BlockSAFU

ADVANCE MANUAL
SMART CONTRACT AUDIT

Project: Bitnou

Website: https://bitnou.com/

BlockSAFU Score:

46

Contract Address:

0x221e4c3BBabC3E33a8Be082c1f96037a9761a9F2

DISCLAMER

BlockSAFU has completed this report to provide a summary of the Smart Contract functions, and any security, dependency, or cybersecurity vulnerabilities. This is often a constrained report on our discoveries based on our investigation and understanding of the current programming versions as of this report's date. To understand the full scope of our analysis, it is vital for you to at the date of this report. To understand the full scope of our analysis, you need to review the complete report. Although we have done our best in conducting our investigation and creating this report, it is vital to note that you should not depend on this report and cannot make any claim against BlockSAFU or its Subsidiaries and Team members on the premise of what has or has not been included in the report. Please remember to conduct your independent examinations before making any investment choices. We do not provide investment advice or in any way claim to determine if the project will be successful or not.

By perusing this report or any portion of it, you concur to the terms of this disclaimer. In the unlikely situation where you do not concur with the terms, you should immediately terminate reading this report, and erase and discard any duplicates of this report downloaded and/or printed by you. This report is given for data purposes as it were and on a non-reliance premise and does not constitute speculation counsel. No one should have any right to depend on the report or its substance, and BlockSAFU and its members (including holding companies, shareholders, backups, representatives, chiefs, officers, and other agents) BlockSAFU and its subsidiaries owe no obligation of care towards you or any other person, nor does BlockSAFU make any guarantee or representation to any individual on the precision or completeness of the report.

ABOUT THE AUDITOR:

BlockSAFU (BSAFU) is an Anti-Scam Token Utility that reviews Smart Contracts and Token information to Identify Rug Pull and Honey Pot scamming activity. BlockSAFUs Development Team consists of several Smart Contract creators, Auditors Developers, and Blockchain experts. BlockSAFU provides solutions, prevents, and hunts down scammers. BSAFU is a utility token with features Audit, KYC, Token Generators, and Bounty Scammers. It will enrich the crypto ecosystem.

OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 7% (owner can’t set fees).
- Sell 7% (owner can’t set fees).

Tx Amount

- Owner cannot set a max tx amount.

Transfer Pausable

- Owner cannot pause.

Blacklist

- Owner cannot blacklist.

Ownership

- Owner cannot take back ownership.

Proxy

- This contract has no proxy.

Anti Whale

- Owner cannot limit the number of wallet holdings.

Trading Cooldown

- Owner cannot set the selling time interval.

Note: Staking contract is not audited

SMART CONTRACT REVIEW

Token Name	**BitnouCoin**
Token Symbol	**BNOU**
Token Decimal	18
Total Supply	10,000,000,000 **BNOU**
Contract Address	0x221e4c3BBabC3E33a8Be082c1f96037a9761a9F2
Deployer Address	0x6b2a856A8954aa86eA66f9729597f4078D03e7a9
Owner Address	0x47a4ea43c6cf05e2541a76903f06d4b24fa4cc81
Tax Fees Buy	7%
Tax Fees Sell	7%
Gas Used for Buy	*will be updated after the DEX listing*
Gas Used for Sell	*will be updated after the DEX listing*
Contract Created	Sep-06-2022 03:34:43 AM +UTC
Initial Liquidity	*will be updated after the DEX listing*
Liquidity Status	Locked
Unlocked Date	*will be updated after the DEX listing*
Verified CA	Yes
Compiler	v0.8.15+commit.e14f2714
Optimization	Yes with 200 runs
Sol License	MIT License
Top 5 Holders	*will be updated after the DEX listing*
Other	default evmVersion

TAX

BUY	7%	Address	**SELL**	7%
Burn Fee	1%	Will be automatic to burn	Burn Fee	1%
Staking Fee	3%	0x56f600852a0f3758f0bc38e047f988923e86771d **(Staking contract is not audited)**	Staking Fee	3%
Liquidity Fee	2%	Will be automatic add liquidity	Liquidity Fee	2%
Team Fee	1%	0xcae49179e8ec892bee725c008e5e091b8b82b9a2	Team Fee	1%

