



BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT



Project: Larva DAO

Website: <https://larvadao.io/>



BlockSAFU Score:

82

Contract Address:

0x1a12f78E01756228A25d7866E96f2637E5471fCE

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

DISCLAIMER

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. BlockSAFU's 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 Larva DAO 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	LarvaDao
Token Symbol	LARD
Token Decimal	9
Total Supply	500,000,000 LARD
Contract Address	0x1a12f78E01756228A25d7866E96f2637E5471fCE
Deployer Address	0x9Aec56a4B3b894aAdCb746637DB94e29CC832631
Owner Address	0x9Aec56a4B3b894aAdCb746637DB94e29CC832631
Tax Fees Buy	Redis fee 2%, Tax fee 2%
Tax Fees Sell	Redis fee 2%, Tax fee 2%
Gas Used for Buy	<i>will be updated after the DEX listing</i>
Gas Used for Sell	<i>will be updated after the DEX listing</i>
Contract Created	Jul-31-2022 05:31:59 PM +UTC
Initial Liquidity	<i>will be updated after the DEX listing</i>
Liquidity Status	Locked
Unlocked Date	<i>will be updated after the DEX listing</i>
Verified CA	Yes
Compiler	v0.8.7+commit.e28d00a7
Optimization	No with 200 runs
Sol License	None License
Top 5 Holders	<i>will be updated after the DEX listing</i>
Other	default evmVersion

TAX

BUY	4%	SELL	4%
Redis Fee	2%	Redis Fee	2%
Tax Fee	2%	Tax Fee	2%

OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 4% (owner cannot set fees over 10%).
- Sell 4% (owner can't set fees over 10%).

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.

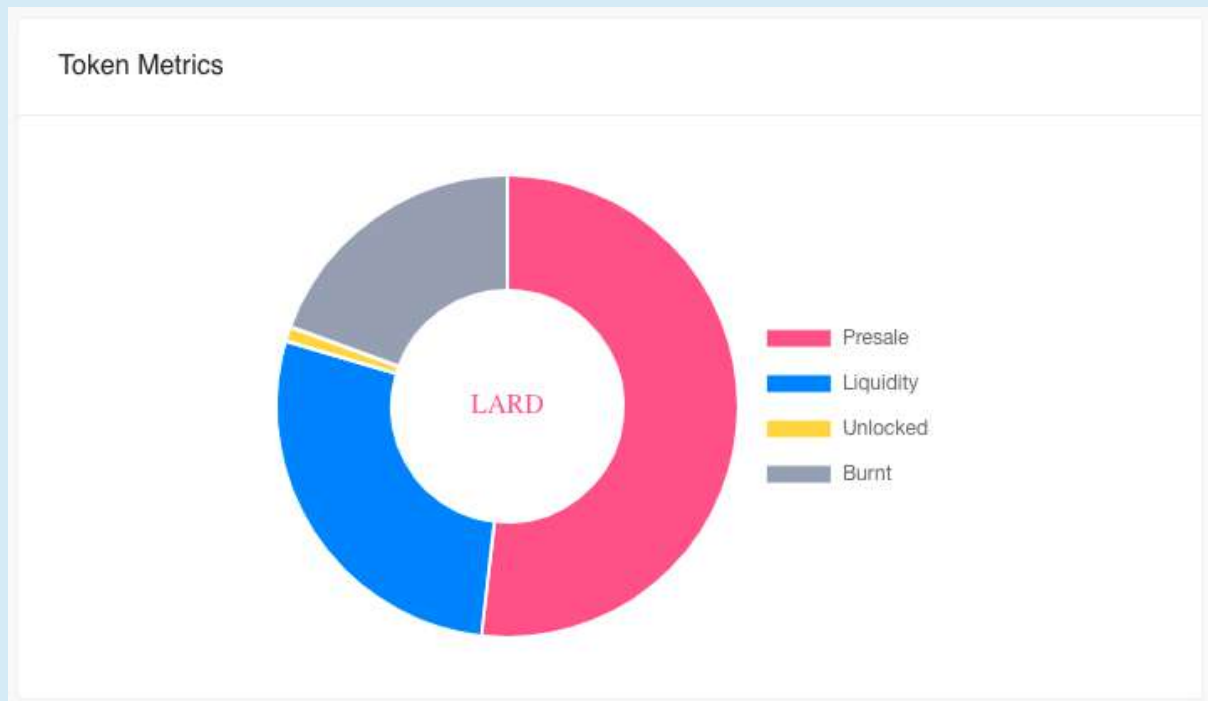
Anti Whale

- Owner cannot limit the number of wallet holdings.

Trading Cooldown

- Owner cannot set the selling time interval.

