



BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT



Project: Doken

Website: <https://doken.dev/>

BlockSAFU Score:

31

Contract Address:

0x0420EB45AC5a4f04763f679c07c3A637741E0289

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 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. 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

Mint Function

- No mint functions.

Fees

- Buy 9% (owner can't set fees over 20%).
- Sell 9% (owner can't set fees over 20%).

Tx Amount

- Owner can't set a max tx amount below 0.7% from total supply.

Transfer Pausable

- **Owner can pause.**

Blacklist

- **Owner can blacklist.**

Ownership

- Owner cannot take back ownership.

Proxy

- This contract has no proxy.

Anti Whale

- Owner can limit the number of wallet holdings.

Trading Cooldown

- Owner cannot set the selling time interval.

SMART CONTRACT REVIEW

Token Name	DoKEN V2
Token Symbol	DKN
Token Decimal	18
Total Supply	1,000,000,000,000 DKN
Contract Address	0x0420EB45AC5a4f04763f679c07c3A637741E0289
Deployer Address	0x56119818404E5Cc64A526e78B67F3dc496bE743d
Owner Address	0x56119818404e5cc64a526e78b67f3dc496be743d
Tax Fees Buy	9%
Tax Fees Sell	9%
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	Sep-17-2022 02:21:13 AM +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	v0.7.6+commit.7338295f
Optimization	Yes with 200 runs
Sol License	MIT License
Top 5 Holders	<i>will be updated after the DEX listing</i>
Other	default evmVersion

TAX

BUY	9%	Address	SELL	9%
Liquidity Fee	2%	Will be automatic add liquidity	Liquidity Fee	2%
Buyback Fee	1%	-	Buyback Fee	1%
Reflection Fee	0%	-	Reflection Fee	0%
Dev Fee	2%	0x5311c06B4cde0e823d9821DE1fFd24485e9c3F2f	Dev Fee	2%
Marketing Fee	3%	0x53cA2a894406848fbB34444859013F068F52FBe1	Marketing Fee	3%
Team Fee	1%	0x53cA2a894406848fbB34444859013F068F52FBe1	Team Fee	1%

Token Metrics

Rank	Address	Quantity	Percentage	Analytics
1	Pinksale: PinkLock V2	858,598,232,919.548	<div><div></div></div> 85.8598%	View
2	0xb5bd94dbd1d3701a6a0e8f421e792d630a1463a9	141,401,767,080.452	<div><div></div></div> 14.1402%	View

[Download CSV Export]

Team Review

The DoKEN V2 team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 4,189 people in their telegram group (count in audit date).

Official Website And Social Media

Website: <https://doken.dev/>

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

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

MANUAL CODE REVIEW

● Minor-risk

2 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);
```

2. Marketing and dev wallet same address
Recommendation: change marketing and dev wallet (must be different), because investor can't tracking if the wallet same)

```
mktFeeReceiver =  
address(0x53cA2a894406848fbB34444859013F068F52FBe1);  
teamFeeReceiver =  
address(0x53cA2a894406848fbB34444859013F068F52FBe1);
```

● Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

● High-Risk

1 high-risk code issues found

Must be fixed, and will bring problems.

1. Owner can set Blacklist Address - **High Risk**

Recommendation: remove this function

Risk Scenario: if owner set blacklist pinksale contract, investor can't claim, and if owner set blacklist pancake, investor can't buy and sell (honeypot)

```
function addToBlackList(address[] calldata addresses) external  
onlyOwner {  
    for (uint256 i; i < addresses.length; ++i) {  
        isBlacklisted[addresses[i]] = true;  
    }  
}  
  
function removeFromBlackList(address account) external  
onlyOwner {  
    isBlacklisted[account] = false;  
}
```

2. Owner can pause transaction - **High Risk**

Recommendation: remove this function, or make a condition owner can't sell if the transaction is paused

Risk Scenario: If the owner disables trade, they can still sell the tokens.

```
function pauseTx(bool _buy, bool _sell) external onlyOwner{
    pauseBuy = _buy;
    pauseSell = _sell;
}
...
function txLimiter(address sender, address recipient, uint256
amount) internal {
    ...
    if(isBuy){
        require(!pauseBuy,"Buy tx is paused");
        ...
    }

    if(isSell){
        require(!pauseSell,"Sell tx is paused");
        ...
    }
}
```


3. Owner can set timelock for buy and sell but owner can exempt - **High Risk**

Recommendation: remove function or owner can't exempt address

Risk Scenario: If the owner pause trade, they can still sell the tokens.

