BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT

INDEPENDENT

Project: INDEPENDENT

Website: https://idpd-token.com/

BlockSAFU Score:

28

Contract Address:

0x9f6bc12147513e1b05802bb09b129Edcd3819C65

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 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. 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

BlockSAFU was commissioned by INDEPENDENT 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	**Independent TOKEN**
Token Symbol	**IDPD**
Token Decimal	9
Total Supply	54,000,000 **IDPD**
Contract Address	0x9f6bc12147513e1b05802bb09b129Edcd3819C65
Deployer Address	0x7C6A8709c63e079FBA97209b1a1CE6829757C902
Owner Address	0x7C6A8709c63e079FBA97209b1a1CE6829757C902
Tax Fees Buy	20%
Tax Fees Sell	20%
Gas Used for Buy	*will be updated after the DEX listing*
Gas Used for Sell	*will be updated after the DEX listing*
Contract Created	Jul-31-2022 01:22:21 AM +UTC
Initial Liquidity	*will be updated after the DEX listing*
Liquidity Status	Locked
Unlocked Date	*will be updated after the DEX listing*
Verified CA	Yes
Compiler	v0.8.9+commit.e5eed63a
Optimization	No with 200 runs
Sol License	MIT License
Top 5 Holders	*will be updated after the DEX listing*
Other	default evmVersion

TAX

BUY	20%	**SELL**	20%
buyback Fee	4%	buyback Fee	4%
charity Fee	2%	charity Fee	2%
liquidity Fee	6%	liquidity Fee	6%
marketing Fee	8%	marketing Fee	8%
project Fee	0%	project Fee	0%

OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 20% (owner can set fees up to 49%).
- Sell 20% (owner can set fees up to 49%).

Tx Amount

- Owner can set max tx amount.

Transfer Pausable

- Owner cannot pause.

Blacklist

- Owner can set blacklist.

Ownership

- Owner cannot take back ownership.
- Owner can set hidden owner

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 Holder

Rank	Address	Quantity	Percentage	Analytics
1	0xc3b721f5e5ddef3f8aff774bb65898484e8fedea	25,920,000	48.0000%	
2	0x7c6a8709c63e079fba97209b1a1ce6829757c902	19,851,113.558480222	36.7613%	
3	0xdd9e7f7f4142ecd620fc4dcd8795840b9308dbb5	2,700,000	5.0000%	
4	0x32c1cd7602d173ef0759d5b2bc1e897fdf3affd0	2,700,000	5.0000%	
5	0xeaa1b319fee12d442ec68ae9a9b0d2589034 0faf	2,700,000	5.0000%	
6	0xe9f5c30892693d856bdfc6ee33b706f4d5586071	20,449.252569779	0.0379%	
7	0x947c1ac9cea8b638d51dcf90c85bbf2fa02afdde	12,023.748336665	0.0223%	
8	0xc4f55844457c55abf07fd8609080c493e9381a3e	11,866.932332274	0.0220%	
9	0x1bf4d0bfebfacd7fff78488ddb6f7e2d4c8eb8b4	10,265.508287434	0.0190%	
10	0x77bbdfb60dfd093cf42fcd19938ebb8d53f77bf8	9,252.14425906	0.0171%	

Team Review

The team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 20 people in their telegram group (count in audit date).

Official Website And Social Media

Website: https://idpd-token.com/

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

Twitter: https://twitter.com/INDEPENDENTTOK

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

1 medium-risk code issues found

Should be fixed, could bring problems.

1. Owner can set fees up to 49%

```
function setFees(uint256 _liquidityFee, uint256 _charityFee, uint256
_marketingFee, uint256 _projectFee, uint256 _buyBackFee, uint256
_feeDenominator) external authorized {
        liquidityFee = _liquidityFee;
        charityFee = _charityFee;
        marketingFee = _marketingFee;
        projectFee = _projectFee;
        buyBackFee = _buyBackFee;
        totalFee =
_liquidityFee.add(_charityFee).add(_marketingFee).add(_projectFee).add(_buyBackF
ee);
        feeDenominator = _feeDenominator;
        require(totalFee < feeDenominator/2, "Fees cannot be more than 50%");
    }
```

High-Risk

3 high-risk code issues found

Must be fixed, and will bring problem.

