BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT

Project: Lucky Shiba

Website: https://shibab.app/

PASSED

BlockSAFU Score:

82

Contract Address:

0x9aEd19e97186f8dE007d0d7C91C576EB3947E9f5

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

BlockSAFU was commissioned by Lucky Shiba to complete a Smart Contract audit. The objective of the Audit is to achieve the following:

- Review the Project and experience and Development team
- Ensure that the Smart Contract functions are necessary and operate as intended.
- Identify any vulnerabilities in the Smart Contract code.

DISCLAIMER: This Audit is intended to inform about token Contract Risks, the result does not imply an endorsement or provide financial advice in any way, all investments are made at your own risk. (https://blocksafu.com/)

SMART CONTRACT REVIEW

Token Name	**Lucky ShibaB**
Token Symbol	**ShiB**
Token Decimal	18
Total Supply	1,000,000,000 **ShiB**
Contract Address	0x9aEd19e97186f8dE007d0d7C91C576EB3947E9f5
Deployer Address	0x3587629d468bbec17f7Eb40b388F97445e467a44
Owner Address	0x3587629d468bbec17f7eb40b388f97445e467a44
Tax Fees Buy	4%
Tax Fees Sell	5%
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-15-2022 05:49:59 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	v0.8.17+commit.8df45f5f
Optimization	No with 200 runs
Sol License	MIT License
Top 5 Holders	*will be updated after the DEX listing*
Other	default evmVersion

TAX

BUY	4%	**Address**	**Sell**	5%
buyTaxBurn	0%	Automatic burn	sellTaxBurn	1%
buyTaxGamePool	2%	0x0EEAaF7726A9C91B5107 C66E95c779402e7dF4B9	sellTaxGamePool	1%
buyTaxMkt	0%	0x6e447ea51254964c4a07E e37e63D0952f8283c53	sellTaxMkt	2%
buyTaxStakingPool	2%	0xff97A65D50723588E0F57 CbAFe9Fd5b386E64426	sellTaxStakingPool	1%

OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 4% (owner can’t set fees over 10%).
- Sell 5% (owner can’t set fees over 10%).

Tx Amount

- Owner cannot set a max tx amount.

Transfer Pausable

- Owner cannot pause.

Blacklist

- Owner cannot set a blacklist.

Ownership

- Owner cannot take back ownership.

Proxy

- This contract has no proxy.

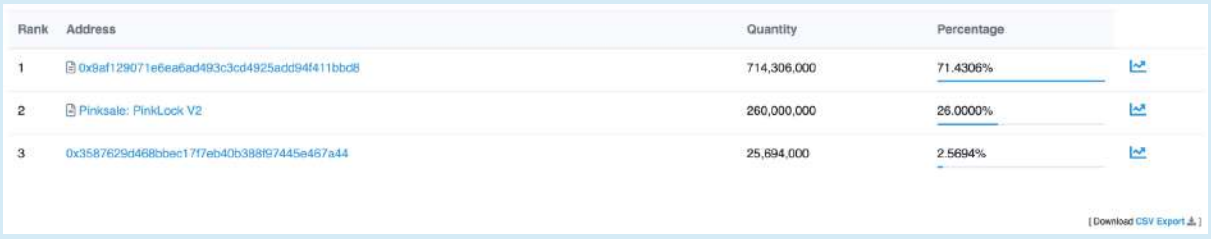
Anti Whale

- Owner cannot limit the number of wallet holdings.

Trading Cooldown

- Owner cannot set the selling time interval.

Token Holder



Rank	Address	Quantity	Percentage
1	0x8af129071e6ea6ad493c3cd4925add94f411bbd8	714,306,000	71.4306%
2	Pinksale: PinkLock V2	260,000,000	26.0000%
3	0x3587629d468bbec17f7eb40b388f97445e467a44	25,694,000	2.5694%

[Download CSV Export]

Team Review

The Lucky Shiba team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 6,462 people in their telegram group (count in audit date).

