Coinsult

Advanced Manual Smart Contract Audit

Project: Dogeball

Website: https://www.dogeball.games/

● **Low-Risk**

6 low-risk code issues found

● **Medium-Risk**

0 medium-risk code issues found

● **High-Risk**

0 high-risk code issues found

Contract Address

0xD30623b84164681D37E578014013be0DB006bbb1

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	0xe00a0a928c83030bf6ec324c495d8eb920312cae	35,000,000,000,000	35.0000%
2	Null Address: 0x000...dEaD	32,000,000,000,000	32.0000%
3	0x7148bc6212b02a60eb26f308be1abfe7e65b7e23	24,500,000,000,000	24.5000%
4	0xc4111596e7a8f55c5c76bf1d1c3c6b8cb1ce1b8c	5,000,000,000,000	5.0000%
5	0xb76204aac2bc34a956f3b41e7cdac9f572255c88	2,000,000,000,000	2.0000%

Source Code

Coinsult was comissioned by Dogeball to perform an audit based on the following smart contract:

https://bscscan.com/address/0xD30623b84164681D37E578014013be0DB006bbb1#code

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

6 low-risk code
issues found

🟡 Medium-Risk

0 medium-risk code
issues found

🔴 High-Risk

0 high-risk code
issues found

The detailed report continues on the next page...

● **Low-Risk:** Could be fixed, will not bring problems.

Contract contains Reentrancy vulnerabilities

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 _transfer(address sender, address recipient, uint256 amount) private returns (bool) {

    require(sender != address(0), "ERC20: transfer from the zero address");
    require(recipient != address(0), "ERC20: transfer to the zero address");
    require(amount &gt; 0, "Transfer amount must be greater than zero");
    if(inSwapAndLiquify)
    {
        return _basicTransfer(sender, recipient, amount);
    }
    else
    {
        if(!isTxLimitExempt[sender] &amp;&amp; !isTxLimitExempt[recipient]) {
            require(amount = _minimumTokensBeforeSwap;

        if (overMinimumTokenBalance &amp;&amp; !inSwapAndLiquify &amp;&amp; !isMarketPair[sender] &a
        {
            if(swapAndLiquifyByLimitOnly)
                contractTokenBalance = _minimumTokensBeforeSwap;
            swapAndLiquify(contractTokenBalance);
        }

         _balances[sender] = _balances[sender].sub(amount, "Insufficient Balance");
```

Recommendation

Apply the check-effects-interactions pattern.

Exploit scenario

```
function withdrawBalance(){
    // send userBalance[msg.sender] Ether to msg.sender
    // if mgs.sender is a contract, it will call its fallback function
    if( ! (msg.sender.call.value(userBalance[msg.sender])() ) ){
        throw;
    }
    userBalance[msg.sender] = 0;
}
```

Bob uses the re-entrancy bug to call withdrawBalance two times, and withdraw more than its initial deposit to the contract.

● **Low-Risk:** Could be fixed, will not bring problems.

Avoid relying on block.timestamp

block.timestamp can be manipulated by miners.

```
function getTime() public view returns (uint256) {
    return block.timestamp;
}
```

Recommendation

Do not use `block.timestamp`, now or `blockhash` as a source of randomness

Exploit scenario

```
contract Game {

    uint reward_determining_number;

    function guessing() external{
      reward_determining_number = uint256(block.blockhash(10000)) % 10;
    }
}
```

Eve is a miner. Eve calls `guessing` and re-orders the block containing the transaction. As a result, Eve wins the game.

● **Low-Risk:** Could be fixed, will not bring problems.

No zero address validation for some functions

Detect missing zero address validation.

```
function setMarketingWalletAddress(address newAddress) external onlyOwner() {
    marketingWalletAddress = payable(newAddress);
}
```

Recommendation

Check that the new address is not zero.

Exploit scenario

```
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```

Bob calls `updateOwner` without specifying the `newOwner`, soBob loses ownership of the contract.

● **Low-Risk:** Could be fixed, will not bring problems.

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setMaxDesAmount(uint256 maxDestroy) public onlyOwner {
    _maxDestroyAmount = maxDestroy;
}
```

Recommendation

Emit an event for critical parameter changes.

Exploit scenario

```
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```

`updateOwner()` has no event, so it is difficult to track off-chain changes in the buy price.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Conformance to Solidity naming conventions

Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
uint256 public _buyLiquidityFee = 2;
uint256 public _buyMarketingFee = 3;
uint256 public _buyTeamFee = 4;
uint256 public _buyDestroyFee = 0;
```

Recommendation

Follow the Solidity naming convention.

Rule exceptions

- Allow constant variable name/symbol/decimals to be lowercase (`ERC20`).
- Allow _ at the beginning of the `mixed_case` match for private variables and unused parameters.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Redundant Statements

Detect the usage of redundant statements that have no effect.

```
function _msgData() internal view virtual returns (bytes memory) {
    this;
    // silence state mutability warning without generating bytecode - see https://github.com/ethereur
    return msg.data;
}
```

Recommendation

Remove redundant statements if they congest code but offer no value.

Exploit scenario

