auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

HASHISH COIN

DATED : 21 JAN 23'

AUDIT SUMMARY

Project name – Hashish

Date: 21 January , 2023

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

Audit Status: **Passed** **(Contract is developed by Pinksale safu dev)**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	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- Goerli:
all tests were done on Goerli network, each test has its transaction has attached to it.

3- Slither : Static Analysis

TESTNET LINKS

All tests were done using this contract, tests are done on goerli

https://goerli.etherscan.io/token/0xbbc165bc2b4fcf7fe2a5615eaaaa70035a089bc1

Token Address: Not Deployed on Chain

Checksum: 421c76d77563afa1914846b010bd164f395bd34c2102e5e99e0cb9cf173c1d87

Deployer: Not Deployed on Chain

Owner: Not Deployed on Chain

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: None

AUDIT METHODOLOGY

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

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

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

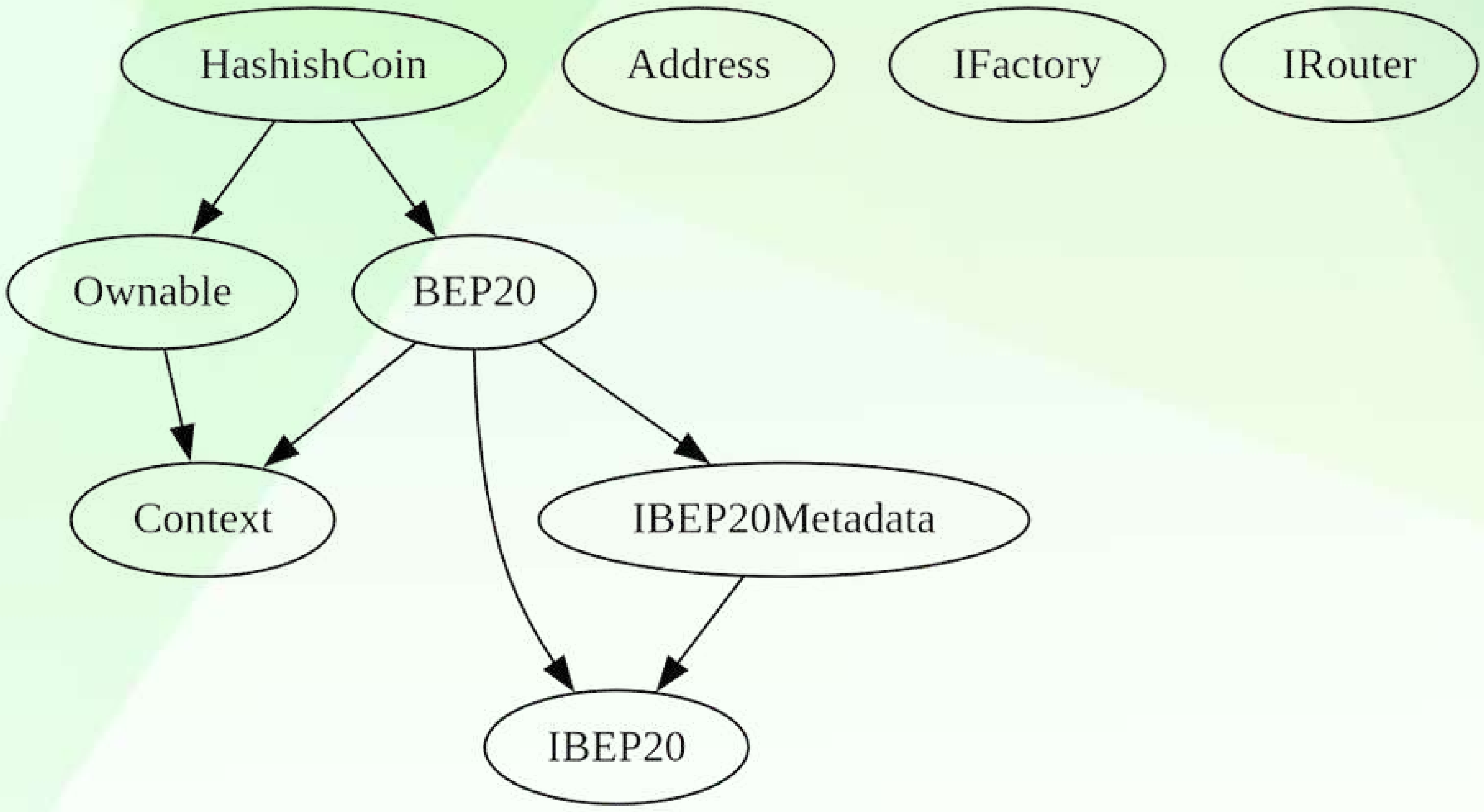
CLASSIFICATION OF RISK

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	0
◆ Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to set taxes (0% tax)
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to set max buy/sell/transfer amounts
- Owner is not able to disable trades
- Owner is not able to mint new tokens

