



BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT



Project: DEEP SKYLINE

Website: wstecserver.com

Contract Address:

0x8938071Dd7A4BaB850AA160590871629B00379e5

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:

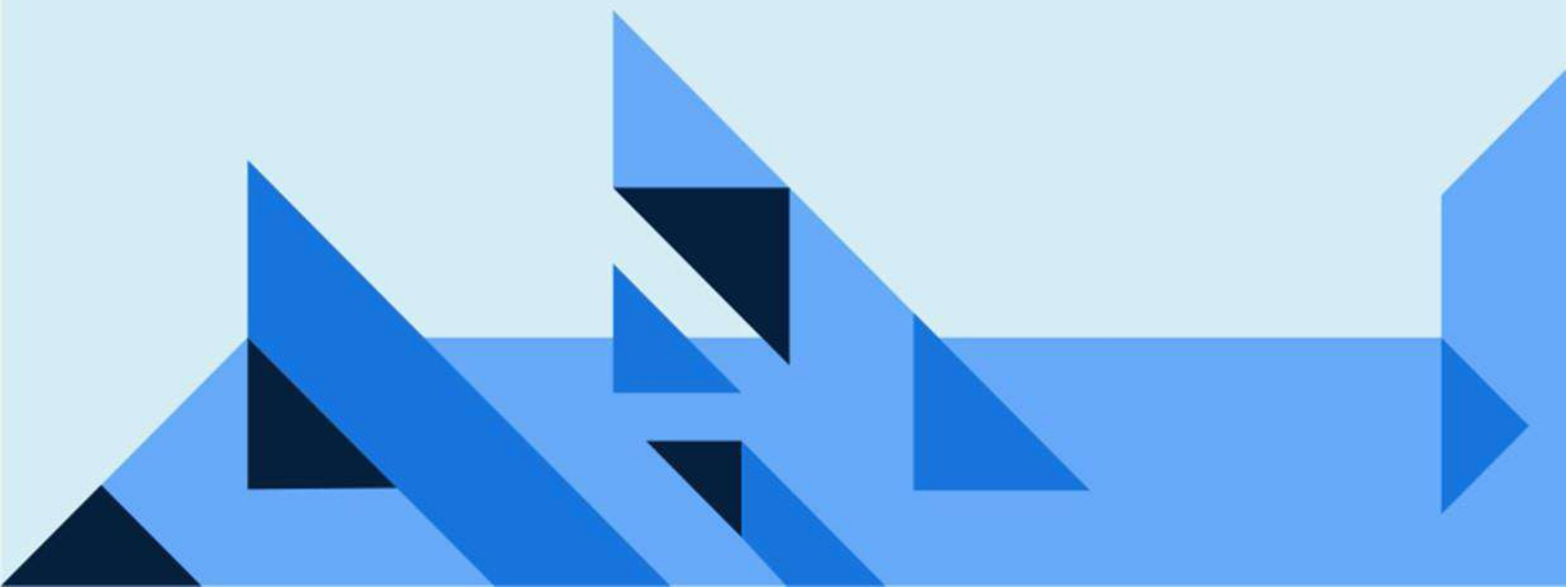
BlockSAFU (BSAFU) is an Anti-Scam Token Utility that reviews Smart Contracts and Token information to Identify Rug Pull and Honey Pot scamming activity. BlockSAFU's Development Team consists of several Smart Contract creators, Auditors Developers, and Blockchain experts. BlockSAFU provides solutions, 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.



BlockSAFU was commissioned by DES 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	DES Token
Token Symbol	DES
Token Decimal	18
Total Supply	1,000,000,000 DES
Contract Address	0x8938071Dd7A4BaB850AA160590871629B00379e5
Deployer Address	0xAc45ECF9F362Eee0ee7f6D1C39Bcc51F796abB09
Owner Address	0xAc45ECF9F362Eee0ee7f6D1C39Bcc51F796abB09
Tax Fees Buy	9%
Tax Fees Sell	9%
Gas Used for Buy	224,743
Gas Used for Sell	143,545
Contract Created	Aug-01-2022 03:26:16 PM +UTC
Initial Liquidity	398.06 USDT
Liquidity Status	Unlocked
Unlocked Date	<i>Need lock</i>
Verified CA	Yes
Compiler	v0.8.7+commit.e28d00a7
Optimization	Yes with 200 runs
Sol License	MIT License
Top Holders	2.09%
Other	default evmVersion

