freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Flash 2.0

$FLASH 2.0

15/07/2023

TOKEN OVERVIEW

Fees

• Buy fees:	10%
• Sell fees:	10%
• Transfer fees:	0%

Fees privileges

• Can change buy fees up to 10%, sell fees up to 10% and transfer fees up to 10%

Ownership

• Owned

Minting

• No mint function

Max Tx Amount / Max Wallet Amount

• Can't change max tx amount and max wallet amount

Blacklist

• Blacklist function not detected

Other privileges

• Contract owner has to call setLaunchInSeconds() function to enable trade

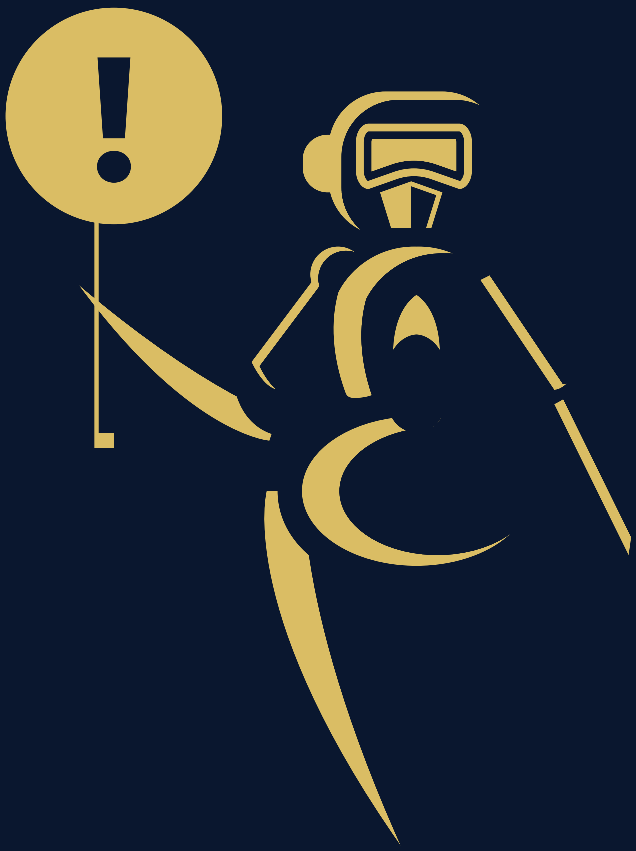
TABLE OF CONTENTS

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
Flash 2.0 (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x878Cc1faE3b8d5cF11C7eF7BCF065e84968a66B5

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 15/07/2023

WEBSITE DIAGNOSTIC

N/A

0-49	50-89	90-100

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

Socials

Twitter	Telegram
N/A	N/A

AUDIT OVERVIEW

82

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

84

ERC Scan
Automatic checks for ERC's conformance

1	High
2	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	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 from tax

```
function setExcludedFromFee(address account, bool exclude) public onlyOwner{
      require(exclude||account!=address(this));
      ExcludedFromFees[account]=exclude;
      emit OnSetExcludedFromFee(account,exclude);
}
```

● Contract owner can exclude/include wallet from reflection

```
function setExcludedFromReflection(address account, bool exclude) public onlyOwner{
      //Contract and PancakePair never can receive reflections
      require(account!=address(this)&&account!=pancakePair);
      //Burn wallet always receives reflections
      require(account!=address(0xdead));
      _excludeFromReflection(account,exclude);
      emit OnSetExcludedFromReflection(account,exclude);
}

function _excludeFromReflection(address account, bool exclude) private{
      require(ExcludedFromReflection[account]!=exclude);
      uint tokens=balanceOf(account);
      ExcludedFromReflection[account]=exclude;
      if(exclude){
         uint shares=Shares[account];
         _totalShares-=shares;
         Shares[account]=0;
         ExcludedBalances[account]=tokens;
         _totalExcludedTokens+=tokens;
      }else{
         ExcludedBalances[account]=0;
         _totalExcludedTokens-=tokens;
         uint shares=SharesFromTokens(tokens);
         Shares[account]=shares;
         _totalShares+=shares;
      }
}
```

