



BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT



Project: Cheemspulse

Website: <https://twitter.com/CheemsPulse>



BlockSAFU Score:

95

Contract Address:

0x1f28a8A4D4deC0fE19451A30445C7047C8f1F860

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.

SMART CONTRACT REVIEW

Token Name	CheemsPulse
Token Symbol	CheemsPulse
Token Decimal	18
Total Supply	1,000,000,000 CHEMPULSE
Contract Address	0x1f28a8A4D4deC0fE19451A30445C7047C8f1F860
Deployer Address	0x2F6cd76A7B8113B1C16dc0F2cF9ffd55a604B012
Owner Address	0x2f6cd76a7b8113b1c16dc0f2cf9ffd55a604b012
Tax Fees Buy	5%
Tax Fees Sell	5%
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	Aug-14-2022 11:46:02 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	v0.8.4+commit.c7e474f2
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	5%	SELL	5%
Liquidity Fee	1%	Liquidity Fee	1%
Marketing Fee	3%	Marketing Fee	3%
Reward Fee	1%	Reward Fee	1%



OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy & Sell 5% (owner can't set fees over 25%).

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.

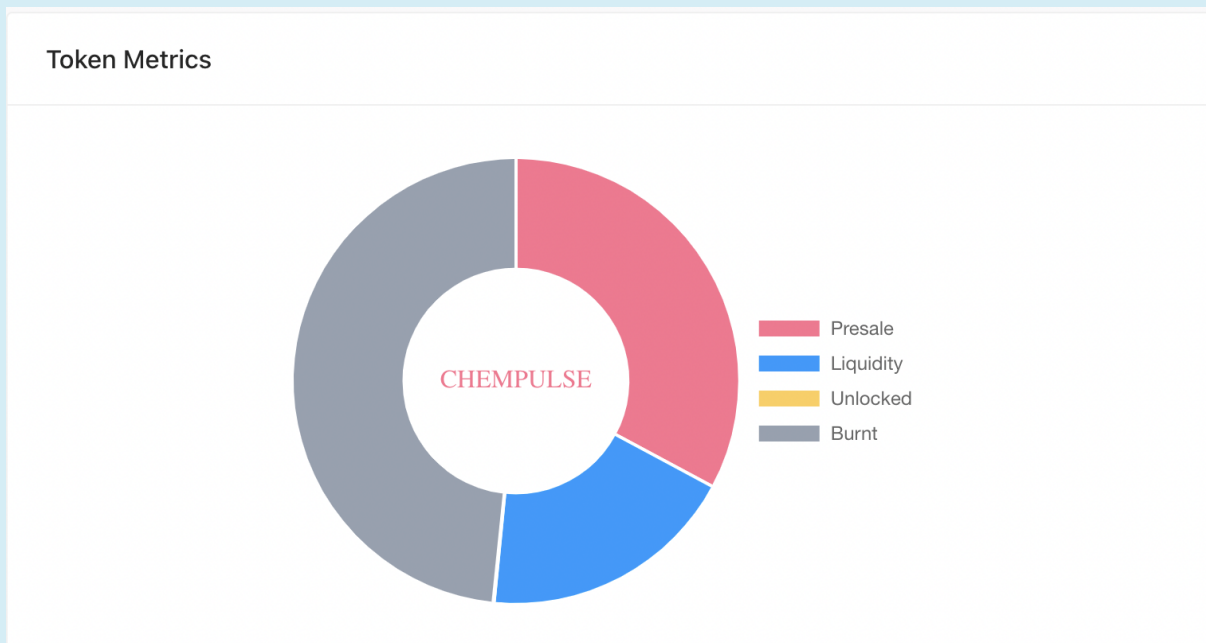
Anti Whale

- Owner cannot limit the number of wallet holdings.

Trading Cooldown

- Owner cannot set the selling time interval.
- 

Token Metrics



Team Review

The CheemsPulse team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 27 people in their telegram group (count in audit date).

Official Website And Social Media

Website: <https://app.cheemspulse.com/>

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

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

MANUAL CODE REVIEW

● Minor-risk

1 minor-risk code issue found

Could be fixed, and will not bring problems.

1. The return value of an external transfer/transferFrom return value is checked.
Recommendation: use SafeERC20, or ensure that the transfer/transferFrom return value is checked

```
function transferFrom(  
    address sender,  
    address recipient,  
    uint256 amount  
) external returns (bool);
```

● Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

● High-Risk

0 high-risk code issues found

Must be fixed, and will bring problem.

● Critical-Risk

0 critical-risk code issues found

Must be fixed, and will bring problem.