TAX

BUY	9%	SELL	9%
Burn Fee	5%	Burn Fee	5%
Fund Fee	0%	Fund Fee	0%
Reward Fee	1%	Reward Fee	1%
NFT Fee	1%	NFT Fee	1%
Other Fee	2%	Other Fee	2%

OVERVIEW

Mint Function

- Potentially semantic imprecision amount of tokonomic. Solidity can't take decimal in amount, if the number is decimal in division result, the solidity will read as an imprecision number, Token Scanner also read this algorithm as a minting event. The recommendation is doing multiply first before division and using Tax Denominator for avoiding imprecision decimal amount.

Fees

- Buy 9% (owner can set fees up to 100%).
- Sell 9% (owner can set fees up to 100%).

Tx Amount

- Owner cannot set max tx amount.

Transfer Pausable

- Owner cannot pause.

Blacklist

- Owner cannot blacklist.

Ownership

- Owner cannot take back ownership.

Proxy

- This contract has no proxy.

Anti-Whale

- Owner cannot limit the number of wallet holdings.

Trading Cooldown

- Owner cannot set the selling time interval.

Token Metrics

Rank	Address	Quantity	Percentage	Analytics
1	0xc8d9db497673412a7e5fde7c96bb16ea55468ac3	972,223,641.294075258254198081	97.2224%	Analytics
2	0x893228c15683714638b26804245b6514083a163a	20,286,305.262741197093290715	2.0298%	Analytics
3	PancakeSwap V2: BSC-USD-DES 5	3,980,173.971213215833662098	0.3980%	Analytics
4	0x85b8cf1310b84854067979a85860571a29fa58d5	457,012.630657724864187555	0.0457%	Analytics
5	Null Address: 0x000...dEaD	128,942.864256792282916862	0.0129%	Analytics
6	0x78e24e002c7a59c875dfa0514118bd16ab017b62	120,432.938583723710447583	0.0120%	Analytics
7	0x2491b0dfb0a01bec429ee86f6590dcb5b85c3635	100,224.916441443010647969	0.0100%	Analytics
8	0x20f504d3f4152dfd29e8e8f8fa2666ceb3225625	100,000	0.0100%	Analytics
9	0x534201a511b51ace9cf576973e931cc9616018b0	86,215.785040770232180909	0.0086%	Analytics
10	0x57c2e97e47459f9680ec1ccf529b27b35631c2e8	70,061.268888248041749883	0.0070%	Analytics

Team Review

The DES team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 16,864 people in their telegram group (count in audit date).

Official Website and Social Media

Website: <https://www.wstecserver.com/>

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

Twitter: https://twitter.com/des_token

MANUAL CODE REVIEW

● Minor-risk

3 minor-risk code issues 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. Lack of Zero Check, the funding address not checkin if the address is zero, consequence the tokenomic can't work like normally.

```
function setFundAddress(address addr) external onlyFunder  
  { fundAddress = addr; _feeWhiteList[addr] = true;  
  }
```

3. Divide before multiply, solidity can't have decimal, if the divition amount is greater than amount, the result is zero, this schema can have unprecision amount
_takeTransfer(sender,recipient,(tAmount / 100) * 1,currentRate)

● Medium-risk

1 medium-risk code issue found

Should be fixed, could bring problems.

Potentially semantic imprecision amount of tokenomic. Solidity can't take decimal in amount, if the number is decimal in division result, solidity will read as imprecision number, Token Scanner also read this algorithm as a minting event. Recommendation is doing multiply first before division and using Tax Denominator for avoid imprecision decimal amount.

```
function _tokenTransfer(
    address sender, address
    recipient, uint256
    tAmount, bool takeFee
) private {
    if (_tOwned[sender] > tAmount) {
        _tOwned[sender] -= tAmount;
    } else {
        _tOwned[sender] = 0; }
    uint256 currentRate = _getRate(); uint256 rAmount =
    tAmount * currentRate; _rOwned[sender] =
    _rOwned[sender] - rAmount;
    uint256 rate; if
    (takeFee) {
        _shareTransfer( sender, fundAddress,
            (tAmount / 100) * otherFee,
            currentRate
        );
        _nftTransfer( sender, fundAddress,
            (tAmount / 100) * nftFee,
            currentRate
        );
    }
    _takeTransfer(

        sender, burnAddress, (tAmount /
        100) * burnFee, currentRate
    );
    _reflectFee(
        (rAmount / 100) * dividendFee,
        (tAmount / 100) * dividendFee );
    rate = fundFee + dividendFee + otherFee + burnFee +
    nftFee;
}
uint256 recipientRate = 100 - rate;
_takeTransfer( sender, recipient, (tAmount /
100) * recipientRate, currentRate
```

```
    );  
}
```

● High-Risk

1 high-risk code issue found

Must be fixed, and will bring problem.

