BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT

Project: ShibaLand

Website: https://www.theshibaland.com/

PASSED

BlockSAFU Score:

82

Contract Address:

0x27713499A72327284bE9fc1F639fF4B191B50d0C

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 to 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

OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 13% (owner can’t set fees over 25%).
- Sell 13% (owner can’t set fees over 25%).

Tx Amount

- Owner cannot set 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.

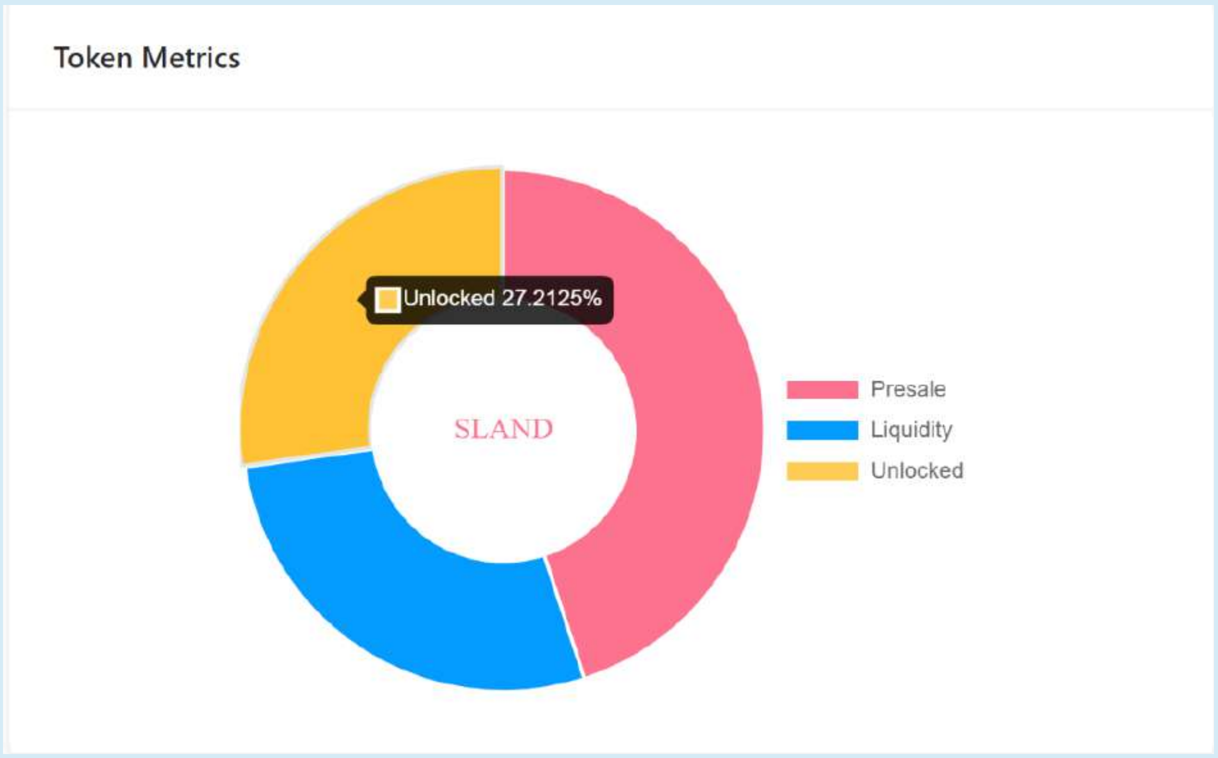
SMART CONTRACT REVIEW

Token Name	**ShibaLand**
Token Symbol	**SLAND**
Token Decimal	9
Total Supply	1,000,000,000 **SLAND**
Contract Address	0x27713499A72327284bE9fc1F639fF4B191B50d0C
Deployer Address	0x2b2273D504832595d4300122Ef68147D124e9D0E
Owner Address	0x2b2273D504832595d4300122Ef68147D124e9D0E
Tax Fees Buy	13%
Tax Fees Sell	13%
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-05-2022 01:06:31 PM +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	v0.8.4+commit.c7e474f2
Optimization	Yes with 200 runs
Sol License	MIT License
Top 5 Holders	*will be updated after the DEX listing*
Other	default evmVersion

TAX

BUY	13%	Address	**SELL**	13%
Buyback Fee	3%	Will be automatic to buyback	Buyback Fee	3%
Marketing Fee	2%	0x2b2273d504832595d4300122ef68147d124e9d0	Marketing Fee	2%
Liquidity Fee	5%	Will be automatic add liquidity	Liquidity Fee	5%
Reflection Fee	3%	will be automatic distribute	Reflection Fee	3%

Token Metrics



Team Review

The ShibaLand team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 121 people in their telegram group (count in audit date).