```
function setBuyTimeLock(uint256 lockTime, bool disabled) external
onlyOwner {
    require(lockTime <= 600); // lock time must be less than
10 minutes
...
}
function setSellTimeLock(uint256 lockTime, bool disabled)
external onlyOwner {
    require(lockTime <= 600); // lock time must be less than
10 minutes
...
}
function setIsSellLockExempt(address holder, bool exempt)
external onlyOwner {
    isSellLockExempt[holder] = exempt;
}
function setIsBuyLockExempt(address holder, bool exempt)
external onlyOwner {
    isBuyLockExempt[holder] = exempt;
}
```

● Critical-Risk

0 critical-risk code issues found

Must be fixed, and will bring problems.

EXTRA NOTES SMART CONTRACT

1. IERC20

```
interface IERC20 {

    function totalSupply() external view returns (uint256);

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

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

    /**
     * @dev Returns the remaining number of tokens that `spender`
will be
     * allowed to spend on behalf of `owner` through
{transferFrom}. This is
     * zero by default.
     *
     * This value changes when {approve} or {transferFrom} are
called.
     */
    // K8u#EL(o)nG3a#t!e c&oP0Y
    function allowance(address owner, address spender) external
view returns (uint256);

    /**
     * @dev Sets `amount` as the allowance of `spender` over the
caller's tokens.
```

```

*
* Returns a boolean value indicating whether the operation
succeeded.
*
* IMPORTANT: Beware that changing an allowance with this
method brings the risk
* that someone may use both the old and the new allowance by
unfortunate
* transaction ordering. One possible solution to mitigate this
race
* condition is to first reduce the spender's allowance to 0
and set the
* desired value afterwards:
*
https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
*
* Emits an {Approval} event.
*/
function approve(address spender, uint256 amount) external
returns (bool);

/**
* @dev Moves `amount` tokens from `sender` to `recipient`
using the
* allowance mechanism. `amount` is then deducted from the
caller's
* allowance.
*
* Returns a boolean value indicating whether the operation
succeeded.
*
* Emits a {Transfer} event.
*/
function transferFrom(address 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);

