auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Phoenix Chain

DATED : 06 Mar 23'

AUDIT SUMMARY

Project name – Phoenix Chain

Date: 06 March, 2023

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	0	2	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 done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0xdf48722bf079b06a373bcde7ed5054b58d3f5adb

Token Information

Token Name : Phoenix Chain

Token Symbol: PHX

Decimals: 18

Token Supply: 1,000,000,000

Token Address:
0x9776191F4ebBBa7f358C1663bF82C0a0906c77Fa

Checksum:
c5319ccc66c9bb06df9042fbd4b810184f4f5c12

Owner:
0xE275535538dB0C5d5eC244aE736b675e91C080f8

TOKEN OVERVIEW

Fees:

Buy Fees: 1%

Sell Fees: 1%

Transfer Fees: 1%

Fees Privilige: None

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: including and excluding from fees

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

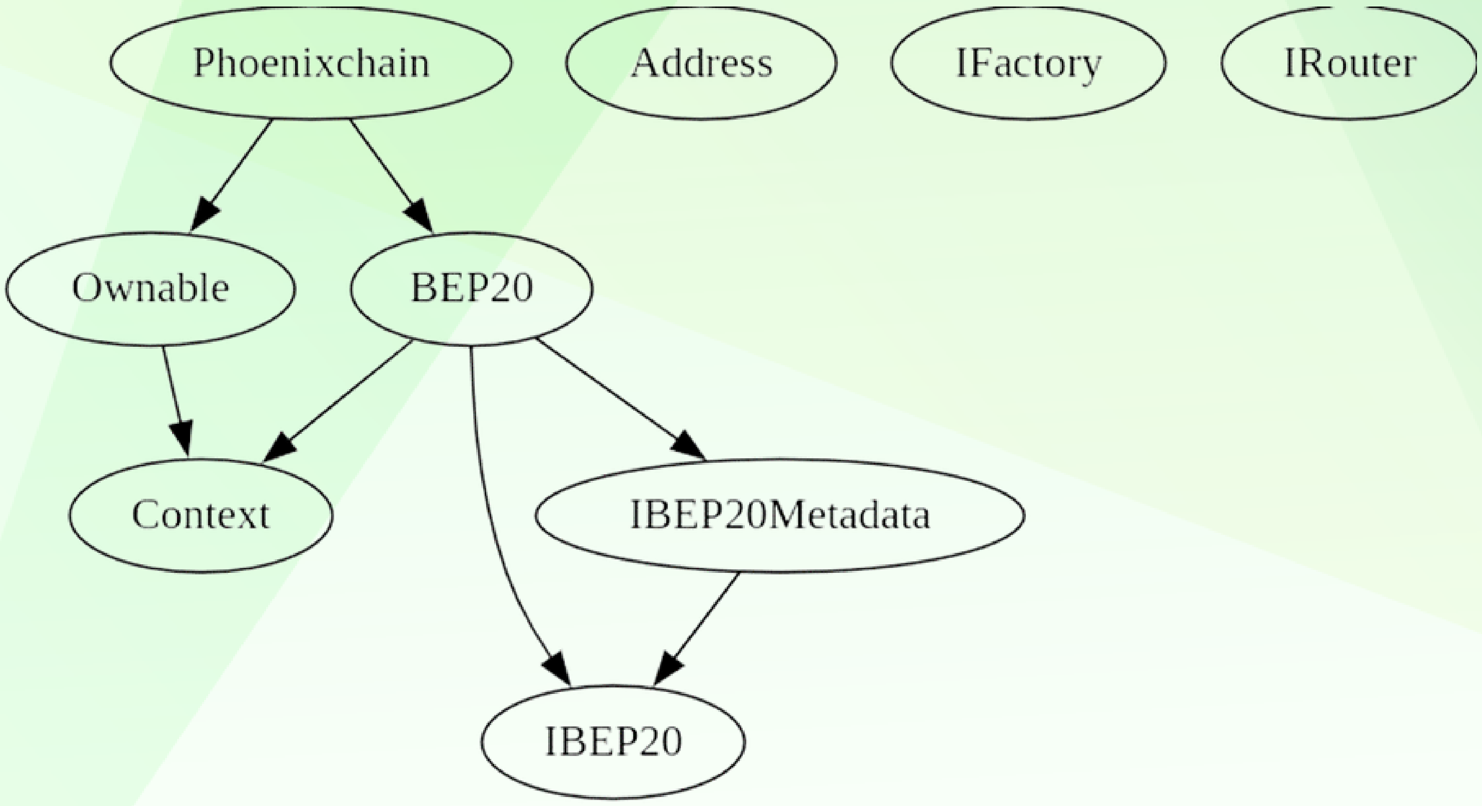
CLASSIFICATION OF RISK

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	0
◆ Low-Risk	2
◆ Gas Optimization / Suggestions	2