Official Website And Social Media

Website: https://shibab.app/

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

Twitter: https://twitter.com/LuckyShibaB

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

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.6.0)
(token/ERC20/IERC20.sol)

pragma solidity ^0.8.0;

/**
 * @dev Interface of the ERC20 standard as defined in the EIP.
 */
interface IERC20 {
    /**
     * @dev Emitted when `value` tokens are moved from one account
(`from`) to
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address indexed to,
uint256 value);

    /**
     * @dev Emitted when the allowance of a `spender` for an
`owner` is set by
     * a call to {approve}. `value` is the new allowance.
     */
    event Approval(address indexed owner, address indexed spender,
uint256 value);

    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns
(uint256);
```

```
    /**
     * @dev Moves `amount` tokens from the caller's account to
`to`.
     *
     * Returns a boolean value indicating whether the operation
succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address to, uint256 amount) external returns
(bool);

    /**
     * @dev Returns the remaining number of tokens that `spender`
will be
     * allowed to spend on behalf of `owner` through
{transferFrom}. This is
     * zero by default.
     *
     * This value changes when {approve} or {transferFrom} are
called.
     */
    function allowance(address owner, address spender) external
view returns (uint256);

    /**
     * @dev Sets `amount` as the allowance of `spender` over the
caller's tokens.
     *
     * Returns a boolean value indicating whether the operation
succeeded.
     *
     * IMPORTANT: Beware that changing an allowance with this
method brings the risk
     * that someone may use both the old and the new allowance by
unfortunate
     * transaction ordering. One possible solution to mitigate this
race
     * condition is to first reduce the spender's allowance to 0
and set the
     * desired value afterwards:
     *
```

```
https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount) external
returns (bool);

    /**
     * @dev Moves `amount` tokens from `from` to `to` using the
     * allowance mechanism. `amount` is then deducted from the
caller's
     * allowance.
     *
     * Returns a boolean value indicating whether the operation
succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(
        address from,
        address to,
        uint256 amount
    ) external returns (bool);
}
```
IERC20 Normal Base Template

2. Lucky Shiba Contract

```
contract ShiB is Context,ERC20,Ownable {
    mapping(address => bool) private liquidityPool;
    mapping(address => bool) private isExcludedFromFee;

    uint8 public buyTaxGamePool;
    uint8 public buyTaxStakingPool;
    uint8 public buyTaxMkt;
    uint8 public buyTaxBurn;

    uint8 public sellTaxGamePool;
    uint8 public sellTaxStakingPool;
    uint8 public sellTaxMkt;
    uint8 public sellTaxBurn;



    address private marketingPool;
    address private gamePool;
    address private stakingPool;

    bool inSwapAndLiquify;

    IUniswapV2Router02 public immutable uniswapV2Router;
    address public immutable uniswapV2Pair;

    /**
     * constructor
     */
    constructor() ERC20("Lucky ShibaB", "ShiB") {
        uint256 _total = 10 ** 9 * 10 ** 18;
        _mint(_msgSender(), _total);
        //set tax
        buyTaxGamePool = 2;
        buyTaxStakingPool = 2;
        buyTaxMkt = 0;
        buyTaxBurn = 0;

        sellTaxGamePool = 1;
        sellTaxStakingPool = 1;
        sellTaxMkt = 2;
        sellTaxBurn = 1;
        marketingPool =
0x6e447ea51254964c4a07Ee37e63D0952f8283c53;
        gamePool = 0x0EEAaF7726A9C91B5107C66E95c779402e7dF4B9;
        stakingPool = 0xff97A65D50723588E0F57CbAFe9Fd5b386E64426;

        //mainnet
        address routerV2 =
0x10ED43C718714eb63d5aA57B78B54704E256024E;
        if(block.chainid == 97)
        {
            //testnet
            routerV2 = 0xD99D1c33F9fC3444f8101754aBC46c52416550D1;
        }

        IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(routerV2);
        uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory())
            .createPair(address(this), _uniswapV2Router.WETH());

        uniswapV2Router = _uniswapV2Router;
    }

    modifier lockTheSwap {
        inSwapAndLiquify = true;
        _;
        inSwapAndLiquify = false;
    }

    function excludeFromFee(address _address) public onlyOwner {
        isExcludedFromFee[_address] = true;
    }

    function includeInFee(address _address) public onlyOwner {
        isExcludedFromFee[_address] = false;
    }


    function setLiquidityPoolStatus(address _lpAddress, bool
_status) external onlyOwner {
        liquidityPool[_lpAddress] = _status;
    }

    function setMarketingPool(address _marketingPool) external
onlyOwner {
        marketingPool = _marketingPool;
    }

    function setGamePool(address _gamePool) external onlyOwner {
        gamePool = _gamePool;
    }

    function setStakingPool(address _stakingPool) external
onlyOwner {
        stakingPool = _stakingPool;
    }

    function setSellTaxes(uint8 _gamePool, uint8 _stakingPool,
uint8 _mktFee, uint8 _burn) external onlyOwner {
        require(_gamePool + _stakingPool + _mktFee + _burn <= 10,
"Sell Tax cannot be greater than 10");
        sellTaxGamePool = _gamePool;
        sellTaxStakingPool = _stakingPool;
        sellTaxMkt = _mktFee;
        sellTaxBurn = _burn;
    }

    function setBuyTaxes(uint8 _gamePool, uint8 _stakingPool,
uint8 _mktFee, uint8 _burn) external onlyOwner {
        require(_gamePool + _stakingPool + _mktFee + _burn <= 10,
"Buy Tax cannot be greater than 10");
        buyTaxGamePool = _gamePool;
        buyTaxStakingPool = _stakingPool;
        buyTaxMkt = _mktFee;
        buyTaxBurn = _burn;
    }

  function _transfer(address sender, address receiver, uint256
amount) internal virtual override {
    require(sender != address(0), "ERC20: transfer from the zero
address");
    require(receiver != address(0), "ERC20: transfer to the zero
address");
```

```
    require(amount > 0, "Transfer amount must be greater than 
zero");

    uint256 taxAmount = 0;

    if(liquidityPool[sender] == true) {
      //buyTax = 
buyTaxGamePool+buyTaxStakingPool+buyTaxMkt+buyTaxBurn;
      taxAmount = (amount * uint256(buyTaxGamePool + 
buyTaxStakingPool + buyTaxMkt + buyTaxBurn)) / 100;
    } else if(liquidityPool[receiver] == true) {
      //sellTax = 
sellTaxGamePool+sellTaxStakingPool+sellTaxMkt+sellTaxBurn;
      taxAmount = (amount * uint256(sellTaxGamePool + 
sellTaxStakingPool + sellTaxMkt + sellTaxBurn)) / 100;
    }

    if(isExcludedFromFee[sender] || isExcludedFromFee[receiver] || 
sender == address(this) || receiver == address(this) || 
inSwapAndLiquify) {
      taxAmount = 0;
    }

    if(taxAmount > 0 && !inSwapAndLiquify) {

        if(liquidityPool[receiver])
        {
            //game
            if(sellTaxGamePool > 0)
            {
                super._transfer(sender, gamePool, amount * 
uint256(sellTaxGamePool) / 100);
            }

            //staking
            if(sellTaxStakingPool > 0)
            {
                super._transfer(sender, stakingPool, amount * 
uint256(sellTaxStakingPool) / 100);
            }

            //mkt
            if(sellTaxMkt > 0)
            {
                uint256 tokenToSwap = amount * uint256(sellTaxMkt)
/ 100;
                if(receiver == uniswapV2Pair)
                {
                    swapTokensForEth(sender, tokenToSwap);
                }
                else
                {
                    super._transfer(sender, marketingPool,
tokenToSwap);
                }