EXTRA NOTES SMART CONTRACT

1. IERC20

```
interface IERC20 {  
    /**  
     * @dev Returns the number of tokens in existence.  
     */  
    function totalSupply() external view returns (uint256);  
    ...  
    function balanceOf(address account) external view returns (uint256);  
    ...  
    function transfer(address recipient, uint256 amount) external returns (bool);  
    ...  
    function allowance(address owner, address spender) external view returns (uint256);  
    ...  
    function approve(address spender, uint256 amount) external returns (bool);  
    ...  
    function transferFrom(  
        address sender,  
        address recipient,  
        uint256 amount  
    ) external returns (bool);  
  
    /**  
     * @dev Emitted when `value` tokens are moved from one account (`from`) to  
     * another (`to`).  
     *  
     * Note that `value` may be zero.  
     */  
    event Transfer(address indexed from, address indexed to, uint256 value);  
    ...  
}
```

IERC20 Normal Base Template

2. SafeMath Contract

```
library SafeMath {
...
    function add(uint256 a, uint256 b) internal pure returns
(uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");
        return c;
    }
...
    function sub(uint256 a, uint256 b, string memory errorMessage)
internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }
    /**
     * @dev Returns the multiplication of two unsigned integers,
reverting on
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
     *
     * - Multiplication cannot overflow.
     */
...
    function mod(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        unchecked {
            require(b > 0, errorMessage);
            return a % b;
        }
    }
}
```

Standard Safemath contract

3. Cheempulse Contract

```
contract BABYTOKENDividendTracker is OwnableUpgradeable,
DividendPayingToken {
    using SafeMath for uint256;
    using SafeMathInt for int256;
    using IterableMapping for IterableMapping.Map;

    IterableMapping.Map private tokenHoldersMap;
    uint256 public lastProcessedIndex;

    mapping(address => bool) public excludedFromDividends;

    mapping(address => uint256) public lastClaimTimes;

    uint256 public claimWait;
    uint256 public minimumTokenBalanceForDividends;

    event ExcludeFromDividends(address indexed account);
    event ClaimWaitUpdated(uint256 indexed newValue, uint256
indexed oldValue);

    event Claim(
        address indexed account,
        uint256 amount,
        bool indexed automatic
    );

    function initialize(
        address rewardToken_,
        uint256 minimumTokenBalanceForDividends_
    ) external initializer {
        DividendPayingToken.__DividendPayingToken_init(
            rewardToken_,
            "DIVIDEND_TRACKER",
            "DIVIDEND_TRACKER"
        );
        claimWait = 3600;
        minimumTokenBalanceForDividends =
minimumTokenBalanceForDividends_;
    }

    function _transfer(
```

```

        address,
        address,
        uint256
    ) internal pure override {
        require(false, "Dividend_Tracker: No transfers allowed");
    }

    function withdrawDividend() public pure override {
        require(
            false,
            "Dividend_Tracker: withdrawDividend disabled. Use the
'claim' function on the main BABYTOKEN contract."
        );
    }

    function excludeFromDividends(address account) external
onlyOwner {
        require(!excludedFromDividends[account]);
        excludedFromDividends[account] = true;

        _setBalance(account, 0);
        tokenHoldersMap.remove(account);

        emit ExcludeFromDividends(account);
    }

    function isExcludedFromDividends(address account)
        public
        view
        returns (bool)
    {
        return excludedFromDividends[account];
    }

    function updateClaimWait(uint256 newClaimWait) external
onlyOwner {
        require(
            newClaimWait >= 3600 && newClaimWait <= 86400,
            "Dividend_Tracker: claimWait must be updated to
between 1 and 24 hours"
        );
        require(

```

```

        newClaimWait != claimWait,
        "Dividend_Tracker: Cannot update claimWait to same
value"
    );
    emit ClaimWaitUpdated(newClaimWait, claimWait);
    claimWait = newClaimWait;
}

function updateMinimumTokenBalanceForDividends(uint256 amount)
    external
    onlyOwner
{
    minimumTokenBalanceForDividends = amount;
}

function getLastProcessedIndex() external view returns
(uint256) {
    return lastProcessedIndex;
}

function getNumberOfTokenHolders() external view returns
(uint256) {
    return tokenHoldersMap.keys.length;
}

function getAccount(address _account)
    public
    view
    returns (
        address account,
        int256 index,
        int256 iterationsUntilProcessed,
        uint256 withdrawableDividends,
        uint256 totalDividends,
        uint256 lastClaimTime,
        uint256 nextClaimTime,
        uint256 secondsUntilAutoClaimAvailable
    )
{
    account = _account;

    index = tokenHoldersMap.getIndexOfKey(account);

