BlockSAFU

ADVANCE MANUAL
SMART CONTRACT AUDIT

Project: ANGE GAME

Website: https://angegame.com/

BlockSAFU Score:

82

Contract Address:

0xBb29DeDEe6Cd1800B84d19471b25aE9122cc414f

DISCLAMER

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

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

ABOUT THE AUDITOR:

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

OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 10% (owner can’t set fees over 25%).
- Sell 10% (owner can’t set fees over 25%).

Tx Amount

- Owner cannot set a max tx amount.

Transfer Pausable

- Owner can’t pause.

Blacklist

- Owner can’t blacklist.

Ownership

- Owner can’t take back ownership.

Proxy

- This contract has no proxy.

Anti Whale

- Owner can’t limit the number of wallet holdings.

Trading Cooldown

- Owner can’t set the selling time interval.

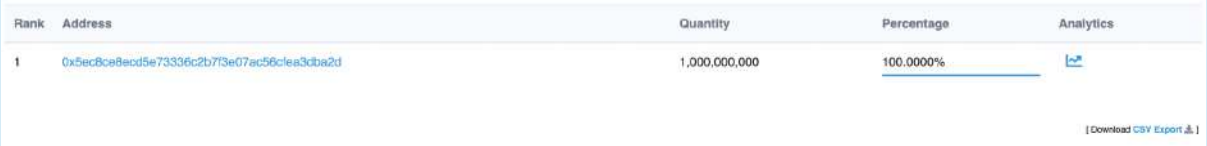
SMART CONTRACT REVIEW

Token Name	**Ange Game Token**
Token Symbol	**ANGE**
Token Decimal	9
Total Supply	1,000,000,000 **ANGE**
Contract Address	0xBb29DeDEe6Cd1800B84d19471b25aE9122cc414f
Deployer Address	0x5ec8ce8eCd5e73336c2B7f3E07ac56cfeA3DBA2D
Owner Address	0x5ec8ce8eCd5e73336c2B7f3E07ac56cfeA3DBA2D
Tax Fees Buy	10%
Tax Fees Sell	10%
Gas Used for Buy	Will be updated after listing on dex
Gas Used for Sell	Will be updated after listing on dex
Contract Created	Oct-03-2022 07:55:07 PM +UTC
Initial Liquidity	Will be updated after listing on dex
Liquidity Status	Locked
Unlocked Date	Will be updated after listing on dex
Verified CA	Yes
Compiler	v0.8.4+commit.c7e474f2
Optimization	Yes with 200 runs
Sol License	MIT License
Other	default evmVersion

TAX

BUY	10%	address	**SELL**	10%
Liquidity Fee	4%	Automatic add liquidity	Liquidity Fee	4%
Marketing Fee	2%	0x7d9efb7ec07baed15937ea46a3403f87d8027c6e	Marketing Fee	2%
Reward Fee	4%	Automatic distribution reward	Reward Fee	4%

Token Holder



Rank	Address	Quantity	Percentage	Analytics
1	0x5ec8ce8ecd5e73336c2b7f3e07ec56c1ea3dba2d	1,000,000,000	100.0000%	

[Download CSV Export]

Team Review

The Agrocoin team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 195 people in their telegram group (count in audit date).