            }

            //burn
            if(sellTaxBurn > 0)
            {
                _burn(sender, amount * uint256(sellTaxBurn) /
100);
            }
        }
        else if(liquidityPool[sender])
        {
            //game
            if(buyTaxGamePool > 0)
            {
                super._transfer(sender, gamePool, amount *
uint256(buyTaxGamePool) / 100);
            }

            //staking
            if(buyTaxStakingPool > 0)
            {
                super._transfer(sender, stakingPool, amount *
uint256(buyTaxStakingPool) / 100);
            }

            //mkt
            if(buyTaxMkt > 0)
            {
                super._transfer(sender, marketingPool, amount *
uint256(buyTaxMkt) / 100);
            }

            //burn
            if(buyTaxBurn > 0)
            {
                _burn(sender, amount * uint256(buyTaxBurn) / 100);
            }
        }

    }
    super._transfer(sender, receiver, amount - taxAmount);
  }

  function swapTokensForEth(address sender, uint256 tokenAmount)
private lockTheSwap  {
        // transfer to contract and swap to bnb
        super._transfer(sender, address(this), tokenAmount);

        // 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),
balanceOf(address(this)));

        // make the swap

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
(
            balanceOf(address(this)),
            0, // accept any amount of ETH
            path,
            marketingPool,
            block.timestamp
        );
    }
}
```

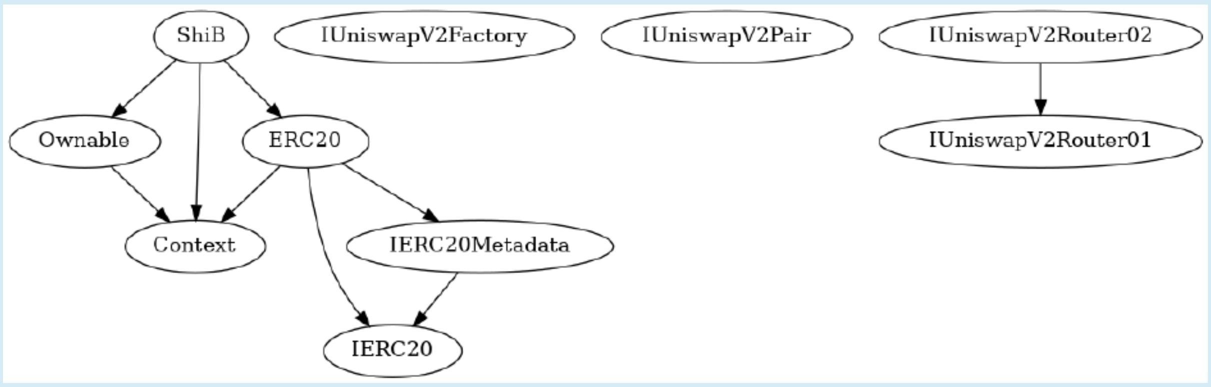
3. Contract Tax

```
     function setSellTaxes(uint8 _gamePool, uint8 _stakingPool,
uint8 _mktFee, uint8 _burn) external onlyOwner {
        require(_gamePool + _stakingPool + _mktFee + _burn <= 10,
"Sell Tax cannot be greater than 10");
        sellTaxGamePool = _gamePool;
        sellTaxStakingPool = _stakingPool;
        sellTaxMkt = _mktFee;
        sellTaxBurn = _burn;
    }

