Coinsult

Advanced Manual Smart Contract Audit

TRUSTSAFU

Project: TrustSAFU
Website: https://trustsafu.com

🟢 **Low-risk**	🟡 **Medium-risk**	🔴 **High-risk**
6 low-risk code issues found	1 medium-risk code issues found	0 high-risk code issues found

Contract address
0xb8CaAC95Cc8BBF898C3Ea97A174a7c93b022DD39

Disclaimer

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Tokenomics

Rank	Address	Quantity (Token)	Percentage
1	0x2e8e55c26340ce6cc9ee3ff9eee6553e8f4cb47a	325,000	100.0000%

Source code

Coinsult was commissioned by TrustSAFU to perform an audit based on the following smart contract:

https://bscscan.com/address/0xb8CaAC95Cc8BBF898C3Ea97A174a7c93b022DD39#code

Manual Code Review

● Low-risk

6 low-risk code issues found.
Could be fixed, will not bring problems.

- Usage of block.timestamp. block.timestamp can be manipulated by miners

  Recommendation: Avoid relying on block.timestamp.

```
        uint256 rebaseRate;
        uint256 deltaTimeFromInit = block.timestamp -
_initRebaseStartTime;
        uint256 deltaTime = block.timestamp - _lastRebasedTime;
        uint256 times = deltaTime.div(15 minutes);
        uint256 epoch = times.mul(15);
```

- Contract contains Reentrancy vulnerabilities: _transfer(address,address,uint256)

  Additional information: This combination increases risk of malicious intent. While it may be justified by some complex mechanics (e.g. rebase, reflections, buyback). More information: Slither

```
    function _transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) internal returns (bool) {

        require(!blacklist[sender] && !blacklist[recipient], "in_blacklist");

        if (inSwap) {
            return _basicTransfer(sender, recipient, amount);
        }
        if (shouldRebase()) {
           rebase();
        }

        if (shouldAddLiquidity()) {
            addLiquidity();
        }

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

        uint256 gonAmount = amount.mul(_gonsPerFragment);
        _gonBalances[sender] = _gonBalances[sender].sub(gonAmount);
        uint256 gonAmountReceived = shouldTakeFee(sender, recipient)
            ? takeFee(sender, recipient, gonAmount)
            : gonAmount;
        _gonBalances[recipient] = _gonBalances[recipient].add(
            gonAmountReceived
        );

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

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

        emit Transfer(
            sender,
            recipient,
            gonAmountReceived.div(_gonsPerFragment)
        );
        return true;
    }
```

- To many digits (Use: Ether suffix, Time suffix, or The scientific notation)

```
    uint256 distributorGas = 500000;
```

- The return value of an external transfer/transferFrom call is not checked

  Recommendation: Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

```
    function distributeDividend(address shareholder) internal {
        if(shares[shareholder].amount == 0){ return; }

        uint256 amount = getUnpaidEarnings(shareholder);
        if(amount > 0){
            totalDistributed = totalDistributed.add(amount);
            BUSD.transfer(shareholder, amount);
            shareholderClaims[shareholder] = block.timestamp;
            shares[shareholder].totalRealised =
shares[shareholder].totalRealised.add(amount);
            shares[shareholder].totalExcluded =
getCumulativeDividends(shares[shareholder].amount);
        }
    }
```

- Calls to a function sending Ether to an arbitrary address.

  `treasuryFee` can withdraw funds from every transaction

```
        (bool success, ) = payable(treasuryReceiver).call{
            value: amountETHToTreasuryAndSIF.mul(treasuryFee).div(
                treasuryFee.add(safuDividendFee)
            ),
            gas: 30000
        }("");
```

- No zero address validation

  Check that the new address is not the zero address

```
    function setLP(address _address) external onlyOwner {
        pairContract = IPancakeSwapPair(_address);
    }
```

