soken

SMART CONTRACT
SECURITY AUDIT

CxCoin

July, 2021

Table of Contents

Disclaimer

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws. We took into consideration smart contract based algorithms, as well. Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it. Before making any judgments, you have to conduct your own independent research. We will discuss this in more depth in the following disclaimer - please read it fully.

DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Soken and its affiliates shall not be held responsible to you or anyone else, nor shall Soken provide any guarantee or representation to any person with regard to the accuracy or integrity of the report. Without any terms, warranties or other conditions other than as set forth in that exclusion and Soken excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills). The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Soken disclaims all responsibility and responsibilities and no claim against Soken is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent).

Security analysis is based only on the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

Procedure

Our analysis contains following steps:

1. Project Analysis;

2. Manual analysis of smart contracts:
- Deploying smart contracts on any of the network(Ropsten/Rinkeby) using Remix IDE
- Hashes of all transaction will be recorded
- Behaviour of functions and gas consumption is noted, as well.

3. Unit Testing:
- Smart contract functions will be unit tested on multiple parameters and under multiple conditions to ensure that all paths of functions are functioning as intended.
- In this phase intended behaviour of smart contract is verified.
- In this phase, we would also ensure that smart contract functions are not consuming unnecessary gas.
- Gas limits of functions will be verified in this stage.

4. Automated Testing:
- Mythril
- Oyente
- Manticore
- Solgraph

Terminology

We categorize the finding into 4 categories based on their vulnerability:

- Low-severity issue — less important, must be analyzed
- Medium-severity issue — important, needs to be analyzed and fixed
- High-severity issue —important, might cause vulnerabilities, must be analyzed and fixed
- Critical-severity issue —serious bug causes, must be analyzed and fixed.

Limitations

The security audit of Smart Contract cannot cover all vulnerabilities. Even if no vulnerabilities are detected in the audit, there is no guarantee that future smart contracts are safe. Smart contracts are in most cases safeguarded against specific sorts of attacks. In order to find as many flaws as possible, we carried out a comprehensive smart contract audit. Audit is a document that is not legally binding and guarantees nothing.

Token Contract Details for 09.07.2021

Contract Name: **CxCoin**

Deployer address: **0xFAF35FB2ce6223FB62c6Df34efcb9A5cFF186Ecb**

Total Supply: **1,000,000,000,000,000**

Token Ticker: **CX**

Decimals: **9**

Token holders: **3**

Transactions count: **10**

Top 100 holders dominance: **100%**

Liquidity fee: **9**

Tax fee: **2**

Total fees: **0**

Uniswap V2 pair: **0xca39681dc88a5b2385b2f30c2c87849a66f54c81**

Contract deployer address:
0x9e5275b5318e852067Bd70D6a1Fc1E9104DE0D8F

Audit Details

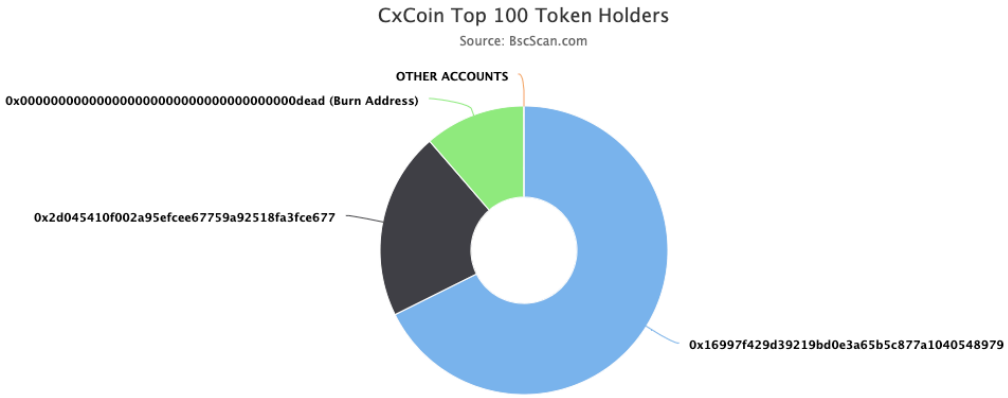
Project Name: **CxCoin**

Deployer address: **0x9e5275b5318e852067Bd70D6a1Fc1E9104DE0D8F**

Blockchain: **Binance Smart Chain**

Project Website: **https://cxcoin.net**

Cxcoin Token Distribution



CXcoin Top 10 Holders

Rank	Address	Quantity	Percentage
1	0x16997f429d39219bd0e3a65b5c877a1040548979	676,480,000,000,426	67.6480%
2	0x2d045410f002a95efcee67759a92518fa3fce677	210,000,000,000,000	21.0000%
3	Burn Address	113,519,999,999,466.270399999	11.3520%