Official Website And Social Media

Website: https://angegame.com/

Telegram Group: https://t.me/angegame2

Twitter: https://twitter.com/AngeGame2

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. ANGEGAME Contract

```
contract LiquidityGeneratorToken is IERC20, Ownable, BaseToken {
    using SafeMath for uint256;
    using Address for address;

    uint256 public constant VERSION = 2;

    uint256 public constant MAX_FEE = 10**4 / 4;

    mapping(address => uint256) private _rOwned;
    mapping(address => uint256) private _tOwned;
    mapping(address => mapping(address => uint256)) private
_allowances;

    mapping(address => bool) private _isExcludedFromFee;
    mapping(address => bool) private _isExcluded;
    address[] private _excluded;

    uint256 private constant MAX = ~uint256(0);
    uint256 private _tTotal;
    uint256 private _rTotal;
    uint256 private _tFeeTotal;

    string private _name;
    string private _symbol;
    uint8 private _decimals;

    uint256 public _taxFee;
    uint256 private _previousTaxFee;

    uint256 public _liquidityFee;
    uint256 private _previousLiquidityFee;

    uint256 public _charityFee;
    uint256 private _previousCharityFee;

    IUniswapV2Router02 public uniswapV2Router;
    address public uniswapV2Pair;
    address public _charityAddress;

    bool inSwapAndLiquify;
    bool public swapAndLiquifyEnabled;
```

```
    uint256 private numTokensSellToAddToLiquidity;

    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
    event SwapAndLiquifyAmountUpdated(uint256 amount);
    event SwapAndLiquify(
        uint256 tokensSwapped,
        uint256 ethReceived,
        uint256 tokensIntoLiqudity
    );

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

    constructor(
        string memory name_,
        string memory symbol_,
        uint256 totalSupply_,
        address router_,
        address charityAddress_,
        uint16 taxFeeBps_,
        uint16 liquidityFeeBps_,
        uint16 charityFeeBps_,
        address serviceFeeReceiver_,
        uint256 serviceFee_
    ) payable {
        if (charityAddress_ == address(0)) {
            require(
                charityFeeBps_ == 0,
                "Cant set both charity address to address 0 and
charity percent more than 0"
            );
        }
        require(
            taxFeeBps_ + liquidityFeeBps_ + charityFeeBps_ <=
MAX_FEE,
            "Total fee is over 25%"
        );
```

```
        _name = name_;
        _symbol = symbol_;
        _decimals = 9;

        _tTotal = totalSupply_;
        _rTotal = (MAX - (MAX % _tTotal));

        _taxFee = taxFeeBps_;
        _previousTaxFee = _taxFee;

        _liquidityFee = liquidityFeeBps_;
        _previousLiquidityFee = _liquidityFee;

        _charityAddress = charityAddress_;
        _charityFee = charityFeeBps_;
        _previousCharityFee = _charityFee;

        numTokensSellToAddToLiquidity = totalSupply_.div(10**3);
// 0.1%

        swapAndLiquifyEnabled = true;

        _rOwned[owner()] = _rTotal;

        IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(router_);
        // Create a uniswap pair for this new token
        uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory())
            .createPair(address(this), _uniswapV2Router.WETH());

        // set the rest of the contract variables
        uniswapV2Router = _uniswapV2Router;

        // exclude owner and this contract from fee
        _isExcludedFromFee[owner()] = true;
        _isExcludedFromFee[address(this)] = true;

        emit Transfer(address(0), owner(), _tTotal);

        emit TokenCreated(
            owner(),
```

```
            address(this),
            TokenType.liquidityGenerator,
            VERSION
        );

        payable(serviceFeeReceiver_).transfer(serviceFee_);
    }

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

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

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

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

    function balanceOf(address account) public view override
returns (uint256) {
        if (_isExcluded[account]) return _tOwned[account];
        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 increaseAllowance(address spender, uint256
addedValue)
        public
        virtual
        returns (bool)
    {
        _approve(
            _msgSender(),
            spender,
```

```
            _allowances[_msgSender()][spender].add(addedValue)
        );
        return true;
    }

    function decreaseAllowance(address spender, uint256
subtractedValue)
        public
        virtual
        returns (bool)
    {
        _approve(
            _msgSender(),
            spender,
            _allowances[_msgSender()][spender].sub(
                subtractedValue,
                "ERC20: decreased allowance below zero"
            )
        );
        return true;
    }

    function isExcludedFromReward(address account) public view
returns (bool) {
        return _isExcluded[account];
    }

    function totalFees() public view returns (uint256) {
        return _tFeeTotal;
    }

    function deliver(uint256 tAmount) public {
        address sender = _msgSender();
        require(
            !_isExcluded[sender],
