auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

BOOK OF MEME ON BSC

DATED : 21 March, 2024

AUDIT SUMMARY

Project name – BOOK OF MEME ON BSC

Date: 21 March, 2024

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

Audit Status: **Passed with high risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	2	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 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/0x11ab793576bd2b6b906f3454413b771a5d5dca5f#code

Token Information

Token Name : BOOK OF MEME ON BSC

Token Symbol: $BOME

Decimals: 9

Token Supply: 420690000000

Network: BscScan

Token Type: BEP-20

Token Address:
0x5bD8107d7524590F9E0617F1b21ac19a104A6044

Checksum:
A2032c616934aeb47e6039f76b20d231

Owner:
0x83fe178DBca67bFaA4BC8247225568AfB3c06Cf1
(at time of writing the audit)

Deployer:
0x83fe178DBca67bFaA4BC8247225568AfB3c06Cf1

TOKEN OVERVIEW

Fees:

Buy Fee: 5%

Sell Fee: 5%

Transfer Fee: 0%

Fees Privilege: Owner

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

AUDIT METHODOLOGY

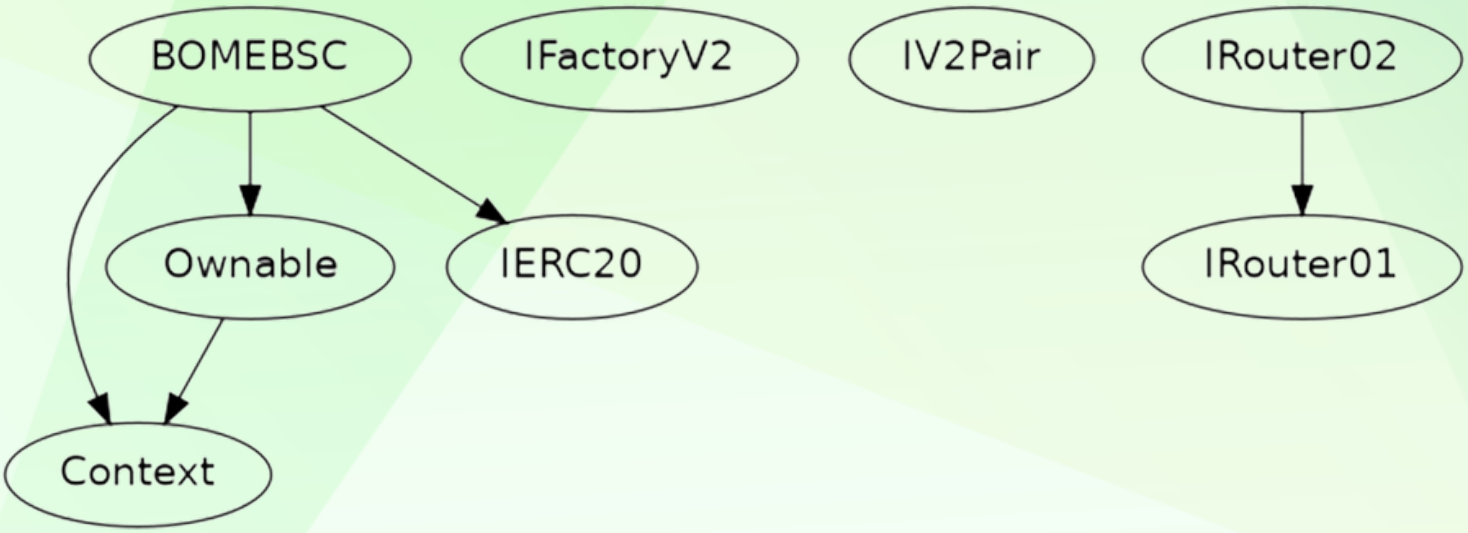
The auditing process will follow a routine as special considerations by Auditace:

- 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

- [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

INHERITANCE TREE



STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.

```
INFO:Detectors:
BOMEBSC.swapAndLiquify(uint256) (BOMEBSC.sol#366-399) ignores return value by swapRouter.addLiquidityETH{value: newBalance}(address(this),secondMath,0,0,DEA
D,block.timestamp) (BOMEBSC.sol#389-396)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in BOMEBSC._transfer(address,address,uint256) (BOMEBSC.sol#314-341):
        External calls:
        - internalSwap((contractTokenBalance * (buyAllocation + sellAllocation)) / 100) (BOMEBSC.sol#328)
                - swapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(contractTokenBalance,0,path,address(this),block.timestamp) (BOMEBSC.sol#411-
419)
                - (success,None) = marketingAddress.call{gas: 35000,value: mktAmount}() (BOMEBSC.sol#425)
                - (success,None) = devWallet.call{gas: 35000,value: devAmount}() (BOMEBSC.sol#426)
        - swapAndLiquify(contractTokenBalance * liquidityAllocation / 100) (BOMEBSC.sol#329)
                - swapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(firstmath,0,path,address(this),block.timestamp) (BOMEBSC.sol#380-385)
                - swapRouter.addLiquidityETH{value: newBalance}(address(this),secondMath,0,0,DEAD,block.timestamp) (BOMEBSC.sol#389-396)
        External calls sending eth:
        - internalSwap((contractTokenBalance * (buyAllocation + sellAllocation)) / 100) (BOMEBSC.sol#328)
                - (success,None) = marketingAddress.call{gas: 35000,value: mktAmount}() (BOMEBSC.sol#425)
                - (success,None) = devWallet.call{gas: 35000,value: devAmount}() (BOMEBSC.sol#426)
        - swapAndLiquify(contractTokenBalance * liquidityAllocation / 100) (BOMEBSC.sol#329)
                - swapRouter.addLiquidityETH{value: newBalance}(address(this),secondMath,0,0,DEAD,block.timestamp) (BOMEBSC.sol#389-396)
        Event emitted after the call(s):
        - SwapAndLiquify() (BOMEBSC.sol#398)
                - swapAndLiquify(contractTokenBalance * liquidityAllocation / 100) (BOMEBSC.sol#329)
        - Transfer(from,address(this),feeAmount) (BOMEBSC.sol#360)
                - amountAfterFee = takeTaxes(from,is_buy(from,to),is_sell(from,to),amount) (BOMEBSC.sol#336)
        - Transfer(from,to,amountAfterFee) (BOMEBSC.sol#337)
Reentrancy in BOMEBSC.swapAndLiquify(uint256) (BOMEBSC.sol#366-399):
        External calls:
        - swapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(firstmath,0,path,address(this),block.timestamp) (BOMEBSC.sol#380-385)
        - swapRouter.addLiquidityETH{value: newBalance}(address(this),secondMath,0,0,DEAD,block.timestamp) (BOMEBSC.sol#389-396)
        External calls sending eth:
        - swapRouter.addLiquidityETH{value: newBalance}(address(this),secondMath,0,0,DEAD,block.timestamp) (BOMEBSC.sol#389-396)
        Event emitted after the call(s):
        - SwapAndLiquify() (BOMEBSC.sol#398)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
Pragma version=0.8.19 (BOMEBSC.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in BOMEBSC.internalSwap(uint256) (BOMEBSC.sol#401-427):
        - (success,None) = marketingAddress.call{gas: 35000,value: mktAmount}() (BOMEBSC.sol#425)
        - (success,None) = devWallet.call{gas: 35000,value: devAmount}() (BOMEBSC.sol#426)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IRouter01.WETH() (BOMEBSC.sol#71) is not in mixedCase
Event BOMEBSC._enableTrading() (BOMEBSC.sol#204) is not in CapWords
Event BOMEBSC._setPresaleAddress(address,bool) (BOMEBSC.sol#205) is not in CapWords
Event BOMEBSC._toggleCanSwapFees(bool) (BOMEBSC.sol#206) is not in CapWords
Event BOMEBSC._changePair(address) (BOMEBSC.sol#207) is not in CapWords