● Contract owner can change buy fees up to 10%, sell fees up to 10% and transfer fees up to 10%

uint constant TAX_DENOMINATOR=10000;

```
function setTaxes(uint Buy, uint Sell, uint Transfer, uint Reflection, uint Liquidity, uint Marketing) public
onlyOwner{
      uint maxTax=TAX_DENOMINATOR/10;
      require(Buy<=maxTax&&Sell<=maxTax&&Transfer<=maxTax);
      require(Reflection+Liquidity+Marketing==TAX_DENOMINATOR);
      _buyTax=Buy;
      _sellTax=Sell;
      _transferTax=Transfer;
      _reflectionTax=Reflection;
      _liquidityTax=Liquidity;
      _marketingTax=Marketing;
      _contractTax=TAX_DENOMINATOR-_reflectionTax;
      emit OnSetTaxes(Buy, Sell, Transfer, Reflection, Liquidity, Marketing);
}
```

First minute after trade enablement, there is a special tax rate of 99% applied to buy transactions

uint constant AntiBotBuyTax=9999;

uint constant BotBuyTaxDuration=1 minutes;

```
function _getStartTax(uint duration, uint maxTax, uint minTax) private view returns (uint){
      uint timeSinceLaunch=block.timestamp-launchTimestamp;
      return maxTax-((maxTax-minTax)*timeSinceLaunch/duration);
}
```

```
transferWithFee function line 176-180

if(block.timestamp<launchTimestamp+BotBuyTaxDuration)
            tax=_getStartTax(BotBuyTaxDuration,AntiBotBuyTax,_buyTax);
         else
            tax=_buyTax;
```

This condition checks if the current timestamp is within a specific duration after the contract launch, defined by BotBuyTaxDuration. If the condition is true, it means that the transaction is occurring during the specified duration, and a special tax rate is applied to bot buy transactions.

● Contract owner can change marketingWallet address

Current value:

marketingWallet: 0xf1a522df5f627984772d8e4036e2880781199a7f

```
function SetMarketingWallet(address newMarketingWallet) public onlyOwner{
      marketingWallet=newMarketingWallet;
      emit OnSetMarketingWallet(newMarketingWallet);
}
```

● Contract owner has to call setLaunchInSeconds() function to enable trade

```
function setLaunchInSeconds(uint secondsUntilLaunch) public onlyOwner{
      setLaunchTimestamp(block.timestamp+secondsUntilLaunch);
}

function setLaunchTimestamp(uint Timestamp) public onlyOwner{
      require(block.timestamp<launchTimestamp);
      require(Timestamp>=block.timestamp);
      launchTimestamp=Timestamp;
      emit OnSetLaunchTimestamp(Timestamp);
}
```

● ReflectOwnerTokens function allows the owner of the contract to reflect a certain amount of their tokens

Reflecting tokens typically means redistributing a portion of the transaction fees or rewards generated by the contract back to token holders. The exact details of how reflection is implemented may vary depending on the specific contract and its tokenomics.

```
function ReflectOwnerTokens(uint amount) public onlyOwner{
      removeTokens(msg.sender,amount);
      reflectTokens(amount);
      emit Transfer(msg.sender,address(0),amount);
}

function reflectTokens(uint tokens) private {
      if(_totalShares==0) return;//if total shares=0 reflection dissapears into nothing
      TokensPerShare+=tokens*DividentMagnifier/_totalShares;
}
```

● Contract owner can change swap settings

uint constant TAX_DENOMINATOR=10000;

```
function setManualSwap(bool manual) public onlyOwner{
      _manualSwap=manual;
      emit onSetManualSwap(manual);
}

function setOverLiquifyTreshold(uint amount) public onlyOwner{
      require(amount<TAX_DENOMINATOR);
      _liquifyTreshold=amount;
      emit OnSetOverLiquifyTreshold(amount);
}

function setSwapTreshold(uint treshold) public onlyOwner{
      require(treshold<=TAX_DENOMINATOR/100);
      _swapTreshold=treshold;
      emit OnSetSwapTreshold(treshold);
}
```

Contract owner can withdraw tokens from smart contract

Native tokens excluded

```
function RescueTokens(address token) public onlyOwner{
      require(token!=address(this)&&token!=pancakePair);
      IBEP20(token).transfer(msg.sender,IBEP20(token).balanceOf(address(this)));
}
```

Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public onlyOwner {
      require(newOwner != address(0), "Ownable: new owner is the zero address");
      emit OwnershipTransferred(_owner, newOwner);
      _owner = newOwner;
}
```

Contract owner can renounce ownership

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

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:	10%
Sell fees:	10%
Transfer fees:	0%
Max TX:	N/A
Max Sell:	N/A

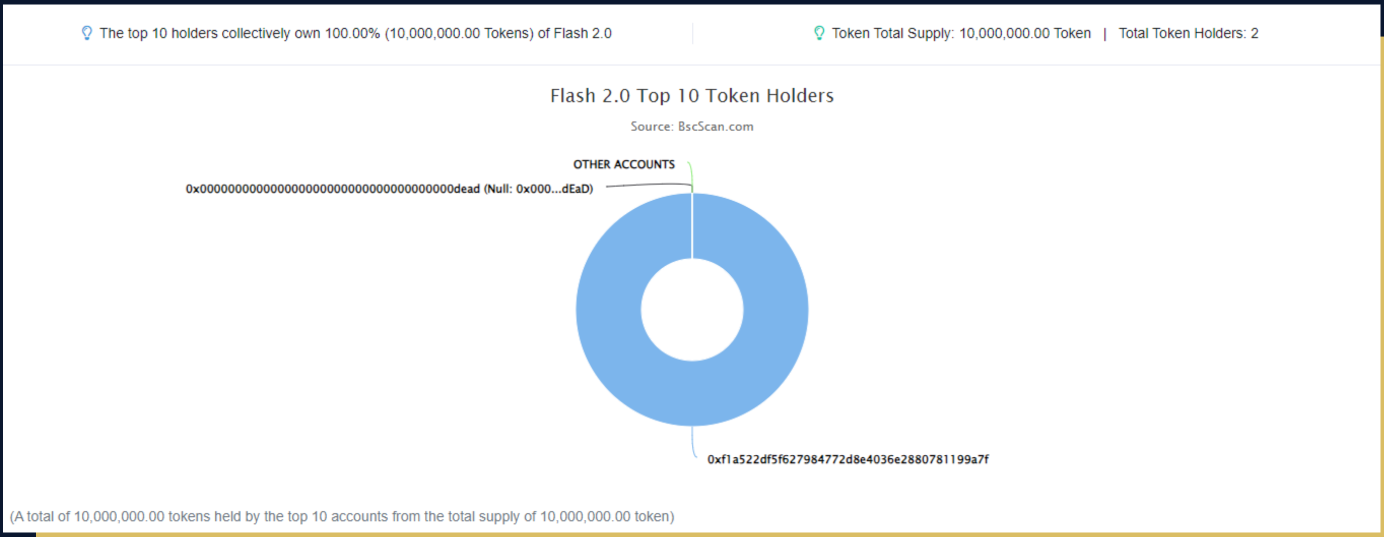
Honeypot Risk

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:	over 99% unlocked tokens

FLASH 2.0 TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS



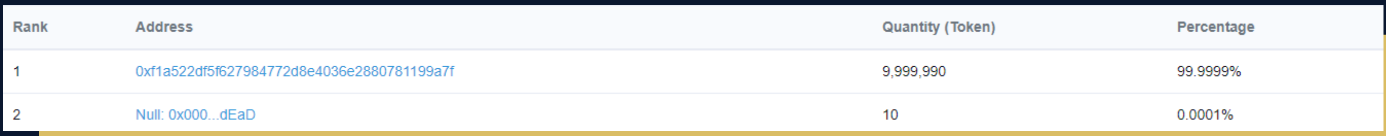
The top 10 holders collectively own 100.00% (10,000,000.00 Tokens) of Flash 2.0

Token Total Supply: 10,000,000.00 Token | Total Token Holders: 2

Flash 2.0 Top 10 Token Holders

Source: BscScan.com

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



Rank	Address	Quantity (Token)	Percentage
1	0xf1a522df5f627984772d8e4036e2880781199a7f	9,999,990	99.9999%
2	Null: 0x000...dEaD	10	0.0001%

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.