freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

TokenName

$TokenSymbol

27/07/2025

TOKEN OVERVIEW

Fees

- Buy fees: 5%
- Sell fees: 0%

Fees privileges

- Can’t change fees

Ownership

- Owned

Minting

- Mint function not detected

Max Tx Amount / Max Wallet Amount

- Can’t change max tx amount and / or max wallet amount

Blacklist

- Blacklist function not detected

Other privileges

- Potential issue with the setProxyContract function in the contract that could allow the owner to set a malicious proxy contract, leading to significant risks

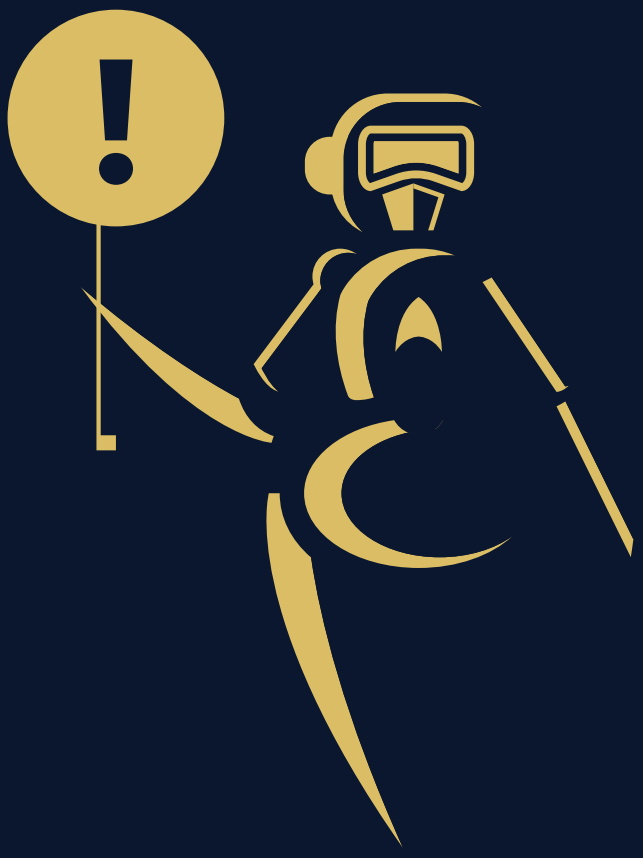
TABLE OF CONTENTS

DISCLAIMER

The information provided on this analysis document is only for general information and should not be used as a reason to invest.

FreshCoins Team will take no payment for manipulating the results of this audit.

The score and the result will stay on this project page information on our website https://freshcoins.io
FreshCoins Team does not guarantees that a project will not sell off team supply, or any other scam strategy (RUG or Honeypot etc)



INTRODUCTION

FreshCoins (Consultant) was contracted by TokenName (Customer) to conduct a Smart Contract Code Review and Security Analysis.

N/A

Network: Binance Smart Chain (BSC)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 27/07/2025

WEBSITE DIAGNOSTIC

N/A

0-49

50-89

90-100

Performance	Accessibility	Best Practices	SEO	Progressive Web App
NA	NA	NA	NA	NA

Socials

X (Twitter)

N/A

Telegram

N/A

AUDIT OVERVIEW

81

Security Score

94

Static Scan
Automatic scanning for common vulnerabilities

90

ERC Scan
Automatic checks for ERC's conformance

1 High

5 Medium

0 Low

2 Optimizations

0 Informational

No.	Issue description	Checking Status
1	Compiler Errors / Warnings	Passed
2	Reentrancy and Cross-function	Low
3	Front running	Low
4	Timestamp dependence	Passed
5	Integer Overflow and Underflow	Passed
6	Reverted DoS	Passed
7	DoS with block gas limit	Passed
8	Methods execution permissions	Passed
9	Exchange rate impact	Passed
10	Malicious Event	Passed
11	Scoping and Declarations	Passed
12	Uninitialized storage pointers	Passed
13	Design Logic	Passed
14	Safe Zeppelin module	Passed

OWNER PRIVILEGES & FINDINGS

Contract owner can change proxyContract address

Centralized Control and Trust Dependency

The setProxyContract function is restricted to the contract's owner via the onlyOwner modifier, meaning only the owner can change the proxyContract address.

The proxyContract plays a significant role in the MonkeyToken and KoalaAutoBuyer contracts

If the proxyContract is set to a malicious or broken contract, the 95% of BNB forwarded to it could be lost or locked.

```
function setProxyContract(address _proxy) external onlyOwner {
    proxyContract = _proxy;
}
```

Reentrancy Vulnerability in receiveBNB and _handleReceivedBNB

Both the receiveBNB function in KoalaAutoBuyer and the _handleReceivedBNB function in MonkeyToken involve external calls to transfer BNB and invoke the proxy contract (receiveBNB calls proxyContract with value). These external calls are not protected against reentrancy in a robust way. Although MonkeyToken._handleReceivedBNB uses the nonReentrant modifier, the KoalaAutoBuyer.receiveBNB function does not, and it performs external calls to the Uniswap router and the KOALA interface (recordLP and _distributeRemainingKOALA). A malicious contract could potentially exploit these calls to re-enter and manipulate state (e.g., liquidity or token balances).

```
function receiveBNB(address sender) external payable nonReentrant {
    require(msg.value > 0, "No BNB sent");
    uint256 glgBalance = _swapBNBForGLG(msg.value);
    uint256 glgToSwap = glgBalance * 70 / 100;
    uint256 koalaBought = _swapGLGForKOALA(glgToSwap);
    uint256 glgForLiquidity = glgBalance * 30 / 100;
    uint256 liquidity = _addLiquidity(glgForLiquidity, koalaBought, sender);
    KOALA(token).recordLP(sender, liquidity);
    _distributeRemainingKOALA(sender);
}
```

Add the nonReentrant modifier to the receiveBNB function in KoalaAutoBuyer to prevent reentrancy attacks.

