QuillAudits

Audit Report
August, 2021

GARFIELD

Contents

Scope of Audit	01
Techniques and Methods	02
Issue Categories	03
Issues Found – Code Review/Manual Testing	04
Automated Testing	26
Disclaimer	42
Summary	43

Scope of Audit

The scope of this audit was to analyze and document the Garfield Token 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	1	2	0	2
Closed	3	4	6	12

Introduction

During the period of **August 03, 2021 to August 04, 2021** - QuillAudits Team performed a security audit for Garfield smart contracts.

The code for the audit was taken from following the official link:

Note	Date	Commit hash
Version 1	August 03	https://bscscan.com/address/0x215afef32923bd3240fa5f9d9ff5d8082fe4bb09#code
Version 2	August 07	https://github.com/AndreaGarfield/GAR-SC-EDIT-/ **Commit ID:** e7d11e2a6a90165ec192186901267e1db9eccb07

Issues Found – Code Review / Manual Testing

High severity issues

1. Tax is applied when more than 0.01% of the total supply is sent

Line	Code

1393	function disruptiveTransfer(address recipient, uint256 amount) public payable returns (bool) {
	_transfer(_msgSender(), recipient, amount, msg.value);
	return true;
	}

Description

According to Anti Pump-Dump-Exit Whales agreements on the garfield.finance, transactions (sell/buy/transfer) that trade more than 0.01% of the total supply will be rejected. If whales want to make a transfer (between 2 wallets) that is larger than 0.01% of the total supply, they can use the Disruptive Transfer feature: the transfer will be charged for 2 BNB (taxed 10%).

This statement was tested and failed during the functional test. As a result, 10% of the tax was applied for the receiver when the sender sent more than 0.01% of the total of the supply along with 2 BNB.

POC

1. The sender sent more than 0.01% of the total supply, which is 2e17 GAR (the sender is not an owner and 2e17 GAR = 0.02% of the total supply).
2. The receiver received less than 2e17 GAR (~90% of 2e17 GAR).

Expected result

The receiver should have received the same amount of GAR sent by the sender without being taxed.

Remediation

Please consider rewriting this function and adding the sender to _isExcludedFromFee when the amount is > 0.01 of the total supply and msg.value is called.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 9940d96cc1528698533cd2e7db9de2bc4689bffc

2. activateContract() function can be called many times

Description
This function is created to activate the contract after deployment. However, the activateContract() function can reset all the critical settings of the contract and is called many times by the owner.

This indicates that the malicious owner either has control over the function or accidentally calls it. As a result, all settings will be reset, and the contract may be put at risk.

Remediation
We recommend putting all the settings into separate functions where it can be called by the owner or initializing it in the constructor. Otherwise, the Auditee should be aware of the aforementioned risks.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 61a51d3b7753c86511d0963b78ac3cd5da72011a

3. time variable is not validated

Description
The _lockTime is set by the owner via the lock() function, where _lockTime = now + time;. According to the condition set in the require() within unlock() function, the contract should be locked until 7 days.

L477: require(now > _lockTime , "Contract is locked until 7 days");

Unfortunately, the time variable can be set to an arbitrary number by the owner, which helps the _lockTime variable bypass the condition in the unlock() function.

Remediation
We recommend adding a check for the time parameter, where the time should be validated corresponding to the condition of the unlock() function.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit f5b0e4d88928ab05c42ed764bb09dec3d7a71fc5

4. Weak PRNG

Line	Code

693	function random(uint256 from, uint256 to, uint256 salty) private view returns (uint256) {
	uint256 seed = uint256(
	keccak256(
	abi.encodePacked(
	block.timestamp + block.difficulty +
	((uint256(keccak256(abi.encodePacked(block.coinbase)))) / (now)) +
	block.gaslimit +
	((uint256(keccak256(abi.encodePacked(msg.sender)))) / (now)) +
	block.number +
	salty
	)
	)
	);
	return seed.mod(to - from) + from;
	}

Description
Weak PRNG due to a modulo on block.timestamp, now, block.difficulty, block.number and block.gaslimit. These can be influenced by miners to some extent so they should be avoided.

Furthermore, the to and from parameters are hard-coded and the salty parameter is set to the balance of the msg.sender.

Remediation
The Auditee should write smart contracts with the notion that block values are not precise, and the use of them can lead to unexpected effects. Alternatively, the Auditee may make use of oracles.

References

- Safety: Timestamp dependence
- Ethereum Smart Contract Best Practices - Timestamp Dependence
- How do Ethereum mining nodes maintain a time consistent with the network?
- Solidity: Timestamp dependency, is it possible to do safely?
- Avoid using block.number as a timestamp

Status: Acknowledged by the Auditee
Even though the fix does not make use of Oracle's solution, the Garfield team made some fixes based on our recommendations by removing block and transaction properties. We address below the fixes introduced up to commit b0347563f3a41bab060921709251a33dd69201d9

Medium severity issues

5. Missing Check for Reentrancy Attack

Description
Calling GARFIELD.transfer() and GARFIELD.transferFrom() might trigger function pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens() and pancakeRouter.addLiquidityETH(), which is implemented by third party at pancakeRouter , in Utils.swapTokensForEth() and Utils.addLiquidity() . If there are vulnerable external calls in pancakeRouter , reentrancy attacks could be conducted because these two functions have state updates and event emits after external calls.

The scope of the audit would treat the third-party implementation at pancakeRouter as a black box and assume its functional correctness. However, third parties may be compromised in the real world that leads to assets being lost or stolen.

Remediation
We recommend applying OpenZeppelin ReentrancyGuard library - nonReentrant modifier for the aforementioned functions to prevent reentrancy attacks.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 5fd35320f0fbb22035ba61204afb857ec761c3f2

6. Centralization Risks

Description

The role owner has the authority to update the critical settings:

- setTaxFeePercent
- setSwapAndLiquifyEnabled
- setMaxTxPercent
- setLiquidityFeePercent
- setExcludeFromMaxTx
- excludeFromFee
- includeInFee
- includeInReward
- excludeFromReward

Remediation

We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

- with reasonable latency for community awareness on privileged operations;
- Multisig with community-voted 3rd-party independent co-signers;
- DAO or Governance module increasing transparency and community involvement;
- Setting _isExcludedFromMaxTx[owner()] = False

Status: Acknowledged by the Auditee

7. Inappropriate Variable Initialization

Line	Code

1311	function setMaxTxPercent(uint256 maxTxPercent) public onlyOwner() {
	_maxTxAmount = _tTotal.mul(maxTxPercent).div(10000);
	}

Description
_maxTxAmount should be 0.05% of the total supply according to the code comment.