Token Metrics



Team Review

The Larva DAO team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 6,370 people in their telegram group (count in audit date).

Official Website And Social Media

Website: <https://larvadao.io/>

Telegram Group: <https://t.me/larvadaoglobal>

Twitter: <https://twitter.com/larvadao>

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 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. Larva DAO Contract

```
contract LARVADAO is Context, IERC20, Ownable {

    using SafeMath for uint256;
    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private
_allowances;
    mapping (address => bool) private _isExcludedFromFee;

    uint256 private constant MAX = ~uint256(0);
    uint256 private _tTotal = 500 * 10**6 * 10**9;
    uint256 private _rTotal = (MAX - (MAX % _tTotal));
    uint256 private _tFeeTotal;

    uint256 private _redisFeeOnBuy = 2;
    uint256 private _redisFeeOnSell = 2;

    uint256 private _taxFeeOnBuy = 2;
    uint256 private _taxFeeOnSell = 2;

    uint256 private _redisFee;
    uint256 private _taxFee;

    string private constant _name = "LarvaDao";
    string private constant _symbol = "LARD";
    uint8 private constant _decimals = 9;

    address payable private _marketingAddress =
payable(0xbe53845c5b4bb6A6BEC976bbf1508E515636412b);

    IUniswapV2Router02 public uniswapV2Router;
    address public uniswapV2Pair;

    bool private inSwap = false;
    bool private swapEnabled = true;

    modifier lockTheSwap {
        inSwap = true;
        _;
        inSwap = false;
    }
}
```

```

    constructor () {
        _rOwned[_msgSender()] = _rTotal;

        IUniswapV2Router02 _uniswapV2Router =
        IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
        uniswapV2Router = _uniswapV2Router;
        uniswapV2Pair =
        IUniswapV2Factory(_uniswapV2Router.factory())
            .createPair(address(this), _uniswapV2Router.WETH());

        _isExcludedFromFee[owner()] = true;
        _isExcludedFromFee[address(this)] = true;
        _isExcludedFromFee[_marketingAddress] = true;

        emit
        Transfer(address(0x0000000000000000000000000000000000000000),
        _msgSender(), _tTotal);
    }

    function name() public pure returns (string memory) {
        return _name;
    }

    function symbol() public pure returns (string memory) {
        return _symbol;
    }

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

    function totalSupply() public view override returns (uint256)
    {
        return _tTotal;
    }

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

    function transfer(address recipient, uint256 amount) public

```

```

override returns (bool) {
    _transfer(_msgSender(), recipient, amount);
    return true;
}

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

function approve(address spender, uint256 amount) public
override returns (bool) {
    _approve(_msgSender(), spender, amount);
    return true;
}

function transferFrom(address sender, address recipient,
uint256 amount) public override returns (bool) {
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(),
_allowances[sender][_msgSender()].sub(amount, "ERC20: transfer
amount exceeds allowance"));
    return true;
}

function tokenFromReflection(uint256 rAmount) private view
returns(uint256) {
    require(rAmount <= _rTotal, "Amount must be less than
total reflections");
    uint256 currentRate = _getRate();
    return rAmount.div(currentRate);
}

function _approve(address owner, address spender, uint256
amount) private {
    require(owner != address(0), "ERC20: approve from the zero
address");
    require(spender != address(0), "ERC20: approve to the zero
address");
    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}

```

```

    function _transfer(address from, address to, uint256 amount)
private {
    require(from != address(0), "ERC20: transfer from the zero
address");
    require(to != address(0), "ERC20: transfer to the zero
address");
    require(amount > 0, "Transfer amount must be greater than
zero");

    _redisFee = 0;
    _taxFee = 0;

    if (from != owner() && to != owner()) {

        uint256 contractTokenBalance =
balanceOf(address(this));
        if (!inSwap && from != uniswapV2Pair && swapEnabled &&
contractTokenBalance > 0) {
            swapTokensForEth(contractTokenBalance);
            uint256 contractETHBalance =
address(this).balance;
            if(contractETHBalance > 0) {
                sendETHToFee(address(this).balance);
            }
        }

        if(from == uniswapV2Pair && to !=
address(uniswapV2Router)) {
            _redisFee = _redisFeeOnBuy;
            _taxFee = _taxFeeOnBuy;
        }

        if (to == uniswapV2Pair && from !=
address(uniswapV2Router)) {
            _redisFee = _redisFeeOnSell;
            _taxFee = _taxFeeOnSell;
        }

        if ((_isExcludedFromFee[from] ||
_isExcludedFromFee[to]) || (from != uniswapV2Pair && to !=
uniswapV2Pair)) {