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 ❗ |
```

CONTRACT ASSESMENT

| └ | 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 |||

| └ | <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 ❗ |

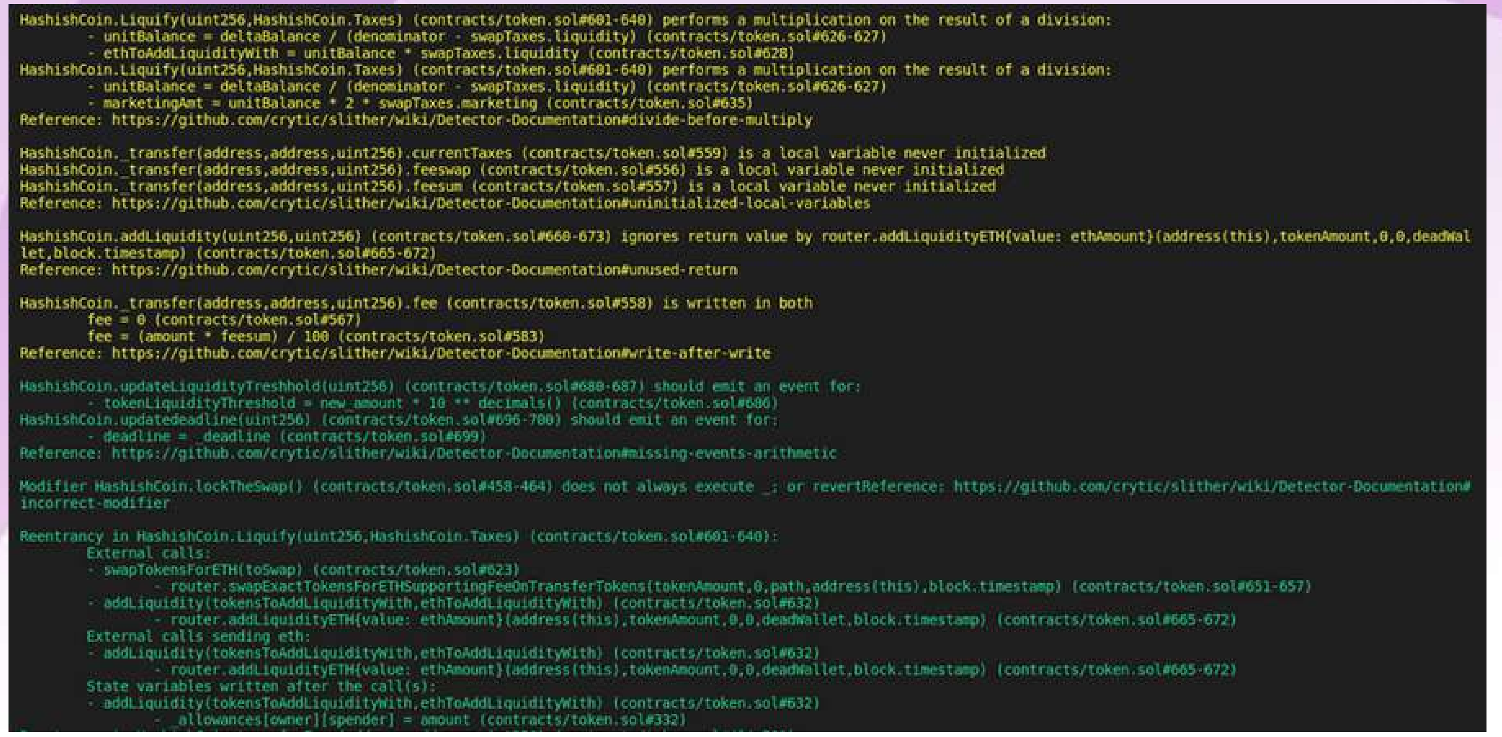