            "Excluded addresses cannot call this function"
        );
        (uint256 rAmount, , , , , , ) = _getValues(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rTotal = _rTotal.sub(rAmount);
        _tFeeTotal = _tFeeTotal.add(tAmount);
    }
```

```
    function reflectionFromToken(uint256 tAmount, bool
deductTransferFee)
        public
        view
        returns (uint256)
    {
        require(tAmount <= _tTotal, "Amount must be less than
supply");
        if (!deductTransferFee) {
            (uint256 rAmount, , , , , , ) = _getValues(tAmount);
            return rAmount;
        } else {
            (, uint256 rTransferAmount, , , , , ) =
_getValues(tAmount);
            return rTransferAmount;
        }
    }

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

    function excludeFromReward(address account) public onlyOwner {
        // require(account !=
0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude
Uniswap router.');
        require(!_isExcluded[account], "Account is already
excluded");
        if (_rOwned[account] > 0) {
            _tOwned[account] =
tokenFromReflection(_rOwned[account]);
        }
```

```
        _isExcluded[account] = true;
        _excluded.push(account);
    }

    function includeInReward(address account) external onlyOwner {
        require(_isExcluded[account], "Account is already
excluded");
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_excluded[i] == account) {
                _excluded[i] = _excluded[_excluded.length - 1];
                _tOwned[account] = 0;
                _isExcluded[account] = false;
                _excluded.pop();
                break;
            }
        }
    }

    function _transferBothExcluded(
        address sender,
        address recipient,
        uint256 tAmount
    ) private {
        (
            uint256 rAmount,
            uint256 rTransferAmount,
            uint256 rFee,
            uint256 tTransferAmount,
            uint256 tFee,
            uint256 tLiquidity,
            uint256 tCharity
        ) = _getValues(tAmount);
        _tOwned[sender] = _tOwned[sender].sub(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _tOwned[recipient] =
_tOwned[recipient].add(tTransferAmount);
        _rOwned[recipient] =
_rOwned[recipient].add(rTransferAmount);
        _takeLiquidity(tLiquidity);
        _takeCharityFee(tCharity);
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
```

```
    }

    function excludeFromFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = true;
    }

    function setTaxFeePercent(uint256 taxFeeBps) external
onlyOwner {
        _taxFee = taxFeeBps;
        require(
            _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
            "Total fee is over 25%"
        );
    }

    function setLiquidityFeePercent(uint256 liquidityFeeBps)
        external
        onlyOwner
    {
        _liquidityFee = liquidityFeeBps;
        require(
            _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
            "Total fee is over 25%"
        );
    }

    function setCharityFeePercent(uint256 charityFeeBps) external
onlyOwner {
        _charityFee = charityFeeBps;
        require(
            _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
            "Total fee is over 25%"
        );
    }

    function setSwapBackSettings(uint256 _amount) external
onlyOwner {
        require(
            _amount >= totalSupply().mul(5).div(10**4),
            "Swapback amount should be at least 0.05% of total
supply"
        );
```

```
        numTokensSellToAddToLiquidity = _amount;
        emit SwapAndLiquifyAmountUpdated(_amount);
    }

    //to recieve ETH from uniswapV2Router when swaping
    receive() external payable {}

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

    function _getValues(uint256 tAmount)
        private
        view
        returns (
            uint256,
            uint256,
            uint256,
            uint256,
            uint256,
            uint256,
            uint256
        )
    {
        (
            uint256 tTransferAmount,
            uint256 tFee,
            uint256 tLiquidity,
            uint256 tCharity
        ) = _getTValues(tAmount);
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee) =
_getRValues(
            tAmount,
            tFee,
            tLiquidity,
            tCharity,
            _getRate()
        );
        return (
            rAmount,
            rTransferAmount,
```

```
            rFee,
            tTransferAmount,
            tFee,
            tLiquidity,
            tCharity
        );
    }

    function _getTValues(uint256 tAmount)
        private
        view
        returns (
            uint256,
            uint256,
            uint256,
            uint256
        )
    {
        uint256 tFee = calculateTaxFee(tAmount);
        uint256 tLiquidity = calculateLiquidityFee(tAmount);
        uint256 tCharityFee = calculateCharityFee(tAmount);
        uint256 tTransferAmount =
tAmount.sub(tFee).sub(tLiquidity).sub(
            tCharityFee
        );
        return (tTransferAmount, tFee, tLiquidity, tCharityFee);
    }

    function _getRValues(
        uint256 tAmount,
        uint256 tFee,