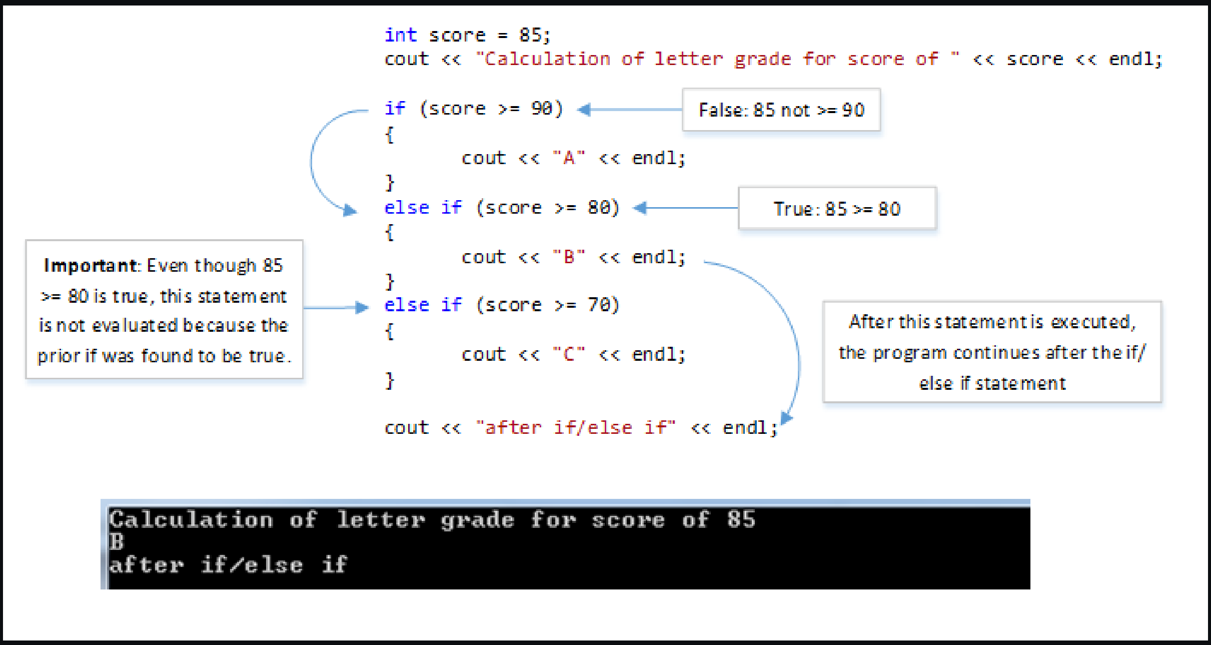
● Medium-risk

1 medium-risk code issues found.
Should be fixed, could bring problems.

- If statements might never be reached

  If the second statement is reached (>= 365 days), then the other two below will not be called upon.

```
    if (deltaTimeFromInit < (365 days)) {
        rebaseRate = 2355;
    } else if (deltaTimeFromInit >= (365 days)) {
        rebaseRate = 211;
    } else if (deltaTimeFromInit >= ((15 * 365 days) / 10)) {
        rebaseRate = 14;
    } else if (deltaTimeFromInit >= (7 * 365 days)) {
        rebaseRate = 2;
    }
```



```
int score = 85;
cout << "Calculation of letter grade for score of " << score << endl;

if (score >= 90)
{
        cout << "A" << endl;
}
else if (score >= 80)
{
        cout << "B" << endl;
}
else if (score >= 70)
{
        cout << "C" << endl;
}

cout << "after if/else if" << endl;
```

False: 85 not >= 90

True: 85 >= 80

Important: Even though 85 >= 80 is true, this statement is not evaluated because the prior if was found to be true.

After this statement is executed, the program continues after the if/ else if statement

```
Calculation of letter grade for score of 85
B
after if/else if
```

● High-risk

0 high-risk code issues found
Must be fixed, and will bring problems.

Extra notes by the team

🟢 Owner can only blacklist bot contracts, not normal addresses

```
    function setBotBlacklist(address _botAddress, bool _flag) external
onlyOwner {
        require(isContract(_botAddress), "only contract address, not
allowed externally owned account");
        blacklist[_botAddress] = _flag;
    }
```

🟢 Owner can not change the fees

```
    uint256 public liquidityFee = 20;
    uint256 public treasuryFee = 25;
    uint256 public safuDividendFee = 70;
    uint256 public sellFee = 20;
    uint256 public autofirePitFee = 25;
    uint256 public totalFee =
        liquidityFee.add(treasuryFee).add(safuDividendFee).add(
            autofirePitFee
        );
    uint256 public feeDenominator = 1000;
```