Official Website And Social Media

Website: https://www.theshibaland.com/

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

Twitter: https://twitter.com/shiblandtoken

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

0 high-risk code issues found

Must be fixed, and will bring problem.

🔴 Critical-Risk

0 critical-risk code issues found

Must be fixed, and will bring problem.

EXTRA NOTES SMART CONTRACT

1. IERC20

```
interface IERC20Extended {
    function totalSupply() external view returns (uint256);

    function decimals() external view returns (uint8);

    function symbol() external view returns (string memory);

    function name() external view returns (string memory);

    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
    );
}
```

IERC20 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. ShibaLand Contract

```
contract BuybackBabyToken is IERC20Extended, Auth, BaseToken {
    using SafeMath for uint256;

    uint256 public constant VERSION = 1;

    address private constant DEAD = address(0xdead);
    address private constant ZERO = address(0);
    uint8 private constant _decimals = 9;

    string private _name;
    string private _symbol;
    uint256 private _totalSupply;

    address public rewardToken;
    IUniswapV2Router02 public router;
    address public pair;
    address public autoLiquidityReceiver;
    address public marketingFeeReceiver;

    uint256 public liquidityFee; // default: 200
    uint256 public buybackFee; // default: 300
    uint256 public reflectionFee; // default: 800
    uint256 public marketingFee; // default: 100
    uint256 public totalFee;
    uint256 public feeDenominator; // default: 10000

    uint256 public targetLiquidity; // default: 25
    uint256 public targetLiquidityDenominator; // default: 100

    uint256 public buybackMultiplierNumerator; // default: 200
    uint256 public buybackMultiplierDenominator; // default: 100
    uint256 public buybackMultiplierTriggeredAt;
    uint256 public buybackMultiplierLength; // default: 30 mins

    bool public autoBuybackEnabled;

    uint256 public autoBuybackCap;
    uint256 public autoBuybackAccumulator;
    uint256 public autoBuybackAmount;
    uint256 public autoBuybackBlockPeriod;
    uint256 public autoBuybackBlockLast;

    DividendDistributor public distributor;

    uint256 public distributorGas;

    bool public swapEnabled;
    uint256 public swapThreshold;

    mapping(address => uint256) private _balances;
    mapping(address => mapping(address => uint256)) private
_allowances;

    mapping(address => bool) public buyBacker;
    mapping(address => bool) public isFeeExempt;
    mapping(address => bool) public isDividendExempt;

    event AutoLiquify(uint256 amountBNB, uint256 amountBOG);
    event BuybackMultiplierActive(uint256 duration);

    bool inSwap;
    modifier swapping() {
        inSwap = true;
        _;
        inSwap = false;
    }

    modifier onlyBuybacker() {
        require(buyBacker[msg.sender] == true, "Not a buybacker");
        _;
    }

    constructor(
        string memory name_,
        string memory symbol_,
        uint256 totalSupply_,
        address rewardToken_,
        address router_,
        uint256[5] memory feeSettings_,
        address serviceFeeReceiver_,
        uint256 serviceFee_
    ) payable Auth(msg.sender) {
        _name = name_;
        _symbol = symbol_;
        _totalSupply = totalSupply_;

        rewardToken = rewardToken_;
        router = IUniswapV2Router02(router_);
        pair = IUniswapV2Factory(router.factory()).createPair(
            address(this),
            router.WETH()
        );
        distributor = new DividendDistributor(rewardToken_,
router_);

        _initializeFees(feeSettings_);
        _initializeLiquidityBuyBack();

        distributorGas = 500000;
        swapEnabled = true;
        swapThreshold = _totalSupply / 20000; // 0.005%

        isFeeExempt[msg.sender] = true;
        isDividendExempt[pair] = true;
        isDividendExempt[address(this)] = true;
        isDividendExempt[DEAD] = true;
        buyBacker[msg.sender] = true;

        autoLiquidityReceiver = msg.sender;
        marketingFeeReceiver = msg.sender;