Owner can change tax fee up to 100%

```
function setDividendFee(uint256 fee) external onlyOwner {  
    dividendFee = fee;  
}  
function setFundFee(uint256 fee) external onlyOwner { fundFee =  
    fee;  
}  
function setOtherFee(uint256 fee) external onlyOwner {  
    otherFee = fee;  
}  
function setNftFee(uint256 fee) external onlyOwner {  
    nftFee = fee;  
}
```

● 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. Ownable Contract

```
abstract contract Ownable { address internal
    _owner;
    event OwnershipTransferred( address indexed
        previousOwner, address indexed
        newOwner
    );
    constructor() { address msgSender =
        msg.sender;
        _owner = msgSender; emit OwnershipTransferred(address(0),
            msgSender);
    }
    function owner() public view returns (address) { return _owner;
    }
    modifier onlyOwner() { require(_owner == msg.sender,
        "!owner"); _; }
    function renounceOwnership() public virtual onlyOwner { emit
        OwnershipTransferred(_owner, address(0)); _owner = address(0); }
    function transferOwnership(address newOwner) public virtual
    onlyOwner { require(newOwner != address(0), "new is 0"); emit
        OwnershipTransferred(_owner, newOwner); _owner =
        newOwner;
    }
}
```

Standard Ownable contract

3. DES Contract

```

abstract contract AbsToken is IERC20, Ownable { mapping(address =>
    mapping(address => uint256)) private
_allowances;
    address public fundAddress; address public
    burnAddress =
0x0000000000000000000000000000000000000000000000000000000000000000dEaD;
    string private _name; string private
    _symbol; uint8 private _decimals;
    uint256 public dividendFee; uint256 public
    fundFee;
    uint256 public otherFee; uint256 public
    nftFee; uint256 public burnFee;
    uint256 public startTradeBlock; mapping(address => bool) private
    _feeWhiteList; mapping(address => bool) private _excludeRewardList;
    mapping(address => bool) public whitelist;
    mapping(address => uint256) private _rOwned;
    mapping(address => uint256) private _tOwned; uint256
    private constant MAX = ~uint256(0); uint256 private _tTotal;
    uint256 private _rTotal; uint256 private _tFeeTotal;
    INFT public nft; mapping(address => bool) private _swapPairList;
    address[] private _whitelistedUsers;
event WhitelistAdded(address indexed account); event WhitelistRemoved(address indexed
account);

    constructor( string memory Name, string
        memory Symbol, uint8 Decimals,
        uint256 Supply, uint256
        DividendFee, uint256 FundFee,
        uint256 OtherFee, uint256 NftFee,
        uint256 BurnFee, address
        FundAddress, address
        ReceivedAddress, address
        NFTAddress
    ){
        _name = Name;
        _symbol = Symbol; _decimals =
        Decimals; nft = INFT(NFTAddress);
        dividendFee = DividendFee; fundFee =
        FundFee; otherFee = OtherFee;
        nftFee = NftFee; burnFee = BurnFee;
        ISwapRouter swapRouter = ISwapRouter(
            0x10ED43C718714eb63d5aA57B78B54704E256024E
        ); _allowances[address(this)][address(swapRouter)] = MAX;
        address mainPair =
        ISwapFactory(swapRouter.factory()).createPair( address(this), // swapRouter.WETH()
            address(0x55d398326f99059fF775485246999027B3197955)
        )
    }
}