1. Owner can set blacklist

```
function manage_blacklist(address[] calldata addresses, bool status) external
onlyOwner {
        require(addresses.length < 501,"GAS Error: max limit is 500 addresses");
        for (uint256 i; i < addresses.length; ++i) {
            isBlacklisted[addresses[i]] = status;
        }
    }
```

2. Owner can set max tx amount

```
function setMaxTxAbsolute(uint256 amount) external authorized {
        _maxTxAmount = amount;
    }
```

3. Owner can set hidden owner

```
function authorize(address adr) public onlyOwner {
        authorizations[adr] = true;
    }
```

Critical-Risk

0 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, 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. Independent Contract

```
contract Independent is IBEP20, Auth {
    using SafeMath for uint256;

    address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
    address DEAD = 0x000000000000000000000000000000000000dEaD;
    address ZERO = 0x0000000000000000000000000000000000000000;

    string constant _name = "Independent TOKEN";
    string constant _symbol = "IDPD";
    uint8 constant _decimals = 9;

    uint256 _totalSupply =  54 * 10**6 * 10**_decimals;

    uint256 public _maxTxAmount = _totalSupply / 50;
    uint256 public _maxWalletToken = _totalSupply / 100;

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

    bool public blacklistMode = true;
    mapping (address => bool) public isBlacklisted;

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

    uint256 public liquidityFee     = 6;
    uint256 public marketingFee     = 8;
    uint256 public projectFee       = 0;
    uint256 public charityFee       = 2;
    uint256 public buyBackFee       = 4;
    uint256 public totalFee         = charityFee + marketingFee +
liquidityFee + projectFee + buyBackFee;
    uint256 public feeDenominator   = 100;

    uint256 public sellMultiplier  = 100;
    uint256 public buyMultiplier   = 0;
    uint256 public transferMultiplier  = 50;

    address public autoLiquidityReceiver;
    address public marketingFeeReceiver;
```

```
    address public projectFeeReceiver;
    address public charityFeeReceiver;
    address public buyBackFeeReceiver;

    uint256 targetLiquidity = 30;
    uint256 targetLiquidityDenominator = 100;

    IDEXRouter public router;
    address public pair;

    bool public tradingOpen = false;

    // Cooldown & timer functionality
    bool public buyCooldownEnabled = true;
    uint8 public cooldownTimerInterval = 45;
    mapping (address => uint) private cooldownTimer;

    bool public swapEnabled = true;
    uint256 public swapThreshold = _totalSupply * 10 / 10000;
    bool inSwap;
    modifier swapping() { inSwap = true; _; inSwap = false; }

    constructor () Auth(msg.sender) {
        router =
IDEXRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
        pair = IDEXFactory(router.factory()).createPair(WBNB,
address(this));

        _allowances[address(this)][address(router)] =
type(uint256).max;

        isFeeExempt[msg.sender] = true;
        isTxLimitExempt[msg.sender] = true;

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

        autoLiquidityReceiver = msg.sender;
        marketingFeeReceiver = msg.sender;
        projectFeeReceiver = msg.sender;
```

```
        charityFeeReceiver = msg.sender;
        buyBackFeeReceiver = msg.sender;

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

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

        if(blacklistMode){
            require(!isBlacklisted[sender] &&
!isBlacklisted[recipient],"Blacklisted");
        }

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

        // cooldown timer, so a bot doesnt do quick trades! 45s
gap between 2 trades - add buy @lebarhamien
        if (sender == pair &&
            buyCooldownEnabled &&
            !isTimelockExempt[recipient]) {
            require(cooldownTimer[recipient] <
block.timestamp,"Please wait for cooldown between buys");
            cooldownTimer[recipient] = block.timestamp +
cooldownTimerInterval;
        }
```

```
        // Checks max transaction limit
        checkTxLimit(sender, amount);

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

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

        uint256 amountReceived = (!shouldTakeFee(sender) ||
!shouldTakeFee(recipient)) ? amount : takeFee(sender, amount,
recipient);
        _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 takeFee(address sender, uint256 amount, address
recipient) internal returns (uint256) {
```

```
        uint256 multiplier = transferMultiplier;
        if(recipient == pair){
            multiplier = sellMultiplier;
        } else if(sender == pair){
            multiplier = buyMultiplier;
        }

        uint256 feeAmount =
amount.mul(totalFee).mul(multiplier).div(feeDenominator * 100);

        uint256 buyBackTokens =
feeAmount.mul(buyBackFee).div(totalFee);
        uint256 contractTokens = feeAmount.sub(buyBackTokens);

        _balances[address(this)] =
_balances[address(this)].add(contractTokens);
        _balances[buyBackFeeReceiver] =
_balances[buyBackFeeReceiver].add(buyBackTokens);
        emit Transfer(sender, address(this), contractTokens);

        if(buyBackTokens > 0){
            emit Transfer(sender, buyBackFeeReceiver,
buyBackTokens);
        }

        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 amountBNBMarketing =
amountBNB.mul(marketingFee).div(totalBNBFee);
        uint256 amountBNBcharityFee =
amountBNB.mul(charityFee).div(totalBNBFee);
        uint256 amountBNBproject =
amountBNB.mul(projectFee).div(totalBNBFee);

        (bool tmpSuccess,) =
payable(marketingFeeReceiver).call{value: amountBNBMarketing, gas:
30000}("");
        (tmpSuccess,) = payable(projectFeeReceiver).call{value:
amountBNBproject, gas: 30000}("");
        (tmpSuccess,) = payable(charityFeeReceiver).call{value:
amountBNBcharityFee, gas: 30000}("");

        tmpSuccess = false;

        if(amountToLiquify > 0){
```