But the _maxTxPercent can be set by the owner by calling setMaxTxPercent function at line 1311

Remediation
We recommend setting the _maxTxPercent as a constant value or consider changing the business logic mentioned in the code comment.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 0f86a14e594a2dc2e54ad7be71b11e46fdf6fd04. Nevertheless, the better way is to make the variable constant.

8. Lack of event emissions

Line	Code
1096	function excludeFromFee(address account) public onlyOwner { _isExcludedFromFee[account] = true; }
1100	function includeInFee(address account) public onlyOwner { _isExcludedFromFee[account] = false; }
1104	function setTaxFeePercent(uint256 taxFee) external onlyOwner() { _taxFee = taxFee; }
1108	function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() { _liquidityFee = liquidityFee; }
1311	function setMaxTxPercent(uint256 maxTxPercent) public onlyOwner() { _maxTxAmount = _tTotal.mul(maxTxPercent).div(10000); }
1315	function setExcludeFromMaxTx(address _address, bool value) public onlyOwner { _isExcludedFromMaxTx[_address] = value; }

Description
The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:

- setLiquidityFeePercent
- setTaxFeePercent
- includeInFee
- excludeFromFee
- setMaxTxPercent
- setExcludeFromMaxTx

Remediation
We recommend emitting an event to log the update of the following variables:

- _maxTxAmount
- _isExcludedFromMaxTx
- _liquidityFee
- _taxFee
- _isExcludedFromFee

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit a970422b652cc9bc66ddbfd54741e26428b4b42f

9. A maximum value of GAR is allowed for the pancakeRouter

Description
A maximum value of GAR owned by the GARFIELD contract is allowed for the pancakeRouter in the activateContract() function.

L1466: _approve(address(this), address(pancakeRouter), 2 ** 256 - 1);

The scope of the audit would treat the third-party implementation at pancakeRouter as a black box and assume its functional correctness. However, third parties may be compromised in the real world that leads to assets being lost or stolen.

Remediation
The Auditee should be aware of the aforementioned risk and ensure that the pancakeRouter is a trusted party during the deployment.

Status: Acknowledged by the Auditee

10. Loops in the Contract are extremely costly

Description
The for loops in the entire codebase includes state variables .length of a non-memory array, in the condition of the for loops. As a result, these state variables consume a lot more extra gas for every iteration of the for loop.

Remediation
We recommend using a local variable instead of a state variable .length in a loop for the entire codebase.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit f5b0e4d88928ab05c42ed764bb09dec3d7a71fc5

Low level severity issues

11. Tautology or Contradiction Issue

Line	Code
1340	require(balanceOf(msg.sender) >= 0, 'Error: must own MRAT to claim reward');

Description
balanceOf(msg.sender) >= 0 is true even if the balance of the msg.sender is 0.
This comparison is a tautology that will waste gas during the execution

Remediation
Fix the incorrect comparison by changing the value type or the comparison.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 7bfdc0fc6ff8f4027d6fc83c4682d0107b46d302

12. Redundant Code

Line	Code

1249	else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
	_transferStandard(sender, recipient, amount);
	}

723	uint256 bnbPool = currentBNBPool;

Description
When the contract enters the branch else if (!_isExcluded[sender] && !_isExcluded[recipient]) or else, the contract will execute the same piece of code _transferStandard(sender, recipient, amount);

The currentBNBPool can be used directly instead of having the bnbPool variable initialized.

Remediation
We recommend removing the following code:\

```
else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
        _transferStandard(sender, recipient, amount);
    }
```

And using the currentBNBPool instead of bnbPool.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 7ddb20dc1ff3084488832b68c31a8a86a966fae2

13. Missing Range Check for Input Variable

Line	Code
1104	function setTaxFeePercent(uint256 taxFee) external onlyOwner() { _taxFee = taxFee; }
1108	function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() { _liquidityFee = liquidityFee; }
1311	function setMaxTxPercent(uint256 maxTxPercent) public onlyOwner() { _maxTxAmount = _tTotal.mul(maxTxPercent).div(10000); }

Description

The role can set the following state variables arbitrary large or small causing potential risks in fees and anti whale :

- _taxFee
- _liquidityFee
- _maxTxAmount

Remediation

We recommend setting ranges and check the following input variables:

- taxFee
- liquidityFee
- maxTxPercent

Status: Closed

The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 196d46bc7e4cfce93a162dc1970417d1c11eb891

14. State Variable Default Visibility

Description

There is an instance in the codebase where the require statement has ambiguous or imprecise error messages.

L1343: require(balanceOf(msg.sender) >= 0, 'Error: must own MRAT to claim reward');

Uninformative error messages greatly damage the overall user experience, thus lowering the system's quality. Consider not only fixing the specific instance mentioned above, but also reviewing the entire codebase to make sure every error message is informative and user-friendly.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 7bfdc0fc6ff8f4027d6fc83c4682d0107b46d302

15. Redundant parameters in calculateBNBReward() function

Description
It was discovered that the _tTotal and ofAddress variables are not used in calculateBNBReward() function.
Unused code is allowed in Solidity, and they do not pose a direct security issue. It is best practice,though, to avoid them as they can:

- cause an increase in computations (and unnecessary gas consumption)
- indicate bugs or malformed data structures, and they are generally a sign of poor code quality
- cause code noise and decrease the readability of the code

Remediation
We recommend removing all unused above-mentioned parameters used in the calculateBNBReward() function.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit a3cbe787da2da67eb2895c79e75f8d25a2d7436a

16. Comparison to boolean constants

Line	Code

1384-1385	_isExcludedFromMaxTx[from] == false && // default will be false
	_isExcludedFromMaxTx[to] == false // default will be false

Description
Boolean constants of the _isExcludedFromMaxTx[from] and _isExcludedFromMaxTx[to] can be used directly and do not need to be compared to true or false.

Remediation
We recommend removing the equality to the boolean constant.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 4c9bfa4e6c11925718f3a69ede4b5f31d0744ffd

Informational

17. Typo

Line	Code
936	uint256 tokensIntoLiqudity

Description
There is a typo in tokensIntoLiqudity.

Remediation
We recommend correcting and changing tokensIntoLiqudity to tokensIntoLiquidity.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit eb033742c13cdf79a6db887757343333abc83804

18. Inappropriate Location of Constant Declaration

Description
A series of constants are declared in the middle of the contract..

Remediation
We recommend declaring constants at the beginning of the contract.

Status: Closed
The Garfield team made some fixes based on our recommendations

19. Redundant Setting

Line	Code

1291,1292, 1295, 1300	rewardCycleBlock = 7 days;
	easyRewardCycleBlock = 1 days;
	disruptiveCoverageFee = 2 ether;
	winningDoubleRewardPercentage = 5;

Description
Variables have been set to appropriate values in the declaration, so there is no need to set them again.