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

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. DoKEN V2 Contract

```
contract DoKENV2 is IERC20, Ownable {
    using SafeMath for uint256;

    // CONSTANT ADDRESSES
    //address BUSD = 0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56;
    address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
    address DEAD = 0x00000000000000000000000000000000dEaD;
    address ZERO = 0x0000000000000000000000000000000000000000;

    string constant _name = "DoKEN V2";
    string constant _symbol = "DKN";
    uint8 constant _decimals = 18;

    uint256 _totalSupply = 1000000000000 * (10 ** _decimals); //
    1,000,000,000,000

    // LIMITTER
    uint256 public _maxBuyTxAmount = (_totalSupply * 10) / 1000;
    // 1%
    uint256 public _maxSellTxAmount = (_totalSupply * 10) / 1000;
    // 1%
    uint256 public _maxWalletSize = (_totalSupply * 20) / 1000; //
    2%

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

    mapping (address => bool) isFeeExempt;
    mapping (address => bool) isTxLimitExempt;
    mapping (address => bool) isDividendExempt;
    mapping (address => bool) public isBlacklisted;

    // Buy & Sell Lock timer
    bool buyLockDisabled = false;
    bool sellLockDisabled = false;
    mapping (address => uint256) private _sellLock;
    mapping (address => uint256) private _buyLock;
    mapping (address => bool) isSellLockExempt;
    mapping (address => bool) isBuyLockExempt;
    uint256 buyLockTime = 10; // 10 seconds
```

```
uint256 sellLockTime = 10; // 10 seconds

// FEES
uint256 liquidityFee = 200; // 3%
uint256 buybackFee = 100;
uint256 reflectionFee = 0; // 0%
uint256 devFee = 200; // 2%
uint256 mktFee = 300; // 3%
uint256 teamFee = 100; // 1%
uint256 totalFee = 900; // LiquidityFee + reflectionFee +
devFee
uint256 feeDenominator = 10000;
uint256 public _sellMultiplier = 1; // cant be changed

address public autoLiquidityReceiver;
address public devFeeReceiver;
address public mktFeeReceiver;
address public teamFeeReceiver;

uint256 targetLiquidity = 25;
uint256 targetLiquidityDenominator = 100;

IDEXRouter public router;
address public pair;

uint256 public launchedAt;

uint256 buybackMultiplierNumerator = 120;
uint256 buybackMultiplierDenominator = 100;
uint256 buybackMultiplierTriggeredAt;
uint256 buybackMultiplierLength = 30 minutes;

bool public autoBuybackEnabled = false;
bool pauseBuy = false;
bool pauseSell = false;

uint256 autoBuybackCap;
uint256 autoBuybackAccumulator;
uint256 autoBuybackAmount = 100;
uint256 autoBuybackBlockPeriod;
uint256 autoBuybackBlockLast;
```

```

//DividendDistributor public distributor;
//uint256 distributorGas = 600000;

bool public swapEnabled = true;
uint256 public swapThreshold = _totalSupply / 2000; // 0.05%
bool inSwap;
modifier swapping() { inSwap = true; _; inSwap = false; }

constructor () {
    router =
IDEXRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
    pair = IDEXFactory(router.factory()).createPair(WBNB,
address(this));
    devFeeReceiver =
address(0x5311c06B4cde0e823d9821DE1fFd24485e9c3F2f);
    mktFeeReceiver =
address(0x53cA2a894406848fbB34444859013F068F52FBe1);
    teamFeeReceiver =
address(0x53cA2a894406848fbB34444859013F068F52FBe1);

    _allowances[address(this)][address(router)] = uint256(-1);
    //distributor = new DividendDistributor(address(router));

    isFeeExempt[msg.sender] = true;
    isFeeExempt[devFeeReceiver] = true;
    isFeeExempt[mktFeeReceiver] = true;

    isTxLimitExempt[msg.sender] = true;
    isTxLimitExempt[devFeeReceiver] = true;
    isTxLimitExempt[mktFeeReceiver] = true;
    isTxLimitExempt[DEAD] = true;
    isTxLimitExempt[ZERO] = true;
    //isTxLimitExempt[address(distributor)] = true;
    isTxLimitExempt[address(this)] = true;

    isSellLockExempt[pair] = true;
    isSellLockExempt[address(router)] = true;
    isSellLockExempt[address(this)] = true;
    isSellLockExempt[msg.sender] = true;

    isBuyLockExempt[pair] = true;
    isBuyLockExempt[address(router)] = true;

```



```

isBuyLockExempt[address(this)] = true;
isBuyLockExempt[msg.sender] = true;

isDividendExempt[pair] = true;
isDividendExempt[address(this)] = true;
isDividendExempt[DEAD] = true;
isDividendExempt[ZERO] = true;

autoLiquidityReceiver = owner();

_balances[msg.sender] = _totalSupply;
emit Transfer(address(0), msg.sender, _totalSupply);
}

receive() external payable { }

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, uint256(-1));
}

function transfer(address recipient, uint256 amount) external
override returns (bool) {
    return _transferFrom(msg.sender, recipient, amount);
}

function transferFrom(address sender, address recipient,
uint256 amount) external override returns (bool) {
    if(_allowances[sender][msg.sender] != uint256(-1)){
        _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); }
    require(!isBlacklisted[sender] &&
!isBlacklisted[recipient], "This address is blacklisted");

    // coniditional Boolean
    bool isTxExempted = (isTxLimitExempt[sender] ||
isTxLimitExempt[recipient]);
    bool isContractTransfer = (sender==address(this) ||
recipient==address(this));
    bool isLiquidityTransfer = ((sender == pair && recipient
== address(router)) || (recipient == pair && sender ==
address(router) ));

    if(!isTxExempted && !isContractTransfer &&
!isLiquidityTransfer ){
        txLimiter(sender,recipient, amount);
    }

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

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

    uint256 amountReceived = shouldTakeFee(sender,recipient) ?
takeFee(sender, recipient, amount) : amount;
    _balances[sender] = _balances[sender].sub(amount,
"Insufficient Balance");
    _balances[recipient] =
_balances[recipient].add(amountReceived);

    //if(!isDividendExempt[sender]){ try
distributor.setShare(sender, _balances[sender]) {} catch {} }
    //if(!isDividendExempt[recipient]){ try
distributor.setShare(recipient, _balances[recipient]) {} catch {}