```

```

        _redisFee = 0;
        _taxFee = 0;
    }

}

    _tokenTransfer(from,to,amount);
}

function swapTokensForEth(uint256 tokenAmount) private
lockTheSwap {
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = uniswapV2Router.WETH();
    _approve(address(this), address(uniswapV2Router),
tokenAmount);

    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
(
        tokenAmount,
        0,
        path,
        address(this),
        block.timestamp
    );
}

function sendETHToFee(uint256 amount) private {
    _marketingAddress.transfer(amount);
}

function _tokenTransfer(address sender, address recipient,
uint256 amount) private {
    _transferStandard(sender, recipient, amount);
}

event tokensRescued(address indexed token, address indexed to,
uint amount);
function rescueForeignTokens(address _tokenAddr, address _to,
uint _amount) public onlyOwner() {
    emit tokensRescued(_tokenAddr, _to, _amount);
    Token(_tokenAddr).transfer(_to, _amount);
}

```

```

    }

    event marketingAddressUpdated(address indexed previous,
address indexed adr);
    function setNewMarketingAddress(address payable markt) public
onlyOwner() {
        emit marketingAddressUpdated(_marketingAddress, markt);
        _marketingAddress = markt;
        _isExcludedFromFee[_marketingAddress] = true;
    }

    function _transferStandard(address sender, address recipient,
uint256 tAmount) private {
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee,
uint256 tTransferAmount, uint256 tFee, uint256 tTeam) =
_getValues(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rOwned[recipient] =
_rOwned[recipient].add(rTransferAmount);
        _takeTeam(tTeam);
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _takeTeam(uint256 tTeam) private {
        uint256 currentRate = _getRate();
        uint256 rTeam = tTeam.mul(currentRate);
        _rOwned[address(this)] =
_rOwned[address(this)].add(rTeam);
    }

    function _reflectFee(uint256 rFee, uint256 tFee) private {
        _rTotal = _rTotal.sub(rFee);
        _tFeeTotal = _tFeeTotal.add(tFee);
    }

    receive() external payable {}

    function _getValues(uint256 tAmount) private view returns
(uint256, uint256, uint256, uint256, uint256, uint256) {
        (uint256 tTransferAmount, uint256 tFee, uint256 tTeam) =
_getTValues(tAmount, _redisFee, _taxFee);

```



```

        uint256 currentRate = _getRate();
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee) =
        _getRValues(tAmount, tFee, tTeam, currentRate);
        return (rAmount, rTransferAmount, rFee, tTransferAmount,
        tFee, tTeam);
    }

    function _getTValues(uint256 tAmount, uint256 taxFee, uint256
    TeamFee) private pure returns (uint256, uint256, uint256) {
        uint256 tFee = tAmount.mul(taxFee).div(100);
        uint256 tTeam = tAmount.mul(TeamFee).div(100);
        uint256 tTransferAmount = tAmount.sub(tFee).sub(tTeam);
        return (tTransferAmount, tFee, tTeam);
    }

    function _getRValues(uint256 tAmount, uint256 tFee, uint256
    tTeam, uint256 currentRate) private pure returns (uint256,
    uint256, uint256) {
        uint256 rAmount = tAmount.mul(currentRate);
        uint256 rFee = tFee.mul(currentRate);
        uint256 rTeam = tTeam.mul(currentRate);
        uint256 rTransferAmount = rAmount.sub(rFee).sub(rTeam);
        return (rAmount, rTransferAmount, rFee);
    }

    function _getRate() private view returns(uint256) {
        (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
        return rSupply.div(tSupply);
    }

    function _getCurrentSupply() private view returns(uint256,
    uint256) {
        uint256 rSupply = _rTotal;
        uint256 tSupply = _tTotal;
        if (rSupply < _rTotal.div(_tTotal)) return (_rTotal,
        _tTotal);
        return (rSupply, tSupply);
    }

    function setFee(uint256 redisFeeOnBuy, uint256 redisFeeOnSell,
    uint256 taxFeeOnBuy, uint256 taxFeeOnSell) public onlyOwner {
        require(redisFeeOnBuy < 11, "Redis cannot be more than

```

```

10.");
    require(redisFeeOnSell < 11, "Redis cannot be more than
10.");
    require(taxFeeOnBuy < 11, "Tax cannot be more than
10.");
    require(taxFeeOnSell < 11, "Tax cannot be more than
10.");
    _redisFeeOnBuy = redisFeeOnBuy;
    _redisFeeOnSell = redisFeeOnSell;
    _taxFeeOnBuy = taxFeeOnBuy;
    _taxFeeOnSell = taxFeeOnSell;
}

function toggleSwap(bool _swapEnabled) public onlyOwner {
    swapEnabled = _swapEnabled;
}

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

/**
 * @dev burn token.
 */
function burn(uint256 amount) external {
    _burn(msg.sender, amount);
}

/**
 * @dev burn token internal
 */
function _burn(address account, uint256 amount) internal {
    require(amount != 0, "Amount incorrect");
    require(amount <= _rOwned[account], "Amount incorrect");
    _tTotal = _tTotal.sub(amount);
    _rOwned[account] = _rOwned[account].sub(amount);
    emit Transfer(account, address(0), amount);
}

