RAPID LABS

SMART CONTRACT SECURITY AUDIT

Final report **Plan: Simple**

Scorpion

May 2024

RL@RAPIDLABS.FINANCE

RAPIDLABS.FINANCE

TABLE OF CONTENTS

1. Introduction 3

2. Analyzed Contracts 3

3. Audit Process 3

3.1 Auto-analysis 3

3.2 Expert audit 3

4. Known issues checked 4

5. Issue Classification 6

6. Issues 6

6.1 High risk issues 6

6.2 Medium risk issues 6

6.3 Low risk issues 7

7. Conclusion 8

8. Disclaimer 9

9. Static analysis 10

10. Accordance to ERC20 standard 33

INTRODUCTION

The report has been prepared for Scorpion.

Name	Scorpion
Audit date	2024-05-01 - 2024-05-01
Language	Solidity
Platform	Binance Smart Chain

ANALYZED CONTRACTS

Name	Address
SCORP	0xa910a46e2f2002fa9b5aa85f35b9440f6dac4b10

AUDIT PROCESS

Our audit structure consists of two stages:

Auto-analysis

- Our automated tools allow us to scan smart contract code and find potential issues
- We hand pick and verify all the issues found by the tools

Expert audit

- Manual analysis of potential issues and vulnerabilities
- Contract code is reviewed thoroughly

KNOWN ISSUES CHECKED

Title	Result
Unencrypted Private Data On-Chain	✓ passed
Code With No Effects	✓ passed
Message call with hardcoded gas amount	✓ passed
Typographical Error	✓ passed
DoS With Block Gas Limit	✓ passed
Presence of unused variables	✓ passed
Incorrect Inheritance Order	✓ passed
Requirement Violation	✓ passed
Weak Sources of Randomness from Chain Attributes	✓ passed
Shadowing State Variables	✓ passed
Incorrect Constructor Name	✓ passed
Block values as a proxy for time	✓ passed
Authorization through tx.origin	✓ passed

DoS with Failed Call	✓ passed
Delegatecall to Untrusted Callee	✓ passed
Use of Deprecated Solidity Functions	✓ passed
Assert Violation	✓ passed
State Variable Default Visibility	✓ passed
Reentrancy	✓ passed
Unprotected SELFDESTRUCT Instruction	✓ passed
Unprotected Ether Withdrawal	✓ passed
Unchecked Call Return Value	✓ passed
Floating Pragma	✓ passed
Outdated Compiler Version	✓ passed
Integer Overflow and Underflow	✓ passed
Function Default Visibility	✓ passed

ISSUE CLASSIFICATION

High risk	Issues leading to assets theft, locking or any other loss of assets or leading to contract malfunctioning.
Medium risk	Issues that can trigger a contract failure of malfunctioning.
Low risk	Issues that do now affect contract functionality. For example, unoptimised gas usage, outdated or unused code, code style violations, etc.

ISSUES

High risk issues

No issues were found

Medium risk issues

1. No contraints on number of excluded address (SCORP)

The contract iterates over all the excluded addresses on every transfer.

```
function _getCurrentSupply() private view returns (uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (
            _rOwned[_excluded[i]] > rSupply ||
            _tOwned[_excluded[i]] > tSupply
        ) return (_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
```

```
        }
```

If the owner adds big number of excluded addresses, this may lead to extreme gas usage on trasfers.

2. ERC20 standard violation (SCORP)

The contract does not allow to transfer zero amount of tokens. However, these transfers according to the ERC20 standard should be treated as normal transfers.

Low risk issues

1. Possible abuse with exclude reward (SCORP)

The owner of the token contract can redistribute part of the tokens from users to a specific account. For this owner can exclude an account from the reward and include it back later. This will redistribute part of the tokens from holders in profit of the included account.

CONCLUSION

Scorpion SCORP contract was audited. 2 medium, 1 low risk issues were found.

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without RapidLabs prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts RapidLabs to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733):
        External calls:
        - swapBack(contractTokenBalance) (contracts/contract.sol#694)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
                - pancakeV2Router.swapExactTokensForETHSupportingFeeOnTransferToken
s(tokenAmount,0,path,address(this),block.timestamp) (contracts/
contract.sol#764-770)
                - (success,None) = address(marketingWallet).call{value:
bnbForMarketing}() (contracts/contract.sol#756)
        External calls sending eth:
        - swapBack(contractTokenBalance) (contracts/contract.sol#694)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
                - (success,None) = address(marketingWallet).call{value:
bnbForMarketing}() (contracts/contract.sol#756)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - _rOwned[marketingWallet] =
_rOwned[marketingWallet].add(rMarketing) (contracts/contract.sol#610-612)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity)
(contracts/contract.sol#598-600)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#822)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#844)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#426)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#868)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#823)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
```

