AUDIT

Reefer Token

CoinMooner

Find Next Moonshot coins

TABLE OF CONTENTS

V. DISCLAIMER

AUDIT SUMMARY

This report was written for Reefer Token (REEFER) in order to find flaws and vulnerabilities in the Reefer Token project's source code, as well as any contract dependencies that weren't part of an officially recognized library.

A comprehensive examination has been performed, utilizing Static Analysis, Manual Review, and Reefer Token Deployment techniques. The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors
- Assessing the codebase to ensure compliance with current best practices and industry standards
- Ensuring contract logic meets the specifications and intentions of the client
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
- Through line-by-line manual review of the entire codebase by industry expert

AUDIT OVERVIEW

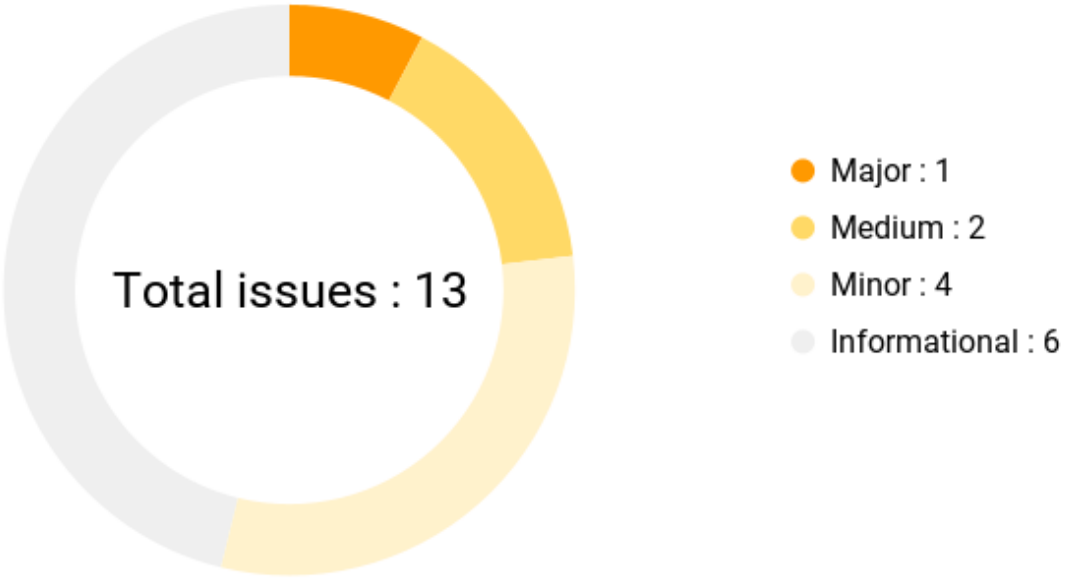
PROJECT SUMMARY

Project name	Reefer Token
Description	REEFER Token is Setting a New Industry Standard on how Legal Dispensaries, Smoke Shops, Vape Shops, Cannabis Lounges, Restaurants, and Gamers Interact. Cannabis, Crypto, NFTs & Metaverse Combined to Birth the REEFER Ecosystem Revolution!
Platform	BNB Smart Chain
Language	Solidity
Codebase	https://bscscan.com/token/0xD3C931F70ff2E98d0639cf4ab29C7FB16b9c5E04

FINDINGS SUMMARY

Vulnerability	Total
Critical	0
Major	1
Medium	2
Minor	4
Informational	6

AUDIT FINDINGS



Code	Title	Severity
STATE-1	State Variables	Informational
PRES-1	setPreSaleStatus	Informational
PRES-2	setPreSaleRate	Informational
ROU-1	updateRouter	Minor
MAR-1	setMarketingWallet	Medium
SET-1	setCommunityWallet	Medium
SET-3	setSwapEnabled	Informational
GAS-1	MoreThan or LessThan Gas Saving	Informational
LIQ-1	addLiquidity	Minor

TRAN-1	_transfer takeFee setter	Informational
TRAN-2	_transfer if check	Minor
TRAN-3	_transferring from address(this) to address(this)	Minor
RESC-1	rescueBNB no receiver input	Major

STATE-1 | State Variables

Description

There are state variables such as the addresses for marketing, dev, and community wallets that are input manually rather than being initialized in the constructor. If these variables remain the same they could be changed to constant variables and in the cases where setters are created they could be removed.

Recommendation

If the address will remain constant then the state variable should be changed to a constant and the setter should be removed.

```
uint256 public constant preSaleRate = <uint256>
…
address public constant marketingWallet = <address>
address public constant devWallet = <address>
address public constant communityWallet = <address>
uint256 public constant gasForProcessing = >uint256>
```

PRES-1 | setPreSaleStatus

Description

Calling this will set the bool to the input value and if the input is the same as the existing state variable the operation will still happen. Could remove the input and set the value to the !state value which effectively toggles the state.