Remediation
We recommend removing redundant variable settings to save gas.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 984c12fa671fdd230a8ed69362249e4fe0a4a09a

20. Incorrect versions of Solidity

Solidity version used: 0.6.8.

Description
solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statements.

Remediation
Deploy with any of the following Solidity versions:

- 0.5.16 - 0.5.17
- 0.6.11 - 0.6.12
- 0.7.5 - 0.7.6

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

Status: Closed
The Garfield team made some fixes based on our recommendations.

21. Incorrect versions of Solidity

Description

It was discovered that the following code are commented in this contract:
L918: // uint256 private _tTotal = 100000000000000000000000;
L1063: // require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Pancake router.');

Unused code is allowed in Solidity, and they do not pose a direct security issue. It is best practice, though, to avoid them as they can:

- cause an increase in computations (and unnecessary gas consumption)
- indicate bugs or malformed data structures, and they are generally a sign of poor code quality
- cause code noise and decrease the readability of the code

Remediation

We recommend removing all unused variables/code from the codebase.

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 74a3184cc030cab1a0bfdc461d7d892e8c2bfc87

22. Solidity Private Modifier Do Not Hide Data

Description

ITo favor explicitness, consider everything that is inside a contract is visible to all observers external to the blockchain. Therefore, the Auditee should be aware that making something private only prevents other contracts from reading or modifying the information, but it will still be visible to the whole world outside of the blockchain.

For example, the following private variables in GARFIELD contract are still visible and accessible:

- uint256 private constant MAX = ~uint256(0);
- uint256 private _tTotal = 100000000000 * 10 ** 9;
- uint256 private _rTotal = (MAX - (MAX % _tTotal));
- uint256 private _tFeeTotal

Status: Acknowledged by the Auditee

23. State variables that could be declared constant

Description

Adding the constant attributes to state variables that never change. Therefore, the following constant state variables should be declared constant to save gas:

- rewardThreshold
- threshHoldTopUpRate

Status: Closed

The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit b895cf05fa940c314b366ad7d0691ae16e45aebc

24. Public function that could be declared external

Description

Using the external attribute for functions never called from the contract. Therefore, the following public functions that are never called by the contract should be declared external to save gas:

- deliver()
- excludeFromReward()
- excludeFromFee()
- includeInFee()
- setExcludeFromMaxTx()
- claimBNBReward()
- disruptiveTransfer()
- activateContract()

Status: Closed
The Garfield team made some fixes based on our recommendations. We address below the fixes introduced up to commit 32b32117beb980ca3d3b4e2f813790b2c62eb47c

25. State Variable Default Visibility

Line	Code
1309	uint256 minTokenNumberToSell

Description
The Visibility of the above-mentioned variable is not defined. Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

The default is internal for state variables, but it should be made explicit.

Remediation
We recommend adding .visibility for these variables. Variables can be specified as being public, internal, or private. Explicitly define visibility for all state variables.

Status: Closed
The Garfield team made some fixes based on our recommendations.

26. Use double quotes for string literals

Line	Code
1342-1343	require(nextAvailableClaimDate[msg.sender] <= block.timestamp, 'Error: next available not reached'); require(balanceOf(msg.sender) >= 0, 'Error: must own MRAT to claim reward');
1362	require(sent, 'Error: Cannot withdraw reward');

Description
Single quote found in the above string variables. Whilst, the double quotes are being utilized for other string literals.

Remediation
We recommend using double quotes for string literals.

Status: Closed
The Garfield team made some fixes based on our recommendations.

27. Variables declared as uint instead of uint256

Description
To favor explicitness, consider changing all instances of uint into uint256 in the entire codebase.

Status: Closed
The Garfield team made some fixes based on our recommendations.

28. Inconsistent coding style

Description
Deviations from the Solidity Style Guide were identified throughout the entire codebase. Taking into consideration how much value a consistent coding style adds to the project's readability, enforcing a standard coding style with the help of linter tools such as Solhint is recommended.

Status: Closed
The Garfield team made some fixes based on our recommendations.

29. block.timestamp may not be reliable

Description
The Time contract uses the block.timestamp as part of the calculations and time checks.
Nevertheless, timestamps can be slightly altered by miners to favor them in contracts that have logics that depend strongly on them.
Consider taking into account this issue and warning the users that such a scenario could happen.

Status: Acknowledged by the Auditee

30. Missing docstrings

Description

It is extremely difficult to locate any contracts or functions, as they lack documentation. One consequence of this is that reviewers' understanding of the code's intention is impeded, which is significant because it is necessary to accurately determine both security and correctness.

They are additionally more readable and easier to maintain when wrapped in docstrings. The functions should be documented so that users can understand the purpose or intention of each function, as well as the situations in which it may fail, who is allowed to call it, what values it returns, and what events it emits.

Status: Closed

The Garfield team made some fixes based on our recommendations.

Functional test

Function Names	Testing results
name()	Passed
symbol()	Passed
decimals()	Passed
totalSupply()	Passed
balanceOf()	Passed
transfer()	Passed
approve()	Passed
allowance()	Passed
transferFrom()	Passed
increaseAllowance()	Passed
decreaseAllowance()	Passed
totalFees()	Passed
deliver()	Passed
tokenFromReflection()	Passed
reflectionFromToken()	Passed
disruptiveTransfer()	Passed
activateContract()	Passed
excludeFromReward()	Passed
includeInReward()	Passed
isExcludedFromReward()	Passed

Function Names	Testing results
setTaxFeePercent()	Passed
setLiquidityFeePercent()	Passed
setSwapAndLiquifyEnabled()	Passed
isExcludedFromFee()	Passed
setMaxTxPercent()	Passed
setExcludeFromMaxTx()	Passed
calculateBNBReward()	Passed
getRewardCycleBlock()	Passed
claimBNBReward()	Passed
lock()	Passed
unlocked()	Passed

