MOONAUDIT

Smart Contract Audit Report

BabyDogeCoin (BABYDOGE)

BEP20 on Binance Smart Chain

Jul 4th, 2022

https://MoonAudit.org

Table of Contents

Overview

Project Summary	
Project Name	BabyDogeCoin (BABYDOGE)
Platform	Binance Smart Chain
Language	Solidity
Contract Type	BEP20
Contract Address	0xc748673057861a797275CD8A068AbB95A902e8de
Contract Owner	0xf103d2AbA493749a402B7dE11cF31f5844062B74
Block Explorer	https://bscscan.com/

Audit Summary	
Delivery Date	Jul 4th, 2022 GMT+0
Block Number	19250729
Static Analysis	Yes
Graphic Analysis	Yes
Logic Disassemble	Yes
Mannual Review	Yes

Vulnerability Summary

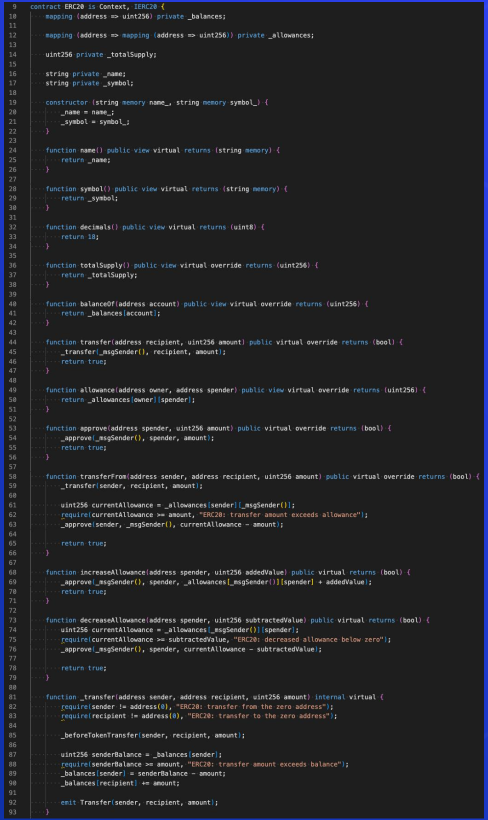
Severity Level	Total	Acknowledged	Alleviated	Resolved
Critical	0	0	0	0
Major	0	0	0	0
Medium	0	0	0	0
Minor	2	2	0	0
Informational	4	4	0	0
Discussion	1	1	0	0

Fully Sanity Checks

	Read	Write	AI Scanned	Human Reviewed	Result	Suggested	Resolved
name()	Yes		Completed	Completed	No Risk		
symbol()	Yes		Completed	Completed	No Risk		
balanceOf()	Yes		Completed	Completed	No Risk		
decimals()	Yes		Completed	Completed	No Risk		
totalSupply()	Yes		Completed	Completed	No Risk		
allowance()	Yes		Completed	Completed	No Risk		
approve()		Yes	Completed	Completed	No Risk		
decreaseAllowance()		Yes	Completed	Completed	✓ Low/No Risk		
increaseAllowance()		Yes	Completed	Completed	✓ Low/No Risk		
renounceOwnership()		Yes	Completed	Completed	✓ Low/No Risk		
transfer()		Yes	Completed	Completed	✓ Low/No Risk		
transferFrom()		Yes	Completed	Completed	✓ Low/No Risk		
transferOwnership()		Yes	Completed	Completed	✓ Low/No Risk		

BEP20 token to the key functions are shown above.

Source Code Analysis



```solidity
contract ERC20 is Context, IERC20 {
    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)) private _allowances;

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;

    constructor (string memory name_, string memory symbol_) {
        _name = name_;
        _symbol = symbol_;
    }

    function name() public view virtual returns (string memory) {
        return _name;
    }

    function symbol() public view virtual returns (string memory) {
        return _symbol;
    }

    function decimals() public view virtual returns (uint8) {
        return 18;
    }

    function totalSupply() public view virtual override returns (uint256) {
        return _totalSupply;
    }

    function balanceOf(address account) public view virtual override returns (uint256) {
        return _balances[account];
    }

    function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
        _transfer(_msgSender(), recipient, amount);
        return true;
    }

    function allowance(address owner, address spender) public view virtual override returns (uint256) {
        return _allowances[owner][spender];
    }

    function approve(address spender, uint256 amount) public virtual override returns (bool) {
        _approve(_msgSender(), spender, amount);
        return true;
    }

    function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
        _transfer(sender, recipient, amount);

        uint256 currentAllowance = _allowances[sender][_msgSender()];
        require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
        _approve(sender, _msgSender(), currentAllowance - amount);

        return true;
    }

    function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
        return true;
    }

    function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
        uint256 currentAllowance = _allowances[_msgSender()][spender];
        require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
        _approve(_msgSender(), spender, currentAllowance - subtractedValue);

        return true;
    }

    function _transfer(address sender, address recipient, uint256 amount) internal virtual {
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");

        _beforeTokenTransfer(sender, recipient, amount);

        uint256 senderBalance = _balances[sender];
        require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
        _balances[sender] = senderBalance - amount;
        _balances[recipient] += amount;

        emit Transfer(sender, recipient, amount);
    }
```

We've found 4 contracts written by BabyDogeCoin team and 4 UniswapV2 interfaces contracts in the project source code and the partial screenshot of the contract code as left side shown.