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

    // Public function
    function setMaxWalletPercent_base1000(uint256
maxWallPercent_base1000) external onlyOwner() {
        _maxWalletToken = (_totalSupply * maxWallPercent_base1000
) / 1000;
    }
    function setMaxTxPercent_base1000(uint256
maxTXPercentage_base1000) external onlyOwner() {
        _maxTxAmount = (_totalSupply * maxTXPercentage_base1000 )
/ 1000;
    }

    function setMaxTxAbsolute(uint256 amount) external authorized
{
        _maxTxAmount = amount;
    }

    function clearStuckBalance(uint256 amountPercentage) external
authorized {
        uint256 amountBNB = address(this).balance;
        payable(msg.sender).transfer(amountBNB * amountPercentage
/ 100);
    }

    // enable cooldown between trades buy @lebarhamien
    function cooldownEnabled(bool _status, uint8 _interval) public
onlyOwner {
        buyCooldownEnabled = _status;
        cooldownTimerInterval = _interval;
    }
```

```
    function setMultipliers(uint256 _buy, uint256 _sell, uint256
_trans) external onlyOwner{
        sellMultiplier = _sell;
        buyMultiplier = _buy;
        transferMultiplier = _trans;
    }

    function tradingStatus(bool _status) external onlyOwner {
        tradingOpen = _status;
    }

    function manage_blacklist_status(bool _status) external
onlyOwner {
        blacklistMode = _status;
    }

    function manage_blacklist(address[] calldata addresses, bool
status) external onlyOwner {
        require(addresses.length < 501,"GAS Error: max limit is
500 addresses");
        for (uint256 i; i < addresses.length; ++i) {
            isBlacklisted[addresses[i]] = status;
        }
    }

    function manage_FeeExempt(address[] calldata addresses, bool
status) external onlyOwner {
        require(addresses.length < 501,"GAS Error: max limit is
500 addresses");
        for (uint256 i; i < addresses.length; ++i) {
            isFeeExempt[addresses[i]] = status;
        }
    }

    function manage_TxLimitExempt(address[] calldata addresses,
bool status) external onlyOwner {
        require(addresses.length < 501,"GAS Error: max limit is
500 addresses");
        for (uint256 i; i < addresses.length; ++i) {
            isTxLimitExempt[addresses[i]] = status;
        }
```

