Coinsult

Advanced Manual Smart Contract Audit

Project: Crypto Sport Token
Website: https://cryptosporttoken.com

🟢 **Low-risk**
8 low-risk code issues found

🟡 **Medium-risk**
0 medium-risk code issues found

🔴 **High-risk**
0 high-risk code issues found

Contract address
0xC058f22d51E777ed6Ed6186a470B51490B413aa6

Disclaimer

Tokenomics

Total Supply: 1,000,000,000,000,000
Total Holders: 131
Top 10 holders:

Rank	Address	Quantity (Token)	Percentage
1	Null Address: 0x000...dEaD	500,000,000,000,000	50.0000%
2	0x12e02a9f229fd42736ebf7ae8f224a6f620a0a2f	163,800,000,000,000	16.3800%
3	0xc32f2bdc715ea265af151483b46032945ed6ce05	27,595,800,000,000	2.7596%
4	0xb0f60003dc826d2e2b2d4beaf3c4684f2ce7606b	25,000,000,000,000	2.5000%
5	0x746bcc0dc397d92562dad126020dfa852523fd32	25,000,000,000,000	2.5000%
6	0x54be5afa01e1f3c2f3a04949fe44fdf22922ad30	25,000,000,000,000	2.5000%
7	0xb11edf5539ee32ab4cd19e47a0a77e1ef6bc230a	25,000,000,000,000	2.5000%
8	0x25a912b477e9a1fbbdf78406ca002e3c0a0f12e6	25,000,000,000,000	2.5000%
9	0xcbbc8f09eed4c5fa18f974f11e643b74fc45aa29	15,000,000,000,000	1.5000%
10	0xda93885022d18d3469f0dea28ebbec61544eb316	15,000,000,000,000	1.5000%

The top 100 holders collectively own 99.94% (999,402,976,648,866.00 Tokens) of CRYPTOSPORTTOKEN

Note: This is a snapshot of when the audit was performed.

Source code

Coinsult was commissioned by Crypto Sport Token to perform an audit based on the following smart contract:

https://bscscan.com/address/0xC058f22d51E777ed6Ed6186a470B51490B413aa6#code

Manual Code Review

🟢 Low-risk

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

- Contract symbol uses a non-alphanumeric characters ($) this can cause problems during CEX listings

```solidity
constructor() public ERC20("CRYPTOSPORTTOKEN", "$CSPT") {
```

- Literals with many digits are difficult to read and review.
  Recommendation: Use Ether suffix, Time suffix, or The scientific notation

```solidity
uint256 public swapTokensAtAmount = 400000000 * (10**18);
_mint(owner(), 1000000000000000 * (10**18));
require(newValue >= 200000 && newValue <= 500000, "gasForProcessing
must be between 200,000 and 500,000");
minimumTokenBalanceForDividends = 200000000 * (10**18); //must hold
2000000+ tokens
```

- Same Solidity version declared multiple times (16x)
  Deploy with any of the following Solidity versions: 0.5.16 - 0.5.17, 0.6.11 - 0.6.12, 0.7.5 - 0.7.6 Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

```solidity
pragma solidity ^0.6.2;
```

- The return value of an external transfer/transferFrom call is not
  checked
  Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

```
function swapAndSendToFee(uint256 tokens, address wallet) private  {
    uint256 initialBUSDBalance =
IERC20(BUSD).balanceOf(address(this));
    swapTokensForBUSD(tokens);
    uint256 newBalance =
(IERC20(BUSD).balanceOf(address(this))).sub(initialBUSDBalance);
    IERC20(BUSD).transfer(wallet, newBalance);
  }
```

- Boolean constants can be used directly and do not need to be compare
  to true or false
  Remove the equality to the boolean constant.

```
function ensureMaxTxAmount(
    address from,
    address to,
    uint256 amount
) private view {
    if (
        _isExcludedFromMaxTx[from] == false && // default will be
false
        _isExcludedFromMaxTx[to] == false // default will be false
    ) {
            require(amount <= _maxTxAmount, "Transfer amount
exceeds the maxTxAmount.");
        }
    }
```

- Missing events for critical arithmetic parameters
  Emit an event for critical parameter changes.

```solidity
    function setMaxTxPercent(uint256 maxTxPercent) public onlyOwner() {
        _maxTxAmount = totalSupply().mul(maxTxPercent).div(10000);
    }


    function setExcludeFromMaxTx(address _address, bool value) public
onlyOwner {
        _isExcludedFromMaxTx[_address] = value;
    }


    function setBuyBackStatus(bool _enabled) external onlyOwner {
        buyBackEnabled = _enabled;
    }


    function setdividendLimitStatus(bool _enabled) external onlyOwner {
        dividendLimit = _enabled;
    }


     function setSwappingStatus(bool _enabled) external onlyOwner {
        swappingenabled = _enabled;
    }


    function changemaxdistributedividendLimit(uint256
_newvalue)external onlyOwner {
        maxdistributeDividends = _newvalue;
    }


    function changebuyBackthresholdLimit(uint256 _newvalue)external
onlyOwner {
        buyBackthresholdLimit = _newvalue;
    }


    function setBuybackAmount(uint256 _amount) external onlyOwner {
        buybackamount = _amount;
    }


    function changeselltax(uint256 _newtax)public onlyOwner {
        selltax = _newtax;
    }
```