INHERITANCE TREE



POINTS TO NOTE

- **Owner is not able to change fees (1% buy and 1% sell)**
- **Owner is not able to set max buy/sell/transfer amount**
- **Owner is not able to blacklist an arbitrary wallet**
- **Owner is not able to disable trades**
- **Owner is not able to mint new tokens**
- **Owner must enable trading in order for everyone to be able to trade (otherwise no one would be able to buy/sell/transfer)**

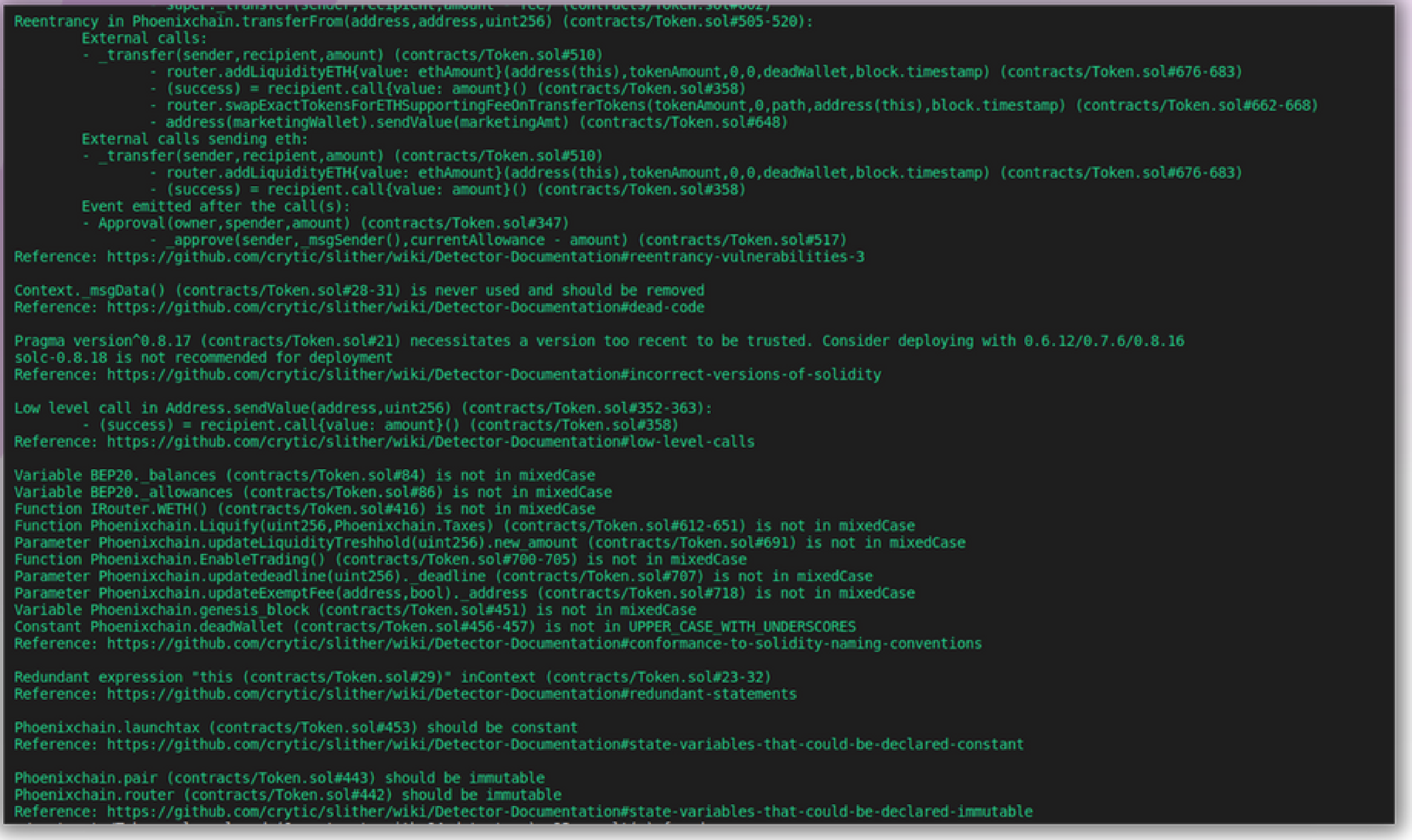
CONTRACT ASSESMENT

```
| Contract |       Type        |       Bases      |                  |                 |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
|    └     | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **IBEP20** | Interface |  |||
| └ | totalSupply | External ❗️ |   |NO❗️ |
| └ | balanceOf | External ❗️ |   |NO❗️ |
| └ | transfer | External ❗️ | 🛑  |NO❗️ |
| └ | allowance | External ❗️ |   |NO❗️ |
| └ | approve | External ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | External ❗️ | 🛑  |NO❗️ |
||||||
| **IBEP20Metadata** | Interface | IBEP20 |||
| └ | name | External ❗️ |   |NO❗️ |
| └ | symbol | External ❗️ |   |NO❗️ |
| └ | decimals | External ❗️ |   |NO❗️ |
||||||
| **BEP20** | Implementation | Context, IBEP20, IBEP20Metadata |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | name | Public ❗️ |   |NO❗️ |
| └ | symbol | Public ❗️ |   |NO❗️ |
| └ | decimals | Public ❗️ |   |NO❗️ |
| └ | totalSupply | Public ❗️ |   |NO❗️ |
| └ | balanceOf | Public ❗️ |   |NO❗️ |
| └ | transfer | Public ❗️ | 🛑  |NO❗️ |
| └ | allowance | Public ❗️ |   |NO❗️ |
| └ | approve | Public ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | Public ❗️ | 🛑  |NO❗️ |
| └ | increaseAllowance | Public ❗️ | 🛑  |NO❗️ |
| └ | decreaseAllowance | Public ❗️ | 🛑  |NO❗️ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | _tokengeneration | Internal 🔒 | 🛑  | |
| └ | _approve | Internal 🔒 | 🛑  | |
||||||
| **Address** | Library |  |||
| └ | sendValue | Internal 🔒 | 🛑  | |
||||||
| **Ownable** | Implementation | Context |||
```

