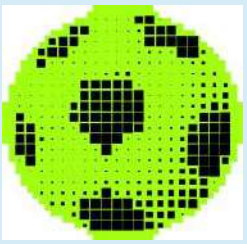
BlockSAFU

ADVANCE MANUAL
SMART CONTRACT AUDIT



Project: QatarBet

Website: https://qatarbet.io/

BlockSAFU Score:

22

Contract Address:

0x7d15884a6F40653f3c853CB1CB52E4fF879085b5

Disclamer: BlockSAFU is not responsible for any financial losses.
Nothing in this contract audit is financial advice, please do your own reasearch.

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 QatarBet 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	**Qatar Bet**
Token Symbol	**QB**
Token Decimal	18
Total Supply	100,000,000 **QB**
Contract Address	0x7d15884a6F40653f3c853CB1CB52E4fF879085b5
Deployer Address	0x37C5387098C9432FF3763D93884c27FC8f059268
Owner Address	0x37c5387098c9432ff3763d93884c27fc8f059268
Tax Fees Buy	5%
Tax Fees Sell	6%
Gas Used for Buy	*will be updated after the DEX listing*
Gas Used for Sell	*will be updated after the DEX listing*
Contract Created	Aug-24-2022 01:49:38 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	v0.7.6+commit.7338295f
Optimization	Yes with 200 runs
Sol License	Unlicense
Top 5 Holders	*will be updated after the DEX listing*
Other	default evmVersion

TAX

BUY	5%	**SELL**	6%
burn Fee	1%	Sell Fee	1%
ecosystem Fee	2%	Burn Fee	1%
liquidity Fee	2%	Ecosystem fee	2%
		Liquidity Fee	3%

OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 5% (owner can’t set fees over 25%).
- Sell 6% (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.

Token Holder

Rank	Address	Quantity	Percentage	Analytics
1	0x37c5387098c9432ff3763d93884c27fc8f059268	100,000,000	100.0000%	

[Download CSV Export]

Team Review

The QatarBet team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 4 people in their telegram group (count in audit date).

Official Website And Social Media

Website: https://qatarbet.io/

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

Twitter: https://twitter.com/Qatarbet68

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

1 medium-risk code issue found

Should be fixed, could bring problems.

1. On testnet test. the contract cannot delete liquidity before its is whitelisted, even though the contract does not have an include fee function so that transactions in the pair can no longer apply fees.

   This is dangerous if there are investors who take the initiative to add liquidity.

High-Risk

0 high-risk code issues found

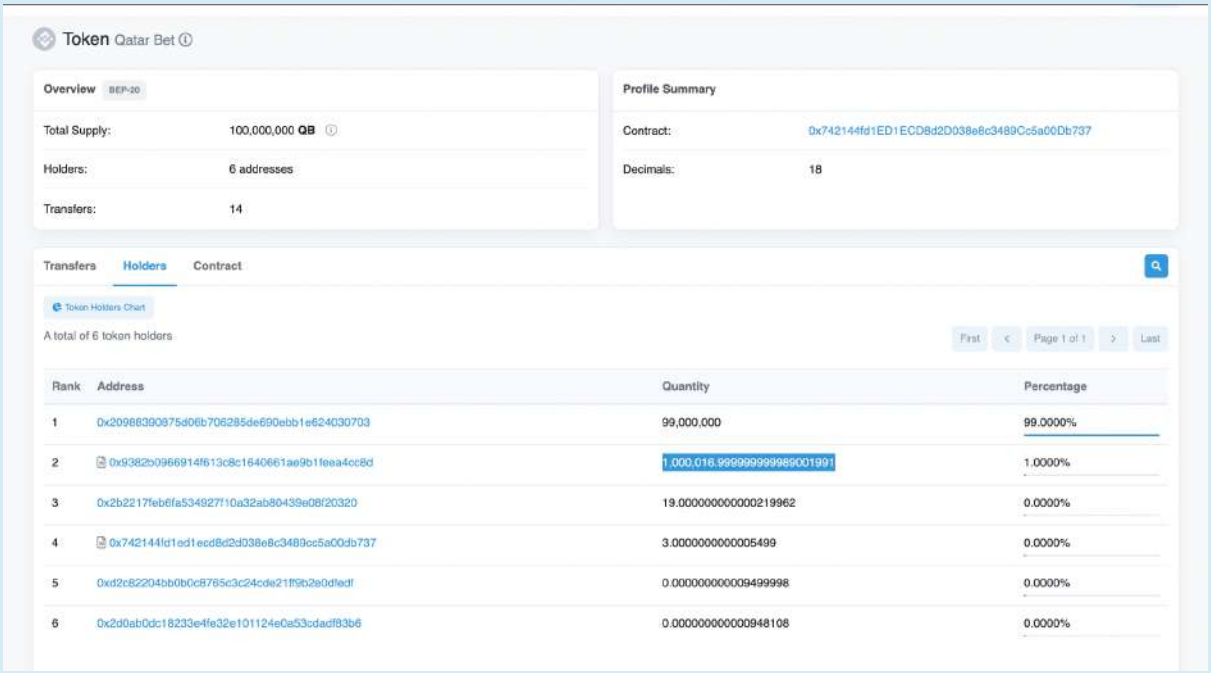
Must be fixed, and will bring problem.