```

```

}

    //try distributor.process(distributorGas) {} catch {}

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

function pauseTx(bool _buy, bool _sell) external onlyOwner{
    pauseBuy = _buy;
    pauseSell = _sell;
}

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 txLimiter(address sender, address recipient, uint256
amount) internal {

    bool isBuy = sender == pair || sender == address(router);
    bool isSell = recipient== pair || recipient ==
address(router);

    if(isBuy){
        require(!pauseBuy, "Buy tx is paused");
        require(amount <= _maxBuyTxAmount, "TX Limit
Exceeded");
        // apply Buy Lock
        if(!buyLockDisabled && !isBuyLockExempt[recipient]) {
            require(_buyLock[recipient] <= block.timestamp ,
            "Pls wait before another buy" );
            _buyLock[recipient] = block.timestamp+buyLockTime;
        }
    }

    if(isSell){

```

```

        require(!pauseSell, "Sell tx is paused");
        require(amount <= _maxSellTxAmount, "TX Limit
Exceeded");
        // apply Sell Lock
        if(!sellLockDisabled && !isSellLockExempt[sender]) {
            require(_sellLock[sender] <= block.timestamp ,
"Pls wait before another sells" );
            _sellLock[sender] = block.timestamp+sellLockTime;
        }

    }

}

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

function getTotalFee(bool selling) public view returns
(uint256) {
    if(selling){ return totalFee.mul(_sellMultiplier); }
    return totalFee;
}

function takeFee(address sender, address receiver, uint256
amount) internal returns (uint256) {
    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 dynamicLiquidityFee =
isOverLiquified(targetLiquidity, targetLiquidityDenominator) ? 0 :
liquidityFee;
        uint256 amountToLiquify =
swapThreshold.mul(dynamicLiquidityFee).div(totalFee).div(2);
        uint256 amountToSwap = swapThreshold.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(dynamicLiquidityFee.div(2));

        uint256 amountBNBLiquidity =
amountBNB.mul(dynamicLiquidityFee).div(totalBNBFee).div(2);
        //uint256 amountBNBReflection =
amountBNB.mul(reflectionFee).div(totalBNBFee);
        uint256 amountBNBDev =
amountBNB.mul(devFee).div(totalBNBFee);
        uint256 amountBNBMkt =
amountBNB.mul(mktFee).div(totalBNBFee);
        uint256 amountBNBTeam =
amountBNB.mul(teamFee).div(totalBNBFee);

        //try distributor.deposit{value: amountBNBReflection}() {}

```

```

catch {}

    sendPayable(amountBNBDev, amountBNBMkt, amountBNBTeam);

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

function sendPayable(uint256 amtDev, uint256 amtMkt, uint256
amtTeam) internal {
    (bool successone,) = payable(devFeeReceiver).call{value:
amtDev, gas: 30000}("");
    (bool successtwo,) = payable(mktFeeReceiver).call{value:
amtMkt, gas: 30000}("");
    (bool successthree,) =
payable(teamFeeReceiver).call{value: amtTeam, gas: 30000}("");
    require(successone && successtwo && successthree,
"receiver rejected ETH transfer");
}
// used for collecting collected Tax to DevFee Address
function withdrawCollectedTax() external onlyOwner {
    uint256 bal = address(this).balance; // return the native
token ( BNB )
    (bool success,) = payable(devFeeReceiver).call{value: bal,
gas: 30000}("");
    require(success, "receiver rejected ETH transfer");
}

function shouldAutoBuyback() internal view returns (bool) {
    return msg.sender != pair
        && !inSwap
        && autoBuybackEnabled
        && autoBuybackBlockLast + autoBuybackBlockPeriod <=
block.number

