auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

BabyDrake

DATED : 14 Feb, 2024

AUDIT SUMMARY

Project name – Baby Drake

Date: 14 Feb, 2024

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	1	0	2
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 conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0xbdc9f41f076b869ff3bed081f26c2c8d9d580431#code

Token Information

Token Name : Baby Drake

Token Symbol: BabyDrake

Decimals: 9

Token Supply: 420000000000000000

Network: Binance smart chain

Token Type: BEP-20

Token Address:
0x7836Ab4ae4d04A48F0A7b25ea7359356A70F4aA4

Checksum:
A2032c616934aeb47e6039f76b20d531

Owner:
0xafCd5Ddd27a4062E404936BCaD4D7f3216aF4240
(at time of writing the audit)

Deployer:
0xafCd5Ddd27a4062E404936BCaD4D7f3216aF4240

TOKEN OVERVIEW

Fees:
Buy Tax: 5%
Sell Tax: 5%
Marketing Tax: 5%

Fees Privilege: Owner

Ownership: Owned

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

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.

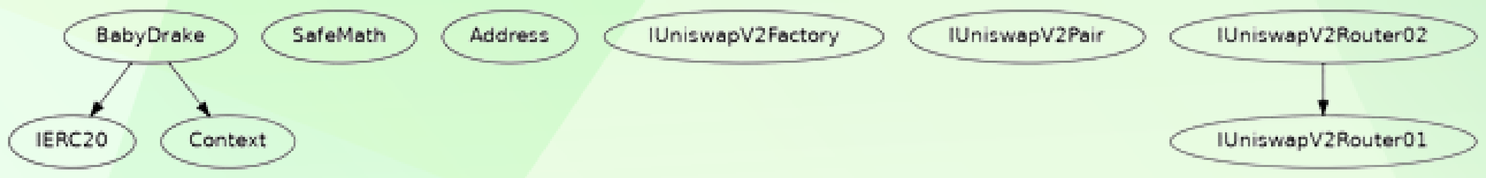
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.

- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

- ✓ Return values of low-level calls
- ✓ Gasless Send
- ✓ Private modifier
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Re-entrancy
- ✓ Using Suicide
- ✓ Tautology or contradiction
- ✓ Gas Limitand Loops
- ✓ Timestamp Dependence
- ✓ Address hardcoded
- ✓ Revert/require functions
- ✓ Exception Disorder
- ✓ Use of tx.origin
- ✓ Using inline assembly
- ✓ Integer overflow/underflow
- ✓ Divide before multiply
- ✓ Dangerous strict equalities
- ✓ Missing Zero Address Validation
- ✓ Using SHA3
- ✓ Compiler version not fixed
- ✓ Using throw