||||||

CONTRACT ASSESMENT

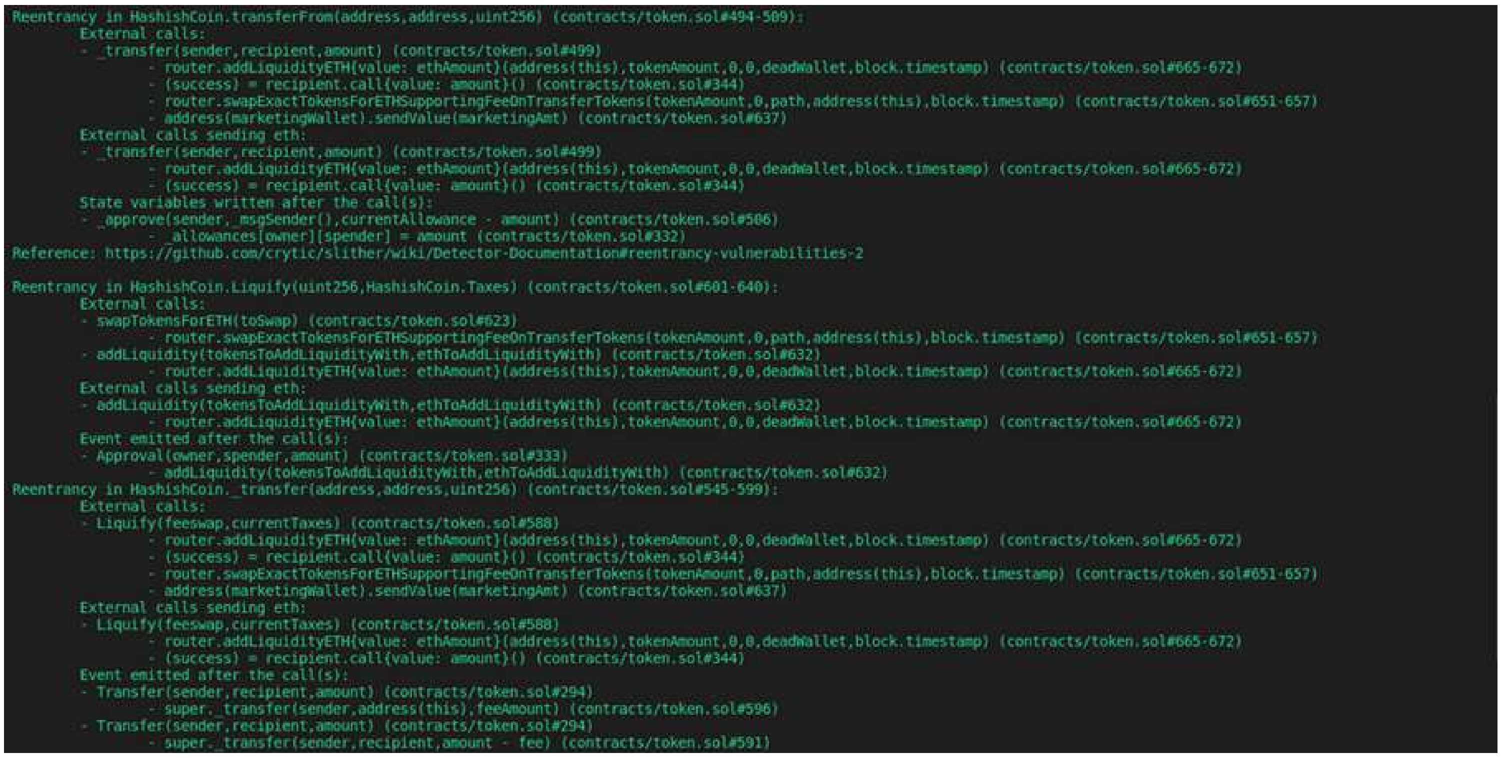
HashishCoin	Implementation	BEP20, Ownable		
└	<Constructor>	Public ❗️	🛑	BEP20
└	approve	Public ❗️	🛑	NO ❗️
└	transferFrom	Public ❗️	🛑	NO ❗️
└	increaseAllowance	Public ❗️	🛑	NO ❗️