```
(contracts/contract.sol#869)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#846)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#428)
        SCORP._rOwned (contracts/contract.sol#158) can be used in cross function
reentrancies:
        - SCORP._getCurrentSupply() (contracts/contract.sol#576-589)
        - SCORP._takeLiquidity(uint256) (contracts/contract.sol#591-605)
        - SCORP._takeMarketing(uint256) (contracts/contract.sol#607-617)
        - SCORP._transferBothExcluded(address,address,uint256) (contracts/
contract.sol#411-433)
        - SCORP._transferFromExcluded(address,address,uint256) (contracts/
contract.sol#853-874)
        - SCORP._transferStandard(address,address,uint256) (contracts/
contract.sol#808-828)
        - SCORP._transferToExcluded(address,address,uint256) (contracts/
contract.sol#830-851)
        - SCORP.balanceOf(address) (contracts/contract.sol#263-266)
        - SCORP.constructor() (contracts/contract.sol#214-245)
        - SCORP.deliver(uint256) (contracts/contract.sol#349-359)
        - SCORP.excludeFromReward(address) (contracts/contract.sol#389-396)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - _rOwned[marketingWallet] =
_rOwned[marketingWallet].add(rMarketing) (contracts/contract.sol#610-612)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity)
(contracts/contract.sol#598-600)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#822)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#844)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#426)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#868)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#823)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#869)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#846)
```

```
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#428)
        SCORP._rOwned (contracts/contract.sol#158) can be used in cross function
reentrancies:
        - SCORP._getCurrentSupply() (contracts/contract.sol#576-589)
        - SCORP._takeLiquidity(uint256) (contracts/contract.sol#591-605)
        - SCORP._takeMarketing(uint256) (contracts/contract.sol#607-617)
        - SCORP._transferBothExcluded(address,address,uint256) (contracts/
contract.sol#411-433)
        - SCORP._transferFromExcluded(address,address,uint256) (contracts/
contract.sol#853-874)
        - SCORP._transferStandard(address,address,uint256) (contracts/
contract.sol#808-828)
        - SCORP._transferToExcluded(address,address,uint256) (contracts/
contract.sol#830-851)
        - SCORP.balanceOf(address) (contracts/contract.sol#263-266)
        - SCORP.constructor() (contracts/contract.sol#214-245)
        - SCORP.deliver(uint256) (contracts/contract.sol#349-359)
        - SCORP.excludeFromReward(address) (contracts/contract.sol#389-396)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - _rOwned[marketingWallet] =
_rOwned[marketingWallet].add(rMarketing) (contracts/contract.sol#610-612)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity)
(contracts/contract.sol#598-600)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#822)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#844)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#426)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/
contract.sol#868)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#823)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#869)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#846)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(contracts/contract.sol#428)
        SCORP._rOwned (contracts/contract.sol#158) can be used in cross function
reentrancies:
```

```
        - SCORP._getCurrentSupply() (contracts/contract.sol#576-589)
        - SCORP._takeLiquidity(uint256) (contracts/contract.sol#591-605)
        - SCORP._takeMarketing(uint256) (contracts/contract.sol#607-617)
        - SCORP._transferBothExcluded(address,address,uint256) (contracts/
contract.sol#411-433)
        - SCORP._transferFromExcluded(address,address,uint256) (contracts/
contract.sol#853-874)
        - SCORP._transferStandard(address,address,uint256) (contracts/
contract.sol#808-828)
        - SCORP._transferToExcluded(address,address,uint256) (contracts/
contract.sol#830-851)
        - SCORP.balanceOf(address) (contracts/contract.sol#263-266)
        - SCORP.constructor() (contracts/contract.sol#214-245)
        - SCORP.deliver(uint256) (contracts/contract.sol#349-359)
        - SCORP.excludeFromReward(address) (contracts/contract.sol#389-396)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - _rTotal = _rTotal.sub(rFee) (contracts/contract.sol#486)
        SCORP._rTotal (contracts/contract.sol#170) can be used in cross function
reentrancies:
        - SCORP._getCurrentSupply() (contracts/contract.sol#576-589)
        - SCORP._reflectFee(uint256,uint256) (contracts/contract.sol#485-488)
        - SCORP.constructor() (contracts/contract.sol#214-245)
        - SCORP.deliver(uint256) (contracts/contract.sol#349-359)
        - SCORP.tokenFromReflection(uint256) (contracts/contract.sol#376-387)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - _rTotal = _rTotal.sub(rFee) (contracts/contract.sol#486)
        SCORP._rTotal (contracts/contract.sol#170) can be used in cross function
reentrancies:
        - SCORP._getCurrentSupply() (contracts/contract.sol#576-589)
        - SCORP._reflectFee(uint256,uint256) (contracts/contract.sol#485-488)
        - SCORP.constructor() (contracts/contract.sol#214-245)
        - SCORP.deliver(uint256) (contracts/contract.sol#349-359)
        - SCORP.tokenFromReflection(uint256) (contracts/contract.sol#376-387)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - _rTotal = _rTotal.sub(rFee) (contracts/contract.sol#486)
        SCORP._rTotal (contracts/contract.sol#170) can be used in cross function
reentrancies:
        - SCORP._getCurrentSupply() (contracts/contract.sol#576-589)
        - SCORP._reflectFee(uint256,uint256) (contracts/contract.sol#485-488)
        - SCORP.constructor() (contracts/contract.sol#214-245)
        - SCORP.deliver(uint256) (contracts/contract.sol#349-359)
```

