SMART CONTRACT SECURITY ASSESSMENT

PROJECT:

REVERSAL

DATE:

20 DECEMBER, 2022

https://t.me/SafuAudit

www.safuaudit.com

Introduction

Client	REVERSAL
Language	Solidity
Contract Address	0xb6cd32e28886ffe505c3dc8af959f41e530e992a
Owner	0xc995211a54e1859b55339ad99d14fc2264b3b278
Deployer	0xc995211a54e1859b55339ad99d14fc2264b3b278
SHA-256 Hash	0406e5e9260a8f0cc48c10d36ac79df57f816292
Decimals	18
Supply	100,000,000,000,000,000
Platform	Binance Smart Chain
Compiler	v0.8.17+commit.8df45f5f
Optimization	Yes with 20000 runs
Website	https://www.restoken.org/
Twitter	https://twitter.com/Restoken
Telegram	https://t.me/ResCoin

Overview

Fees

- Buy fees: 7.56%
- Sell fees: 7.56%

Fees privileges

- Owner can't modify fees

Ownership

- Owned

Minting

- No

Max Tx Amount

- Can't set max Tx

Pause

- Can't pause

Blacklist

- Can't blacklist

Other Privileges

- Owner can exclude users from paying fees

Table Of Contents

Risk Classification

Critical

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium

Issues on this level could potentially bring problems and should eventually be fixed.

Minor

Issues on this level are minor details and warning that can remain unfixed but would be better fixed at some point in the future

Informational

Information level is to offer suggestions for improvement of efficacity or security for features with a risk free factor.

Contract Inspection

Files Description Table

File Name	SHA-1 Hash
RES.sol	0406e5e9260a8f0cc48c10d36ac79df57f816292

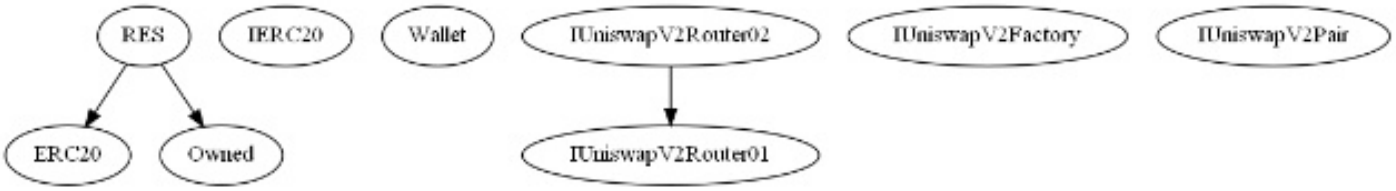
Contracts Description Table

ERC20	Implementation			
IERC20	Interface			
Wallet	Implementation			
Owned	Implementation			
IUniswapV2Router01	Interface			
IUniswapV2Router02	Interface	IUniswapV2Router01		
IUniswapV2Factory	Interface			
IUniswapV2Pair	Interface			
RES	Implementation	ERC20, Owned		
└	<Constructor>	Public ❗	🛑	ERC20 Owned
└	approveForSwap	Public ❗	🛑	NO❗
└	rebase	External ❗	🛑	NO❗
└	handleReserves	External ❗	🛑	NO❗
└	_beforeTransfer	Internal 🔒		
└	_allocationFees	Private 🔐	🛑	
└	_swapToken	Private 🔐	🛑	
└	setNoFee	External ❗	🛑	onlyOwner
└	setWhitelists	External ❗	🛑	onlyOwner
└	startRun	External ❗	🛑	onlyOwner
└	setConfig	External ❗	🛑	onlyOwner
└	setWorker	External ❗	🛑	onlyOwner
└	setMarketing	External ❗	🛑	onlyOwner
└	setMaster	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💲	Function is payable

Contract Inheritance



Inheritance is a feature of the object-oriented programming language. It is a way of extending the functionality of a program, used to separate the code, reduces the dependency, and increases the re-usability of the existing code. Solidity supports inheritance between smart contracts, where multiple contracts can be inherited into a single contract.

Findings

ID	Category	Issue	Severity
CE-OF	Centralization	Owner Accessible Functions	Optimization
CS-01	Coding Standards	Custom Contract for Ownership	Optimization
CS-02	Coding Standards	Using timestamp for comparisons	Optimization

CE-OF Owner Accessible Functions

Lines # -

```
setOwner()
setNoFee()
setWhitelists()
startRun()
setConfig()
setWorker()
setMarketing()
setMaster()
```

Description

The role OnlyOwner has authority over the above functions that can change the project functionality. Any compromise to the owner account may allow a hacker to take advantage of this authority.

Recommendation

We advise the client to carefully manage the privilege accounts' private key to avoid any potential risks of being hacked, or renounce ownership at some point in time.

CS-01 Custom Contract For Ownership

Lines # 293

```
abstract contract Owned {
...
}
```

Description

Contract uses a custom implementation for owner authorization

Recommendation

We recommend to use the OpenZeppelin Ownable contract. This will provide you with a series of tools that can be useful throughout the life of the contract. (https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol)

CS-02 Using Timestamp For Comparisons

Lines # 170, 667

```
function permit(address owner, address spender,
        uint256 value,
        uint256 deadline,
        uint8 v,
        bytes32 r,
        bytes32 s
    ) public virtual {
        require(deadline >= block.timestamp, "PERMIT_DEADLINE_EXPIRED");
...
}
 function handleReserves() external {
        Config memory cfg = config;
        require(
            msg.sender == owner ||
                uint256(cfg.handleInterval) + lastHandleTime < block.timestamp,
            "sleeping"
        );
```

Description

The above functions rely on block.timestamp for comparisons.

Recommendation

Avoid relying on block.timestamp. as it can be manipulated by miners.

Disclaimer

SafuAudit.com is not a financial institution and the information provided on this website does not constitute investment advice, financial advice, trading advice, or any other sort of advice. You should not treat any of the website's content as such. Investing in crypto assets carries a high level of risk and does not hold guarantees for not sustaining financial loss due to their volatility.

Accuracy of Information
SafuAudit will strive to ensure the accuracy of the information listed on this website although it will not hold any responsibility for any missing or wrong information. SafuAudit provides all information as is. You understand that you are using any and all information available here at your own risk. Any use or reliance on our content and services is solely at your own risk and discretion.

The purpose of the audit is to analyze the on-chain smart contract source code and to provide a basic overview of the project.

While we have used all the information available to us for this straightforward investigation, you should not rely on this report only — we recommend proceeding with several independent audits Be aware that smart contracts deployed on a blockchain aren't secured enough against external vulnerability or a hack. Be aware that active smart contract owner privileges constitute an elevated impact on the smart contract safety and security. Therefore, SafuAudit does not guarantee the explicit security of the audited smart contract. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

SAFUAUDIT
SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY

"Only in growth, reform, and change, paradoxically enough, is true security to be found."