└	decreaseAllowance	Public ❗️	🛑	NO ❗️
└	transfer	Public ❗️	🛑	NO ❗️
└	_transfer	Internal 🔒	🛑	
└	Liquify	Private 🔐	🛑	lockTheSwap
└	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

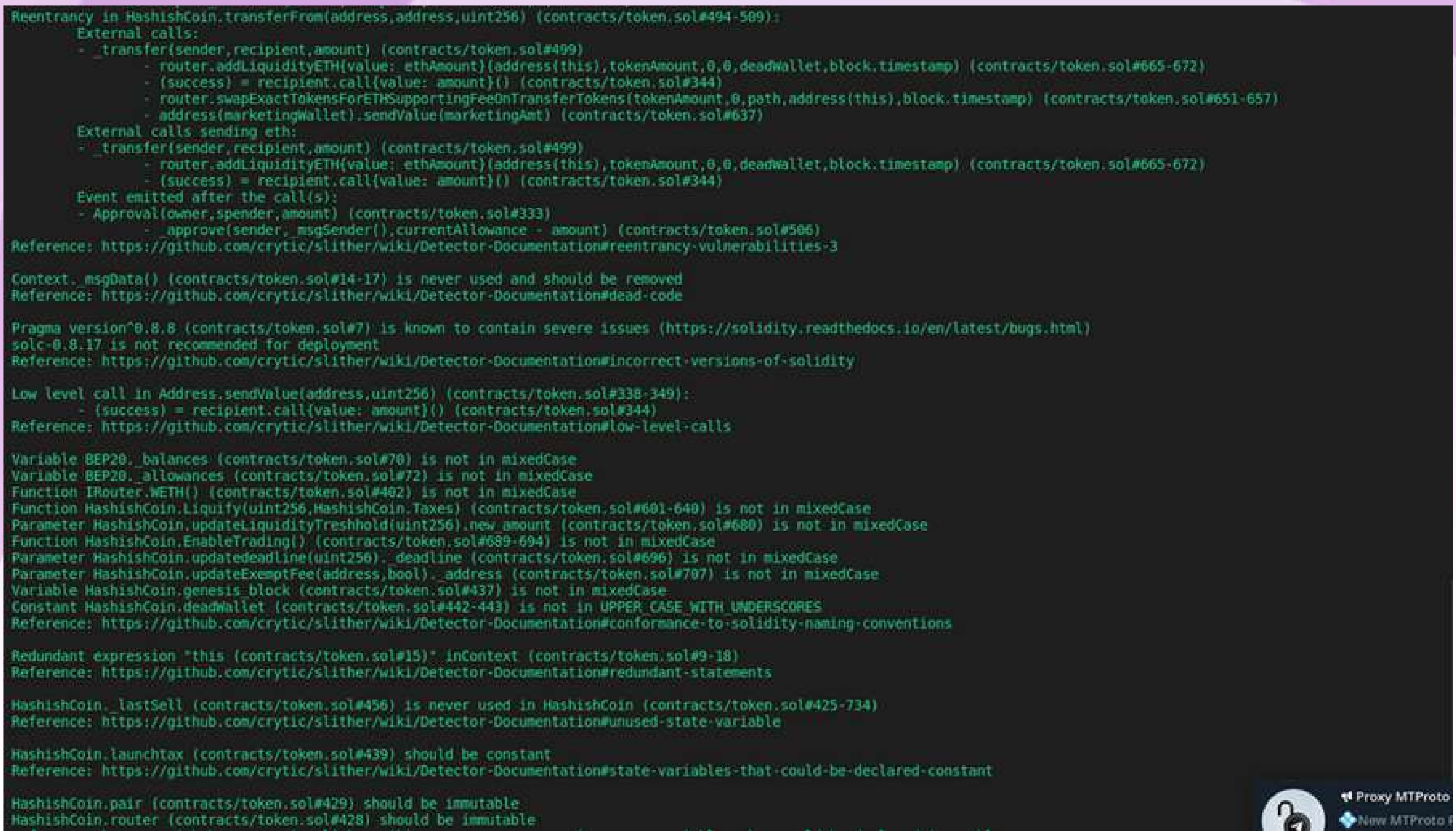