```

```

        && address(this).balance >= autoBuybackAmount;
    }

    function triggerManualBuyback(uint256 amount, bool
triggerBuybackMultiplier) external onlyOwner {
        buyTokens(amount, devFeeReceiver);
        if(triggerBuybackMultiplier){
            buybackMultiplierTriggeredAt = block.timestamp;
            emit BuybackMultiplierActive(buybackMultiplierLength);
        }
    }

    function clearBuybackMultiplier() external onlyOwner {
        buybackMultiplierTriggeredAt = 0;
    }

    function triggerAutoBuyback() internal {
        buyTokens(autoBuybackAmount, devFeeReceiver);
        autoBuybackBlockLast = block.number;
        autoBuybackAccumulator =
autoBuybackAccumulator.add(autoBuybackAmount);
        if(autoBuybackAccumulator > autoBuybackCap){
autoBuybackEnabled = false; }
    }

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

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

    function setAutoBuybackSettings(bool _enabled, uint256 _cap,

```

```

uint256 _amount, uint256 _period) external onlyOwner {
    autoBuybackEnabled = _enabled;
    autoBuybackCap = _cap;
    autoBuybackAccumulator = 0;
    autoBuybackAmount = _amount.div(100);
    autoBuybackBlockPeriod = _period;
    autoBuybackBlockLast = block.number;
}

function setBuybackMultiplierSettings(uint256 numerator,
uint256 denominator, uint256 length) external onlyOwner {
    require(numerator / denominator <= 2 && numerator >
denominator);
    buybackMultiplierNumerator = numerator;
    buybackMultiplierDenominator = denominator;
    buybackMultiplierLength = length;
}

/**
 *
 * SETTER SECTIONS
 *
 */

function setMaxWallet(uint256 numerator, uint256 divisor)
external onlyOwner{
    require(numerator > 0 && divisor > 0 && divisor <= 10000);
    _maxWalletSize = _totalSupply.mul(numerator).div(divisor);
}

function setDevFee(uint256 fee) external onlyOwner {
    // total fee should not be more than 20%;
    uint256 simulatedFee =
fee.add(liquidityFee).add(buybackFee).add(teamFee).add(mktFee);
    require(simulatedFee <= 2000, "Fees too high !!");
    devFee = fee;
    totalFee = simulatedFee;
}

function setBuybackFee(uint256 fee) external onlyOwner {
    // total fee should not be more than 20%;
    uint256 simulatedFee =

```



```

fee.add(liquidityFee).add(devFee).add(teamFee).add(mktFee);
    require(simulatedFee <= 2000, "Fees too high !!");
    buybackFee = fee;
    totalFee = simulatedFee;
}
function setLpFee(uint256 fee) external onlyOwner {
    // total fee should not be more than 20%;
    uint256 simulatedFee =
fee.add(devFee).add(buybackFee).add(teamFee).add(mktFee);
    require(simulatedFee <= 2000, "Fees too high !!");
    liquidityFee = fee;
    totalFee = simulatedFee;
}
function setTeamFee(uint256 fee) external onlyOwner {
    // total fee should not be more than 20%;
    uint256 simulatedFee =
fee.add(devFee).add(buybackFee).add(liquidityFee).add(mktFee);
    require(simulatedFee < 2000, "Fees too high !!");
    teamFee = fee;
    totalFee = simulatedFee;
}
function setMarketingFee(uint256 fee) external onlyOwner {
    // total fee should not be more than 20%;
    uint256 simulatedFee =
fee.add(devFee).add(buybackFee).add(liquidityFee).add(teamFee);
    require(simulatedFee < 2000, "Fees too high !!");
    mktFee = fee;
    totalFee = simulatedFee;
}

function setBuyTxMaximum(uint256 max) external onlyOwner{
    uint256 minimumTreshold = (_totalSupply * 7) / 1000; //
0.7% is the minimum tx limit, we cant set below this
    uint256 simulatedMaxTx = (_totalSupply * max) / 1000;
    require(simulatedMaxTx >= minimumTreshold, "Tx Limit is
too low");
    _maxBuyTxAmount = simulatedMaxTx;
}

function setSellTxMaximum(uint256 max) external onlyOwner {
    uint256 minimumTreshold = (_totalSupply * 7) / 1000; //
0.7% is the minimum tx limit, we cant set below this

