMARY

**Project name** – Coin31

**Date**: 5 July, 2024

**Scope of Audit-** Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

**Audit Status:** **High Risk Major Flag**

## Issues Found

| Status | Critical | High | Medium | Low | Suggestion |
|---|---|---|---|---|---|
| Open | 0 | 2 | 0 | 3 | 1 |
| Acknowledged | 0 | 0 | 0 | 0 | 0 |
| Resolved | 0 | 0 | 0 | 0 | 0 |

# USED TOOLS

## Tools:

**1- Manual Review:**
A line by line code review has been performed by audit ace team.

**2- BSC Test Network:** All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

**3- Slither :**
The code has undergone static analysis using Slither.

**Testnet version:**
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:

https://testnet.bscscan.com/address/0x7524bdc3604a3953a10436afaae68654bf9a93d5#code

# Token Information

**Token Address:** --

**Name:** --

**Symbol:** --

**Decimals:** --

**Network:** --

**Token Type:** --

**Owner:** --

**Deployer:** --

**Token Supply:** --

**Checksum:** Ae032c616934aeb47e6039f76b20d513

**Testnet:**
https://testnet.bscscan.com/address/0x7524bdc3604a3953a10436afaae68654bf9a93d5#code

# TOKEN OVERVIEW

**Buy Fee:** 0-0%

**Sell Fee:** 0-0%

**Transfer Fee:** 0-0%

**Fee Privilege:** Owner

**Ownership:** Owned

**Minting:** None

**Max Tx:** No

**Blacklist:** Yes

# AUDIT METHODOLOGY

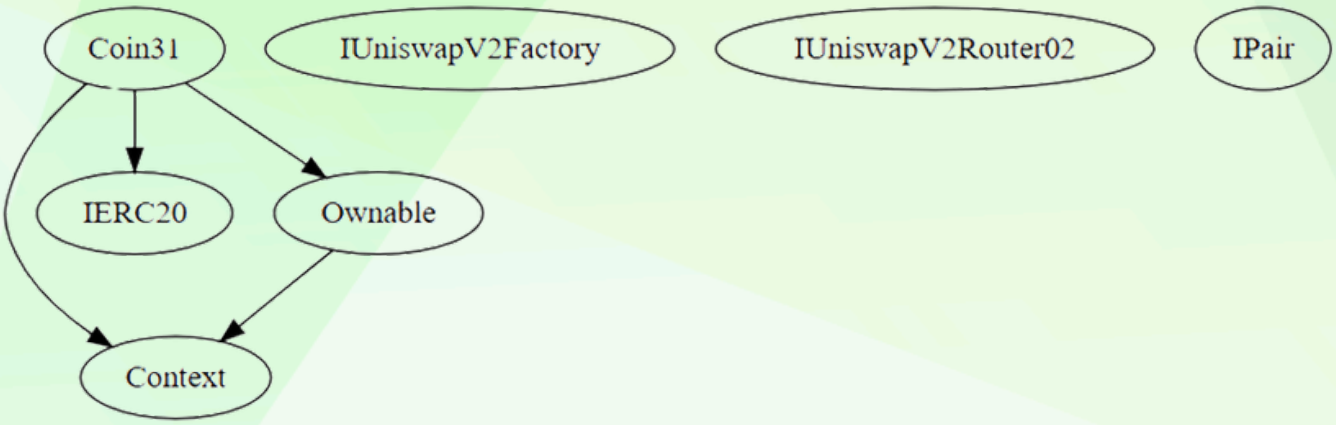
The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

# VULNERABILITY CHECKLIST

- [x] Return values of low-level calls
- [x] Private modifier
- [x] Multiple Sends
- [x] Using Suicide
- [x] Gas Limitand Loops
- [x] Address hardcoded
- [x] Exception Disorder
- [x] Using inline assembly
- [x] Divide before multiply
- [x] Missing Zero Address Validation
- [x] Compiler version not fixed
- [x] **Gasless Send**
- [x] Using block.timestamp
- [x] Re-entrancy
- [x] Tautology or contradiction
- [x] Timestamp Dependence
- [x] Revert/require functions
- [x] Use of tx.origin
- [x] Integer overflow/underflow
- [x] Dangerous strict equalities
- [x] Using SHA3
- [x] Using throw

# INHERITANCE TREE



# POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can Enable trading.
- The owner can recover BEP20.
- The owner can add/remove the W address.
- The owner can add/remove the B address.

