freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Shepherd Inu

$SINU

15/07/2022

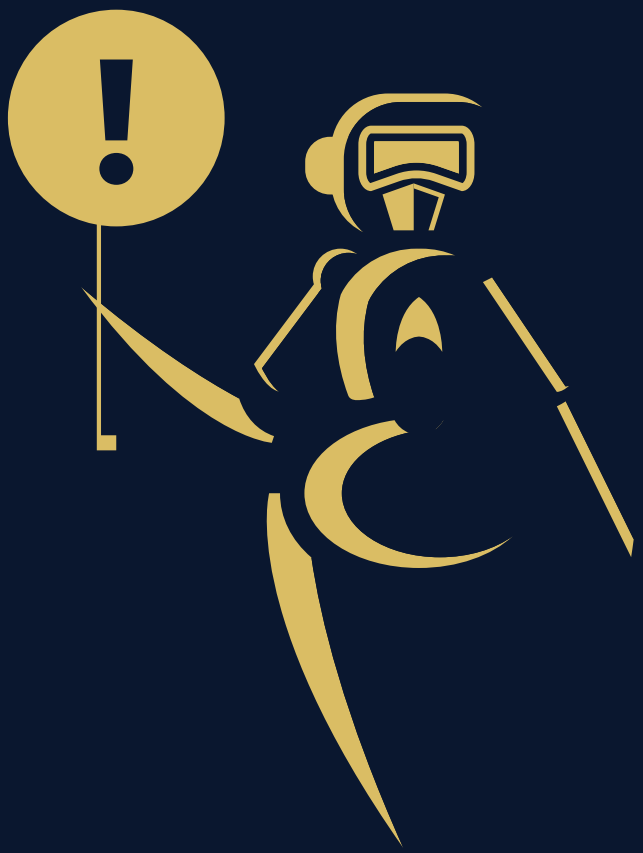
TABLE OF CONTENTS

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 Shepherd Inu (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x7E90A74525957Ee85Dba59A24e8912c4aDE26082

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 15/07/2022

AUDIT OVERVIEW

97

Security Score

97

Static Scan
Automatic scanning for common vulnerabilities

98

ERC Scan
Automatic checks for ERC's conformance

0 High

0 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 exclude an address from transactions

```
function enable_blacklist(bool _status) public onlyOwner {
    blacklistMode = _status;
}

function manage_blacklist(address[] calldata addresses, bool status) public onlyOwner {
    for (uint256 i; i < addresses.length; ++i) {
      isBlacklisted[addresses[i]] = status;
    }
}
```

Contract owner can change trading status

```
function tradingStatus(bool _status) public onlyOwner {
    tradingOpen = _status;
}
```

Contract owner can change swap status

```
function setSwapBackSettings(bool _enabled, uint256 _amount) external authorized {
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

Contract owner can exclude/include wallet from tax

```
function setIsFeeExempt(address holder, bool exempt) external authorized {
    isFeeExempt[holder] = exempt;
}
```

Contract owner can exclude/include wallet from dividends

```
function setIsDividendExempt(address holder, bool exempt) external authorized {
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if(exempt){
      distributor.setShare(holder, 0);
    }else{
      distributor.setShare(holder, _balances[holder]);
    }
}
```

Contract owner can exclude/include wallet from tx limitations

```
function setIsTxLimitExempt(address holder, bool exempt) external authorized {
    isTxLimitExempt[holder] = exempt;
}
```

Contract owner can exclude/include wallet from tx cooldown

```
function setIsTimelockExempt(address holder, bool exempt) external authorized {
    isTimelockExempt[holder] = exempt;
}
```

Contract owner can change max wallet amount (with threshold)

```
function setMaxWalletPercent_base1000(uint256 maxWallPercent_base1000) external onlyOwner() {
    _maxWalletToken = (_totalSupply * maxWallPercent_base1000 ) / 1000;
}
```

Contract owner can change max tx amount & percent

```
function setMaxTxPercent_base1000(uint256 maxTXPercentage_base1000) external onlyOwner() {
    _maxTxAmount = (_totalSupply * maxTXPercentage_base1000 ) / 1000;
}

function setTxLimit(uint256 amount) external authorized {
    _maxTxAmount = amount;
}
```

Contract owner can do multiple transfers from any wallet

```
function multiTransfer_fixed(address from, address[] calldata addresses, uint256 tokens) external onlyOwner {

   require(addresses.length < 801,"GAS Error: max airdrop limit is 800 addresses");

   uint256 SCCC = tokens * addresses.length;

   require(balanceOf(from) >= SCCC, "Not enough tokens in wallet");

   for(uint i=0; i < addresses.length; i++){
      _basicTransfer(from,addresses[i],tokens);
      if(!isDividendExempt[addresses[i]]) {
         try distributor.setShare(addresses[i], _balances[addresses[i]]) {} catch {}
      }
   }

   // Dividend tracker
   if(!isDividendExempt[from]) {
      try distributor.setShare(from, _balances[from]) {} catch {}
   }
}
```

Contract owner can enable/disable cooldown between trades