        _allowances[address(this)][address(router)] =
_totalSupply;
        _allowances[address(this)][address(pair)] = _totalSupply;

        _balances[msg.sender] = _totalSupply;
        emit Transfer(address(0), msg.sender, _totalSupply);

        emit TokenCreated(
            msg.sender,
            address(this),
            TokenType.buybackBaby,
            VERSION
        );
        payable(serviceFeeReceiver_).transfer(serviceFee_);
    }

    function _initializeFees(uint256[5] memory feeSettings_)
internal {
        _setFees(
            feeSettings_[0], // liquidityFee
            feeSettings_[1], // buybackFee
            feeSettings_[2], // reflectionFee
            feeSettings_[3], // marketingFee
            feeSettings_[4] // feeDenominator
        );
    }

    function _initializeLiquidityBuyBack() internal {
        targetLiquidity = 25;
        targetLiquidityDenominator = 100;

        buybackMultiplierNumerator = 200;
        buybackMultiplierDenominator = 100;
        buybackMultiplierLength = 30 minutes;
    }

    receive() external payable {}

    function totalSupply() external view override returns
(uint256) {
        return _totalSupply;
    }

    function decimals() external pure override returns (uint8) {
        return _decimals;
    }

    function symbol() external view override returns (string
memory) {
        return _symbol;
    }

    function name() external view override returns (string memory)
{
        return _name;
    }

    function balanceOf(address account) public view override
returns (uint256) {
        return _balances[account];
    }

    function allowance(address holder, address spender)
        external
        view
        override
        returns (uint256)
    {
        return _allowances[holder][spender];
    }

    function approve(address spender, uint256 amount)
        public
        override
        returns (bool)
    {
        _allowances[msg.sender][spender] = amount;
        emit Approval(msg.sender, spender, amount);
        return true;
    }

    function approveMax(address spender) external returns (bool) {
        return approve(spender, _totalSupply);
    }

    function transfer(address recipient, uint256 amount)
        external
        override
        returns (bool)
    {
        return _transferFrom(msg.sender, recipient, amount);
    }

    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external override returns (bool) {
        if (_allowances[sender][msg.sender] != _totalSupply) {
            _allowances[sender][msg.sender] =
_allowances[sender][msg.sender]
                .sub(amount, "Insufficient Allowance");
        }

        return _transferFrom(sender, recipient, amount);
    }

    function _transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) internal returns (bool) {
        if (inSwap) {
            return _basicTransfer(sender, recipient, amount);
        }

        if (shouldSwapBack()) {
            swapBack();
        }
        if (shouldAutoBuyback()) {
            triggerAutoBuyback();
        }

        _balances[sender] = _balances[sender].sub(
            amount,
            "Insufficient Balance"
        );

        uint256 amountReceived = shouldTakeFee(sender)
            ? takeFee(sender, recipient, amount)
            : amount;

        _balances[recipient] =
_balances[recipient].add(amountReceived);

        if (!isDividendExempt[sender]) {
            try distributor.setShare(sender, _balances[sender]) {}
catch {}
```

```
        }
        if (!isDividendExempt[recipient]) {
            try
                distributor.setShare(recipient,
_balances[recipient])
            {} catch {}
        }

        try distributor.process(distributorGas) {} catch {}

        emit Transfer(sender, recipient, amountReceived);
        return true;
    }

    function _basicTransfer(
        address sender,
        address recipient,
        uint256 amount
    ) internal returns (bool) {
        _balances[sender] = _balances[sender].sub(
            amount,
            "Insufficient Balance"
        );
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
        return true;
    }