```
HashishCoin.Liquify(uint256,HashishCoin.Taxes) (contracts/token.sol#601-640) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - swapTaxes.liquidity) (contracts/token.sol#626-627)
        - ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity (contracts/token.sol#628)
HashishCoin.Liquify(uint256,HashishCoin.Taxes) (contracts/token.sol#601-640) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - swapTaxes.liquidity) (contracts/token.sol#626-627)
        - marketingAmt = unitBalance * 2 * swapTaxes.marketing (contracts/token.sol#635)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

HashishCoin._transfer(address,address,uint256).currentTaxes (contracts/token.sol#559) is a local variable never initialized
HashishCoin._transfer(address,address,uint256).feeswap (contracts/token.sol#556) is a local variable never initialized
HashishCoin._transfer(address,address,uint256).feesum (contracts/token.sol#557) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

HashishCoin.addLiquidity(uint256,uint256) (contracts/token.sol#660-673) ignores return value by router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWal
let,block.timestamp) (contracts/token.sol#665-672)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

HashishCoin._transfer(address,address,uint256).fee (contracts/token.sol#558) is written in both
        fee = 0 (contracts/token.sol#567)
        fee = (amount * feesum) / 100 (contracts/token.sol#583)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write

HashishCoin.updateLiquidityTreshhold(uint256) (contracts/token.sol#680-687) should emit an event for:
        - tokenLiquidityThreshold = new_amount * 10 ** decimals() (contracts/token.sol#686)
HashishCoin.updatedeadline(uint256) (contracts/token.sol#696-700) should emit an event for:
        - deadline = _deadline (contracts/token.sol#699)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Modifier HashishCoin.lockTheSwap() (contracts/token.sol#458-464) does not always execute _; or revertReference: https://github.com/crytic/slither/wiki/Detector-Documentation#
incorrect-modifier

Reentrancy in HashishCoin.Liquify(uint256,HashishCoin.Taxes) (contracts/token.sol#601-640):
        External calls:
        - swapTokensForETH(toSwap) (contracts/token.sol#623)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/token.sol#651-657)
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#632)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#665-672)
        External calls sending eth:
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#632)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#665-672)
        State variables written after the call(s):
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#632)
                - _allowances[owner][spender] = amount (contracts/token.sol#332)
```



```
Reentrancy in HashishCoin.transferFrom(address,address,uint256) (contracts/token.sol#494-509):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/token.sol#499)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#665-672)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#344)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/token.sol#651-657)
                - address(marketingWallet).sendValue(marketingAmt) (contracts/token.sol#637)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/token.sol#499)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#665-672)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#344)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),currentAllowance - amount) (contracts/token.sol#506)
                - _allowances[owner][spender] = amount (contracts/token.sol#332)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in HashishCoin.Liquify(uint256,HashishCoin.Taxes) (contracts/token.sol#601-640):
        External calls:
        - swapTokensForETH(toSwap) (contracts/token.sol#623)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/token.sol#651-657)
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#632)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#665-672)
        External calls sending eth:
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#632)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#665-672)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/token.sol#333)
                - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#632)
Reentrancy in HashishCoin._transfer(address,address,uint256) (contracts/token.sol#545-599):
        External calls:
        - Liquify(feeswap,currentTaxes) (contracts/token.sol#588)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#665-672)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#344)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/token.sol#651-657)
                - address(marketingWallet).sendValue(marketingAmt) (contracts/token.sol#637)
        External calls sending eth:
        - Liquify(feeswap,currentTaxes) (contracts/token.sol#588)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#665-672)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#344)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (contracts/token.sol#294)
                - super._transfer(sender,address(this),feeAmount) (contracts/token.sol#596)
        - Transfer(sender,recipient,amount) (contracts/token.sol#294)
                - super._transfer(sender,recipient,amount - fee) (contracts/token.sol#591)
```

STATIC ANALYSIS