```
        - SCORP.tokenFromReflection(uint256) (contracts/contract.sol#376-387)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - _tOwned[marketingWallet] =
_tOwned[marketingWallet].add(tMarketing) (contracts/contract.sol#614-616)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity)
(contracts/contract.sol#602-604)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (contracts/
contract.sol#425)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (contracts/
contract.sol#867)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount)
(contracts/contract.sol#845)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount)
(contracts/contract.sol#427)
        SCORP._tOwned (contracts/contract.sol#159) can be used in cross function
reentrancies:
        - SCORP._getCurrentSupply() (contracts/contract.sol#576-589)
        - SCORP._takeLiquidity(uint256) (contracts/contract.sol#591-605)
        - SCORP._takeMarketing(uint256) (contracts/contract.sol#607-617)
        - SCORP._transferBothExcluded(address,address,uint256) (contracts/
contract.sol#411-433)
        - SCORP._transferFromExcluded(address,address,uint256) (contracts/
contract.sol#853-874)
        - SCORP._transferToExcluded(address,address,uint256) (contracts/
contract.sol#830-851)
        - SCORP.balanceOf(address) (contracts/contract.sol#263-266)
        - SCORP.excludeFromReward(address) (contracts/contract.sol#389-396)
        - SCORP.includeInReward(address) (contracts/contract.sol#398-409)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - _tOwned[marketingWallet] =
_tOwned[marketingWallet].add(tMarketing) (contracts/contract.sol#614-616)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity)
(contracts/contract.sol#602-604)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (contracts/
contract.sol#425)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (contracts/
contract.sol#867)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount)
(contracts/contract.sol#845)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount)
(contracts/contract.sol#427)
```

```
        SCORP._tOwned (contracts/contract.sol#159) can be used in cross function
reentrancies:
        - SCORP._getCurrentSupply() (contracts/contract.sol#576-589)
        - SCORP._takeLiquidity(uint256) (contracts/contract.sol#591-605)
        - SCORP._takeMarketing(uint256) (contracts/contract.sol#607-617)
        - SCORP._transferBothExcluded(address,address,uint256) (contracts/
contract.sol#411-433)
        - SCORP._transferFromExcluded(address,address,uint256) (contracts/
contract.sol#853-874)
        - SCORP._transferToExcluded(address,address,uint256) (contracts/
contract.sol#830-851)
        - SCORP.balanceOf(address) (contracts/contract.sol#263-266)
        - SCORP.excludeFromReward(address) (contracts/contract.sol#389-396)
        - SCORP.includeInReward(address) (contracts/contract.sol#398-409)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - _tOwned[marketingWallet] =
_tOwned[marketingWallet].add(tMarketing) (contracts/contract.sol#614-616)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity)
(contracts/contract.sol#602-604)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (contracts/
contract.sol#425)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (contracts/
contract.sol#867)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount)
(contracts/contract.sol#845)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount)
(contracts/contract.sol#427)
        SCORP._tOwned (contracts/contract.sol#159) can be used in cross function
reentrancies:
        - SCORP._getCurrentSupply() (contracts/contract.sol#576-589)
        - SCORP._takeLiquidity(uint256) (contracts/contract.sol#591-605)
        - SCORP._takeMarketing(uint256) (contracts/contract.sol#607-617)
        - SCORP._transferBothExcluded(address,address,uint256) (contracts/
contract.sol#411-433)
        - SCORP._transferFromExcluded(address,address,uint256) (contracts/
contract.sol#853-874)
        - SCORP._transferToExcluded(address,address,uint256) (contracts/
contract.sol#830-851)
        - SCORP.balanceOf(address) (contracts/contract.sol#263-266)
        - SCORP.excludeFromReward(address) (contracts/contract.sol#389-396)
        - SCORP.includeInReward(address) (contracts/contract.sol#398-409)
        - liquidityFee = _buyLiquidityFee (contracts/contract.sol#704)
```

```
        SCORP.liquidityFee (contracts/contract.sol#187) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
        - SCORP.calculateLiquidity(uint256) (contracts/contract.sol#631-637)
        - SCORP.liquidityFee (contracts/contract.sol#187)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - liquidityFee = _sellLiquidityFee (contracts/contract.sol#710)
        SCORP.liquidityFee (contracts/contract.sol#187) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
        - SCORP.calculateLiquidity(uint256) (contracts/contract.sol#631-637)
        - SCORP.liquidityFee (contracts/contract.sol#187)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - liquidityFee = 0 (contracts/contract.sol#716)
        SCORP.liquidityFee (contracts/contract.sol#187) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
        - SCORP.calculateLiquidity(uint256) (contracts/contract.sol#631-637)
        - SCORP.liquidityFee (contracts/contract.sol#187)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - liquidityFee = _previousLiquidityFee (contracts/
contract.sol#654)
                - liquidityFee = 0 (contracts/contract.sol#647)
        SCORP.liquidityFee (contracts/contract.sol#187) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
```