    function shouldTakeFee(address sender) internal view returns
(bool) {
        return !isFeeExempt[sender];
    }

    function getTotalFee(bool selling) public view returns
(uint256) {
        if (selling) {
            return getMultipliedFee();
        }
        return totalFee;
    }

    function getMultipliedFee() public view returns (uint256) {
        if (

buybackMultiplierTriggeredAt.add(buybackMultiplierLength) >
            block.timestamp
        ) {
            uint256 remainingTime = buybackMultiplierTriggeredAt
                .add(buybackMultiplierLength)
                .sub(block.timestamp);
            uint256 feeIncrease = totalFee
                .mul(buybackMultiplierNumerator)
                .div(buybackMultiplierDenominator)
                .sub(totalFee);
            return
                totalFee.add(

feeIncrease.mul(remainingTime).div(buybackMultiplierLength)
                );
        }
        return totalFee;
    }

    function takeFee(
        address sender,
        address receiver,
        uint256 amount
    ) internal returns (uint256) {
        uint256 feeAmount = amount.mul(getTotalFee(receiver ==
pair)).div(
            feeDenominator
        );

        _balances[address(this)] =
_balances[address(this)].add(feeAmount);
        emit Transfer(sender, address(this), feeAmount);

        return amount.sub(feeAmount);
    }

    function shouldSwapBack() internal view returns (bool) {
        return
            msg.sender != pair &&
            !inSwap &&
            swapEnabled &&
            _balances[address(this)] >= swapThreshold;
    }

    function swapBack() internal swapping {
        uint256 dynamicLiquidityFee = isOverLiquified(
            targetLiquidity,
            targetLiquidityDenominator
        )
            ? 0
            : liquidityFee;
        uint256 amountToLiquify = swapThreshold
            .mul(dynamicLiquidityFee)
            .div(totalFee)
            .div(2);
        uint256 amountToSwap = swapThreshold.sub(amountToLiquify);

        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = router.WETH();
        uint256 balanceBefore = address(this).balance;

        router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            amountToSwap,
            0,
            path,
            address(this),
            block.timestamp
        );

        uint256 amountBNB =
address(this).balance.sub(balanceBefore);

        uint256 totalBNBFee =
totalFee.sub(dynamicLiquidityFee.div(2));

        uint256 amountBNBLiquidity = amountBNB
            .mul(dynamicLiquidityFee)
            .div(totalBNBFee)
            .div(2);
        uint256 amountBNBReflection =
amountBNB.mul(reflectionFee).div(
            totalBNBFee
        );
        uint256 amountBNBMarketing =
amountBNB.mul(marketingFee).div(
            totalBNBFee
        );

        try distributor.deposit{ value: amountBNBReflection }() {}
catch {}

payable(marketingFeeReceiver).transfer(amountBNBMarketing);

        if (amountToLiquify > 0) {
            router.addLiquidityETH{ value: amountBNBLiquidity }(
                address(this),
                amountToLiquify,
                0,
                0,
                autoLiquidityReceiver,
                block.timestamp
            );
            emit AutoLiquify(amountBNBLiquidity, amountToLiquify);
        }
    }

    function shouldAutoBuyback() internal view returns (bool) {
        return
            msg.sender != pair &&
            !inSwap &&
            autoBuybackEnabled &&
            autoBuybackBlockLast + autoBuybackBlockPeriod <=
block.number && // After N blocks from last buyback
            address(this).balance >= autoBuybackAmount;
    }

    function triggerZeusBuyback(uint256 amount, bool
triggerBuybackMultiplier)
        external
        authorized
    {
        buyTokens(amount, DEAD);
        if (triggerBuybackMultiplier) {
            buybackMultiplierTriggeredAt = block.timestamp;
            emit BuybackMultiplierActive(buybackMultiplierLength);
        }
    }

    function clearBuybackMultiplier() external authorized {
        buybackMultiplierTriggeredAt = 0;
    }

    function triggerAutoBuyback() internal {
