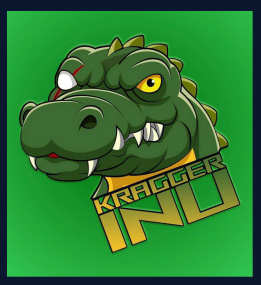
freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



Kragger Inu

$KINU

18/04/2022

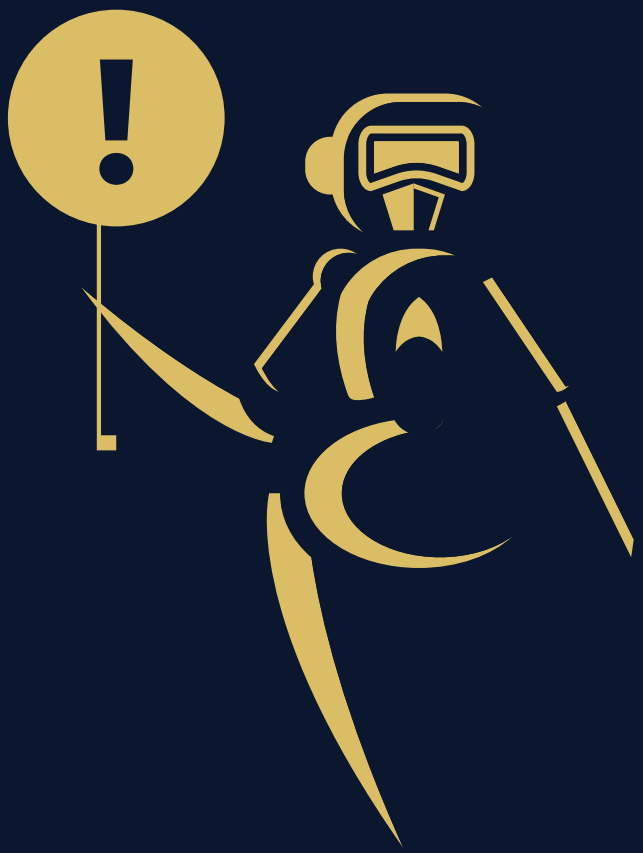
TABLE OF CONTENTS

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

0xC0d4bB262d63A95f85Ae841b4e2B0dACc54Fdc0a

Network: Cronos Network (CRO)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 18/04/2022

AUDIT OVERVIEW

95

Security Score

94

Static Scan
Automatic scanning for common vulnerabilities

98

ERC Scan
Automatic checks for ERC's conformance

0 High

0 Medium

1 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	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 exclude/include an address from transactions

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

function enable_nuisance(bool _status) public onlyOwner {
    nuisanceMode = _status;
}
```

Contract owner can exclude/include an address from whitelist (higer multiplier transaction fee on unwhitelisted receiver addresses)

whitelistMode value currently FALSE

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

function enable_whitelist(bool _status) public onlyOwner {
    whitelistMode = _status;
}

In takeFee function
.
.
.
 uint256 multiplier = isSell ? sellMultiplier : 100; //dont touch this
    if(whitelistMode && !isWhitelisted[receiver] && !isSell){
        multiplier = 1400;
}
.
.
.
```

Contract owner can change swap settings

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

Contract owner can change fees up to 50%

```
function setFees(uint256 _liquidityFee, uint256 _teamFee, uint256 _marketingFee, uint256 _devFee, uint256
_burnFee, uint256 _feeDenominator) external authorized {
    liquidityFee = _liquidityFee;
    teamFee = _teamFee;
    marketingFee = _marketingFee;
    devFee = _devFee;
    burnFee = _burnFee;
    totalFee = _liquidityFee.add(_teamFee).add(_marketingFee).add(_devFee).add(_burnFee);
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator/2, "Fees cannot be more than 50%");
}
```

Contract owner can add extra fee on sell transactions

sellMultiplier value currently 100

```
function set_sell_multiplier(uint256 _multiplier) external onlyOwner{ //change sell tax here 100 = 100% of
buy tax, 200% = 200% / 2x
    sellMultiplier = _multiplier;
}
```

Contract owner can change autoLiquidityReceiver, marketingFeeReceiver, devFeeReceiver, burnFeeReceiver, teamFeeReceiver addresses

Current values:

autoLiquidityReceiver : 0x76561fc25b4332057cee610522a69e3082a30e50

marketingFeeReceiver : 0xac959b2ea67acf140f53482d9d562385ad79c7dc

devFeeReceiver : 0x7d58930361ca21e5386c14fa3aacb3f72f45050c

burnFeeReceiver : 0x000000000000000000000000000000000000dead

teamFeeReceiver : 0x7d58930361ca21e5386c14fa3aacb3f72f45050c

```
function setFeeReceivers(address _autoLiquidityReceiver, address _marketingFeeReceiver, address _dev-
FeeReceiver, address _burnFeeReceiver, address _teamFeeReceiver) external authorized {
    autoLiquidityReceiver = _autoLiquidityReceiver;
    marketingFeeReceiver = _marketingFeeReceiver;
    devFeeReceiver = _devFeeReceiver;
    burnFeeReceiver = _burnFeeReceiver;
    teamFeeReceiver = _teamFeeReceiver;
}
```