```
        - SCORP.calculateLiquidity(uint256) (contracts/contract.sol#631-637)
        - SCORP.liquidityFee (contracts/contract.sol#187)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - liquidityFee = _previousLiquidityFee (contracts/
contract.sol#654)
                - liquidityFee = 0 (contracts/contract.sol#647)
        SCORP.liquidityFee (contracts/contract.sol#187) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
        - SCORP.calculateLiquidity(uint256) (contracts/contract.sol#631-637)
        - SCORP.liquidityFee (contracts/contract.sol#187)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - liquidityFee = _previousLiquidityFee (contracts/
contract.sol#654)
                - liquidityFee = 0 (contracts/contract.sol#647)
        SCORP.liquidityFee (contracts/contract.sol#187) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
        - SCORP.calculateLiquidity(uint256) (contracts/contract.sol#631-637)
        - SCORP.liquidityFee (contracts/contract.sol#187)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - marketingFee = _buyMarketingFee (contracts/contract.sol#705)
        SCORP.marketingFee (contracts/contract.sol#188) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
```

```
        - SCORP.calculateMarketingFee(uint256) (contracts/contract.sol#623-629)
        - SCORP.marketingFee (contracts/contract.sol#188)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - marketingFee = _sellMarketingFee (contracts/contract.sol#711)
        SCORP.marketingFee (contracts/contract.sol#188) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
        - SCORP.calculateMarketingFee(uint256) (contracts/contract.sol#623-629)
        - SCORP.marketingFee (contracts/contract.sol#188)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - marketingFee = 0 (contracts/contract.sol#717)
        SCORP.marketingFee (contracts/contract.sol#188) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
        - SCORP.calculateMarketingFee(uint256) (contracts/contract.sol#623-629)
        - SCORP.marketingFee (contracts/contract.sol#188)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - marketingFee = _previousMarketingFee (contracts/
contract.sol#653)
                - marketingFee = 0 (contracts/contract.sol#646)
        SCORP.marketingFee (contracts/contract.sol#188) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
        - SCORP.calculateMarketingFee(uint256) (contracts/contract.sol#623-629)
        - SCORP.marketingFee (contracts/contract.sol#188)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
```

```
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - marketingFee = _previousMarketingFee (contracts/
contract.sol#653)
                - marketingFee = 0 (contracts/contract.sol#646)
        SCORP.marketingFee (contracts/contract.sol#188) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
        - SCORP.calculateMarketingFee(uint256) (contracts/contract.sol#623-629)
        - SCORP.marketingFee (contracts/contract.sol#188)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - marketingFee = _previousMarketingFee (contracts/
contract.sol#653)
                - marketingFee = 0 (contracts/contract.sol#646)
        SCORP.marketingFee (contracts/contract.sol#188) can be used in cross
function reentrancies:
        - SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733)
        - SCORP.calculateMarketingFee(uint256) (contracts/contract.sol#623-629)
        - SCORP.marketingFee (contracts/contract.sol#188)
        - SCORP.removeAllFee() (contracts/contract.sol#639-649)
        - SCORP.restoreAllFee() (contracts/contract.sol#651-656)
        - SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462)
        - SCORP.swapBack(uint256) (contracts/contract.sol#734-757)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancy-vulnerabilities
INFO:Detectors:
SCORP.swapBack(uint256) (contracts/contract.sol#734-757) performs a multiplication
on the result of a division:
        - tokensForLiquidity = contractBalance.mul(liquidityFee).div(100)
(contracts/contract.sol#736)
```

```
        - liquidityTokens = (contractBalance * tokensForLiquidity) /
(totalTokensToSwap / 2) (contracts/contract.sol#743)
SCORP.swapBack(uint256) (contracts/contract.sol#734-757) performs a multiplication
on the result of a division:
        - marketingTokens = contractBalance.mul(marketingFee).div(100) (contracts/
contract.sol#737)
        - bnbForMarketing = (bnbBalance * marketingTokens) / (totalTokensToSwap -
(tokensForLiquidity / 2)) (contracts/contract.sol#750)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-
before-multiply
INFO:Detectors:
SCORP.addLiquidity(uint256,uint256) (contracts/contract.sol#773-783) ignores
return value by pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-
return
INFO:Detectors:
SCORP.allowance(address,address).owner (contracts/contract.sol#277) shadows:
        - Ownable.owner() (contracts/contract.sol#96-98) (function)
SCORP._approve(address,address,uint256).owner (contracts/contract.sol#663)
shadows:
        - Ownable.owner() (contracts/contract.sol#96-98) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-
variable-shadowing
INFO:Detectors:
SCORP.setSellFees(uint256,uint256,uint256,uint256) (contracts/
contract.sol#446-462) should emit an event for:
        - _sellRefFee = tFee (contracts/contract.sol#454)
        - refFee = _sellRefFee (contracts/contract.sol#455)
        - _sellLiquidityFee = lFee (contracts/contract.sol#456)
        - liquidityFee = _sellLiquidityFee (contracts/contract.sol#457)
        - _sellMarketingFee = mFee (contracts/contract.sol#458)
        - marketingFee = _sellMarketingFee (contracts/contract.sol#459)
        - _sellBurnFee = bFee (contracts/contract.sol#460)
        - burnFee = _sellBurnFee (contracts/contract.sol#461)
SCORP.setNumTokensSellToAddToLiquidity(uint256) (contracts/contract.sol#469-475)
should emit an event for:
        - numTokensSellToAddToLiquidity = amount * 10 ** _decimals (contracts/
contract.sol#474)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-
events-arithmetic
```