INHERITANCE TREE

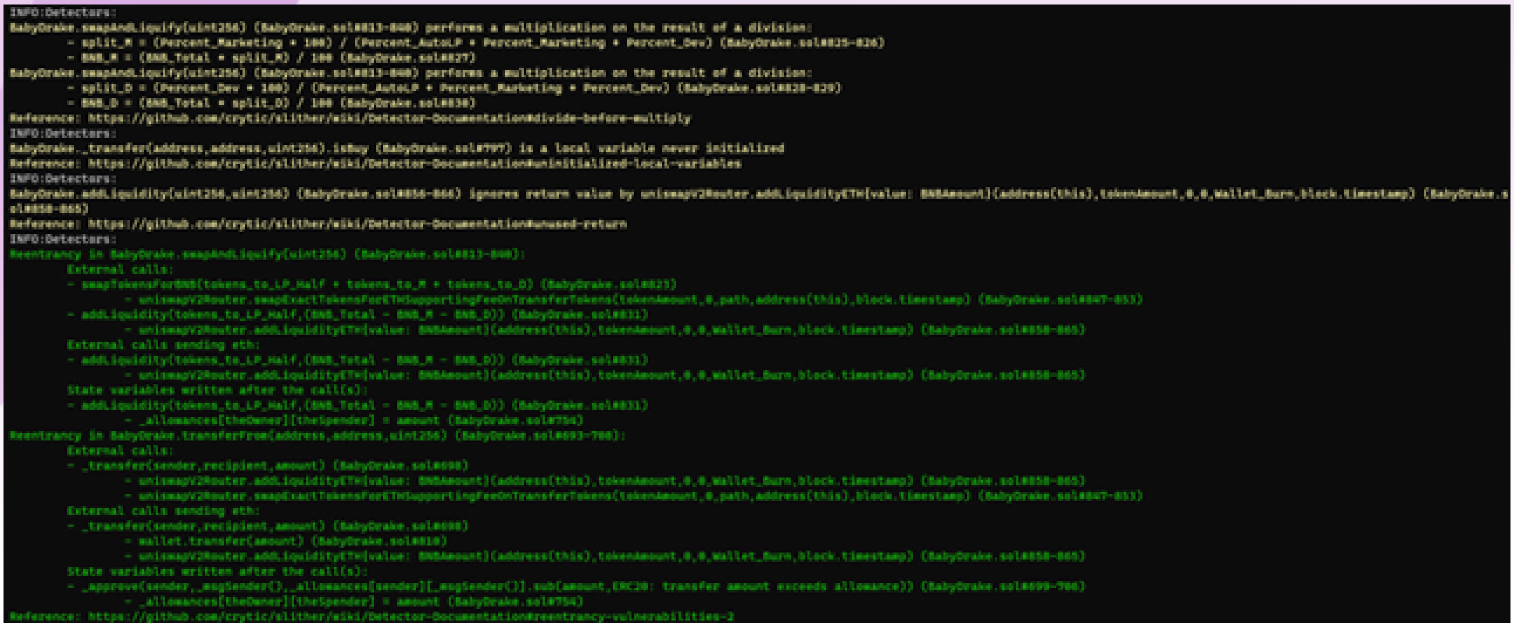


$$\begin{tikzcd}
\text{BabyDrake} \arrow[d] \arrow[dr] & & \text{SafeMath} & \text{Address} & \text{IUniswapV2Factory} & \text{IUniswapV2Pair} & \text{IUniswapV2Router02} \arrow[d] \\