        buyTokens(autoBuybackAmount, DEAD);
        autoBuybackBlockLast = block.number;
        autoBuybackAccumulator =
autoBuybackAccumulator.add(autoBuybackAmount);
        if (autoBuybackAccumulator > autoBuybackCap) {
            autoBuybackEnabled = false;
        }
    }

    function buyTokens(uint256 amount, address to) internal
swapping {
        address[] memory path = new address[](2);
        path[0] = router.WETH();
        path[1] = address(this);

        router.swapExactETHForTokensSupportingFeeOnTransferTokens{
            value: amount
        }(0, path, to, block.timestamp);
    }

    function setAutoBuybackSettings(
        bool _enabled,
        uint256 _cap,
        uint256 _amount,
        uint256 _period
    ) external authorized {
        autoBuybackEnabled = _enabled;
        autoBuybackCap = _cap;
        autoBuybackAccumulator = 0;
        autoBuybackAmount = _amount;
        autoBuybackBlockPeriod = _period;
        autoBuybackBlockLast = block.number;
    }

    function setBuybackMultiplierSettings(
        uint256 numerator,
        uint256 denominator,
        uint256 length
    ) external authorized {
        require(numerator / denominator <= 2 && numerator >
denominator);
        buybackMultiplierNumerator = numerator;
        buybackMultiplierDenominator = denominator;
        buybackMultiplierLength = length;
    }

    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]);
        }
    }

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

    function setBuyBacker(address acc, bool add) external
authorized {
        buyBacker[acc] = add;
    }

    function setFees(
        uint256 _liquidityFee,
        uint256 _buybackFee,
        uint256 _reflectionFee,
        uint256 _marketingFee,
        uint256 _feeDenominator
    ) public authorized {
        _setFees(
            _liquidityFee,
            _buybackFee,
            _reflectionFee,
            _marketingFee,
            _feeDenominator
        );
    }

    function _setFees(
        uint256 _liquidityFee,
        uint256 _buybackFee,
        uint256 _reflectionFee,
        uint256 _marketingFee,
        uint256 _feeDenominator
    ) internal {
        liquidityFee = _liquidityFee;
        buybackFee = _buybackFee;
        reflectionFee = _reflectionFee;
        marketingFee = _marketingFee;
        totalFee =
_liquidityFee.add(_buybackFee).add(_reflectionFee).add(
            _marketingFee
        );
        feeDenominator = _feeDenominator;
        require(
            totalFee < feeDenominator / 4,
            "Total fee should not be greater than 1/4 of fee
denominator"
        );
    }

    function setFeeReceivers(
        address _autoLiquidityReceiver,
        address _marketingFeeReceiver
    ) external authorized {
        autoLiquidityReceiver = _autoLiquidityReceiver;
        marketingFeeReceiver = _marketingFeeReceiver;
    }
    function setSwapBackSettings(bool _enabled, uint256 _amount)
        external
        authorized
    {
        swapEnabled = _enabled;
        swapThreshold = _amount;
    }

    function setTargetLiquidity(uint256 _target, uint256
_denominator)
        external
        authorized
    {
        targetLiquidity = _target;
        targetLiquidityDenominator = _denominator;
    }

    function setDistributionCriteria(
        uint256 _minPeriod,
        uint256 _minDistribution
    ) external authorized {
        distributor.setDistributionCriteria(_minPeriod,
_minDistribution);
    }
    function setDistributorSettings(uint256 gas) external
authorized {
        require(gas < 750000, "Gas must be lower than 750000");
        distributorGas = gas;
    }
    function getCirculatingSupply() public view returns (uint256)
{
        return
_totalSupply.sub(balanceOf(DEAD)).sub(balanceOf(ZERO));
    }

    function getLiquidityBacking(uint256 accuracy)
        public
        view
        returns (uint256)
    {
        return
accuracy.mul(balanceOf(pair).mul(2)).div(getCirculatingSupply());
```

```
    }
    function isOverLiquified(uint256 target, uint256 accuracy)
        public
        view
        returns (bool)
    {
        return getLiquidityBacking(accuracy) > target;
    }
}
```

