QuillAudits

Audit Report
July, 2021

Contents

Scope of Audit

The scope of this audit was to analyze and document the Dalmatian-INU smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	2	0
Closed	0	0	1	6

Introduction

During the period of **July 28,2021 to July 29, 2021** - QuillAudits Team performed a security audit for Dalmatian-INU smart contracts.

The code for the audit was taken from following the official link: https://github.com/PranavGarg01/Dalmatian-INU/blob/main/contracts/Dalmatian.sol

Note	Date	Commit hash
Version 1	28/07/2021	ab8df9f888fcfe4c4d623c03241d1d9d898ff5f7
Version 2	30/07/2021	ef4d28a6df79ab1cc134be94182098cf6af8539c

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found

Medium severity issues

No issues were found

Low level severity issues

1. **Fees is not restored after a special fee transaction**

   Description
   In addLiquidityETH function, the owner gets DALMI tokens from the Pool. If the private key of the owner's wallet is compromised, then it will create a problem.

   Remediation
   Ideally this can be a governance smart contract. On another hand, the owner can accept this risk and handle the private key very securely.

   Status: Acknowledged by the Auditee
   Comments from developer: "Contract ownership will be renounced in future."

2. **Possible to gain ownership**

   Description
   Possible to gain ownership after renouncing the contract ownership. Owner can renounce ownership and make a contract without the owner but he can regain ownership by following the steps below:

   1. Owner calls the lock function in the contract to set the current owner as _previousOwner.
   2. Owner calls unlock to unlock the contract and set _owner = _previousOwner.
   3. Owner called renounceOwnership to leave the contract without the owner.
   4. Owner calls unlock to regain ownership.

Remediation
We suggest removing these lock/unlock functions as this seems not serving a great purpose. Otherwise, always renounce ownership first before calling the lock function.

Status: Fixed
Comments from developer: “The lock and unlock functions have been removed. ”

3. Infinite loop

Line	Code

1117	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);
	}

1025	///@dev Include a user in tax fees reward
	function includeInReward(address account) external onlyOwner() {
	require(_isExcluded[account], "Account is already included");
	for (uint256 i = 0; i < _excluded.length; i++) {
	if (_excluded[i] == account) {
	_excluded[i] = _excluded[_excluded.length - 1];
	_tOwned[account] = 0;
	_isExcluded[account] = false;
	_excluded.pop();
	break;
	}
	}
	}

Description
In includeInReward & _getCurrentSupply functions , for loop do not have _excluded length limit , which costs more gas.

Remediation
We recommend keeping excluded wallets limited, to prevent infinite loop possibility.

Status: Acknowledged by the Auditee
Comments from developer: “Owner will keep a check on the number of excluded wallets.”

Informational

4. Use latest solidity version

pragma solidity 0.8.4

Description
Using the latest solidity will prevent any compiler level bugs.

Remediation
Please use 0.8.6 which is the latest version.

Status: Fixed
This issue was reported in Version1 and found fixed in Version2.

5. Make variables constant

Line	Code
879	string private _name = "Dalmatian INU"; string private _symbol = "DALMI"; uint8 private _decimals = 9;

856	uint256 public easyRewardCycleBlock = 1 days;
	uint256 public disruptiveCoverageFee = 4 ether; // antiwhale
	bool public swapAndLiquifyEnabled = false; // should be true

Description
Values defined for variables easyRewardCycleBlock and disruptiveCoverageFee will be set with the same values in the activateContract function.

Remediation
So, please make them constant by adding keyword 'constant' and remove the same assignments for easyRewardCycleBlock and disruptiveCoverageFee variables in the activateContract function. It will save some gas.

Status: Fixed
This issue was reported in Version1 and found fixed in Version2.

6. Function input parameters lack of check

Line	Code

1151	///@dev To send tokens for BNB rewards to contract address
	function _takeBNB(uint256 tBNB) private {
	uint256 currentRate = _getRate();
	uint256 rBNB = tBNB.mul(currentRate);
	_rOwned[address(this)] = _rOwned[address(this)].add(rBNB);
	if (_isExcluded[address(this)]){
	_tOwned[address(this)] = _tOwned[address(this)].add(tBNB);
	}
	emit Transfer(tx.origin, address(this), tBNB);
	}
	
	function calculateTaxFee(uint256 _amount) private view returns (uint256) {
	return _amount.mul(_taxFee).div(
	10 ** 2
	);
	}
	
	function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
	return _amount.mul(_liquidityFee).div(
	10 ** 2
	);
	}
	
	function calculateBNBFee(uint256 _amount) private view returns (uint256) {
	return _amount.mul(_bnbFee).div(
	10 ** 2
	);
	}

1094	function _getRValues(uint256 tAmount, uint256 tFee, uint256 tLiquidity,uint256 tBNB, uint256 currentRate) private pure returns (uint256, uint256, uint256) {
	uint256 rAmount = tAmount.mul(currentRate);
	uint256 rFee = tFee.mul(currentRate);
	uint256 rLiquidity = tLiquidity.mul(currentRate);
	uint256 rBNB = tBNB.mul(currentRate);
	uint256 rTransferAmount = rAmount.sub(rFee).sub(rLiquidity).sub(rBNB);
	return (rAmount, rTransferAmount, rFee);
	}

Line	Code

986	///@dev Give away tAmount of tokens to contract as fees
	function deliver(uint256 tAmount) external {
	address sender = _msgSender();
	require(!_isExcluded[sender], "Excluded addresses cannot call this function");
	(uint256 rAmount,,,,,,) = _getValues(tAmount);
	_rOwned[sender] = _rOwned[sender].sub(rAmount);
	_rTotal = _rTotal.sub(rAmount);
	_tFeeTotal = _tFeeTotal.add(tAmount);
	}

1129	function _takeLiquidity(uint256 tLiquidity) private {
	//Liquidty + BuyBack
	uint256 currentRate = _getRate();
	uint256 tLiquidity1 = tLiquidity.mul(9).div(14); //2.5% for liquidity + 2% for buyback
	uint256 rLiquidity1 = tLiquidity1.mul(currentRate);
	_rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity1);
	if (_isExcluded[address(this)])
	_tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity1);
	
	//adding liquidity and buyback fees to total
	totalCollectedFees += tLiquidity1;
	
	//Marketing
	currentRate = _getRate();
	tLiquidity1 = tLiquidity.mul(5).div(14); //2.5% for marketing
	uint256 rLiquidity2 = tLiquidity1.mul(currentRate);
	_rOwned[address(MARKETING_WALLET)] = _rOwned[address(MARKETING_WALLET)].add(rLiquidity2);
	if (_isExcluded[address(MARKETING_WALLET)])
	_tOwned[address(MARKETING_WALLET)] = _tOwned[address(MARKETING_WALLET)].add(tLiquidity1);
	emit Transfer(tx.origin, address(MARKETING_WALLET), tLiquidity1);
	}
	