```
    }

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

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

    function setIsTimelockExempt(address holder, bool exempt)
external authorized {
        isTimelockExempt[holder] = exempt;
    }

    function setFees(uint256 _liquidityFee, uint256 _charityFee,
uint256 _marketingFee, uint256 _projectFee, uint256 _buyBackFee,
uint256 _feeDenominator) external authorized {
        liquidityFee = _liquidityFee;
        charityFee = _charityFee;
        marketingFee = _marketingFee;
        projectFee = _projectFee;
        buyBackFee = _buyBackFee;
        totalFee =
_liquidityFee.add(_charityFee).add(_marketingFee).add(_projectFee)
.add(_buyBackFee);
        feeDenominator = _feeDenominator;
        require(totalFee < feeDenominator/2, "Fees cannot be more
than 50%");
    }

    function setFeeReceivers(address _autoLiquidityReceiver,
address _marketingFeeReceiver, address _projectFeeReceiver,
address _buyBackFeeReceiver, address _charityFeeReceiver) external
authorized {
        autoLiquidityReceiver = _autoLiquidityReceiver;
        marketingFeeReceiver = _marketingFeeReceiver;
        projectFeeReceiver = _projectFeeReceiver;
        buyBackFeeReceiver = _buyBackFeeReceiver;
        charityFeeReceiver = _charityFeeReceiver;
```

```
    }

    function setSwapBackSettings(bool _enabled, uint256 _amount)
external authorized {
        swapEnabled = _enabled;
        swapThreshold = _amount;
    }

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

    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 multiTransfer(address from, address[] calldata
addresses, uint256[] calldata tokens) external onlyOwner {

        require(addresses.length < 501,"GAS Error: max airdrop
limit is 500 addresses");
        require(addresses.length == tokens.length,"Mismatch
between Address and token count");

        uint256 SCCC = 0;

        for(uint i=0; i < addresses.length; i++){
```

```
            SCCC = SCCC + tokens[i];
        }

        require(balanceOf(from) >= SCCC, "Not enough tokens in
wallet");

        for(uint i=0; i < addresses.length; i++){
            _basicTransfer(from,addresses[i],tokens[i]);
        }
    }

event AutoLiquify(uint256 amountBNB, uint256 amountTokens);

}
```

4. Tax Fee contract

```
function setFees(uint256 _liquidityFee, uint256 _charityFee,
uint256 _marketingFee, uint256 _projectFee, uint256 _buyBackFee,
uint256 _feeDenominator) external authorized {
        liquidityFee = _liquidityFee;
        charityFee = _charityFee;
        marketingFee = _marketingFee;
        projectFee = _projectFee;
        buyBackFee = _buyBackFee;
        totalFee =
_liquidityFee.add(_charityFee).add(_marketingFee).add(_projectFee)
.add(_buyBackFee);
        feeDenominator = _feeDenominator;
        require(totalFee < feeDenominator/2, "Fees cannot be more
than 50%");
    }
```

The owner can set fees over 25% (49%) - **Medium risk**

5. Blacklist

```
function manage_blacklist(address[] calldata addresses, bool
status) external onlyOwner {
        require(addresses.length < 501,"GAS Error: max limit is
500 addresses");
        for (uint256 i; i < addresses.length; ++i) {
            isBlacklisted[addresses[i]] = status;
        }
    }
```

The owner can set blacklist Address. - **warning**

6. maxTxAmount

```
function setMaxTxAbsolute(uint256 amount) external authorized {
        _maxTxAmount = amount;
    }
```

The Owner can set max tx amount < 0.1 from total supply **-warning**

7. Hidden Owner