        uint256 tLiquidity,
        uint256 tCharity,
        uint256 currentRate
    )
        private
        pure
        returns (
            uint256,
            uint256,
            uint256
        )
```

```
    {
        uint256 rAmount = tAmount.mul(currentRate);
        uint256 rFee = tFee.mul(currentRate);
        uint256 rLiquidity = tLiquidity.mul(currentRate);
        uint256 rCharity = tCharity.mul(currentRate);
        uint256 rTransferAmount =
rAmount.sub(rFee).sub(rLiquidity).sub(
            rCharity
        );
        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;
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (
                _rOwned[_excluded[i]] > rSupply ||
                _tOwned[_excluded[i]] > tSupply
            ) return (_rTotal, _tTotal);
            rSupply = rSupply.sub(_rOwned[_excluded[i]]);
            tSupply = tSupply.sub(_tOwned[_excluded[i]]);
        }
        if (rSupply < _rTotal.div(_tTotal)) return (_rTotal,
_tTotal);
        return (rSupply, tSupply);
    }

    function _takeLiquidity(uint256 tLiquidity) private {
        uint256 currentRate = _getRate();
        uint256 rLiquidity = tLiquidity.mul(currentRate);
        _rOwned[address(this)] =
_rOwned[address(this)].add(rLiquidity);
        if (_isExcluded[address(this)])
            _tOwned[address(this)] =
_tOwned[address(this)].add(tLiquidity);
```

```
    }

    function _takeCharityFee(uint256 tCharity) private {
        if (tCharity > 0) {
            uint256 currentRate = _getRate();
            uint256 rCharity = tCharity.mul(currentRate);
            _rOwned[_charityAddress] =
_rOwned[_charityAddress].add(rCharity);
            if (_isExcluded[_charityAddress])
                _tOwned[_charityAddress] =
_tOwned[_charityAddress].add(
                    tCharity
                );
            emit Transfer(_msgSender(), _charityAddress,
tCharity);
        }
    }

    function calculateTaxFee(uint256 _amount) private view returns
(uint256) {
        return _amount.mul(_taxFee).div(10**4);
    }

    function calculateLiquidityFee(uint256 _amount)
        private
        view
        returns (uint256)
    {
        return _amount.mul(_liquidityFee).div(10**4);
    }

    function calculateCharityFee(uint256 _amount)
        private
        view
        returns (uint256)
    {
        if (_charityAddress == address(0)) return 0;
        return _amount.mul(_charityFee).div(10**4);
    }

    function removeAllFee() private {
        _previousTaxFee = _taxFee;
```

```
        _previousLiquidityFee = _liquidityFee;
        _previousCharityFee = _charityFee;

        _taxFee = 0;
        _liquidityFee = 0;
        _charityFee = 0;
    }

    function restoreAllFee() private {
        _taxFee = _previousTaxFee;
        _liquidityFee = _previousLiquidityFee;
        _charityFee = _previousCharityFee;
    }

    function isExcludedFromFee(address account) public view
returns (bool) {
        return _isExcludedFromFee[account];
    }

    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");

        // is the token balance of this contract address over the
min number of
        // tokens that we need to initiate a swap + liquidity
lock?
        // also, don't get caught in a circular liquidity event.
        // also, don't swap & liquify if sender is uniswap pair.
        uint256 contractTokenBalance = balanceOf(address(this));

        bool overMinTokenBalance = contractTokenBalance >=
            numTokensSellToAddToLiquidity;
        if (
            overMinTokenBalance &&
            !inSwapAndLiquify &&
            from != uniswapV2Pair &&
            swapAndLiquifyEnabled
        ) {
            contractTokenBalance = numTokensSellToAddToLiquidity;
            //add liquidity
            swapAndLiquify(contractTokenBalance);
        }

        //indicates if fee should be deducted from transfer
        bool takeFee = true;

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

        //transfer amount, it will take tax, burn, liquidity fee
        _tokenTransfer(from, to, amount, takeFee);
    }

    function swapAndLiquify(uint256 contractTokenBalance) private
lockTheSwap {
        // split the contract balance into halves
        uint256 half = contractTokenBalance.div(2);
        uint256 otherHalf = contractTokenBalance.sub(half);
```

```
        // capture the contract's current ETH balance.
        // this is so that we can capture exactly the amount of
ETH that the
        // swap creates, and not make the liquidity event include
any ETH that
        // has been manually sent to the contract
        uint256 initialBalance = address(this).balance;

        // swap tokens for ETH
        swapTokensForEth(half); // <- this breaks the ETH -> HATE
swap when swap+liquify is triggered

        // how much ETH did we just swap into?
        uint256 newBalance =
address(this).balance.sub(initialBalance);

        // add liquidity to uniswap
        addLiquidity(otherHalf, newBalance);

        emit SwapAndLiquify(half, newBalance, otherHalf);
    }

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

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

        // make the swap

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