\text{IERC20} & \text{Context} & & & & & \text{IUniswapV2Router01}
\end{tikzcd}$$

STATIC ANALYSIS

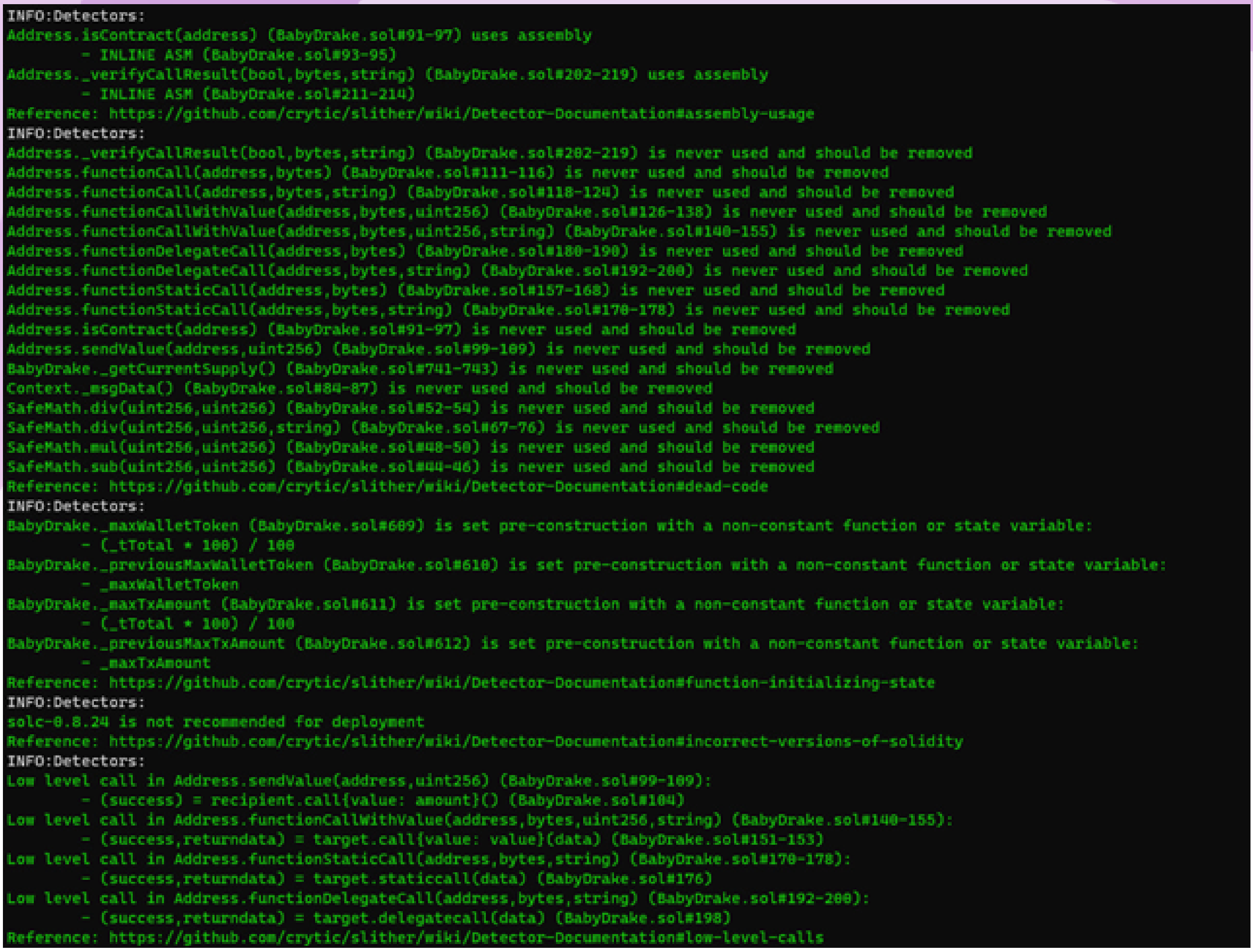
A static analysis of the code was performed using Slither.