CONTRACT ASSESMENT

```
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | owner | Public ❗ |   |NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | _setOwner | Private 🔐 | 🛑  | |
||||||
| **IFactory** | Interface |  |||
| └ | createPair | External ❗ | 🛑  |NO❗ |
||||||
| **IRouter** | Interface |  |||
| └ | factory | External ❗ |   |NO❗ |
| └ | WETH | External ❗ |   |NO❗ |
| └ | addLiquidityETH | External ❗ |  💵 |NO❗ |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗ | 🛑  |NO❗ |
||||||
| **Phoenixchain** | Implementation | BEP20, Ownable |||
| └ | <Constructor> | Public ❗ | 🛑  | BEP20 |
| └ | approve | Public ❗ | 🛑  |NO❗ |
| └ | transferFrom | Public ❗ | 🛑  |NO❗ |
| └ | increaseAllowance | Public ❗ | 🛑  |NO❗ |
| └ | decreaseAllowance | Public ❗ | 🛑  |NO❗ |
| └ | transfer | Public ❗ | 🛑  |NO❗ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | Liquify | Private 🔐 | 🛑  | lockTheSwap |
| └ | swapTokensForETH | Private 🔐 | 🛑  | |
| └ | addLiquidity | Private 🔐 | 🛑  | |
| └ | updateLiquidityProvide | External ❗ | 🛑  | onlyOwner |
| └ | updateLiquidityTreshhold | External ❗ | 🛑  | onlyOwner |
| └ | EnableTrading | External ❗ | 🛑  | onlyOwner |
| └ | updatedeadline | External ❗ | 🛑  | onlyOwner |
| └ | updateMarketingWallet | External ❗ | 🛑  | onlyOwner |
| └ | updateExemptFee | External ❗ | 🛑  | onlyOwner |
| └ | bulkExemptFee | External ❗ | 🛑  | onlyOwner |
| └ | rescueBNB | External ❗ | 🛑  | onlyOwner |
| └ | rescueBSC20 | External ❗ | 🛑  | onlyOwner |
| └ | <Receive Ether> | External ❗ |  💵 |NO❗ |
|  Symbol  |  Meaning  |
|:--------:|-----------|
|    🛑    | Function can modify state |
|    💵    | Function is payable |
```