Automated Testing

Slither

```
INFO:Detectors:
Utils.swapETHForTokens(address,address,uint256) (GARFIELD.sol#786-805) sends eth to
arbitrary user
    Dangerous calls:
    - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
ethAmount}(0,path,address(recipient),block.timestamp + 360) (GARFIELD.sol#799-804)
Utils.addLiquidity(address,address,uint256,uint256) (GARFIELD.sol#807-824) sends eth to
arbitrary user
    Dangerous calls:
    - pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-
send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in GARFIELD._transfer(address,address,uint256,uint256)
(GARFIELD.sol#1210-1235):
    External calls:
    - swapAndLiquify(from,to) (GARFIELD.sol#1223)
        - pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
        -
pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,pa
th,address(this),block.timestamp) (GARFIELD.sol#777-783)
        - Utils.swapTokensForEth(address(pancakeRouter),tokenAmountToBeSwapped)
(GARFIELD.sol#1432)
        -
Utils.addLiquidity(address(pancakeRouter),owner(),otherPiece,bnbToBeAddedToLiquidity)
(GARFIELD.sol#1445)
    External calls sending eth:
    - swapAndLiquify(from,to) (GARFIELD.sol#1223)
        - pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
    State variables written after the call(s):
    - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
        - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (GARFIELD.sol#1166)
        - _rOwned[sender] = _rOwned[sender].sub(rAmount) (GARFIELD.sol#1272)
        - _rOwned[sender] = _rOwned[sender].sub(rAmount) (GARFIELD.sol#1263)
        - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (GARFIELD.sol#1264)
        - _rOwned[sender] = _rOwned[sender].sub(rAmount) (GARFIELD.sol#1088)
        - _rOwned[sender] = _rOwned[sender].sub(rAmount) (GARFIELD.sol#1283)
```

```
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (GARFIELD.sol#1284)
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (GARFIELD.sol#1274)
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (GARFIELD.sol#1090)
- _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
    - _rTotal = _rTotal.sub(rFee) (GARFIELD.sol#1121)
- _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
    - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (GARFIELD.sol#1168)
    - _tOwned[sender] = _tOwned[sender].sub(tAmount) (GARFIELD.sol#1087)
    - _tOwned[sender] = _tOwned[sender].sub(tAmount) (GARFIELD.sol#1282)
    - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (GARFIELD.sol#1273)
    - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (GARFIELD.sol#1089)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities
INFO:Detectors:
Reentrancy in GARFIELD.claimBNBReward() (GARFIELD.sol#1341-1363):
    External calls:
    -
Utils.swapETHForTokens(address(pancakeRouter),address(0x0000000000000000000000
0000000000000dEaD),reward.div(5)) (GARFIELD.sol#1349-1353)
    State variables written after the call(s):
    - nextAvailableClaimDate[msg.sender] = block.timestamp + getRewardCycleBlock()
(GARFIELD.sol#1358)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1
INFO:Detectors:
GARFIELD.claimBNBReward() (GARFIELD.sol#1341-1363) contains a tautology or
contradiction:
    - require(bool,string)(balanceOf(msg.sender) >= 0,Error: must own MRAT to claim reward)
(GARFIELD.sol#1343)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction
INFO:Detectors:
Utils.addLiquidity(address,address,uint256,uint256) (GARFIELD.sol#807-824) ignores return
value by pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
GARFIELD.allowance(address,address).owner (GARFIELD.sol#1003) shadows:
    - Ownable.owner() (GARFIELD.sol#428-430) (function)
GARFIELD._approve(address,address,uint256).owner (GARFIELD.sol#1202) shadows:
    - Ownable.owner() (GARFIELD.sol#428-430) (function)
GARFIELD.calculateBNBReward(address).totalSupply (GARFIELD.sol#1320-1323) shadows:
    - GARFIELD.totalSupply() (GARFIELD.sol#989-991) (function)
    - IBEP20.totalSupply() (GARFIELD.sol#21) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-
shadowing
```

```
INFO:Detectors:
Reentrancy in GARFIELD._transfer(address,address,uint256,uint256)
(GARFIELD.sol#1210-1235):
    External calls:
    - swapAndLiquify(from,to) (GARFIELD.sol#1223)
        - pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
        -
pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,pa
th,address(this),block.timestamp) (GARFIELD.sol#777-783)
        - Utils.swapTokensForEth(address(pancakeRouter),tokenAmountToBeSwapped)
(GARFIELD.sol#1432)
        -
Utils.addLiquidity(address(pancakeRouter),owner(),otherPiece,bnbToBeAddedToLiquidity)
(GARFIELD.sol#1445)
    External calls sending eth:
    - swapAndLiquify(from,to) (GARFIELD.sol#1223)
        - pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
    State variables written after the call(s):
    - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
        - _liquidityFee = _previousLiquidityFee (GARFIELD.sol#1195)
        - _liquidityFee = 0 (GARFIELD.sol#1190)
    - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
        - _previousLiquidityFee = _liquidityFee (GARFIELD.sol#1187)
    - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
        - _previousTaxFee = _taxFee (GARFIELD.sol#1186)
    - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
        - _tFeeTotal = _tFeeTotal.add(tFee) (GARFIELD.sol#1122)
    - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
        - _taxFee = _previousTaxFee (GARFIELD.sol#1194)
        - _taxFee = 0 (GARFIELD.sol#1189)
    - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
        - nextAvailableClaimDate[recipient] = nextAvailableClaimDate[recipient] +
Utils.calculateTopUpClaim(currentRecipientBalance,basedRewardCycleBlock,threshHoldTopU
pRate,amount) (GARFIELD.sol#1369-1374)
Reentrancy in GARFIELD.constructor(address) (GARFIELD.sol#951-975):
    External calls:
    - pancakePair =
IPancakeFactory(_pancakeRouter.factory()).createPair(address(this),_pancakeRouter.WETH()
) (GARFIELD.sol#958-959)
    State variables written after the call(s):
    - _isExcludedFromFee[owner()] = true (GARFIELD.sol#965)
    - _isExcludedFromFee[address(this)] = true (GARFIELD.sol#966)
    - _isExcludedFromMaxTx[owner()] = true (GARFIELD.sol#969)
    - _isExcludedFromMaxTx[address(this)] = true (GARFIELD.sol#970)
```

```
_isExcludedFromMaxTx[address(0x000000000000000000000000000000000000dEaD)
] = true (GARFIELD.sol#971)
    - _isExcludedFromMaxTx[address(0)] = true (GARFIELD.sol#972)
    - pancakeRouter = _pancakeRouter (GARFIELD.sol#962)
Reentrancy in GARFIELD.transferFrom(address,address,uint256) (GARFIELD.sol#1012-1016):
    External calls:
    - _transfer(sender,recipient,amount,0) (GARFIELD.sol#1013)
        - pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
        -
pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,pa
th,address(this),block.timestamp) (GARFIELD.sol#777-783)
        - Utils.swapTokensForEth(address(pancakeRouter),tokenAmountToBeSwapped)
(GARFIELD.sol#1432)
        -
Utils.addLiquidity(address(pancakeRouter),owner(),otherPiece,bnbToBeAddedToLiquidity)
(GARFIELD.sol#1445)
    External calls sending eth:
    - _transfer(sender,recipient,amount,0) (GARFIELD.sol#1013)
        - pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
    State variables written after the call(s):
    - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20:
transfer amount exceeds allowance)) (GARFIELD.sol#1014)
        - _allowances[owner][spender] = amount (GARFIELD.sol#1206)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2
INFO:Detectors:
Reentrancy in GARFIELD._transfer(address,address,uint256,uint256)
(GARFIELD.sol#1210-1235):
    External calls:
    - swapAndLiquify(from,to) (GARFIELD.sol#1223)
        - pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
        -
pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,pa
th,address(this),block.timestamp) (GARFIELD.sol#777-783)
        - Utils.swapTokensForEth(address(pancakeRouter),tokenAmountToBeSwapped)
(GARFIELD.sol#1432)
        -
Utils.addLiquidity(address(pancakeRouter),owner(),otherPiece,bnbToBeAddedToLiquidity)
(GARFIELD.sol#1445)
    External calls sending eth:
    - swapAndLiquify(from,to) (GARFIELD.sol#1223)
        - pancakeRouter.addLiquidityETH{value:
```

```
ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360)
(GARFIELD.sol#816-823)
    Event emitted after the call(s):
    - Transfer(sender,recipient,tTransferAmount) (GARFIELD.sol#1267)
        - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
    - Transfer(sender,recipient,tTransferAmount) (GARFIELD.sol#1277)
        - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
    - Transfer(sender,recipient,tTransferAmount) (GARFIELD.sol#1287)
        - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
    - Transfer(sender,recipient,tTransferAmount) (GARFIELD.sol#1093)
        - _tokenTransfer(from,to,amount,takeFee) (GARFIELD.sol#1234)
Reentrancy in GARFIELD.claimBNBReward() (GARFIELD.sol#1341-1363):
    External calls:
    -
Utils.swapETHForTokens(address(pancakeRouter),address(0x000000000000000000000000
0000000000000dEaD),reward.div(5)) (GARFIELD.sol#1349-1353)
    Event emitted after the call(s):
    - ClaimBNBSuccessfully(msg.sender,reward,nextAvailableClaimDate[msg.sender])
(GARFIELD.sol#1359)
Reentrancy in GARFIELD.constructor(address) (GARFIELD.sol#951-975):
    External calls:
    - pancakePair =
IPancakeFactory(_pancakeRouter.factory()).createPair(address(this),_pancakeRouter.WETH()
) (GARFIELD.sol#958-959)
    Event emitted after the call(s):
    - Transfer(address(0),_msgSender(),_tTotal) (GARFIELD.sol#974)
Reentrancy in GARFIELD.swapAndLiquify(address,address) (GARFIELD.sol#1398-1449):
    External calls:
    - Utils.swapTokensForEth(address(pancakeRouter),tokenAmountToBeSwapped)
(GARFIELD.sol#1432)
    -
Utils.addLiquidity(address(pancakeRouter),owner(),otherPiece,bnbToBeAddedToLiquidity)
(GARFIELD.sol#1445)
    Event emitted after the call(s):
    - SwapAndLiquify(piece,deltaBalance,otherPiece) (GARFIELD.sol#1447)
Reentrancy in GARFIELD.transferFrom(address,address,uint256) (GARFIELD.sol#1012-1016):
    External calls:
    - _transfer(sender,recipient,amount,0) (GARFIELD.sol#1013)
        - pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,pa
th,address(this),block.timestamp) (GARFIELD.sol#777-783)
Utils.swapTokensForEth(address(pancakeRouter),tokenAmountToBeSwapped)
(GARFIELD.sol#1432)
        -
```

```
Utils.addLiquidity(address(pancakeRouter),owner(),otherPiece,bnbToBeAddedToLiquidity)
(GARFIELD.sol#1445)
    External calls sending eth:
    - _transfer(sender,recipient,amount,0) (GARFIELD.sol#1013)
        - pancakeRouter.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (GARFIELD.sol#816-823)
    Event emitted after the call(s):
    - Approval(owner,spender,amount) (GARFIELD.sol#1207)
        - _approve(sender,_msgSender(),_allowances[sender]
[_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance))
(GARFIELD.sol#1014)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (GARFIELD.sol#475-480) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(now > _lockTime,Contract is locked until 7 days) (GARFIELD.sol#477)
Utils.isLotteryWon(uint256,uint256) (GARFIELD.sol#709-713) uses timestamp for comparisons
    Dangerous comparisons:
    - luckyNumber <= winPercentage (GARFIELD.sol#712)
GARFIELD.getRewardCycleBlock() (GARFIELD.sol#1336-1339) uses timestamp for
comparisons
    Dangerous comparisons:
    - block.timestamp >= disableEasyRewardFrom (GARFIELD.sol#1337)
GARFIELD.claimBNBReward() (GARFIELD.sol#1341-1363) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(nextAvailableClaimDate[msg.sender] <= block.timestamp,Error: next
available not reached) (GARFIELD.sol#1342)
GARFIELD.ensureMaxTxAmount(address,address,uint256,uint256) (GARFIELD.sol#1377-1391)
uses timestamp for comparisons
    Dangerous comparisons:
    - value < disruptiveCoverageFee && block.timestamp >= disruptiveTransferEnabledFrom
(GARFIELD.sol#1387)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (GARFIELD.sol#280-289) uses assembly
    - INLINE ASM (GARFIELD.sol#287)
Address._functionCallWithValue(address,bytes,uint256,string) (GARFIELD.sol#373-394) uses
assembly
    - INLINE ASM (GARFIELD.sol#386-389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
GARFIELD.ensureMaxTxAmount(address,address,uint256,uint256) (GARFIELD.sol#1377-1391)
compares to a boolean constant:
    -_isExcludedFromMaxTx[from] == false && _isExcludedFromMaxTx[to] == false
(GARFIELD.sol#1384-1385)
```

