freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Burnny Inu

$BURNNY

04/11/2023

TOKEN OVERVIEW

Fees

- Buy fees: 4%
- Sell fees: 4%

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 and / or max wallet amount

Blacklist

- Blacklist function not detected

Other privileges

- Can exclude / include from fees
- Contract owner has to call enableTrading function to enable trade

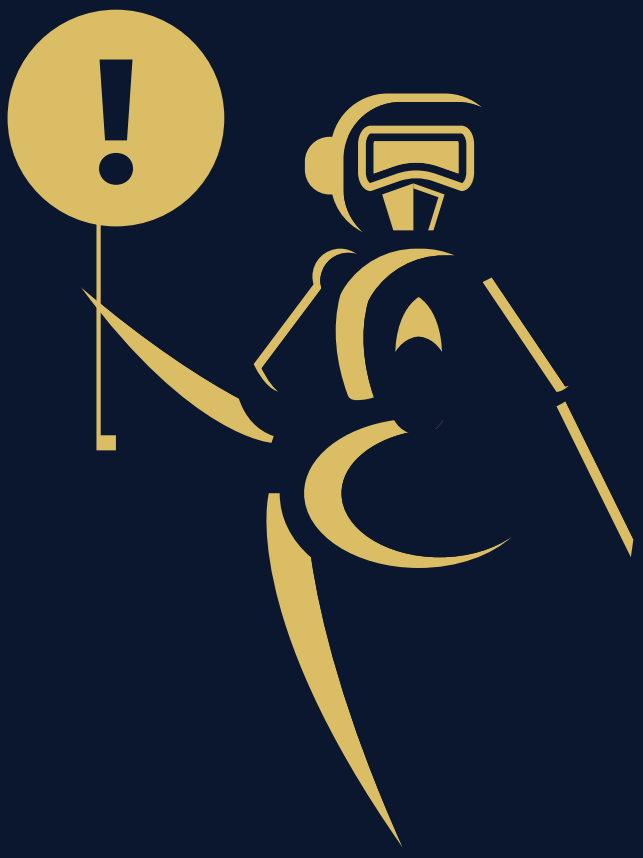
TABLE OF CONTENTS

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

0x4875b51147321d53Dd6D9197a4E96F663Fd04418

Network: Ethereum (ETH)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 29/11/2023

WEBSITE DIAGNOSTIC

https://burnnytoken.com/

0-49 50-89 90-100

Performance Accessibility Best Practices SEO Progressive Web App

Socials

Twitter

https://twitter.com/BurnnyToken

Telegram

https://t.me/BurnnyToken

AUDIT OVERVIEW

84

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

82

ERC Scan
Automatic checks for ERC's conformance

1 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	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 an address from transactions
- Contract owner can exclude/include wallet from tax

```
function excludeFromFee(address _address) external onlyOwner {
      require(exemptFee[_address] != true,"Account is already excluded");
      exemptFee[_address] = true;
     emit ExcludeFromFeeUpdated(_address);
}

function includeFromFee(address _address) external onlyOwner {
      require(exemptFee[_address] != false,"Account is already excluded");
      exemptFee[_address] = false;
     emit includeFromFeeUpdated(_address);
}
```

- Contract owner has to call enableTrading function to enable trade

Please note that any wallet excluded from fees retains the ability to engage in trading, even in situations where trading has been disabled

```
function enableTrading() external onlyOwner {
      require(!tradingEnabled, "Cannot re-enable trading");
      tradingEnabled = true;
      providingLiquidity = true;
     emit TradingOpenUpdated();
}

_transferFrom function line 456
.
.
.
if (!exemptFee[sender] && !exemptFee[recipient]) {
        require(tradingEnabled, "Trading not enabled");
}
.
.
.
```

- Contract owner can change swap settings

```
function setSwapBackEnable(bool state) external onlyOwner {
      providingLiquidity = state;
}
```

```
function setSwapTokens(uint256 new_amount) external onlyOwner {
        require(new_amount <= 4200000000, "Swap threshold amount should be lower or equal to 1% of
tokens");
        require(new_amount >= 420000000,  "Swap threshold amount should be greater than or equal to 0.1% of
tokens");
        swapTokens = new_amount * 10**decimals();
    }
```