```

```

        uint256 simulatedMaxTx = (_totalSupply * max) / 1000;
        require(simulatedMaxTx >= minimumTreshold, "Tx Limit is
too low");
        _maxSellTxAmount = simulatedMaxTx;
    }

    function setBuyTimeLock(uint256 lockTime, bool disabled)
external onlyOwner {
        require(lockTime <= 600); // lock time must be less than
10 minutes
        buyLockTime = lockTime;
        buyLockDisabled = disabled;
    }

    function setSellTimeLock(uint256 lockTime, bool disabled)
external onlyOwner {
        require(lockTime <= 600); // lock time must be less than
10 minutes
        sellLockTime = lockTime;
        sellLockDisabled = disabled;
    }

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

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

    // function setIsDividendExempt(address holder, bool exempt)
external onlyOwner {
        // isDividendExempt[holder] = exempt;
        // if(exempt){
        // distributor.setShare(holder, 0);
        // }else{
        // distributor.setShare(holder, _balances[holder]);
        // }
        // }

```

```

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

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

    function setFeeReceivers(address _autoLiquidityReceiver,
address _devFeeReceiver, address _mktFeeReceiver, address
_teamFeeReceiver) external onlyOwner {
        autoLiquidityReceiver = _autoLiquidityReceiver;
        devFeeReceiver = _devFeeReceiver;
        mktFeeReceiver = _mktFeeReceiver;
        teamFeeReceiver = _teamFeeReceiver;
    }

    function setSwapBackSettings(bool _enabled, uint256 _amount)
external onlyOwner {
        swapEnabled = _enabled;
        swapThreshold = _amount.div(100);
    }

    function setTargetLiquidity(uint256 _target, uint256
_denominator) external onlyOwner {
        targetLiquidity = _target;
        targetLiquidityDenominator = _denominator;
    }

    function addToBlackList(address[] calldata addresses) external
onlyOwner {
        for (uint256 i; i < addresses.length; ++i) {
            isBlacklisted[addresses[i]] = true;
        }
    }

    function removeFromBlackList(address account) external
onlyOwner {
        isBlacklisted[account] = false;
    }

```

```

    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;
    }

    // function getDividendStats(address holder) public view
returns (uint256 [3] memory) {
    //     return distributor.getDividendStats(holder);
    // }

    event AutoLiquify(uint256 amountBNB, uint256 amountBOG);
    event BuybackMultiplierActive(uint256 duration);

    function totalSupply() external view override returns
    (uint256) { return _totalSupply; }
    function decimals() external pure returns (uint8) { return
    _decimals; }
    function symbol() external pure returns (string memory) {
    return _symbol; }
    function name() external pure returns (string memory) { return
    _name; }
    function getOwner() external view 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]; }

}

```

4. Tax Fee Function (Cannot set over 20%)

```
function setDevFee(uint256 fee) external onlyOwner {  
    // total fee should not be more than 20%;  
    uint256 simulatedFee =  
fee.add(liquidityFee).add(buybackFee).add(teamFee).add(mktFee);  
    require(simulatedFee <= 2000, "Fees too high !!");  
    devFee = fee;  
    totalFee = simulatedFee;  
}  
function setBuybackFee(uint256 fee) external onlyOwner {  
    // total fee should not be more than 20%;  
    uint256 simulatedFee =  
fee.add(liquidityFee).add(devFee).add(teamFee).add(mktFee);  
    require(simulatedFee <= 2000, "Fees too high !!");  
    buybackFee = fee;  
    totalFee = simulatedFee;  
}  
function setLpFee(uint256 fee) external onlyOwner {  
    // total fee should not be more than 20%;  
    uint256 simulatedFee =  
fee.add(devFee).add(buybackFee).add(teamFee).add(mktFee);  
    require(simulatedFee <= 2000, "Fees too high !!");  
    liquidityFee = fee;  
    totalFee = simulatedFee;  
}  
function setTeamFee(uint256 fee) external onlyOwner {  
    // total fee should not be more than 20%;  
    uint256 simulatedFee =  
fee.add(devFee).add(buybackFee).add(liquidityFee).add(mktFee);  
    require(simulatedFee < 2000, "Fees too high !!");  
    teamFee = fee;  
    totalFee = simulatedFee;  
}  
function setMarketingFee(uint256 fee) external onlyOwner {  
    // total fee should not be more than 20%;  
    uint256 simulatedFee =  
fee.add(devFee).add(buybackFee).add(liquidityFee).add(teamFee);  
    require(simulatedFee < 2000, "Fees too high !!");  
    mktFee = fee;  
    totalFee = simulatedFee;  
}
```