	///@dev To send tokens for BNB rewards to contract address
	function _takeBNB(uint256 tBNB) private {
	uint256 currentRate = _getRate();
	uint256 rBNB = tBNB.mul(currentRate);
	_rOwned[address(this)] = _rOwned[address(this)].add(rBNB);
	if (_isExcluded[address(this)]){
	_tOwned[address(this)] = _tOwned[address(this)].add(tBNB);
	}
	emit Transfer(tx.origin, address(this), tBNB);
	}

Description

Variable validation is not performed in below functions : deliver = tAmount | _getRValues = tAmount , tFee , tLiquidity , tBNB | _takeBNB = tBNB | calculateTaxFee = _amount | calculateBNBFee = _amount | setMaxTxPercent = maxTxPercent | calculateLiquidityFee = _amount | _reflectFee = rFee.

Remediation

Put validation : variable should not be empty and > 0.

Status: Fixed

Comments from Auditee: “Private or internal functions listed in the report do not need any check for parameters because they are called from public/external functions which already have required checks in place. The deliver function, which is an external function, does not need this check because even if input passed to it is 0, there will be no error or problems in the contract working.”

7. Critical operation lacks event log

Description

missing event log for :

- removeAllFee
- restoreAllFee
- calculateBNBReward
- activateContract
- deliver

Remediation

Please write an event log for listed events.

Status: Fixed

Comments from developer: The following functions do not need any events :

- removeAllFee
- restoreAllFee
- calculateBNBReward

removeAllFee and restoreAllFee are functions which will be executed at every transaction, and are required only for the internal functioning of the contract. Emitting events for these functions will not be logical.

calculateBNBReward function is a view function, which means it will not change the state of the contract. The function will be used to calculate the BNB reward redeemable by the user. There is no need to emit events for transactions that do not change the state.

8. External instead of public

Description

Consider specifying function visibility to “external” instead of “public”, if that function is not being called internally. It will save some gas as well.
https://ethereum.stackexchange.com/questions/32353/what-is-the-difference-between-an-internal-external-and-public-private-function/32464

Status: Fixed
This issue was reported in Version1 and found fixed in Version2.

9. HardCoded Address

Line	Code

833	`contract Dalmatian is Context, IBEP20, Ownable, ReentrancyGuard {`
	`  using SafeMath for uint256;`
	`  using Address for address;`
	
	`  address payable public constant CURELLA_WALLET = payable(address(0xE3d3D329e2287A7922D35F1D23e0FC146a7c25C9));`
	`  address public constant MARKETING_WALLET = address(0xcA3c07f6764b7c6Ce9555279dE9ec82455e12725);`

Description
There are hardcoded wallet addresses.

Remediation
Owner must confirm before deploying.

Status: Fixed
Comments from developer: “The addresses have been verified. ”

Functional test

Function Names	Testing results
name	Passed
symbol	Passed
decimals	Passed
totalSupply	Passed
balanceOf	Passed
transfer	Passed
allowance	Passed
approve	Passed
transferFrom	Passed
increaseAllowance	Passed
decreaseAllowance	Passed
isExcludedFromReward	Passed
totalFees	Passed
deliver	Passed
reflectionFromToken	Passed
tokenFromReflection	Passed
excludeFromReward	Passed
includeInReward	Passed
_transferBothExcluded	Passed
excludeFromFee	Passed

Function Names	Testing results
includeInFee	Passed
setTaxFeePercent	Passed
setLiquidityFeePercent	Passed
setBNBFeePercent	Passed
setSwapAndLiquifyEnabled	Passed
receive	Passed
_reflectFee	Passed
_getValues	Passed
_getTValues	Passed
_getRValues	Passed
_getRate	Passed
_getCurrentSupply	Passed
_takeLiquidity	Passed
_takeBNB	Passed
calculateTaxFee	Passed
calculateLiquidityFee	Passed
calculateBNBFee	Passed
removeAllFee	Passed
restoreAllFee	Passed
isExcludedFromFee	Passed

Function Names	Testing results
_approve	Passed
_transfer	Passed
_tokenTransfer	Passed
_transferStandard	Passed
_transferToExcluded	Passed
_transferFromExcluded	Passed
setMaxTxPercent	Passed
setExcludeFromMaxTx	Passed
calculateBNBReward	Passed
getRewardCycleBlock	Passed
claimBNBReward	Passed
topUpClaimCycleAfterTransfer	Passed
ensureMaxTxAmount	Passed
disruptiveTransfer	Passed
swapAndLiquify	Passed
swapBurnAndLiquify	Passed
activateContract	Passed
changerewardCycleBlock	Passed
reflectionfeestartstop	Passed
changethreshHoldTopUpRate	Passed

Function Names	Testing results
nonReentrant	Passed
isHuman	Passed
owner	Passed
onlyOwner	Passed
renounceOwnership	Passed
transferOwnership	Passed
geUnlockTime	Passed
lock	Passed
unlock	Passed
_msgSender	Passed
_msgData	Passed