Critical-Risk

1 critical-risk code issues found

Must be fixed, and will bring problem.

1. Hidden Mint



Token Qatar Bet

Overview BEP-20

Total Supply:	100,000,000 QB
Holders:	6 addresses
Transfers:	14

Profile Summary

Contract:	0x742144fd1ED1ECD8d2D038e8c3489Cc5a00Db737
Decimals:	18

Transfers | Holders | Contract

Token Holders Chart

A total of 6 token holders

First < Page 1 of 1 > Last

Rank	Address	Quantity	Percentage
1	0x20986390875d06b706285de690ebb1e624030703	99,000,000	99.0000%
2	0x9382b0966914f613c8c1640661ae9b1feea4cc8d	1,000,016.999999999989001991	1.0000%
3	0x2b2217feb6fa534927f10a32ab80439e08f20320	19.000000000000219962	0.0000%
4	0x742144fd1ed1ecd8d2d038e8c3489cc5a00db737	3.0000000000005499	0.0000%
5	0xd2c82204bb0b0c8765c3c24cde21ff9b2e0dfedf	0.000000000009499998	0.0000%
6	0x2d0ab0dc18233e4fe32e101124e0a53cdadf83b6	0.000000000000948108	0.0000%

The contract have hidden mint

Total Supply Default: 100,000,000 **QB**

After Transaction Total Supply Increase to: 100,000,039 **QB**

(This contract indicated have hidden mint function)

EXTRA NOTES SMART CONTRACT

1. IERC20

```
interface IERC20 {
    /**
     * @dev Returns the number of tokens in existence.
     */
    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);

    /**
     * @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);
...
}
```