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (GARFIELD.sol#373-394) is never used and should be removed
Address.functionCall(address,bytes) (GARFIELD.sol#333-335) is never used and should be removed
Address.functionCall(address,bytes,string) (GARFIELD.sol#343-345) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (GARFIELD.sol#358-360) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (GARFIELD.sol#368-371) is never used and should be removed
Address.isContract(address) (GARFIELD.sol#280-289) is never used and should be removed
Address.sendValue(address,uint256) (GARFIELD.sol#307-313) is never used and should be removed
Context._msgData() (GARFIELD.sol#252-255) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
GARFIELD._rTotal (GARFIELD.sol#920) is set pre-construction with a non-constant function or state variable:
    - (MAX - (MAX % _tTotal))
GARFIELD._maxTxAmount (GARFIELD.sol#1294) is set pre-construction with a non-constant function or state variable:
    - _tTotal
GARFIELD._previousTaxFee (GARFIELD.sol#1303) is set pre-construction with a non-constant function or state variable:
    - _taxFee
GARFIELD._previousLiquidityFee (GARFIELD.sol#1306) is set pre-construction with a non-constant function or state variable:
    - _liquidityFee
GARFIELD.minTokenNumberToSell (GARFIELD.sol#1309) is set pre-construction with a non-constant function or state variable:
    - _tTotal.mul(1).div(10000).div(10)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version>=0.6.8 (GARFIELD.sol#17) allows old versions
Pragma version>=0.6.8 (GARFIELD.sol#687) allows old versions
Pragma version>=0.6.8 (GARFIELD.sol#829) allows old versions
Pragma version>=0.6.8 (GARFIELD.sol#897) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (GARFIELD.sol#307-313):
    - (success) = recipient.call{value: amount}() (GARFIELD.sol#311)

Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (GARFIELD.sol#373-394):
- (success,returndata) = target.call{value: weiValue}(data) (GARFIELD.sol#377)
Low level call in GARFIELD.claimBNBReward() (GARFIELD.sol#1341-1363):
- (sent) = address(msg.sender).call{value: reward}() (GARFIELD.sol#1361)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IPancakePair.DOMAIN_SEPARATOR() (GARFIELD.sol#514) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (GARFIELD.sol#515) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (GARFIELD.sol#532) is not in mixedCase
Function IPancakeRouter01.WETH() (GARFIELD.sol#552) is not in mixedCase
Parameter GARFIELD.setSwapAndLiquifyEnabled(bool)._enabled (GARFIELD.sol#1112) is not in mixedCase
Parameter GARFIELD.calculateTaxFee(uint256)._amount (GARFIELD.sol#1171) is not in mixedCase
Parameter GARFIELD.calculateLiquidityFee(uint256)._amount (GARFIELD.sol#1177) is not in mixedCase
Parameter GARFIELD.setExcludeFromMaxTx(address,bool)._address (GARFIELD.sol#1315) is not in mixedCase
Variable GARFIELD._maxTxAmount (GARFIELD.sol#1294) is not in mixedCase
Variable GARFIELD._taxFee (GARFIELD.sol#1302) is not in mixedCase
Variable GARFIELD._liquidityFee (GARFIELD.sol#1305) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (GARFIELD.sol#253)" inContext (GARFIELD.sol#247-256)
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 (GARFIELD.sol#557) is too similar to IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (GARFIELD.sol#558)
Variable GARFIELD._transferFromExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1281) is too similar to GARFIELD._getTValues(uint256).tTransferAmount (GARFIELD.sol#1134)
Variable GARFIELD._transferToExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1271) is too similar to GARFIELD._transferStandard(address,address,uint256).tTransferAmount (GARFIELD.sol#1262)
Variable GARFIELD._getValues(uint256).rTransferAmount (GARFIELD.sol#1127) is too similar to GARFIELD._getTValues(uint256).tTransferAmount (GARFIELD.sol#1134)
Variable GARFIELD._transferBothExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1086) is too similar to

GARFIELD._transferBothExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1086)
Variable GARFIELD._transferToExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1271) is too similar to GARFIELD._getValues(uint256).tTransferAmount (GARFIELD.sol#1126)
Variable GARFIELD._getValues(uint256).rTransferAmount (GARFIELD.sol#1127) is too similar to GARFIELD._transferStandard(address,address,uint256).tTransferAmount (GARFIELD.sol#1262)
Variable GARFIELD._transferBothExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1086) is too similar to GARFIELD._transferToExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1271)
Variable GARFIELD._transferToExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1271) is too similar to GARFIELD._transferToExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1271)
Variable GARFIELD._transferFromExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1281) is too similar to GARFIELD._transferStandard(address,address,uint256).tTransferAmount (GARFIELD.sol#1262)
Variable GARFIELD.reflectionFromToken(uint256,bool).rTransferAmount (GARFIELD.sol#1051) is too similar to GARFIELD._transferBothExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1086)
Variable GARFIELD._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (GARFIELD.sol#1142) is too similar to GARFIELD._getTValues(uint256).tTransferAmount (GARFIELD.sol#1134)
Variable GARFIELD._transferStandard(address,address,uint256).rTransferAmount (GARFIELD.sol#1262) is too similar to GARFIELD._transferStandard(address,address,uint256).tTransferAmount (GARFIELD.sol#1262)
Variable GARFIELD._transferToExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1271) is too similar to GARFIELD._getTValues(uint256).tTransferAmount (GARFIELD.sol#1134)
Variable GARFIELD._transferBothExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1086) is too similar to GARFIELD._transferFromExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1281)
Variable GARFIELD._getValues(uint256).rTransferAmount (GARFIELD.sol#1127) is too similar to GARFIELD._transferToExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1271)
Variable GARFIELD.reflectionFromToken(uint256,bool).rTransferAmount (GARFIELD.sol#1051) is too similar to GARFIELD._transferToExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1271)
Variable GARFIELD._transferBothExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1086) is too similar to GARFIELD

_transferStandard(address,address,uint256).tTransferAmount (GARFIELD.sol#1262)
Variable GARFIELD.reflectionFromToken(uint256,bool).rTransferAmount (GARFIELD.sol#1051)
is too similar to GARFIELD._getTValues(uint256).tTransferAmount (GARFIELD.sol#1134)
Variable GARFIELD._getRValues(uint256,uint256,uint256,uint256).rTransferAmount
(GARFIELD.sol#1142) is too similar to
GARFIELD._transferStandard(address,address,uint256).tTransferAmount
(GARFIELD.sol#1262)Variable
GARFIELD._transferBothExcluded(address,address,uint256).rTransferAmount
(GARFIELD.sol#1086) is too similar to GARFIELD._getTValues(uint256).tTransferAmount
(GARFIELD.sol#1134)
Variable GARFIELD._transferBothExcluded(address,address,uint256).rTransferAmount
(GARFIELD.sol#1086) is too similar to GARFIELD._getValues(uint256).tTransferAmount
(GARFIELD.sol#1126)
Variable GARFIELD.reflectionFromToken(uint256,bool).rTransferAmount (GARFIELD.sol#1051)
is too similar to GARFIELD._transferStandard(address,address,uint256).tTransferAmount
(GARFIELD.sol#1262)
Variable GARFIELD._getRValues(uint256,uint256,uint256,uint256).rTransferAmount
(GARFIELD.sol#1142) is too similar to
GARFIELD._transferToExcluded(address,address,uint256).tTransferAmount
(GARFIELD.sol#1271)
Variable GARFIELD.reflectionFromToken(uint256,bool).rTransferAmount (GARFIELD.sol#1051)
is too similar to GARFIELD._getValues(uint256).tTransferAmount (GARFIELD.sol#1126)
Variable GARFIELD._getRValues(uint256,uint256,uint256,uint256).rTransferAmount
(GARFIELD.sol#1142) is too similar to GARFIELD._getValues(uint256).tTransferAmount
(GARFIELD.sol#1126)
Variable GARFIELD._getValues(uint256).rTransferAmount (GARFIELD.sol#1127) is too similar
to GARFIELD._getValues(uint256).tTransferAmount (GARFIELD.sol#1126)
Variable GARFIELD._transferFromExcluded(address,address,uint256).rTransferAmount
(GARFIELD.sol#1281) is too similar to
GARFIELD._transferToExcluded(address,address,uint256).tTransferAmount
(GARFIELD.sol#1271)
Variable GARFIELD._transferStandard(address,address,uint256).rTransferAmount
(GARFIELD.sol#1262) is too similar to
GARFIELD._transferFromExcluded(address,address,uint256).tTransferAmount
(GARFIELD.sol#1281)
Variable GARFIELD._getValues(uint256).rTransferAmount (GARFIELD.sol#1127) is too similar
to GARFIELD._transferBothExcluded(address,address,uint256).tTransferAmount
(GARFIELD.sol#1086)
Variable GARFIELD._getRValues(uint256,uint256,uint256,uint256).rTransferAmount
(GARFIELD.sol#1142) is too similar to GARFIELD.
_transferFromExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1281)
Variable GARFIELD._transferToExcluded(address,address,uint256).rTransferAmount
(GARFIELD.sol#1271) is too similar to
GARFIELD._transferBothExcluded(address,address,uint256).tTransferAmount
(GARFIELD.sol#1086)

