Audit Report for **Coin31**

Date: 02 July 2024

Audit result: **Failed**

**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

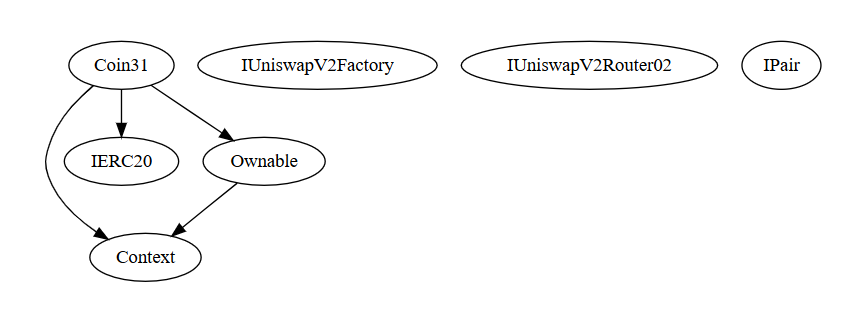
**Ownership:** Owned

**Minting:** None

**Max Tx:** No

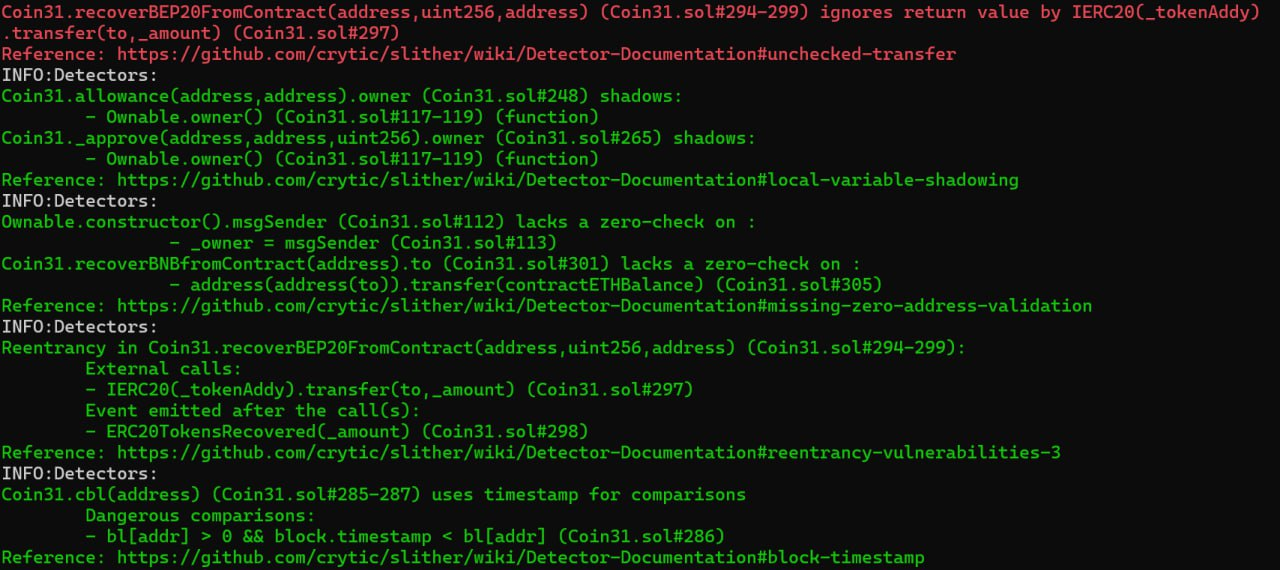
**Blacklist:** Yes

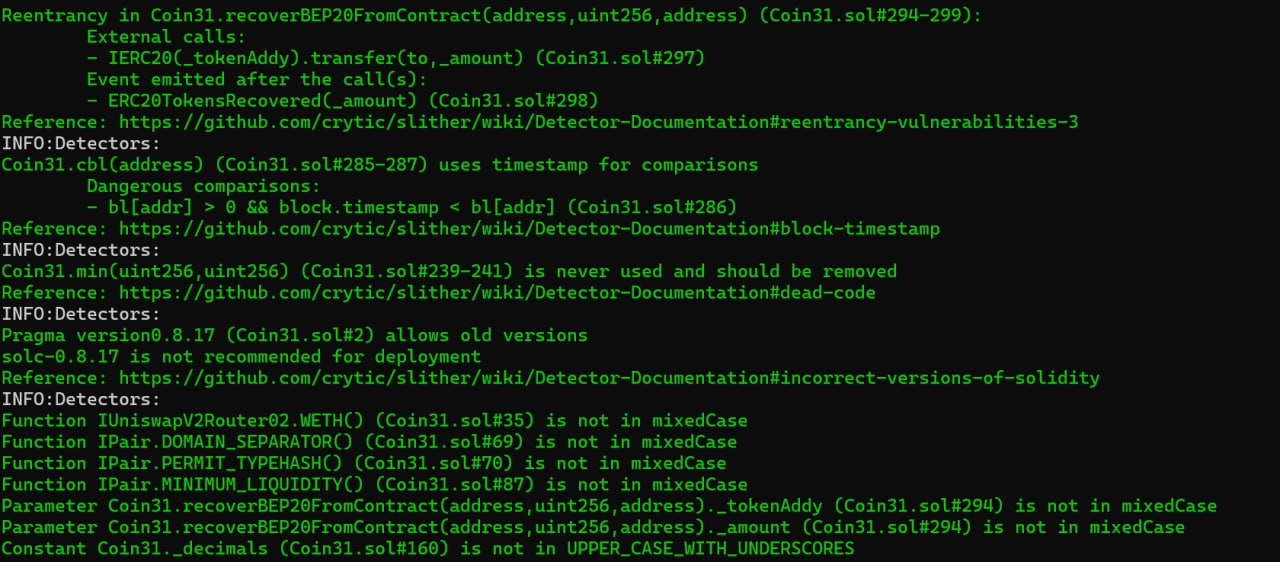
**Inheritance Tree**

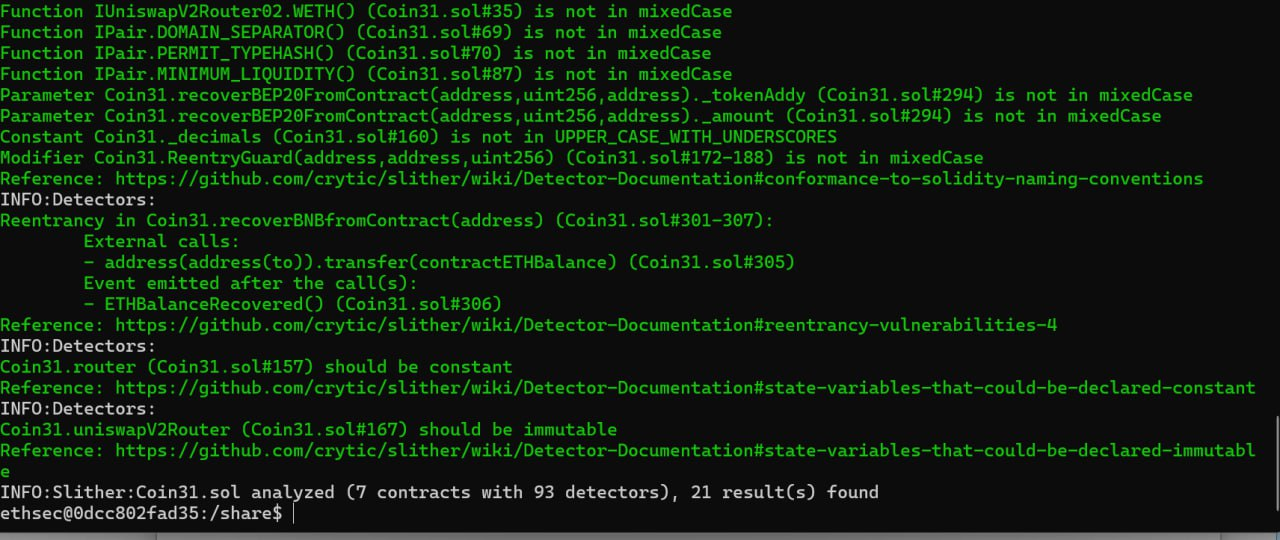


**Static Analysis**

A static analysis of the code was performed using Slither. No issues were found.







**Functional Tests**

**Router (PCS V2):**

**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>

**Ownership Privileges:**

- 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.