Event BOMEBSC._changeThreshold(uint256) (BOMEBSC.sol#208) is not in CapWords
Event BOMEBSC._changeWallets(address,address) (BOMEBSC.sol#209) is not in CapWords
Event BOMEBSC._changeFees(uint256,uint256) (BOMEBSC.sol#210) is not in CapWords
Function BOMEBSC.is_buy(address,address) (BOMEBSC.sol#287-290) is not in mixedCase
Function BOMEBSC.is_sell(address,address) (BOMEBSC.sol#292-295) is not in mixedCase
Constant BOMEBSC._name (BOMEBSC.sol#188) is not in UPPER_CASE_WITH_UNDERSCORES
Constant BOMEBSC._symbol (BOMEBSC.sol#189) is not in UPPER_CASE_WITH_UNDERSCORES
Constant BOMEBSC._decimals (BOMEBSC.sol#191) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (BOMEBSC.sol#17)" inContext (BOMEBSC.sol#8-20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (BOMEBSC.sol#83) is too similar to IRouter01
.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (BOMEBSC.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
BOMEBSC.buyAllocation (BOMEBSC.sol#182) should be constant
BOMEBSC.liquidityAllocation (BOMEBSC.sol#184) should be constant
BOMEBSC.sellAllocation (BOMEBSC.sol#183) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0xa4b0781c7084d5f1d864f5e37ba2332afc43cc2fa005b921e15638cbe33fffe8

2- Change Wallets (passed):

https://testnet.bscscan.com/tx/0x364b36964d3687696798097 04b352fde5184a830136143bf5061fa9d13805e4d

3- Enable Trading (passed):

https://testnet.bscscan.com/tx/0xa0eb964f4fa4dd52f82f04f9ac36b350fcc2bf318e674cf840988464458c4203

4- Transfer (passed):

https://testnet.bscscan.com/tx/0x084ea0f840de94a9409ca052a6a53fdf7a3b8e4967550b01ab572da071ba31ae

POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can set no fee wallet.
- The owner can changeLp pair address.
- The owner set change wallets.
- The owner can set a pre-sale address.
- The owner can enable trading.

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	2
Medium-Risk	0
Low-Risk	1
Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

Function: Enabling Trades

Status: Open

Overview:

The Enable Trading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function enableTrading() external onlyOwner {
            require(!isTradingEnabled, "Trading already enabled");
            isTradingEnabled = true;
            emit _enableTrading();
        }
```

Suggestion:

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner.

MANUAL TESTING

Centralization – Missing Require Check

Severity: High

Function: ChangeWallets

Status: Open

Overview:

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner sets 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 changeWallets(address newBuy, address newDev) external onlyOwner {
        require(newBuy != address(0),"BOME: Address Zero");
        require(newDev != address(0),"BOME: Address Zero");
        marketingAddress = payable(newBuy);
        devWallet = payable(newDev);
        emit _changeWallets(newBuy, newDev);
    }
```

Suggestion:

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

MANUAL TESTING

Centralization – Missing Zero Address

Severity: Low

Status: Open

Overview:

Functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
function setNoFeeWallet(address account, bool enabled) public onlyOwner {
        _noFee[account] = enabled;
    }
```

Suggestion:

It is suggested that the address should not be zero or dead.

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 to avoid them.

```
function _msgData() internal view returns (bytes memory) {
        this;
        return msg.data;
    }
}
interface IV2Pair {
    function factory() external view returns (address);
    function getReserves() external view returns (uint112 reserve0, uint112
reserve1, uint32 blockTimestampLast);
    function sync() external;
}
```

Suggestion:

To reduce high gas fees. It is suggested to remove unused code from the contract.

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