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. Qatar Bet Contract

```
contract QatarBet is ERC20Detailed, Ownable {
    using SafeMath for uint256;
    using SafeMathInt for int256;

    string public constant _name = "Qatar Bet";
    string public constant _symbol = "QB";
    uint8 public constant _decimals = 18;

    IPancakeSwapPair public pairContract;
    mapping(address => bool) _isFeeExempt;

    modifier validRecipient(address to) {
        require(to != address(0x0));
        _;
    }

    uint256 public constant MAX_UINT256 = ~uint256(0);
    uint8 public constant RATE_DECIMALS = 7;

    uint256 private constant MAX_AMMOUT_TO_WITH_DRAW = 1 * 10**6 *
10**_decimals;

    uint256 private constant INITIAL_FRAGMENTS_SUPPLY =
        100 * 10**6 * 10**_decimals;

    //Buy 5%, Sell 6%
    uint256 public ecosystemFee = 2;
    uint256 public liquidityFee = 2;
    uint256 public burnFee = 1;
    uint256 public sellFee = 1;

    uint256 public totalFee =
ecosystemFee.add(liquidityFee).add(burnFee);

    uint256 public constant feeDenominator = 100;

    address constant DEAD =
0x000000000000000000000000000000000000dEaD;
    address constant ZERO =
0x0000000000000000000000000000000000000000;
```

```
    address public ecosystemReceiver;
    address public pairAddress;

    IPancakeSwapRouter public router;

    bool public antiBotEnable = false;
    uint256 public constant antiTime = 5 minutes;
    uint256 public lastAntiTime = 0;

    address public pair;
    bool inSwap = false;

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

    uint256 private constant TOTAL_GONS =
        MAX_UINT256 - (MAX_UINT256 % INITIAL_FRAGMENTS_SUPPLY);

    bool public _autoAddLiquidity;
    uint256 public _lastAddLiquidityTime;
    uint256 public _totalSupply;
    uint256 private _gonsPerFragment;

    mapping(address => uint256) private _gonBalances;
    mapping(address => mapping(address => uint256)) private
_allowedFragments;

    constructor() ERC20Detailed(_name, _symbol, uint8(_decimals))
Ownable() {
        router =
IPancakeSwapRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);

        pair = IPancakeSwapFactory(router.factory()).createPair(
            router.WETH(),
            address(this)
        );

        ecosystemReceiver =
0x97A95C2837e611d46fD4f8c198cFB91D8F2eC179;
```

```
        _allowedFragments[address(this)][address(router)] =
uint256(-1);

        pairAddress = pair;
        pairContract = IPancakeSwapPair(pair);

        _totalSupply = INITIAL_FRAGMENTS_SUPPLY;
        _gonBalances[msg.sender] = TOTAL_GONS;
        _gonsPerFragment = TOTAL_GONS.div(_totalSupply);

        _autoAddLiquidity = true;
        _isFeeExempt[ecosystemReceiver] = true;
        _isFeeExempt[msg.sender] = true;
        _isFeeExempt[address(this)] = true;

        emit Transfer(address(0x0), msg.sender, _totalSupply);
    }

    function transfer(address to, uint256 value)
        external
        override
        validRecipient(to)
        returns (bool)
    {
        _transferFrom(msg.sender, to, value);
        return true;
    }

    function transferFrom(
        address from,
        address to,
        uint256 value
    ) external override validRecipient(to) returns (bool) {
        if (_allowedFragments[from][msg.sender] != uint256(-1)) {
            _allowedFragments[from][msg.sender] =
_allowedFragments[from][
                msg.sender
            ].sub(value, "Insufficient Allowance");
        }
        _transferFrom(from, to, value);
        return true;
```

```
    }

    function _basicTransfer(
        address from,
        address to,
        uint256 amount
    ) internal returns (bool) {
        uint256 gonAmount = amount.mul(_gonsPerFragment);
        _gonBalances[from] = _gonBalances[from].sub(gonAmount);
        _gonBalances[to] = _gonBalances[to].add(gonAmount);
        return true;
    }

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

        if (shouldAddLiquidity()) {
            addLiquidity();
        }

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

        uint256 gonAmount = amount.mul(_gonsPerFragment);
        _gonBalances[sender] =
_gonBalances[sender].sub(gonAmount);
        uint256 gonAmountReceived = shouldTakeFee(sender,
recipient)
            ? takeFee(sender, recipient, gonAmount)
            : gonAmount;
        _gonBalances[recipient] = _gonBalances[recipient].add(
            gonAmountReceived
        );

        emit Transfer(
```

```
            sender,
            recipient,
            gonAmountReceived.div(_gonsPerFragment)
        );
        return true;
    }

    function takeFee(
        address sender,
        address recipient,
        uint256 gonAmount
    ) internal returns (uint256) {
        uint256 _totalFee = totalFee;
        uint256 activeTime = lastAntiTime + antiTime;

        if (recipient == pair) {
            _totalFee = totalFee.add(sellFee);
        }

        if(antiBotEnable &&  block.timestamp < activeTime){
            _totalFee = 25;
        }

        uint256 feeAmount =
gonAmount.mul(_totalFee).div(feeDenominator);

        _gonBalances[address(this)] =
_gonBalances[address(this)].add(feeAmount);

        _gonBalances[DEAD] =
_gonBalances[DEAD].add(gonAmount.mul(burnFee).div(feeDenominator))
;

        _gonBalances[ecosystemReceiver] =
_gonBalances[ecosystemReceiver].add(gonAmount.mul(liquidityFee).di
v(feeDenominator));

        emit Transfer(sender, address(this),
feeAmount.div(_gonsPerFragment));
        return gonAmount.sub(feeAmount);
    }
```

```
    function swapBack() internal swapping {
        uint256 amountToSwap = _gonBalances[address(this)].div(
            _gonsPerFragment
        );

        if (amountToSwap == 0) {
            return;
        }

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

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

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

        uint256 _currentFee = ecosystemFee.add(burnFee);

        (bool success, ) = payable(ecosystemReceiver).call{