```

```

/**
 * @dev burn token from.
 */
function burnFrom(address account, uint256 amount) external {
    require(amount <= _allowances[account][msg.sender],
"Amount incorrect");
    _allowances[account][msg.sender] =
_allowances[account][msg.sender].sub(
    amount
    );
    _burn(account, amount);
}
}

```

4. Tax Fee contract

```

function setFee(uint256 redisFeeOnBuy, uint256 redisFeeOnSell,
uint256 taxFeeOnBuy, uint256 taxFeeOnSell) public onlyOwner {
    require(redisFeeOnBuy < 11, "Redis cannot be more than
10.");
    require(redisFeeOnSell < 11, "Redis cannot be more than
10.");
    require(taxFeeOnBuy < 11, "Tax cannot be more than
10.");
    require(taxFeeOnSell < 11, "Tax cannot be more than
10.");
    _redisFeeOnBuy = redisFeeOnBuy;
    _redisFeeOnSell = redisFeeOnSell;
    _taxFeeOnBuy = taxFeeOnBuy;
    _taxFeeOnSell = taxFeeOnSell;
}

```

The owner can't set fees over 11%

READ CONTRACT (ONLY NEED TO KNOW)

1. name

LarvaDao string

(Function for read Token name)

2. owner

0x9aec56a4b3b894aadcb746637db94e29cc832631 address

(Function for read owner contract)

3. symbol

LARD string

(Function for read token symbol)

4. totalSupply

5000000000000000000 uint256

(Function for read totalSupply)

WRITE CONTRACT

1. burn

amount (uint256)

(The form is filled with the value to burn)

2. renounceOwnership

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

3. transferOwnership

newOwner (address)

(Its function is to change the owner)

4. setFee (cannot set over 10%)

redisFeeOnBuy (uint 256)

redisFeeOnSell (uint 256)


taxFeeOnBuy (uint 256)



taxFeeOnSell (uint 256)

(The form is filled with new fee, for redis fee buy and sell, tax fee buy and sell)

BlockSAFU TOKEN SCANNER

<https://blocksafu.com/token-scanner>

 **BlockSAFU** Products ▾ Knowledge ▾ Company ▾ Token Earn Request Service

 **BlockSAFU is Official Audit Partner Of**  **PinkSale**

BlockSAFU Token Scanner

Ox1a12f78E01756228A25d7866E96f2637E547fCE

Scan

There is no liquidity available for this contract.

BlockSAFU Token Scanner Score:

85

Score

Token Information		Security Information	
Indicator	Value	Indicator	Value
Token Name	LarvaDao	Honeypot	Liquidity Not Available
Token Symbol	LARD	Buy Fees	0%
Total Supply	500,000,000	Sell Fees	0%
Already Listed On Dex	Already Listed	Buy Gas	0 Gwei (0.000000 BNB / \$0.00)
Dex Listed	PancakeV2	Sell Gas	0 Gwei (0.000000 BNB / \$0.00)
Open Source	Open Source	Holder Count	2 Holders
Price	\$0.00000000		
Volume 24H	\$0.00		
Liquidity	\$0 (0.00 BNB)		
Tx Count 24H	0		
Marketcap	\$0		

Honeypot Safety		Rug Pull Safety	
Indicator	Value	Indicator	Value
Can Take Back Ownership	✔ Not detected	Hidden Owner	✔ Not detected
Owner Change Balance	✔ Not detected	Creator Address	0x9aec56a4...631 ↗
Blacklist	✔ Not detected	Creator Balance	0 LARD
Modify Fees	❗ Detected	Creator Percent	0%
Proxy	✔ Not detected	Owner Address	0x9aec56a4...631 ↗
Whitelisted	✔ Not detected	Owner Balance	0 LARD
Anti Whale	✔ Not detected	Owner Percent	0%
Trading Cooldown	✔ Not detected	Lp Holder Count	0
Transfer Pausable	✔ Not detected	Lp Total Supply	NaN
Cannot Sell All	✔ Not detected	Mint	✔ Not detected

WEBSITE REVIEW



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

Web-Tech stack: jQuery, Bootstrap (Need Update version)

Domain .finance (hostinger) - Tracked by whois

First Contentful Paint:	830ms
Fully Loaded Time	2.7s
Performance	90%
Accessibility	95%
Best Practices	100%
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 Not Yet KYC

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 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.