```

```

    );
    _swapPairList[mainPair] = true; _excludeRewardList[mainPair] = true;
    uint256 tTotal = Supply * 10**_decimals;

    int256 rTotal = (MAX - (MAX % tTotal)); _rOwned[ReceivedAddress] =
    rTotal; _tOwned[ReceivedAddress] = tTotal; emit Transfer(address(0),
    ReceivedAddress, tTotal); _rTotal = rTotal; _tTotal = tTotal;
    fundAddress = FundAddress;
    _feeWhiteList[FundAddress] = true;
    _feeWhiteList[ReceivedAddress] = true;
    _feeWhiteList[address(this)] = true;
    _feeWhiteList[msg.sender] = true;
    _feeWhiteList[address(swapRouter)] = true;
}
function symbol() external view override returns (string
memory) { return _symbol;
}
function name() external view override returns (string memory)
{ return _name;
}
function decimals() external view override returns (uint8) { return _decimals;
}
function totalSupply() external view override returns
(uint256) { return _tTotal;
}
function balanceOf(address account) public view override
returns (uint256) {
if (_excludeRewardList[account]) {

    return _tOwned[account];
} return tokenFromReflection(_rOwned[account]);
}
function transfer(address recipient, uint256 amount) public override returns
(bool)
{ _transfer(msg.sender, 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(msg.sender, spender, amount); return true;
}
function transferFrom( address sender,
    address recipient, uint256 amount

```

```

) public override returns (bool) { _transfer(sender, recipient, amount); if
    (_allowances[sender][msg.sender] != MAX) {
    _allowances[sender][msg.sender] = _allowances[sender][msg.sender] amount;

    }
    return true;
}
function totalFees() public view returns (uint256) { return _tFeeTotal;
}
function tokenFromReflection(uint256 rAmount) public view
returns (uint256)
{ uint256 currentRate = _getRate(); return rAmount /
    currentRate;
} function _getRate() private view returns (uint256) {
    if (_rTotal < _tTotal) { return 1;
    } return _rTotal / _tTotal;
}
function _approve( address owner,
    address spender, uint256 amount
) private {
    _allowances[owner][spender] = amount; emit Approval(owner,
    spender, amount);
}
function _transfer( address from,
    address to, uint256 amount
) private { bool takeFee = false;

    if (_swapPairList[from] || _swapPairList[to]) { if (!_feeWhiteList[from] &&
        !_feeWhiteList[to]) {
        takeFee = true;
    }
    }
    _tokenTransfer(from, to, amount, takeFee); }
function _tokenTransfer( address
    sender, address recipient, uint256
    tAmount, bool takeFee
) private {
    if (_tOwned[sender] > tAmount) { _tOwned[sender] -= tAmount;
    } else {
        _tOwned[sender] = 0;
    }
    uint256 currentRate = _getRate(); uint256 rAmount = tAmount *
    currentRate; _rOwned[sender] = _rOwned[sender] - rAmount;
    uint256 rate; if (takeFee) {
        _shareTransfer( sender, fundAddress,
            (tAmount / 100) * otherFee, currentRate
    );
};

```



```

        _nftTransfer( sender,
            fundAddress,
            (tAmount / 100) * nftFee, currentRate
        );
        _takeTransfer( sender,
            burnAddress,
            (tAmount / 100) * burnFee, currentRate
        );
        _reflectFee(
            (rAmount / 100) * dividendFee,
            (tAmount / 100) * dividendFee
        );
        rate = fundFee + dividendFee + otherFee + burnFee +
nftFee;
    }
    uint256 recipientRate = 100 - rate;
    _takeTransfer( sender,
        recipient,
        (tAmount / 100) * recipientRate, currentRate
    ); }
    function _funTransfer( address
        sender, address recipient,
        uint256 tAmount
    ) private { if (_tOwned[sender] > tAmount) {

        _tOwned[sender] -= tAmount;
    } else {
        _tOwned[sender] = 0;
    }
    uint256 currentRate = _getRate(); uint256 rAmount =
    tAmount * currentRate; _rOwned[sender] = _rOwned[sender]
    - rAmount;
    _takeTransfer(sender, fundAddress, (tAmount / 100) * 99,
currentRate);
    _takeTransfer(sender, recipient, (tAmount / 100) * 1,
currentRate);
    }
    function _burnTransfer( address
        sender, address to, uint256
        tAmount, uint256
        currentRate
    ) private {
        _tOwned[to] += tAmount;
        emit Transfer(sender, to, tAmount);
    }
}