    function setBuyTaxes(uint8 _gamePool, uint8 _stakingPool,
uint8 _mktFee, uint8 _burn) external onlyOwner {
        require(_gamePool + _stakingPool + _mktFee + _burn <= 10,
"Buy Tax cannot be greater than 10");
        buyTaxGamePool = _gamePool;
        buyTaxStakingPool = _stakingPool;
        buyTaxMkt = _mktFee;
        buyTaxBurn = _burn;
    }
```

Set tax Contract, cannot set fee over 10%

Contract Inheritance



READ CONTRACT (ONLY NEED TO KNOW)

1. buyTaxBurn
0 uint256
(Shows buyTaxBurn Fee)

2. buyTaxGamePool
2 uint256
(Shows buyTaxGamePool Fee)

3. buyTaxMkt
0 uint256
(Shows buyTaxMkt Fee)

4. buyTaxStakingPool
2 uint256
(Shows buyTaxStakingPool Fee)

5. name
Lucky ShibB string
(Function for read Token name))

WRITE CONTRACT

1. setBuyTaxes
_gamePool (uint8)
_stakingPool (uint8)
_mktFee (uint8)
_burn (uint8)
(Function for set buy tax fee)

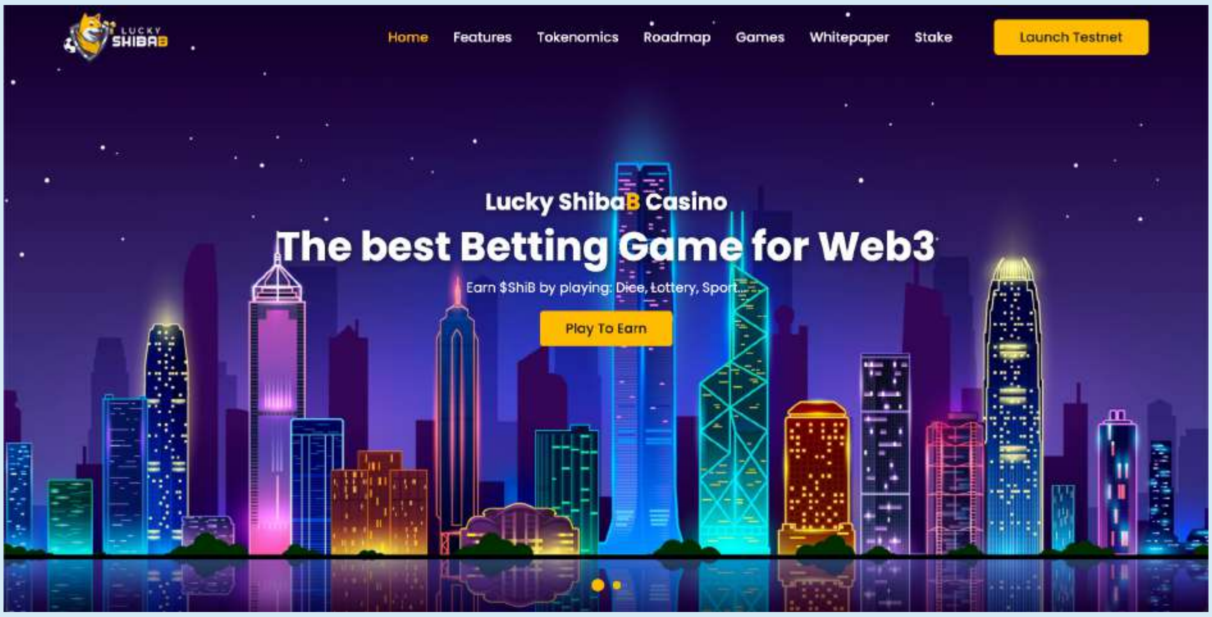
2. setBuyTaxes
_gamePool (uint8)
_stakingPool (uint8)
_mktFee (uint8)
_burn (uint8)
(Function for set sell tax fee)

3. setStakingPool
_stakingPool (address)
(Function for set staking pool address)

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

5. 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: Bootstrap, cloudflare

Domain .app (Cloudflare) - Tracked by whois

First Contentful Paint:	1.9s
Fully Loaded Time	3.4s
Performance	95%
Accessibility	90%
Best Practices	100%
SEO	70%

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 on PinkSale

HONEYPOT REVIEW

- 🟢 Ability to sell.
- 🟢 The owner is not able to pause the contract.
- 🟢 The owner can’t set fees over 10%

Note: Please check the disclaimer above. Note, 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.