- ## No zero address validation
  Check that the new address is not the zero address.

```solidity
    function updateUniswapV2Router(address newAddress) public onlyOwner
{
        require(newAddress != address(uniswapV2Router), "The router
already has that address");
        emit UpdateUniswapV2Router(newAddress,
address(uniswapV2Router));
        uniswapV2Router = IUniswapV2Router02(newAddress);
        address _uniswapV2Pair =
IUniswapV2Factory(uniswapV2Router.factory())
            .createPair(address(this), uniswapV2Router.WETH());
        uniswapV2Pair = _uniswapV2Pair;
    }

    function setMarketingWallet(address payable wallet) external
onlyOwner{
        _marketingWalletAddress = wallet;
    }

    function setDevelopementWallet(address payable wallet) external
onlyOwner{
        _DevelopementWalletAddress = wallet;
    }
```

- ## Contract contains Reentrancy vulnerabilities:
  ## _transfer(address,address,uint256)
  This contract seems to have a dividend tracker but has unchecked transfer
  vulnerabilities

```solidity
function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal override {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");
        require(!_isBlacklisted[from] && !_isBlacklisted[to], 'Blacklisted address');

        if(amount == 0) {
            super._transfer(from, to, 0);
            return;
        }

        ensureMaxTxAmount(from, to, amount);

        uint256 contractTokenBalance = balanceOf(address(this));
```

```solidity
        bool canSwap = contractTokenBalance >= swapTokensAtAmount;

        if( canSwap &&
            !swapping &&
            !automatedMarketMakerPairs[from] &&
            from != owner() &&
            to != owner() &&
            swappingenabled
        ) {
            swapping = true;
            uint256 marketingTokens = contractTokenBalance.mul(marketingFee).div(totalFees);
            swapAndSendToFee(marketingTokens,_marketingWalletAddress);
             uint256 DevelopementTokens = contractTokenBalance.mul(DevelopementFee).div(totalFees);
            swapAndSendToFee(DevelopementTokens,_DevelopementWalletAddress);
            uint256 swapTokens = contractTokenBalance.mul(liquidityFee).div(totalFees);
            swapAndLiquify(swapTokens);
            uint256 buybackTokens = contractTokenBalance.mul(buybackFee).div(totalFees);
            sellForBuyback(buybackTokens);
            uint256 sellTokens = balanceOf(address(this));
            swapAndSendDividends(sellTokens);
            swapping = false;
        }

        bool takeFee = !swapping;
        if(_isExcludedFromFees[from] || _isExcludedFromFees[to]) {
            takeFee = false;
        }

        if(takeFee) {
            uint256 fees = amount.mul(totalFees).div(100);
            if(automatedMarketMakerPairs[to] && from != address(this) && from != owner()){
                fees += amount.mul(selltax).div(100);


            if (buyBackEnabled && address(this).balance > buybackamount && amount > buyBackthresholdLimit ) {
                totalbuyback = totalbuyback.add(buybackamount);
                swapETHForTokens(deadWallet, buybackamount);
            }


            }
            amount = amount.sub(fees);
            super._transfer(from, address(this), fees);
        }

        super._transfer(from, to, amount);

        try dividendTracker.setBalance(payable(from), balanceOf(from)) {} catch {}
        try dividendTracker.setBalance(payable(to), balanceOf(to)) {} catch {}

        if(!swapping) {
            uint256 gas = gasForProcessing;

            try dividendTracker.process(gas) returns (uint256 iterations, uint256 claims, uint256
lastProcessedIndex) {
                emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas, tx.origin);
            }
            catch {

            }
        }
    }



        if( canSwap &&
```

🟡 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 problems.

Extra notes by the team

🟡 Owner can exclude from fees

🟡 Contract has relative high fees

🟡 Trade simulation shows high gas fees. This may indicate either inefficient code or overcomplicated logic.
    Contract is very long and uses much logic

🟡 The ownership of the contract isn't renounced

🟡 Contract has an internal minting function
    Never called to increase the total supply, but can be picked up by automatic checkers and therefore blacklisted on specific sites.