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

Native tokens excluded

```
function recoverERCFromContract(address _tokenAddy, uint256 _amount) external onlyOwner {
        require(_tokenAddy != address(this), "Owner can't claim contract's balance of its own tokens");
        require(_amount > 0, "Amount should be greater than zero");
        require(_amount <= IBEP20(_tokenAddy).balanceOf(address(this)), "Insufficient Amount");
        IBEP20(_tokenAddy).transfer(marketingWallet1, _amount);
       emit ERC20TokensRecovered(_amount);
    }

    function recoverETHfromContract() external {
        uint256 contractETHBalance = address(this).balance;
        require(contractETHBalance > 0, "Amount should be greater than zero");
        require(contractETHBalance <= address(this).balance, "Insufficient Amount");
        payable(address(marketingWallet1)).transfer(contractETHBalance);
       emit ETHBalanceRecovered();
    }
```

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

```
function setBuyTaxes(uint256 _marketing1, uint256 _marketing2, uint256 _liquidity) external onlyOwner {
        buytaxes = Taxes(_marketing1, _marketing2, _liquidity);
        require((_marketing1 + _marketing2 + _liquidity) <= 10,"Must keep fees at 10% or less");
    }

    function setSellTaxes(uint256 _marketing1, uint256 _marketing2, uint256 _liquidity) external onlyOwner {
        sellTaxes = Taxes(_marketing1, _marketing2, _liquidity);
        require((_marketing1 + _marketing2 + _liquidity) <= 10,"Must keep fees at 10% or less");
    }
```

Contract owner can renounce ownership

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

Contract owner can transfer ownership

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

function _setOwner(address newOwner) private {
    address oldOwner = _owner;
    _owner = newOwner;
    emit OwnershipTransferred(oldOwner, newOwner);
}
```

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:	4%
Sell fees:	4%
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

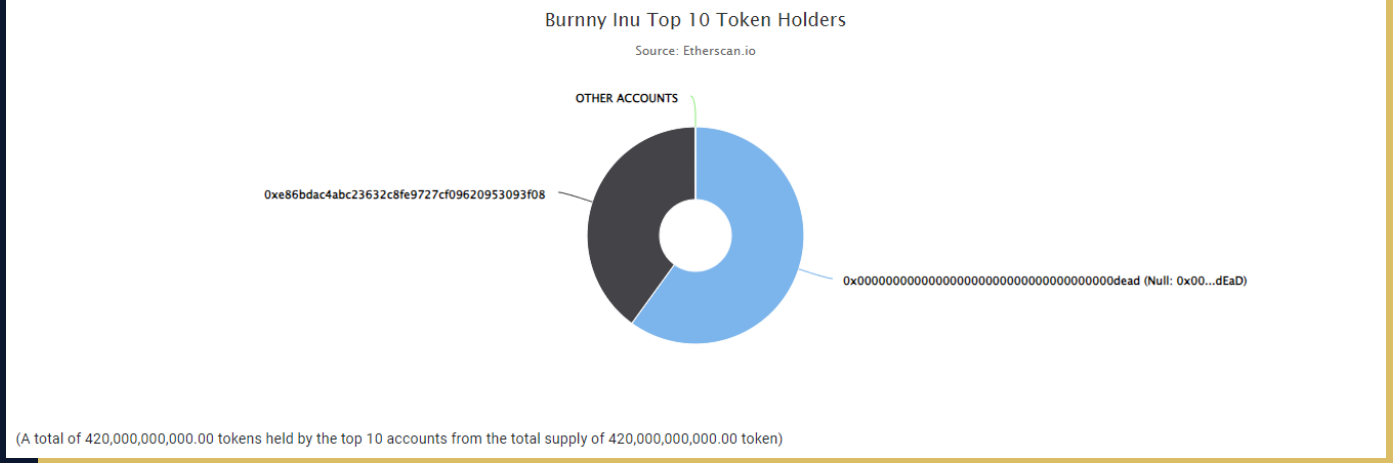
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

BURNNY TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (420,000,000,000.00 Tokens) of Burnny Inu

Token Total Supply: 420,000,000,000.00 Token | Total Token Holders: 2



(A total of 420,000,000,000.00 tokens held by the top 10 accounts from the total supply of 420,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	Null: 0x00...dEaD	252,014,700,000	60.0035%
2	0xe86BdA...53093f08	167,985,300,000	39.9965%

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.