soken

SMART CONTRACT
SECURITY AUDIT

CobraSwap

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 28.07.2021

Contract Name: **CobraSwap Token**

Deployer address: **0xd9Dd9C4f7B0DAE94C199b21D63E27310cD08dCEF**

Total Supply: **201,139,365,145**

Token Tracker: **COBRA**

Decimals: **18**

Token holders: **681**

Transactions count: **59600**

Top 100 holders dominance: **99.73%**

Contract deployer address:

0xd9Dd9C4f7B0DAE94C199b21D63E27310cD08dCEF

Audit Details

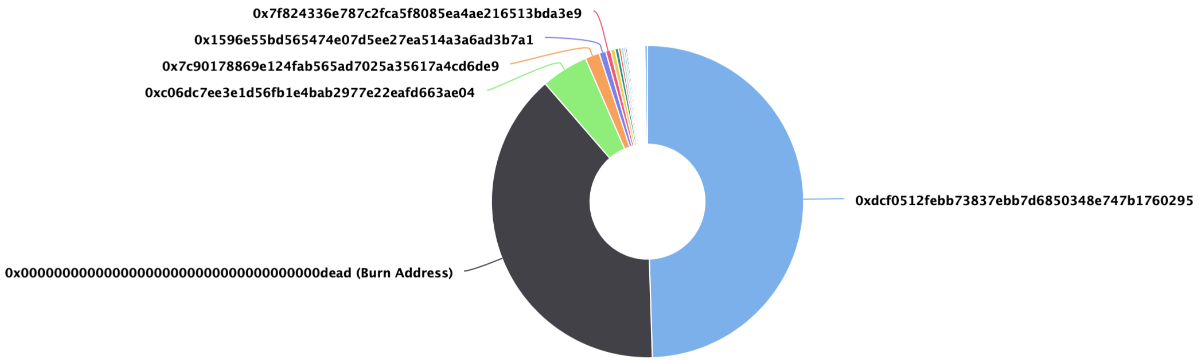
Project Name: **CobraSwap**

Language: **Solidity**

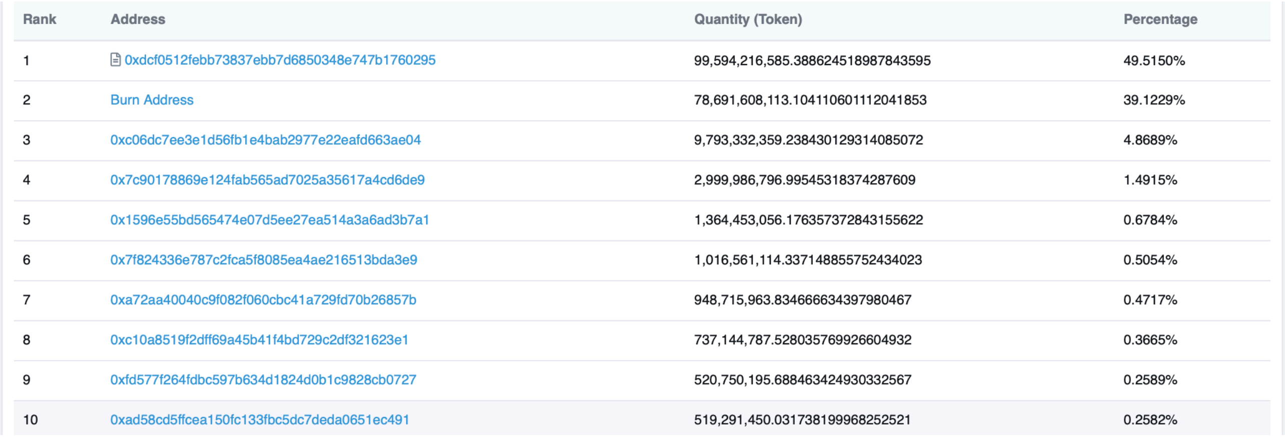
Blockchain: **Binance Smart Chain**

Project Website: **cobraswap.finance**

Cobra Token Distribution



Cobra Top 10 Holders



Rank	Address	Quantity (Token)	Percentage
1	📄 0xdcf0512febb73837ebb7d6850348e747b1760295	99,594,216,585.388624518987843595	49.5150%
2	Burn Address	78,691,608,113.104110601112041853	39.1229%
3	0xc06dc7ee3e1d56fb1e4bab2977e22eafd663ae04	9,793,332,359.238430129314085072	4.8689%
4	0x7c90178869e124fab565ad7025a35617a4cd6de9	2,999,986,796.99545318374287609	1.4915%
5	0x1596e55bd565474e07d5ee27ea514a3a6ad3b7a1	1,364,453,056.176357372843155622	0.6784%
6	0x7f824336e787c2fca5f8085ea4ae216513bda3e9	1,016,561,114.337148855752434023	0.5054%
7	0xa72aa40040c9f082f060cbc41a729fd70b26857b	948,715,963.834666634397980467	0.4717%
8	0xc10a8519f2dff69a45b41f4bd729c2df321623e1	737,144,787.528035769926604932	0.3665%
9	0xfd577f264fdbc597b634d1824d0b1c9828cb0727	520,750,195.688463424930332567	0.2589%
10	0xad58cd5ffcea150fc133fbc5dc7deda0651ec491	519,291,450.031738199968252521	0.2582%

Contract Function Details

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

+ [Int] IBEP20.sol
- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] decimals #
- [Ext] symbol
- [Ext] getOwner
- [Ext] balanceOf #
- [Ext] transfer #
- [Ext] allowance #
- [Ext] approve
- [Ext] transferFrom #

+ [Lib] SafeMath
- [Int] tryAdd
- [Int] trySub
- [Int] tryMul