🟢 Owner can not exclude from fees

🟡 Owner can withdraw all tokens in the contract to Treusury wallet

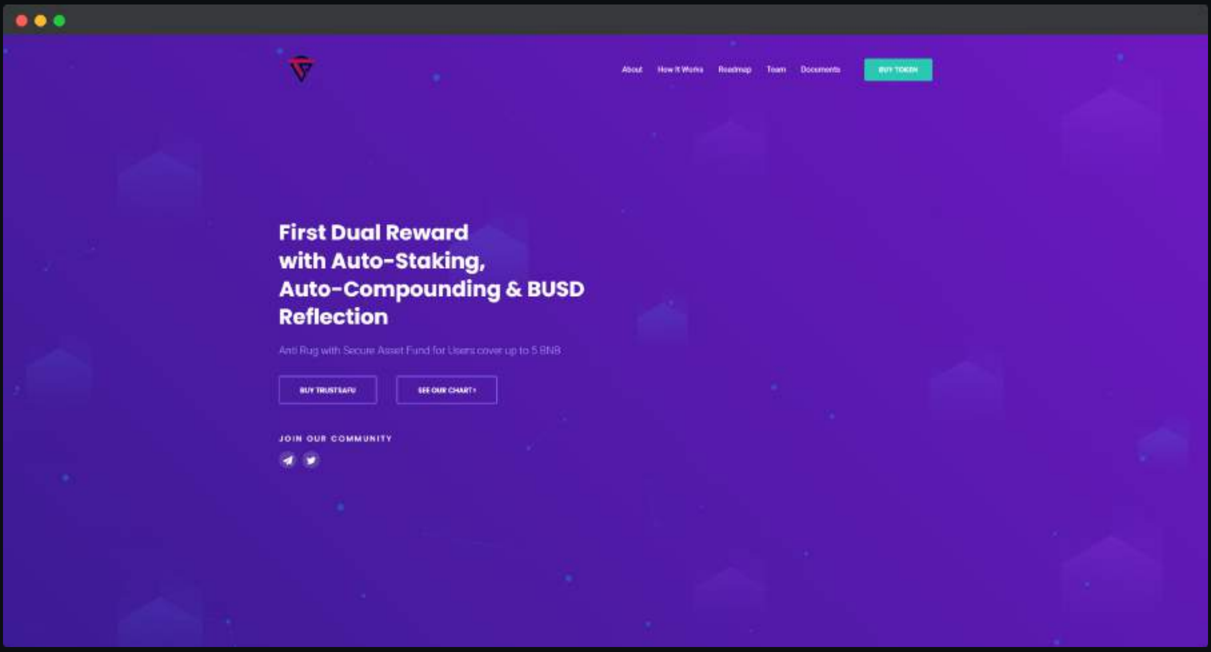
```
    function withdrawAllToTreasury() external swapping onlyOwner {

        uint256 amountToSwap = _gonBalances[address(this)].div(_gonsPerFragment);
        require( amountToSwap > 0,"There is no TrustSAFU token deposited in token
contract");
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = router.WETH();
        router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            amountToSwap,
            0,
            path,
            treasuryReceiver,
            block.timestamp
        );
    }
```

● Ownership is not renounced

Contract Snapshot

```
contract TrustSAFU is ERC20Detailed, Ownable {

    using SafeMath for uint256;
    using SafeMathInt for int256;

    event LogRebase(uint256 indexed epoch, uint256 totalSupply);

    IPancakeSwapPair public pairContract;
    mapping(address => bool) _isFeeExempt;

    modifier validRecipient(address to) {
        require(to != address(0x0));
        _;
    }

    uint256 public constant DECIMALS = 5;
    uint256 public constant MAX_UINT256 = ~uint256(0);
    uint8 public constant RATE_DECIMALS = 7;

    uint256 private constant INITIAL_FRAGMENTS_SUPPLY =
        325 * 10**3 * 10**DECIMALS;

    uint256 public liquidityFee = 20;
    uint256 public treasuryFee = 25;
    uint256 public safuDividendFee = 70;
    uint256 public sellFee = 20;
    uint256 public autofirePitFee = 25;
    uint256 public totalFee =
        liquidityFee.add(treasuryFee).add(safuDividendFee).add(
            autofirePitFee
        );
    uint256 public feeDenominator = 1000;

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

    address public autoLiquidityReceiver;
    address public treasuryReceiver;

    DividendDistributor distributor;
```

Website Review



First Dual Reward
with Auto-Staking,
Auto-Compounding & BUSD
Reflection

Anti Rug with Secure Asset Fund for Users cover up to 5 BNB

Coinsult checks the website completely manually and looks for visual, technical and textual errors. We also look at the security, speed and accessibility of the website. In short, a complete check to see if the website meets the current standard of the web development industry.

- 🟢 Mobile Friendly
- 🟢 Contains no jQuery errors
- 🟢 SSL Secured
- 🟢 No major spelling errors

Loading speed: 88%

Rug-pull Review

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

🟢 Locked Liquidity (no liquidity yet)

🟢 Large unlocked wallets

- Note: Tokens not distributed yet

🟡 No doxxed Team (KYC is planned in the future with Coinsult)

Honeypot Review

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

🟢 Ability to sell

🟢 Owner is not able to pause the contract

🟡 Router can be changed

Note: Please check the disclaimer above and note, the audit makes no statements or warranties on 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.