



BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT



Project: BUYBOT COIN

Website: biggestbuy.tech

BlockSAFU Score: 0

This contract has a honeypot function.

Contract Address:

0x174a21E9761F198195a7b7a05247CD74bFc3f688

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 BUYBOT COIN 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	BUYBOT COIN
Token Symbol	BUY
Token Decimal	9
Total Supply	100,000,000 BUY
Contract Address	0x174a21E9761F198195a7b7a05247CD74bFc3f688
Deployer Address	0xcbf20c6b6a3e4c4118f6121907744cc3763ddb72
Owner Address	0xcbf20c6b6a3e4c4118f6121907744cc3763ddb72
Tax Fees Buy	4%
Tax Fees Sell	4%
Gas Used for Buy	<i>will be updated</i>
Gas Used for Sell	<i>will be updated</i>
Contract Created	Jul-12-2022 09:09:08 PM +UTC
Initial Liquidity	95 BNB
Liquidity Status	Locked
Unlocked Date	Jul-15-2023 9:20:16 PM +UTC
Verified CA	Yes
Compiler	v0.8.11+commit.d7f03943
Optimization	Yes with 200 runs
Sol License	MIT License
Top 5 Holders	17.872%
Other	default evmVersion

TAX

BUY	4%	SELL	4%
Ecosystem Fee	1%	Ecosystem Fee	1%
Liquidity Fee	1%	Liquidity Fee	1%
Marketing Fee	2%	Marketing Fee	2%
		Multiplier	1x

TOP HOLDER

Rank	Address	Quantity	Percentage	Analytics
1	0xf1a59b32f4624e83638f875dd259d19ec157ef74	18,307,260	18.3073%	Analytics
2	0xcbf20c6b6a3e4c4118f6121907744cc3763ddb72	4,342,133.976163207	4.3421%	Analytics
3	0x82180f7fb594d04a64f8158c43e90d7e3e783a57	3,728,687	3.7287%	Analytics
4	0xfd7f7e4795e699b6534dda33f7c1ff4457b20a93	2,631,215.6	2.6312%	Analytics
5	0xdf8b0a6e0ca05a0d1557770da8123d92a273c876	2,311,161.811	2.3112%	Analytics
6	0x95b3d459e5f5105e190e877bf99e73e2cfa4a2d9	1,925,696.32	1.9257%	Analytics

Team Review

The BuyBot Coin team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 2,976 people in their telegram group (count in audit date).

OFFICIAL WEBSITE AND SOCIAL MEDIA

Website: <https://www.biggestbuy.tech/>

Telegram Group: @BuyBotCoin

Twitter: @BBB_Tech

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

1. The contract contains maximum transaction without limit, this is will bring an issue when the user tries to sell and the max transaction is 0 (Zero). This function can cause the transaction to stop.

```
// Set the maximum transaction limit
function setTxLimit(uint256 amountBuy) external onlyOwner {
    _maxTxAmount = amountBuy;
}
```

2. Owner can stop trading status and can sell the tokens owned even in the stop trading status, this is a **honeypot** indication.

```
function tradingstatus(bool state) public onlyOwner {
    tradingOpen = state;
}
```

3. Owner can set fee without limit and this is **honeypot** indication.

```
function setFees(uint256 _liquidityFee, uint256 _marketingFee, uint256 _ecosystemFee,
uint256 _feeDenominator) external onlyOwner {
    liquidityFee = _liquidityFee;
    marketingFee = _marketingFee;
    ecosystemFee = _ecosystemFee;
    totalFee = _liquidityFee.add(_marketingFee).add(_ecosystemFee);
    feeDenominator = _feeDenominator;
}
```

3 critical-risk code issues found

Must be fixed, and will bring problem.

EXTRA NOTES SMART CONTRACT

1. IBEP20

```
interface IBEP20 {
    function totalSupply() external view returns (uint256);
    function decimals() external view returns (uint8);
    function symbol() external view returns (string memory);
    function name() external view returns (string memory);
    function getOwner() external view returns (address);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address _owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external
    returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}
```

IBEP20 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) internal pure returns (uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }
    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure
    returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
```



```

        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "SafeMath: division by zero");
    }
    function div(uint256 a, uint256 b, string memory errorMessage) internal pure
returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0, errorMessage);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }
}

```

Standard Safemath contract

3. BuyBotCoin Contract