Contract Function Details

+ Context
- [Int] _msgSender
- [Int] _msgData

+ [Int] IERC20
- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] transfer #
- [Ext] allowance
- [Ext] approve #
- [Ext] transferFrom #

+ [Lib] SafeMath
- [Int] add
- [Int] sub
- [Int] sub
- [Int] mul
- [Int] div
- [Int] div
- [Int] mod
- [Int] mod

+ [Lib] Address
- [Int] isContract
- [Int] sendValue #
- [Int] functionCall #
- [Int] functionCall #
- [Int] functionCallWithValue #
- [Int] functionCallWithValue #
- [Prv] _functionCallWithValue #

+ Ownable (Context)
- [Pub] <Constructor> #
- [Pub] owner
- [Pub] renounceOwnership #
    - modifiers: onlyOwner
- [Pub] transferOwnership #
    - modifiers: onlyOwner
- [Pub] getUnlockTime
- [Pub] getTime
- [Pub] lock #
    - modifiers: onlyOwner
- [Pub] unlock #

+ [Int] IUniswapV2Factory

- [Ext] feeTo
- [Ext] feeToSetter
- [Ext] getPair
- [Ext] allPairs
- [Ext] allPairsLength
- [Ext] createPair #
- [Ext] setFeeTo #
- [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair

- [Ext] name
- [Ext] symbol
- [Ext] decimals
- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] allowance
- [Ext] approve #
- [Ext] transfer #
- [Ext] transferFrom #
- [Ext] DOMAIN_SEPARATOR
- [Ext] PERMIT_TYPEHASH
- [Ext] nonces
- [Ext] permit #
- [Ext] MINIMUM_LIQUIDITY
- [Ext] factory
- [Ext] token0
- [Ext] token1
- [Ext] getReserves
- [Ext] price0CumulativeLast
- [Ext] price1CumulativeLast
- [Ext] kLast
- [Ext] burn #
- [Ext] swap #
- [Ext] skim #
- [Ext] sync #
- [Ext] initialize #

+ [Int] IUniswapV2Router01

- [Ext] factory
- [Ext] WETH

- [Ext] addLiquidity #
- [Ext] addLiquidityETH ($)
- [Ext] removeLiquidity #
- [Ext] removeLiquidityETH #
- [Ext] removeLiquidityWithPermit #
- [Ext] removeLiquidityETHWithPermit #
- [Ext] swapExactTokensForTokens #
- [Ext] swapTokensForExactTokens #
- [Ext] swapExactETHForTokens ($)
- [Ext] swapTokensForExactETH #
- [Ext] swapExactTokensForETH #
- [Ext] swapETHForExactTokens ($)
- [Ext] quote
- [Ext] getAmountOut
- [Ext] getAmountIn
- [Ext] getAmountsOut
- [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)

- [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
- [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
- [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
- [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
- [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ CxCoin (Context, IERC20, Ownable)
- [Pub] <Constructor> #
- [Pub] name
- [Pub] symbol

- [Pub] decimals
- [Pub] totalSupply
- [Pub] balanceOf
- [Pub] transfer #
- [Pub] allowance
- [Pub] approve #
- [Pub] transferFrom #
- [Pub] increaseAllowance #
- [Pub] decreaseAllowance #
- [Pub] isExcludedFromReward
- [Pub] totalFees
- [Pub] minimumTokensBeforeSwapAmount
- [Pub] buyBackUpperLimitAmount
- [Pub] deliver #

- [Pub] reflectionFromToken
- [Pub] tokenFromReflection
- [Pub] excludeFromReward #
    - modifiers: onlyOwner

- [Ext] includeInReward #
    - modifiers: onlyOwner
- [Prv] _approve #
- [Prv] _transfer #
- [Prv] swapTokens #
    - modifiers: lockTheSwap
- [Prv] buyBackTokens #
    - modifiers: lockTheSwap
- [Prv] swapTokensForEth #
- [Prv] swapETHForTokens #
- [Prv] addLiquidity #
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [Prv] _transferToExcluded #
- [Prv] _transferFromExcluded #
- [Prv] _transferBothExcluded #
- [Prv] _reflectFee #
- [Prv] _getValues
- [Prv] _getTValues
- [Prv] _getRValues
- [Prv] _getRate
- [Prv] _getCurrentSupply
- [Prv] _takeLiquidity #
- [Prv] calculateTaxFee
- [Prv] calculateLiquidityFee
- [Prv] removeAllFee #
- [Prv] restoreAllFee #
- [Pub] isExcludedFromFee
- [Pub] excludeFromFee #