```

```

iterationsUntilProcessed = -1;

if (index >= 0) {
    if (uint256(index) > lastProcessedIndex) {
        iterationsUntilProcessed = index.sub(
            int256(lastProcessedIndex)
        );
    } else {
        uint256 processesUntilEndOfArray =
tokenHoldersMap.keys.length >
        lastProcessedIndex
        ?
tokenHoldersMap.keys.length.sub(lastProcessedIndex)
        : 0;

        iterationsUntilProcessed = index.add(
            int256(processesUntilEndOfArray)
        );
    }
}

withdrawableDividends = withdrawableDividendOf(account);
totalDividends = accumulativeDividendOf(account);

lastClaimTime = lastClaimTimes[account];

nextClaimTime = lastClaimTime > 0 ?
lastClaimTime.add(claimWait) : 0;

secondsUntilAutoClaimAvailable = nextClaimTime >
block.timestamp
    ? nextClaimTime.sub(block.timestamp)
    : 0;
}

function getAccountAtIndex(uint256 index)
public
view
returns (
    address,
    int256,

```

```

        uint256,
        uint256,
        uint256,
        uint256,
        uint256,
        uint256
    )
}

if (index >= tokenHoldersMap.size()) {
    return (address(0), -1, -1, 0, 0, 0, 0, 0);
}

address account = tokenHoldersMap.getKeyAtIndex(index);

return getAccount(account);
}

function canAutoClaim(uint256 lastClaimTime) private view
returns (bool) {
    if (lastClaimTime > block.timestamp) {
        return false;
    }

    return block.timestamp.sub(lastClaimTime) >= claimWait;
}

function setBalance(address payable account, uint256
newBalance)
    external
    onlyOwner
{
    if (excludedFromDividends[account]) {
        return;
    }
    if (newBalance >= minimumTokenBalanceForDividends) {
        _setBalance(account, newBalance);
        tokenHoldersMap.set(account, newBalance);
    } else {
        _setBalance(account, 0);
        tokenHoldersMap.remove(account);
    }
    processAccount(account, true);
}

```

```

    }

    function process(uint256 gas)
        public
        returns (
            uint256,
            uint256,
            uint256
        )
    {
        uint256 numberOfTokenHolders =
tokenHoldersMap.keys.length;

        if (numberOfTokenHolders == 0) {
            return (0, 0, lastProcessedIndex);
        }

        uint256 _lastProcessedIndex = lastProcessedIndex;

        uint256 gasUsed = 0;

        uint256 gasLeft = gasleft();

        uint256 iterations = 0;
        uint256 claims = 0;

        while (gasUsed < gas && iterations < numberOfTokenHolders)
        {
            _lastProcessedIndex++;

            if (_lastProcessedIndex >=
tokenHoldersMap.keys.length) {
                _lastProcessedIndex = 0;
            }

            address account =
tokenHoldersMap.keys[_lastProcessedIndex];

            if (canAutoClaim(lastClaimTimes[account])) {
                if (processAccount(payable(account), true)) {
                    claims++;
                }
            }
        }
    }

```

```
    }

    iterations++;

    uint256 newGasLeft = gasleft();

    if (gasLeft > newGasLeft) {
        gasUsed = gasUsed.add(gasLeft.sub(newGasLeft));
    }

    gasLeft = newGasLeft;
}

lastProcessedIndex = _lastProcessedIndex;

return (iterations, claims, lastProcessedIndex);
}

function processAccount(address payable account, bool
automatic)
    public
    onlyOwner
    returns (bool)
{
    uint256 amount = _withdrawDividendOfUser(account);
    if (amount > 0) {
        lastClaimTimes[account] = block.timestamp;
        emit Claim(account, amount, automatic);
        return true;
    }
    return false;
}
}
```

4. Tax Fee contract

```
function setTokenRewardsFee(uint256 value) external onlyOwner {
    tokenRewardsFee = value;
    totalFees =
tokenRewardsFee.add(liquidityFee).add(marketingFee);
    require(totalFees <= 25, "Total fee is over 25%");
}

function setLiquiditFee(uint256 value) external onlyOwner {
    liquidityFee = value;
    totalFees =
tokenRewardsFee.add(liquidityFee).add(marketingFee);
    require(totalFees <= 25, "Total fee is over 25%");
}

function setMarketingFee(uint256 value) external onlyOwner {
    marketingFee = value;
    totalFees =
tokenRewardsFee.add(liquidityFee).add(marketingFee);
    require(totalFees <= 25, "Total fee is over 25%");
}
```

The owner can't set fees over 25%

5. PinkAntiBot

```
interface IPinkAntiBot {
    function setTokenOwner(address owner) external;

    function onPreTransferCheck(
        address from,
        address to,
        uint256 amount
    ) external;
}
...
function setEnableAntiBot(bool _enable) external onlyOwner {
    enableAntiBot = _enable;
}
```

The owner can set antibot to enable or not.