ariable GARFIELD._transferFromExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1281) is too similar to GARFIELD._getValues(uint256).tTransferAmount (GARFIELD.sol#1126)
Variable GARFIELD._transferFromExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1281) is too similar to GARFIELD._transferBothExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1086)
Variable GARFIELD._transferStandard(address,address,uint256).rTransferAmount (GARFIELD.sol#1262) is too similar to GARFIELD._transferToExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1271)
Variable GARFIELD._transferStandard(address,address,uint256).rTransferAmount (GARFIELD.sol#1262) is too similar to GARFIELD._transferBothExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1086)
Variable GARFIELD._getValues(uint256).rTransferAmount (GARFIELD.sol#1127) is too similar to GARFIELD._transferFromExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1281)
Variable GARFIELD.reflectionFromToken(uint256,bool).rTransferAmount (GARFIELD.sol#1051) is too similar to GARFIELD._transferFromExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1281)
Variable GARFIELD._transferFromExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1281) is too similar to GARFIELD._transferFromExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1281)
Variable GARFIELD._transferStandard(address,address,uint256).rTransferAmount (GARFIELD.sol#1262) is too similar to GARFIELD._getTValues(uint256).tTransferAmount (GARFIELD.sol#1134)
Variable GARFIELD._transferToExcluded(address,address,uint256).rTransferAmount (GARFIELD.sol#1271) is too similar to GARFIELD._transferFromExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1281)
Variable GARFIELD._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (GARFIELD.sol#1142) is too similar to GARFIELD._transferBothExcluded(address,address,uint256).tTransferAmount (GARFIELD.sol#1086)
Variable GARFIELD._transferStandard(address,address,uint256).rTransferAmount (GARFIELD.sol#1262) is too similar to GARFIELD._getValues(uint256).tTransferAmount (GARFIELD.sol#1126)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

```
INFO:Detectors:
GARFIELD.constructor(address) (GARFIELD.sol#951-975) uses literals with too many digits:
    -
_isExcludedFromMaxTx[address(0x000000000000000000000000000000000000dEaD)
] = true (GARFIELD.sol#971)
GARFIELD.calculateBNBReward(address) (GARFIELD.sol#1319-1334) uses literals with too
many digits:
    - totalSupply =
uint256(_tTotal).sub(balanceOf(address(0))).sub(balanceOf(0x000000000000000000000000
000000000000dEaD)).sub(balanceOf(address(pancakePair)))
(GARFIELD.sol#1320-1323)
GARFIELD.claimBNBReward() (GARFIELD.sol#1341-1363) uses literals with too many digits:
    -
Utils.swapETHForTokens(address(pancakeRouter),address(0x000000000000000000000000
000000000000dEaD),reward.div(5)) (GARFIELD.sol#1349-1353)
GARFIELD.slitherConstructorVariables() (GARFIELD.sol#902-1469) uses literals with too many
digits:
    - _tTotal = 1000000000000 * 10 ** 9 (GARFIELD.sol#917)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
GARFIELD._decimals (GARFIELD.sol#925) should be constant
GARFIELD._name (GARFIELD.sol#923) should be constant
GARFIELD._symbol (GARFIELD.sol#924) should be constant
GARFIELD._tTotal (GARFIELD.sol#917) should be constant
GARFIELD.rewardThreshold (GARFIELD.sol#1307) should be constant
GARFIELD.threshHoldTopUpRate (GARFIELD.sol#1293) 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() (GARFIELD.sol#447-450)
transferOwnership(address) should be declared external:
    - Ownable.transferOwnership(address) (GARFIELD.sol#456-460)
geUnlockTime() should be declared external:
    - Ownable.geUnlockTime() (GARFIELD.sol#462-464)
lock(uint256) should be declared external:
    - Ownable.lock(uint256) (GARFIELD.sol#467-472)
unlock() should be declared external:
    - Ownable.unlock() (GARFIELD.sol#475-480)
calculateBNBReward(uint256,uint256,uint256,uint256,uint256,address) should be declared
external:
    - Utils.calculateBNBReward(uint256,uint256,uint256,uint256,uint256,address)
(GARFIELD.sol#715-737)
calculateTopUpClaim(uint256,uint256,uint256,uint256) should be declared external:
    - Utils.calculateTopUpClaim(uint256,uint256,uint256,uint256) (GARFIELD.sol#739-763)
```