- [Int] tryMod
- [Int] tryDiv
- [Int] add
- [Int] sub
- [Int] sub
- [Int] mul
- [Int] div
- [Int] div
- [Int] mod
- [Int] mod

+ [Lib] Address.sol
- [Int] isContract
- [Int] sendValue #
- [Int] functionCall #
- [Int] functionCall #
- [Int] functionCallWithValue #
- [Int] functionCallWithValue #
- [Int] functionStaticCall #
- [Int] functionStaticCall #
- [Int] functionDelegateCall #
- [Int] functionDelegateCall #
- [Int] verifyCallResult #

+ Ownable.sol (Context)
- [Pub] <Constructor> #
- [Pub] owner
- [Pub] onlyOwner
- [Pub] renounceOwnership #
    - modifiers: onlyOwner
- [Pub] transferOwnership #
    - modifiers: onlyOwner

+ [Int] IUniswapV2Factory.sol

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

+ [Int] IUniswapV2Pair.sol

- [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] mint

- [Ext] burn #
- [Ext] swap #
- [Ext] skim #
- [Ext] sync #
- [Ext] initialize #

+ [Int] IUniswapV2Router01.sol

- [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.sol (IUniswapV2Router01)

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

+ CobraToken.sol

- [Pub] mint
- [Pub] _transfer #
- [Pub] swapAndLiquify
- [Pub] swapTokensForEth
- [Pub] addLiquidity
- [Pub] maxTransferAmount
- [Pub] updateTransferTaxRate
- [Pub] isExcludedFromAntiWhale

- [Pub] updateBurnRate
- [Pub] updateMaxTransferAmountRate
- [Pub] updateMinAmountToLiquify
- [Pub] setExcludedFromAntiWhale
- [Pub] updateSwapAndLiquifyEnabled
- [Pub] updateCobraSwapRouter
- [Pub] operator
- [Pub] transferOperator

- [Ext] delegates
- [Ext] transferOperator
- [Ext] delegateBySig
- [Ext] getCurrentVotes
- [Ext] getPriorVotes

- [Int] _delegate
- [Int] _moveDelegates
- [Int] _writeCheckpoint
- [Int] safe32
- [Int] getChainId

+ BEP20
- [Ext] getOwner
- [Pub] name
- [Pub] symbol
- [Pub] decimals
- [Pub] totalSupply
- [Pub] balanceOf
- [Pub] allowance
- [Pub] approve #
- [Pub] transfer #
- [Pub] transferFrom #
- [Pub] increaseAllowance
- [Pub] decreaseAllowance
- [Pub] mint