No issues were found.

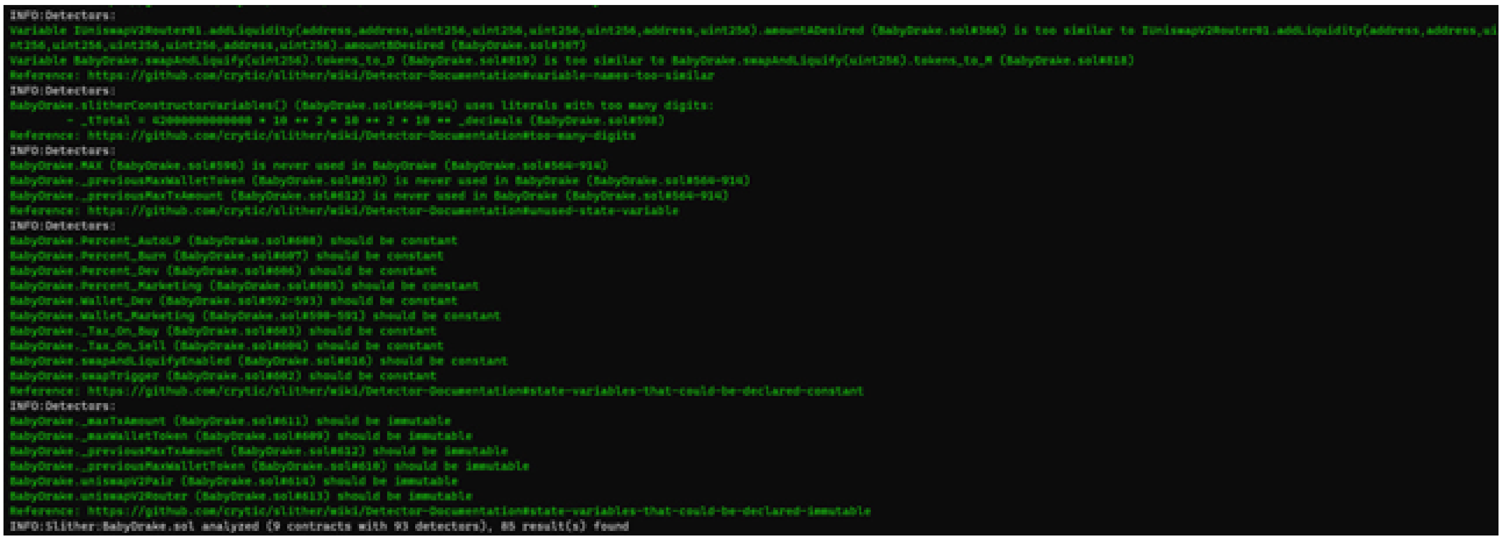


```
INFO:Detectors:
BabyDrake.swapAndLiquify(uint256) (BabyDrake.sol#813-840) performs a multiplication on the result of a division:
        - split_M = (Percent_Marketing * 100) / (Percent_AutoLP + Percent_Marketing + Percent_Dev) (BabyDrake.sol#825-826)
        - BNB_M = (BNB_Total * split_M) / 100 (BabyDrake.sol#827)
BabyDrake.swapAndLiquify(uint256) (BabyDrake.sol#813-840) performs a multiplication on the result of a division:
        - split_D = (Percent_Dev * 100) / (Percent_AutoLP + Percent_Marketing + Percent_Dev) (BabyDrake.sol#828-829)
        - BNB_D = (BNB_Total * split_D) / 100 (BabyDrake.sol#830)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
BabyDrake._transfer(address,address,uint256).isBuy (BabyDrake.sol#797) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
BabyDrake.addLiquidity(uint256,uint256) (BabyDrake.sol#856-866) ignores return value by uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Burn,block.timestamp) (BabyDrake.s
ol#858-865)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in BabyDrake.swapAndLiquify(uint256) (BabyDrake.sol#813-840):
        External calls:
        - swapTokensForBNB(tokens_to_LP_Half + tokens_to_M + tokens_to_D) (BabyDrake.sol#823)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (BabyDrake.sol#847-853)
        - addLiquidity(tokens_to_LP_Half,(BNB_Total - BNB_M - BNB_D)) (BabyDrake.sol#831)
                - uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Burn,block.timestamp) (BabyDrake.sol#858-865)
        External calls sending eth:
        - addLiquidity(tokens_to_LP_Half,(BNB_Total - BNB_M - BNB_D)) (BabyDrake.sol#831)
                - uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Burn,block.timestamp) (BabyDrake.sol#858-865)
        State variables written after the call(s):
        - addLiquidity(tokens_to_LP_Half,(BNB_Total - BNB_M - BNB_D)) (BabyDrake.sol#831)
                - _allowances[theOwner][theSpender] = amount (BabyDrake.sol#754)
Reentrancy in BabyDrake.transferFrom(address,address,uint256) (BabyDrake.sol#693-700):
        External calls:
        - _transfer(sender,recipient,amount) (BabyDrake.sol#698)
                - uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Burn,block.timestamp) (BabyDrake.sol#858-865)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (BabyDrake.sol#847-853)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (BabyDrake.sol#698)
                - wallet.transfer(amount) (BabyDrake.sol#810)
                - uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Burn,block.timestamp) (BabyDrake.sol#858-865)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (BabyDrake.sol#699-700)
                - _allowances[theOwner][theSpender] = amount (BabyDrake.sol#754)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

STATIC ANALYSIS