```
Reentrancy in HashishCoin.transferFrom(address,address,uint256) (contracts/token.sol#494-509):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/token.sol#499)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#665-672)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#344)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/token.sol#651-657)
                - address(marketingWallet).sendValue(marketingAmt) (contracts/token.sol#637)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/token.sol#499)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#665-672)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#344)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/token.sol#333)
                - _approve(sender,_msgSender(),currentAllowance - amount) (contracts/token.sol#506)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Context._msgData() (contracts/token.sol#14-17) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.8 (contracts/token.sol#7) is known to contain severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
solc-0.8.17 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#338-349):
        - (success) = recipient.call{value: amount}() (contracts/token.sol#344)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BEP20._balances (contracts/token.sol#70) is not in mixedCase
Variable BEP20._allowances (contracts/token.sol#72) is not in mixedCase
Function IRouter.WETH() (contracts/token.sol#402) is not in mixedCase
Function HashishCoin.Liquify(uint256,HashishCoin.Taxes) (contracts/token.sol#601-640) is not in mixedCase
Parameter HashishCoin.updateLiquidityTreshhold(uint256).new_amount (contracts/token.sol#680) is not in mixedCase
Function HashishCoin.EnableTrading() (contracts/token.sol#689-694) is not in mixedCase
Parameter HashishCoin.updatedeadline(uint256)._deadline (contracts/token.sol#696) is not in mixedCase
Parameter HashishCoin.updateExemptFee(address,bool)._address (contracts/token.sol#707) is not in mixedCase
Variable HashishCoin.genesis_block (contracts/token.sol#437) is not in mixedCase
Constant HashishCoin.deadWallet (contracts/token.sol#442-443) 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#15)" inContext (contracts/token.sol#9-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

HashishCoin._lastSell (contracts/token.sol#456) is never used in HashishCoin (contracts/token.sol#425-734)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

HashishCoin.launchtax (contracts/token.sol#439) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

HashishCoin.pair (contracts/token.sol#429) should be immutable
HashishCoin.router (contracts/token.sol#428) should be immutable
```

Result => No issues found

FUNCTIONAL TESTING

Functionality tests for ERC20 tokens includes:
- adding liquidity
- buying / selling /transferring (for non-excluded wallets)
-checking tax conversions, tax destinations
- checking auto liquidity

1- Adding Liquidity:

liquidity added on Uniswap v2:

https://goerli.etherscan.io/tx/0x1d15ed6be85175f62c0f7b91d112cee3c76960f1326bfef606e8e6b22629854f

no issue were found on adding liquidity.

2- Buying from a non-excluded wallet:

https://goerli.etherscan.io/tx/0x1397c3631b8717037b4d98e017c9121a1cb77335c94d3355f89c8e96ecd493f7

1% tax on buy, transferred to contract (not reached swap threshold yet)

3- Selling from a non-excluded wallet

https://goerli.etherscan.io/tx/0x04b101f95e26facd3c4838b530e460de4cd1b4deb9194fc07a1518093af9007a

FUNCTIONAL TESTING

1% tax on sell, transferred to contract (not reached swap threshold yet)

4- Swap and Liquifiy

since liquidity tax is 0 and taxes can not be changed later, then auto-liquidity is disabled forever. But to check marketing tax, we transferred 1M tokens to the contract to reach swap threshold and then we performed a sell:

https://goerli.etherscan.io/tx/0x3b18e01dc9ca444073cbd1417baded936fb841d09a16c5188cdde8964adf03b3

marketing wallet received converted ETH tokens received from swapping taxes.

MANUAL TESTING

NO RISKS WERE FOUND IN THE CONTRACT

Social Media Overview

Here are the Social Media Accounts of

Hashish Coin

https://t.me/HASHISHCOIN

https://twitter.com/HASHISHCOIN/status/1616449119803015168

https://hashishcoin.org/

DISCLAIMER

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