Token Metrics

Rank	Address	Quantity	Percentage	Analytics
1	0x56f600852a0f3758f0bc38e047f988923e86771d	8,000,000,000	80.0000%	
2	0x47a4ea43c6cf05e2541a76903f06d4b24fa4cc81	1,000,000,000	10.0000%	
3	0xa7b77e35b9f0bb0f0f842bceeec12abd8a501e49	400,000,000	4.0000%	
4	0xcae49179e8ec892bee725c008e5e091b8b82b9a2	300,000,000	3.0000%	
5	0xf8e20f15ebaa485975601ce06dcf347b8ad3819f	200,000,000	2.0000%	
6	0xc3e23d6006c5049bb4ad408d394767f0e431b7f6	100,000,000	1.0000%	

[Download CSV Export]

Team Review

The Bnou team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 18 people in their telegram group (count in audit date).

Official Website And Social Media

Website: https://bitnou.com/

Telegram Group: https://t.me/bitnouofficial_english

Discord: https://discord.com/invite/5Qb4bM7zYA

MANUAL CODE REVIEW

Minor-risk

1 minor-risk code issue found

Could be fixed, and will not bring problems.

1. The return value of an external transfer/transferFrom return value is checked. Recommendation: use SafeERC20, or ensure that the transfer/transferFrom return value is checked

```
function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);
```

Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

High-Risk

1 high-risk code issues found

Must be fixed, and will bring problems.

1. if staking contract or team wallet set to zero address, transaction will be honeypot

```
function _transfer(address from, address to, uint256 amount)
internal override
    {
        require(from != address(0), "BEP20: transfer from the zero
address");
        require(to != address(0), "BEP20: transfer to the zero
address");

        if(from != owner() && to != owner()) {
            require(amount <= maxTxAmount, "Transfer amount
exceeds the maxTxAmount.");
        }

        if(amount == 0) {
            super._transfer(from, to, 0);
            return;
        }

        uint256 contractTokenBalance = balanceOf(address(this));

        bool canSwap = contractTokenBalance >= swapTokensAtAmount;

        if( canSwap && !swapping && from != owner())
        {
            swapping = true;

            uint256 swapTokens =
contractTokenBalance.mul(LiquidityFee).div(totalFees);
            swapAndLiquify(swapTokens);

            uint256 safeTokens =
contractTokenBalance.mul(StakingFee).div(totalFees);
            super._transfer(address(this), stakingContract,
safeTokens);
```

```
            uint256 burnTokens =
contractTokenBalance.mul(BurnFee).div(totalFees);
            super._transfer(address(this), deadWallet,
burnTokens);

            uint256 teamTokens = balanceOf(address(this));
            super._transfer(address(this), teamWallet,
teamTokens);

            swapping = false;
        }

        bool takeFee = !swapping;

        // if any account belongs to _isExcludedFromFee account
then remove the fee
        if(_isExcludedFromFees[from] || _isExcludedFromFees[to]) {
            takeFee = false;
        }

        if(takeFee) {
            uint256 fees = amount.mul(totalFees).div(100);
            amount = amount.sub(fees);

            super._transfer(from, address(this), fees);
        }

        checkForWhale(amount, from, to);
        super._transfer(from, to, amount);
    }
```

Critical-Risk

0 critical-risk code issues found

Must be fixed, and will bring problems.