```
INFO:Detectors:
Address.isContract(address) (BabyDrake.sol#91-97) uses assembly
        - INLINE ASM (BabyDrake.sol#93-95)
Address._verifyCallResult(bool,bytes,string) (BabyDrake.sol#202-219) uses assembly
        - INLINE ASM (BabyDrake.sol#211-214)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (BabyDrake.sol#202-219) is never used and should be removed
Address.functionCall(address,bytes) (BabyDrake.sol#111-116) is never used and should be removed
Address.functionCall(address,bytes,string) (BabyDrake.sol#118-124) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (BabyDrake.sol#126-138) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (BabyDrake.sol#140-155) is never used and should be removed
Address.functionDelegateCall(address,bytes) (BabyDrake.sol#180-190) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (BabyDrake.sol#192-200) is never used and should be removed
Address.functionStaticCall(address,bytes) (BabyDrake.sol#157-168) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (BabyDrake.sol#170-178) is never used and should be removed
Address.isContract(address) (BabyDrake.sol#91-97) is never used and should be removed
Address.sendValue(address,uint256) (BabyDrake.sol#99-109) is never used and should be removed
BabyDrake._getCurrentSupply() (BabyDrake.sol#741-743) is never used and should be removed
Context._msgData() (BabyDrake.sol#84-87) is never used and should be removed
SafeMath.div(uint256,uint256) (BabyDrake.sol#52-54) is never used and should be removed
SafeMath.div(uint256,uint256,string) (BabyDrake.sol#67-76) is never used and should be removed
SafeMath.mul(uint256,uint256) (BabyDrake.sol#48-50) is never used and should be removed
SafeMath.sub(uint256,uint256) (BabyDrake.sol#44-46) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
BabyDrake._maxWalletToken (BabyDrake.sol#609) is set pre-construction with a non-constant function or state variable:
        - (_tTotal * 100) / 100
BabyDrake._previousMaxWalletToken (BabyDrake.sol#610) is set pre-construction with a non-constant function or state variable:
        - _maxWalletToken
BabyDrake._maxTxAmount (BabyDrake.sol#611) is set pre-construction with a non-constant function or state variable:
        - (_tTotal * 100) / 100
BabyDrake._previousMaxTxAmount (BabyDrake.sol#612) is set pre-construction with a non-constant function or state variable:
        - _maxTxAmount
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state
INFO:Detectors:
solc-0.8.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (BabyDrake.sol#99-109):
        - (success) = recipient.call{value: amount}() (BabyDrake.sol#104)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (BabyDrake.sol#140-155):
        - (success,returndata) = target.call{value: value}(data) (BabyDrake.sol#151-153)
Low level call in Address.functionStaticCall(address,bytes,string) (BabyDrake.sol#170-178):
        - (success,returndata) = target.staticcall(data) (BabyDrake.sol#176)
Low level call in Address.functionDelegateCall(address,bytes,string) (BabyDrake.sol#192-200):
        - (success,returndata) = target.delegatecall(data) (BabyDrake.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```



```
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (BabyDrake.sol#366) is too similar to IUniswapV2Router01.addLiquidity(address,address,ui
nt256,uint256,uint256,uint256,address,uint256).amountBDesired (BabyDrake.sol#367)
Variable BabyDrake.swapAndLiquify(uint256).tokens_to_D (BabyDrake.sol#819) is too similar to BabyDrake.swapAndLiquify(uint256).tokens_to_M (BabyDrake.sol#818)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
BabyDrake.slitherConstructorVariables() (BabyDrake.sol#564-914) uses literals with too many digits:
        - _tTotal = 4200000000000 * 10 ** 2 * 10 ** 2 * 10 ** _decimals (BabyDrake.sol#598)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
BabyDrake.MAX (BabyDrake.sol#596) is never used in BabyDrake (BabyDrake.sol#564-914)
BabyDrake._previousMaxWalletToken (BabyDrake.sol#610) is never used in BabyDrake (BabyDrake.sol#564-914)
BabyDrake._previousMaxTxAmount (BabyDrake.sol#612) is never used in BabyDrake (BabyDrake.sol#564-914)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
BabyDrake.Percent_AutoLP (BabyDrake.sol#608) should be constant
BabyDrake.Percent_Burn (BabyDrake.sol#607) should be constant
BabyDrake.Percent_Dev (BabyDrake.sol#606) should be constant
BabyDrake.Percent_Marketing (BabyDrake.sol#605) should be constant
BabyDrake.Wallet_Dev (BabyDrake.sol#592-593) should be constant
BabyDrake.Wallet_Marketing (BabyDrake.sol#590-591) should be constant
BabyDrake._Tax_On_Buy (BabyDrake.sol#603) should be constant
BabyDrake._Tax_On_Sell (BabyDrake.sol#604) should be constant
BabyDrake.swapAndLiquifyEnabled (BabyDrake.sol#616) should be constant
BabyDrake.swapTrigger (BabyDrake.sol#602) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
BabyDrake._maxTxAmount (BabyDrake.sol#611) should be immutable
BabyDrake._maxWalletToken (BabyDrake.sol#609) should be immutable
BabyDrake._previousMaxTxAmount (BabyDrake.sol#612) should be immutable
BabyDrake._previousMaxWalletToken (BabyDrake.sol#610) should be immutable
BabyDrake.uniswapV2Pair (BabyDrake.sol#614) should be immutable
BabyDrake.uniswapV2Router (BabyDrake.sol#613) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:BabyDrake.sol analyzed (9 contracts with 93 detectors), 85 result(s) found
```