Automated Testing

Slither

```
INFO:Detectors:
Utils.swapETHForTokens(address,address,uint256) (Dalmatian.sol#741-760) sends eth to arbitrary user
        Dangerous calls:
        - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.timestamp +
360) (Dalmatian.sol#754-759)
Utils.addLiquidity(address,address,uint256,uint256) (Dalmatian.sol#762-779) sends eth to arbitrary user
        Dangerous calls:
        - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.sol#771-7
78)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in Dalmatian._transfer(address,address,uint256,uint256) (Dalmatian.sol#1212-1261):
        External calls:
        - swapAndLiquify(from,to) (Dalmatian.sol#1225)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - Utils.swapTokensForEth(address(pancakeRouter),EthPortion) (Dalmatian.sol#1458)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Dal
matian.sol#732-738)
                - Utils.swapETHForTokens(address(pancakeRouter),address(0xdEaD),burnAmountEth) (Dalmatian.sol#1467)
                - Utils.addLiquidity(address(pancakeRouter),owner(),liqPortion,liqBalance) (Dalmatian.sol#1470)
                - Utils.swapTokensForEth(address(pancakeRouter),sellTokens) (Dalmatian.sol#1439)
        External calls sending eth:
        - swapAndLiquify(from,to) (Dalmatian.sol#1225)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rBNB) (Dalmatian.sol#1155)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity1) (Dalmatian.sol#1134)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Dalmatian.sol#1287)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Dalmatian.sol#1297)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Dalmatian.sol#1042)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Dalmatian.sol#1309)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Dalmatian.sol#1288)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Dalmatian.sol#1299)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Dalmatian.sol#1310)
                - _rOwned[address(MARKETING_WALLET)] = _rOwned[address(MARKETING_WALLET)].add(rLiquidity2) (Dalmatian.sol#1145)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Dalmatian.sol#1044)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _rTotal = _rTotal.sub(rFee) (Dalmatian.sol#1083)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tBNB) (Dalmatian.sol#1157)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity1) (Dalmatian.sol#1136)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (Dalmatian.sol#1308)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (Dalmatian.sol#1041)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (Dalmatian.sol#1298)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (Dalmatian.sol#1043)
                - _tOwned[address(MARKETING_WALLET)] = _tOwned[address(MARKETING_WALLET)].add(tLiquidity1) (Dalmatian.sol#1147)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - totalCollectedFees += tLiquidity1 (Dalmatian.sol#1139)
Reentrancy in Dalmatian.claimBNBReward() (Dalmatian.sol#1349-1366):
        External calls:
        - (sentTreasury) = address(CURELLA_WALLET).call{value: reward.div(10)}() (Dalmatian.sol#1356)
        State variables written after the call(s):
        - nextAvailableClaimDate[msg.sender] = block.timestamp + getRewardCycleBlock() (Dalmatian.sol#1361)
Reentrancy in Dalmatian.swapAndLiquify(address,address) (Dalmatian.sol#1403-1443):
        External calls:
        - swapBurnAndLiquify(burnTokens,swapTokens) (Dalmatian.sol#1435)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - Utils.swapTokensForEth(address(pancakeRouter),EthPortion) (Dalmatian.sol#1458)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Dal
```

```
        - swapBurnAndLiquify(burnTokens,swapTokens) (Dalmatian.sol#1435)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - Utils.swapTokensForEth(address(pancakeRouter),EthPortion) (Dalmatian.sol#1458)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Dal
matian.sol#732-738)
                - Utils.swapETHForTokens(address(pancakeRouter),address(0xdEaD),burnAmountEth) (Dalmatian.sol#1467)
                - Utils.addLiquidity(address(pancakeRouter),owner(),liqPortion,liqBalance) (Dalmatian.sol#1470)
        External calls sending eth:
        - swapBurnAndLiquify(burnTokens,swapTokens) (Dalmatian.sol#1435)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
        State variables written after the call(s):
        - totalCollectedFees = totalCollectedFees.sub(swapTokens).sub(burnTokens) (Dalmatian.sol#1436)
Reentrancy in Dalmatian.swapAndLiquify(address,address) (Dalmatian.sol#1403-1443):
        External calls:
        - swapBurnAndLiquify(burnTokens,swapTokens) (Dalmatian.sol#1435)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - Utils.swapTokensForEth(address(pancakeRouter),EthPortion) (Dalmatian.sol#1458)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Dal
matian.sol#732-738)
                - Utils.swapETHForTokens(address(pancakeRouter),address(0xdEaD),burnAmountEth) (Dalmatian.sol#1467)
                - Utils.addLiquidity(address(pancakeRouter),owner(),liqPortion,liqBalance) (Dalmatian.sol#1470)
        - Utils.swapTokensForEth(address(pancakeRouter),sellTokens) (Dalmatian.sol#1439)
        External calls sending eth:
        - swapBurnAndLiquify(burnTokens,swapTokens) (Dalmatian.sol#1435)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
        State variables written after the call(s):
        - inSwapAndLiquify = false (Dalmatian.sol#1441)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
Dalmatian._takeLiquidity(uint256) (Dalmatian.sol#1129-1149) performs a multiplication on the result of a division:
        -tLiquidity1 = tLiquidity.mul(9).div(14) (Dalmatian.sol#1132)
        -rLiquidity1 = tLiquidity1.mul(currentRate) (Dalmatian.sol#1133)
Dalmatian._takeLiquidity(uint256) (Dalmatian.sol#1129-1149) performs a multiplication on the result of a division:
        -tLiquidity1 = tLiquidity.mul(5).div(14) (Dalmatian.sol#1143)
        -rLiquidity2 = tLiquidity1.mul(currentRate) (Dalmatian.sol#1144)
Dalmatian.swapAndLiquify(address,address) (Dalmatian.sol#1403-1443) performs a multiplication on the result of a division:
        -feesPercent = totalCollectedFees.mul(1000000).div(balanceOf(address(this))) (Dalmatian.sol#1425)
        -swapTokens = contractTokenBalance.mul(feesPercent).mul(5).div(9).div(1000000) (Dalmatian.sol#1430)
Dalmatian.swapAndLiquify(address,address) (Dalmatian.sol#1403-1443) performs a multiplication on the result of a division:
        -feesPercent = totalCollectedFees.mul(1000000).div(balanceOf(address(this))) (Dalmatian.sol#1425)
        -burnTokens = contractTokenBalance.mul(feesPercent).mul(4).div(9).div(1000000) (Dalmatian.sol#1433)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Utils.addLiquidity(address,address,uint256,uint256) (Dalmatian.sol#762-779) ignores return value by pancakeRouter.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.sol#771-778)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Dalmatian.allowance(address,address).owner (Dalmatian.sol#951) shadows:
        - Ownable.owner() (Dalmatian.sol#414-416) (function)
Dalmatian._approve(address,address,uint256).owner (Dalmatian.sol#1204) shadows:
        - Ownable.owner() (Dalmatian.sol#414-416) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in Dalmatian._transfer(address,address,uint256,uint256) (Dalmatian.sol#1212-1261):
        External calls:
        - swapAndLiquify(from,to) (Dalmatian.sol#1225)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - Utils.swapTokensForEth(address(pancakeRouter),EthPortion) (Dalmatian.sol#1458)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
```

```
ol#771-778)
                - Utils.swapTokensForEth(address(pancakeRouter),EthPortion) (Dalmatian.sol#1458)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Dal
matian.sol#732-738)
                - Utils.swapETHForTokens(address(pancakeRouter),address(0xdEaD),burnAmountEth) (Dalmatian.sol#1467)