# STATIC ANALYSIS

```
Coin31.recoverBEP20FromContract(address,uint256,address) (Coin31.sol#294-299) ignores return value by IERC20(_tokenAddy)
.transfer(to,_amount) (Coin31.sol#297)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Coin31.allowance(address,address).owner (Coin31.sol#248) shadows:
        - Ownable.owner() (Coin31.sol#117-119) (function)
Coin31._approve(address,address,uint256).owner (Coin31.sol#265) shadows:
        - Ownable.owner() (Coin31.sol#117-119) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Ownable.constructor().msgSender (Coin31.sol#112) lacks a zero-check on :
                - _owner = msgSender (Coin31.sol#113)
Coin31.recoverBNBfromContract(address).to (Coin31.sol#301) lacks a zero-check on :
                - address(address(to)).transfer(contractETHBalance) (Coin31.sol#305)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Coin31.recoverBEP20FromContract(address,uint256,address) (Coin31.sol#294-299):
        External calls:
        - IERC20(_tokenAddy).transfer(to,_amount) (Coin31.sol#297)
        Event emitted after the call(s):
        - ERC20TokensRecovered(_amount) (Coin31.sol#298)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Coin31.cbl(address) (Coin31.sol#285-287) uses timestamp for comparisons
        Dangerous comparisons:
        - bl[addr] > 0 && block.timestamp < bl[addr] (Coin31.sol#286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Reentrancy in Coin31.recoverBEP20FromContract(address,uint256,address) (Coin31.sol#294-299):
        External calls:
        - IERC20(_tokenAddy).transfer(to,_amount) (Coin31.sol#297)
        Event emitted after the call(s):
        - ERC20TokensRecovered(_amount) (Coin31.sol#298)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Coin31.cbl(address) (Coin31.sol#285-287) uses timestamp for comparisons
        Dangerous comparisons:
        - bl[addr] > 0 && block.timestamp < bl[addr] (Coin31.sol#286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Coin31.min(uint256,uint256) (Coin31.sol#239-241) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.17 (Coin31.sol#2) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Router02.WETH() (Coin31.sol#35) is not in mixedCase
Function IPair.DOMAIN_SEPARATOR() (Coin31.sol#69) is not in mixedCase
Function IPair.PERMIT_TYPEHASH() (Coin31.sol#70) is not in mixedCase
Function IPair.MINIMUM_LIQUIDITY() (Coin31.sol#87) is not in mixedCase
Parameter Coin31.recoverBEP20FromContract(address,uint256,address)._tokenAddy (Coin31.sol#294) is not in mixedCase
Parameter Coin31.recoverBEP20FromContract(address,uint256,address)._amount (Coin31.sol#294) is not in mixedCase
Constant Coin31._decimals (Coin31.sol#160) is not in UPPER_CASE_WITH_UNDERSCORES
```

```
Function IUniswapV2Router02.WETH() (Coin31.sol#35) is not in mixedCase
Function IPair.DOMAIN_SEPARATOR() (Coin31.sol#69) is not in mixedCase
Function IPair.PERMIT_TYPEHASH() (Coin31.sol#70) is not in mixedCase
Function IPair.MINIMUM_LIQUIDITY() (Coin31.sol#87) is not in mixedCase
Parameter Coin31.recoverBEP20FromContract(address,uint256,address)._tokenAddy (Coin31.sol#294) is not in mixedCase
Parameter Coin31.recoverBEP20FromContract(address,uint256,address)._amount (Coin31.sol#294) is not in mixedCase
Constant Coin31._decimals (Coin31.sol#160) is not in UPPER_CASE_WITH_UNDERSCORES
Modifier Coin31.ReentryGuard(address,address,uint256) (Coin31.sol#172-188) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in Coin31.recoverBNBfromContract(address) (Coin31.sol#301-307):
        External calls:
        - address(address(to)).transfer(contractETHBalance) (Coin31.sol#305)
        Event emitted after the call(s):
        - ETHBalanceRecovered() (Coin31.sol#306)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Coin31.router (Coin31.sol#157) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
Coin31.uniswapV2Router (Coin31.sol#167) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutabl
e
INFO:Slither:Coin31.sol analyzed (7 contracts with 93 detectors), 21 result(s) found
ethsec@0dcc802fad35:/share$
```

**Result => A static analysis of contract's source code has been performed using slither,**

**No major issues were found in the output**

# FUNCTIONAL TESTING

**1- Approve (passed):**

https://testnet.bscscan.com/tx/0x9c0807151d43d7d282edc303798b7ddc79bd2ac68ee0ab022a8d18612908e586

**2- Add W (passed):**

https://testnet.bscscan.com/tx/0x75d626917d66133f37d128c777c6cd53ac9232f5da1eb9960960c0451d4ff645

**3- Add B (passed):**

https://testnet.bscscan.com/tx/0xa1c56f67c4b0efdd44da147ec9b08c9a4d1bb2b03ccb09805d901936523e1562