Recommendation

Remove the input and change the value to the opposite of the current state preSaleEnabled = !state;

PRES-2 | setPreSaleRate

Description

Calling this will set the bool to the input value and if the input is the same as the existing state variable the operation will still happen. Could remove the input and set the value to the !state value which effectively toggles the state.

Recommendation

Remove the input and change the value to the opposite of the current state preSaleEnabled = !state;

ROU-1 | updateRouter

Description

The update router prevents the address being set from being an existing address but it does not check the address does not equal a zero address. This is a minor issue as the updateRouter function could be called immediately to reset the value. However, any calls to addLiquidity and swapTokensForBNB would revert as the functions are not implemented on the zero address.

Recommendation

Add a require statement check that the new address is not address zero

require(newAddress != address(0), "ZERO_ADDRESS);

MAR-1 | setMarketingWallet

Description

There are no checks that the address being set for the wallet does not equal address zero. This means if a zero address was passed in by mistake and without recognition the swapAndLiquidify function would send funds to the wrong address. As there is a setter, it can be easily rectified but it may be safer to impose a check to prevent this from occurring.

Recommendation

Add a require statement check that the new address it not address zero

```
require(newWallet != address(0), "ZERO_ADDRESS);
```

SET-1 | setCommunityWallet

Description

There are no checks that the address being set for the wallet does not equal address zero. This means if a zero address was passed in by mistake and without recognition the swapAndLliquidify function would send funds to the wrong address. As there is a setter, it can be easily rectified but it may be safer to impose a check to prevent this from occurring.

Recommendation

Add a require statement check that the new address is not address zero

require(newWallet != address(0), “ZERO_ADDRESS);

SET-2 | setSwapEnabled

Description

Calling this will set the bool to the input value and if the input is the same as the existing state variable the operation will still happen. Could remove the input and set the value to the !state value which effectively toggles the state.

Recommendation

Remove the input and change the value to the opposite of the current state preSaleEnabled = !state;

GAS-1 | MoreThan or LessThan Gas Saving

Description

There are a series of operations where more than or equal to and less than or equal to is used to check the uint256 value. If there is a preference to save gas then checking just more than or less than will reduce the amount of gas spent as less opcodes are used

Recommendation

Change more than or equal to and less than or equal to, to more than and less than. Only if there is a preference to save gas.

LIQ-1 | addLiquidity

Description

There is no check that the msg.value does not equal 0 when this function is called, which means if the amount being sent is 0 then the transaction will revert as ETH liquidity is being added. As there is no check before the call to the add liquidity function, it will mean that the gas that is used before this point will be lost by the caller.

Recommendation

Add require statement that the msg.value does not equal 0.

```
require(msg.value != 0, “NO_ETH_SENT”);
_approve(address(this), address(router), tokenAmount);
```

TRAN-1 | _transfer takeFee setter

Description

The boolean for takeFee is set based on swapping but there are no instances where the _transfer function is called and the modifier lockTheSwap is also called. This means the value for swapping will always be true.

TRAN-2 | _transfer if check

Description

In addition, there is a boolean check for isExcludedFromFess where the takeFee is set to false if either of the addresses are excluded. Instead of changing the boolean, the check could be if both addresses are false then execute the logic in the if statement checking takeFee.

Recommendation

Change the logic for the if check to check if both of the addresses are false and then execute.

```
if(!_isExcludedFromFees[from] && !_isExcludedFromFees[to]) {
        Uint256 fees = amount * totalTax / 100;
amount = amount - fees;
super._transfer(from, address(this), fees);
}
```

TRAN-3 | _transferring from address(this) to address(this)

Description

The takeFee function transfers tokens from the input parameter of from to address(this). However, the _transfer function is called from buyTokens() and the input passed as from is address(this), which means when the takeFee logic tries to transfer tokens the from and to addresses will both be address(this).

It's unclear what the plan is with regards to the fee being taken, as the tokens are being transferred in exchange for the ether being sent to the contract. If there is a global tax they would like to charge for all tokens purchased then the better approach would be to calculate the amount when fee is deducted then send that amount of tokens.

Recommendation

Remove the transfer function as it does not do anything.

```
if(!_isExcludedFromFees[from] && !_isExcludedFromFees[to]) {
        Uint256 fees = amount * totalTax / 100;
amount = amount - fees;
}
```

RESC-1 | rescueBNB no receiver input for sendValue

Description

There is no receiver input provided for the sendValue function when it is called and the address being sent from is the owner rather than the contract.

Recommendation

address(this).sendValue(payable(owner()), address(this).balance)

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement.

This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement.

This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CoinMooner's prior written consent.This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CoinMooner to perform a security assessment.

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or

legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project.

This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk.

CoinMooner's position is that each company and individual are responsible for their own due diligence and continuous security. CoinMooner's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or fun.