STATIC ANALYSIS



```
Reentrancy in Phoenixchain.transferFrom(address,address,uint256) (contracts/Token.sol#505-520):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#510)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/Token.sol#676-683)
                - (success) = recipient.call{value: amount}() (contracts/Token.sol#358)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#662-668)
                - address(marketingWallet).sendValue(marketingAmt) (contracts/Token.sol#648)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#510)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/Token.sol#676-683)
                - (success) = recipient.call{value: amount}() (contracts/Token.sol#358)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/Token.sol#347)
                - _approve(sender,_msgSender(),currentAllowance - amount) (contracts/Token.sol#517)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Context._msgData() (contracts/Token.sol#28-31) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/Token.sol#21) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.18 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#352-363):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#358)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BEP20._balances (contracts/Token.sol#84) is not in mixedCase
Variable BEP20._allowances (contracts/Token.sol#86) is not in mixedCase
Function IRouter.WETH() (contracts/Token.sol#416) is not in mixedCase
Function Phoenixchain.Liquify(uint256,Phoenixchain.Taxes) (contracts/Token.sol#612-651) is not in mixedCase
Parameter Phoenixchain.updateLiquidityTreshhold(uint256).new_amount (contracts/Token.sol#691) is not in mixedCase
Function Phoenixchain.EnableTrading() (contracts/Token.sol#700-705) is not in mixedCase
Parameter Phoenixchain.updatedeadline(uint256)._deadline (contracts/Token.sol#707) is not in mixedCase
Parameter Phoenixchain.updateExemptFee(address,bool)._address (contracts/Token.sol#718) is not in mixedCase
Variable Phoenixchain.genesis_block (contracts/Token.sol#451) is not in mixedCase
Constant Phoenixchain.deadWallet (contracts/Token.sol#456-457) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#29)" inContext (contracts/Token.sol#23-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Phoenixchain.launchtax (contracts/Token.sol#453) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

Phoenixchain.pair (contracts/Token.sol#443) should be immutable
Phoenixchain.router (contracts/Token.sol#442) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

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

No major issues were found in the output

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

All the functionalities have been tested, no issues were found

1- Adding liquidity **(passed):**
https://testnet.bscscan.com/tx/0xae1d5e33c5828ee1bcb9f1151ea0bee869f741809b40fc0c36c47cfede74a42b