FUNCTIONAL TESTING

1- Approve (passed)**:**
https://testnet.bscscan.com/tx/0xf00707b66ac0559e37c156367ace7282ac6398e7e0c2123b715fb1c89be25e97

2- Increase Allowance (passed)**:**
https://testnet.bscscan.com/tx/0x66ba12a7d372780fd3c8d2496f1d45cfda752e1ebb79ff4ebadcc185891cedd6

3- Decrease Allowance (passed)**:**
https://testnet.bscscan.com/tx/0x2eaaed4ba307784054651397 82da15774bb5dadc596e88df259e1d0cae0aba8b

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.

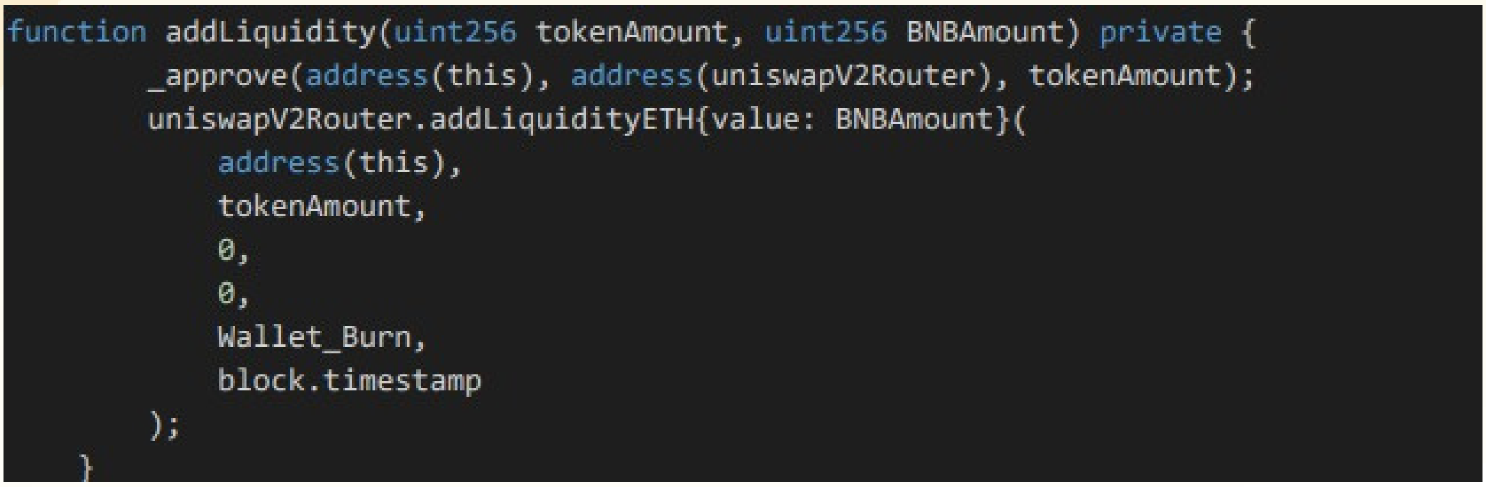
Findings

Severity	Found
◆ **Critical**	0
◆ **High-Risk**	0
◆ **Medium-Risk**	1
◆ **Low-Risk**	0
◆ **Gas Optimization / Suggestions**	2

MANUAL TESTING

Centralization – Liquidity is added to EOA
Severity: Medium
Function: Add Liquidity
Status: Open

Overview:
Liquidity is adding to EOA. It may be drained by the Wallet Burn.



```
function addLiquidity(uint256 tokenAmount, uint256 BNBAmount) private {
        _approve(address(this), address(uniswapV2Router), tokenAmount);
        uniswapV2Router.addLiquidityETH{value: BNBAmount}(
            address(this),
            tokenAmount,
            0,
            0,
            Wallet_Burn,
            block.timestamp
        );
    }
```

Suggestion:
It is suggested that the address should be a contract address or a dead address.