- BabyDogeCoin (CoinToken)
- IERC20
- Ownable
- Context
- IUniswapV2Factory
- IUniswapV2Router01
- IUniswapV2Router02
- IUniswapV2Pair

respectively.

MoonAudit.org

Minor Issues

1.The variable `owner` shadowing
- CoinToken.allowance(address,address).owner (BabyDogeCoin.sol#787)
- CoinToken._approve(address,address,uint256).owner (BabyDogeCoin.sol#998)

2.Missing an event while changing state variable
- CoinToken.setTaxFeePercent(uint256) (BabyDogeCoin.sol#887-889)
- CoinToken.setLiquidityFeePercent(uint256) (BabyDogeCoin.sol#891-893)
- CoinToken.setNumTokensSellToAddToLiquidity(uint256) (BabyDogeCoin.sol#895-897)
- CoinToken.setMaxTxPercent(uint256) (BabyDogeCoin.sol#899-901)

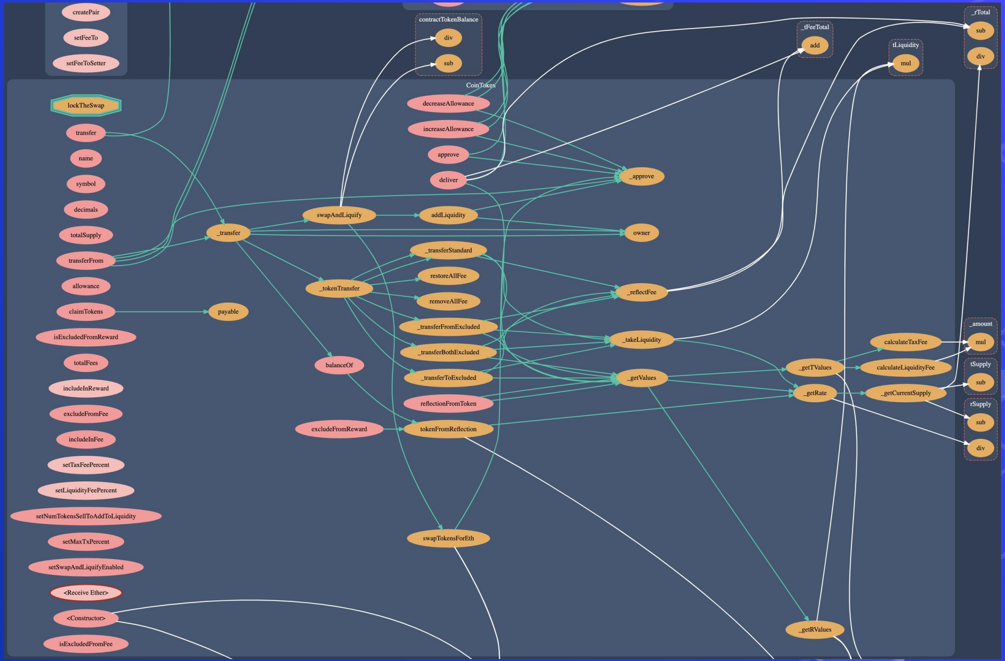
Informational Issues

1.Costly operations in a loop
- CoinToken.includeInReward(address) (BabyDogeCoin.sol#856-867)

2.Never used functions
- Address._functionCallWithValue(address,bytes,uint256,string)
- Address.functionCall(address,bytes) (BabyDogeCoin.sol#321-323)
- Address.functionCall(address,bytes,string) (BabyDogeCoin.sol#331-333)
- Address.functionCallWithValue(address,bytes,uint256) (BabyDogeCoin.sol#346-348)
- Address.functionCallWithValue(address,bytes,uint256,string) (BabyDogeCoin.sol#356-359)
- Address.isContract(address) (BabyDogeCoin.sol#268-277)
- Address.sendValue(address,uint256) (BabyDogeCoin.sol#295-301)
- Context._msgData() (BabyDogeCoin.sol#240-243)
- SafeMath.mod(uint256,uint256) (BabyDogeCoin.sol#213-215)
- SafeMath.mod(uint256,uint256,string) (BabyDogeCoin.sol#229-232)