**4- Enable Trading (passed):**

https://testnet.bscscan.com/tx/0xb07e66d7469d4b0c73e32257877fdf2dcd4ce252a1ff233c3b44384a0c56e2f2

**5- Remove B (passed):**

https://testnet.bscscan.com/tx/0xa79685d48a61334e1f6a98b74cd9de998110d445935300e399592fc377ed9210

**6- Remove W (passed):**

https://testnet.bscscan.com/tx/0xca19a435820ca4ea3fb00c778c324f82be1ccd39fda1387d667a84a53ca3b87e

# CLASSIFICATION OF RISK

| Severity | Description |
| --- | --- |
| Critical | These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away. |
| High-Risk | A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way. |
| Medium-Risk | A vulnerability that could affect the desired outcome of executing the contract in a specific scenario. |
| Low-Risk | A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective. |
| Gas Optimization /Suggestion | A vulnerability that has an informational character but is not affecting any of the code. |

# Findings

| Severity | Found |
| --- | --- |
| Critical | 0 |
| High-Risk | 2 |
| Medium-Risk | 0 |
| Low-Risk | 3 |
| Gas Optimization / Suggestions | 1 |

# MANUAL TESTING

## Centralization – Enabling Trades

**Severity: High**

**Function: EnableTrading**

**Status: Open**

**Overview:**

The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function enableTrading() external onlyOwner() {
    tradeEnable = !tradeEnable;
 }
```

**Suggestion:**

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

# MANUAL TESTING

**Centralization – Owner can blacklist wallets.**
**Severity: High**
**Function:**
**Status: Open**

**Overview:**
The owner can blacklist wallets from transferring of tokens for an indefinite period of time which is not recommended. Which can lock user's token.

```
    ftrace | funcSig
    function addB(address addr↑, uint256 period↑) public onlyOwner {
        bl[addr↑] = block.timestamp + period↑;
    }

    ftrace | funcSig
    function removeB(address addr↑) public onlyOwner {
        bl[addr↑] = block.timestamp;
    }

}
```

# MANUAL TESTING

## Centralization – Missing Events

**Severity: Low**

**Subject: Missing Events**

**Status: Open**

**Overview:**
They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function enableTrading() external onlyOwner() {
    tradeEnable = !tradeEnable;
  }
function addW(address addr) external onlyOwner {
    isW[addr] = true;
  }
function removeW(address addr) external onlyOwner {
    isW[addr] = false;
  }
function addB(address addr, uint256 period) public onlyOwner {
    bl[addr] = block.timestamp + period;
  }
function removeB(address addr) public onlyOwner {
    bl[addr] = block.timestamp;
  }
```

**Suggestion:**
Emit an event for critical changes.

# MANUAL TESTING

## Centralization – Missing Zero Address

**Severity: Low**

**Subject: Zero Check**

**Status: Open**

Overview:
functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
 function addW(address addr) external onlyOwner {
    isW[addr] = true;
  }
function removeW(address addr) external onlyOwner {
    isW[addr] = false;
  }
function addB(address addr, uint256 period) public onlyOwner {
    bl[addr] = block.timestamp + period;
  }
function removeB(address addr) public onlyOwner {
  bl[addr] = block.timestamp;
  }

function recoverBNBfromContract(address to) external onlyOwner {
    uint256 contractETHBalance = address(this).balance;
    require(contractETHBalance > 0, "Amount should be greater than zero");
    require(contractETHBalance <= address(this).balance, "Insufficient
Amount");
    payable(address(to)).transfer(contractETHBalance);
    emit ETHBalanceRecovered();
  }
```

**Suggestion:**
It is suggested that the address should not be zero or dead.

# MANUAL TESTING

## Centralization – Local variable Shadowing

**Severity: Low**

**Subject: Variable Shadowing**

**Status: Open**

**Overview:**

```
function allowance(address owner, address spender) public view returns
(uint256) {
    return _allowances[owner][spender];
  }
function _approve(address owner, address spender, uint256 amount) private {
    require(owner != address(0), "ERC20: approve from the zero address");
    require(spender != address(0), "ERC20: approve to the zero address");
    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
  }
```

**Suggestion:**

Rename the local variables that shadow another component.

# MANUAL TESTING

## Optimization

**Severity: Optimization**

**Subject: Remove unused code.**

**Status: Open**

**Overview:**

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice. though to avoid them.

```
function min(uint256 a, uint256 b) private pure returns (uint256) {
    return (a > b) ? b : a;
```

# 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. 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 Auditace 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 Auditace 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 Auditace team disclaims any liability for the resulting losses.

# ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.


**https://auditace.tech/**

**https://t.me/Audit_Ace**

**https://twitter.com/auditace_**

**https://github.com/Audit-Ace**