```
    }

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

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

    //this method is responsible for taking all fee, if takeFee is
true
    function _tokenTransfer(
        address sender,
        address recipient,
        uint256 amount,
        bool takeFee
    ) private {
        if (!takeFee) removeAllFee();

        if (_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferFromExcluded(sender, recipient, amount);
        } else if (!_isExcluded[sender] && _isExcluded[recipient])
{
            _transferToExcluded(sender, recipient, amount);
        } else if (!_isExcluded[sender] &&
!_isExcluded[recipient]) {
            _transferStandard(sender, recipient, amount);
        } else if (_isExcluded[sender] && _isExcluded[recipient])
{
            _transferBothExcluded(sender, recipient, amount);
        } else {
            _transferStandard(sender, recipient, amount);
```

```
        }

        if (!takeFee) restoreAllFee();
    }

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

    function _transferToExcluded(
        address sender,
        address recipient,
        uint256 tAmount
    ) private {
        (
            uint256 rAmount,
            uint256 rTransferAmount,
            uint256 rFee,
            uint256 tTransferAmount,
            uint256 tFee,
            uint256 tLiquidity,
            uint256 tCharity
        ) = _getValues(tAmount);
```

```
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _tOwned[recipient] =
_tOwned[recipient].add(tTransferAmount);
        _rOwned[recipient] =
_rOwned[recipient].add(rTransferAmount);
        _takeLiquidity(tLiquidity);
        _takeCharityFee(tCharity);
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _transferFromExcluded(
        address sender,
        address recipient,
        uint256 tAmount
    ) private {
        (
            uint256 rAmount,
            uint256 rTransferAmount,
            uint256 rFee,
            uint256 tTransferAmount,
            uint256 tFee,
            uint256 tLiquidity,
            uint256 tCharity
        ) = _getValues(tAmount);
        _tOwned[sender] = _tOwned[sender].sub(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rOwned[recipient] =
_rOwned[recipient].add(rTransferAmount);
        _takeLiquidity(tLiquidity);
        _takeCharityFee(tCharity);
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }
}
```

4. Tax Fee contract

```
function setTaxFeePercent(uint256 taxFeeBps) external onlyOwner {
        _taxFee = taxFeeBps;
        require(
            _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
            "Total fee is over 25%"
```