3.Variables is not in mixedCase
- Function IUniswapV2Pair.DOMAIN_SEPARATOR() (BabyDogeCoin.sol#499)
- Function IUniswapV2Pair.PERMIT_TYPEHASH() (BabyDogeCoin.sol#500)
- Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (BabyDogeCoin.sol#517)
- Function IUniswapV2Router01.WETH() (BabyDogeCoin.sol#539)

4.Gas saving: functions should be declared external instead public
- CoinToken.setNumTokensSellToAddToLiquidity(uint256) (BabyDogeCoin.sol#895-897)
- CoinToken.setMaxTxPercent(uint256) (BabyDogeCoin.sol#899-901)
- CoinToken.setSwapAndLiquifyEnabled(bool) (BabyDogeCoin.sol#903-906)
- CoinToken.claimTokens() (BabyDogeCoin.sol#963-965)
- CoinToken.isExcludedFromFee(address) (BabyDogeCoin.sol#994-996)



BabyDogeCoin (CoinToken) Contract

- name()
- symbol()
- decimals()
- totalSupply()
- balanceOf()
- allowance()

Read functions are running as expected while analyzing at the time of this writing.

- claimTokens()
- transferFrom()
- includeInReward()
- excludeFromFee()
- includeInFee()
- setTaxFeePercent()
- setLiquidityFeePercent()
- setNumTokensSellToAddToLiquidity()
- setMaxTxPercent()
- setSwapAndLiquidityEnabled()
- transfer()
- increaseAllowance()
- decreaseAllowance()
- approve()
- receive(), the ether receiving function

Write functions are in no risk at the time of this writing. But some infomational issues should be resolved.

Contract Ownership

Contract Ownership Has Not Been Renounced at the Time of Audit.

The contract ownership is not currently renounced.

We just placed the contract of the owner address below for you to look up:

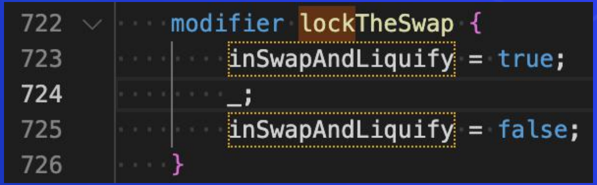
0xf103d2AbA493749a402B7dE11cF31f5844062B74

Some feasible suggestions that would also mitigate the potential risk at a different level for priviledged ownership.

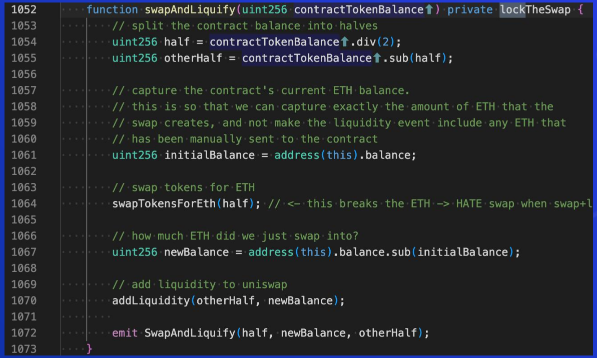
- Time-lock with reasonable latency, e.g., 48 hours for awareness on priviledged operations
- Assignment of priviledged roles to multi-signature wallets to prevent a single point of failure, for example, due to the private key compromised

Liquidity Ownership

There's a Lock Swap/Liquidity logic has Found in the Contract.



```
    modifier lockTheSwap {
        inSwapAndLiquify = true;
        _;
        inSwapAndLiquify = false;
    }
```



```
    function swapAndLiquify(uint256 contractTokenBalance) private lockTheSwap {
        // split the contract balance into halves
        uint256 half = contractTokenBalance.div(2);
        uint256 otherHalf = contractTokenBalance.sub(half);

        // capture the contract's current ETH balance.
        // this is so that we can capture exactly the amount of ETH that the
        // swap creates, and not make the liquidity event include any ETH that
        // has been manually sent to the contract
        uint256 initialBalance = address(this).balance;

        // swap tokens for ETH
        swapTokensForEth(half); // <- this breaks the ETH -> HATE swap when swap+l

        // how much ETH did we just swap into?
        uint256 newBalance = address(this).balance.sub(initialBalance);

        // add liquidity to uniswap
        addLiquidity(otherHalf, newBalance);

        emit SwapAndLiquify(half, newBalance, otherHalf);
    }
```

The lockTheSwap is for preventing repeatedly to send transactions on PancakeSwap; To lock before the transaction and to unlock if everything work well as expected.

MoonAudit.org

Mint Function

The Contract Cannot Mint New $BABYDOGE Tokens.

We do understand that Mint functions are crucial to the functionality of the project, it's core related to its investors.