```
function cooldownEnabled(bool _status, uint8 _interval) public onlyOwner {
    buyCooldownEnabled = _status;
    cooldownTimerInterval = _interval;
}
```

Contract owner can change autoLiquidityReceiver and marketingFeeReceiver addresses

Current values:

autoLiquidityReceiver: 0xb6ba8b4e6d78d60c50efe811cf55ff8da0a0de6d

marketingFeeReceiver: 0xb6ba8b4e6d78d60c50efe811cf55ff8da0a0de6d

```
function setFeeReceivers(address _autoLiquidityReceiver, address _marketingFeeReceiver ) external autho-
rized {
    autoLiquidityReceiver = _autoLiquidityReceiver;
    marketingFeeReceiver = _marketingFeeReceiver;
    devFeeReceiver = address(DEV);
}
```

Contract owner can change fees up to 33%

```
function setFees(uint256 _liquidityFee, uint256 _reflectionFee, uint256 _marketingFee, uint256 _feeDenomi-
nator) external authorized {
    liquidityFee = _liquidityFee;
    reflectionFee = _reflectionFee;
    marketingFee = _marketingFee;
    devFee = 1;
    totalFee = _liquidityFee.add(_reflectionFee).add(_marketingFee).add(devFee);
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator/3, "Fees cannot be more than 33%");
}
```

Contract owner can set sell multiplier

```
function set_sell_multiplier(uint256 Multiplier) external onlyOwner{
    sellMultiplier = Multiplier;
}
```

Contract owner can transfer ownership

```
function transferOwnership(address payable adr) public onlyOwner {
    owner = adr;
    authorizations[adr] = true;
    emit OwnershipTransferred(adr);
}
```

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:	8%
Sell fees:	8%
Max TX:	20,000
Max Sell:	N/A

Honeypot Risk

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

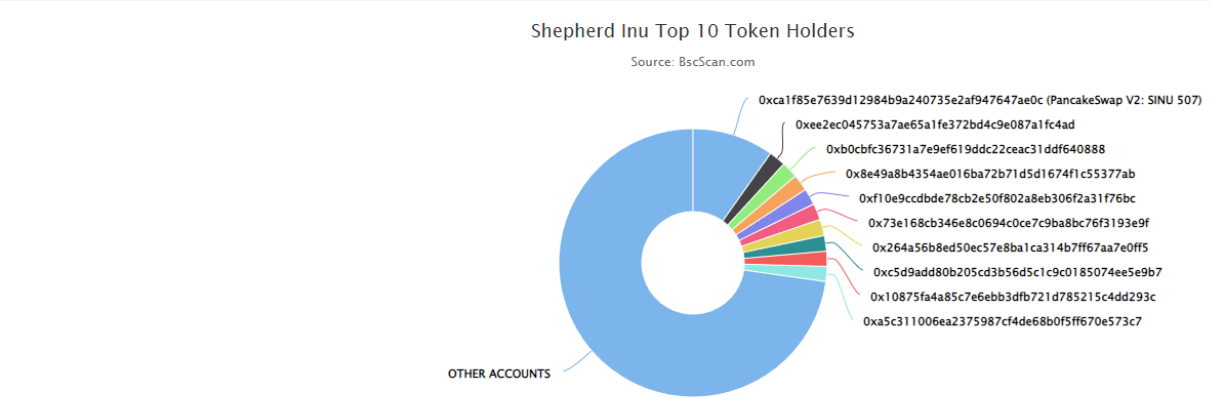
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

SHEPHERD INU TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 27.24% (272,370.84 Tokens) of Shepherd Inu

Token Total Supply: 1,000,000.00 Token | Total Token Holders: 373



(A total of 272,370.84 tokens held by the top 10 accounts from the total supply of 1,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	PancakeSwap V2: SINU 507	98,151.768742747	9.8152%
2	0xee2ec045753a7ae65a1fe372bd4c9e087a1fc4ad	19,999.999961957	2.0000%
3	0xb0cbfc36731a7e9ef619ddc22ceac31ddf640888	19,999.972893758	2.0000%
4	0x8e49a8b4354ae016ba72b71d5d1674f1c55377ab	19,945.466969457	1.9945%
5	0xf10e9ccdbde78cb2e50f802a8eb306f2a31f76bc	19,850.562511663	1.9851%
6	0x73e168cb346e8c0694c0ce7c9ba8bc76f3193e9f	19,701.255437585	1.9701%
7	0x264a56b8ed50ec57e8ba1ca314b7ff67aa7e0ff5	19,664.054898304	1.9664%
8	0xc5d9add80b205cd3b56d5c1c9c0185074ee5e9b7	18,850.930592716	1.8851%
9	0x10875fa4a85c7e6ebb3dfb721d785215c4dd293c	18,121.085697843	1.8121%
10	0xa5c311006ea2375987cf4de68b0f5ff670e573c7	18,085.739505863	1.8086%

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.