5. Pause Transfer (Owner can pause trade) - High Risk

```
function pauseTx(bool _buy, bool _sell) external onlyOwner{
    pauseBuy = _buy;
    pauseSell = _sell;
}
```

6. Owner can set timelock for buy and sell but owner can exempt - High Risk

```
function setBuyTimeLock(uint256 lockTime, bool disabled) external
onlyOwner {
    require(lockTime <= 600); // lock time must be less than
10 minutes
    buyLockTime = lockTime;
    buyLockDisabled = disabled;
}

function setSellTimeLock(uint256 lockTime, bool disabled)
external onlyOwner {
    require(lockTime <= 600); // lock time must be less than
10 minutes
    sellLockTime = lockTime;
    sellLockDisabled = disabled;
}

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

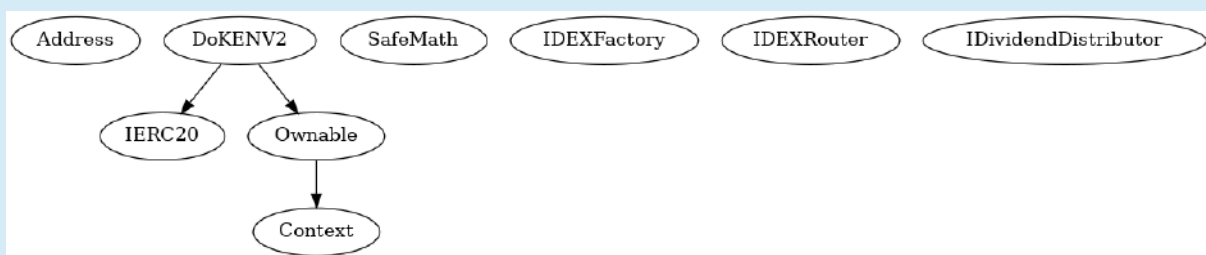
function setIsBuyLockExempt(address holder, bool exempt)
external onlyOwner {
    isBuyLockExempt[holder] = exempt;
}
```

7. Owner can set max tx amount for buy and sell, but can't set below 0.7% from total supply

```
function setBuyTxMaximum(uint256 max) external onlyOwner{
    uint256 minimumTreshold = (_totalSupply * 7) / 1000; //
    0.7% is the minimum tx limit, we cant set below this
    uint256 simulatedMaxTx = (_totalSupply * max) / 1000;
    require(simulatedMaxTx >= minimumTreshold, "Tx Limit is
too low");
    _maxBuyTxAmount = simulatedMaxTx;
}

function setSellTxMaximum(uint256 max) external onlyOwner {
    uint256 minimumTreshold = (_totalSupply * 7) / 1000; //
    0.7% is the minimum tx limit, we cant set below this
    uint256 simulatedMaxTx = (_totalSupply * max) / 1000;
    require(simulatedMaxTx >= minimumTreshold, "Tx Limit is
too low");
    _maxSellTxAmount = simulatedMaxTx;
}
```

CONTRACT INHERITANCE



READ CONTRACT (ONLY NEED TO KNOW)

1. _maxBuyTxAmount

```
100000000000000000000000000000000 uint256
```

(Function for read max buy tx amount)

2. _maxSellTxAmount

100000000000000000000000000000000 uint256

(Function for read max sell tx amount)

3. maxWalletSize

`100000000000000000000000000000000 uint256`

(Function for read max hold amount wallet)

4. devFeeReceiver

0x5311c06b4cde0e823d9821de1ffd24485e9c3f2f address

(Function for read dev fee receiver)

5. getOwner

0x56119818404e5cc64a526e78b67f3dc496be743d address

(Function for read owner address)

6. mktFeeReceiver

0x53ca2a894406848fbb34444859013f068f52fbe1 address

(Function for read marketing fee receiver)

7. name

DoKEN V2 string

(Function for read Token name)

WRITE CONTRACT

1. addToBlacklist - **High Risk**

addresses (address[])

(Function for blacklist address - onlyOwner)

2. pauseTx - **High Risk**

_buy (bool)

_sell (bool)

(Function for pause trade, buy or sell, fill with true or false)

3. setBuyTxMaximum

max (uint256)