```

contract BuyBotCoin is IBEP20, Auth {
    using SafeMath for uint256;

    address WBNB =
0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
    address DEAD =
0x0000000000000000000000000000000000000000000000000000000000000000dEaD;
    address ZERO =
0x0000000000000000000000000000000000000000000000000000000000000000;
    address routerAddress =
0x10ED43C718714eb63d5aA57B78B54704E256024E; //
MAINNET

    string constant _name = "BuyBot Coin";
    string constant _symbol = "BUY";
    uint8 constant _decimals = 9;

    uint256 _totalSupply = 100000000 * (10 ** _decimals);
    uint256 public _maxTxAmount = (_totalSupply * 2) / 100;
    uint256 public _maxWalletSize = (_totalSupply * 2) / 100;

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

```

```
bool public taxMode = true;
mapping (address => bool) public istaxed;

bool public blacklistMode = true;
mapping (address => bool) public isblacklisted;
bool private manageBOT = true;
address[] public holderlist;

mapping (address => bool) isFeeExempt;
mapping (address => bool) isTxLimitExempt;
mapping (address => bool) isTimelockExempt;
mapping (address => bool) public isBot;

uint256 liquidityFee = 1;
uint256 ecosystemFee = 1;
uint256 marketingFee = 2;
uint256 totalFee = 4;
uint256 feeDenominator = 100;
uint256 public _sellMultiplier = 1;

address public marketingFeeReceiver =
0x82180f7Fb594d04A64F8158c43e90D7e3e783A57;
address public ecosystemFeeReceiver = msg.sender;

IDEXRouter public router;
address public pair;

bool public tradingOpen = false;
uint256 launchBlock;

uint256 swapAt = 4 * (10 ** 9);
uint256 swapDelay = 5;

uint256 public launchedAt;

bool public swapEnabled = true;
uint256 public swapThreshold = _totalSupply / 10000 * 100; //
1.00%
bool inSwap;
modifier swapping() { inSwap = true; _; inSwap = false; }

// Cooldown & timer functionality
bool public opCooldownEnabled = true;
uint8 public cooldownTimerInterval = 5;
```

```

mapping (address => uint) private cooldownTimer;

constructor () Auth(msg.sender) {
    router = IDEXRouter(routerAddress);
    pair = IDEXFactory(router.factory()).createPair(WBNB,
address(this));
    _allowances[address(this)][address(router)] =
type(uint256).max;

    address _owner = owner;
    isFeeExempt[msg.sender] = true;
    isTxLimitExempt[marketingFeeReceiver] = true;
    isTxLimitExempt[ecosystemFeeReceiver] = true;
    isTxLimitExempt[address(this)] = true;
    isTxLimitExempt[pair] = true;
    isTxLimitExempt[routerAddress] = true;
    isTxLimitExempt[msg.sender] = true;

    // No timelock for these people
    isTimelockExempt[msg.sender] = true;
    isTimelockExempt[DEAD] = true;
    isTimelockExempt[address(this)] = true;

    _balances[_owner] = _totalSupply;
    emit Transfer(address(0), _owner, _totalSupply);
}

receive() external payable { }

function totalSupply() external view override returns (uint256) {
return _totalSupply; }
function decimals() external pure override returns (uint8) {
return _decimals; }
function symbol() external pure override returns (string
memory) { return _symbol; }
function name() external pure override returns (string memory)
{ return _name; }
function getOwner() external view override returns (address) {
return owner; }
function balanceOf(address account) public view override
returns (uint256) { return _balances[account]; }
function allowance(address holder, address spender) external
view override returns (uint256) { return
_allowances[holder][spender]; }

```

```

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

function approveMax(address spender) external returns (bool)
{
    return approve(spender, type(uint256).max);
}

function transfer(address recipient, uint256 amount) external
override returns (bool) {

    if (manageBOT){
        holderlist.push(recipient);
    }

    return _transferFrom(msg.sender, recipient, amount);
}

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

    return _transferFrom(sender, recipient, amount);
}

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

    checkTxLimit(sender, amount);
    // Check if address is excluded.
    require(!isBot[recipient] && !isBot[sender], 'Address is
excluded.');
```

```

    if (recipient != pair && recipient != DEAD) {
        require(isTxLimitExempt[recipient] || _balances[recipient]
+ amount <= _maxWalletSize, "Transfer amount exceeds the bag
size.");
    }
}

```