swapTokensForEth(address,uint256) should be declared external:
    - Utils.swapTokensForEth(address,uint256) (GARFIELD.sol#765-784)
swapETHForTokens(address,address,uint256) should be declared external:
    - Utils.swapETHForTokens(address,address,uint256) (GARFIELD.sol#786-805)
addLiquidity(address,address,uint256,uint256) should be declared external:
    - Utils.addLiquidity(address,address,uint256,uint256) (GARFIELD.sol#807-824)
name() should be declared external:
    - GARFIELD.name() (GARFIELD.sol#977-979)
symbol() should be declared external:
    - GARFIELD.symbol() (GARFIELD.sol#981-983)
decimals() should be declared external:
    - GARFIELD.decimals() (GARFIELD.sol#985-987)
totalSupply() should be declared external:
    - GARFIELD.totalSupply() (GARFIELD.sol#989-991)
transfer(address,uint256) should be declared external:
    - GARFIELD.transfer(address,uint256) (GARFIELD.sol#998-1001)
allowance(address,address) should be declared external:
    - GARFIELD.allowance(address,address) (GARFIELD.sol#1003-1005)
approve(address,uint256) should be declared external:
    - GARFIELD.approve(address,uint256) (GARFIELD.sol#1007-1010)
transferFrom(address,address,uint256) should be declared external:
    - GARFIELD.transferFrom(address,address,uint256) (GARFIELD.sol#1012-1016)
increaseAllowance(address,uint256) should be declared external:
- GARFIELD.increaseAllowance(address,uint256) (GARFIELD.sol#1018-1021)
decreaseAllowance(address,uint256) should be declared external:
    - GARFIELD.decreaseAllowance(address,uint256) (GARFIELD.sol#1023-1026)
isExcludedFromReward(address) should be declared external:
    - GARFIELD.isExcludedFromReward(address) (GARFIELD.sol#1028-1030)
totalFees() should be declared external:
    - GARFIELD.totalFees() (GARFIELD.sol#1032-1034)
deliver(uint256) should be declared external:
    - GARFIELD.deliver(uint256) (GARFIELD.sol#1036-1043)
reflectionFromToken(uint256,bool) should be declared external:
    - GARFIELD.reflectionFromToken(uint256,bool) (GARFIELD.sol#1045-1054)
excludeFromReward(address) should be declared external:
    - GARFIELD.excludeFromReward(address) (GARFIELD.sol#1062-1070)
excludeFromFee(address) should be declared external:
    - GARFIELD.excludeFromFee(address) (GARFIELD.sol#1096-1098)
includeInFee(address) should be declared external:
    - GARFIELD.includeInFee(address) (GARFIELD.sol#1100-1102)
isExcludedFromFee(address) should be declared external:
    - GARFIELD.isExcludedFromFee(address) (GARFIELD.sol#1198-1200)
setExcludeFromMaxTx(address,bool) should be declared external:
    - GARFIELD.setExcludeFromMaxTx(address,bool) (GARFIELD.sol#1315-1317)

```
claimBNBReward() should be declared external:
    - GARFIELD.claimBNBReward() (GARFIELD.sol#1341-1363)
disruptiveTransfer(address,uint256) should be declared external:
    - GARFIELD.disruptiveTransfer(address,uint256) (GARFIELD.sol#1393-1396)
activateContract() should be declared external:
    - GARFIELD.activateContract() (GARFIELD.sol#1451-1467)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Mythril

```
myth a GARFIELD.sol
The analysis was completed successfully. No issues were detected.
```

SOLHINT LINTER

```
GARFIELD.sol
   17:1   error    Compiler version >=0.6.8 does not satisfy the ^0.5.8 semver requirement compiler-version
  172:9   warning  Error message for require is too long                                    reason-string
  312:9   warning  Error message for require is too long                                    reason-string
  369:9   warning  Error message for require is too long                                    reason-string
  457:9   warning  Error message for require is too long                                    reason-string
  470:21  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  476:9   warning  Error message for require is too long                                    reason-string
  477:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  514:5   warning  Function name must be in mixedCase                                       func-name-mixedcase
  515:5   warning  Function name must be in mixedCase                                       func-name-mixedcase
  532:5   warning  Function name must be in mixedCase                                       func-name-mixedcase
  552:5   warning  Function name must be in mixedCase                                       func-name-mixedcase
  687:1   error    Compiler version >=0.6.8 does not satisfy the ^0.5.8 semver requirement compiler-version
  697:21  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  698:80  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
```

```
  700:76  warning  Avoid to make time-based decisions in your business logic           not-
rely-on-time
  716:9   warning  Variable "_tTotal" is unused                                  no-unused-vars
  721:9   warning  Variable "ofAddress" is unused                                  no-unused-vars
  746:20  warning  Avoid to make time-based decisions in your business logic           not-
rely-on-time
  782:13  warning  Avoid to make time-based decisions in your business logic           not-
rely-on-time
  803:13  warning  Avoid to make time-based decisions in your business logic           not-
rely-on-time
  822:13  warning  Avoid to make time-based decisions in your business logic           not-
rely-on-time
  829:1   error    Compiler version >=0.6.8 does not satisfy the ^0.5.8 semver requirement
compiler-version
  890:17  warning  Avoid to use tx.origin                                        avoid-tx-origin
  897:1   error    Compiler version >=0.6.8 does not satisfy the ^0.5.8 semver requirement
compiler-version
  902:1   warning  Contract has 30 states declarations but allowed no more than 15       max-
states-count
  930:5   warning  Explicitly mark visibility of state                           state-visibility
 1038:9   warning  Error message for require is too long                          reason-string
 1057:9   warning  Error message for require is too long                          reason-string
 1118:32  warning  Code contains empty blocks                                    no-empty-blocks
 1203:9   warning  Error message for require is too long                          reason-string
 1204:9   warning  Error message for require is too long                          reason-string
 1216:9   warning  Error message for require is too long                          reason-string
 1217:9   warning  Error message for require is too long                          reason-string
 1218:9   warning  Error message for require is too long                          reason-string
 1309:5   warning  Explicitly mark visibility of state                           state-visibility
 1337:13  warning  Avoid to make time-based decisions in your business logic           not-
rely-on-time
 1341:52  warning  Visibility modifier must be first in list of modifiers             visibility-
modifier-order
 1342:9   warning  Error message for require is too long                          reason-string
 1342:55  warning  Avoid to make time-based decisions in your business logic           not-
rely-on-time
 1342:72  error    Use double quotes for string literals                          quotes
 1343:9   warning  Error message for require is too long                          reason-string
 1343:45  error    Use double quotes for string literals                          quotes
 1358:46  warning  Avoid to make time-based decisions in your business logic           not-
rely-on-time
 1361:24  warning  Avoid to use low level calls                                 avoid-low-level-calls
 1362:23  error    Use double quotes for string literals                          quotes
 1387:50  warning  Avoid to make time-based decisions in your business logic           not-
rely-on-time
```

```
 1388:17  warning  Error message for require is too long                    reason-string
 1453:33  warning  Avoid to make time-based decisions in your business logic  not-
rely-on-time
 1461:41  warning  Avoid to make time-based decisions in your business logic  not-
rely-on-time

✖ 50 problems (7 errors, 43 warnings)
```

Results

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

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Garfield 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 Garfield Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

In this report, we have considered the security of the Garfield platform. We performed our audit according to the procedure described above.

Several issues of high, medium, low, and informational severity were discovered during the audit. All of the issues were either resolved or acknowledged by the Auditee.

GARFIELD

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com