            value:
amountETHToTreasuryAndReward.mul(ecosystemFee).div(_currentFee),
            gas: 30000
        }("");

        require(success, "TransferHelper: BNB_TRANSFER_FAILED");

    }

    function addLiquidity() internal swapping {
        uint256 autoLiquidityAmount =
_gonBalances[ecosystemReceiver].div(
```

```
            _gonsPerFragment
        );
        _gonBalances[address(this)] =
_gonBalances[address(this)].add(
            _gonBalances[ecosystemReceiver]
        );
        _gonBalances[ecosystemReceiver] = 0;
        uint256 amountToLiquify = autoLiquidityAmount.div(2);
        uint256 amountToSwap =
autoLiquidityAmount.sub(amountToLiquify);

        if (amountToSwap == 0) {
            return;
        }
        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 amountETHLiquidity =
address(this).balance.sub(balanceBefore);

        if (amountToLiquify > 0 && amountETHLiquidity > 0) {
            router.addLiquidityETH{value: amountETHLiquidity}(
                address(this),
                amountToLiquify,
                0,
                0,
                ecosystemReceiver,
                block.timestamp
            );
        }
        _lastAddLiquidityTime = block.timestamp;
```

```
    }

    function withdrawAllToTreasury() external swapping onlyOwner {
        uint256 amountToSwap = _gonBalances[address(this)].div(
            _gonsPerFragment
        );
        require(
            amountToSwap > 0,
            "There is no token deposited in token contract"
        );
        //Max amount to swap 1% of total supply
        uint256 realAmountToSwap = amountToSwap >
MAX_AMMOUT_TO_WITH_DRAW ? MAX_AMMOUT_TO_WITH_DRAW : amountToSwap;

        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = router.WETH();
        router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            realAmountToSwap,
            0,
            path,
            ecosystemReceiver,
            block.timestamp
        );
    }

    function shouldTakeFee(address from, address to)
        internal
        view
        returns (bool)
    {
        return (pair == from || pair == to) &&
!_isFeeExempt[from];
    }

    function shouldSwapBack() internal view returns (bool) {
        return !inSwap && msg.sender != pair;
    }

    function shouldAddLiquidity() internal view returns (bool) {
        return
            _autoAddLiquidity &&
```

```
            !inSwap &&
            msg.sender != pair &&
            block.timestamp >= (_lastAddLiquidityTime + 720
minutes);
    }

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

    function decreaseAllowance(address spender, uint256
subtractedValue)
        external
        returns (bool)
    {
        uint256 oldValue = _allowedFragments[msg.sender][spender];
        if (subtractedValue >= oldValue) {
            _allowedFragments[msg.sender][spender] = 0;
        } else {
            _allowedFragments[msg.sender][spender] = oldValue.sub(
                subtractedValue
            );
        }
        emit Approval(
            msg.sender,
            spender,
            _allowedFragments[msg.sender][spender]
        );
        return true;
    }

    function increaseAllowance(address spender, uint256
addedValue)
        external
        returns (bool)
    {
        _allowedFragments[msg.sender][spender] =
```

```
_allowedFragments[msg.sender][
            spender
        ].add(addedValue);
        emit Approval(
            msg.sender,
            spender,
            _allowedFragments[msg.sender][spender]
        );
        return true;
    }

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

    function checkFeeExempt(address _addr) external view returns
(bool) {
        return _isFeeExempt[_addr];
    }

    function getCirculatingSupply() public view returns (uint256)
{
        return

(TOTAL_GONS.sub(_gonBalances[DEAD]).sub(_gonBalances[ZERO])).div(
                _gonsPerFragment
            );
    }

    function isNotInSwap() external view returns (bool) {
        return !inSwap;
    }

    function manualSync() external {
        IPancakeSwapPair(pair).sync();
    }
```

```
    function setFeeReceivers(
        address _ecosystemReceiver
    ) external onlyOwner {
        ecosystemReceiver = _ecosystemReceiver;
    }

    function setTaxFee(uint256 _ecosystemFee, uint256 _burnFee,
uint256 _liquidityFee, uint256 _sellFee) external onlyOwner {

require(_ecosystemFee.add(_burnFee).add(_liquidityFee).add(_sellFe
e) <= 25, "Fees must be less than 25%");
        ecosystemFee = _ecosystemFee;
        burnFee = _burnFee;
        liquidityFee = _liquidityFee;
        sellFee = _sellFee;
        totalFee = ecosystemFee.add(liquidityFee).add(burnFee);
    }

    function setEnableAntiBot() external onlyOwner {
        require(lastAntiTime == 0, "Cannot enable the flag antibot
anymore");
        antiBotEnable = true;
        lastAntiTime = block.timestamp;
    }

    function setAutoAddLiquidity(bool _flag) external onlyOwner {
        if (_flag) {
            _autoAddLiquidity = _flag;
            _lastAddLiquidityTime = block.timestamp;
        } else {
            _autoAddLiquidity = _flag;
        }
    }

    function getLiquidityBacking(uint256 accuracy)
        external
        view
        returns (uint256)
    {
        uint256 liquidityBalance =
_gonBalances[pair].div(_gonsPerFragment);
```

```
        return

accuracy.mul(liquidityBalance.mul(2)).div(getCirculatingSupply());
    }

    function setWhitelist(address _addr) external onlyOwner {
        _isFeeExempt[_addr] = true;
    }

    function setPairAddress(address _pairAddress) external
onlyOwner {
        pairAddress = _pairAddress;
    }

    function setLP(address _address) external onlyOwner {
        pairContract = IPancakeSwapPair(_address);
    }

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

    function balanceOf(address who) external view override returns
(uint256) {
        return _gonBalances[who].div(_gonsPerFragment);
    }

    receive() external payable {}
}
```