(Function for set buy tx amount - can't set below 0.7% from total supply)

4. setSellTxMaximum

max (uint256)

(Function for set sell tx amount - can't set below 0.7% from total supply)

5. renounceOwnership

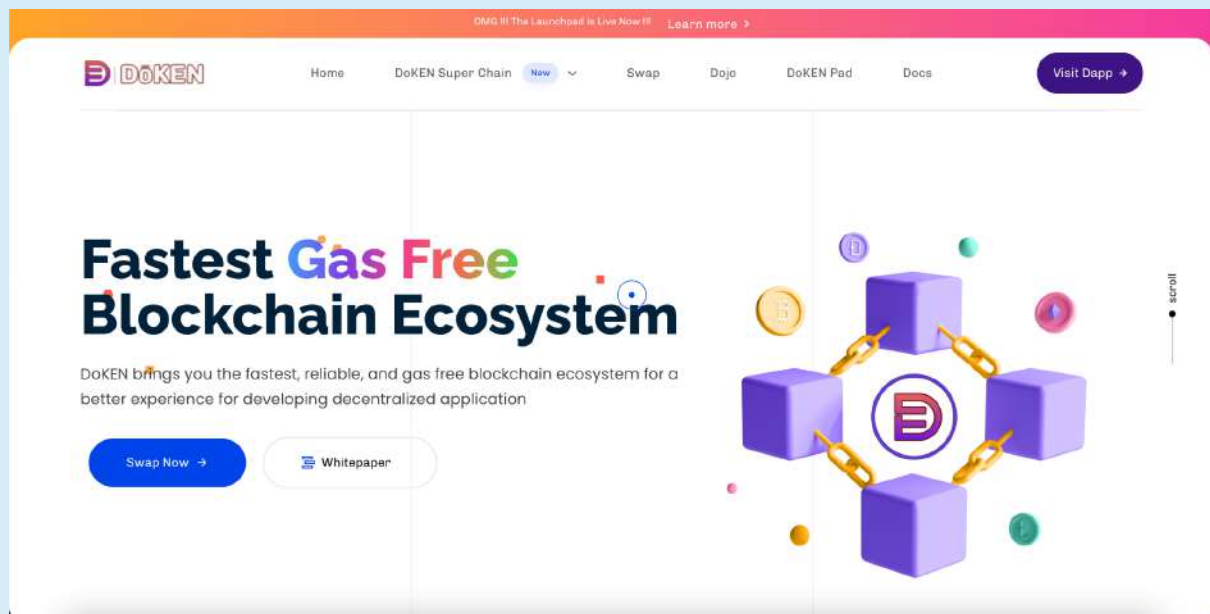
(Renouncing ownership will leave the contract without an owner, thereby removing any functionality that is only available to the owner)

6. transferOwnership

newOwner (address)

(Its function is to change the owner)

WEBSITE REVIEW



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

Web-Tech stack: Wordpress, Bootstrap, Elementor


Domain .com (Hostgator) - Tracked by whois

First Contentful Paint:	1.1s
Fully Loaded Time	3.3s
Performance	80%
Accessibility	87%
Best Practices	83%
SEO	91%

BlockSAFU Token Scanner



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

lockSAFU | Don't give a chance for scammers!

BlockSAFU

Products ▾ Knowledge ▾ Company ▾ Token Earn

Request Service

BlockSAFU is Official Audit Partner Of PinkSale

BlockSAFU Token Scanner

0x0420EB45AC5a4f04763f679c07c3A63774fE0289

Scan

There is no liquidity available for this contract.

BlockSAFU Token Scanner Score:

65

Score

Token Information

Indicator	Value
Token Name	DOKEN V2
Token Symbol	DKN
Total Supply	1,000,000,000,000
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	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	2 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	0x56119818...43d 🔗
Creator Balance	1,000,000,000,000 DKN
Creator Percent	100%
Owner Address	0x56119818...43d 🔗
Owner Balance	0 DKN
Owner Percent	0%
Lp Holder Count	0
Lp Total Supply	NaN
Mint	✔ Not detected

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 is not KYC by Blocksafu

HONEYPOT REVIEW

- Ability to sell.
- The owner is able to pause the contract.
- The owner can set lock exempt
- The owner can't set fees over 20%

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.