



SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



FROG INU

Frog Inu
\$FGI

25/05/2022

TABLE OF CONTENTS

- 1 **DISCLAIMER**
- 2 **INTRODUCTION**
- 3-4 **AUDIT OVERVIEW**
- 5-6 **OWNER PRIVILEGES**
- 7 **CONCLUSION AND ANALYSIS**
- 8 **TOKEN DETAILS**
- 9 **FROG INU TOKEN ANALYTICS &
TOP 10 TOKEN HOLDERS**
- 10 **TECHNICAL DISCLAIMER**



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 Honey pot etc)



INTRODUCTION

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

0xE5c355c993dE69C06b5DD1A0c9a2c39b4Cde5511

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 **25/05/2022**



AUDIT OVERVIEW



Security Score



Static Scan

Automatic scanning for common vulnerabilities



ERC Scan

Automatic checks for ERC's conformance



High



Medium



Low



Optimizations



Informational



No.	Issue description	Checking Status
1	Compiler Errors / Warnings	Passed
2	Reentrancy and Cross-function	Passed
3	Front running	Passed
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

Contract owner can't mint tokens after initial contract deploy

Contract owner can't exclude an address from transactions

Contract owner can exclude/include wallet(s) from tax

```
function excludeMultipleAccountsFromFees(address[] calldata accounts, bool excluded) public onlyOwner {  
    for(uint256 i = 0; i < accounts.length; i++) {  
        _isExcludedFromFee[accounts[i]] = excluded;  
    }  
}
```

Contract owner can withdraw tokens from smart contract

```
function rescueForeignTokens(address _tokenAddr, address _to, uint _amount) public onlyDev() {  
    emit tokensRescued(_tokenAddr, _to, _amount);  
    Token(_tokenAddr).transfer(_to, _amount);  
}
```

Contract owner can change `_developmentAddress` and `_marketingAddress` addresses

`_developmentAddress` : `0x6204f9aA0B08D306d76127358F9b225C5496f8C4`

`_marketingAddress` : `0xE2EC412fa68CB667E47b13E410Ce31ea730071F8`

```
function setNewDevAddress(address payable dev) public onlyDev() {  
    emit devAddressUpdated(_developmentAddress, dev);  
    _developmentAddress = dev;  
    _isExcludedFromFee[_developmentAddress] = true;  
}  
  
function setNewMarketingAddress(address payable markt) public onlyDev() {  
    emit marketingAddressUpdated(_marketingAddress, markt);  
    _marketingAddress = markt;  
    _isExcludedFromFee[_marketingAddress] = true;  
}
```

Contract owner can change swap settings

```
function toggleSwap(bool _swapEnabled) public onlyDev {  
    swapEnabled = _swapEnabled;  
}
```

Contract owner can change buy fees up to 16% and sell fees up to 16%

```
function setFee(uint256 redisFeeOnBuy, uint256 redisFeeOnSell, uint256 taxFeeOnBuy, uint256 taxFeeOnSell)
public onlyDev {
    require(redisFeeOnBuy < 11, "Redis cannot be more than 10.");
    require(redisFeeOnSell < 11, "Redis cannot be more than 10.");
    require(taxFeeOnBuy < 7, "Tax cannot be more than 6.");
    require(taxFeeOnSell < 7, "Tax cannot be more than 6.");
    _redisFeeOnBuy = redisFeeOnBuy;
    _redisFeeOnSell = redisFeeOnSell;
    _taxFeeOnBuy = taxFeeOnBuy;
    _taxFeeOnSell = taxFeeOnSell;
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    emit OwnershipTransferred(_owner, address(0));
    _owner = address(0);
}
```

Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    emit OwnershipTransferred(_owner, newOwner);
    _owner = newOwner;
}
```



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 issue during the first review.

TOKEN DETAILS

Details

Buy fees:	10%
Sell fees:	10%
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:	Clean



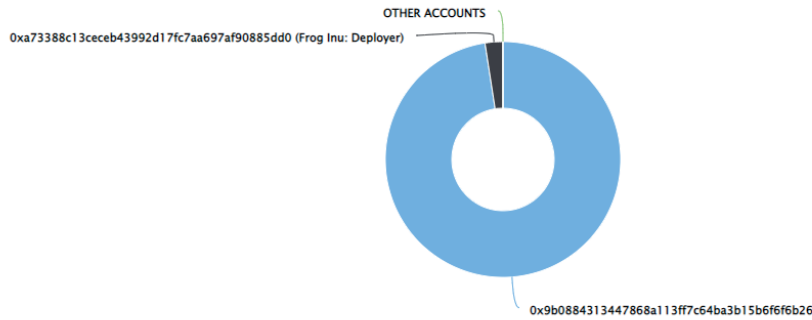
FROG INU TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (1,000,000,000,000,000.00 Tokens) of Frog Inu

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 2

Frog Inu Top 10 Token Holders

Source: BscScan.com



(A total of 1,000,000,000,000,000.00 tokens held by the top 10 accounts from the total supply of 1,000,000,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x9b0884313447868a113ff7c64ba3b15b6f6f6b26	975,600,000,000,000	97.5600%
2	Frog Inu: Deployer	24,400,000,000,000	2.4400%

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.

