freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Kitty Run

$KTR

15/06/2024

TOKEN OVERVIEW

Fees

- Buy fees: 0%
- Sell fees: 3%

Fees privileges

- Can change buy fees up to 10% and sell fees up to 10%

Ownership

- Owned

Minting

- No mint function

Max Tx Amount / Max Wallet Amount

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

Blacklist

- Blacklist not function detected

Other privileges

- Can exclude / include from fees

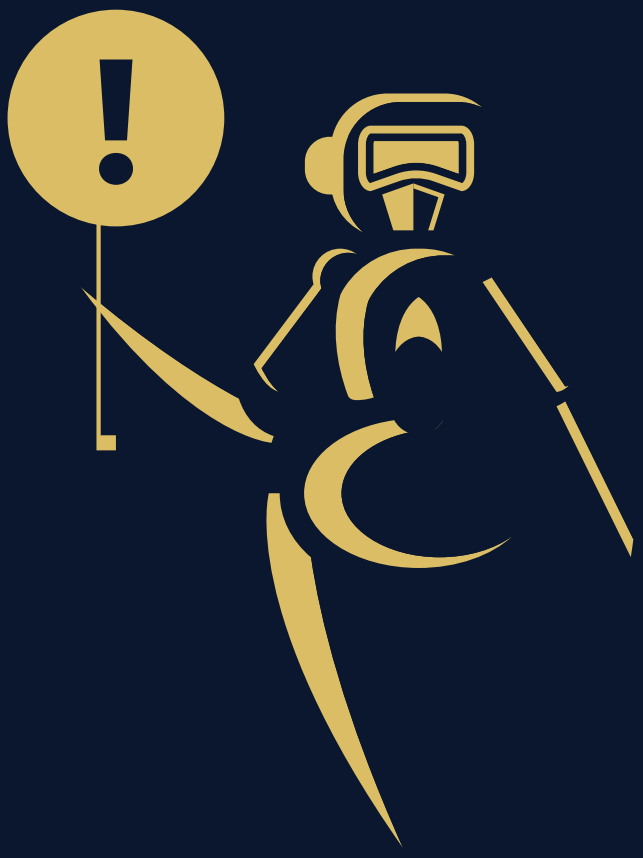
TABLE OF CONTENTS

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
Kitty Run (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x081cACFd43Fdcdd4CdFC822d9653625d144d3De5

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 15/06/2024

WEBSITE DIAGNOSTIC

https://kittyrun.io/

0-49 50-89 90-100

Performance Accessibility Best Practices SEO Progressive Web App

Socials

Twitter

https://x.com/kittyrun_io

Telegram

https://t.me/Kittyrun_Official

AUDIT OVERVIEW

88

Security Score

98

Static Scan

Automatic scanning for common vulnerabilities

84

ERC Scan

Automatic checks for ERC's conformance

0 High

1 Medium

0 Low

0 Optimizations

0 Informational

No.	Issue description	Checking Status
1	Compiler Errors / Warnings	Passed
2	Reentrancy and Cross-function	Passed
3	Front running	Low
4	Timestamp dependence	Passed
5	Integer Overflow and Underflow	Passed
6	Reverted DoS	Passed
7	DoS with block gas limit	Low
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

- Contract owner can't mint tokens after initial contract deploy
- Contract owner can't exclude addresses from transactions
- Contract owner can exclude/include wallet from tax

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
    _excludeFromFees(account, excluded);
}

function _excludeFromFees(address account, bool excluded) private {
    _isExcludedFromFees[account] = excluded;
    emit ExcludeFromFees(account, excluded);
}
```

- Burnable token

```
function burn(uint256 value) public virtual {
    _burn(_msgSender(), value);
}

function _burn(address account, uint256 value) internal {
    if (account == address(0)) {
        revert ERC20InvalidSender(address(0));
    }
    _update(account, address(0), value);
}
```

- Contract owner can change buy fees up to 10% and sell fees up to 10%

```
function updateFees(uint256 _buyFee, uint256 _sellFee) public onlyOwner {
    _updateFees(_buyFee, _sellFee);
}

function _updateFees(uint256 _buyFee, uint256 _sellFee) private {
    require(_buyFee <= 1_000 && _sellFee <= 1_000, "Invalid fees"); // _buyFee and _sellFee <= 10%
    buyFee = _buyFee;
    sellFee = _sellFee;
}
```

- Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    _transferOwnership(address(0));
}
```

Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    if (newOwner == address(0)) {
      revert OwnableInvalidOwner(address(0));
    }
    _transferOwnership(newOwner);
}

function _transferOwnership(address newOwner) internal virtual {
    address oldOwner = _owner;
    _owner = newOwner;
    emit OwnershipTransferred(oldOwner, newOwner);
}
```

Contract owner has ability to retrieve any token held by the contract

Native tokens NOT excluded

```
function withdrawStuck() external onlyOwner {
    uint256 balance = IERC20(address(this)).balanceOf(address(this));
    IERC20(address(this)).transfer(msg.sender, balance);
    payable(msg.sender).transfer(address(this).balance);
}

function withdrawStuckEth(address toAddr) external onlyOwner {
    (bool success, ) = toAddr.call{value: address(this).balance}("");
    require(success);
}
```

Contract owner can change feeWallet address

Current value:

feeWallet: 0x487b15B23EB4661bD9b1EA9FE2965677bdc05500

```
function updateFeeWallet(address _feeWallet) external onlyOwner {
    _updateFeeWallet(_feeWallet);
}

function _updateFeeWallet(address _feeWallet) private {
    require(_feeWallet != address(0), "Zero address");
    feeWallet = _feeWallet;
}
```

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 no HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	0%
Sell fees:	3%
Max TX:	N/A
Max Wallet:	N/A

Honeypot Risk

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

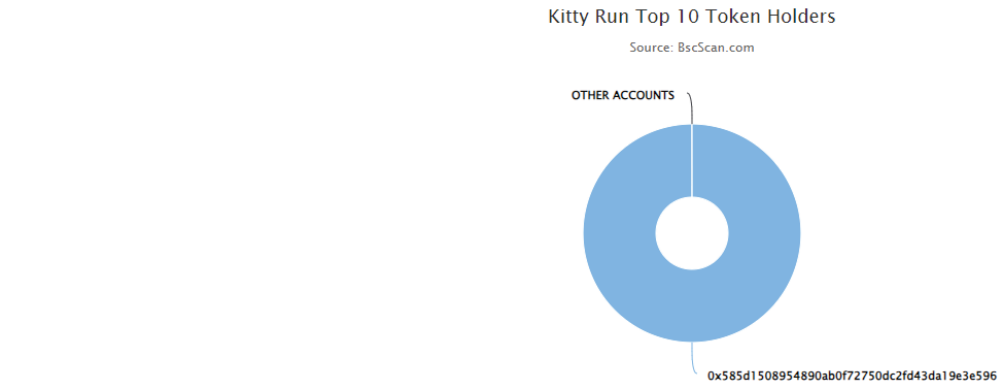
Rug Pull Risk

Liquidity:	N/A
Holders:	100% unlocked tokens

KTR TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (3,200,000,000.00 Tokens) of Kitty Run

Token Total Supply: 3,200,000,000.00 Token | Total Token Holders: 1



(A total of 3,200,000,000.00 tokens held by the top 10 accounts from the total supply of 3,200,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x585D1508...a19E3E596	3,200,000,000	100.0000%

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.