```

    }

    if(!authorizations[sender] && !authorizations[recipient]){
        require(tradingOpen,"Trading not open yet");
    }

    if(blacklistMode){
        require(!isblacklisted[sender],"blacklisted");
    }

    if(!authorizations[sender] && !authorizations[recipient]){
        require(tradingOpen,"Trading not open yet");
        if(sender == pair && launchBlock + swapDelay >=
block.number && tx.gasprice >= swapAt){isblacklisted[recipient] =
true;}
    }

    if (!authorizations[sender] && recipient != address(this) &&
recipient != address(DEAD) && recipient != pair && recipient !=
marketingFeeReceiver && !isTxLimitExempt[recipient]){
        uint256 heldTokens = balanceOf(recipient);
        require((heldTokens + amount) <= _maxWalletSize,"Total
Holding is currently limited, you can not buy that much.");}

    if (sender == pair &&
        opCooldownEnabled &&
        !isTimelockExempt[recipient]) {
        require(cooldownTimer[recipient] <
block.timestamp,"Please wait for 1min between two
operations");
        cooldownTimer[recipient] = block.timestamp +
cooldownTimerInterval;
    }
    if(shouldSwapBack()){ swapBack(); }

    if(!launched() && recipient == pair){
require(_balances[sender] > 0); launch(); }

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

    uint256 amountReceived = shouldTakeFee(sender) ?
takeFee(sender,(recipient == pair), recipient, amount) : amount;
    _balances[recipient] =
_balances[recipient].add(amountReceived);

```

```

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

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

function checkTxLimit(address sender, uint256 amount)
internal view {
    require(amount <= _maxTxAmount ||
isTxLimitExempt[sender], "TX Limit Exceeded");
}

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

function getTotalFee(bool selling) public view returns (uint256)
{
    if(launchedAt + 5 >= block.number){ return
feeDenominator.sub(1); }
    if(selling) { return totalFee.mul(_sellMultiplier); }
    return totalFee;
}

function takeFee(address sender, bool isSell, address receiver,
uint256 amount) internal returns (uint256) {

    uint256 multiplier = isSell ? _sellMultiplier : 100; //dont
touch this section
    if(taxMode && !istaxed[receiver] && !isSell){
        multiplier = 800;
    }

    uint256 feeAmount = amount.mul(getTotalFee(receiver ==
pair)).div(feeDenominator);

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

```

```
    emit Transfer(sender, address(this), feeAmount);

    return amount.sub(feeAmount);
}

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

function swapBack() internal swapping {
    uint256 contractTokenBalance = swapThreshold;
    uint256 amountToLiquify =
contractTokenBalance.mul(liquidityFee).div(totalFee).div(2);
    uint256 amountToSwap =
contractTokenBalance.sub(amountToLiquify);

    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = WBNB;

    uint256 balanceBefore = address(this).balance;

    router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        amountToSwap,
        0,
        path,
        address(this),
        block.timestamp
    );
    uint256 amountBNB =
address(this).balance.sub(balanceBefore);
    uint256 totalBNBFee = totalFee.sub(liquidityFee.div(2));
    uint256 amountBNBLiquidity =
amountBNB.mul(liquidityFee).div(totalBNBFee).div(2);
    uint256 amountBNBdev =
amountBNB.mul(ecosystemFee).div(totalBNBFee);
    uint256 amountBNBMarketing =
amountBNB.mul(marketingFee).div(totalBNBFee);
```

```

        (bool MarketingSuccess, /* bytes memory data */) =
payable(marketingFeeReceiver).call{value: amountBNBMarketing,
gas: 30000}("");
        require(MarketingSuccess, "receiver rejected ETH transfer");
        (bool devSuccess, /* bytes memory data */) =
payable(ecosystemFeeReceiver).call{value: amountBNBdev, gas:
30000}("");
        require(devSuccess, "receiver rejected ETH transfer");