2- Buying when excluded (0% tax) **(passed):**
https://testnet.bscscan.com/tx/0x378b8cbe315a2031411c75b7dc9c761a31d4a806c1a4e38afc620a694822ebc7

3- Selling when excluded (0% tax) **(passed):**
https://testnet.bscscan.com/tx/0xb8bd1aeb8861ee2bca427c5526a9a5fb46c45ff82c6d6587819ab97415830c86

4- Transferring when excluded (0% tax) **(passed):**
https://testnet.bscscan.com/tx/0xb4c3a0311c130ffc9d5d9ab2a304c0daf2ef58476f3294be968ed7db771cf4ff

5- Buying when not excluded (1% tax) **(passed):**
https://testnet.bscscan.com/tx/0x878ec41f08f8e851da634b3d2194129f9cd4e904a10dfd7d96c9c9e248099eb1

6- Selling when not excluded (1% tax) **(passed):**
https://testnet.bscscan.com/tx/0x56265db1b049555bc5692b687913b3c784e20c5b63f3c6d61f2d07a04d79e16f

FUNCTIONAL TESTING

7- Transferring when not excluded (1% tax) (passed):
https://testnet.bscscan.com/tx/0xa23a98adb90daf79edcf6f902c7a09bd93015c94ce252b54c26a877d974115de

8- Internal swap (passed):
prize pool wallet received ETH
https://testnet.bscscan.com/address/0x4d150dafe944ed0c917a739dcf5a0432f8791cbf#internaltx

MANUAL TESTING

Low Risk Issues

Issue: no way to enable auto-liquidity

Type : Logical

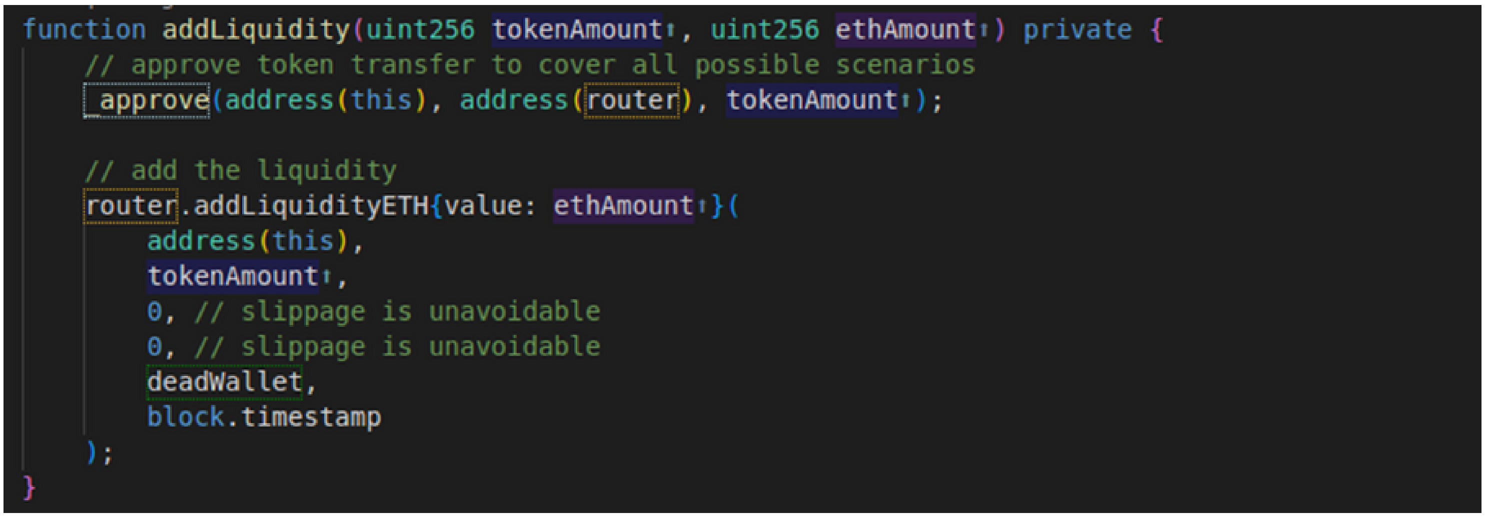
Function:---

Line: ---

Severity: Low

Overview:

Although the contract has implemented the necessary functionality for auto-liquidity, the feature is deemed redundant since the liquidity tax is set at 0 and is non-upgradable. As a result, the functionality serves no practical purpose.



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

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

Recommendation:

- delete this auto-liquidity functions from the contract to reduce overall gas usage
- add the necessary function to be able to update liquidity tax

MANUAL TESTING

Low Risk Issues

Issue: marketing wallet can reject sells if set to a contract

Type : Logical

Function:updateMarketingWallet

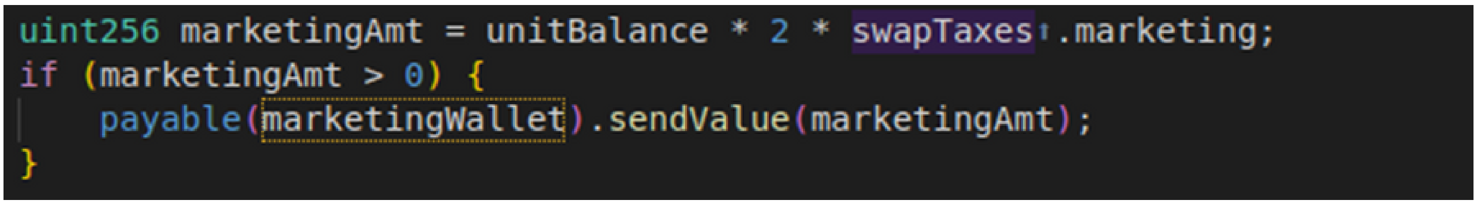
Line: 672-679

Severity: Low

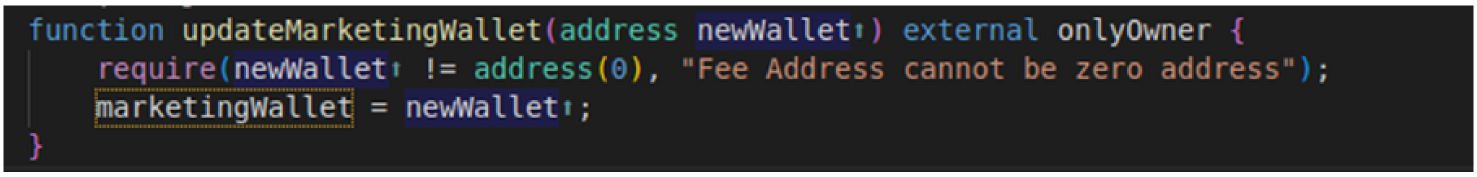
Overview:

there is an *updateMarketingWallet* function that is used to update the marketing wallet address. However, if the address provided for the marketing wallet is a contract that rejects receiving ether, it can lead to a potential issue. This is because when an internal swap occurs, only on sells, some ETH will be sent to the marketing wallet, and since it is a contract that rejects receiving ether, this can cause the sells to fail and be reverted.

Based on the likelihood of occurrence, we categorize this issue as "low" severity. While it is important to review the marketing wallet contract and ensure that it can receive ETH to prevent any issues during internal swaps, this issue is rare and may not occur frequently enough to pose a significant risk. Nonetheless, it is still recommended to take the necessary precautions to mitigate any potential risk