Lack of Input Validation

The receiveGLG function in KoalaAutoBuyer checks if the caller is the token contract (require(msg.sender == token, "Only MonkeyToken can call")) and verifies the GLG balance (require(glgBalance >= glgAmount, "No GLG to swap")). However, it does not validate the glgAmount parameter beyond ensuring it is not greater than the contract's GLG balance. If a malicious MonkeyToken contract calls receiveGLG with an excessively large or zero glgAmount, it could lead to unexpected behavior, such as failed swaps or incorrect token transfers.

Hardcoded Addresses Without Update Mechanism

The contract uses hardcoded addresses for feeWallet, feeWallet2, feeWallet3, glgAddress, emergencyAdmin, and routerAddress without mechanisms to update them. If any address becomes compromised or obsolete (e.g., a wallet is hacked or the router is deprecated), the contract cannot adapt, risking loss of funds or functionality.

```
address public glgAddress = 0xC0EdcDdd6d5417c22467e3d5642Efa1820E454f8;
address public feeWallet = 0xfBaFCf805dbC9C1A934FFc8D467A9484eB68eA50;
address public feeWallet2 = 0x7f7A65D12828D64Be950106e7424E004DF38c16D;
address public feeWallet3 = 0x63b7Ec0BC30d387Cd7738DE117195f3e531D40c1;
address public routerAddress = 0x10ED43C718714eb63d5aA57B78B54704E256024E;
```

Add owner-controlled functions to update these addresses, with events emitted for transparency. Include checks to prevent setting addresses to address(0) or invalid contracts.

Potential Division by Zero

In settleBurnAndLP, the dividend share calculation holders[holder] * round.lpAmount / totalLP assumes totalLP is non-zero. If holderList is non-empty but all holders have zero balance (e.g., due to a bug in recordLP or removeLPAndSend), totalLP could be zero, causing a division-by-zero error and halting dividend distribution.

```
uint256 share = holders[holder] * round.lpAmount / totalLP;
```

Add a check to ensure totalLP > 0 before performing the division. If totalLP is zero, skip the distribution or handle it gracefully (e.g., mark the round as finished).

Unchecked External Calls

The _handleReceivedBNB function makes external calls to feeWallet and proxyContract using .call. While it checks the success of these calls, it doesn't verify the recipient addresses are contracts or handle partial success scenarios, which could lead to silent failures or stuck funds if the recipients revert unexpectedly.

```
(bool success1, ) = feeWallet.call{value: feeAmount}("");

(bool success,) = proxyContract.call{value: remainingBNB}(abi.encodeWithSignature("receiveBNB(address)",
sender));
```

Verify that feeWallet and proxyContract are valid contract addresses using Address.isContract from OpenZeppelin. Consider using a try-catch block for external calls to handle failures gracefully and log failures via events.

Redundant Approval

In the _addLiquidity function of KoalaAutoBuyer, the contract approves both glgAddress and token twice (reset to zero and then set the desired amount). While this is safe, it incurs unnecessary gas costs due to redundant approvals.

Combine approvals into a single call where possible, or check if the current allowance is sufficient before resetting

```
function _addLiquidity(uint256 glgAmount, uint256 koalaAmount, address sender) internal returns (uint256)
{
    if (IERC20(glgAddress).allowance(address(this), address(router)) < glgAmount) {
        IERC20(glgAddress).approve(address(router), glgAmount);
    }
    if (IERC20(token).allowance(address(this), address(router)) < koalaAmount) {
        IERC20(token).approve(address(router), koalaAmount);
    }
    ...
}
```

Unused Function

The getExpectedAmountOut function in KoalaAutoBuyer is defined but not used anywhere in the contract. Unused code can clutter the contract and increase gas costs during deployment.

Remove the unused function or document its intended use for future implementation. If it's meant for external use, make it external view and add NatSpec documentation.

Workflow

MonkeyToken acts as the main token contract and entry point for users sending BNB. It takes a 5% fee and forwards the remaining BNB to KoalaAutoBuyer.

KoalaAutoBuyer processes the BNB to acquire GLG and Monkey7 tokens, adds liquidity to the Monkey7/GLG pool, and returns any remaining Monkey7 tokens to MonkeyToken for distribution to referrers.

The contracts are tightly coupled, with MonkeyToken relying on KoalaAutoBuyer for liquidity provision and KoalaAutoBuyer relying on MonkeyToken for recording LP holders and distributing dividends.

Recommendation:

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. The risk can be prevented by temporarily locking the contract or renouncing ownership.

CONCLUSION AND ANALYSIS

Smart Contracts within the scope were manually reviewed and analyzed with static tools.

Audit report overview contains all found security vulnerabilities and other issues in the reviewed code.

Found 1 HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	5%
Sell fees:	0%
Max TX:	N/A
Max Sell:	N/A

Honeypot Risk

Ownership:	Owned
Blacklist:	Not detected
Modify Max TX:	Not detected
Modify Max Sell:	Not detected
Disable Trading:	Not detected

Rug Pull Risk

Liquidity:	N/A
Holders:	100% unlocked tokens

TOKENSYMBOL TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

TECHNICAL DISCLAIMER

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. The audit can't guarantee the explicit security of the audited project / smart contract.