AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Baby Doge CEO

DATED : 06 Mar 23'

AUDIT SUMMARY

Project name – Phoenix Chain

Date: 06 March, 2023

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

Audit Status: Passed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	1	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 done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0x3B451a1A1F16B50d01ba98C72b4b50F25a98b0E5

Token Information

Token Name : Baby Doge CEO

Token Symbol: BCEO

Decimals: 9

Token Supply: 420,000,000,000,000,000

Token Address:
0x8D875ABca035858C901Fb3B61a98179aA2cA7ed9

Checksum:
48ce0e526f89e88b78c4a39577f922507480fcd6

Owner:
0xAE6A8763191534AD4cf5FbaAEF53b1788148B501

TOKEN OVERVIEW

Fees:

Buy Fees: 10%

Sell Fees: 10%

Transfer Fees: 10%

Fees Privilige: None

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: including and excluding from fees and rewards

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

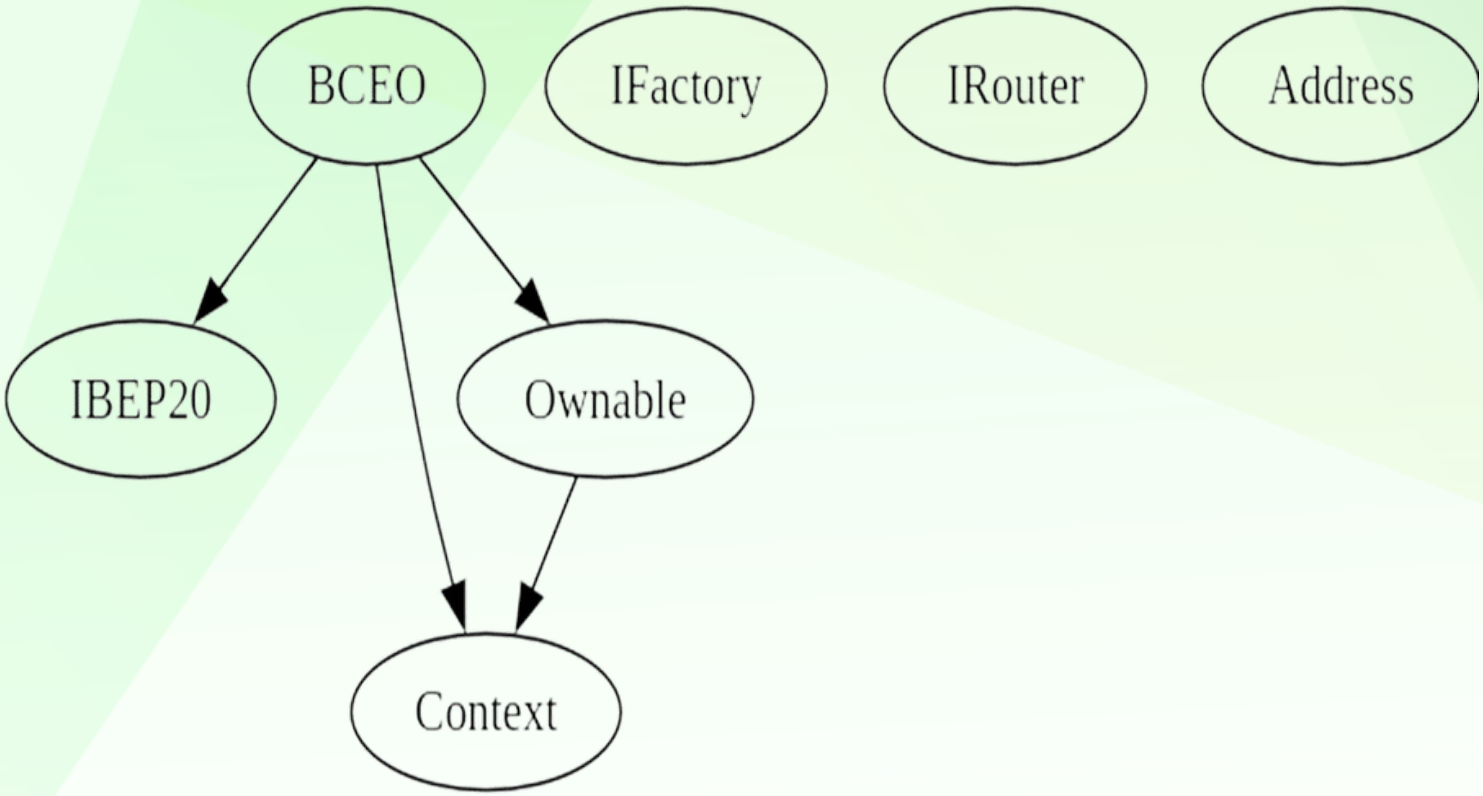
CLASSIFICATION OF RISK

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	0
Medium-Risk	0
Low-Risk	1
Gas Optimization / Suggestions	1

INHERITANCE TREE



POINTS TO NOTE

- ·Owner is not able to change fees (10% buy, 10% sell and 10% transfer)
- ·Owner is not able to set max buy/sell/transfer amount
- ·Owner is not able to blacklist an arbitrary wallet
- ·Owner is not able to disable trades
- ·Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
IFactory	Interface			
└	createPair	External ❗	🛑	NO❗
IRouter	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
Address	Library			
└	sendValue	Internal 🔒	🛑	
BCEO	Implementation	Context, IBEP20, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗

CONTRACT ASSESMENT

└	allowance	Public ❗		|NO❗
└	approve	Public ❗	🛑	|NO❗
└	transferFrom	Public ❗	🛑	|NO❗
└	increaseAllowance	Public ❗	🛑	|NO❗
└	decreaseAllowance	Public ❗	🛑	|NO❗
└	transfer	Public ❗	🛑	|NO❗
└	isExcludedFromReward	Public ❗		|NO❗
└	reflectionFromToken	Public ❗		|NO❗
└	EnableTrading	External ❗	🛑	onlyOwner
└	updatedeadline	External ❗	🛑	onlyOwner
└	tokenFromReflection	Public ❗		|NO❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	isExcludedFromFee	Public ❗		|NO❗
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	addLiquidity	Private 🔐	🛑	