4. Tax Fee contract

```
function setTaxFee(uint256 _ecosystemFee, uint256 _burnFee,
uint256 _liquidityFee, uint256 _sellFee) external onlyOwner {

require(_ecosystemFee.add(_burnFee).add(_liquidityFee).add(_sellFe
e) <= 25, "Fees must be less than 25%");
        ecosystemFee = _ecosystemFee;
        burnFee = _burnFee;
        liquidityFee = _liquidityFee;
        sellFee = _sellFee;
        totalFee = ecosystemFee.add(liquidityFee).add(burnFee);
    }
```

The owner can’t set fees over 25%

READ CONTRACT (ONLY NEED TO KNOW)

1. MAX_UINT256
115792089237316195423570985008687907853269984665640564039457584007913129639935 uint256
(Shows Contract Max UINT)

2. RATE_DECIMALS
7 uint8
(Shows the rate decimals)

3. _autoAddLiquidity
True bool
(Function For set auto add liquidity)

4. decimals
18 uint8
(Function for read decimals)

5. _lastAddLiquidityTime
0 uint256
(Function for read last add liquidity time)

6. _name
Qatar Bet string
(Function for read marketing fee)

7. _symbol
QB string
(Function for read Token symbol)

WRITE CONTRACT

1. setAutoAddLiquidity
_flag (bool)
(The form is filled with the true or false for active auto add liquidity)