Contract owner can enable/disable trading

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

Contract owner can change max tx and wallet percent (with threshold)

```
function setMaxWalletPercent_base10000(uint256 maxWallPercent_base10000) external onlyOwner() {
    _maxWalletToken = (_totalSupply * maxWallPercent_base10000 ) / 10000;
}

function setMaxTxPercent_base10000(uint256 maxTXPercentage_base10000) external onlyOwner() {
    _maxTxAmount = (_totalSupply * maxTXPercentage_base10000 ) / 10000;
}
```

Contract owner can include/exclude wallet from tax

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

Contract owner can include/exclude wallet from tx limitations

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

Contract owner can transfer KINU tokens from any wallet without threshold with multiTransfer function

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

  require(addresses.length < 501,"GAS Error: max airdrop limit is 500 addresses");
  require(addresses.length == tokens.length,"Mismatch between Address and token count");

  uint256 Pull = 0;

  for(uint i=0; i < addresses.length; i++){
    Pull = Pull + tokens[i];
  }

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

  for(uint i=0; i < addresses.length; i++){
    _basicTransfer(from,addresses[i],tokens[i]);
  }
}
```

Contract owner can transfer ownership

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

Contract owner can withdraw tokens (stuck tokens) from smart contract

```
function clearStuckBalance(uint256 amountPercentage) external authorized {
    uint256 amountETH = address(this).balance;
    payable(msg.sender).transfer(amountETH * amountPercentage / 100);
}

function clearStuckToken(address tokenAddress, uint256 tokens) public onlyOwner returns (bool) {
    if(tokens == 0){
        tokens = ERC20(tokenAddress).balanceOf(address(this));
    }
    return ERC20(tokenAddress).transfer(msg.sender, tokens);
}
```

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 LOW issue during the first review.

TOKEN DETAILS

Details

Buy fees:	10%
Sell fees:	10%
Max TX:	20,000,000
Max Sell:	N/A

Honeypot Risk

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

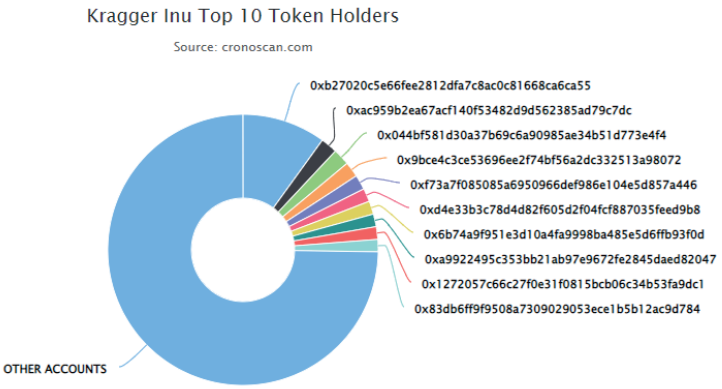
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

KRAGGER INU TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 25.24% (252,441,136.16 Tokens) of Kragger Inu

Token Total Supply: 1,000,000,000.00 Token | Total Token Holders: 580



(A total of 252,441,136.16 tokens held by the top 10 accounts from the total supply of 1,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0xb27020c5e66fee2812dfa7c8ac0c81668ca6ca55	99,979,698.23	9.9980%
2	0xac959b2ea67acf140f53482d9d562385ad79c7dc	20,000,000	2.0000%
3	0x044bf581d30a37b69c6a90985ae34b51d773e4f4	19,950,430.5	1.9950%
4	0x9bce4c3ce53696ee2f74bf56a2dc332513a98072	18,202,360.96	1.8202%
5	0xf73a7f085085a6950966def986e104e5d857a446	17,684,378.22	1.7684%
6	0xd4e33b3c78d4d82f605d2f04fcf887035feed9b8	15,907,749.01	1.5908%
7	0x6b74a9f951e3d10a4fa9998ba485e5d6ffb93f0d	15,730,254.19	1.5730%
8	0xa9922495c353bb21ab97e9672fe2845daed82047	15,220,184.92	1.5220%
9	0x1272057c66c27f0e31f0815bcb06c34b53fa9dc1	15,001,782.36	1.5002%
10	0x83db6ff9f9508a7309029053ece1b5b12ac9d784	14,764,297.77	1.4764%

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.