```
INFO:Detectors:
Reentrancy in SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733):
        External calls:
        - swapBack(contractTokenBalance) (contracts/contract.sol#694)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
                - pancakeV2Router.swapExactTokensForETHSupportingFeeOnTransferToken
s(tokenAmount,0,path,address(this),block.timestamp) (contracts/
contract.sol#764-770)
                - (success,None) = address(marketingWallet).call{value:
bnbForMarketing}() (contracts/contract.sol#756)
        External calls sending eth:
        - swapBack(contractTokenBalance) (contracts/contract.sol#694)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
                - (success,None) = address(marketingWallet).call{value:
bnbForMarketing}() (contracts/contract.sol#756)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - _previousBurnFee = burnFee (contracts/contract.sol#643)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - _previousBurnFee = burnFee (contracts/contract.sol#643)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - _previousBurnFee = burnFee (contracts/contract.sol#643)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - _previousLiquidityFee = liquidityFee (contracts/
contract.sol#642)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - _previousLiquidityFee = liquidityFee (contracts/
contract.sol#642)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - _previousLiquidityFee = liquidityFee (contracts/
contract.sol#642)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - _previousMarketingFee = marketingFee (contracts/
contract.sol#641)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
```

```
                - _previousMarketingFee = marketingFee (contracts/
contract.sol#641)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - _previousMarketingFee = marketingFee (contracts/
contract.sol#641)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - _previousTaxFee = refFee (contracts/contract.sol#640)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - _previousTaxFee = refFee (contracts/contract.sol#640)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - _previousTaxFee = refFee (contracts/contract.sol#640)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - _tFeeTotal = _tFeeTotal.add(tFee) (contracts/contract.sol#487)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - _tFeeTotal = _tFeeTotal.add(tFee) (contracts/contract.sol#487)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - _tFeeTotal = _tFeeTotal.add(tFee) (contracts/contract.sol#487)
        - burnFee = _buyBurnFee (contracts/contract.sol#706)
        - burnFee = _sellBurnFee (contracts/contract.sol#712)
        - burnFee = 0 (contracts/contract.sol#718)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - burnFee = _previousBurnFee (contracts/contract.sol#655)
                - burnFee = 0 (contracts/contract.sol#648)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - burnFee = _previousBurnFee (contracts/contract.sol#655)
                - burnFee = 0 (contracts/contract.sol#648)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - burnFee = _previousBurnFee (contracts/contract.sol#655)
                - burnFee = 0 (contracts/contract.sol#648)
        - refFee = _buyRefFee (contracts/contract.sol#703)
        - refFee = _sellRefFee (contracts/contract.sol#709)
        - refFee = 0 (contracts/contract.sol#715)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#723)
                - refFee = _previousTaxFee (contracts/contract.sol#652)
                - refFee = 0 (contracts/contract.sol#645)
        - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
                - refFee = _previousTaxFee (contracts/contract.sol#652)
                - refFee = 0 (contracts/contract.sol#645)
```

```
        - _tokenTransfer(from,to,amount,takeFee) (contracts/contract.sol#731)
                - refFee = _previousTaxFee (contracts/contract.sol#652)
                - refFee = 0 (contracts/contract.sol#645)
Reentrancy in SCORP.swapBack(uint256) (contracts/contract.sol#734-757):
        External calls:
        - swapTokensForEth(contractBalance - liquidityTokens) (contracts/
contract.sol#745)
                - pancakeV2Router.swapExactTokensForETHSupportingFeeOnTransferToken
s(tokenAmount,0,path,address(this),block.timestamp) (contracts/
contract.sol#764-770)
        - addLiquidity(liquidityTokens,bnbForLiquidity) (contracts/
contract.sol#754)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
        External calls sending eth:
        - addLiquidity(liquidityTokens,bnbForLiquidity) (contracts/
contract.sol#754)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
        State variables written after the call(s):
        - addLiquidity(liquidityTokens,bnbForLiquidity) (contracts/
contract.sol#754)
                - _allowances[owner][spender] = amount (contracts/
contract.sol#670)
Reentrancy in SCORP.transferFrom(address,address,uint256) (contracts/
contract.sol#295-310):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/contract.sol#300)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
                - pancakeV2Router.swapExactTokensForETHSupportingFeeOnTransferToken
s(tokenAmount,0,path,address(this),block.timestamp) (contracts/
contract.sol#764-770)
                - (success,None) = address(marketingWallet).call{value:
bnbForMarketing}() (contracts/contract.sol#756)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/contract.sol#300)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
```