EXTRA NOTES SMART CONTRACT

1. IBEP20

```
interface IBEP20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns
(uint256);
    function transfer(address recipient, uint256 amount) external
returns (bool);
    function allowance(address owner, address spender) external
view returns (uint256);
    function approve(address spender, uint256 amount) external
returns (bool);
    function transferFrom(address sender, address recipient,
uint256 amount) external returns (bool);
    event Transfer(address indexed from, address indexed to,
uint256 value);
    event Approval(address indexed owner, address indexed spender,
uint256 value);
}
```

IBEP20 Normal Base Template

2. SafeMath Contract

```
library SafeMath {
...
    function add(uint256 a, uint256 b) internal pure returns
(uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");
        return c;
    }
...
    function sub(uint256 a, uint256 b, string memory errorMessage)
internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }
    /**
     * @dev Returns the multiplication of two unsigned integers,
reverting on
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
     *
     * - Multiplication cannot overflow.
     */
  ...
    function mod(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        unchecked {
            require(b > 0, errorMessage);
            return a % b;
        }
    }
}
```

Standard Safemath contract

3. BnouCoin Contract

```
contract BitnouCoin is BEP20 , Ownable{
    using SafeMath for uint256;

    IUniswapV2Router02 public uniswapV2Router;
    address public  uniswapV2Pair;

    bool private swapping;

    address public constant deadWallet =
address(0x000000000000000000000000000000000000dEaD);

    uint256 public constant _totalSupply = 10_000_000_000 *
(10**18);
    uint256 public constant maxTxAmount = 100_000_000 * (10**18);
    uint256 public constant maxLimit =  100_000_000 * (10**18);
    uint256 public swapTokensAtAmount = 10_000 * (10**18);

    uint256 public constant LiquidityFee = 2;
    uint256 public constant StakingFee = 3;
    uint256 public constant BurnFee = 1;
    uint256 public constant TeamFee = 1;

    address public stakingContract;
    address public teamWallet;

    uint256 public totalFees =
LiquidityFee.add(StakingFee).add(BurnFee).add(TeamFee);

    mapping (address => bool) private _isExcludedFromFees;
    mapping (address => bool) private _isExcludedFromWhale;

    event UpdateUniswapV2Router(address indexed newAddress,
address indexed oldAddress);
    event ExcludeMultipleAccountsFromFees(address[] accounts, bool
isExcluded);

    event SwapAndLiquify(uint256 tokensSwapped, uint256
ethReceived, uint256 tokensIntoLiqudity);

    constructor(address _initializer) BEP20("BitnouCoin", "BNOU")
```

```
{

        IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
         // Create a uniswap pair for this new token
        address _uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory())
            .createPair(address(this), _uniswapV2Router.WETH());

        uniswapV2Router = _uniswapV2Router;
        uniswapV2Pair = _uniswapV2Pair;

        // exclude from paying fees or having max transaction
amount
        excludeFromFees(owner(), true);
        excludeFromFees(address(this), true);
        excludeFromFees(_initializer, true);
        //exclude from whale.
        excludeWalletsFromWhales();
        _isExcludedFromWhale[_initializer]=true;
        /*
            _mint is an internal function in BEP20.sol that is
only called here,
            and CANNOT be called ever again
        */
        _mint(_initializer, _totalSupply);
        _transferOwnership(_initializer);
    }

    receive() external payable { }

    function updateUniswapV2Router(address newAddress) public
onlyOwner {
        require(newAddress != address(uniswapV2Router),
"BitnouCoin: The router already has that address");
        emit UpdateUniswapV2Router(newAddress,
address(uniswapV2Router));
        uniswapV2Router = IUniswapV2Router02(newAddress);
        address _uniswapV2Pair =
IUniswapV2Factory(uniswapV2Router.factory())
            .createPair(address(this), uniswapV2Router.WETH());
        uniswapV2Pair = _uniswapV2Pair;
```

```
    }

    function excludeFromFees(address account, bool excluded)
public onlyOwner {
        require(_isExcludedFromFees[account] != excluded,
"BitnouCoin: Account is already the value of 'excluded'");
        _isExcludedFromFees[account] = excluded;
    }

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