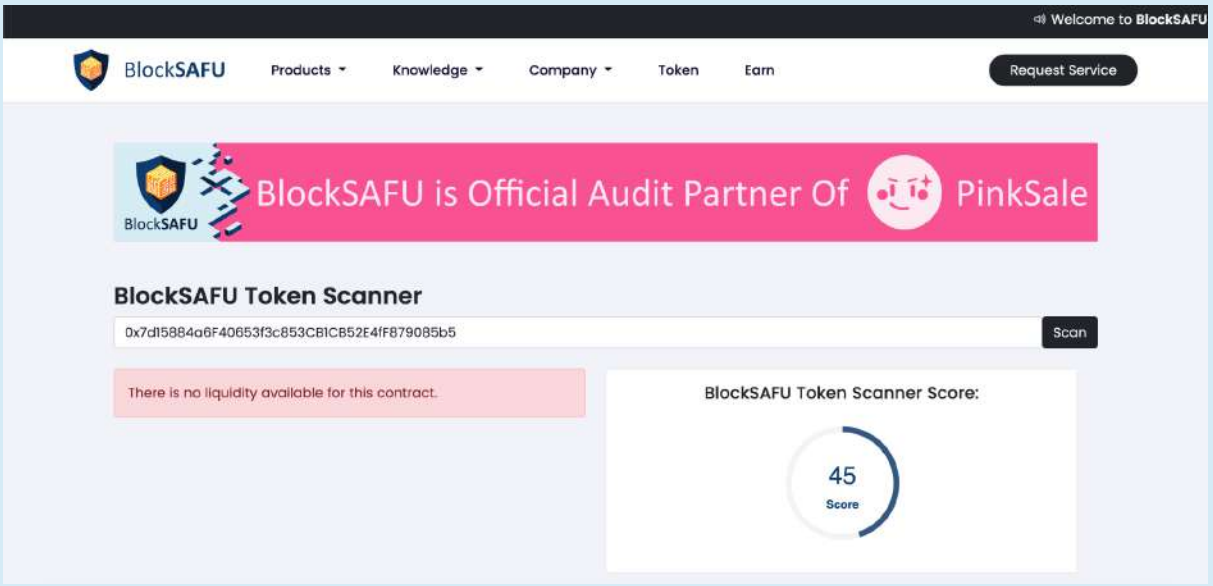
2. setEnableAntiBot
(The call function for enable anti bot)

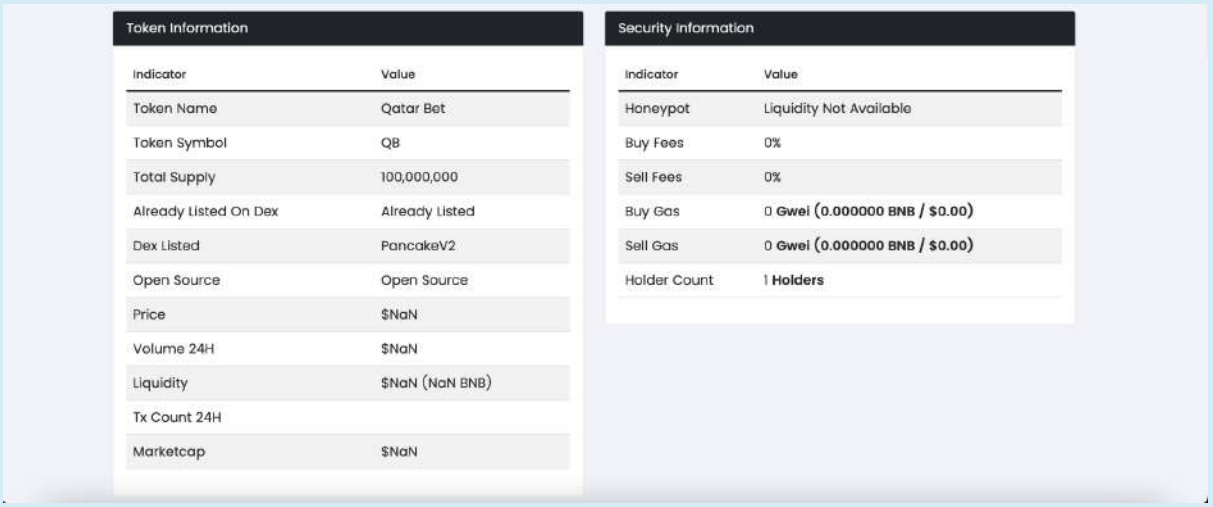
3. transferOwnership
newOwner (address)
(Its function is to change the owner)