- [Int] _transfer
- [Int] _mint
- [Int] _burn
- [Int] _approve
- [Int] _burnFrom

($) = payable function
# = non-constant function

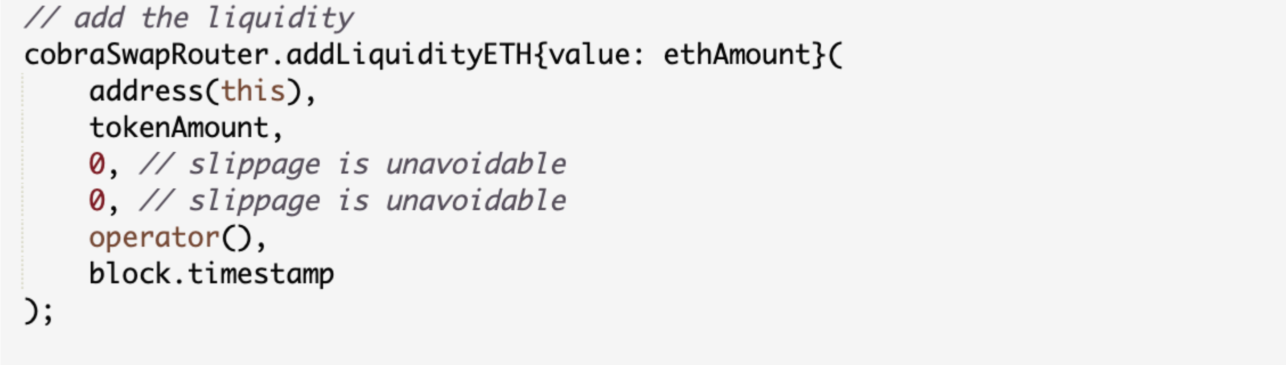
Vulnerabilities checking Status

Issue Description	Checking Status
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Timestamp Dependence	Completed
Shadowing State Variables	Completed
DoS with Failed Call	Completed
DoS with Block Gas Limit	Low issues
Outdated Complier Version	Completed
Assert Violation	Completed
Use of Deprecated Solidity Functions	Completed
Integer Overflow and Underflow	Completed
Function Default Visibility	Completed
Malicious Event Log	Completed
Math Accuracy	Completed
Design Logic	Completed
Fallback Function Security	Completed
Cross-function Race Conditions	Completed
Safe Zeppelin Module	Completed

Security Issues

1) addLiquidityETH function issue:

The return values of addLiquidityETH are not properly handled.



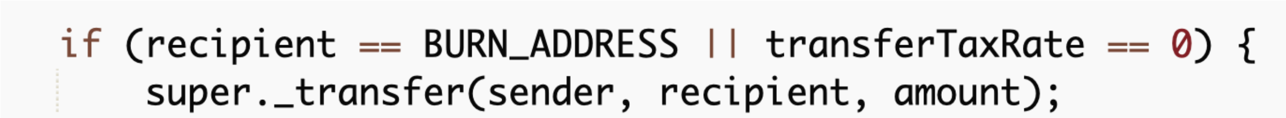
```
// add the liquidity
cobraSwapRouter.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0, // slippage is unavoidable
    0, // slippage is unavoidable
    operator(),
    block.timestamp
);
```

Recommendation:

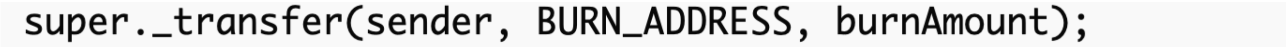
We recommend using variables to receive the return values of the functions mentioned above and to handle both success and failure cases if needed by the business logic.

2) Inappropriate Burn Method:

In the following code snippets, the token burn is accomplished by sending burnAmount tokens to the BURN_ADDRESS



```
if (recipient == BURN_ADDRESS || transferTaxRate == 0) {
    super._transfer(sender, recipient, amount);
```



```
super._transfer(sender, BURN_ADDRESS, burnAmount);
```

Although token burn can be achieved by transferring tokens directly to the "dead" address, making these transferred tokens not available from the users, the number of transferred tokens is not deducted from the total token supply, not reflecting the actual token supply that is available to all users.

Recommendation:

We recommend using a dedicated token burn function for handing the burning, instead of sending the tokens to the "zero" address.

3) Contract Gains Non-withdrawable BNB via the swapAndLiquify:

The swapAndLiquify function converts half of the minAmountToLiquify Cobra tokens to BNB. The other half of Cobra tokens and part of the converted BNB are deposited into the Cobra-BNB pool on Uniswap as liquidity.

```
function swapAndLiquify() private lockTheSwap transferTaxFree
```

For every swapAndLiquify function call, a small amount of BNB leftover in the contract. This is because the price of Cobra drops after swapping the first half of Cobra tokens into BNBs, and the other half of Cobra tokens require less than the converted BNB to be paired with it when adding liquidity. The contract doesn't appear to provide a way to withdraw those BNB, and they will be locked in the contract forever.

Recommendation:

It's not ideal that more and more BNB are locked into the contract over time. The simplest solution is to add a withdraw function in the contract to withdraw BNB. Other approaches that benefit the Cobra token holders can be:

- Distribute BNB to Cobra token holders proportional to the amount of token they hold.
- Use leftover BNB to buy back Cobra tokens from the market to increase the price of Cobra token.

Conclusion

Low-severity issues exist within smart contracts. Smart contracts are free from any critical or high-severity issues.

NOTE: Please check the disclaimer above and note, that audit makes no statements or warranties on business model, investment attractiveness or code sustainability.

Audited by soken