    - modifiers: onlyOwner
- [Pub] includeInFee #
    - modifiers: onlyOwner

- [Ext] setTaxFeePercent #
    - modifiers: onlyOwner

- [Ext] setLiquidityFeePercent #
    - modifiers: onlyOwner

- [Ext] setMaxTxAmount #
    - modifiers: onlyOwner

- [Ext] setMarketingDivisor #
    - modifiers: onlyOwner

- [Ext] setNumTokensSellToAddToLiquidity #
    - modifiers: onlyOwner

- [Ext] setBuybackUpperLimit #
    - modifiers: onlyOwner

- [Ext] setMarketingAddress #
    - modifiers: onlyOwner

- [Pub] setSwapAndLiquifyEnabled #
    - modifiers: onlyOwner

- [Pub] setBuyBackEnabled #
    - modifiers: onlyOwner

- [Ext] prepareForPreSale #
    - modifiers: onlyOwner

- [Ext] afterPreSale #
    - modifiers: onlyOwner

- [Prv] transferToAddressETH #

($) = payable function
= non-constant function

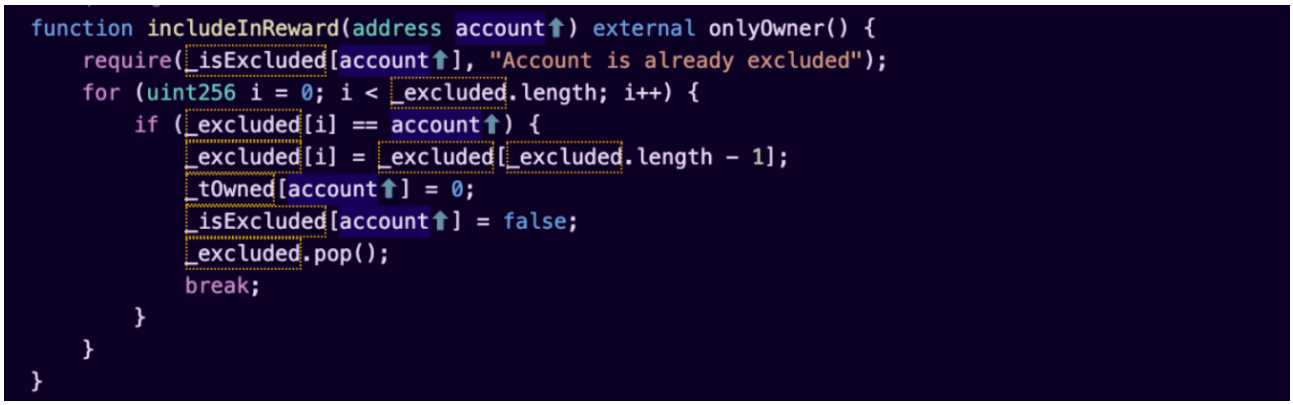
Vulnerabilities checking Status

Issue Description	Checking Status
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Compiler Errors	Completed
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Compiler Errors	Completed
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Compiler Errors	Completed
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed

Security Issues

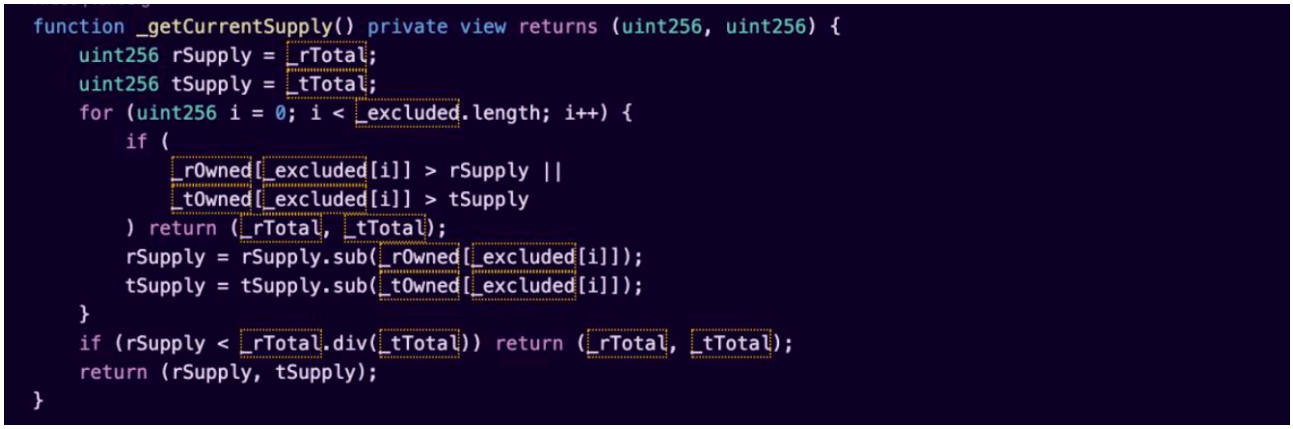
Out of Gas Issue:

The function includeInReward() uses the loop to find and remove addresses from the _excluded list. Function will be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.



```
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

The function _getCurrentSupply also uses the loop for evaluating total supply. It also could be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.



```
function _getCurrentSupply() private view returns (uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (
            _rOwned[_excluded[i]] > rSupply ||
            _tOwned[_excluded[i]] > tSupply
        ) return (_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

Recommendation:
Check that the excluded array length is not too big.

Owner privileges (In the period when the owner is not renounced)

Owner can change tax and liquidity fees.

```
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee↑) external onlyOwner() {
    _taxFee = taxFee↑;
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee↑) external onlyOwner() {
    _liquidityFee = liquidityFee↑;
}
```

Owner can change maximum transaction amount.

```
ftrace | funcSig
function setMaxTxAmount(uint256 maxTxAmount↑) external onlyOwner() {
    _maxTxAmount = maxTxAmount↑;
}
```

Owner can exclude from the fee.

```
    function excludeFromFee(address account↑) public onlyOwner {
        _isExcludedFromFee[account↑] = true;
    }
```

Owner can change marketingDivisor.

```
ftrace | funcSig
function setMarketingDivisor(uint256 divisor↑) external onlyOwner() {
    marketingDivisor = divisor↑;
}
```

Owner can change minimum number of tokens to add to liquidity.

```
ftrace | funcSig
function setNumTokensSellToAddToLiquidity(uint256 _minimumTokensBeforeSwap↑) external onlyOwner() {
    minimumTokensBeforeSwap = _minimumTokensBeforeSwap↑;
}
```

Owner can change buyBackUpperLimit.

```
ftrace | funcSig
function setBuybackUpperLimit(uint256 buyBackLimit↑) external onlyOwner() {
    buyBackUpperLimit = buyBackLimit↑ * 10**18;
}
```

Owner can change marketing address.

```
ftrace | funcSig
function setMarketingAddress(address _marketingAddress⬆) external onlyOwner() {
    marketingAddress = payable(_marketingAddress⬆);
}
```

Owner can enable and disable buyBack.

```
ftrace | funcSig
function setBuyBackEnabled(bool _enabled⬆) public onlyOwner {
    buyBackEnabled = _enabled⬆;
    emit BuyBackEnabledUpdated(_enabled⬆);
}
```

Owner can enable before and after presale modes.

```
ftrace | funcSig
function prepareForPreSale() external onlyOwner {
    setSwapAndLiquifyEnabled(false);
    _taxFee = 0;
    _liquidityFee = 0;
    _maxTxAmount = 1000000000 * 10**6 * 10**9;
}

ftrace | funcSig
function afterPreSale() external onlyOwner {
    setSwapAndLiquifyEnabled(true);
    _taxFee = 2;
    _liquidityFee = 9;
    _maxTxAmount = 3000000 * 10**6 * 10**9;
}
```

Owner can lock and unlock. By the way, using these functions the owner could retake privileges even after the ownership was renounced.

```
function lock(uint256 time⬆) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = block.timestamp + time⬆;
    emit OwnershipTransferred(_owner, address(0));
}

function unlock() public virtual {
    require(_previousOwner == msg.sender, "You don't have permission to unlock");
    require(block.timestamp > _lockTime , "Contract is locked until 7 days");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

Conclusion

Low-severity issues exist within smart contracts. Because it is out of scope, the security of a liquidity pair contract is not tested. The marketing address receives one-third of the liquidity. The funds raised will be used for other transfers and operations that are unrelated to this contract.

The team did not provide any information on liquidity locking.