└	swapTokensForBNB	Private 🔐	🛑	
└	bulkExcludeFee	External ❗	🛑	onlyOwner
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	updateDevWallet	External ❗	🛑	onlyOwner
└	updateOpsWallet	External ❗	🛑	onlyOwner
└	updateSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	updateSwapEnabled	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueAnyBEP20Tokens	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	|NO❗

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BCEO.includeInReward(address) (contracts/Token.sol#418-429) has costly operations inside a loop:
        - _excluded.pop() (contracts/Token.sol#425)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (contracts/Token.sol#60-63) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

BCEO._rTotal (contracts/Token.sol#179) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.17 (contracts/Token.sol#21) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.18 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#140-151):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#146)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter.WETH() (contracts/Token.sol#116) is not in mixedCase
Struct BCEO.valuesFromGetValues (contracts/Token.sol#216-230) is not in CapWords
Function BCEO.EnableTrading() (contracts/Token.sol#384-389) is not in mixedCase
Parameter BCEO.updatedeadline(uint256)._deadline (contracts/Token.sol#391) is not in mixedCase
Parameter BCEO.updateSwapEnabled(bool)._enabled (contracts/Token.sol#807) is not in mixedCase
Parameter BCEO.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (contracts/Token.sol#819) is not in mixedCase
Parameter BCEO.rescueAnyBEP20Tokens(address,address,uint256)._to (contracts/Token.sol#820) is not in mixedCase
Parameter BCEO.rescueAnyBEP20Tokens(address,address,uint256)._amount (contracts/Token.sol#821) is not in mixedCase
Constant BCEO._decimals (contracts/Token.sol#175) is not in UPPER_CASE_WITH_UNDERSCORES
Variable BCEO.genesis_block (contracts/Token.sol#183) is not in mixedCase
Constant BCEO._name (contracts/Token.sol#191) is not in UPPER_CASE_WITH_UNDERSCORES
Constant BCEO._symbol (contracts/Token.sol#192) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#61)" inContext (contracts/Token.sol#55-64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

