auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Monbie

DATED : 06 November 23'

MANUAL TESTING

Centralization – Buy and Sell
Severity: High
function: setBuyTaxes/setSellTaxes
Status: Open

Overview:
The owner can set the buy and sell fees to more than 25% which is not recommended.

```
 function setBuyTaxes(uint256 newLiquidityTax, uint256 newMarketingTax,
uint256 newBuyBackBurnTax) external onlyOwner() {
    _buyLiquidityFee = newLiquidityTax;
    _buyMarketingFee = newMarketingTax;
    _buyBuyBackBurnFee = newBuyBackBurnTax;

    _totalTaxIfBuying =
_buyLiquidityFee.add(_buyMarketingFee).add(_buyBuyBackBurnFee);
  }

function setSellTaxes(uint256 newLiquidityTax, uint256 newMarketingTax,
uint256 newBuyBackBurnTax) external onlyOwner() {
    _sellLiquidityFee = newLiquidityTax;
    _sellMarketingFee = newMarketingTax;
    _sellBuyBackBurnFee = newBuyBackBurnTax;

    _totalTaxIfSelling =
_sellLiquidityFee.add(_sellMarketingFee).add(_sellBuyBackBurnFee);
  }
```

Suggestion
It is recommended that no fees in the contract should be more than 25% in the contract.

MANUAL TESTING

Centralization – owner can lock the function.
Severity: High
function: setWalletLimit
Status: Open

Overview:
The owner can set any arbitrary value in the max wallet limit including zero which can simply lock the transfer function as there must be a certain threshold so that the value cannot be less than that particular amount.

```
function setWalletLimit(uint256 newLimit) external onlyOwner
{ ///@audit owner can lock function -- high (any value can be set to wallet limit), missing event
    _walletMax = newLimit;
  }
```

AUDIT SUMMARY

Project name – Monbie

Date: 06 November 2023

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	1	4	0
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/0x4aa5accd7c9468063f1c6109ecc1326be57addac#code

Token Information

Token Address:
0xe07b5E63463090E09c4a46A75dc34b7F71D3d445

Name: Monbie

Symbol: $MNB

Decimals: 9

Network: Binance smart chain

Token Type: ERC20

Owner: 0xf9f4623DfBbC74A784eaa748943539e5363f600A

Deployer:
0xf9f4623DfBbC74A784eaa748943539e5363f600A

Token Supply: 1,000,000

Checksum: 41c9b770f58bb7804e2f92724679f1f4

Testnet version:
The tests were performed using the contract deployed on the Binance smart chain Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0x4aa5accd7c9468063f1c6109ecc1326be57addac#code

TOKEN OVERVIEW

buy fee: 100%

Sell fee: 100%

transfer fee: 0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max wallet: Yes

Max Trx: Yes

Blacklist: No

Other Privileges:

- Initial distribution of the tokens
- Modifying fees
- Enabling trades
-bulk exempts fee
-Modify amm, router,

Liquidity is dding to owner wallet

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.