                - Utils.addLiquidity(address(pancakeRouter),owner(),liqPortion,liqBalance) (Dalmatian.sol#1470)
                - Utils.swapTokensForEth(address(pancakeRouter),sellTokens) (Dalmatian.sol#1439)
        External calls sending eth:
        - swapAndLiquify(from,to) (Dalmatian.sol#1225)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
        State variables written after the call(s):
        - _bnbFee = 1 (Dalmatian.sol#1242)
        - _bnbFee = 11 (Dalmatian.sol#1250)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _bnbFee = _previousBNBFee (Dalmatian.sol#1196)
                - _bnbFee = 0 (Dalmatian.sol#1189)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _liquidityFee = _previousLiquidityFee (Dalmatian.sol#1195)
                - _liquidityFee = 0 (Dalmatian.sol#1188)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _previousBNBFee = _bnbFee (Dalmatian.sol#1185)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _previousLiquidityFee = _liquidityFee (Dalmatian.sol#1184)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _previousTaxFee = _taxFee (Dalmatian.sol#1183)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _tFeeTotal = _tFeeTotal.add(tFee) (Dalmatian.sol#1084)
        - _taxFee = 10 (Dalmatian.sol#1239)
        - _taxFee = 0 (Dalmatian.sol#1247)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _taxFee = _previousTaxFee (Dalmatian.sol#1194)
                - _taxFee = 0 (Dalmatian.sol#1187)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - _taxFee = 0 (Dalmatian.sol#1187)
        - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
                - nextAvailableClaimDate[recipient] = nextAvailableClaimDate[recipient] + Utils.calculateTopUpClaim(currentRecipientBalan
ce,basedRewardCycleBlock,threshHoldTopUpRate,amount) (Dalmatian.sol#1372-1377)
Reentrancy in Dalmatian.constructor(address) (Dalmatian.sol#894-923):
        External calls:
        - pancakePair = IPancakeFactory(_pancakeRouter.factory()).createPair(address(this),_pancakeRouter.WETH()) (Dalmatian.sol#901-902)
        State variables written after the call(s):
        - _isExcludedFromFee[owner()] = true (Dalmatian.sol#908)
        - _isExcludedFromFee[address(this)] = true (Dalmatian.sol#909)
        - _isExcludedFromFee[MARKETING_WALLET] = true (Dalmatian.sol#910)
        - _isExcludedFromFee[CURELLA_WALLET] = true (Dalmatian.sol#911)
        - _isExcludedFromMaxTx[owner()] = true (Dalmatian.sol#916)
        - _isExcludedFromMaxTx[address(this)] = true (Dalmatian.sol#917)
        - _isExcludedFromMaxTx[address(0xdEaD)] = true (Dalmatian.sol#918)
        - _isExcludedFromMaxTx[address(0)] = true (Dalmatian.sol#919)
        - _isExcludedFromMaxTx[address(CURELLA_WALLET)] = true (Dalmatian.sol#920)
        - pancakeRouter = _pancakeRouter (Dalmatian.sol#905)
Reentrancy in Dalmatian.transferFrom(address,address,uint256) (Dalmatian.sol#960-964):
        External calls:
        - _transfer(sender,recipient,amount,0) (Dalmatian.sol#961)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - Utils.swapTokensForEth(address(pancakeRouter),EthPortion) (Dalmatian.sol#1458)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Dal
matian.sol#732-738)
                - Utils.swapETHForTokens(address(pancakeRouter),address(0xdEaD),burnAmountEth) (Dalmatian.sol#1467)
                - Utils.addLiquidity(address(pancakeRouter),owner(),liqPortion,liqBalance) (Dalmatian.sol#1470)
                - Utils.swapTokensForEth(address(pancakeRouter),sellTokens) (Dalmatian.sol#1439)
        External calls sending eth:
        - _transfer(sender,recipient,amount,0) (Dalmatian.sol#961)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
        State variables written after the call(s):
```

```
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (Dalmatian
.sol#962)
                - _allowances[owner][spender] = amount (Dalmatian.sol#1208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Dalmatian._transfer(address,address,uint256,uint256) (Dalmatian.sol#1212-1261):
        External calls:
        - swapAndLiquify(from,to) (Dalmatian.sol#1225)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - Utils.swapTokensForEth(address(pancakeRouter),EthPortion) (Dalmatian.sol#1458)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Dal
matian.sol#732-738)
                - Utils.swapETHForTokens(address(pancakeRouter),address(0xdEaD),burnAmountEth) (Dalmatian.sol#1467)
                - Utils.addLiquidity(address(pancakeRouter),owner(),liqPortion,liqBalance) (Dalmatian.sol#1470)
                - Utils.swapTokensForEth(address(pancakeRouter),sellTokens) (Dalmatian.sol#1439)
        External calls sending eth:
        - swapAndLiquify(from,to) (Dalmatian.sol#1225)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
        Event emitted after the call(s):
        - Transfer(tx.origin,address(this),tBNB) (Dalmatian.sol#1159)
                - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
        - Transfer(sender,recipient,tTransferAmount) (Dalmatian.sol#1292)
                - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
        - Transfer(sender,recipient,tTransferAmount) (Dalmatian.sol#1314)
                - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
        - Transfer(sender,recipient,tTransferAmount) (Dalmatian.sol#1303)
                - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
        - Transfer(tx.origin,address(MARKETING_WALLET),tLiquidity1) (Dalmatian.sol#1148)
                - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
        - Transfer(tx.origin,address(MARKETING_WALLET),tLiquidity1) (Dalmatian.sol#1148)
                - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
        - Transfer(sender,recipient,tTransferAmount) (Dalmatian.sol#1048)
                - _tokenTransfer(from,to,amount,takeFee) (Dalmatian.sol#1260)
Reentrancy in Dalmatian.claimBNBReward() (Dalmatian.sol#1349-1366):
        External calls:
        - (sentTreasury) = address(CURELLA_WALLET).call{value: reward.div(10)}() (Dalmatian.sol#1356)
        Event emitted after the call(s):
        - ClaimBNBSuccessfully(msg.sender,reward,nextAvailableClaimDate[msg.sender]) (Dalmatian.sol#1362)
Reentrancy in Dalmatian.constructor(address) (Dalmatian.sol#894-923):
        External calls:
        - pancakePair = IPancakeFactory(_pancakeRouter.factory()).createPair(address(this),_pancakeRouter.WETH()) (Dalmatian.sol#901-902)
        Event emitted after the call(s):
        - Transfer(address(0),_msgSender(),_tTotal) (Dalmatian.sol#922)
Reentrancy in Dalmatian.swapBurnAndLiquify(uint256,uint256) (Dalmatian.sol#1446-1473):
        External calls:
        - Utils.swapTokensForEth(address(pancakeRouter),EthPortion) (Dalmatian.sol#1458)
        - Utils.swapETHForTokens(address(pancakeRouter),address(0xdEaD),burnAmountEth) (Dalmatian.sol#1467)
        - Utils.addLiquidity(address(pancakeRouter),owner(),liqPortion,liqBalance) (Dalmatian.sol#1470)
        Event emitted after the call(s):
        - SwapBurnLiquify(liqPortion,newBalance,liqBalance,burnAmountEth) (Dalmatian.sol#1472)
Reentrancy in Dalmatian.transferFrom(address,address,uint256) (Dalmatian.sol#960-964):
        External calls:
        - _transfer(sender,recipient,amount,0) (Dalmatian.sol#961)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - Utils.swapTokensForEth(address(pancakeRouter),EthPortion) (Dalmatian.sol#1458)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Dal
matian.sol#732-738)
                - Utils.swapETHForTokens(address(pancakeRouter),address(0xdEaD),burnAmountEth) (Dalmatian.sol#1467)
                - Utils.addLiquidity(address(pancakeRouter),owner(),liqPortion,liqBalance) (Dalmatian.sol#1470)
                - Utils.swapTokensForEth(address(pancakeRouter),sellTokens) (Dalmatian.sol#1439)
        External calls sending eth:
        - _transfer(sender,recipient,amount,0) (Dalmatian.sol#961)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
```

```
matian.sol#732-738)
                - Utils.swapETHForTokens(address(pancakeRouter),address(0xdEaD),burnAmountEth) (Dalmatian.sol#1467)