```
uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
if (marketingAmt > 0) {
    payable(marketingWallet).sendValue(marketingAmt);
}
```



```
function updateMarketingWallet(address newWallet) external onlyOwner {
    require(newWallet != address(0), "Fee Address cannot be zero address");
    marketingWallet = newWallet;
}
```

MANUAL TESTING

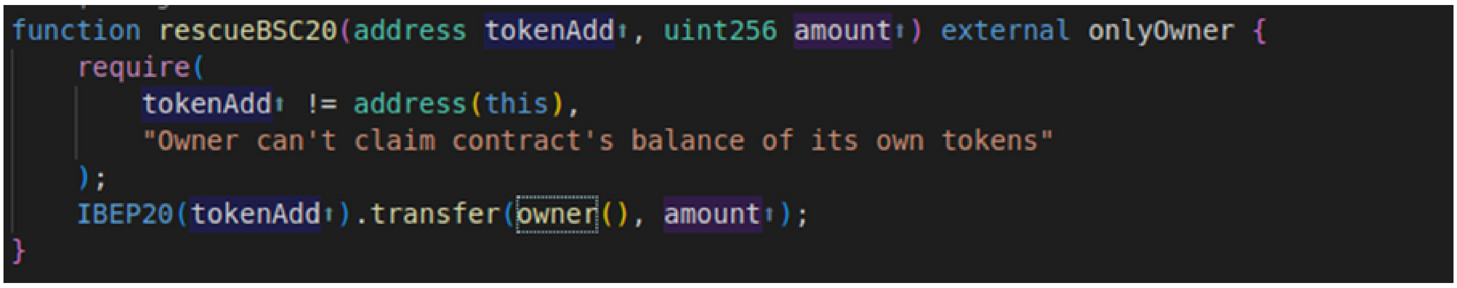
Suggestions & Recommendations:

S-1 : allowing owner to withdraw native tokens from the contract

Preventing the owner from withdrawing native tokens from the contract can provide a level of transparency and assurance to investors that their fees are being used in a way that benefits all investors. This approach can also help to prevent potential misuse or mismanagement of the collected fees.

On the other hand, allowing the owner to withdraw the native tokens can provide more flexibility and control over how the collected fees are used. This can be particularly useful in situations where the collected fees need to be used for purposes other than marketing or providing liquidity.

Hence, It may be beneficial to consider implementing additional safeguards or controls to prevent potential misuse of the collected fees, regardless of whether the owner is allowed to withdraw them or not.



```
function rescueBSC20(address tokenAdd, uint256 amount) external onlyOwner {
    require(
        tokenAdd != address(this),
        "Owner can't claim contract's balance of its own tokens"
    );
    IBEP20(tokenAdd).transfer(owner(), amount);
}
```

MANUAL TESTING

Gas Optimizations

The suggestions described here are meant to improve the overall gas usage of the contract for each buy, sell, or transfer.
- Declare this variables as immutable:

- **router [L-457]**
- **pair [L-458]**

- [L-659 / L-673] : approve router once at constructor with uint256 max to avoid further gas usage of this call
- [L-657] : save router.WETH in a constant variable to avoid further gas usage of accessing this variable from router variable

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