```
        );
    }

    function setLiquidityFeePercent(uint256 liquidityFeeBps)
        external
        onlyOwner
    {
        _liquidityFee = liquidityFeeBps;
        require(
            _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
            "Total fee is over 25%"
        );
    }

    function setCharityFeePercent(uint256 charityFeeBps) external
onlyOwner {
        _charityFee = charityFeeBps;
        require(
            _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
            "Total fee is over 25%"
        );
    }
```

The owner can’t set fees over 25%

READ CONTRACT (ONLY NEED TO KNOW)

1. Version
2 uint256

(Shows Contract Versions)

2. _charityAddress
0x7d9efb7ec07baed15937ea46a3403f87d8027c6e address
(Shows charity wallet address)

3. _charityFee
200 uint256
(Function for read charity fee)

4. _liquidityFee
400 uint256
(Function for read liquidity fee)

5. _taxFee
400 uint256
(Function for read token tax fee)

6. name
Ange Game Token string
(Function for read Token name)

WRITE CONTRACT

1. renounceOwnership
(Renouncing ownership will leave the contract without an

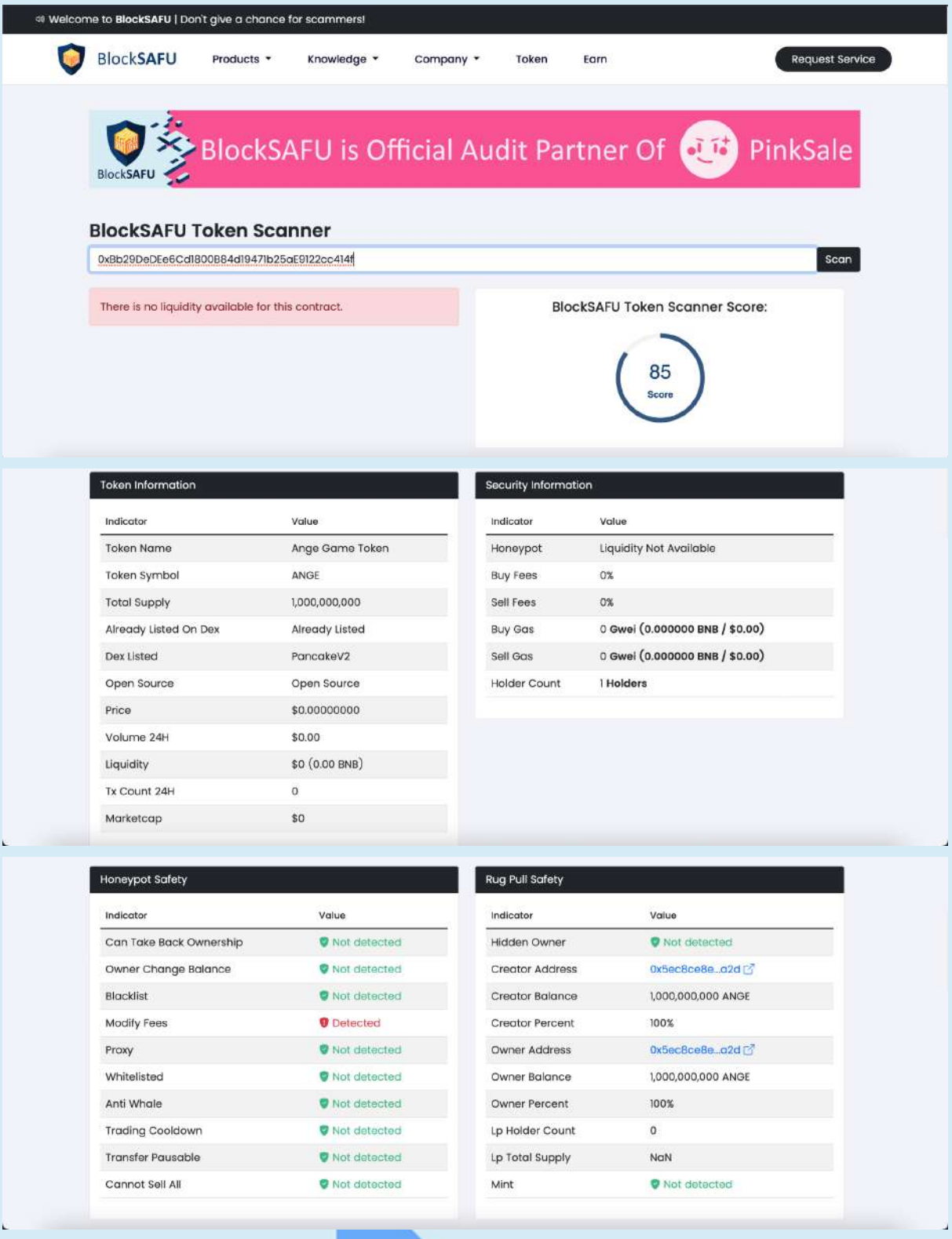
owner, thereby removing any functionality that is only available to the owner)

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

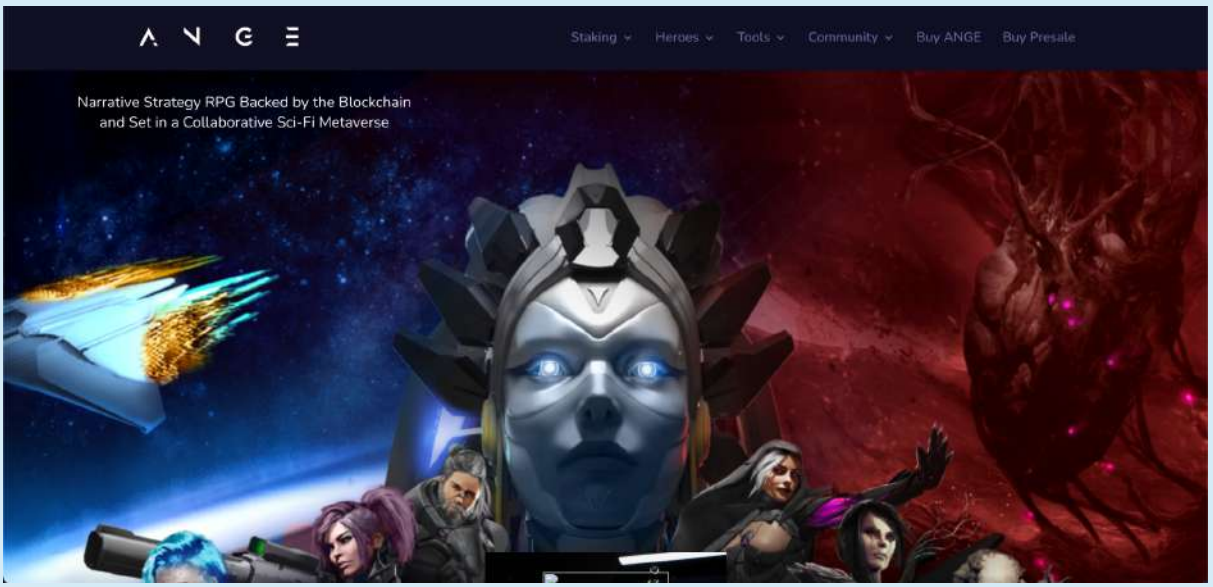
3. setLiquidityFeePercent (cannot set over 25%)
liquidityFeeBps (uint 256)
(The form is filled with new fee, for change liquidity fee)

4. setCharityFeePercent (cannot set over 25%)
charityFeeBps (uint 256)
(The form is filled with new fee, for change charity fee)

5. setTaxFeePercent
taxFeeBps (uint 256)
(The form is filled with new fee, for change Tax fee)



WEBSITE REVIEW



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

Web-Tech stack: Wordpress, sectigo

Domain .com (namecheaphosting) - Tracked by whois

First Contentful Paint:	1.4s
Fully Loaded Time	18.5s
Performance	75%
Accessibility	82%
Best Practices	67%
SEO	75%

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 listing dex

- TOP 5 Holder.

  will be updated after listing dex

- The Team is not KYC By 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.