                - Utils.addLiquidity(address(pancakeRouter),owner(),liqPortion,liqBalance) (Dalmatian.sol#1470)
                - Utils.swapTokensForEth(address(pancakeRouter),sellTokens) (Dalmatian.sol#1439)
        External calls sending eth:
        - _transfer(sender,recipient,amount,0) (Dalmatian.sol#961)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (Dalmatian.s
ol#771-778)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.time
stamp + 360) (Dalmatian.sol#754-759)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Dalmatian.sol#1209)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (D
almatian.sol#962)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (Dalmatian.sol#461-466) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > _lockTime,Contract is locked until 7 days) (Dalmatian.sol#463)
Dalmatian.getRewardCycleBlock() (Dalmatian.sol#1343-1346) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= disableEasyRewardFrom (Dalmatian.sol#1344)
Dalmatian.claimBNBReward() (Dalmatian.sol#1349-1366) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(nextAvailableClaimDate[msg.sender] <= block.timestamp,Error: next available not reached) (Dalmatian.sol#13
50)
Dalmatian.ensureMaxTxAmount(address,address,uint256,uint256) (Dalmatian.sol#1380-1394) uses timestamp for comparisons
        Dangerous comparisons:
        - value < disruptiveCoverageFee && block.timestamp >= disruptiveTransferEnabledFrom (Dalmatian.sol#1390)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (Dalmatian.sol#266-275) uses assembly
        - INLINE ASM (Dalmatian.sol#273)
Address._functionCallWithValue(address,bytes,uint256,string) (Dalmatian.sol#359-380) uses assembly
        - INLINE ASM (Dalmatian.sol#372-375)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Dalmatian.ensureMaxTxAmount(address,address,uint256,uint256) (Dalmatian.sol#1380-1394) compares to a boolean constant:
INFO:Detectors:
Dalmatian.ensureMaxTxAmount(address,address,uint256,uint256) (Dalmatian.sol#1380-1394) compares to a boolean constant:
        -_isExcludedFromMaxTx[from] == false && _isExcludedFromMaxTx[to] == false (Dalmatian.sol#1387-1388)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.8.4', '>=0.6.8']
        - 0.8.4 (Dalmatian.sol#3)
        - >=0.6.8 (Dalmatian.sol#673)
        - ABIEncoderV2 (Dalmatian.sol#831)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (Dalmatian.sol#359-380) is never used and should be removed
Address.functionCall(address,bytes) (Dalmatian.sol#319-321) is never used and should be removed
Address.functionCall(address,bytes,string) (Dalmatian.sol#329-331) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Dalmatian.sol#344-346) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Dalmatian.sol#354-357) is never used and should be removed
Address.isContract(address) (Dalmatian.sol#266-275) is never used and should be removed
Address.sendValue(address,uint256) (Dalmatian.sol#293-299) is never used and should be removed
Context._msgData() (Dalmatian.sol#238-241) is never used and should be removed
SafeMath.mod(uint256,uint256) (Dalmatian.sol#211-213) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Dalmatian.sol#227-230) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Dalmatian._rTotal (Dalmatian.sol#852) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Dalmatian._maxTxAmount (Dalmatian.sol#858) is set pre-construction with a non-constant function or state variable:
        - _tTotal
Dalmatian._previousTaxFee (Dalmatian.sol#868) is set pre-construction with a non-constant function or state variable:
        - _taxFee
Dalmatian._previousLiquidityFee (Dalmatian.sol#871) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
Dalmatian._previousBNBFee (Dalmatian.sol#874) is set pre-construction with a non-constant function or state variable:
        - _bnbFee
Dalmatian.minTokenNumberToSell (Dalmatian.sol#877) is set pre-construction with a non-constant function or state variable:
        - _tTotal.mul(1).div(100000)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version0.8.4 (Dalmatian.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.6.8 (Dalmatian.sol#673) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Dalmatian.sol#293-299):
        - (success) = recipient.call{value: amount}() (Dalmatian.sol#297)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (Dalmatian.sol#359-380):
        - (success,returndata) = target.call{value: weiValue}(data) (Dalmatian.sol#363)
Low level call in Dalmatian.claimBNBReward() (Dalmatian.sol#1349-1366):
        - (sentTreasury) = address(CURELLA_WALLET).call{value: reward.div(10)}() (Dalmatian.sol#1356)
        - (sent) = address(msg.sender).call{value: reward}() (Dalmatian.sol#1364)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IPancakePair.DOMAIN_SEPARATOR() (Dalmatian.sol#500) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (Dalmatian.sol#501) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (Dalmatian.sol#518) is not in mixedCase
Function IPancakeRouter01.WETH() (Dalmatian.sol#538) is not in mixedCase
Parameter Dalmatian.setSwapAndLiquifyEnabled(bool)._enabled (Dalmatian.sol#1074) is not in mixedCase
Parameter Dalmatian.calculateTaxFee(uint256)._amount (Dalmatian.sol#1162) is not in mixedCase
Parameter Dalmatian.calculateLiquidityFee(uint256)._amount (Dalmatian.sol#1168) is not in mixedCase
Parameter Dalmatian.calculateBNBFee(uint256)._amount (Dalmatian.sol#1174) is not in mixedCase
Parameter Dalmatian.setExcludeFromMaxTx(address,bool)._address (Dalmatian.sol#1323) is not in mixedCase
Parameter Dalmatian.reflectionfeestartstop(bool)._value (Dalmatian.sol#1510) is not in mixedCase
Parameter Dalmatian.changethreshHoldTopUpRate(uint256)._newrate (Dalmatian.sol#1516) is not in mixedCase
Variable Dalmatian._maxTxAmount (Dalmatian.sol#858) is not in mixedCase
Variable Dalmatian._taxFee (Dalmatian.sol#867) is not in mixedCase
Variable Dalmatian._liquidityFee (Dalmatian.sol#870) is not in mixedCase
Variable Dalmatian._bnbFee (Dalmatian.sol#873) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Dalmatian.sol#239)" inContext (Dalmatian.sol#233-242)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Dalmatian.sol#543
) is too similar to IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Dalmat
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Dalmatian.sol#239)" inContext (Dalmatian.sol#233-242)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Dalmatian.sol#543
) is too similar to IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Dalmat
ian.sol#544)
Variable Dalmatian._takeLiquidity(uint256).rLiquidity1 (Dalmatian.sol#1133) is too similar to Dalmatian._takeLiquidity(uint256).rLiquidit
y2 (Dalmatian.sol#1144)
Variable Dalmatian._transferStandard(address,address,uint256).rTransferAmount (Dalmatian.sol#1286) is too similar to Dalmatian._getTValue
s(uint256).tTransferAmount (Dalmatian.sol#1098)
Variable Dalmatian.reflectionFromToken(uint256,bool).rTransferAmount (Dalmatian.sol#1002) is too similar to Dalmatian._transferFromExclud
ed(address,address,uint256).tTransferAmount (Dalmatian.sol#1307)
Variable Dalmatian._transferFromExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1307) is too similar to Dalmatian._trans
ferFromExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1307)
Variable Dalmatian._transferStandard(address,address,uint256).rTransferAmount (Dalmatian.sol#1286) is too similar to Dalmatian._transferF
romExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1307)
Variable Dalmatian.reflectionFromToken(uint256,bool).rTransferAmount (Dalmatian.sol#1002) is too similar to Dalmatian._getTValues(uint256
).tTransferAmount (Dalmatian.sol#1098)
Variable Dalmatian._transferStandard(address,address,uint256).rTransferAmount (Dalmatian.sol#1286) is too similar to Dalmatian._transferS
tandard(address,address,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian._transferStandard(address,address,uint256).rTransferAmount (Dalmatian.sol#1286) is too similar to Dalmatian._transferB
othExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1040)
Variable Dalmatian._transferStandard(address,address,uint256).rTransferAmount (Dalmatian.sol#1286) is too similar to Dalmatian._getValues
(uint256).tTransferAmount (Dalmatian.sol#1089)
Variable Dalmatian._getValues(uint256).rTransferAmount (Dalmatian.sol#1090) is too similar to Dalmatian._getTValues(uint256).tTransferAmo
unt (Dalmatian.sol#1098)
Variable Dalmatian._transferBothExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1040) is too similar to Dalmatian._trans
ferBothExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1040)
Variable Dalmatian._transferFromExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1307) is too similar to Dalmatian._getTV
alues(uint256).tTransferAmount (Dalmatian.sol#1098)
Variable Dalmatian._transferStandard(address,address,uint256).rTransferAmount (Dalmatian.sol#1286) is too similar to Dalmatian._transferT
oExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1296)
Variable Dalmatian._transferBothExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1040) is too similar to Dalmatian._trans
ferToExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1296)
Variable Dalmatian._getValues(uint256).rTransferAmount (Dalmatian.sol#1090) is too similar to Dalmatian._transferToExcluded(address,addre
```

```
(uint256).tTransferAmount (Dalmatian.sol#1089)
Variable Dalmatian._getValues(uint256).rTransferAmount (Dalmatian.sol#1090) is too similar to Dalmatian._getTValues(uint256).tTransferAmo
unt (Dalmatian.sol#1098)
Variable Dalmatian._transferBothExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1040) is too similar to Dalmatian._trans
ferBothExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1040)
Variable Dalmatian._transferFromExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1307) is too similar to Dalmatian._getTV
alues(uint256).tTransferAmount (Dalmatian.sol#1098)
Variable Dalmatian._transferStandard(address,address,uint256).rTransferAmount (Dalmatian.sol#1286) is too similar to Dalmatian._transferT
oExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1296)
Variable Dalmatian._transferBothExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1040) is too similar to Dalmatian._trans
ferToExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1296)
Variable Dalmatian._getValues(uint256).rTransferAmount (Dalmatian.sol#1090) is too similar to Dalmatian._transferToExcluded(address,addre
ss,uint256).tTransferAmount (Dalmatian.sol#1296)
Variable Dalmatian._transferBothExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1040) is too similar to Dalmatian._getTV
alues(uint256).tTransferAmount (Dalmatian.sol#1098)
Variable Dalmatian._getValues(uint256).rTransferAmount (Dalmatian.sol#1090) is too similar to Dalmatian._transferFromExcluded(address,add
ress,uint256).tTransferAmount (Dalmatian.sol#1307)
Variable Dalmatian._getValues(uint256).rTransferAmount (Dalmatian.sol#1090) is too similar to Dalmatian._transferFromExcluded(address,add
ress,uint256).tTransferAmount (Dalmatian.sol#1307)
Variable Dalmatian._transferBothExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1040) is too similar to Dalmatian._trans
ferFromExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1307)
Variable Dalmatian._takeLiquidity(uint256).rLiquidity1 (Dalmatian.sol#1133) is too similar to Dalmatian._takeLiquidity(uint256).tLiquidit
y1 (Dalmatian.sol#1132)
Variable Dalmatian._transferFromExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1307) is too similar to Dalmatian._trans
ferToExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1296)
Variable Dalmatian._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Dalmatian.sol#1107) is too similar to Dalmatian.
_transferStandard(address,address,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian._transferToExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1296) is too similar to Dalmatian._transfe
rStandard(address,address,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Dalmatian.sol#1107) is too similar to Dalmatian.
_getTValues(uint256).tTransferAmount (Dalmatian.sol#1098)
Variable Dalmatian._transferToExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1296) is too similar to Dalmatian._getTVal
ues(uint256).tTransferAmount (Dalmatian.sol#1098)
Variable Dalmatian._transferFromExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1307) is too similar to Dalmatian._trans
ferStandard(address,address,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian.reflectionFromToken(uint256,bool).rTransferAmount (Dalmatian.sol#1002) is too similar to Dalmatian._transferStandard(a
ddress,address,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian.reflectionFromToken(uint256,bool).rTransferAmount (Dalmatian.sol#1002) is too similar to Dalmatian._getValues(uint256)
.tTransferAmount (Dalmatian.sol#1089)
Variable Dalmatian._transferFromExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1307) is too similar to Dalmatian._trans
```

```
Variable Dalmatian._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Dalmatian.sol#1107) is too similar to Dalmatian.
_transferStandard(address,address,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian._transferToExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1296) is too similar to Dalmatian._transfe
rStandard(address,address,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Dalmatian.sol#1107) is too similar to Dalmatian.
_getTValues(uint256).tTransferAmount (Dalmatian.sol#1098)
Variable Dalmatian._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Dalmatian.sol#1107) is too similar to Dalmatian.
_getTValues(uint256).tTransferAmount (Dalmatian.sol#1098)
Variable Dalmatian._transferToExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1296) is too similar to Dalmatian._getTVal
ues(uint256).tTransferAmount (Dalmatian.sol#1098)
Variable Dalmatian._transferFromExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1307) is too similar to Dalmatian._trans
ferStandard(address,address,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian.reflectionFromToken(uint256,bool).rTransferAmount (Dalmatian.sol#1002) is too similar to Dalmatian._transferStandard(a
ddress,address,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian.reflectionFromToken(uint256,bool).rTransferAmount (Dalmatian.sol#1002) is too similar to Dalmatian._getValues(uint256)
.tTransferAmount (Dalmatian.sol#1089)
Variable Dalmatian._transferFromExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1307) is too similar to Dalmatian._trans
ferBothExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1040)
Variable Dalmatian.reflectionFromToken(uint256,bool).rTransferAmount (Dalmatian.sol#1002) is too similar to Dalmatian._transferToExcluded
(address,address,uint256).tTransferAmount (Dalmatian.sol#1296)
Variable Dalmatian.reflectionFromToken(uint256,bool).rTransferAmount (Dalmatian.sol#1002) is too similar to Dalmatian._transferBothExclud
ed(address,address,uint256).tTransferAmount (Dalmatian.sol#1040)
Variable Dalmatian._transferBothExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1040) is too similar to Dalmatian._getVa
lues(uint256).tTransferAmount (Dalmatian.sol#1089)
Variable Dalmatian._transferFromExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1307) is too similar to Dalmatian._getVa
lues(uint256).tTransferAmount (Dalmatian.sol#1089)
Variable Dalmatian._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Dalmatian.sol#1107) is too similar to Dalmatian.
_getValues(uint256).tTransferAmount (Dalmatian.sol#1089)
Variable Dalmatian._transferToExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1296) is too similar to Dalmatian._getValu
es(uint256).tTransferAmount (Dalmatian.sol#1089)
Variable Dalmatian._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Dalmatian.sol#1107) is too similar to Dalmatian.
_transferFromExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1307)
Variable Dalmatian._transferToExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1296) is too similar to Dalmatian._transfe
rFromExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1307)
Variable Dalmatian._getValues(uint256).rTransferAmount (Dalmatian.sol#1090) is too similar to Dalmatian._transferBothExcluded(address,add
ress,uint256).tTransferAmount (Dalmatian.sol#1040)
Variable Dalmatian._getValues(uint256).rTransferAmount (Dalmatian.sol#1090) is too similar to Dalmatian._transferStandard(address,address
,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian._getValues(uint256).rTransferAmount (Dalmatian.sol#1090) is too similar to Dalmatian._getValues(uint256).tTransferAmou
nt (Dalmatian.sol#1089)
```

```
Variable Dalmatian._transferToExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1296) is too similar to Dalmatian._transfe
rToExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1296)
Variable Dalmatian._transferBothExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1040) is too similar to Dalmatian._trans
ferStandard(address,address,uint256).tTransferAmount (Dalmatian.sol#1286)
Variable Dalmatian._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Dalmatian.sol#1107) is too similar to Dalmatian.
_transferBothExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1040)
Variable Dalmatian._transferToExcluded(address,address,uint256).rTransferAmount (Dalmatian.sol#1296) is too similar to Dalmatian._transfe
rBothExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1040)
Variable Dalmatian._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Dalmatian.sol#1107) is too similar to Dalmatian.
_transferToExcluded(address,address,uint256).tTransferAmount (Dalmatian.sol#1296)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
Dalmatian.setMaxTxPercent(uint256) (Dalmatian.sol#1318-1320) uses literals with too many digits:
        - _maxTxAmount = _tTotal.mul(maxTxPercent).div(100000) (Dalmatian.sol#1319)
Dalmatian.swapAndLiquify(address,address) (Dalmatian.sol#1403-1443) uses literals with too many digits:
        - feesPercent = totalCollectedFees.mul(1000000).div(balanceOf(address(this))) (Dalmatian.sol#1425)
Dalmatian.swapAndLiquify(address,address) (Dalmatian.sol#1403-1443) uses literals with too many digits:
        - swapTokens = contractTokenBalance.mul(feesPercent).mul(5).div(9).div(1000000) (Dalmatian.sol#1430)
Dalmatian.swapAndLiquify(address,address) (Dalmatian.sol#1403-1443) uses literals with too many digits:
        - burnTokens = contractTokenBalance.mul(feesPercent).mul(4).div(9).div(1000000) (Dalmatian.sol#1433)
Dalmatian.slitherConstructorVariables() (Dalmatian.sol#833-1522) uses literals with too many digits:
        - _tTotal = 1000000000 * 10 ** 6 * 10 ** 9 (Dalmatian.sol#851)
Dalmatian.slitherConstructorVariables() (Dalmatian.sol#833-1522) uses literals with too many digits:
        - minTokenNumberToSell = _tTotal.mul(1).div(100000) (Dalmatian.sol#877)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Dalmatian._decimals (Dalmatian.sol#881) should be constant
Dalmatian._name (Dalmatian.sol#879) should be constant
Dalmatian._symbol (Dalmatian.sol#880) should be constant
Dalmatian._tTotal (Dalmatian.sol#851) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Dalmatian.sol#433-436)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Dalmatian.sol#442-446)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (Dalmatian.sol#448-450)
```

```
        - Ownable.transferOwnership(address) (Dalmatian.sol#442-446)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (Dalmatian.sol#448-450)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (Dalmatian.sol#453-458)
unlock() should be declared external:
        - Ownable.unlock() (Dalmatian.sol#461-466)
calculateBNBReward(uint256,uint256,uint256) should be declared external:
        - Utils.calculateBNBReward(uint256,uint256,uint256) (Dalmatian.sol#679-692)
calculateTopUpClaim(uint256,uint256,uint256,uint256) should be declared external:
        - Utils.calculateTopUpClaim(uint256,uint256,uint256,uint256) (Dalmatian.sol#694-718)
swapTokensForEth(address,uint256) should be declared external:
        - Utils.swapTokensForEth(address,uint256) (Dalmatian.sol#720-739)
swapETHForTokens(address,address,uint256) should be declared external:
        - Utils.swapETHForTokens(address,address,uint256) (Dalmatian.sol#741-760)
addLiquidity(address,address,uint256,uint256) should be declared external:
        - Utils.addLiquidity(address,address,uint256,uint256) (Dalmatian.sol#762-779)
name() should be declared external:
        - Dalmatian.name() (Dalmatian.sol#925-927)
symbol() should be declared external:
        - Dalmatian.symbol() (Dalmatian.sol#929-931)
decimals() should be declared external:
        - Dalmatian.decimals() (Dalmatian.sol#933-935)
totalSupply() should be declared external:
        - Dalmatian.totalSupply() (Dalmatian.sol#937-939)
transfer(address,uint256) should be declared external:
        - Dalmatian.transfer(address,uint256) (Dalmatian.sol#946-949)
allowance(address,address) should be declared external:
        - Dalmatian.allowance(address,address) (Dalmatian.sol#951-953)
approve(address,uint256) should be declared external:
        - Dalmatian.approve(address,uint256) (Dalmatian.sol#955-958)
transferFrom(address,address,uint256) should be declared external:
        - Dalmatian.transferFrom(address,address,uint256) (Dalmatian.sol#960-964)
increaseAllowance(address,uint256) should be declared external:
        - Dalmatian.increaseAllowance(address,uint256) (Dalmatian.sol#966-969)
decreaseAllowance(address,uint256) should be declared external:
        - Dalmatian.decreaseAllowance(address,uint256) (Dalmatian.sol#971-974)
isExcludedFromFee(address) should be declared external:
        - Dalmatian.isExcludedFromFee(address) (Dalmatian.sol#1200-1202)
claimBNBReward() should be declared external:
        - Dalmatian.claimBNBReward() (Dalmatian.sol#1349-1366)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Dalmatian.sol analyzed (12 contracts with 75 detectors), 144 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above, according to their level of severity.

SOLIDITY STATIC ANALYSIS

SOLIDITY STATIC ANALYSIS

last results for:
contracts/Dalmatian.sol

Security

Transaction origin:

INTERNAL ERROR in module Transaction origin: can't convert undefined to object
Pos: not available

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: can't convert undefined to object
Pos: not available

Inline assembly:

INTERNAL ERROR in module Inline assembly: can't convert undefined to object
Pos: not available

Block timestamp:

INTERNAL ERROR in module Block timestamp: can't convert undefined to object
Pos: not available

Low level calls:

INTERNAL ERROR in module Low level calls: can't convert undefined to object
Pos: not available

Selfdestruct:

INTERNAL ERROR in module Selfdestruct: can't convert undefined to object
Pos: not available

Gas & Economy

This on local calls:

INTERNAL ERROR in module This on local calls: can't convert undefined to object
Pos: not available

Delete dynamic array:

INTERNAL ERROR in module Delete dynamic array: can't convert undefined to object
Pos: not available

For loop over dynamic array:

INTERNAL ERROR in module For loop over dynamic array: can't convert undefined to object
Pos: not available

Ether transfer in loop:

INTERNAL ERROR in module Ether transfer in loop: can't convert undefined to object
Pos: not available

ERC

ERC20:

INTERNAL ERROR in module ERC20: can't convert undefined to object
Pos: not available

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: can't convert undefined to object
Pos: not available

Similar variable names:

INTERNAL ERROR in module Similar variable names: can't convert undefined to object
Pos: not available

No return:

INTERNAL ERROR in module No return: can't convert undefined to object
Pos: not available

Guard conditions:

INTERNAL ERROR in module Guard conditions: can't convert undefined to object
Pos: not available

String length:

INTERNAL ERROR in module String length: can't convert undefined to object
Pos: not available

SOLHINT LINTER