```
contract RedundantStatementsContract {

    constructor() public {
        uint; // Elementary Type Name
        bool; // Elementary Type Name
        RedundantStatementsContract; // Identifier
    }

    function test() public returns (uint) {
        uint; // Elementary Type Name
        assert; // Identifier
        test; // Identifier
        return 777;
    }
}
```

Each commented line references types/identifiers, but performs no action with them, so no code will be generated for such statements and they can be removed.

Owner privileges

- 🟢 Owner cannot set fees higher than 25%
- 🟡 Owner can change max transaction amount
- 🟡 Owner can exclude from fees
- 🟡 Owner can pause the contract

Extra notes by the team

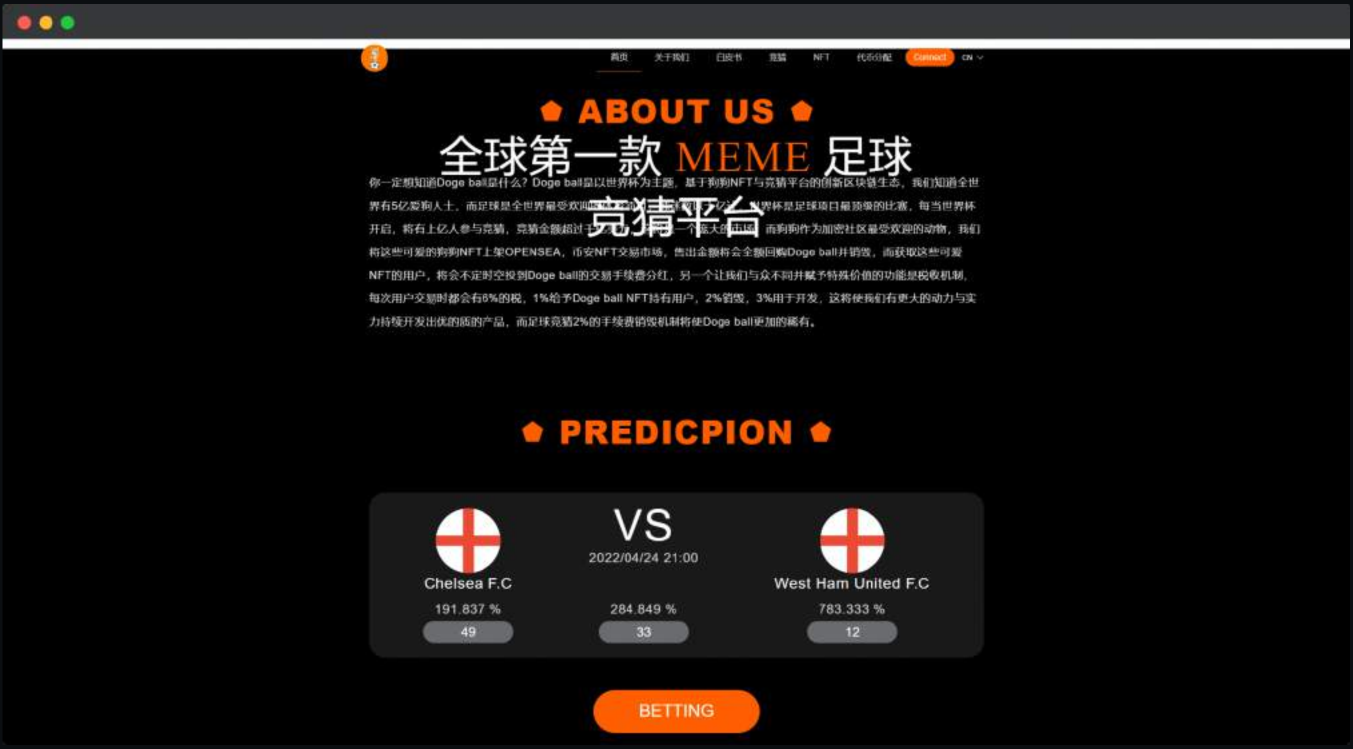
No notes

Contract Snapshot

```
contract TokenTool is Context, IERC20, Ownable {

using SafeMath for uint256;
using Address for address;

string private _name;
string private _symbol;
uint8 private _decimals;
address payable public marketingWalletAddress;
address payable public teamWalletAddress;
address public deadAddress = 0x000000000000000000000000000000000000dEaD;

mapping (address =&gt; uint256) _balances;
mapping (address =&gt; mapping (address =&gt; uint256)) private _allowances;

mapping (address =&gt; bool) public isExcludedFromFee;
mapping (address =&gt; bool) public isWalletLimitExempt;
mapping (address =&gt; bool) public isTxLimitExempt;
mapping (address =&gt; bool) public isMarketPair;

uint256 public _buyLiquidityFee = 2;
uint256 public _buyMarketingFee = 3;
uint256 public _buyTeamFee = 4;
uint256 public _buyDestroyFee = 0;
```

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.



ABOUT US

全球第一款 MEME 足球竞猜平台

你一定想知道Doge ball是什么？Doge ball是以世界杯为主题，基于狗狗NFT与竞猜平台的创新区块链生态，我们知道全世界有5亿爱狗人士，而足球是全世界最受欢迎[illegible]世界杯是足球项目最顶级的比赛，每当世界杯开启，将有上亿人参与竞猜，竞猜金额超过[illegible]一个庞大的市场，而狗狗作为加密社区最受欢迎的动物，我们将这些可爱的狗狗NFT上架OPENSEA，币安NFT交易市场，售出金额将会全额回购Doge ball并销毁，而获取这些可爱NFT的用户，将会不定时空投到Doge ball的交易手续费分红，另一个让我们与众不同并赋予特殊价值的功能是税收机制，每次用户交易时都会有6%的税，1%给予Doge ball NFT持有用户，2%销毁，3%用于开发，这将使我们有更大的动力与实力持续开发出优质的产品，而足球竞猜2%的手续费销毁机制将使Doge ball更加的稀有。

PREDICPION

Chelsea F.C	VS 2022/04/24 21:00	West Ham United F.C
191.837 %	284.849 %	783.333 %
49	33	12

- 🟢 Mobile Friendly
- 🟢 Does not contain jQuery errors
- 🟢 SSL Secured
- 🟢 No major spelling errors

Project Overview

● Not KYC verified by Coinsult