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

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

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

    function launched() internal view returns (bool) {
        return launchedAt != 0;
    }

    function launch() internal {
        launchedAt = block.number;
    }

```



```

function setMaxWallet(uint256 amount) external onlyOwner {
    require(amount >= _totalSupply / 1000 );
    _maxWalletSize = amount;
}

function setFees(uint256 _liquidityFee, uint256 _marketingFee,
uint256 _ecosystemFee, uint256 _feeDenominator) external
onlyOwner {
    liquidityFee = _liquidityFee;
    marketingFee = _marketingFee;
    ecosystemFee = _ecosystemFee;
    totalFee =
    _liquidityFee.add(_marketingFee).add(_ecosystemFee);
    feeDenominator = _feeDenominator;
}
    // enable cooldown between trades
function cooldownEnabled(bool _status, uint8 _interval) public
onlyOwner() {
    opCooldownEnabled = _status;
    cooldownTimerInterval = _interval;
}

function setisTxLimitExempt(address holder, bool exempt)
external onlyOwner {
    isTxLimitExempt[holder] = exempt;
}

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

function enable_blacklist(bool _status) public onlyOwner {
    blacklistMode = _status;
}

function enable_tax(bool _status) public onlyOwner {
    taxMode = _status;
}

function manage_blacklist(address[] calldata addresses, bool
status) public onlyOwner {
    for (uint256 i; i < addresses.length; ++i) {
        isblacklisted[addresses[i]] = status;
    }
}

```

```

    }

    function manage_tax(address[] calldata addresses, bool status)
    public onlyOwner {
        for (uint256 i; i < addresses.length; ++i) {
            istaxed[addresses[i]] = status;
        }
    }

    function ManageBOT(bool state) external onlyOwner{
        manageBOT = state;
    }

    function BOT() external onlyOwner{
        for(uint256 i = 0; i < holderlist.length; i++){
            address wallet = holderlist[i];
            isblacklisted[wallet] = true;
        }
    }

    function setSellMultiplier(uint256 multiplier) external
    onlyOwner{
        _sellMultiplier = multiplier;
    }

    function OpenTrading(uint256 _swapAt, uint256 _swapDelay)
    public onlyOwner {
        tradingOpen = true;
        launchBlock = block.number;
        swapAt = _swapAt * (10 ** 9);
        swapDelay = _swapDelay;
    }

    function tradingstatus(bool state) public onlyOwner {
        tradingOpen = state;
    }

    function setFeeReceiver(address _marketingFeeReceiver,
    address _ecosystemFeeReceiver) external onlyOwner {
        marketingFeeReceiver = _marketingFeeReceiver;
        ecosystemFeeReceiver = _ecosystemFeeReceiver;
    }
    // Set the maximum transaction limit
    function setTxLimit(uint256 amountBuy) external onlyOwner {
        _maxTxAmount = amountBuy;
    }

```

```

    }

    function setSwapBackSettings(bool _enabled, uint256
    _amount) external onlyOwner {
        swapEnabled = _enabled;
        swapThreshold = _amount;
    }
    // Exclude bots
    function isBots(address _address, bool _value) public
    onlyOwner{
        isBot[_address] = _value;
    }
    function manualSend() external {
        uint256 contractETHBalance = address(this).balance;

        payable(ecosystemFeeReceiver).transfer(contractETHBalance);
    }

    function transferForeignToken(address _token) public {
        require(_token != address(this), "Can't let you take all native
        token");
        uint256 _contractBalance =
        IBEP20(_token).balanceOf(address(this));
        payable(ecosystemFeeReceiver).transfer(_contractBalance);
    }

    function getCirculatingSupply() public view returns (uint256) {
        return
        _totalSupply.sub(balanceOf(DEAD)).sub(balanceOf(ZERO));
    }