```
contracts/Dalmatian.sol:3:1: Error: Compiler version 0.8.4 does not satisfy the r semver requirement

contracts/Dalmatian.sol:405:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)

contracts/Dalmatian.sol:456:21: Error: Avoid making time-based decisions in your business logic

contracts/Dalmatian.sol:463:17: Error: Avoid making time-based decisions in your business logic

contracts/Dalmatian.sol:500:5: Error: Function name must be in mixedCase

contracts/Dalmatian.sol:501:5: Error: Function name must be in mixedCase

contracts/Dalmatian.sol:518:5: Error: Function name must be in mixedCase

contracts/Dalmatian.sol:538:5: Error: Function name must be in mixedCase

contracts/Dalmatian.sol:673:1: Error: Compiler version >=0.6.8 does not satisfy the r semver requirement

contracts/Dalmatian.sol:701:20: Error: Avoid to make time-based decisions in your business logic

contracts/Dalmatian.sol:737:13: Error: Avoid to make time-based decisions in your business logic

contracts/Dalmatian.sol:758:13: Error: Avoid to make time-based decisions in your business logic

contracts/Dalmatian.sol:777:13: Error: Avoid to make time-based decisions in your business logic

contracts/Dalmatian.sol:800:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)

contracts/Dalmatian.sol:826:17: Error: Avoid to use tx.origin

contracts/Dalmatian.sol:833:1: Error: Contract has 32 states declarations but allowed no more than 15
```

```
contracts/Dalmatian.sol:894:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)

contracts/Dalmatian.sol:1080:32: Error: Code contains empty blocks

contracts/Dalmatian.sol:1148:23: Error: Avoid to use tx.origin

contracts/Dalmatian.sol:1159:23: Error: Avoid to use tx.origin

contracts/Dalmatian.sol:1344:13: Error: Avoid to make time-based decisions in your business
logic

contracts/Dalmatian.sol:1349:52: Error: Visibility modifier must be first in list of modifiers

contracts/Dalmatian.sol:1350:55: Error: Avoid to make time-based decisions in your business
logic

contracts/Dalmatian.sol:1350:72: Error: Use double quotes for string literals

contracts/Dalmatian.sol:1351:44: Error: Use double quotes for string literals

contracts/Dalmatian.sol:1356:32: Error: Avoid to use low level calls.

contracts/Dalmatian.sol:1357:31: Error: Use double quotes for string literals

contracts/Dalmatian.sol:1361:46: Error: Avoid to make time-based decisions in your business
logic

contracts/Dalmatian.sol:1364:24: Error: Avoid using low level calls.

contracts/Dalmatian.sol:1365:23: Error: Use double quotes for string literals

contracts/Dalmatian.sol:1390:50: Error: Avoid to make time-based decisions in your business
logic

contracts/Dalmatian.sol:1449:9: Error: Variable name must be in mixedCase

contracts/Dalmatian.sol:1478:33: Error: Avoid to make time-based decisions in your business
logic

contracts/Dalmatian.sol:1484:41: Error: Avoid to make time-based decisions in your business
logic
```

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Dalmatian-INU platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Dalmatian-INU Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract, but relying on other contracts might cause Reentrancy Vulnerability.

The majority of the concerns addressed above have been acknowledged, implemented and verified.

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com