```
                - (success,None) = address(marketingWallet).call{value:
bnbForMarketing}() (contracts/contract.sol#756)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender]
[_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (contracts/
contract.sol#301-308)
                - _allowances[owner][spender] = amount (contracts/
contract.sol#670)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in SCORP._transfer(address,address,uint256) (contracts/
contract.sol#674-733):
        External calls:
        - swapBack(contractTokenBalance) (contracts/contract.sol#694)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
                - pancakeV2Router.swapExactTokensForETHSupportingFeeOnTransferToken
s(tokenAmount,0,path,address(this),block.timestamp) (contracts/
contract.sol#764-770)
                - (success,None) = address(marketingWallet).call{value:
bnbForMarketing}() (contracts/contract.sol#756)
        External calls sending eth:
        - swapBack(contractTokenBalance) (contracts/contract.sol#694)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
                - (success,None) = address(marketingWallet).call{value:
bnbForMarketing}() (contracts/contract.sol#756)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#827)
                - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#827)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/
contract.sol#731)
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#827)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/
contract.sol#723)
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#850)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/
contract.sol#731)
```

```
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#873)
                - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#850)
                - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#873)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/
contract.sol#731)
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#873)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/
contract.sol#723)
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#850)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/
contract.sol#723)
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#432)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/
contract.sol#723)
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#432)
                - _tokenTransfer(from,_burnAddress,tokensForBurn,false) (contracts/
contract.sol#729)
        - Transfer(sender,recipient,tTransferAmount) (contracts/contract.sol#432)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/
contract.sol#731)
Reentrancy in SCORP.swapBack(uint256) (contracts/contract.sol#734-757):
        External calls:
        - swapTokensForEth(contractBalance - liquidityTokens) (contracts/
contract.sol#745)
                - pancakeV2Router.swapExactTokensForETHSupportingFeeOnTransferToken
s(tokenAmount,0,path,address(this),block.timestamp) (contracts/
contract.sol#764-770)
        - addLiquidity(liquidityTokens,bnbForLiquidity) (contracts/
contract.sol#754)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
        External calls sending eth:
        - addLiquidity(liquidityTokens,bnbForLiquidity) (contracts/
contract.sol#754)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
```

```
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/contract.sol#671)
                - addLiquidity(liquidityTokens,bnbForLiquidity) (contracts/
contract.sol#754)
Reentrancy in SCORP.transferFrom(address,address,uint256) (contracts/
contract.sol#295-310):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/contract.sol#300)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
                - pancakeV2Router.swapExactTokensForETHSupportingFeeOnTransferToken
s(tokenAmount,0,path,address(this),block.timestamp) (contracts/
contract.sol#764-770)
                - (success,None) = address(marketingWallet).call{value:
bnbForMarketing}() (contracts/contract.sol#756)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/contract.sol#300)
                - pancakeV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,_burnAddress,block.timestamp) (contracts/
contract.sol#775-782)
                - (success,None) = address(marketingWallet).call{value:
bnbForMarketing}() (contracts/contract.sol#756)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/contract.sol#671)
                - _approve(sender,_msgSender(),_allowances[sender]
[_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (contracts/
contract.sol#301-308)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
SCORP.includeInReward(address) (contracts/contract.sol#398-409) has costly
operations inside a loop:
        - _excluded.pop() (contracts/contract.sol#405)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop
INFO:Detectors:
SafeMath.div(uint256,uint256,string) (contracts/contract.sol#63-72) is never used
and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
SCORP.refFee (contracts/contract.sol#186) is set pre-construction with a non-
```

```
constant function or state variable:
        - _buyRefFee
SCORP.liquidityFee (contracts/contract.sol#187) is set pre-construction with a non-
constant function or state variable:
        - _buyLiquidityFee
SCORP.marketingFee (contracts/contract.sol#188) is set pre-construction with a non-
constant function or state variable:
        - _buyMarketingFee
SCORP.burnFee (contracts/contract.sol#189) is set pre-construction with a non-
constant function or state variable:
        - _buyBurnFee
SCORP._previousTaxFee (contracts/contract.sol#191) is set pre-construction with a
non-constant function or state variable:
        - refFee
SCORP._previousMarketingFee (contracts/contract.sol#192) is set pre-construction
with a non-constant function or state variable:
        - marketingFee
SCORP._previousLiquidityFee (contracts/contract.sol#193) is set pre-construction
with a non-constant function or state variable:
        - liquidityFee
SCORP._previousBurnFee (contracts/contract.sol#194) is set pre-construction with a
non-constant function or state variable:
        - burnFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-
initializing-state
INFO:Detectors:
Pragma version0.8.24 (contracts/contract.sol#3) necessitates a version too recent
to be trusted. Consider deploying with 0.8.18.
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 SCORP.swapBack(uint256) (contracts/contract.sol#734-757):
        - (success,None) = address(marketingWallet).call{value: bnbForMarketing}()
(contracts/contract.sol#756)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls
INFO:Detectors:
Variable Ownable._owner (contracts/contract.sol#83) is not in mixedCase
Function IPancakeRouter01.WETH() (contracts/contract.sol#128) is not in mixedCase
Parameter SCORP.setMarketingWallet(address)._addr (contracts/contract.sol#464) is
not in mixedCase
```