MANUAL TESTING

Optimization

Severity: Informational

Subject: Remove Safe Math

Status: Open

Line: 39-77

Overview:
compiler version above 0.8.0 can control arithmetic overflow/underflow, it is recommended to remove the unwanted code to avoid high gas fees.

MANUAL TESTING

Optimization

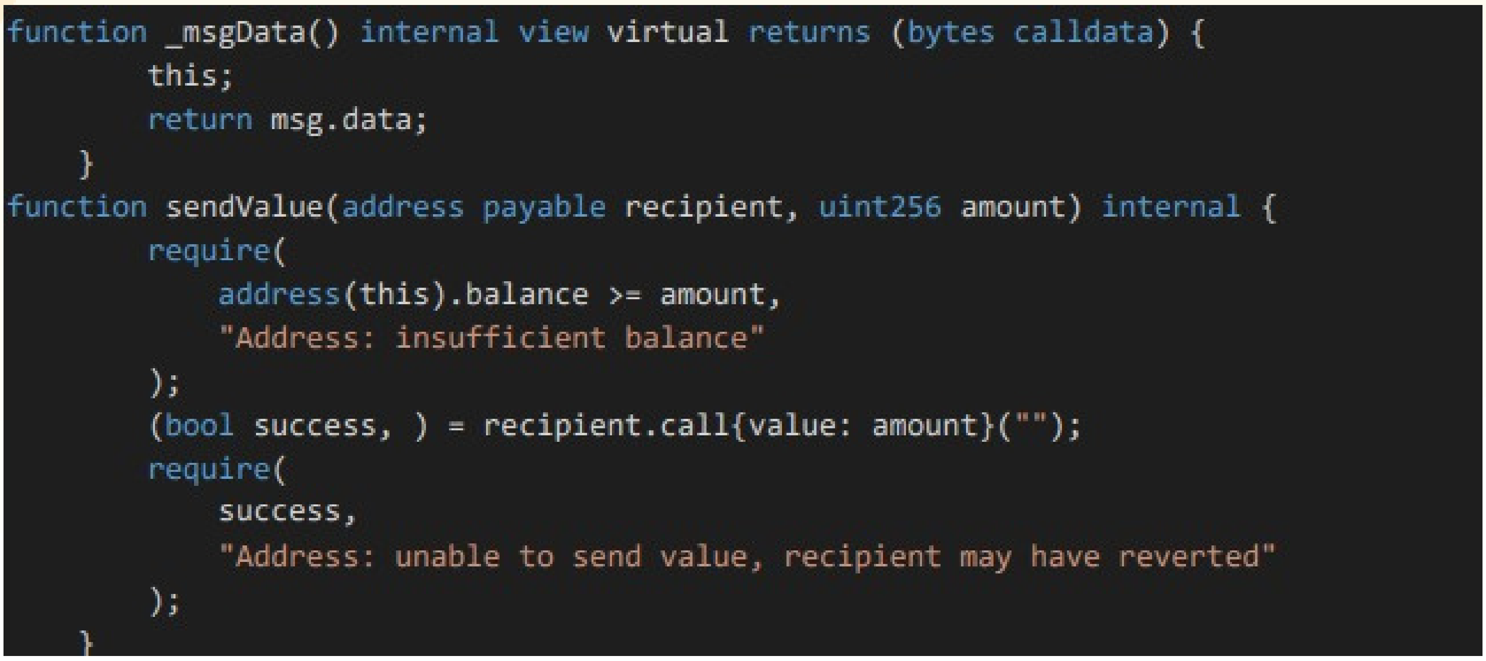
Severity: Optimization

Subject: Remove unused code

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice. though to avoid them.



```
function _msgData() internal view virtual returns (bytes calldata) {
        this;
        return msg.data;
    }
function sendValue(address payable recipient, uint256 amount) internal {
        require(
            address(this).balance >= amount,
            "Address: insufficient balance"
        );
        (bool success, ) = recipient.call{value: amount}("");
        require(
            success,
            "Address: unable to send value, recipient may have reverted"
        );
    }
```

DISCLAIMER

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