    function setSwapTokensAtAmount(uint256 _value) external
onlyOwner
    {
        swapTokensAtAmount = _value;
    }

    function isExcludedFromFees(address account) public view
returns(bool) {
        return _isExcludedFromFees[account];
    }

    function _transfer(address from, address to, uint256 amount)
internal override
    {
        require(from != address(0), "BEP20: transfer from the zero
address");
        require(to != address(0), "BEP20: transfer to the zero
address");

        if(from != owner() && to != owner()) {
            require(amount <= maxTxAmount, "Transfer amount
exceeds the maxTxAmount.");
        }

        if(amount == 0) {
```

```
            super._transfer(from, to, 0);
            return;
        }

        uint256 contractTokenBalance = balanceOf(address(this));

        bool canSwap = contractTokenBalance >= swapTokensAtAmount;

        if( canSwap && !swapping && from != owner())
        {
            swapping = true;

            uint256 swapTokens =
contractTokenBalance.mul(LiquidityFee).div(totalFees);
            swapAndLiquify(swapTokens);

            uint256 safeTokens =
contractTokenBalance.mul(StakingFee).div(totalFees);
            super._transfer(address(this), stakingContract,
safeTokens);

            uint256 burnTokens =
contractTokenBalance.mul(BurnFee).div(totalFees);
            super._transfer(address(this), deadWallet,
burnTokens);

            uint256 teamTokens = balanceOf(address(this));
            super._transfer(address(this), teamWallet,
teamTokens);

            swapping = false;
        }

        bool takeFee = !swapping;

        // if any account belongs to _isExcludedFromFee account
then remove the fee
        if(_isExcludedFromFees[from] || _isExcludedFromFees[to]) {
            takeFee = false;
        }

        if(takeFee) {
```

```
            uint256 fees = amount.mul(totalFees).div(100);
            amount = amount.sub(fees);

            super._transfer(from, address(this), fees);
        }

        checkForWhale(amount, from, to);
        super._transfer(from, to, amount);
    }

    function swapAndLiquify(uint256 tokens) private
    {
        uint256 halfLiquidityTokens = tokens.div(2);
        uint256 swapableTokens = tokens.sub(halfLiquidityTokens);
        uint256 initialBalance = address(this).balance;
        swapTokensForEth(swapableTokens);
        // how much ETH did we just swap into?
        uint256 bnbForLiquidity =
address(this).balance.sub(initialBalance);
        // add liquidity to uniswap
        addLiquidity(halfLiquidityTokens, bnbForLiquidity);
        emit SwapAndLiquify(halfLiquidityTokens, bnbForLiquidity,
halfLiquidityTokens);
    }

    function swapTokensForEth(uint256 tokenAmount) private {
        // generate the uniswap pair path of token -> weth
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();

        _approve(address(this), address(uniswapV2Router),
tokenAmount);

        // make the swap

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
(
            tokenAmount,
            0, // accept any amount of ETH
            path,
            address(this),
```

```
            block.timestamp
        );

    }

    function addLiquidity(uint256 tokenAmount, uint256 ethAmount)
private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(uniswapV2Router),
tokenAmount);

        // add the liquidity
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            address(0),
            block.timestamp
        );
    }

    function setFeeReceivers(
        address _stakingContract,
        address _teamWallet
    ) external onlyOwner {
        stakingContract = _stakingContract;
        teamWallet = _teamWallet;
    }

    //////--------  ANTI WHALE -------////--------
    function excludeWalletsFromWhales() private
    {
        _isExcludedFromWhale[owner()]=true;
        _isExcludedFromWhale[address(this)]=true;
        _isExcludedFromWhale[address(0)]=true;
        _isExcludedFromWhale[uniswapV2Pair]=true;
    }

    function checkForWhale(uint amount, address from, address to)
    private view
    {
```

```
        uint256 newBalance = balanceOf(to).add(amount);
        if(!_isExcludedFromWhale[from] &&
!_isExcludedFromWhale[to])
        {
            require(newBalance <= maxLimit, "Exceeding max tokens
limit in the wallet");
        }
    }

    function setExcludedFromWhale(address account, bool _enabled)
public onlyOwner
    {
        _isExcludedFromWhale[account] = _enabled;
    }
}
```