```
Parameter SCORP.setSwapAndLiquifyEnabled(bool)._enabled (contracts/
contract.sol#477) is not in mixedCase
Parameter SCORP.calculateTaxFee(uint256)._amount (contracts/contract.sol#619) is
not in mixedCase
Parameter SCORP.calculateMarketingFee(uint256)._amount (contracts/
contract.sol#623) is not in mixedCase
Parameter SCORP.calculateLiquidity(uint256)._amount (contracts/contract.sol#631)
is not in mixedCase
Constant SCORP._burnAddress (contracts/contract.sol#167) is not in
UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable SCORP._transferBothExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#418) is too similar to
SCORP._transferStandard(address,address,uint256).tTransferAmount (contracts/
contract.sol#817)
Variable SCORP._transferStandard(address,address,uint256).rTransferAmount
(contracts/contract.sol#815) is too similar to
SCORP._transferStandard(address,address,uint256).tTransferAmount (contracts/
contract.sol#817)
Variable SCORP._transferFromExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#860) is too similar to
SCORP._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#862)
Variable SCORP._transferBothExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#418) is too similar to
SCORP._transferToExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#839)
Variable SCORP._transferFromExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#860) is too similar to
SCORP._getValues(uint256).tTransferAmount (contracts/contract.sol#504)
Variable SCORP._transferStandard(address,address,uint256).rTransferAmount
(contracts/contract.sol#815) is too similar to
SCORP._transferToExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#839)
Variable SCORP._transferFromExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#860) is too similar to
SCORP._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#420)
Variable SCORP._transferToExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#837) is too similar to
```

```
SCORP._transferToExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#839)
Variable SCORP._transferFromExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#860) is too similar to
SCORP._transferStandard(address,address,uint256).tTransferAmount (contracts/
contract.sol#817)
Variable SCORP._transferStandard(address,address,uint256).rTransferAmount
(contracts/contract.sol#815) is too similar to
SCORP._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#862)
Variable SCORP._transferFromExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#860) is too similar to
SCORP._transferToExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#839)
Variable SCORP._transferStandard(address,address,uint256).rTransferAmount
(contracts/contract.sol#815) is too similar to
SCORP._getValues(uint256).tTransferAmount (contracts/contract.sol#504)
Variable SCORP._transferBothExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#418) is too similar to
SCORP._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#420)
Variable SCORP._transferFromExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#860) is too similar to
SCORP._getTValues(uint256).tTransferAmount (contracts/contract.sol#540-542)
Variable SCORP._transferStandard(address,address,uint256).rTransferAmount
(contracts/contract.sol#815) is too similar to
SCORP._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#420)
Variable SCORP._getValues(uint256).rTransferAmount (contracts/contract.sol#509) is
too similar to SCORP._getValues(uint256).tTransferAmount (contracts/
contract.sol#504)
Variable SCORP._transferToExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#837) is too similar to
SCORP._transferStandard(address,address,uint256).tTransferAmount (contracts/
contract.sol#817)
Variable SCORP.reflectionFromToken(uint256,bool).rTransferAmount (contracts/
contract.sol#371) is too similar to
SCORP._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#420)
Variable SCORP.reflectionFromToken(uint256,bool).rTransferAmount (contracts/
contract.sol#371) is too similar to SCORP._getTValues(uint256).tTransferAmount
(contracts/contract.sol#540-542)
```

```
Variable
SCORP._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount
(contracts/contract.sol#565-567) is too similar to
SCORP._transferStandard(address,address,uint256).tTransferAmount (contracts/
contract.sol#817)
Variable SCORP._transferBothExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#418) is too similar to
SCORP._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#862)
Variable SCORP._getValues(uint256).rTransferAmount (contracts/contract.sol#509) is
too similar to
SCORP._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#420)
Variable SCORP._transferBothExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#418) is too similar to
SCORP._getValues(uint256).tTransferAmount (contracts/contract.sol#504)
Variable SCORP._getValues(uint256).rTransferAmount (contracts/contract.sol#509) is
too similar to SCORP._getTValues(uint256).tTransferAmount (contracts/
contract.sol#540-542)
Variable SCORP.reflectionFromToken(uint256,bool).rTransferAmount (contracts/
contract.sol#371) is too similar to
SCORP._transferStandard(address,address,uint256).tTransferAmount (contracts/
contract.sol#817)
Variable
SCORP._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount
(contracts/contract.sol#565-567) is too similar to
SCORP._transferToExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#839)
Variable SCORP._getValues(uint256).rTransferAmount (contracts/contract.sol#509) is
too similar to SCORP._transferStandard(address,address,uint256).tTransferAmount
(contracts/contract.sol#817)
Variable SCORP._transferBothExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#418) is too similar to
SCORP._getTValues(uint256).tTransferAmount (contracts/contract.sol#540-542)
Variable SCORP._transferToExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#837) is too similar to
SCORP._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#862)
Variable SCORP.reflectionFromToken(uint256,bool).rTransferAmount (contracts/
contract.sol#371) is too similar to
SCORP._transferToExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#839)
```