```

```

function _takeTransfer( address
    sender, address to, uint256
    tAmount, uint256
    currentRate
) private {
    _tOwned[to] += tAmount;
    uint256 rAmount = tAmount * currentRate; _rOwned[to]
    = _rOwned[to] + rAmount; emit Transfer(sender, to,
    tAmount);
} function
_nftTransfer(address
    sender, address to, uint256
    tAmount, uint256
    currentRate
) private {
    uint256 total = nft.totalSupply(); if (total == 0) {
        return;
    } uint256 sendAmount = tAmount / total; uint256 rAmount =
    sendAmount * currentRate;
    for (uint256 index = 1; index < total + 1; index++) {
        _tOwned[nft.ownerOf(index)] += tAmount;
        _rOwned[nft.ownerOf(index)] =
    _rOwned[nft.ownerOf(index)] + rAmount;
    } }
function _shareTransfer( address
    sender, address to, uint256
    tAmount, uint256
    currentRate
) private { address[] memory __whitelistedUsers = new address[](
    _whitelistedUsers.length ); uint256 amount = _whitelistedUsers.length;
    for (uint256 index = 0; index < _whitelistedUsers.length;
index++) { if (!whitelist[_whitelistedUsers[index]]) { amount = amount - 1;
    } } uint256 senAmount = tAmount / amount; uint256 rAmount =
    senAmount * currentRate; for (uint256 i = 0; i < _whitelistedUsers.length; i++)
    { if (whitelist[_whitelistedUsers[i]]) {
        _tOwned[_whitelistedUsers[i]] =
    _tOwned[_whitelistedUsers[i]] +

```

```

        senAmount;
        _rOwned[_whitelistedUsers[i]] =
            _rOwned[_whitelistedUsers[i]] + rAmount;
    }
}
function _reflectFee(uint256 rFee, uint256 tFee) private {
    _rTotal = _rTotal - rFee;
    _tFeeTotal = _tFeeTotal + tFee; }
receive() external payable {}
function claimBalance() external {
    payable(fundAddress).transfer(address(this).balance);
}
function claimToken(address token, uint256 amount) external {
    IERC20(token).transfer(fundAddress, amount); }
function setFundAddress(address addr) external onlyFunder { fundAddress =
    addr;
    _feeWhiteList[addr] = true;
}
function setFeeWhiteList(address addr, bool enable) external
onlyFunder {
    _feeWhiteList[addr] = enable; }
function changeWhiteList(address addr, bool enable) external
onlyOwner { whitelist[addr] = enable;
}

function setSwapPairList(address addr, bool enable) external
onlyFunder {
    _swapPairList[addr] = enable; if (enable)
    {
        _excludeRewardList[addr] = true; } }
function setExcludeReward(address addr, bool enable) external
onlyFunder {
    _tOwned[addr] = balanceOf(addr);
    _rOwned[addr] = _tOwned[addr] * _getRate();
    _excludeRewardList[addr] = enable; }
function setDividendFee(uint256 fee) external onlyOwner { dividendFee = fee;
}
function setFundFee(uint256 fee) external onlyOwner { fundFee = fee;
}
function setOtherFee(uint256 fee) external onlyOwner { otherFee = fee;
}
function setNftFee(uint256 fee) external onlyOwner { nftFee = fee;
}
function startTrade() external onlyOwner { require(0 == startTradeBlock,
"trading");

```

```

        startTradeBlock = block.number;
    }
    function closeTrade() external onlyOwner { startTradeBlock = 0;
    }
    modifier onlyFunder() { require(_owner == msg.sender || fundAddress ==
msg.sender,
"!Funder");
    _; }
    function addWhitelist(address[] memory accounts) external
onlyOwner { for (uint256 i = 0; i < accounts.length; i++) {
        require(accounts[i] != address(0), "IDOSale:
ZERO_ADDRESS"); if (!whitelist[accounts[i]]) { whitelist[accounts[i]] =
        true; _whitelistedUsers.push(accounts[i]); emit
        WhitelistAdded(accounts[i]);
        }
    }
    }
    function removeWhitelist(address[] memory accounts) external
onlyOwner { for (uint256 i = 0; i < accounts.length; i++) {
        require(accounts[i] != address(0), "IDOSale:
ZERO_ADDRESS"); if (whitelist[accounts[i]]) { whitelist[accounts[i]] =
        false; emit WhitelistRemoved(accounts[i]);
        }
    }
    }
    function whitelistedUsers() public view returns (address[]
memory) {
        address[] memory __whitelistedUsers = new address[](
        _whitelistedUsers.length ); for (uint256 i = 0; i <
        _whitelistedUsers.length; i++) { if (!whitelist[_whitelistedUsers[i]]) {
        continue;
        }
        __whitelistedUsers[i] = _whitelistedUsers[i]; } return
        __whitelistedUsers; }
    }