🔴 Owner can change the fees without a limit

🔴 Owner can blacklist accounts from selling

Contract Snapshot

```solidity
contract CRYPTOSPORTTOKEN is ERC20, Ownable {
    using SafeMath for uint256;

    IUniswapV2Router02 public uniswapV2Router;
    address public  uniswapV2Pair;
    bool private swapping;
    CRYPTOSPORTTOKENDividendTracker public dividendTracker;
    address public constant deadWallet =
0x000000000000000000000000000000000000dEaD;
    address public constant BUSD =
address(0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56);
    uint256 public swapTokensAtAmount = 400000000 * (10**18);
    bool public swappingenabled = true;
    mapping(address => bool) public _isBlacklisted;

    uint256 public BUSDRewardsFee = 8;
    uint256 public liquidityFee = 2;
    uint256 public marketingFee = 3;
    uint256 public DevelopementFee = 3;
    uint256 public buybackFee = 1;
    uint256 public selltax = 5;
    uint256 public totalFees =
BUSDRewardsFee.add(liquidityFee).add(marketingFee).add(buybackFee).add(
DevelopementFee);

    address public _marketingWalletAddress =
0xCbBc8F09eed4c5fA18f974F11E643B74Fc45AA29;
    address public _DevelopementWalletAddress =
0xDa93885022D18D3469F0dEa28ebbEc61544eB316;
    uint256 public gasForProcessing = 300000;

    // buyback parameters
    uint public buybackamount = 0.001 ether;
    uint public totalbuyback = 0;
    uint public accumulatedBuyback = 0;
    uint public buyBackthresholdLimit = 0;
    bool public buyBackEnabled = true;

    // dividends parameters
    bool public dividendLimit = false;
```
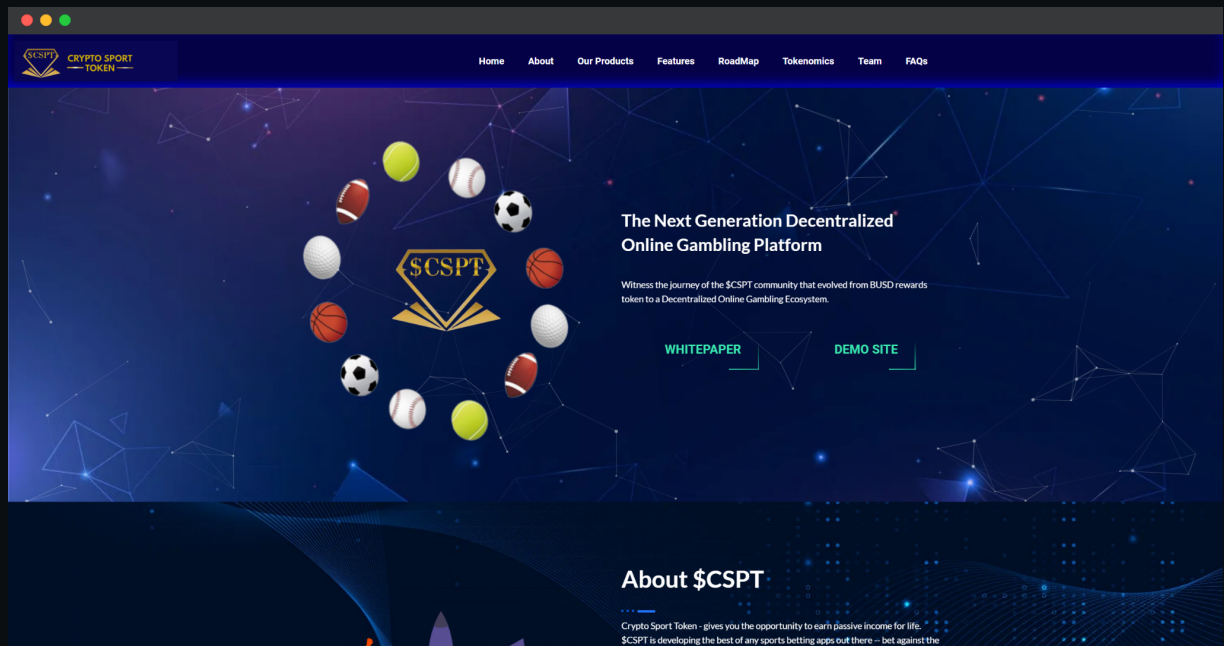
Website Review

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: 78%

Rug-pull Review

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

🟡 Locked Liquidity (no liquidity yet)

🟢 No large unlocked wallets

🟢 Doxxed Team (KYC)

Honeypot Review

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

🟡 Ability to sell
    Owner can set fees without a limit & can blacklist

🟢 Owner is not able to pause the contract
    Owner can set fees without a limit

🟢 Correct router hard coded in the contract
    0x10ED43C718714eb63d5aA57B78B54704E256024E

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.