READ CONTRACT (ONLY NEED TO KNOW)

1. BurnFee
1 uint256
(Function for read burn fee)

2. LiquidityFee
2 uint256
(Function for read Liquidity fee)

3. StakingFee
3 uint256
(Function for read Staking fee)

4. TeamFee
1 uint256
(Function for read Team fee)

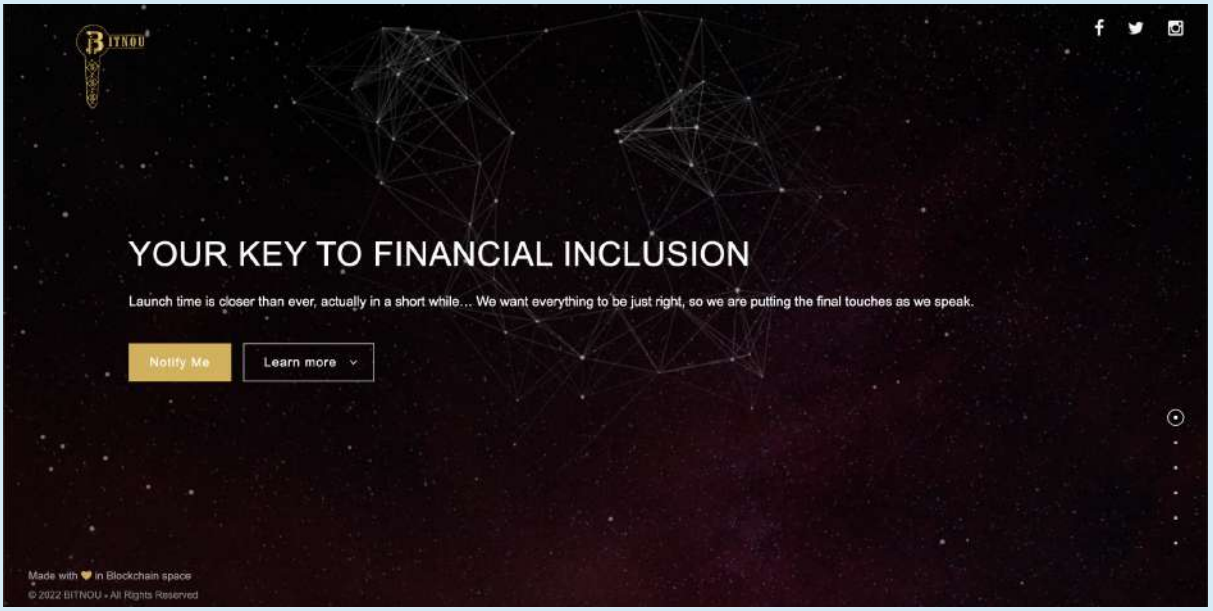
7. name
BnouCoin string
(Function for read Token name)

WRITE CONTRACT

1. renounceOwnership
(Renouncing ownership will leave the contract without an owner, thereby removing any functionality that is only available to the owner)

2. transferOwnership
newOwner (address)
(Its function is to change the owner)

WEBSITE REVIEW



- **Mobile Friendly**
- **Contains no code error**
- **SSL Secured (By Let's Encrypt SSL)**

Web-Tech stack: Apache, Bootstrap, Animate css

Domain .com (Hostgator) - Tracked by whois

First Contentful Paint:	1.6s
Fully Loaded Time	5.6s
Performance	46%
Accessibility	79%
Best Practices	58%
SEO	80%

RUG-PULL REVIEW

Based on the available information analyzed by us, we come to the following conclusions:

- Locked Liquidity (Locked by pinksale)

  (*Will be updated after DEX listing*)

- TOP 5 Holder.

  (*Will be updated after DEX listing*)

- The Team KYC by Blocksafu

HONEYPOT REVIEW

- Ability to sell.
- The owner is not able to pause the contract.
- The owner can’t set fees

Note: Please check the disclaimer above and note, that the audit makes no statements or warranties on the business model, investment attractiveness, or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by the project owner.