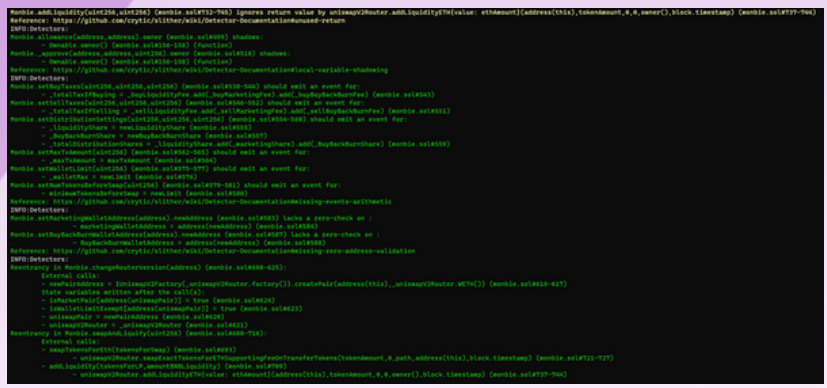
Findings

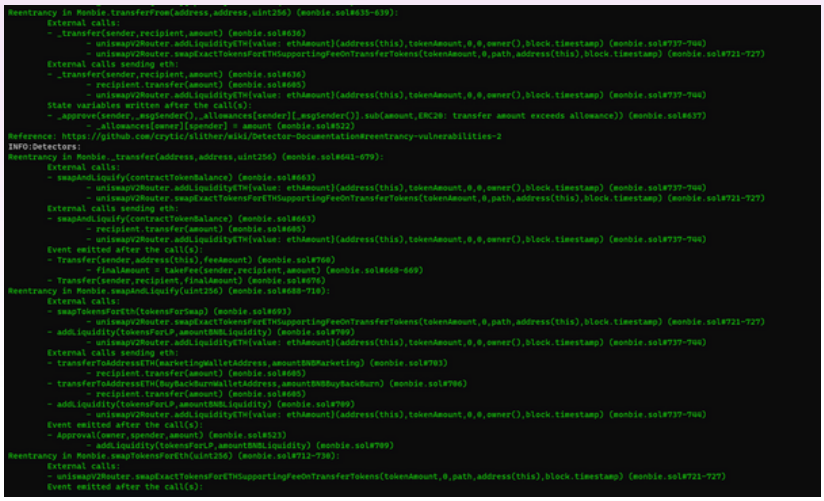
Severity	Found
Critical	0
High-Risk	2
Medium-Risk	1
Low-Risk	4
Gas Optimization / Suggestions	0

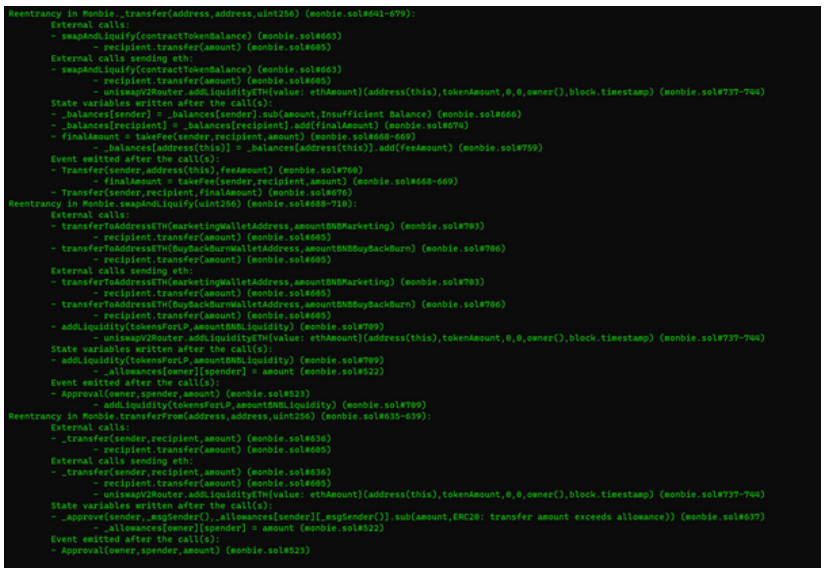
POINTS TO NOTE

- Owner can waive ownership.
- Owner can transfer the ownership.
- Owner can add market pair.
- Owner can exempt wallets from transaction limit.
- Owner can exclude wallet from fees.
- Owner can update tax more than 25%.
- The owner can update any arbitrary value in the distribution share between liquidity, marketing and buyback.
- Owner can update the transaction amount of not less than 100*10**7*10*9 tokens.
- Owner can exclude wallets from wallet limit.
- Owner can enable/disable wallet limit.
- Owner can set any arbitrary value in the minimumtokenbeforeswap amount.
- Owner can change the swap threshold of not more than 1% of total supply.

STATIC ANALYSIS







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

No major issues were found in the output

FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0x479d9201aa6f07dd8905440503cf1e0f267c855cdb17cc247895ead8c2f94221https://testnet.bscscan.com/tx/0x479d9201aa6f07dd8905440503cf1e0f267c855cdb17cc247895ead8c2f94221

2- Set Is Wallet Limit Exempt (passed):

https://testnet.bscscan.com/tx/0x1608ca15b7dcf862e191c70ed47fffb986f93d9baf4e14fef8b0d4c2cbab2956

3- Set Is Tx Limit Exempt (passed):

https://testnet.bscscan.com/tx/0xac9c9913c2edd9a772d29dba692becce81eefd6ddef9e30b69d153cf145f75b5

4- Set Is Excluded From Fee (passed):

https://testnet.bscscan.com/tx/0xa0b4c8e705ada1b7bbd20b64821800e616c1b1fb321c85dbedb029b6223c0aeb

5- Transfer (passed):

https://testnet.bscscan.com/tx/0xcc32696542fcbd5faf610b486b4694949d37fe787d18f2d33e68e565a10d10af

6- Transfer Ownership (passed):

https://testnet.bscscan.com/tx/0x07a1191b3a82d79b1c529e816dabb88a07039ef8bb91842a012f2246097d15d5

MANUAL TESTING

Centralization – Buy and Sell
Severity: High
function: setBuyTaxes/setSellTaxes
Status: Open