READ CONTRACT (ONLY NEED TO KNOW)

1. Version

1 uint256

(Shows Contract Versions)

2. owner

0x2f6cd76a7b8113b1c16dc0f2cf9ffd55a604b012 address

3. _marketingWalletAddress

0xda6fa0ac96d28c90d38f64b7268eafec623903d1 address

(Shows marketing wallet address)

4. enableAntiBot

True bool

(Function for read anti bot active or not)

5. liquidityFee

1 uint256

(Function for read liquidity fee)

6. marketingFee

3 uint256

(Function for read marketing fee)

7. name

CheemsPulse 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. transferOwnership

newOwner (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. setTokenRewardsFee

value (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 Token Scanner

Scan

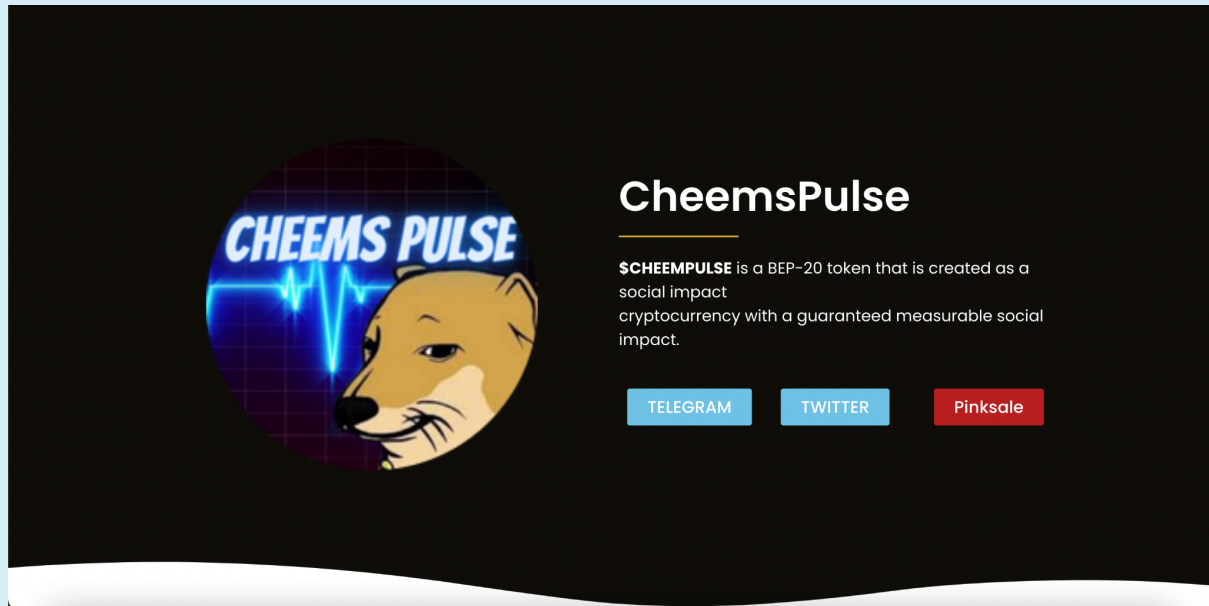
There is no liquidity available for this contract.

BlockSAFU Token Scanner Score:

80
Score

Token Information		Security Information	
Indicator	Value	Indicator	Value
Token Name	CheemsPulse	Honeypot	Liquidity Not Available
Token Symbol	CHEMPULSE	Buy Fees	0%
Total Supply	1,000,000,000	Sell Fees	0%
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	0 Holders
Price	\$NaN		
Volume 24H	\$NaN		
Liquidity	\$NaN (NaN BNB)		
Tx Count 24H			
Marketcap	\$NaN		

Honeypot Safety		Rug Pull Safety	
Indicator	Value	Indicator	Value
Can Take Back Ownership	✔ Not detected	Hidden Owner	✔ Not detected
Owner Change Balance	✔ Not detected	Creator Address	0x2f6cd76a...012 ↗
Blacklist	✔ Not detected	Creator Balance	1,000,000,000 CHEMPULSE
Modify Fees	❌ Detected	Creator Percent	100%
Proxy	✔ Not detected	Owner Address	0x2f6cd76a...012 ↗
Whitelisted	✔ Not detected	Owner Balance	1,000,000,000 CHEMPULSE
Anti Whale	❌ Detected	Owner Percent	100%
Trading Cooldown	✔ Not detected	Lp Holder Count	0
Transfer Pausable	✔ Not detected	Lp Total Supply	NaN
Cannot Sell All	✔ Not detected	Mint	✔ Not detected

WEBSITE REVIEW



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

Web-Tech stack: jQuery (Need update version), Bootstrap (Need Update version)

Domain .com (hostinger) - Tracked by whois

First Contentful Paint:	1.7ms
Fully Loaded Time	2.1s
Performance	86%
Accessibility	97%
Best Practices	92%
SEO	82%

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 By Blocksafu

HONEYPOT REVIEW

- Ability to sell.
- The owner is not able to pause the contract.
- The owner can't set fees over 25%
- PinkAntiBot

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.