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	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 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/0xeab79b7fa6d1af6a0a96f27463d6cce7de23633b#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:
0x9243CEC02656f9FF55c09df3665C76605C1a913f

Checksum:
A2032c616934aeb47e6039f76b20d221

Owner:
0x860A19f92FC2390eDc775bC3AC34F57425bEEB23
(at time of writing the audit)

Deployer:
0x4AC8cb73913a9A7e34f82Fac6877af647673210b

TOKEN OVERVIEW

Fees:
Buy Tax: 5%
Sell Tax: 5%
Marketing Tax: 0%

Fees Privilege: Owner

Ownership: Owned

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

AUDIT METHODOLOGY

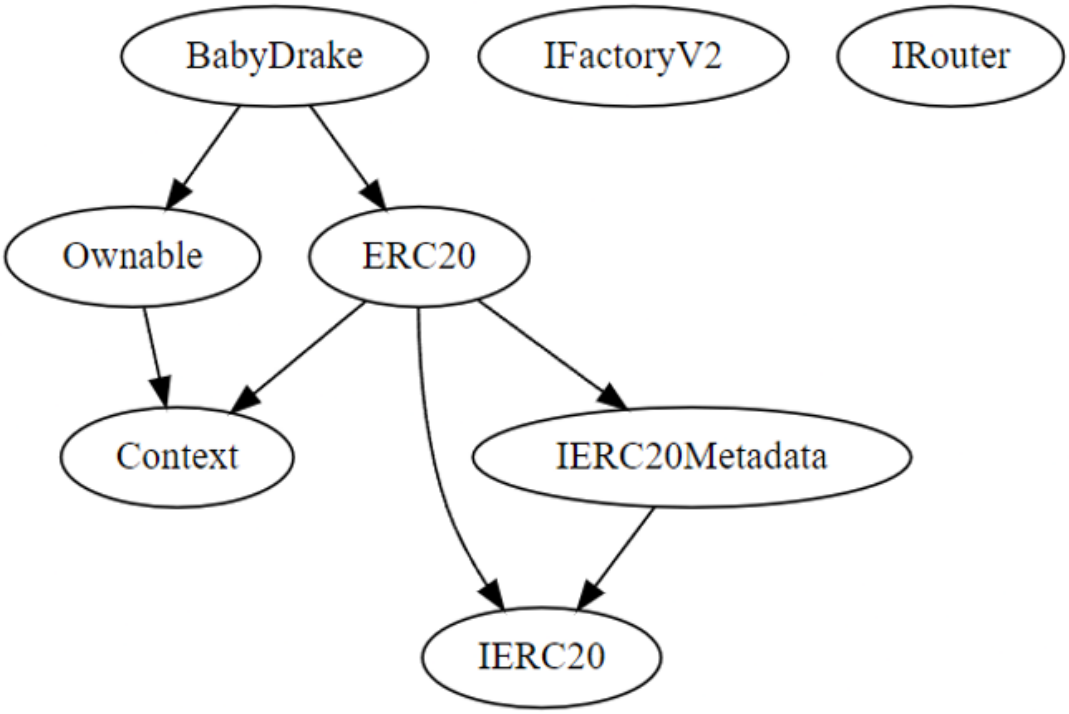
The auditing process will follow a routine as special considerations by Auditace:

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

INHERITANCE TREE



STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.

```
INFO:Detectors:
Reentrancy in BabyDrake._performInternalSwap() (BabyDrake.sol#695-701):
        External calls:
        - _internalSwap() (BabyDrake.sol#698)
                - IRouter(uniswapRouter).swapExactTokensForETHSupportingFeeOnTransferTokens(tokenBalance,0,path,marketingWallet,block.timestamp) (BabyDrake.sol#675-686)
        State variables written after the call(s):
        - swapping = false (BabyDrake.sol#699)
        BabyDrake.swapping (BabyDrake.sol#602) can be used in cross function reentrancies:
        - BabyDrake._performInternalSwap() (BabyDrake.sol#695-701)
Reentrancy in BabyDrake._transfer(address,address,uint256) (BabyDrake.sol#704-722):
        External calls:
        - _performInternalSwap() (BabyDrake.sol#715)
                - IRouter(uniswapRouter).swapExactTokensForETHSupportingFeeOnTransferTokens(tokenBalance,0,path,marketingWallet,block.timestamp) (BabyDrake.sol#675-686)
        - _performInternalSwap() (BabyDrake.sol#717)
                - IRouter(uniswapRouter).swapExactTokensForETHSupportingFeeOnTransferTokens(tokenBalance,0,path,marketingWallet,block.timestamp) (BabyDrake.sol#675-686)
        State variables written after the call(s):
        - super._transfer(_from,address(this),feeAmount) (BabyDrake.sol#719)
                - _balances[from] = fromBalance - amount (BabyDrake.sol#351)
                - _balances[to] += amount (BabyDrake.sol#353)
        ERC20._balances (BabyDrake.sol#160) can be used in cross function reentrancies:
        - ERC20._mint(address,uint256) (BabyDrake.sol#369-379)
        - ERC20._transfer(address,address,uint256) (BabyDrake.sol#338-358)
        - ERC20.balanceOf(address) (BabyDrake.sol#213-215)
        - super._transfer(_from,_to,_amount - feeAmount) (BabyDrake.sol#721)
                - _balances[from] = fromBalance - amount (BabyDrake.sol#351)
                - _balances[to] += amount (BabyDrake.sol#353)
        ERC20._balances (BabyDrake.sol#160) can be used in cross function reentrancies:
        - ERC20._mint(address,uint256) (BabyDrake.sol#369-379)
        - ERC20._transfer(address,address,uint256) (BabyDrake.sol#338-358)
        - ERC20.balanceOf(address) (BabyDrake.sol#213-215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
BabyDrake._transfer(address,address,uint256).feeAmount (BabyDrake.sol#709) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in BabyDrake._transfer(address,address,uint256) (BabyDrake.sol#704-722):
        External calls:
        - _performInternalSwap() (BabyDrake.sol#715)
                - IRouter(uniswapRouter).swapExactTokensForETHSupportingFeeOnTransferTokens(tokenBalance,0,path,marketingWallet,block.timestamp) (BabyDrake.sol#675-686)
        - _performInternalSwap() (BabyDrake.sol#717)
                - IRouter(uniswapRouter).swapExactTokensForETHSupportingFeeOnTransferTokens(tokenBalance,0,path,marketingWallet,block.timestamp) (BabyDrake.sol#675-686)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (BabyDrake.sol#355)
                - super._transfer(_from,_to,_amount - feeAmount) (BabyDrake.sol#721)
        - Transfer(from,to,amount) (BabyDrake.sol#355)
                - super._transfer(_from,address(this),feeAmount) (BabyDrake.sol#719)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (BabyDrake.sol#14-16) is never used and should be removed
ERC20._burn(address,uint256) (BabyDrake.sol#392-407) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in BabyDrake.withdrawETH() (BabyDrake.sol#730-733):
        - (success) = msg.sender.call{value: address(this).balance}() (BabyDrake.sol#731)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IRouter.WETH() (BabyDrake.sol#584) is not in mixedCase
Parameter BabyDrake.setWhitelisted(address,bool)._user (BabyDrake.sol#629) is not in mixedCase
Parameter BabyDrake.setWhitelisted(address,bool)._yesno (BabyDrake.sol#629) is not in mixedCase
Parameter BabyDrake.withdrawERC20Tokens(address)._token (BabyDrake.sol#725) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
BabyDrake.buyFee (BabyDrake.sol#599) should be constant
BabyDrake.sellFee (BabyDrake.sol#600) should be constant
BabyDrake.uniswapFactory (BabyDrake.sol#596) should be constant
BabyDrake.uniswapRouter (BabyDrake.sol#597) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
BabyDrake.pair (BabyDrake.sol#598) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:BabyDrake.sol analyzed (8 contracts with 93 detectors), 17 result(s) found
```

FUNCTIONAL TESTING

1- Approve (passed):
https://testnet.bscscan.com/tx/0x69cb572b0a1b1aa327783894582cd6b06dc66fc4ce24b64af5df693ddfe2a989

2- Increase Allowance (passed):
https://testnet.bscscan.com/tx/0x1cd04e648d867e3ecb3501dda0df8aef18c9cf2982319a58cd85715466240021

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0xa77cfbdb6d776e51269fc71a4db9fd6b3a9bb8a003af304ffc58576c058011dd

4- Set Whitelisted (passed):
https://testnet.bscscan.com/tx/0xd51e90f72f86f820a469a3e14d0584c02132407a480a7641514b130c3f9add04

5- Update Marketing Wallet (passed):
https://testnet.bscscan.com/tx/0xcbe94734bea9beed8523013a259f589214b4a893945111f18d853436bbac7dbf

POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can whitelist the address.
- The owner can update the marketing wallet address.
- The owner can withdraw ETH.

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	1

MANUAL TESTING

Centralization – Missing Require Check.

Severity: Medium

Function: UpdateMarketingWallet

Status: Open

Overview:

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner will set the address to the contract address, then the Eth will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
function updateMarketingWallet(address marketing_) public onlyOwner {
        require(marketing_ != address(0), "address zero not accepted");
        marketingWallet = marketing_;
        emit MarketinWalletUpdated(marketing_);
    }
```

Suggestion: It is recommended that the address should not be able to set as a contract address.

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) {
        return msg.data;
    }
```

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