BCEO._lastSell (contracts/Token.sol#170) is never used in BCEO (contracts/Token.sol#154-831)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

BCEO._tTotal (contracts/Token.sol#178) should be constant
BCEO.deadWallet (contracts/Token.sol#186) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

BCEO.pair (contracts/Token.sol#173) should be immutable
BCEO.router (contracts/Token.sol#172) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

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

No major issues were found in the output

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

All the functionalities have been tested, no issues were found

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0xc85fc5d14df91a1417cf04f9ff097c1a6f789a336df3dde3f7abdd738d8981e2

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x510cdba7d190fa7d1027671bd3ba478079927a5bedf6d18fd48a799b7a6a34cf

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xd322d008766212c3646a77241ebc887d0aec614f5abbf64af657f47506e6bb89

4- Transferring when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x160fe4800b81f1499fe08c401fb7bd31702062c5e46209975867d7916a6a5286

5- Buying when not excluded (10% tax) (passed):
https://testnet.bscscan.com/tx/0x6fab7ad77f68db39cf096cda0bce312dfd2b6db75ef3419a5bfc56932ee2b106

6- Selling when not excluded (10% tax) (passed):
https://testnet.bscscan.com/tx/0xd805357ca517ee4c817ab6cb8b9d5b78b72a256bfc7ea3e6e5bf2a49b44cefdf

FUNCTIONAL TESTING

7- Transferring when not excluded (10% tax) (passed):
https://testnet.bscscan.com/tx/0xc78335e7daf1ccedf6ec0fe1998843166964099810ca45c862c04df0153e3e2

8- Internal swap (passed):
prize pool wallet received ETH
https://testnet.bscscan.com/address/0xaaa245d0b7acc5129d4284d61e127b612e5219c8#internaltx

dev wallet received ETH during above sells (non-excluded)
https://testnet.bscscan.com/address/0x640e8938b5aecca43c0f67c0b566900590188faf#internaltx

ops wallet received ETH during above sells (non-excluded)
https://testnet.bscscan.com/address/0xc520247f4f0c9b3e6819ac6d9822f83535179f75#internaltx

9- Reflections (passed):
Monitored balances of multiple wallets, each one's balance increased after each trade, this means reflections are working. Also excluding and including to reflections have been tested and passed. (can be found at testnet link)

MANUAL TESTING

Low Risk Issues

Issue: no way to enable auto-liquidity
Type : Logical
Function:---
Line: ---
Severity: Low

Overview:
Although the contract has implemented the necessary functionality for auto-liquidity, the feature is deemed redundant since the liquidity tax is set at 0 and is non-upgradable. As a result, the functionality serves no practical purpose.

```
function addLiquidity(uint256 tokenAmount, uint256 bnbAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(router), tokenAmount);

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

Recommendation:

- **delete this auto-liquidity functions from the contract to reduce overall gas usage**
- **add the necessary function to be able to update liquidity tax**

MANUAL TESTING

Suggestions & Recommendations:

S-1 : allowing owner to withdraw native tokens from the contract

Preventing the owner from withdrawing native tokens from the contract can provide a level of transparency and assurance to investors that their fees are being used in a way that benefits all investors. This approach can also help to prevent potential misuse or mismanagement of the collected fees.

On the other hand, allowing the owner to withdraw the native tokens can provide more flexibility and control over how the collected fees are used. This can be particularly useful in situations where the collected fees need to be used for purposes other than marketing or providing liquidity.

Hence, It may be beneficial to consider implementing additional safeguards or controls to prevent potential misuse of the collected fees, regardless of whether the owner is allowed to withdraw them or not.

```
//Use this in case BEP20 Tokens are sent to the contract by mistake
ftrace | funcSig
function rescueAnyBEP20Tokens(
    address _tokenAddr,
    address _to,
    uint256 _amount
) public onlyOwner {
    require(
        _tokenAddr != address(this),
        "Owner can't claim contract's balance of its own tokens"
    );
    IBEP20(_tokenAddr).transfer(_to, _amount);
}
```

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