4. Tax Fee contract

```
function _setFees(
        uint256 _liquidityFee,
        uint256 _buybackFee,
        uint256 _reflectionFee,
        uint256 _marketingFee,
        uint256 _feeDenominator
    ) internal {
        liquidityFee = _liquidityFee;
        buybackFee = _buybackFee;
        reflectionFee = _reflectionFee;
        marketingFee = _marketingFee;
        totalFee =
_liquidityFee.add(_buybackFee).add(_reflectionFee).add(
            _marketingFee
        );
        feeDenominator = _feeDenominator;
        require(
            totalFee < feeDenominator / 4,
            "Total fee should not be greater than 1/4 of fee
denominator"
        );
    }
```

The owner can't set fees over 25%

5. Owner can set operator

```
function authorize(address adr) public onlyOwner {
        authorizations[adr] = true;
    }
```

READ CONTRACT (ONLY NEED TO KNOW)

1. Version
1 uint256
(Shows Contract Versions)

2. _marketingFeeReceiver
0x2b2273d504832595d4300122ef68147d124e9d0e address
(Shows marketing wallet address)

3. buybackFee
300 uint256
(Function for read buyback fee)

4. liquidityFee
500 uint256
(Function for read liquidity fee)

5. marketingFee
200 uint256
(Function for read marketing fee)

5. reflectionFee
200 uint256
(Function for read reward fee)

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

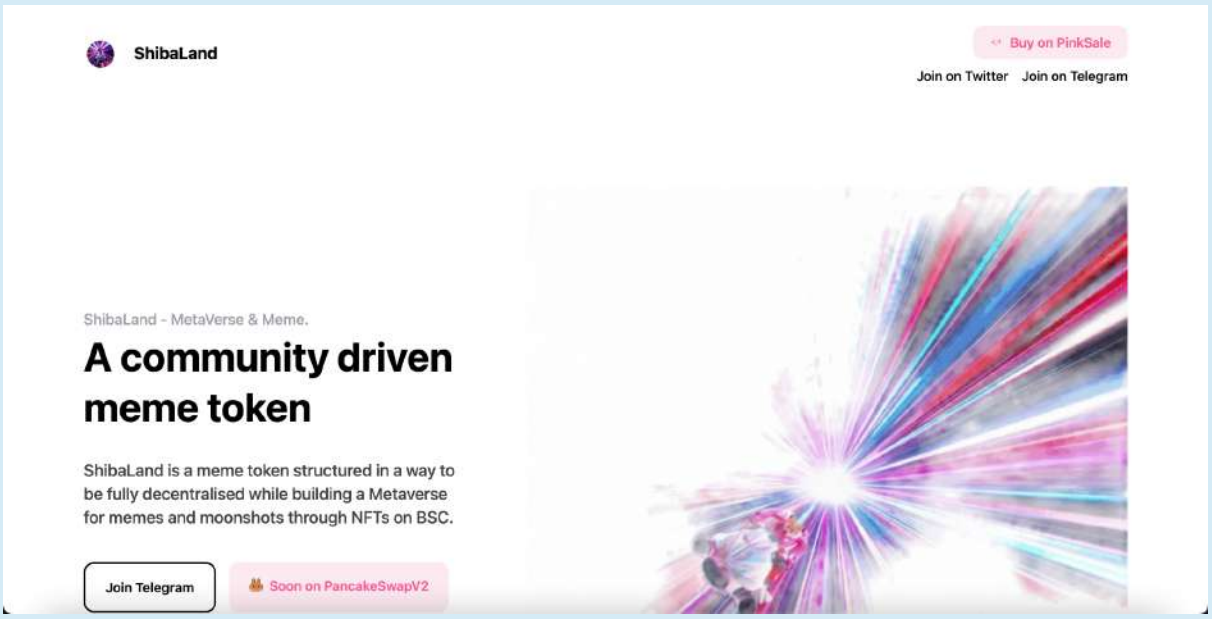
WRITE CONTRACT

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)

4. setFees (can't set over 25%)
_liquidityFee (uint 256)
_buybackFee (uint 256)
_reflectionFee (uint 256)
_marketingFee (uint 256)
_feeDenominator (uint 256)
(The form is filled with new fee, for change all fee and denominator)

WEBSITE REVIEW


ShibaLand

Buy on PinkSale

Join on Twitter Join on Telegram

ShibaLand - MetaVerse & Meme.

A community driven meme token

ShibaLand is a meme token structured in a way to be fully decentralised while building a Metaverse for memes and moonshots through NFTs on BSC.

Join Telegram

Soon on PancakeSwapV2

● **Mobile Friendly**

● **Contains no code error**

● **SSL Secured (By Zero SSL)**

Web-Tech stack: Apache , Ubuntu

Domain .com (godaddy) - Tracked by whois

First Contentful Paint:	1.6s
Fully Loaded Time	5.6s
Performance	66%
Accessibility	88%
Best Practices	92%
SEO	90%

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 No KYC On Blocksafu

HONEYPOT REVIEW

- ● Ability to sell.
- ● The owner is not able to pause the contract.
- ● The owner can't set fees over 25%

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.