```
Variable SCORP._transferToExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#837) is too similar to
SCORP._getValues(uint256).tTransferAmount (contracts/contract.sol#504)
Variable SCORP._transferStandard(address,address,uint256).rTransferAmount
(contracts/contract.sol#815) is too similar to
SCORP._getTValues(uint256).tTransferAmount (contracts/contract.sol#540-542)
Variable
SCORP._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount
(contracts/contract.sol#565-567) is too similar to
SCORP._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#862)
Variable SCORP._getValues(uint256).rTransferAmount (contracts/contract.sol#509) is
too similar to SCORP._transferToExcluded(address,address,uint256).tTransferAmount
(contracts/contract.sol#839)
Variable
SCORP._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount
(contracts/contract.sol#565-567) is too similar to
SCORP._getValues(uint256).tTransferAmount (contracts/contract.sol#504)
Variable SCORP.reflectionFromToken(uint256,bool).rTransferAmount (contracts/
contract.sol#371) is too similar to
SCORP._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#862)
Variable SCORP._transferToExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#837) is too similar to
SCORP._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#420)
Variable SCORP._transferToExcluded(address,address,uint256).rTransferAmount
(contracts/contract.sol#837) is too similar to
SCORP._getTValues(uint256).tTransferAmount (contracts/contract.sol#540-542)
Variable SCORP.reflectionFromToken(uint256,bool).rTransferAmount (contracts/
contract.sol#371) is too similar to SCORP._getValues(uint256).tTransferAmount
(contracts/contract.sol#504)
Variable SCORP._getValues(uint256).rTransferAmount (contracts/contract.sol#509) is
too similar to
SCORP._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#862)
Variable
SCORP._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount
(contracts/contract.sol#565-567) is too similar to
SCORP._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/
contract.sol#420)
Variable
```

```
SCORP._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount
(contracts/contract.sol#565-567) is too similar to
SCORP._getTValues(uint256).tTransferAmount (contracts/contract.sol#540-542)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-
names-too-similar
INFO:Detectors:
SCORP.constructor() (contracts/contract.sol#214-245) uses literals with too many
digits:
        - _tTotal = 1000000000 * 10 ** _decimals (contracts/contract.sol#218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits
INFO:Detectors:
SCORP._buyBurnFee (contracts/contract.sol#179) should be constant
SCORP._buyLiquidityFee (contracts/contract.sol#177) should be constant
SCORP._buyMarketingFee (contracts/contract.sol#178) should be constant
SCORP._buyRefFee (contracts/contract.sol#176) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant
INFO:Detectors:
SCORP.pancakeV2Router (contracts/contract.sol#196) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-immutable
INFO:Slither:. analyzed (8 contracts with 85 detectors), 83 result(s) found
```

ACCORDANCE TO ERC20 STANDARD

```
## Check functions
[▉] totalSupply() is present
        [▉] totalSupply() -> (uint256) (correct return type)
        [▉] totalSupply() is view
[▉] balanceOf(address) is present
        [▉] balanceOf(address) -> (uint256) (correct return type)
        [▉] balanceOf(address) is view
[▉] transfer(address,uint256) is present
        [▉] transfer(address,uint256) -> (bool) (correct return type)
        [▉] Transfer(address,address,uint256) is emitted
[▉] transferFrom(address,address,uint256) is present
        [▉] transferFrom(address,address,uint256) -> (bool) (correct return type)
        [▉] Transfer(address,address,uint256) is emitted
[▉] approve(address,uint256) is present
        [▉] approve(address,uint256) -> (bool) (correct return type)
        [▉] Approval(address,address,uint256) is emitted
[▉] allowance(address,address) is present
        [▉] allowance(address,address) -> (uint256) (correct return type)
        [▉] allowance(address,address) is view
[▉] name() is present
        [▉] name() -> (string) (correct return type)
        [▉] name() is view
[▉] symbol() is present
        [▉] symbol() -> (string) (correct return type)
        [▉] symbol() is view
[▉] decimals() is present
        [ ] decimals() -> (uint256) should return uint8
        [▉] decimals() is view

## Check events
[▉] Transfer(address,address,uint256) is present
        [▉] parameter 0 is indexed
        [▉] parameter 1 is indexed
[▉] Approval(address,address,uint256) is present
        [▉] parameter 0 is indexed
        [▉] parameter 1 is indexed
```

```
[▉] SCORP has increaseAllowance(address,uint256)
```

RAPID LABS

YOUR PROJECT SECURED

RL@RAPIDLABS.FINANCE

RAPIDLABS.FINANCE