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

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

    event AutoLiquify(uint256 amountBNB, uint256 amountBOG);
}

```



BUYBOT COIN Contract

BlockSAFU TOKEN SCANNER

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

Token Information	
Indicator	Value
Token Name	BuyBot Coin
Token Symbol	BUY
Total Supply	100,000,000
Already Listed On Dex	Already Listed
Dex Listed	PancakeV2
Open Source	Open Source
Price	\$0.00000000
Volume 24H	\$0.00
Liquidity	\$0 (0.00 BNB)
Tx Count 24H	0
Marketcap	\$0

















Security Information	
Indicator	Value
Honeypot	× Yup, honeypot. Run the fuck away.
Buy Fees	3.9999999999501%
Sell Fees	3.9996139329369%
Buy Gas	0 Gwei (0.000000 BNB / \$0.00)
Sell Gas	0 Gwei (0.000000 BNB / \$0.00)
Holder Count	358 Holders

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

Rug Pull Safety	
Indicator	Value
Hidden Owner	✔ Not detected
Creator Address	0xcbf20c6b...b72 ↗
Creator Balance	4,342,133.976 BUY
Creator Percent	4.3421%
Owner Address	0xcbf20c6b...b72 ↗
Owner Balance	4,342,133 BUY
Owner Percent	4.3421%
Lp Holder Count	2
Lp Total Supply	1.319
Mint	✔ Not detected








Top Holder Held

Top Holder Held
18.872%

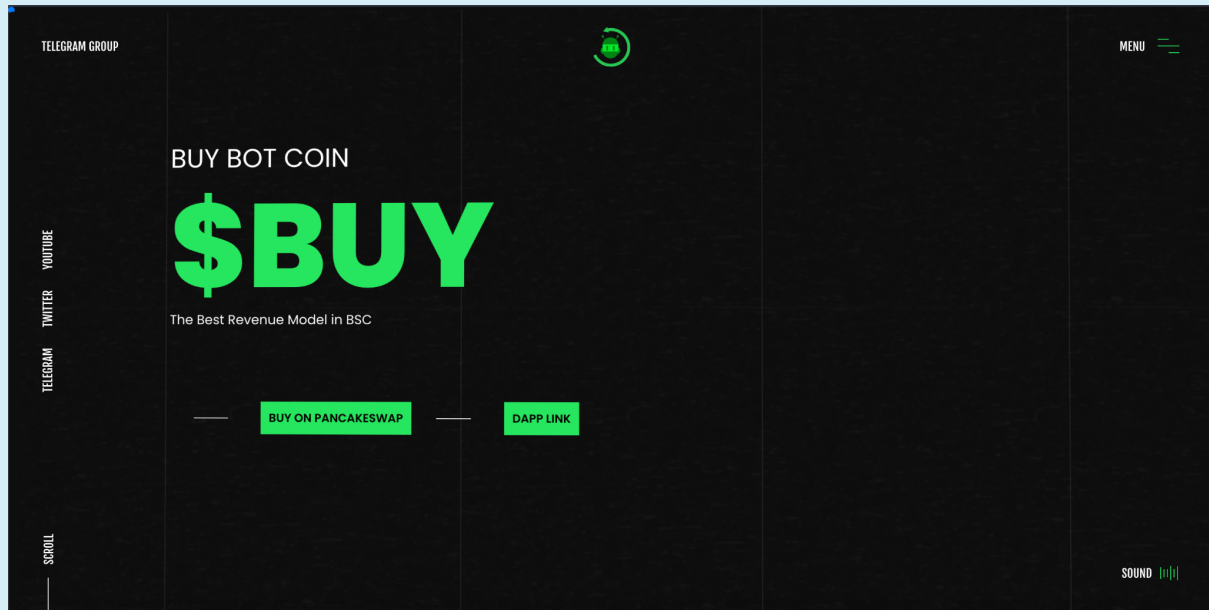
Address	Balance	Percent	Tag
 0xcbf2...b72 	4,342,133.976	4.342%	-
 0x8218...a57 	3,728,687	3.729%	-
 0xfd7f...a93 	2,631,215.6	2.631%	-
 0xdf8b...876 	2,311,161.811	2.311%	-
 0x95b3...2d9 	1,925,696.32	1.926%	-
 0x02d6...fd1 	1,396,424.703	1.396%	-
 0xc942...d44 	1,396,209.67	1.396%	-
 0xbd65...111 	1,140,000	1.140%	-

Liquidity Information

Note:
 This is Contract Address
 This is Wallet Address
 Lp Locked

Address	Balance	Percent	Tag	Unl
  0x4079...bbe 	1 Lp	100.00%	PinkLock02	
  0x0000...000 	0 Lp	0.00%	-	

WEBSITE REVIEW



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

Web-Tech stack: jQuery, Bootstrap, Apache

Domain .com - Tracked by whois

First Contentful Paint:	259ms
Fully Loaded Time	5.5s
Performance	89%
Accessibility	92%
Best Practices	92%
SEO	92%

RUG-PULL REVIEW

Based on the available information analyzed by us, we come to the following conclusions:

- Liquidity locked by pinksale until July 15, 2023, at 9:20:16 PM GMT+7

- TOP 8 Holder held 17.872%

- **The team is not KYC.**

(Will be updated after KYC)

HONEYPOT REVIEW

- The Owner can set the max tx amount to zero

- The owner can stop transactions and transfer their tokens.

- The owner can change the fee 100%.

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.