**Findings:**

**Critical**: 0

**High**: 2

**Medium**: 1

**Low**: 3

**Informational** **&** **Optimizations**: 1

**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.

**Centralization – Owner can blacklist wallets.**

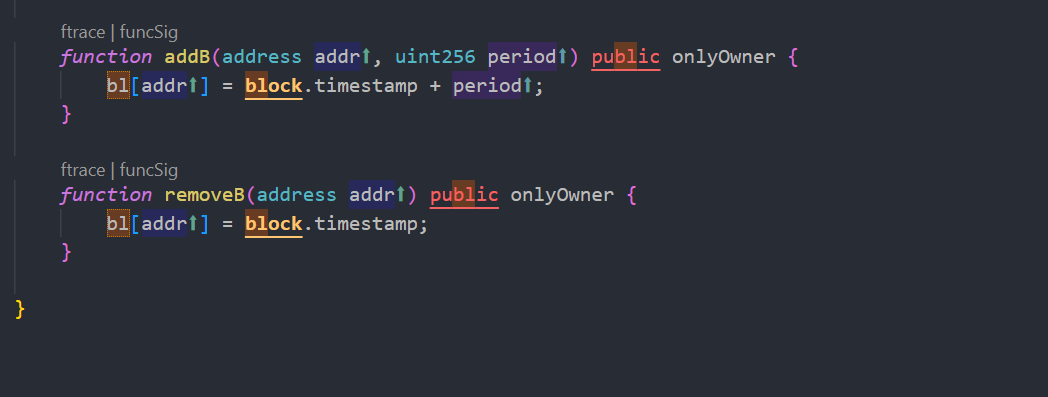
**Severity**: **High**

**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.

****

**Suggestion:** There should be a locking period so that the wallet cannot be locked for an indefinite. Period of time.

**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.

**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.

**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.

**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;