```
function authorize(address adr) public onlyOwner {
        authorizations[adr] = true;
    }
```

The owner can set hidden owner **-warning**

READ CONTRACT (ONLY NEED TO KNOW)

1. _maxTxAmount
1080000000000000 uint256
(Shows Contract Max TX Amount)

2. _maxWalletToken
540000000000000 address
(Shows Contract Max Wallet Token)

3. _blacklistMode
true bool
(Shows blacklistMode)

4. buyBack Fee
4 uint256
(Function for read buyback fee)

5. buyBackFeeReceiver
0xd7aceef654a98e77000b0caf78aff01058b462ed address
(Function for read buyback fee receiver)

6. buyCooldownEnabled
True bool
(Function for cooldown buy)

WRITE CONTRACT

1. cooldownEnabled
_status bool
_interval uint8
(The form is filled with the status and time for enable cooldown trade)

2. manageBlacklist
addresses (address[])
status (bool)
(The form is filled with the address and status for manage blacklist address)

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

BlockSAFU TOKEN SCANNER

https://blocksafu.com/token-scanner

Welcome to **BlockSAFU** | Don't give a chance for scammers!

BlockSAFU Products Knowledge Company Token Earn Request Service

BlockSAFU Token Scanner

0x9f6bc12147513e1b05802bb09b129Edcd3819C65 Scan

There is no liquidity available for this contract.

BlockSAFU Token Scanner Score:

0
Score

Token Information

Indicator	Value
Token Name	Independent TOKEN
Token Symbol	IDPD
Total Supply	54,000,000
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	42 **Holders**

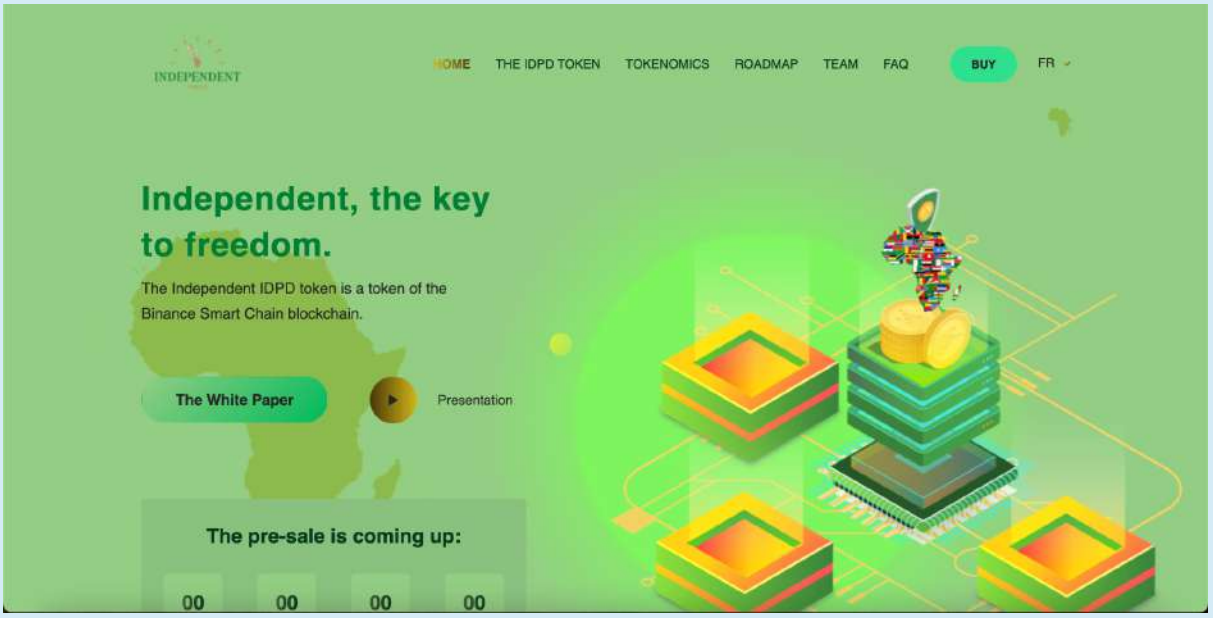
Honeypot Safety

Indicator	Value
Can Take Back Ownership	Not detected
Owner Change Balance	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	Detected
Creator Address	0x7c6a8709...902
Creator Balance	19,851,113.558 IDPD
Creator Percent	36.761300000000006%
Owner Address	0x7c6a8709...902
Owner Balance	19,851,113 IDPD
Owner Percent	36.761300000000006%
Lp Holder Count	0
Lp Total Supply	NaN
Mint	Not detected

WEBSITE REVIEW



- **Mobile Friendly**
- **Contains no code error**
- **SSL Secured (By R3 SSL)**

Web-Tech stack: jQuery, Bootstrap, Particle js

Domain .com (IONOS SE) - Tracked by whois

First Contentful Paint:	1.3s
Fully Loaded Time	2.6s
Performance	86%
Accessibility	94%
Best Practices	75%
SEO	90%

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 No KYC On Blocksafu

HONEYPOT REVIEW

- Ability to sell.
- The owner is not able to pause the contract.
- The owner can set fees up to 49%
- The owner can set blacklist
- The owner can set max tx amount

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.