Overview:
The owner can set the buy and sell fees to more than 25% which is not recommended.

```
 function setBuyTaxes(uint256 newLiquidityTax, uint256 newMarketingTax,
uint256 newBuyBackBurnTax) external onlyOwner() {
    _buyLiquidityFee = newLiquidityTax;
    _buyMarketingFee = newMarketingTax;
    _buyBuyBackBurnFee = newBuyBackBurnTax;

    _totalTaxIfBuying =
_buyLiquidityFee.add(_buyMarketingFee).add(_buyBuyBackBurnFee);
  }

function setSellTaxes(uint256 newLiquidityTax, uint256 newMarketingTax,
uint256 newBuyBackBurnTax) external onlyOwner() {
    _sellLiquidityFee = newLiquidityTax;
    _sellMarketingFee = newMarketingTax;
    _sellBuyBackBurnFee = newBuyBackBurnTax;

    _totalTaxIfSelling =
_sellLiquidityFee.add(_sellMarketingFee).add(_sellBuyBackBurnFee);
  }
```

Suggestion
It is recommended that no fees in the contract should be more than 25% in the contract.

MANUAL TESTING

Centralization – owner can lock the function.
Severity: High
function: setWalletLimit
Status: Open

Overview:
The owner can set any arbitrary value in the max wallet limit including zero which can simply lock the transfer function as there must be a certain threshold so that the value cannot be less than that particular amount.

```
function setWalletLimit(uint256 newLimit) external onlyOwner
{ ///@audit owner can lock function -- high (any value can be set to wallet limit), missing event
    _walletMax = newLimit;
  }
```

MANUAL TESTING

Centralization – liquidity is adding to the owner's wallet.
Severity: Medium
function: addLiquidity
Status: Open

Overview:
The contract's liquidity is automatically added to the 'owner' address, which is not recommended because, in an extreme scenario, this can be used to drain liquidity from the contract.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
  // approve token transfer to cover all possible scenarios
  _approve(address(this), address(uniswapV2Router), tokenAmount);

  // add the liquidity
  uniswapV2Router.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0, // slippage is unavoidable
    0, // slippage is unavoidable
    owner(),
    block.timestamp
  );
}
```

Suggestion
it is recommended that the liquidity should be added to contract address or dead address

MANUAL TESTING

Centralization – liquidity is adding to the owner's wallet.
Severity: Medium
function: addLiquidity
Status: Open

Overview:
The contract's liquidity is automatically added to the 'owner' address, which is not recommended because, in an extreme scenario, this can be used to drain liquidity from the contract.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
  // approve token transfer to cover all possible scenarios
  _approve(address(this), address(uniswapV2Router), tokenAmount);

  // add the liquidity
  uniswapV2Router.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0, // slippage is unavoidable
    0, // slippage is unavoidable
    owner(),
    block.timestamp
  );
}
```

Suggestion
it is recommended that the liquidity should be added to contract address or dead address

MANUAL TESTING

Severity: Low
subject: floating Pragma Solidity version
Status: Open

Overview:
It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

pragma solidity ^0.8.4;

Suggestion
Adding the latest constant version of solidity is recommended, as this prevents the
unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

MANUAL TESTING

Severity: Low
subject: Local variable shadowing
Status: Open

Overview:
Smart contract shadowing state variables vulnerability happens when the same variable is declared in two places in the contract. This behavior results in important data alteration which could have a dangerous impact on the business logic.

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

```
function _approve(address owner, address spender, uint256 amount) private {
    require(owner != address(0), "ERC20: approve from the zero address");
    require(spender != address(0), "ERC20: approve to the zero address");

    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
  }
```

Suggestion:
Review storage variable layouts for your contract systems carefully and remove any ambiguities. Always check for compiler warnings as they can flag the issue within a single contract.

MANUAL TESTING

Severity: Low
subject: Missing Events
Status: Open

Overview:
They serve as a mechanism for emitting and recording data onto the blockchain, making it
transparent and easily accessible.

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {
    require(maxTxAmount <= (100 * 10**7 * 10**9), "Max wallet");
    _maxTxAmount = maxTxAmount;
  }

  function enableDisableWalletLimit(bool newValue) external onlyOwner {
    checkWalletLimit = newValue;
  }

  function setIsWalletLimitExempt(address holder, bool exempt) external onlyOwner {
    isWalletLimitExempt[holder] = exempt;
  }
```

MANUAL TESTING

Severity: Low
subject: Missing zero address/Dead address Check
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 setNumTokensBeforeSwap(uint256 newLimit) external onlyOwner() {
  minimumTokensBeforeSwap = newLimit;
}

function setMarketingWalletAddress(address newAddress) external
onlyOwner() {
  marketingWalletAddress = payable(newAddress);
} //owner can

function setBuyBackBurnWalletAddress(address newAddress) external
onlyOwner() {
  BuyBackBurnWalletAddress = payable(newAddress);
}
```

Suggestion:
To avoid such problems, you should add the following lines to the "updateOwner()" function:

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