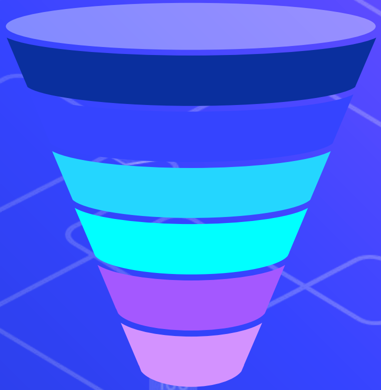
But a mint function was not found in the contract code.

Burn Function

The Contract does not have a Burn Function.

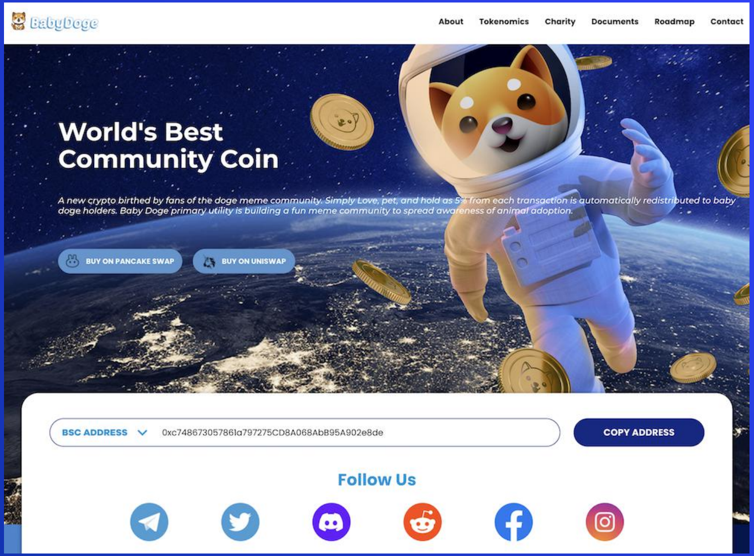
Although there's no burn function implemented in the contract, but people or the owner can still send tokens to the zero address periodically.

Sending to zero address, this kind of behavior is resemblance to the burn function.



MoonAudit.org

Present Mode



World's Best Community Coin

A new crypto birthed by fans of the doge meme community. Simply Love, pet, and hold as 5% from each transaction is automatically redistributed to baby doge holders. Baby Doge primary utility is building a fun meme community to spread awareness of animal adoption.

BSC ADDRESS 0xc748673057861a797275CD8A068AbB95A902e8de

The left image is an actual snapshot of the current live website.

The website was registered on Feb-02-2021.

Team Location

BabyDogeCoin Official Website: https://babydoge.com/

MoonAudit.org

General Web Security

DOMAIN

A valid domain hosted by GoDaddy.com.

Registered on 02-Feb-2021

babydoge.com

Social Media Accounts

A bundle of social media accounts was found.

Twitter: https://twitter.com/babydogecoin
https://discord.com/invite/babydogecoin
https://instagram.com/thebabydogecoin

A legal SSL certificate was found.
Expired at 14-Dec-2022

Signature Algorithm is sha256WithRSAEncryption

SSL CERTIFICATE

No malware found.
No injected spam found.
No internal server errors.

Domain is marked clean by Google and McAfee.

SPAM/MALWARE

Disclaimer

The opinions expressed in this document are for general informational purposes only and are not intended to provide specific advice or recommendations for any individual or on any specific investment. It is only intended to provide education and public knowledge regarding to this projects. This audit is only applied to the type of auditing specified in this report and the scope

of given in the results. Other unknown security vulnerabilities are beyond responsibility. MoonAudit only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, MoonAudit lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The smart contract analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to MoonAudit or was publicly available before the issuance of this report (issuance of report recorded via block number on cover page), if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, MoonAudit assumes no responsibility for the resulting loss or adverse effects. Due to the technical limitations of any organization, this report conducted by MoonAudit still has the possibility that the entire risk cannot be completely detected. MoonAudit disclaims any liability for the resulting losses.

MoonAudit provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Even projects with a low risk score have been known to pull liquidity, sell all team tokens, or exit scam. Please exercise caution when dealing with any cryptocurrency related platforms. The final interpretation of this statement belongs to MoonAudit.

MoonAudit highly advises against using cryptocurrencies as speculative investments and they should be used solely for the utility they aim to provide.

About

MoonAudit has founded in 2021 by a squad of elite geeks on blockchain research and we analyze the loopholes in most smart contracts in ethereum-based chains. We offer the best-in-class report for your smart contracts auditing. Customer trusts smart contract, more trust security assessment report.

Your Smart Contract, MoonAudit Report.

We guard the blockchain security.

Thank You

MOONAUDIT ORG. HAS BEEN COMPLETED FOR BABYDOGE AT BLOCK NUMBER: 19250729

THIS AUDIT IS ONLY VALID IF VIEWED ON HTTPS://MOONAUDIT.ORG

https://MoonAudit.org

https://t.me/MoonAudit