```

4. Tax Fee contract

```
function setDividendFee(uint256 fee) external onlyOwner { dividendFee =  
    fee;  
}  
function setFundFee(uint256 fee) external onlyOwner { fundFee = fee;  
}  
function setOtherFee(uint256 fee) external onlyOwner { otherFee = fee;  
}  
function setNftFee(uint256 fee) external onlyOwner { nftFee = fee;  
}
```

The owner can set fees up to 100% - this can cause a honeypot

READ CONTRACT (ONLY NEED TO KNOW)

1. Version

1 uint256

(Shows Contract Versions)

2. lockToken

0x1fea9245376f256228cbea767a2e0fd48fbd0fec address

(Shows the smart contract token which is locked)

3. _marketingWalletAddress

0x262c5d453ca7c1420066c3fb668dde8c550a03c5 address

(Shows marketing wallet address)

4. enableAntiBotTrue bool

(Function for read anti bot active or not)

5. liquidityFee

1 uint256

(Function for read liquidity fee)

6. marketingFee

1 uint256

(Function for read marketing fee)

7. name

CheemsGrow string

(Function for read Token name)

WRITE CONTRACT

1. setEnableAntiBot

_enable (bool)

(The form is filled with the true or false for active or deactivate anti bot)

2. renounceOwnership

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

3. transferOwnershipnewOwner (address)

(Its function is to change the owner)

4. setLiquiditFee (cannot set over 25%) value (uint 256)

(The form is filled with new fee, for change liquidity fee)

5. setMarketingFee (cannot set over 25%) value (uint 256)


(The form is filled with new fee, for change marketing fee)



6. setTokenRewardsFeevalue (uint 256)

(The form is filled with new fee, for change Token Rewards fee)

BlockSAFU TOKEN SCANNER

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

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

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

BlockSAFU Token Scanner

0x8938071Dd7A4B850AA160590871629B00379e5

Scan

This contract is Honeypot!

Security Note:
This contract has function minting token!

BlockSAFU Token Scanner Score:

0

Score

semantic error causes contract minting

Token Information		Security Information	
Indicator	Value	Indicator	Value
Token Name	DES Token	Honeypot	× Yup, honeypot. Run the fuck away.
Token Symbol	DES	Buy Fees	8.9999986149007%
Total Supply	1,000,000,000	Sell Fees	8.997156818733%
Already Listed On Dex	Already Listed	Buy Gas	0 Gwei (0.000000 BNB / \$0.00)
Dex Listed	PancakeV2	Sell Gas	0 Gwei (0.000000 BNB / \$0.00)
Open Source	Open Source	Holder Count	10,242 Holders
Price	\$0.00655100		
Volume 24H	\$3764.77		
Liquidity	\$26,074 (86.93 BNB)		
Tx Count 24H	46		
Marketcap	\$6,766,442		

The owner can set fees up to 100% - this can cause a honeypot

Honeypot Safety

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

Rug Pull Safety

Indicator	Value
Hidden Owner	🛡️ Not detected
Creator Address	0xac45ecf9...b09 🔗
Creator Balance	0 DES
Creator Percent	0%
Owner Address	0xac45ecf9...b09 🔗
Owner Balance	0 DES
Owner Percent	0%
Lp Holder Count	96
Lp Total Supply	278,152,377
Mint	🚫 Detected

WEBSITE REVIEW



- Mobile Friendly
- Contains no code error
- SSL Secured

Web-Tech stack: jQuery, Wordpress

Domain .com (WildWest) - Tracked by whois

First Contentful Paint:	1.2s
Fully Loaded Time	4.4s
Performance	88%
Accessibility	94%
Best Practices	75%
SEO	75%

RUG-PULL REVIEW

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

- Locked Liquidity
 Unlocked
- TOP Holder Held 2.09%
- Not yet KYC team.
 (Will be updated after KYC)

HONEYPOT REVIEW

- Ability to sell.
- The owner can pause the contract (trade disable).
- The owner can set fees up to 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.