4. setTaxFee (cannot set over 25%)
_ecosystemFee uint256
_burnFee uint256
_liquidityFee uint256
_sellFee uint256
(The form is filled with new fee, for change all tax fee)

BlockSAFU TOKEN SCANNER

https://blocksafu.com/token-scanner



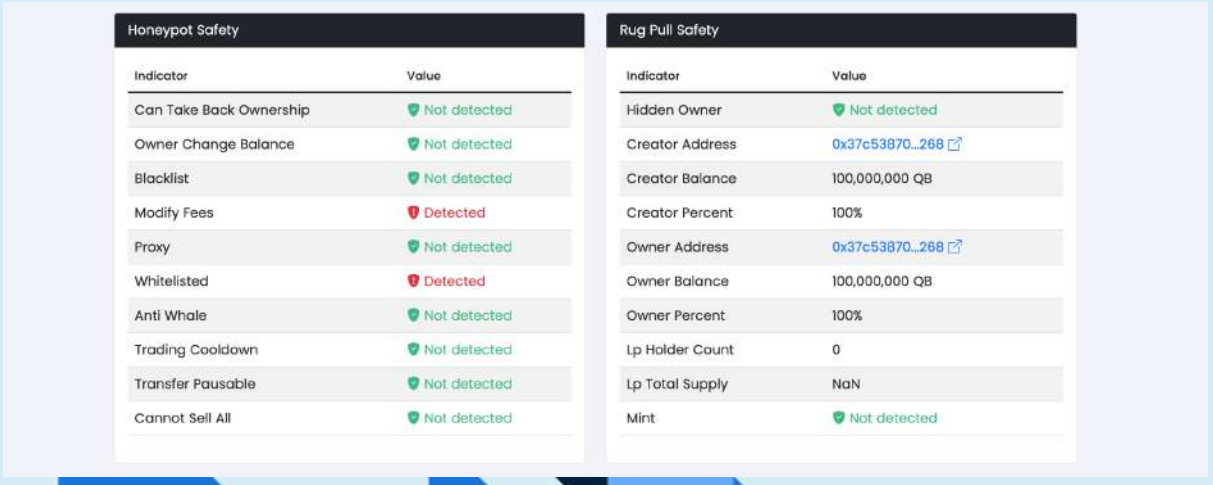


Token Information

Indicator	Value
Token Name	Qatar Bet
Token Symbol	QB
Total Supply	100,000,000
Already Listed On Dex	Already Listed
Dex Listed	PancakeV2
Open Source	Open Source
Price	$NaN
Volume 24H	$NaN
Liquidity	$NaN (NaN BNB)
Tx Count 24H	
Marketcap	$NaN

Security Information

Indicator	Value
Honeypot	Liquidity Not Available
Buy Fees	0%
Sell Fees	0%
Buy Gas	0 **Gwei (0.000000 BNB / $0.00)**
Sell Gas	0 **Gwei (0.000000 BNB / $0.00)**
Holder Count	1 **Holders**



Honeypot Safety

Indicator	Value
Can Take Back Ownership	Not detected
Owner Change Balance	Not detected
Blacklist	Not detected
Modify Fees	Detected
Proxy	Not detected
Whitelisted	Detected
Anti Whale	Not detected
Trading Cooldown	Not detected
Transfer Pausable	Not detected
Cannot Sell All	Not detected

Rug Pull Safety

Indicator	Value
Hidden Owner	Not detected
Creator Address	0x37c53870...268
Creator Balance	100,000,000 QB
Creator Percent	100%
Owner Address	0x37c53870...268
Owner Balance	100,000,000 QB
Owner Percent	100%
Lp Holder Count	0
Lp Total Supply	NaN
Mint	Not detected

WEBSITE REVIEW

- **Mobile Friendly**
- **Contains no code error**
- **SSL Secured (By E1 SSL)**

Web-Tech stack: React, Next Js, Node js

Domain .io - Tracked by whois

First Contentful Paint:	771ms
Fully Loaded Time	1.7s
Performance	94%
Accessibility	73%
Best Practices	83%
SEO	82%

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 Not Yet 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.